THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :
Fabrice GUET

le mercredi 13 décembre 2017

Titre :

Étude de l'application de la théorie des valeurs extrêmes pour l'estimation fiable et robuste du pire temps d'exécution probabiliste

---

École doctorale et discipline ou spécialité :

ED MITT : Sureté de logiciel et calcul de haute performance

Unité de recherche :

Équipe d'accueil ISAE-ONERA MOIS

Directeur(s) de Thèse :

M. Jérôme MORIO (directeur de thèse)
M. Luca SANTINELLI (co-directeur de thèse)

Jury :

M. Laurent GEORGE Enseignant-chercheur ESIEE - Président
Mme Liliana CUCU-GROSJEAN Chercheur INRIA Paris - Rapporteur
M. Mathieu JAN Ingénieur-chercheur CEA - Rapporteur
M. Jérôme MORIO Ingénieur de Recherche ONERA - Directeur de thèse
Mme Isabelle PUAUT Professeur Université de Rennes 1
M. Luca SANTINELLI Ingénieur de Recherche ONERA - Co-directeur de thèse

Remerciements

Je tiens à remercier Messieurs Luca Santinelli et Jérôme Morio, mes directeurs de thèse, pour m'avoir offert l'opportunité de réaliser cette thèse de doctorat. Merci de m'avoir fait confiance, guidé, et conseillé pendant ces trois années. Merci de m'avoir accueilli dans votre équipe dans laquelle j'ai pu m'épanouir scientifiquement et humainement grâce aux nombreux débats animés que nous avons pu avoir que ce soit à propos de la théorie des valeurs extrêmes, des systèmes temps réels ou de la présentation des travaux. Merci pour votre travail au quotidien, et vos encouragements qui m'ont permis de soutenir cette thèse.

Je souhaite également remercier les membres du jury pour avoir accepté d'évaluer mes travaux de thèse et pour leur intérêt concernant cette étude. Je tiens particulièrement à remercier Madame Liliana Cucu-Grosjean et Monsieur Mathieu Jan pour leur travail de rapporteur.

Je remercie toutes les personnes de l'ONERA avec qui j'ai pu travailler et échanger, Nicolas Gobillot, David Doose, Charles Lesire, Eric Noulard, Jean-Loup Bussenot et les personnes de l'ancien Département Traitement de l'Information et Modélisation que j'ai pu cotoyer. Je remercie les personnes que j'ai pu rencontrer lors de ma thèse à Porto ou dans des conférences. J'adresse ma gratitude envers mes anciens camarades thésards, anciens et actuels, avec qui j'ai partagé nombre de bons moments que ce soit autour d'un café ou d'une partie de cartes.

Je tiens à remercier très vivement mes parents pour votre éducation, et pour m'avoir soutenu tout au long de mes études et aujourd'hui encore.

Enfin, je remercie ma compagne qui me soutient au quotidien et me fait le plus beau des cadeaux.

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

ACS	Abstract Cache State
AST	Abstract Syntax Tree
BB	Basic Block
BDS	Brock Dechert Scheinkman
BCET	Best-Case Execution Time
BM	Block Maxima
CDF	Cumulative Distribution Function
CFG	Control Flow Graph
COTS	Commercial Off The Shelf
CUDA	Compute Unified Device Architecture
CE	Cross Entropy
DAG	Directed Acyclic Graph
DAL	Design Assurance Level
DM	Deadline Monotonic
DIAGXTRM	DIAGnostic de l'application de la théorie des valeurs eXTRêMes
EDF	Earliest Deadline First
EI	Extremal Index
ETP	Execution Time Profile
EVT	Extreme Value Theory
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GEV	Generalized Extreme Value
GPD	Generalized Pareto Distribution
GPU	Graphical Processor Unit
ICDF	Inverse Cumulative Distribution Function
IG	Interference Generator
IPET	Implicit Path Enumeration Technique
KPSS	Kwiatowski Phillips Schmidt Shin
LRU	Least Recently Used
LRR	Least Recently Replaced
MBDTA	Measurement Based Deterministic Timing Analysis
MBPTA	Measurement Based Probabilistic Timing Analysis
MDA	Maximum Domain of Attraction
MLE	Maximum Likelihood Estimator
NA	Not Available
NOC	Network On Chip
POT	Peaks Over Threshold

pWCET	probabilistic Worst-Case Execution Time
RM	Rate Monotonic
RR	Round Robin
SDTA	Static Deterministic Timing Analysis
SE	Standard Error
SM	Streaming Multiprocessor
SPTA	Static Probabilistic Timing Analysis
TSR	Task Spectral Representation
WCET	Worst-Case Execution Time

Première partie

Introduction

Dans les systèmes informatiques temps réel, les tâches logicielles doivent fournir un résultat correct mais également délivrer le résultat dans un temps imparti. Ainsi, il est primordial d'estimer de manière fiable le pire temps d'exécution des tâches logicielles pour assurer la prédictibilité du système. Le pire temps d'exécution de chaque tâche composant l'application temps réel en charge du contrôle de la physique d'un système critique est utilisé dans les analyses d'ordonnancement qui garantissent le respect des contraintes temporelles du système. Si l'estimation du pire temps d'exécution n'est pas fiable, les risques de dépassement d'échéance temporelle lors du déploiement du système temps réel sont plus importants avec des conséquences potentiellement catastrophiques sur le système critique. Ainsi, le pire temps d'exécution doit être sûr, il est potentiellement supérieur à tous les temps d'exécution possibles. Il est également de préférence précis, et donc pas trop surestimé par rapport à la valeur théorique inconnue.

Sachant qu'un système informatique temps réel est composé d'une architecture matérielle (processeur) exécutant une application temps réel, le temps d'exécution d'une tâche logicielle est directement fonction du matériel. Les architectures matérielles actuelles peuvent être prises sur étagère, ou être conçues maison. Dans le cas d'architectures conçues spécifiquement pour des besoins temps réel, le coût de conception est très élevé en temps et en argent. Le développement d'une telle architecture n'est donc pas forcément possible dans le cycle de développement d'un projet embarquant le système temps réel. Les architectures prises sur étagère bénéficient de composants d'optimisation qui participent à la réduction en moyenne du temps d'exécution des tâches logicielles. Cependant, du fait de la complexité des composants d'optimisation, il est trop coûteux d'avoir une connaissance exacte du pire temps d'exécution. Le secret industriel ainsi que la complexité des architectures actuelles et des tâches logicielles dégradent la confiance et la précision des estimations de pire temps d'exécution. La garantie de la sûreté et de la précision du pire temps d'exécution sont au prix de moyens coûteux et d'analyses complexes. Ce type d'architectures est majoritaire dans le commerce puisque l'industrie des processeurs est dirigée par les systèmes informatiques sans contrainte temporelle.

Depuis une quinzaine d'années maintenant, plusieurs travaux étudient l'estimation du pire temps d'exécution de tâches s'exécutant sur de telles architectures optimisées. Les travaux de [BE00] considèrent par exemple des suites de mesures du temps d'exécution sur des processeurs modernes. La théorie des valeurs extrêmes selon l'approche des maxima par bloc est ensuite appliquée à ces suites de mesures. Les auteurs modélisent par une loi de Gumbel le comportement extrême du temps d'exécution de la tâche mesurée. Ce modèle obtenu fournit une estimation du pire temps d'exécution avec une probabilité de dépassement associée. Dans cette étude, les auteurs montrent également que leur approche est applicable pour toutes les formes de processeur qui induisent de la variabilité dans le temps d'exécution d'une tâche logicielle. Cependant, aucune procédure d'évaluation de la fiabilité des estimations produites n'est considérée, sachant que l'application de la théorie des valeurs extrêmes peut renvoyer des estimations inexploitables car non fiables par rapport aux exigences de la théorie.

L'étude menée dans [HHM09] emploie également la méthode des maxima par bloc et la loi de Gumbel pour modéliser les temps d'exécution extrêmes d'une tâche. Les auteurs

introduisent un algorithme mettant en œuvre le test d'adéquation du χ^2 avec la loi de Gumbel tout en itérant sur la taille des blocs de maxima jusqu'à ce que le test soit validé. Ils ont enfin évalué leur approche sur un cas d'étude réel. Les auteurs obtiennent ainsi des estimations de pire temps d'exécution issues d'un modèle respectant le test du χ^2 et fiabilisant ainsi les estimations produites selon un critère d'adéquation du modèle aux mesures. Néanmoins, l'approche ne s'intéresse d'une part qu'à la seule loi de Gumbel pour modéliser les temps d'exécution extrêmes. D'autre part, l'évaluation de la fiabilité ne considère qu'un seul critère de validation d'appartenance à une loi. Or, il existe d'autres lois associées à la théorie des valeurs extrêmes. De plus, la théorie des valeurs extrêmes requiert davantage de critères d'application pour fiabiliser les estimations produites.

Les auteurs de [CGSH+12] considèrent des tests d'hypothèse concernant les critères d'application de la théorie des valeurs extrêmes toujours selon l'approche des maxima par bloc et en utilisant la loi de Gumbel. Les auteurs appliquent la théorie des valeurs extrêmes dans le cas de mesures indépendantes et identiquement distribuées (iid). Ils introduisent également un critère de convergence des mesures. Le modèle probabiliste estimé des temps d'exécution extrêmes est appelé pire temps d'exécution probabiliste. Les tests introduits permettent de valider l'indépendance des mesures ainsi que l'adéquation des mesures au modèle estimé à la manière du test du χ^2 dans [HHM09]. Cependant, le choix de la taille de bloc pour extraire les maxima reste arbitraire. De plus, dans le cas iid, il est nécessaire de connaître la distribution initiale théorique des temps d'exécution pour inférer avec confiance un modèle selon l'approche des blocs maxima et de la loi de Gumbel. Or, cette loi initiale des temps d'exécution n'est pas connue. Enfin, le modèle est toujours restreint à la loi de Gumbel qui peut fournir des modèles imprécis par rapport aux mesures.

Ces travaux sont à l'origine des méthodes probabilistes basées mesures d'estimation du pire temps d'exécution. Suite aux travaux de [CGSH+12] et particulièrement aux réflexions sur l'indépendance des mesures et la nécessité de valider les exigences de la théorie des valeurs extrêmes, les approches probabilistes basées mesures considèrent deux stratégies différentes. La première stratégie consiste à introduire davantage de tests d'hypothèse pour valider l'application de la théorie des valeurs extrêmes à une suite de mesures. Les travaux de [BSBT14, SMDJ14] emploient une batterie de tests d'hypothèses pour vérifier que les mesures sont indépendantes et identiquement distribuées. La seconde stratégie vise à randomiser les architectures matérielles pour garantir formellement l'indépendance des mesures et l'application de la théorie des valeurs extrêmes. Ces travaux sont développés dans le cadre du projet européen PROXIMA [DVA+14]. Dans ce projet, les architectures matérielles sont composées de mémoires caches avec des politiques de remplacement aléatoire pour briser la dépendance entre les exécutions de la tâche étudiée [WKL+13].

En conclusion, les approches développées pour estimer le pire temps d'exécution de tâches s'exécutant sur des processeurs modernes font usage de la théorie des valeurs extrêmes appliquée à des traces de mesure de temps d'exécution. Les techniques utilisent la méthode des maxima par bloc et la loi de Gumbel pour modéliser les temps d'exécution extrêmes de la tâche aussi rassemblés sous le pire temps d'exécution

probabiliste. Les auteurs ont mis en lumière la nécessité de fiabiliser les estimations de pire temps d'exécution probabiliste pour être exploitables lors du déploiement du système temps réel. Cependant, en ne considérant que la loi de Gumbel, les modèles se retrouvent biaisés, et souvent imprécis par rapport aux mesures. Les auteurs n'ont considéré que le cas de mesures indépendantes et identiquement distribuées pour vérifier la fiabilité des estimations. Néanmoins, il est impossible de garantir ces exigences en considérant des mesures de temps d'exécution sur des processeurs du commerce. Enfin, il n'existe aucune méthode de sélection des conditions de mesure du temps d'exécution d'une tâche pour produire des estimations robustes par rapport aux conditions d'exécution du système temps réel.

C'est dans ce contexte que l'étude de cette thèse prend place. Cette thèse consiste à évaluer la fiabilité des estimations basées mesures de pire temps d'exécution probabiliste pour des tâches s'exécutant sur des processeur pris sur étagère. Cette évaluation a pour but d'étudier la pertinence de l'application de la théorie des valeurs extrêmes à des suites de mesures du temps d'exécution de tâches logicielles s'exécutant sur des processeurs du commerce. De plus, les conditions de mesure pour lesquelles les suites de mesures sont réalisées impactent la robustesse et la représentatibilité des estimations par rapport au pire cas avéré. Cette thèse s'intéresse également à déterminer les conditions de mesure qui favorisent l'application de la théorie des valeurs extrêmes pour obtenir des estimations de pire temps d'exécution probabiliste robustes au pire cas.

L'analyse bibliographique (Chapitre 1) introduit dans un premier temps les éléments matériels et les analyses d'ordonnancement à la base des systèmes informatiques temps réel. Ces considérations permettent d'appréhender les composants matériels qui impactent l'estimation du pire temps d'exécution. Dans un deuxième temps (Chapitre 2), les différentes classes de méthodes existantes d'estimation du pire temps d'exécution sont présentées. L'étude des différentes méthodes suggère le besoin d'une méthode rapide, simple et fiable d'estimation du pire temps d'exécution notamment dans le cas de processeurs pris sur étagère. Nous décidons dans ce but d'explorer les approches probabilistes basées mesures qui semblent répondre à ce besoin.

Dans la suite (Chapitres 3 et 4), les formalismes et le matériel considérés dans cette thèse sont présentés. Ces chapitres introduisent les outils utiles aux contributions qui s'ensuivent.

La démarche a consisté dans un premier temps (Chapitre 5), à étudier les conditions de l'application de la théorie des valeurs extrêmes dans le cas de mesures stationnaires. Cette étude a pour but d'évaluer la fiabilité du pire temps d'exécution probabiliste estimé à partir d'une suite de mesures du temps d'exécution d'une tâche s'exécutant sur un processeur du commerce. Ainsi, un outil DIAGXTRM pour le DIAGnostique de l'application de la théorie des valeurs eXTRêMes à une suite de mesure est développé. L'outil intègre également des algorithmes pour robustifier l'estimation du pire temps d'exécution probabiliste relativement aux incertitudes de mesure.

L'outil DIAGXTRM a été appliqué (Chapitre 6) à un ensemble de traces issues de la

mesure du temps d'exécution de tâches et d'architectures matérielles différentes. L'application de l'outil à des traces provenant de cas d'étude différents a permis de mettre en évidence les capacités et particulièrement les limites de l'application de la théorie des valeurs extrêmes relativement aux types de tâches logicielles et d'architectures matérielles. De plus, l'étude met en lumière des conditions de mesure qui favorisent l'application de la théorie des valeurs extrêmes, mais également les conditions qui fournissent des estimations non fiables.

Dans la suite (Chapitre 7), les interférences matérielles entre des tâches s'exécutant en parallèle sont considérées pour favoriser la variabilité des temps d'exécution. Un algorithme qui détermine les tâches à exécuter en parallèle de la tâche étudiée est développé pour générer les temps d'exécution les plus longs. L'algorithme a pour de but de produire systématiquement un pire temps d'exécution probabiliste estimé représentatif du pire niveau d'interférences.

Enfin (Chapitre 8), une représentation spatiale du comportement temporel des tâches multichemin est proposée pour étudier l'impact des paramètres d'entrée des tâches étudiées sur l'application de la théorie des valeurs extrêmes. Le chapitre introduit des règles de mesure pour notamment garantir la stationnarité et la continuité relative des suites de mesures. De plus, un algorithme qui détermine les chemins de la tâche à exécuter est développé pour garantir la représentatibilité et la sûreté du pire temps d'exécution probabiliste estimé.

Deuxième partie

Bibliographie

Chapitre 1

Présentation des systèmes temps réel

Un système critique est un système dont la défaillance entraîne des conséquences catastrophiques comme la perte du système ou celle de vies humaines. La physique d'un système critique est contrôlée par un systeme informatique embarqué temps réel. Ainsi, un système temps réel est une application logicielle exécutée sur une architecture matérielle qui intéragit avec le système critique via des capteurs et des actionneurs. L'état du système critique variant en fonction du temps, le système temps réel doit répondre en temps contraint pour assurer le bon fonctionnement du système critique.

Dans ce chapitre les éléments qui constituent un système informatique temps réel sont présentés. En premier lieu, nous analysons de manière succinte le fonctionnement des architectures matérielles sur lesquelles s'exécute l'application logicielle (Section 1.1). En deuxième lieu, les méthodes mises en œuvre au niveau logiciel pour garantir le bon fonctionnement des systèmes critiques (Section 1.3) seront décrites.

1.1 Architectures matérielles

Le processeur est le cœur d'exécution des opérations des tâches logicielles constituant l'application temps réel. Une tâche logicielle étant une séquence d'instructions, le processeur lit et exécute ces instructions. Parmi ces instructions, on trouve les accès aux mémoires, les opérations arithmétiques et logiques, ou les sauts conditionnels. Il est possible d'écrire du code machine avec le langage assembleur même si les instructions sont sous forme binaire. Néanmoins, il est de plus en plus rare de développer en langage assembleur. Plusieurs langages plus haut niveau ont ainsi été développés comme le langage C ou encore le langage ADA. Les codes écrits dans ces langages haut niveau doivent être compilés pour être exécutés par le processeur.

Dans cette section, les différents types d'architecture existants sont présentés (Section 1.1.1). Nous abordons notamment les mécanismes des composants constituant la majorité des architectures actuelles et étudier leur impact sur l'exécution logicielle. Ces composants sont le pipeline d'instructions (Section 1.1.2) et les mémoires caches (Section 1.1.3).

1.1.1 Types d'architecture

Une architecture matérielle est composée d'un cœur d'exécution des instructions et de mémoires contenant les instructions à exécuter et les données sur lesquelles opérer. Pour accéder aux mémoires le cœur, utilise un bus d'accès mémoire. Le cœur, le bus et les mémoires caches sont assemblés sur le même support formant le processeur.

Les architectures diffèrent de la manière dont les instructions et données sont stockées dans la (les) mémoire(s). L'architecture de *Von Neumann* considère un même espace mémoire pour stocker données et instructions. Données et instructions partagent alors le même espace d'adressage. En conséquence, on accède aux données et aux instructions via le même bus ce qui peut dégrader les performances. L'architecture *Harvard* sépare physiquement données et instructions. Cette architecture considère donc des espaces d'adressage différents et des bus d'accès distincts. Cette dernière a été modifiée pour donner l'architecture *Harvard modifié* qui considère un même espace d'adressage pour les données et les instructions mais possède deux bus distincts pour y accéder.

Une architecture d'ordinateur peut aussi comporter plusieurs processeurs. On parle dans ce cas d'ordinateur multiprocesseur. Les processeurs peuvent avoir un accès symétrique à la mémoire, dans ce cas la mémoire est alors partagée entre tous les processeurs. A l'inverse, la mémoire peut être non uniforme. Dans ce cas, les temps d'accès à la mémoire dépendent de la distance entre la mémoire et le processeur qui souhaite y accéder.

Une architecture d'ordinateur peut comporter plusieurs cœurs par processeur. On parle alors de processeur multicœur, les cœurs étant tous sur le même support. Il existe aussi des processeurs manycœur, qui comportent en général plus de cœurs que les processeurs multicœur. La réelle différence est la connexion entre les cœurs par un réseau et non plus par un bus. Les accès au réseau sont explicitement déclarés dans l'implémentation du code.

Dans la suite, les différents composants qui permettent l'optimisation de l'exécution de tâches logicielles notamment en terme de temps seront abordés.

1.1.2 Pipeline d'instructions

L'exécution d'une instruction dans un cœur se fait au travers de cinq étapes dans un pipeline d'instructions :

— charger l'instruction dans le pipeline.
— décoder l'instruction.
— exécuter l'instruction.
— accéder à la mémoire.
— stocker le résultat dans la mémoire.

Chaque étape est réalisée dans un étage dédié du pipeline. En supposant que chaque étape prend un cycle d'horloge, il faut donc cinq cycles pour exécuter une instruction. Pour trois instructions à exécuter, quinze cycles sont alors nécessaires.

Néanmoins, cette exécution n'est pas optimale du point de vue du pipeline. A la

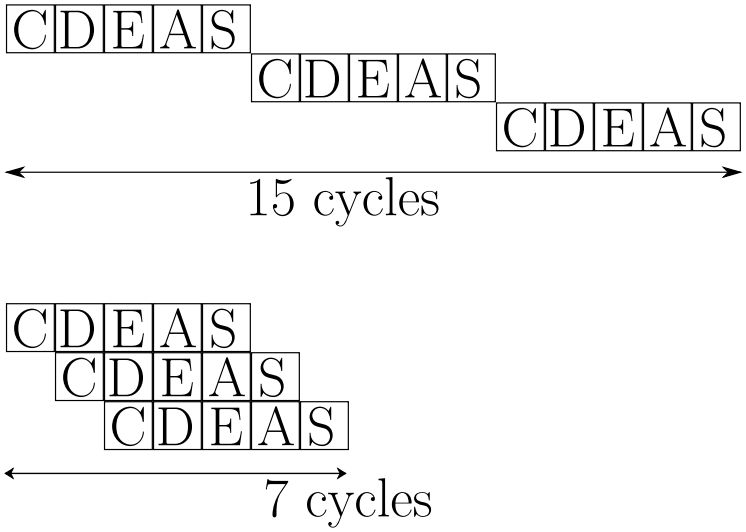


FIGURE 1.1 – Fonctionnement des étages sans et avec pipeline pour trois instructions. C : charger, D : décoder, E : exécuter, A : accéder, S : stocker.

manière d'une chaîne de montage, dès qu'un étage est vidé on peut passer à l'instruction suivante. Ainsi l'exécution des instructions dans le pipeline est parallélisée. Au lieu de quinze cycles pour exécuter trois instructions, sept cycles sont finalement pris. Le fonctionnement du pipeline est illustré à l'aide du schéma de la Figure 1.1.

Néanmoins, la parallélisation des exécutions des instructions peut entraîner des accès concurrents à des espaces mémoires. Ces accès concurrents dégradent les performances du pipeline.

Prédiction de branchement Au cours de l'exécution d'un programme logiciel, il est possible d'atteindre un branchement conditionnel. Selon la valeur de la variable du branchement, le programme peut exécuter une branche ou une autre. Tant que la valeur de la variable n'est pas connue il impossible de déterminer quelle branche sera suivie. Si la branche n'est pas connue, alors le pipeline ne peut charger les instructions suivantes.

Pour éviter l'arrêt du pipeline, une unité parie sur le branchement qui va être pris. Le pipeline va donc charger les instructions sur lesquelles il a parié. Si le pari échoue, alors il faudra vider le pipeline et charger les bonnes instructions. Pour effectuer le pari, un algorithme est implémenté dans l'unité du pipeline responsable. Ces choix peuvent se faire au moment de la compilation ou alors selon l'historique d'exécution.

Superscalarité Certains processeurs présentent des pipelines avec plusieurs exemplaires d'un étage. Si tous les étages d'un pipeline sont redondés n fois alors le processeur est dit superscalaire de degré n. Néanmoins la superscalarité augmente le nombre

de conflits possibles entre les étages. En moyenne, la superscalarité permet de diminuer le temps d'exécution des instructions d'une tâche.

Ordonnancement des instructions Pour réduire le nombre de conflits dû au parallélisme, il est possible de décider de l'ordre des instructions à exécuter dans le pipeline. Ce changement d'ordonnancement peut se faire au moment de la compilation ou alors directement par le pipeline. Ce type de pipeline est dit *out-of-order*.

1.1.3 Mémoires Caches

Les données et instructions nécessaires à l'exécution des instructions de la tâche logicielle sont stockées dans la mémoire principale. Cette mémoire principale possède une grande capacité mais est particulièrement lente. Par conséquent, un accès à la mémoire principale prend plusieurs cycles. Les mémoires caches sont des mémoires intermédiaires qui permettent d'accéder plus rapidement aux données, et instructions afin de réduire le temps d'exécution.

Hiérarchie mémoire Plusieurs mémoires caches sont organisées selon différents niveaux constituant une hiérarchie mémoire. La mémoire la plus proche du cœur constitue le premier niveau aussi noté L1. Il s'agit d'une mémoire de faible capacité mais très rapide. L'espace d'adressage du niveau L1 est inclus dans le niveau L2 et ainsi de suite jusqu'à la mémoire principale. Plus la mémoire est proche du cœur, plus sa capacité est faible mais son temps d'accès rapide.

Dans le commerce, on trouve généralement des processeurs avec trois niveaux de cache. De plus, données et instructions sont souvent séparées dans le premier niveau alors qu'elles sont unifiées dans les autres niveaux. Un exemple d'une telle hiérarchie mémoire est donnée dans le schéma de la Figure 1.2 dans le cas d'un processeur avec deux cœurs d'exécution qui contiennent chacun des registres mémoires, les mémoires les plus rapides mais aussi les plus coûteuse, et dont le premier niveau de cache sépare instructions et données. Les niveaux de cache sont reliés entre eux par des bus d'accès mémoire. La documentation de chaque processeur renseigne les temps d'accès ou latence des différents niveaux de cache.

Cohérence de cache Un aspect critique des hiérarchies mémoires et particulièrement dans le cas de processeurs multicœur est d'assurer la cohérence de cache. Pour au moins deux cœurs qui accèdent à la même donnée, alors cette donnée doit avoir le même état dans toutes les mémoires caches. Si trop de données sont partagées et accédées en écriture entre différents cœurs alors l'avantage d'une hiérarchie mémoire peut être perdu car il faut systématiquement écrire dans la mémoire principale.

Fonctionnement du cache Pour accéder rapidement aux données, il faut qu'elles soient au plus près du cœur. Lors d'un accès mémoire, le cache de premier niveau vérifie si la donnée y est présente, et sinon passe au niveau suivant, ainsi de suite jusqu'à la

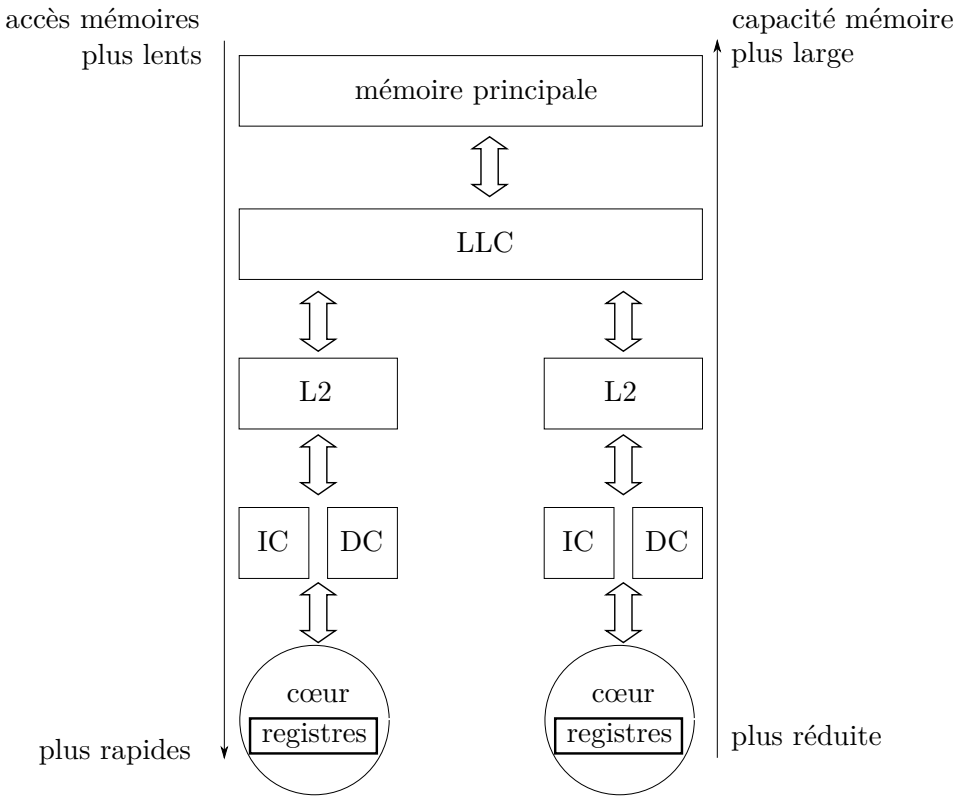


FIGURE 1.2 – Exemple de hiérarchie mémoire composée de trois niveaux de mémoires caches. LLC : Last Level Cache pour dernier niveau de cache, IC : Instruction Cache pour cache d'instructions, DC : Data Cache pour cache de données.

mémoire principale. Si la donnée n'est pas dans le cache, alors elle la copie depuis les mémoires de niveau supérieur. Chaque cache copie la donnée utile dans les caches de différents niveaux de manière inclusive selon les tailles de cache.

Si la donnée accédée est bien dans le cache, il s'agit d'un succès ou *cache hit*. Dans le cas contraire, il s'agit d'un défaut de cache ou *cache miss*. A la première exécution, aucune donnée n'est dans le cache ce qui produit des *cold misses*.

Pour rendre compte des performances des mémoires caches on s'intéresse à deux principes vérifiés par les mémoires caches :

-principe de localité spatiale : les données proches dans l'espace d'adressage d'une donnée accédée ont une probabilité plus grande d'être accédées par la suite.

-principe de localité temporelle : une donnée accédée ont une probabilité plus grande d'être de nouveau accédée dans la suite immédiate de l'exécution des instructions.

Le fonctionnement du cache est basé sur des *blocs* de cache. Le cache étant un ensemble de blocs et les blocs un ensemble d'octets dont la taille varie d'une architecture

à l'autre. Les blocs contiennent un bit de validité, un numéro de bloc et des données utiles ou mots. Si le cœur modifie la valeur d'une donnée du bloc, alors le bit de validité passe à 1 notifiant que la valeur est différente de celle contenue dans les autres mémoires. Le numéro du bloc permet de connaître la position du bloc dans les mémoires. Un bloc est la plus petite unité qui peut être échangée entre deux caches de niveaux différents. Un mot est la plus petite unité qui peut transiter entre le cache de premier niveau et le cœur.

Associativité du cache Les caches ont des capacités limitées et cette capacité est de plus en plus faible à mesure que les mémoires sont proches du cœur. Il faut donc faire des choix sur les blocs à conserver et à remplacer dans les mémoires caches. Ces choix dépendent de l'associativité du cache. Un cache divisé en ensembles de N blocs est d'associativité N. Les choix sur le placement et le remplacement des blocs dans le cache diffèrent suivant l'associativité.

-totalement associatif : si l'associativité est égale au nombre de blocs que contient le cache alors il est totalement associatif. Un bloc peut être placé n'importe où dans le cache.

-correspondance directe : si l'associativité est égale à un alors le cache est à correspondance directe. Chaque bloc a son propre emplacement déterminé par son numéro.

-associatif par ensemble : si l'associativité est égale à N différent des cas précédents alors le cache est dit associatif par ensemble. Un bloc peut être placé de manière libre dans un ensemble donné du cache déterminé par le numéro de bloc.

Politiques de remplacement Dans le cas d'un cache à correspondance directe, les blocs sont placés selon leur numéro. Cette méthode est rapide et simple à mettre en œuvre. Néanmoins, si un programme fait souvent usage de deux blocs situés au même emplacement alors les deux ne cesseront de se remplacer mutuellement et d'entraîner des *misses*.

Dans les autres cas, il faut décider quel bloc de l'ensemble va être remplacer dans le cas d'un *miss*. Deux politiques sont essentiellement mises en œuvre. La politique du bloc le plus ancien (Least Recently Used LRU) où l'élément accédé le moins récemment de l'ensemble est remplacé en priorité. La politique aléatoire consiste à tirer aléatoirement le bloc qui sera remplacé.

On résume le fonctionnement des mémoires caches grâce au schéma de la Figure 1.3. Cet exemple présente un niveau de mémoire avec une large capacité, et des caches mémoires dont la capacité est de quatre blocs mémoires et de différente associativité. Les blocs sont associés à des numéros dans la mémoire la plus large qui permettent d'ordonner les blocs. Dans le cas du cache totalement associatif (TA) les blocs sont rangés du plus récemment utilisé au plus ancien. Si un nouveau bloc est accédé alors le bloc en dernière position du cache est retiré. Dans le cas du cache à correspondance directe (CD) les

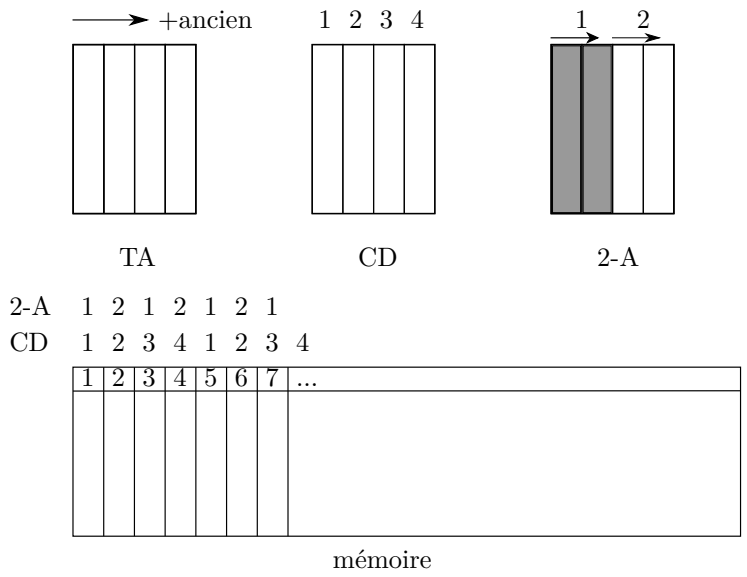


FIGURE 1.3 – Fonctionnement d'une mémoire cache dans les cas Totalement Associatif (TA), Correspondance Directe (CD) et Associatif de degré N (N-A) où N vaut 2 dans ce cas.

blocs sont rangés selon leur numéro dans la mémoire la plus large modulo la taille du cache. Le cas du cache associatif de degré 2 (2-A) fait usage des mécanismes des deux types de caches TA et CD mais pour deux caches TA de taille 2.

1.2 Prédictibilité des systèmes temps réel

1.2.1 Système critique

L'architecture matérielle est à l'interface entre le système critique et l'application temps réel. Les capteurs du système critique ainsi que ses actuateurs sont interfacés à l'architecture matérielle. Les données des capteurs sont lues par l'application temps réel et les tâches logicielles associées effectuent les calculs pour contrôler les actuateurs afin d'assurer le bon comportement du système critique. L'imbrication entre le matériel et le logiciel du système temps réel est le point clé pour garantir ce bon comportement.

Rappelons qu'un système temps réel n'est pas forcément un système rapide mais un système prédictible. Si une donnée est lue au mauvais moment, ou un ordre est envoyé au mauvais moment, alors la physique du système critique est modifiée incorrectement ce qui entraîne une défaillance et ainsi la perte du système critique. Les architectures matérielles ont connu nombre d'optimisations pour réduire le temps d'exécution des

tâches logicielles, mais ces optimisations sont elles compatibles avec le niveau de criticité exigé ?

1.2.2 Niveaux de criticité

Pour assurer le bon comportement du système critique, le système temps réel embarqué doit répondre en temps contraint. Plus particulièrement, chaque tâche logicielle composant l'application temps réel doit être exécutée avant une échéance temporelle. La prédictibilité du système temps réel dépend du respect de ces échéances.

La criticité du système à réguler impose un niveau d'exigence sur le degré de prédictibilité du système temps réel. Le niveau d'exigence requiert des niveaux de tolérance sur les dépassements d'échéance des tâches de l'application temps réel. Il existe deux niveaux de criticité des systèmes temps réel :

-temps réel dur : les échéances doivent obligatoirement être respectées. Le non respect provoque un évènement catastrophique pouvant correspondre à la perte du système ou des pertes humaines.

-temps réel souple : les dépassements d'échéance sont tolérés. Le dépassement des échéances entraîne une dégradation de la qualité de service du système.

Il existe également des systèmes temps réel dits de criticié mixte. Dans ce cas, certaines tâches de l'application temps réel ont des contraintes dures et d'autres souples.

1.3 Ordonnancement des applications temps réel

Considérons le cas d'une application temps réel τ comme un ensemble de tâches logicielles $\tau = \{\tau_1, \ldots, \tau_n\}$ préemptibles (stoppées par une tâche de plus haute priorité), exécutées périodiquement et indépendantes i.e. qu'il n'y a pas de synchronisation entre elles. Chacune des tâches composant l'application est soumise à des contraintes temporelles. Pour assurer la prédictibilité du système temps réel, on réalise une analyse d'ordonnancement des tâches de l'application. L'algorithme d'ordonnancement prend en compte des paramètres spécifiques pour chaque tâche afin de satisfaire leurs contraintes temporelles.

Premièrement, nous présentons le modèle de tâche couramment utilisé dans les analyses d'ordonnancement (Section 1.3.1) et la manière dont il est représenté graphiquement (Section 1.3.2). Deuxièmement, nous abordons les algorithmes d'ordonnancement principaux (Section 1.3.3) et les analyses de faisabilité associées (Section 1.3.4). Ensuite, nous décrivons quelques protocoles d'accès aux ressources partagées (Section 1.3.5). Enfin, le cas de l'ordonnancement multiprocesseur (Section 1.3.6) et familles d'ordonnanceur (Section 1.3.7) sont considérés.

1.3.1 Modèle de tâche

Chaque tâche logicielle τ_i est modélisée selon trois paramètres pour réaliser une analyse d'ordonnancement : la période T_i, l'échéance relative D_i et le temps d'exécution

C_i de la tâche τ_i. Pour une analyse d'ordonnancement, une tâche τ_i est donc représentée par le triplet (T_i, D_i, C_i).

La période de la tâche est l'écart de temps entre les différentes exécutions de la tâche. Les tâches de l'application temps réel étant périodiques, elles sont exécutées plusieurs fois à intervalle de temps régulier, chaque exécution étant une instance d'exécution. On réfère à la j-ème instance d'exécution ou instance de tâche de τ_i par le terme $\tau_{i,j}$. L'échéance relative D_i de la tâche est la durée avant laquelle la tâche doit avoir terminé son exécution relativement à sa date de réveil à la j-ème exécution $R_{i,j}$. Le temps d'exécution de la tâche est donné par son pire temps d'exécution pour garantir la prédictibilité du système. Il se peut que la période soit égale à l'échéance relative.

Il existe d'autres modèles de tâche dans le cas de tâches :

-apériodiques : tâches activées de manière irrégulière.

-sporadiques : tâches activées de manière irrégulière mais avec une contrainte sur la durée minimum entre l'arrivée de deux exécutions consécutives.

Dans certains cas, toutes les tâches ne démarrent pas en même temps. On précise alors pour chaque tâche sa date d'activation O_i qui est la date de première exécution. Ainsi les tâches sont données par le quadruplet (O_i, T_i, D_i, C_i). Si toutes les dates d'activation sont égales, alors les tâches sont dites synchrones.

1.3.2 Représentation graphique du modèle de tâche

La Figure 1.4 résume graphiquement les paramètres du modèle de tâche utilisé par les analyses d'ordonnancement pour la j-ème instance de tâche de τ_i qui a pour paramètres le triplet (T_i, D_i, C_i). La date de réveil de la j-ème instance de tâche $R_{i,j}$ indique la date à laquelle la tâche τ_i est prête à s'exécuter. Néanmoins, une autre tâche peut terminer son exécution après la date de réveil $R_{i,j}$ retardant l'exécution de la tâche τ_i.

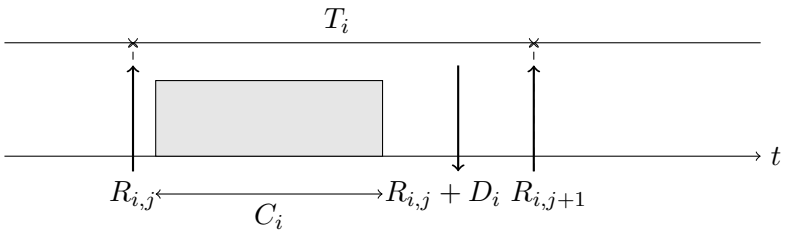


FIGURE 1.4 – Représentation graphique du modèle de tâche périodique

1.3.3 Algorithmes d'ordonnancement

Les algorithmes d'ordonnancement définissent l'ordre d'exécution des tâches sur le processeur, monocœur dans ce cas. Ces algorithmes considèrent le modèle de tâche précédent pour toutes les tâches composant l'application temps réel et ordonnancent les tâches par ordre de priorité. Les algorithmes répondant aux mêmes hypothèses sur

les paramètres des tâches définissent une classe d'algorithme. Un algorithme est optimal tel que s'il ne trouve pas d'ordonnancement pour l'application donnée alors aucun autre algorithme de la même classe ne peut en trouver. Un algorithme est préemptif s'il permet d'interrompre l'exécution d'une tâche de plus basse priorité pour permettre à une tâche de plus haute priorité de s'exécuter. Il existe deux catégories d'algorithmes d'ordonnancement, ceux *à priorité fixe* et ceux *à priorité dynamique*.

Algorithmes à priorité fixe Un ordonnanceur est à priorité fixe si les tâches conservent la même priorité pour toutes leurs instances de tâche. L'ordonnanceur dans ce cas est dit hors ligne puisqu'il propose un ordonnancement calculé au préalable et n'est pas modifié lors de l'exécution du système. Les deux principaux algorithmes à priorité fixe sont Deadline Monotonic (DM) et Rate Monotonic (RM).

-Deadline Monotonic : cet algorithme concerne des tâches périodiques. La priorité d'une tâche est inversement proportionnelle à son échéance relative [LW82]. La tâche avec l'échéance relative la plus courte est donc la plus prioritaire. Il est optimal dans la classe des algorithmes préemptifs à priorité fixe pour des tâches périodiques indépendantes à échéance relative inférieure à la période i.e. $D_i < T_i$.

-Rate Monotonic : cet algorithme concerne des tâches périodiques seulement. Il est tel que la priorité d'une tâche est inversement proportionnelle à sa période [LL73]. La tâche avec la période la plus courte est donc la plus prioritaire. Il est optimal dans la classe des algorithmes préemptifs à priorité fixe pour des tâches périodiques indépendantes à échéance relative égale à la période i.e. $D_i = T_i$.

Algorithmes à priorité dynamique Un ordonnanceur est à priorité dynamique si la priorité d'une instance de tâche est calculée en ligne. Le principal algorithme d'ordonnancement à priorité dynamique est Earliest Deadline First [LL73] (EDF).

Il est tel qu'à une date donnée la tâche la plus prioritaire, parmi les tâches prêtes, est celle dont l'échéance absolue est la plus proche. L'échéance absolue est la date donnée par la somme de la date d'arrivée t et de l'échéance relative D_i. L'algorithme est optimal dans la classe des algorithmes préemptifs à priorité dynamique pour des tâches périodiques indépendantes et dont l'échéance est inférieure à la période.

1.3.4 Analyses de faisabilité

L'ordonnancement de l'application temps réel répond à des conditions de *faisabilité* :
- si une condition suffisante de l'algorithme est respectée, alors l'application est ordonnançable par cet algorithme.
- si une condition nécessaire de l'algorithme n'est pas respectée, alors l'application n'est pas ordonnançable par cet algorithme.

L'algorithme d'ordonnancement n'est pas *optimal* si l'agorithme s'arrête au premier ordonnancement faisable qui n'est pas nécessairement le meilleur. Des conditions d'ordon-

nançabilité sont déterminées selon l'algorithme. Nous allons adresser les conditions des algorithmes présentés précédemment i.e. DM, RM et EDF.

L'analyse de faisabilité de l'ordonnancement par ces algorithmes peut être basée sur le taux d'utilisation de chaque tâche. Le taux d'utilisation u_i d'une tâche est la fraction de temps que la tâche consomme sur un processeur pour s'exécuter tel que $u_i = \frac{C_i}{T_i}$. Dans les trois cas, la somme des taux d'utilisation de chaque tâche inférieure à un, i.e. $\sum_{i=1}^{n} u_i \leq 1$, est une condtion nécessaire d'ordonnançabilité. Cette condition n'est pas suffisante pour tous les algorithmes. La condition suffisante dans chaque cas est :

-DM, RM : $\sum_{i=1}^{n} u_i \leq n\left(2^{\frac{1}{n}} - 1\right)$.

-EDF : $\sum_{i=1}^{n} u_i \leq 1$.

Dans le cas des algorithmes DM et RM, la condition donnée est suffisante mais pas nécessaire. Il se peut que la charge totale soit supérieure à la borne et que le système soit tout de même ordonnançable par DM ou RM. Or comme $\lim_{n\to\infty} \sum_{i=1}^{n} u_i \leq n\left(2^{\frac{1}{n}} - 1\right) = \ln(2) \simeq 0.69$, alors il est possible de conclure que des applications avec une charge d'utilisation inférieure à 0.69 sont ordonnançables par DM et RM.

L'analyse de faisabilité peut être basée d'autre part sur l'analyse des pires temps de réponse. Le temps de réponse d'une tâche est la différence entre la date de fin d'exécution d'une instance de tâche et la date de réveil de l'instance correspondante. Dans un contexte préemptif, une tâche de basse priorité s'exécutant peut être interrompue par une tâche de plus haute priorité prête à s'exécuter. Ainsi, le temps de réponse de la tâche de basse priorité comporte le temps d'exécution de la tâche qui l'a préemptée. De plus, une tâche peut être préemptée par plusieurs tâches. Le calcul du pire temps de réponse $\mathcal{R}_i$ de la tâche τ_i est basé sur l'ensemble $hp(i)$ des indices des tâches de plus haute priorité que τ_i et selon le processus récursif suivant :

$$\mathcal{R}_i^0 = 0 \tag{1.1}$$

$$\mathcal{R}_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{\mathcal{R}_i^n}{T_j} \right\rceil C_j, \ i > 0, \tag{1.2}$$

avec $\lceil . \rceil$ la fonction qui arrondit à l'entier supérieur. Le processus s'arrête si l'une des deux conditions est satisfaite :

— lorsque deux exécutions consécutives renvoient la même valeur soit $\mathcal{R}_i^{n+1} = \mathcal{R}_i^n = \mathcal{R}_i$.

— lorsque une valeur atteint la valeur de l'échéance relative soit $\mathcal{R}_i^{n+1} = D_i$.

Dans le second cas, l'application n'est pas ordonnançable. L'application est ordonnançable si le pire temps de réponse est inférieur à l'échéance $\mathcal{R}_i \leq D_i$.

1.3.5 Protocoles d'accès aux ressources partagées

Une ressource est un élément logiciel qui peut être utilisé par une tâche pour avancer dans son exécution. En cas de ressources partagées, la ressource ne doit pas être

accédée par différentes tâches simultanément. La ressource est alors mutuellement exclusive. Les tâches nécessitant d'accéder à la même ressource sont des tâches concurrentes. La partie de code d'une tâche qui accède à une ressource est une section critique. En particulier, lorsqu'une tâche de haute priorité préempte la tâche s'exécutant dans sa section critique, la tâche de haute priorité ne peut entrer dans sa section critique et son exécution est bloquée. Pour éviter ce genre de cas, les systèmes opératoires temps réel implémentent des mécanismes pour synchroniser les accès aux ressources partagées comme le sémaphore. Un sémaphore est un jeton qu'une tâche doit acquérir pour entrer dans sa section critique. Lorsque la tâche sort de sa section critique, elle relâche le sémaphore et une autre tâche peut l'acquérir pour entrer dans sa section critique. Une tâche souhaitant acquérir un sémaphore détenu par une autre tâche est en *attente* du signal de relâchement du sémaphore. On résume les états d'une instance de tâche par la machine à états de la Figure 1.5. La tâche est réveillée à un instant donné par sa date de réveil et/ou sa période et est en état PRETE. L'ordonnanceur peut l'élire pour la lancer et alors la tâche s'EXECUTE. Lors de son exécution, la tâche peut se retrouver en ATTENTE d'un sémaphore pour entrer dans sa section critique et doit attendre le signal du sémaphore correspondant pour se retrouver PRETE et pouvoir être relancée par l'ordonnanceur. Dans un autre cas la tâche peut être préemptée par une tâche de plus haute priorité PRETE, et se retrouve de nouveau en état PRETE pour être élue par l'ordonnanceur lorsque le cœur est libre.

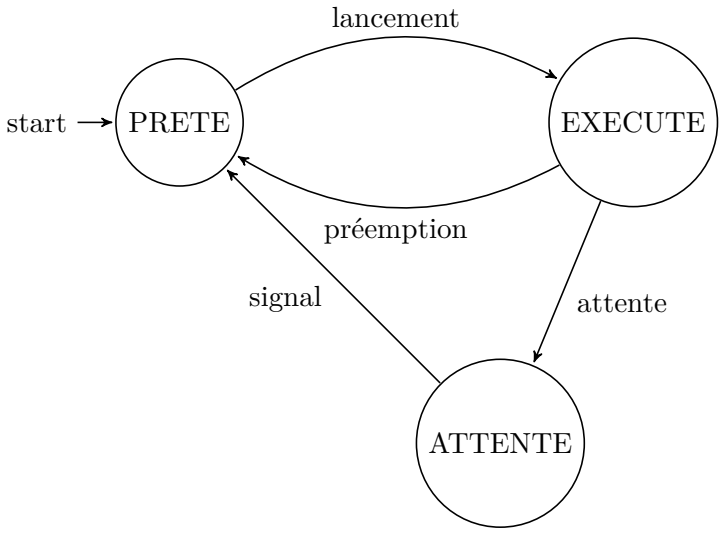


FIGURE 1.5 – Machine à états d'une instance de tâche.

Le partage de ressource mutuellement exclusive entre différentes tâches peut conduire au phénomène d'inversion de priorité. En reprenant le cas de blocage ci-avant, la tâche de plus basse priorité doit terminer son exécution dans sa section décrite pour débloquer la tâche de plus haute priorité. En conséquence, la tâche de plus basse priorité préempte la tâche de haute priorité, c'est une inversion de priorité. De plus, si une tâche de priorité

intermédaire est activée, alors elle va préempter la tâche de priorité basse s'exécutant, bloquant encore plus longtemps la tâche de haute priorité. Dans ce cas, la tâche de haute priorité peut se retrouver bloquée pendant une durée non bornée. Certains protocoles ont été implémentés pour pallier à ce problème d'inversion de priorité, comme le *protocole d'héritage de priorité* et celui de *plafonnement de priorité* [SRL90] applicables pour les applications à priorité fixe. La politique de *pile de ressource* est applicable pour les applications à priorité fixe et dynamiques.

1.3.6 Ordonnancement multicœur/multiprocesseur

On considère de manière équivalente une architecture composée de m processeurs monocœur et une architecture dotée d'un processeur multicœur contenant m cœurs. Ce sont les m cœurs d'exécution qui sont dimensionnants. Si les cœurs d'exécution ont des propriétés différentes, comme leur capacité de calcul, alors l'architecture est hétérogène, sinon elle est homogène. Dans la suite de cette thèse on ne considérera que les architectures homogènes.

L'exécution multicœur d'une application permet d'exécuter plusieurs tâches en parallèle. Néanmoins, un cœur ne peut exécuter qu'au plus une tâche à chaque instant et une tâche s'exécute sur au plus un cœur à chaque instant. On distingue deux familles d'algorithmes d'ordonnancement multicœur, *par partionnement* ou *global*. Ces deux familles sont incomparables.

Dans le cas de l'ordonnancement par partitionnement, l'application τ est découpée en m sous-ensembles de tâches $\tau^1, \ldots, \tau^m$ chacun attribué à un cœur. Les tâches ne sont pas autorisées à migrer i.e. s'exécuter sur un cœur puis un autre et ainsi de suite. Enfin, on applique un algorithme d'ordonnancement monocœur pour chaque sous-ensemble de tâche. On souhaite placer dans m cœurs les sous-ensembles de tâches de taille et de taux d'utilisation total différent. La taille des cœurs correspond à l'utilisation maximale que l'on peut atteindre, $\ln(2)$ si l'on utilise l'algorithme RM et 1 si l'on utilise EDF. Ce problème est NP-difficile et des heuristiques sont utilisées pour approcher la solution optimale [LL85].

Dans le cas global, il s'agit d'attribuer à chaque instant les m processeurs aux tâches prêtes les plus prioritaires. Dans ce cas, les tâches peuvent migrer i.e. une tâche peut commencer son exécution sur un cœur, être préemptée et reprendre son exécution sur un autre cœur. Naturellement, les algorithmes RM et EDF ont été appliqués dans le cas global et sont notés dans ce cas g-RM et g-EDF dans la littérature. Néanmoins, de telles transpositions ne permettent pas d'atteindre l'optimalité dans le cas multicœur. La mise en place de méthodes dites équitables [BCPV96] (fairness) permettent d'atteindre l'optimalité vis-à-vis de la limite de taux d'utilisation de chaque cœur. Une méthode est équitable si chaque tâche τ_i reçoit en moyenne une part de ressource de calcul similaire au fil du temps.

L'ordonnancement multicœur pose un problème d'allocation spatiale des tâches de l'application temps réel sur les différents cœurs du processeur.

1.3.7 Autres familles d'ordonnanceur

Avec prise en compte des interférences Les optimisations matérielles des plateformes visent à réduire en moyenne le temps d'exécution logiciel. Néanmoins, les architectures multicœur permettant l'exécution en parallèle de plusieurs tâches, ces dernières peuvent interférer entre elles via leurs accès mémoires. Par exemple, les auteurs de [MGP16] prennent en compte les interférences entre les tâches qui souhaitent accéder à la mémoire principale partagée. Dans cet article, les auteurs proposent un test suffisant d'ordonnançabilité basé sur les pires temps de réponse. En plus de l'équation (1.2), les auteurs calculent le pire temps de réponse de chaque tâche dû à ses accès à la mémoire principale. Cet accès peut être retardé à cause d'autres taches accédant également à la mémoire principale. Avec ce nouveau modèle de calcul du pire temps de réponse, les auteurs choisissent une répartition possible des tâches sur la plateforme et testent l'ordonnançabilité. Si le test n'est pas vérifié, alors les auteurs choisissent une nouvelle répartition. Les auteurs justifient la prise en compte des interférences dans l'analyse d'ordonnançabilité car les analyses de pire temps d'exécution sont trop complexes dans le cas d'architectures multicœur.

A criticité mixte On trouve également des systèmes temps réel intégrant à la fois des tâches de criticité élevée et de criticité basse. Les tâches sont décrites par leur niveau d'assurance (DAL Design Assurance Level), qui va du niveau A, criticité haute, au niveau E, criticité basse, comme le précise la DO-178B [Pot12]. Par exemple, à bord d'un avion, les tâches relatives au pilote automatique sont de criticité élevée DAL A, et répondent à des contraintes temps réel dures, alors que les tâches liées au divertissement des passagers sont elles de faible criticité DAL E. Dans le modèle de criticité mixte, l'exécution des tâches importantes avant leur échéance doit être garantie alors que les tâches de faible criticité peuvent dépasser leur échéance. Le modèle de tâche de Vestal [Ves07, ENNT15] formalise l'approche des systèmes à criticicité mixte. Ce modèle de tâche considère plusieurs temps d'exécution au lieu d'un seul habituellement pour faciliter l'analyse d'ordonnançabilité et assurer à la fois l'exécution de toutes les tâches et l'exécution sûre des tâches de haute priorité. Dans un modèle à deux temps d'exécution, on a $C_i(\text{Lo})$ et $C_i(\text{Hi})$, Lo pour la criticité basse et Hi pour la criticité haute. En pratique, la confiance ou fiabilité dans la borne $C_i(\text{Hi})$ est plus importante et la borne est souvent plus large que $C_i(\text{Lo})$. En plus de ce modèle de tâche, des analyses d'ordonnancement sont étudiées pour l'exécution de tâches de différentes criticités [DB11]. Dans l'industrie, les tâches de différentes criticités sont spatiallement et temporellement partitionnées. Ces partitionnements tendent à éviter que les tâches de basse criticité interférent avec les tâches de haute criticité.

1.3.8 Résumé

La prédictibilité d'un système temps réel est décidée par l'analyse d'ordonnancement des tâches logicielles composant l'application temps réel. Il existe divers types d'analyses d'ordonnancement et plusieurs dizaines d'algorithmes d'ordonnancement associés.

Les choix concernant l'analyse et l'algorithme à utiliser pour décider de l'ordonnancement d'une application temps réel revient au concepteur de cette dernière. Enfin, pour garantir l'analyse d'ordonnancement, il convient d'estimer de manière sûre le pire temps d'exécution de chaque tâche.

Chapitre 2

Présentation des méthodes d'estimation du pire temps d'exécution

L'exécution sûre d'un système critique contrôlé par un système embarqué temps réel dépend de l'ordonnancement de l'application temps réel qui nécessite l'estimation du pire temps d'exécution (Worst Case Execution Time WCET) de chaque tâche. Il est nécessaire de considérer le pire cas pour garantir la prédictibilité du système. Le problème de l'estimation du WCET d'une tâche est représenté dans la Figure 2.1.

Le WCET correspond à la plus petite borne supérieure sur tous les temps d'exécution possibles d'une tâche. Il s'agit donc d'une valeur théorique qu'il faut estimer ou majorer puisque sa connaissance exacte est trop coûteuse voire impossible. Inversement, le plus grand minorant sur tous les temps d'exécution possibles d'une tâche est le meilleur temps d'exécution (Best Case Execution Time BCET). Le WCET estimé d'une tâche répond de manière stricte au critère de *sûreté*, i.e. il est plus grand que tous les temps d'exécution possibles de la tâche. De plus, il est préférable qu'il soit *précis*, i.e. il n'est pas trop surestimé par rapport au vrai WCET de la tâche. Si le WCET estimé est trop éloigné de la valeur théorique, alors il est pessimiste. En effet, l'ordonnançabilité d'une application dépend fortement de sa charge d'utilisation du processeur et donc des valeurs de WCET C_i de chaque tâche τ_i. En conséquence, plus la charge d'utilisation est réduite, plus il y a de chance que l'application soit ordonnançable.

On distingue parmi les méthodes d'estimation du WCET les méthodes déterministes et les méthodes probabilistes. Les méthodes déterministes (Section 2.1) fournissent une valeur unique du WCET estimé. Au contraire, les méthodes probabilistes (Section 2.2) fournissent plusieurs valeurs de WCET respectivement associées à une probabilité.

De plus, ces méthodes peuvent être statiques ou basées mesures. Les méthodes statiques sont basées sur la modélisation précise de la tâche étudiée ainsi que de la plateforme sur laquelle la tâche est amenée à être exécutée. Les méthodes basées mesures consistent à exécuter la tâche étudiée sur la plateforme considérée et à mesurer son temps d'exécution. Historiquement, les méthodes d'estimation du WCET sont à l'origine déterministes. L'apparition des méthodes probabilistes coïncide avec la volonté (et/ou

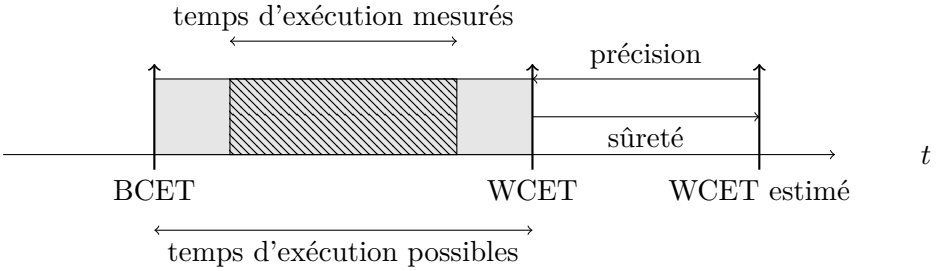


FIGURE 2.1 – Le problème de l'estimation du WCET.

la nécessité) d'employer des processeurs *complexes*, i.e. des processeurs comportant des mécanismes d'optimisation, pour les systèmes temps réel.

2.1 Méthodes déterministes d'estimation du WCET

Dans cette section, les méthodes déterministes d'estimation du WCET sont présentées. L'objectif des méthodes déterministes d'estimation du WCET est de fournir une unique borne supérieure sur tous les temps d'exécution possibles de la tâche étudiée. Pour aborder les notions générales et les enjeux de l'analyse temporelle de tâches logicielles, les approches statiques (Static Deterministic Timing Analysis SDTA) sont considérées en premier lieu (Section 2.1.1). Dans un deuxième temps, les approches déterministes basées mesures sont décrites (Section 2.1.2) (Measurement Based Deterministic Timing Analysis MBDTA).

2.1.1 Méthodes déterministes statiques

Les méthodes d'analyse statique du WCET reposent sur l'analyse du code de la tâche ou analyse haut niveau, et sur l'analyse du processeur ou analyse bas niveau. La chaîne de travail des méthodes statiques est généralement décrite comme dans la Figure 2.2.

L'entrée de l'analyse est la tâche à étudier donnée soit par le code source, soit par son exécutable. La deuxième entrée de l'analyse est le modèle du processeur considéré.

Analyse de flot Dans un premier temps, une analyse de flot de la tâche est réalisée. L'analyse de flot permet de déterminer les chemins possibles de la tâche. Un chemin est un ensemble d'opérations réalisées par la tâche qui peut être constitué d'un ensemble de branchements conditionnels if et de boucles for ou $while$. Le chemin exécuté de la tâche dépend de son vecteur de paramètres d'entrée qui va déterminer les branches à suivre et le nombre de boucles à réaliser. Cette analyse peut être basée soit sur le code source de la tâche ou bien sur son exécutable.

Selon la méthode employée, l'ensemble des chemins déterminés se rapproche de l'ensemble des chemins possibles. En effet, selon le degré de connaissance sur la tâche, il se peut que l'analyse de flot retourne des chemins infaisables lors de l'exécution de la tâche.

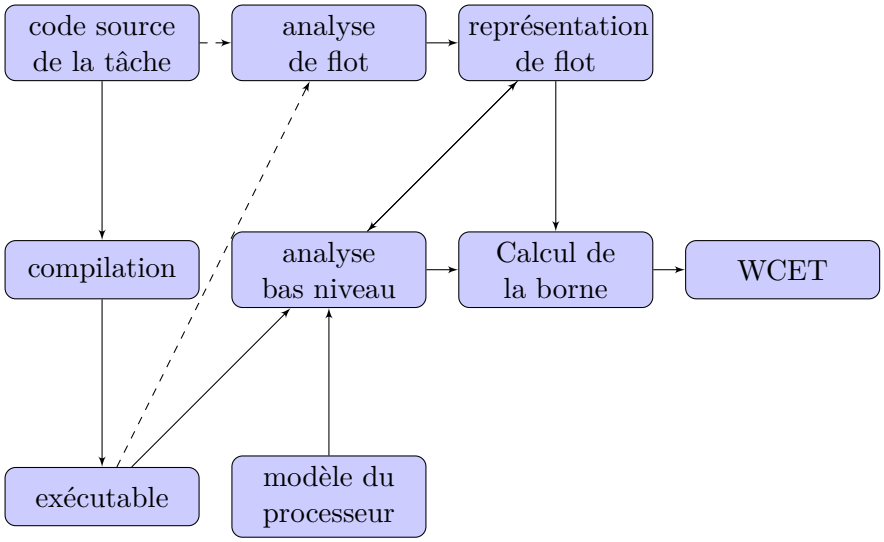


FIGURE 2.2 – Composants principaux d'une analyse statique du WCET d'une tâche [EES+03, WEE+08].

Les chemins infaisables peuvent retourner une estimation du WCET trop pessimiste. Le nombre de chemins déterminés par l'analyse est souvent supérieur au nombre de chemins possibles. Pour restreindre le nombre de chemins d'exécution possibles, l'analyse de flot doit être capable de [CPRS03] :

— déterminer la borne supérieure sur le nombre d'itérations des boucles.
— contraindre le choix d'un branchement conditionnel dans le cas où la variable de décision est une constante.
— restreindre le nombre de chemins d'exécution possibles notamment dans le cas de chemins mutuellement exclusifs.

On considère en pratique deux méthodes pour obtenir ces informations sur le flot de contrôle de la tâche étudiée. La première, et la plus répandue, repose sur des annotations manuelles du programmeur. Ces annotations peuvent être intégrées dans le code source de la tâche. Par exemple, le langage C est étendu dans [PK89] par des syntaxes additionnelles pour borner le nombre d'itérations des boucles. Les annotations peuvent également être prises en compte de manière interactive lors de l'analyse comme dans [KWH95]. Il existe également des méthodes automatiques qui restent difficile à généraliser. On peut citer une méthode automatique basée sur l'interprétation abstraite [EES+03], afin d'abstraire les données d'entrée de la tâche et déterminer les chemins possibles. D'autres méthodes font usage de la satisfaction de contraintes [RMMA15] en utilisant la *Satisfaction Modulo Theory* qui étend les méthodes de satisfaction à des types autres que des entiers.

Représentation de flot L'analyse de flot enrichit la représentation de flot de la tâche étudiée par des informations qui ne peuvent être contenues dans la représentation

précédente. Deux types de représentation existent, le graphe de flot de contrôle et les arbres syntaxiques abstraits. Pour illustrer ces deux représentations, nous considérons la tâche de l'Exemple 2.1.

Exemple 2.1 (*Exemple de code source d'une tâche*)
Pour exemple nous prenons le calcul de la suite de Fibonacci inspiré du code C provenant du benchmark Mälardelen [MAL13].

```
1  int fib(int n){
     int i, Fnew, Fold, temp,ans; //BB_1
3    if(n==1){
       ans=n //BB_2
5    }else{
       Fnew = 1;  Fold = 0; //BB_3
7      for ( i = 2;
       i <= 30 && i <= n;
9      i++ ){
         temp = Fnew; //BB_4
11       Fnew = Fnew + Fold;
         Fold = temp;
13     }
       ans = Fnew; //BB_5
15   }
     return ans; //BB_6
17 }
```

Le *graphe de flot de contrôle* (Control Flow Graph CFG) d'une tâche est un graphe orienté. Les nœuds sont des *blocs de bases* (Basic Block BB) et les arcs représentent les relations de précédence entre les blocs. Un bloc de base est une suite d'instructions séquentielles. La tâche à analyser est découpée en blocs de base comme dans l'Exemple 2.1. Les blocs de base BB_i pour i de 1 à 6 sont indiqués en commentaire du code. Un arc orienté va du nœud père au nœud fils. Le graphe de flot de contrôle correspondant est donné dans la Figure 2.3

Ce CFG a un nœud d'entrée BB_1 et un nœud de sortie BB_6. Un branchement conditionnel comme au niveau de BB_1 a plusieurs nœuds fils, et dans ce cas deux. Dans cet exemple de CFG, il est impossible de connaître le nombre de boucles d'itération de BB_4. Une analyse de flot de contrôle révèle que le nombre maximum d'itérations de cette boucle est de 29.

Une autre catégorie de représentation de flot existe sous la forme d'*arbres syntaxiques abstraits* (Abstract Syntax Tree AST). Dans un AST, des branches représentent les structures du langage, et des feuilles les blocs de base. Les structures du langages peuvent être :

— les séquences SEQ qui possèdent au moins un fils (branche ou feuille).
— les boucles LOOP qui possèdent deux fils, un pour le test et un pour le corps de

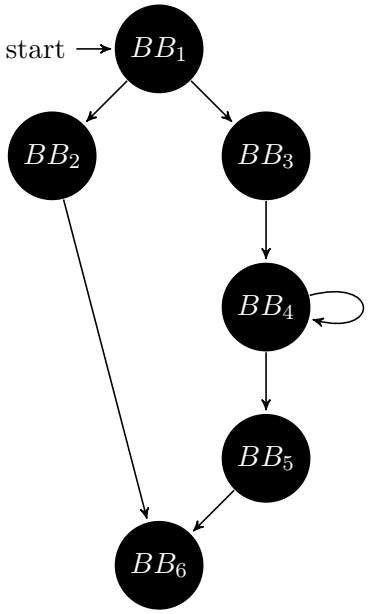


FIGURE 2.3 – Graphe de flot de contrôle de la tâche fib.

la boucle.

— les branchements conditionnels IF qui possèdent trois fils, un fils test if, un fils *then* et un fils *else*.

— les appels de fonction CALL qui ne possèdent pas de fils.

L'AST de la tâche exemple fib est donné dans la Figure 2.4.

Nous avons produit ici un CFG et un AST de la tâche fib à partir de son code source en langage C. Il est également possible de produire des CFGs et ASTs à partir de l'exécutable binaire de la tâche étudiée comme dans [MBBF16].

Calcul de la borne supérieure sur le pire temps d'exécution On distingue dans ce paragraphe deux méthodes de calcul de la borne sur le WCET qui est soit une estimation du WCET théorique soit une majoration.

La première méthode est basée sur la représentation de flot sous la forme d'AST. [PK89] donne ainsi un ensemble de règles pour calculer le temps d'exécution maximal d'une tâche à partir d'un AST. Ces règles dépendent de la structure du programme et des fils de chaque nœud notés S_i. Les règles de calcul ainsi définies sont données dans le Tableau 2.1. Dans ce cas, on note WCET la fonction qui associe à un fils son pire temps d'exécution.

Le WCET d'une séquence SEQ de fils est la somme des WCETs de ses fils. Le WCET d'un branchement conditionnel IF est le WCET maximum entre ses deux fils *then* et *else*. Le WCET d'une boucle LOOP est le WCET de la boucle multiplié par le nombre maximum d'itérations de la boucle. Le WCET calculé est sûr si les WCETs de tous les blocs de base le sont.

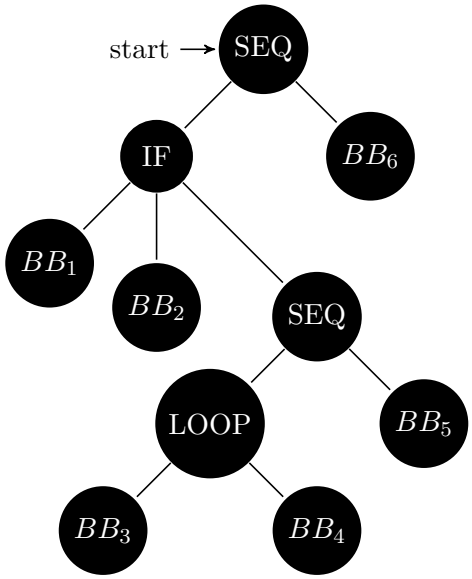


FIGURE 2.4 – Arbre syntaxique abstrait de la tâche fib.

Structure	Fils	Règle
SEQ	$S_1, \ldots, S_n$	$\sum$ WCET(S_i)
IF	$if(S_1)$ *then* S_2 *else* S_3	WCET(S_1)+max(WCET(S_2),WCET(S_3))
LOOP	$for(S_1)$ *do* S_2	WCET(S_1)+N×(WCET(S_1)+WCET(S_2))

TABLE 2.1 – Règles de calcul du WCET dans le cas d'une méthode basée sur les ASTs. N est le nombre maximum d'itérations de la boucle for.

La deuxième méthode est basée sur le CFG d'une tâche et utilise la programmation linéaire en nombre entier. Elle met en œuvre la technique d'énumération implicite de chemin (Implicit Path Enumeration IPET). La technique consiste à écrire un système d'équations représentant les contraintes données par le CFG. Les variables du système d'équations sont le temps d'exécution t_i et le nombre d'exécutions f_i de chaque bloc de base BB_i. Les contraintes structurelles pour représenter les chemins possibles sont les suivantes :

- les nœuds d'entrée et de sortie sont exécutés exactement une fois.
- le nombre d'exécutions de chaque bloc de base est égal à la somme des nombres d'exécution des blocs précédents *prec* (suivants *succ*) le bloc de base i.e. $\forall i \; f_i = \sum_{a \in prec(i)} f_a = \sum_{b \in succ(i)} f_b$.
- le nombre maximum d'itérations d'une boucle.
- les chemins mutuellement exclusifs sont exprimés par $f_i + f_j = 1$.
- autres contraintes plus complexes.

Le problème considéré est équivalent à un problème d'optimisation qui consiste à maxi-

miser la fonction objective étant la somme des temps d'exécution t_i de chaque bloc multiplié par leur nombre d'exécutions f_i i.e. maximiser $\sum_i t_i \times f_i$. Cette technique est néanmoins limitée par la complexité de calcul qu'elle requiert. Plusieurs outils d'analyse du WCET comme AiT, Bound-T, Heptane, OTAWA et SWEET [WEE+08] sont basés sur la technique IPET.

Dans l'exemple de la tâche fib, BB_1 et BB_6 étant respectivement les blocs d'entrée et de sortie alors $f_1 = f_6 = 1$. De plus, le nombre d'itération maximum de la boucle étant de 29 alors $f_4 \leq 29$.

Analyse bas niveau L'analyse bas niveau dans les approches SDTA correspond à déterminer le comportement de la tâche sur un processeur donné. C'est une étape critique dans l'approche car le degré de précision de l'analyse impacte directement le pessimisme de l'estimation de la borne sur le WCET. L'analyse fournit en outre les temps d'exécution des blocs de base à l'analyse haut niveau.

La complexité de l'analyse dépend de la complexité du processeur considéré. Pour une architecture *simple*, processeurs et microcontrolleurs 8 bits, 16 bits, 32 bits, le temps d'exécution d'une instruction dépend uniquement de son type et de ses opérandes. Dans le cas d'une architecture *complexe*, i.e. comprenant des mécanismes d'optimisation des performances d'exécution, le temps d'exécution d'une instruction dépend de plusieurs effets. Tout d'abord on trouve les effets provenant du parallélisme des instructions dans le pipeline d'instructions. Il faut également considérer les effets provenant de la présence ou non des données ou de l'instruction dans le cache respectif. Les analyses de pire temps d'exécution considèrent l'exécution de la tâche sans interférence e.g., des interruptions ou des préemptions [EES+03]. Le prise en compte des interférences au niveau des mémoires caches est laissée aux analyses d'ordonnançabilité [BMSO+96]. Ce choix provient de la complexité de prendre en compte les interférences dans l'estimation sûre du WCET [MGP16].

L'intérêt d'une exécution pipelinée des instructions est de pouvoir paralléliser leur exécution et donc ne pas avoir à attendre la fin d'une instruction pour exécuter une instruction indépendante. En conséquence, si l'on ajoute simplement les temps d'exécution de chaque instruction, l'estimation est trop pessimiste. La prise en compte du pipeline dans l'analyse considère usuellement des tables de réservation des étages du pipeline par les instructions. En se basant sur la table de réservation établie, le temps d'exécution de chaque instruction est déduit par une somme.

L'utilisation des mémoires caches permet d'accéder rapidement à des blocs mémoires, d'instructions ou de données. Si le bloc demandé n'est pas dans le cache, alors c'est un *miss* sinon c'est un *hit*. Un $miss$ entraîne un délai supplémentaire. Si l'on ne considère que des $misses$ pour chaque accès mémoire, alors l'estimation du WCET est trop pessimiste. La connaissance des blocs présents dans le cache permet de réduire le pessimisme de l'estimation. Néanmoins, une analyse concrète des contenus de la mémoire cache est trop complexe. Par conséquent, une méthode abstraite a été mise en place [CC77]. L'analyse abstraite des états du cache (Abstract Cache State ACS) est basée sur le CFG de la tâche. En suivant le CFG, l'approche ACS procède à trois analyses. Une analyse *must*

(doit) détermine les blocs dont on est sûr de la présence dans le cache. Une analyse *may* (peut) détermine les blocs qui pourraient être dans le cache. Une analyse de persistance détermine les blocs qui vont rester dans le cache e.g., un premier accès entraîne un *miss* mais le bloc reste par la suite dans le cache et ne provoque que des *hits*. Une fois les états du cache déterminés à chaque point de l'exécution de la tâche pour chaque type d'analyse, on peut déterminer quel bloc va entraîner un *miss* ou un *hit*. Si le bloc est dans les états de l'analyse *must*, il provoquera toujours un *hit*, *always-hit*. Si le bloc n'est pas dans les états de l'analyse *may*, il provoquera toujours un *miss*, *always-miss*. Si le bloc est dans les états de l'analyse de persistance, il entraîne en premier un *miss* puis des *hits*, *first-miss*. Si l'on ne sait rien sur le bloc il est non classé, *not classified*.

Des travaux s'intéressent également à la prise en compte des délais dûs aux bus d'accès mémoire selon le protocole utilisé. Par exemple, [RMMA15] prend en compte l'ordonnancement des accès mémoires dans le bus de type Time Division Multiple Access dans son analyse de l'estimation du WCET.

Anomalies temporelles Une des complexités de l'analyse bas niveau est la présence d'anomalies temporelles que peut entraîner le processeur considéré [RWT+06]. Dans le cas d'anomalies temporelles, le pire cas local, au niveau des instructions, ne mène pas au pire cas global, i.e. au niveau de la tâche. Les anomalies temporelles ont principalement lieu à cause des choix du pipeline. Les pipelines modernes peuvent spéculer sur les branches qui vont être exécutées. Néanmoins, la mauvaise prédiction d'instructions après un *hit* peut être plus coûteux que dans le cas où l'accès mémoire fut un *miss*. Un autre cas d'anomalie temporelle concerne le cas d'ordonnancements différents des mêmes instructions dans le pipeline qui donnent différents temps d'exécution. En conséquence, l'hypothèse selon laquelle il faut considérer les pires cas locaux n'est pas une hypothèse sûre.

Surcouts temporels dus à la préemption Une autre difficulté de l'analyse bas niveau est la prise en compte des surcouts temporels dus à la préemption. Une tâche se retrouve préemptée lorsqu'une tâche de plus haute priorité est prête à s'exécuter. Dans ce cas, la tâche qui préempte vide le cache des données et instructions de la tâche préemptée pour charger les siennes. Lorsque la tâche préemptée reprendra son exécution, elle devra alors charger de nouveau ses données et instructions dans le cache entraînant des surcouts temporels qui n'auraient pas eu lieu en cas de non préemption. Comme décrit dans [AB09], il faut ajouter au WCET le nombre du surcouts temporels dus à la préemption.

2.1.2 Méthodes déterministes basées mesures

Les méthodes basées mesures peuvent considérer la mesure du temps d'exécution bout en bout de la tâche ou la mesure du temps d'exécution des blocs de base de la tâche.

Dans le premier cas, le choix du paramètre d'entrée de la tâche est primordial pour

l'estimation finale. En effet, si l'on considère de suite la pire entrée de la tâche i.e. celle qui donne le pire chemin, alors on obtient une mesure du WCET. Si cette pire entrée est inconnue, il faut la rechercher. Pour garantir le propriété de sûreté de l'estimation de la borne sur le WCET, la recherche des paramètres d'entrée peut être exhaustive ou peut converger vers la pire entrée de la tâche. Une recherche exhaustive est impossible en pratique car elle nécessiterait un nombre irréalisable de mesures. Par exemple, pour la fonction $foo(x, y, z)$ avec les entrées x, y et z des entiers codés sur 16 bits, il faudrait potentiellement réaliser $2^{16} \times 2^{16} \times 2^{16} \approx 3 \times 10^{14}$ mesures [EES+03].

La recherche naïve des pires entrées d'une tâche pour l'estimation du WCET est souvent trop coûteuse pour être appliquée. Des méthodes ont alors été mises en œuvre pour réduire le nombre de mesures à réaliser tout en garantissant la sûreté de l'estimation du WCET par ces techniques.

[WMMR05] cherche à couvrir de manière exhaustive les chemins qui constituent la tâche étudiée. Si l'ensemble des entrées possibles est connu, alors on choisit une entrée avec laquelle la tâche est exécutée. Le chemin correspondant est alors exécuté. On détermine, par résolution de contraintes, les entrées qui conduisent au même chemin fournissant ainsi un ensemble d'entrées associées au chemin exécuté précédemment. On choisit de nouveau une entrée dans l'ensemble de départ, n'appartenant pas à l'ensemble d'entrées précédent, et on répète l'opération jusqu'à ce que l'ensemble des entrées de la tâche soit couvert. Dans le cas où l'ensemble des paramètres d'entrée a été couvert, il suffit de prendre le temps d'exécution du chemin le plus grand comme estimation sûre du WCET.

[PN98] utilise un algorithme génétique pour exécuter les chemins d'exécution les plus longs. Dans un premier temps, des entrées sont générées aléatoirement et l'on mesure le temps d'exécution des chemins respectifs. Les entrées qui donnent les temps d'exécution les plus élevés sont conservés et mutés pour exécuter les nouveaux chemins associés dans le but de maximiser le temps d'exécution. Quand le critère d'arrêt est atteint, on choisit le temps d'exécution généré le plus élevé. Néanmoins, il est difficile de garantir la sûreté de cette méthode puisque tous les chemins ne sont pas couverts.

Dans [LHT00], l'auteur utilise la représentation CFG pour déduire un système d'équations dont les inconnues sont les temps d'exécution de chaque bloc de base. La mesure du temps d'exécution de chaque bloc de base est difficile à réaliser car la granularité imposée est trop fine. L'auteur mesure donc le temps d'exécution d'ensembles de blocs de base qui vont servir à résoudre le système d'équations.

Dans l'industrie, le concepteur choisit les cas de tests à réaliser pour obtenir des mesures du temps d'exécution. Une marge de sécurité de l'ordre de 20% est ajoutée au temps d'exécution maximal mesuré. Cette marge n'a pas de raison scientifique mais semble suffisante compte tenu de la simplicité des processeurs utilisés.

La limitation des méthodes basées mesures est due aux états des composants du processeur utilisés qui sont difficiles à imposer. Il est nécessaire que ces composants soient dans leur pire état pour garantir la sûreté de la mesure. Déterminer et imposer un tel état pour chaque composant est très difficile et limite l'utilisation des approches déterministes basées mesures dans le cas de processeurs complexes.

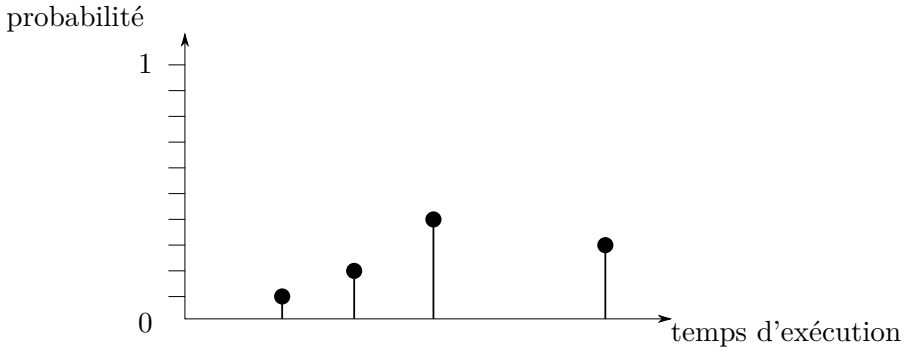


FIGURE 2.5 – Représentation graphique d'un ETP. Chaque barre est un temps d'exécution possible de la tâche étudiée.

2.2 Méthodes probabilistes

La complexité croissante des processeurs considérés dans les analyses d'estimation du WCET a mis en évidence le non déterminisme temporel du temps d'exécution [PMB13]. Ce non déterminisme a poussé le développement de méthodes probabilistes d'estimation du WCET. Les méthodes probabilistes considèrent le temps d'exécution d'une tâche comme une variable aléatoire. Une variable aléatoire est représentée par une distribution de probabilité qui associe à un ensemble de temps d'exécution une probabilité d'occurrence.

Les variables aléatoires qui représentent les valeurs prises par le temps d'exécution d'une tâche sont appelées profil de temps d'exécution (Execution Time Profile ETP). Un exemple d'ETP est donné dans la Figure 2.5. Comme dans les approches déterministes, l'objectif est de borner de manière supérieure les ETPs de la tâche. Cette borne est définie comme le pire temps d'exécution probabiliste (probabilistic Worst Case Execution Time pWCET). Le pWCET est également un ETP.

Tout comme les méthodes déterministes, leurs homologues probabilistes peuvent être statiques ou basées mesures. Néanmoins, l'utilisation des probabilités pour le problème de l'estimation du WCET a d'abord considéré des mesures (Section 2.2.1) (Measurement Based Probabilistic Timing Analysis MBPTA). Les méthodes statiques probabilistes sont spécifiques à des architectures matérielles particulières (Section 2.2.2) (Static Probabilistic Timing Analysis SPTA).

2.2.1 Méthodes probabilistes basées mesures

Deux principales méthodes existent parmi les approches MBPTA. La première est basée sur l'application de la théorie des valeurs extrêmes (Extreme Value Theory EVT). La seconde considère la composition des ETPs de chaque bloc de base dans la représentation CFG de la tâche pour construire l'ETP de la tâche.

Basées sur l'EVT Afin de couvrir les incertitudes de mesure dues à la variabilité du temps d'exécution, les approches basées mesures utilisent l'EVT. L'EVT est un modèle probabiliste pour les comportements extrêmes des phénomènes aléatoires [EKM97]. Elle est largement utilisée dans les domaines de la météorologie et de la finance.

En comparaison, la méthode de Monte-Carlo considère un nombre donné de mesures d'un phénomène aléatoire. Pour estimer la probabilité d'un évènement qui est de l'ordre de 10^{-9} avec un taux de 10% d'erreur, l'approche par Monte-Carlo nécessite l'observation de plus de 10^{11} mesures. Un tel nombre de mesures est bien trop coûteux pour des problèmes réels. En conséquence, l'EVT est utilisée sur un ensemble plus réduit d'observations dans le but d'inférer des valeurs du paramètre mesuré qui ont de très faibles probabilités d'apparition, i.e. des évènements extrêmes. La distribution de valeurs inférées modélise la queue de distribution du paramètre mesuré, ici le temps d'exécution.

L'EVT pour le calcul du WCET est appliquée dans [EB01] pour prendre en compte les optimisations des processeurs. Les auteurs collectent des mesures du temps d'exécution de la tâche considérée dans une trace. L'application de l'EVT est basée sur le théorème de Fisher-Tippett [EKM97] qui définit la distribution des maxima de la trace des temps d'exécution. Cette distribution est celle des Valeurs Extrêmes Généralisées (Generalized Extreme Value distribution GEV). En pratique, la trace de temps d'exécution est divisée en blocs de même taille. Le temps d'exécution maximum de chaque bloc est extrait et ces derniers constituent les maxima de la trace. Un algorithme d'estimation est appliqué pour déterminer les paramètres de la distribution GEV qui correspond le plus à la distribution expérimentale des maxima. La distribution déterminée est le pWCET de la tâche.

L'application de l'EVT permet d'inférer des valeurs de temps d'exécution difficiles à mesurer car ils ne seraient observables seulement en imposant le pire état des composants du processeur ou en imposant une combinaison inconnue d'états qui serait pire. Or, nous avons abordé précédemment la difficulté d'imposer de tels états pour les approches basées mesures motivant l'introduction d'un cadre de travail basé sur les statistiques. En revanche, l'approche requiert toujours des garanties sur la couverture des chemins de la tâche.

Par convolution L'approche MBPTA par convolution est utilisée dans l'outil RapiTime [BCP03] de calcul du pWCET. Cette approche reconsidère les règles de calcul du WCET de [PN98] résumées dans le tableau 2.1. De même, l'approche MBPTA par convolution nécessite la connaissance de l'AST de la tâche.

Au lieu d'avoir une seule valeur de WCET par bloc de base, l'approche par convolution utilise l'ETP de chaque bloc de base. Dans cette approche, l'opération de convolution entre les ETPs d'une séquence de blocs de base remplace l'opération d'addition. Considérons deux blocs de base BB_1 et BB_2 composant une séquence de blocs de base, et leur ETP respectif $\mathcal{X}$ et $\mathcal{Y}$. Notons $\mathcal{Z}$ l'ETP de la séquence BB_1, BB_2. La convolution des ETPs $\mathcal{Z} = \mathcal{X} \oplus \mathcal{Y}$ est calculée comme suivant :

$$P(\mathcal{Z} = z) = \sum_{t=-\infty}^{t=+\infty} P(\mathcal{X} = t)P(\mathcal{Y} = z - t). \tag{2.1}$$

Les variables t et z sont des temps d'exécution dans l'ETP que peut prendre le bloc de base correspondant. L'approche introduit d'autres règles pour les cas des branchements conditionnels et des boucles.

Cette approche est basée mesures puisqu'elle nécessite des mesures du temps d'exécution de chaque bloc de base. Pour chaque bloc de base, plusieurs mesures sont réalisées pour obtenir l'ETP du bloc de base. Avec les ETPs de chaque bloc de base, l'approche suit les chemins de la tâche en suivant son AST et combine les ETPs selon les règles de calcul définies. La combinaison des ETPs des blocs de base permet de déduire le pWCET de la tâche en ne conservant que les temps d'exécution les plus élevés i.e. ceux qui sont dans la queue de l'ETP.

2.2.2 Méthodes probabilistes statiques

Les approches SPTA ont été développées pour les processeurs monocœur dont les mémoires caches font usage de politiques de placement et de remplacement aléatoires [KAQC13a]. Les architectures avec de telles mémoires caches sont les architectures randomisées.

Les approches SPTA utilisent également la convolution d'ETPs pour déterminer le comportement temporel de la tâche. L'application de la méthode consiste à suivre un chemin et à effectuer la convolution des ETPs de chaque bloc de base. Il s'agit de la même technique présentée dans l'approche MBPTA par convolution. En revanche, les approches SPTA déterminent les ETPs de chaque bloc de base grâce à la connaissance des accès mémoires compilés sous la forme d'une trace. Par exemple, pour un accès à la référence mémoire M, et une trace d'accès aux références mémoires : $MB_1B_2 \dots B_kM$ i.e. un accès à la référence M, puis au bloc B_1, puis B_2 jusqu'à B_k et enfin de nouveau M. Les accès B_i sont des accès à des blocs de cache tous différents et en particulier différents du bloc contenant la référence M. Sachant que les politiques de remplacement du cache considéré sont aléatoires, alors pour chaque accès mémoire on peut estimer la probabilité d'un *hit* pour l'accès à une référence mémoire. Plusieurs estimateurs ont été étudiés [Zho10, KAQC13a] mais celui qui surestime le moins la probabilité cherchée est celui de [AD14]. L'estimateur $\hat{P}^M(k)$ proposé de la probabilité que le prochain accès mémoire à la référence M soit un *hit* est une fonction de l'associativité N du cache et de la distance de réutilisation k (de B_1 à B_k) de la référence M telle que :

$$\hat{P}^M(k) = \begin{cases} \left(\frac{N-1}{N}\right)^k & N > k, \\ 0 & otherwise. \end{cases} \tag{2.2}$$

Connaissant les probabilités de *hit* après estimation pour chaque accès mémoire et les latences d'accès à la mémoire cache, on peut construire l'ETP de la tâche considérée par convolution et déduire son pWCET. La convolution des ETPs est garantie par l'indépendance entre les blocs de base puisque l'architecture considérée contient des caches randomisés. L'approche SPTA considérée ne prend en compte que le cas d'une tâche avec un seul chemin et requiert la connaissance de la trace des accès mémoires.

Des travaux plus récents ont étudié l'approche SPTA dans le cas de tâches avec plusieurs chemins [LGAD15].

Ces approches sont également utilisées pour considérer les fautes matérielles qui peuvent survenir dans les mémoires caches [CSHB16]. Par exemple, dans des applications spatiales, les radiations peuvent provoquer des fautes mémoires. Ces fautes arrivent de manière aléatoire avec une probabilité d'occurrence donnée et impactent négativement le temps d'exécution de la tâche puisqu'elles suppriment un bloc dans la mémoire cache. Dans [CSHB16], les auteurs considèrent un processeur aléatoire monocœur affecté par des radiations. Les auteurs utilisent les chaines de Markov pour modéliser les accès à la mémoire cache en prenant en compte les probabilités de faute et les probabilités de *hit*. Les auteurs ont pu reconstituer le pWCET de la tâche à partir de la chaine de Markov grâce à la connaissance des temps d'accès au cache en fonction d'un *hit* ou d'un *miss*.

2.3 Discussion sur les méthodes d'estimation du WCET

Au fil de l'évolution des architectures matérielles, les méthodes d'estimation du WCET ont été adaptées pour pallier aux nouvelles complexités matérielles. Avec les contraintes industrielles de production, de compétitivité et de vente, les processeurs optimisent en moyenne le temps d'exécution au prix du déterminisme temporel et rendent les estimations du WCET très complexes. Certains travaux développent des architectures maison plus déterministes et qui vont faciliter l'estimation du WCET [SAA+15]. Ces développements restent coûteux et dégradent la vitesse d'exécution moyenne des tâches. A l'inverse, les autres plateformes achetées dans le commerce sont des architectures sur étagère (COTS Commercial Off The Shelf).

C'est pourquoi le problème de l'estimation du WCET peut être vu comme un non problème [Pus02]. En effet, si les fabricants de processeur privilégiaient le déterminisme temporel et moins l'optimisation temporelle, alors l'estimation du WCET en serait facilité. Néanmoins, les architectures d'ordinateur privilégiant le déterminisme temporel se font rares afin de répondre à la demande plus importante de processeurs plus rapides en moyenne.

Par conséquent, l'étude de nouvelles méthodes d'estimation du WCET reste donc indispensable si l'on souhaite utiliser des architectures modernes COTS dans des applications embarquées à moindre coût. Néanmoins, les méthodes développées restent souvent ségréguées selon l'approche initiale considérée (SDTA, MBDTA, MBPTA, SPTA). De plus, chaque approche demeure particulière du niveau de complexité de l'architecture considérée : les approches statiques pour des processeurs simples et les approches basées mesures pour des processeurs complexes. Enfin, il est nécessaire d'étudier des méthodes qui permettent de tirer parti de l'optimisation en moyenne du temps d'exécution qu'autorisent de telles architectures. Les approches MBPTA basées sur l'EVT semblent constituer une option réaliste pour répondre à ces constats. Ces approches sont basées mesures, elles sont donc simples et rapides à mettre en place. En particulier, elles ne nécessitent pas de modéliser l'architecture envisagée. De plus, l'utilisation des probabilités permet de faire un compromis entre performance et sûreté en fonction des exigences du système.

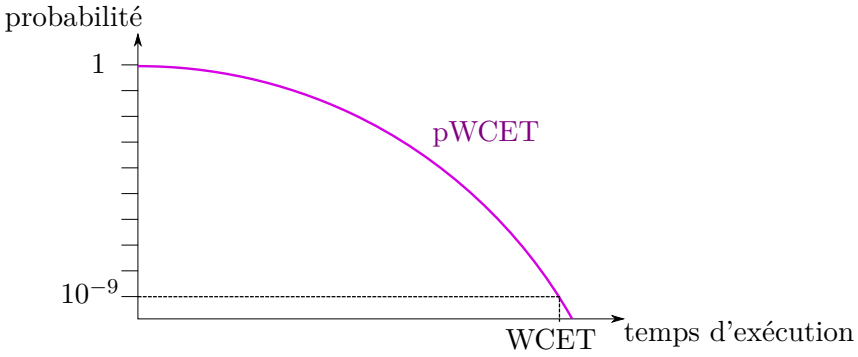


FIGURE 2.6 – Exemple de pWCET.

2.4 Détail des approches MBPTA pour l'estimation du pWCET

Différentes méthodes pour l'estimation du WCET ont été présentées dans la section précédente. Nous avons conclu sur une comparaison de ces méthodes par rapport aux objectifs de cette thèse qui portent sur l'estimation rapide, simple et fiable du WCET de tâches s'exécutant sur des processeurs COTS. Il en résulte que les méthodes probabilistes basées mesures permettraient de satisfaire ces exigences. Les approches MBPTA peuvent être basées soit sur l'EVT, soit sur la convolution. Les travaux de cette thèse portent sur les approches basées sur l'EVT car elles permettent de modéliser le comportement pire cas à partir de mesures bout en bout de la tâche. Le pWCET étant la pire distribution des temps d'exécution de la tâche, un WCET est déduit du pWCET en fonction d'une probabilité désirée e.g., 10^{-9}, comme dans la Figure 2.6.

Dans cette section, nous présentons les bases théoriques de l'EVT (Section 2.4.1), ainsi que les approches statistiques existantes qui en découlent (Section 2.4.3). Ensuite, les études existantes dans la littérature temps réel qui ont appliqué l'EVT pour le problème de l'estimation du pWCET et du WCET sont abordées (Section 2.4.4). Enfin, le cas particulier des architectures randomisées développées pour satisfaire les conditions de l'EVT est considéré (Section 2.4.5). En dernier lieu, nous conclurons sur les motivations pour cette thèse au regard de ces détails supplémentaires sur les approches MBPTA (Section 2.4.6).

2.4.1 La théorie des valeurs extrêmes

L'EVT est la théorie pour modéliser les évènements rares et prédire l'improbable. Elle a principalement été appliquée en ingénierie civile, en climatologie et en finance pour estimer la probabilité d'évènements qui requièrent des coûts d'expérimentation ou de simulation trop élevés [KPN02, Mul06]. Les assureurs l'utilisent par exemple pour estimer les plus grosses pertes qu'ils peuvent subir en vendant des assurances contre les catastrophes naturelles [Res97].

L'EVT considère la suite de réalisations $X_1, \ldots, X_n$ indépendantes et identiquement distribuées (iid) de la variable aléatoire $\mathcal{X}$. La définition d'une variable aléatoire est donnée dans la Définition 2.1.

Définition 2.1 (*Variable aléatoire*)
Une variable aléatoire à valeurs dans $\mathbb{R}$ ou une partie de $\mathbb{R}$ est une application définie sur l'ensemble des résultats possibles d'une expérience aléatoire.

— Les réalisations de la variable aléatoire sont *indépendantes* entre elles si la réalisation d'une n'a pas d'effet sur la réalisation de la seconde.
— La distribution de probabilité ou fonction de répartition F (Cumulative Distribution Function CDF) est une fonction d'une variable réelle x qui donne la probabilité que la variable aléatoire $\mathcal{X}$ soit inférieure à x i.e. $F(x) = P(\mathcal{X} \leq x)$. Les réalisations sont *identiquement distribuées* si elles sont issues d'une même distribution de probabilité F de la variable aléatoire $\mathcal{X}$.

Si la fonction de répartition F est continue alors la variable aléatoire $\mathcal{X}$ est continue sinon elle est discrète comme dans le cas de la Figure 2.5.

Pour caractériser le comportement pire cas, nous nous intéressons à la loi limite des maxima des variables aléatoires. Considérons donc la suite des maxima M_n telle que $M_1 = X_1$ et $M_n = \max(X_1, \ldots, X_n)$ pour $n \geq 2$. L'EVT s'intéresse aux lois limites qui modélisent la distribution de probabilité des maxima lorsqu'ils sont normalisés et centrés i.e. pour deux suites $c_n > 0$ et $d_n \in \mathbb{R}$.Cette démarche est analogue au cas du théorème central limite mais pour la moyenne des variables aléatoires. Concernant l'EVT, le théorème de Fisher-Tippett [EKM97] donne les lois limites des maxima.

Théorème 1 (Théorème de Fisher-Tippett)**.** *Considérons la suite de variables aléatoires iid $X_1, \ldots, X_n$ et la suite des maxima M_n. S'il existe deux suites de normalisation $c_n > 0$ et $d_n \in \mathbb{R}$ et une distribution non dégénérée H_ξ telles que :*

$$\frac{M_n - d_n}{c_n} \xrightarrow{d} H_\xi(x), \tag{2.3}$$

alors H_ξ est une distribution d'extremum généralisée (GEV) parmi les distributions suivantes :

— *Fréchet :*

$$H_\xi(x) = \begin{cases} 0 & x \leq 0 \\ \exp\left(-x^{-1\backslash\xi}\right) & x > 0 \end{cases} \tag{2.4}$$

— *Weibull :*

$$H_\xi(x) = \begin{cases} \exp\left(-(-x)^{1\backslash\xi}\right) & x \leq 0 \\ 1 & x > 0 \end{cases} \tag{2.5}$$

— *Gumbel :*

$$H_\xi(x) = \exp\left(-e^{(-x)}\right) \; x \in \mathbb{R}, \tag{2.6}$$

où $\xi > 0$.

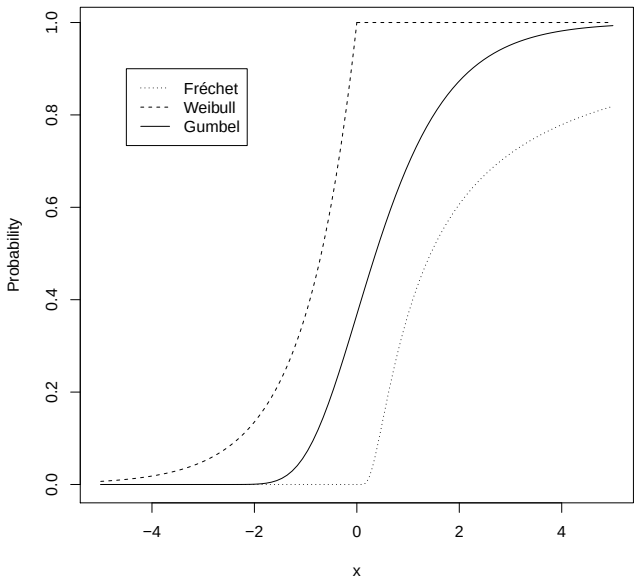


FIGURE 2.7 – Tracé des distributions GEV pour x autour de 0, et pour $\xi = 1$.

L'équation (2.3) signifie que la suite des maxima centrés et normalisés tend vers la loi limite H_ξ en distribution ($\xrightarrow{d}$) à mesure que n tend vers l'infini. Cette loi limite H_ξ est l'une des trois lois de la famille des distributions GEV. Les lois GEV ont des comportements différents sur l'ensemble des réels en fonction du paramètre ξ. En particulier, elles ne convergent pas à la même vitesse vers 1 quand x tend vers l'infinie par valeurs positives. La distribution de Weibull converge et atteint 1 en 0, alors que les deux autres distributions tendent asymptotiquement vers 1 quand x tend vers l'infini par valeurs positives. Dans le cas d'une distribution de Fréchet, plus la valeur de ξ est élevée plus la convergence est lente. En conséquence, le paramètre ξ est appelé paramètre de forme. Le comportement de ces lois en fonction de différents paramètres de forme ξ sont représentés dans la Figure 2.7.

De plus, si le théorème de Fisher-Tippett est vérifié pour des variables aléatoires issues de la fonction de répartition F, i.e. s'il existe des constantes de normalisation telles que la suite de maxima tende en distribution vers une loi GEV H_ξ, alors F est dans le domaine d'attraction maximum de H_ξ que l'on note $F \in MDA(H_\xi)$. On peut citer quelques fonctions F [EKM97] qui répondent à ce critère. Par exemple, les lois de Pareto et de Cauchy sont dans le domaine d'attraction de la distribution de Fréchet. La loi uniforme sur l'ensemble $(0, 1)$ est dans le domaine d'attraction de la distribution de Weibull. La loi exponentielle et la loi normale sont dans le domaine d'attraction de la distribution de Gumbel.

2.4.2 Equivalence par passage aux pics de dépassement

Ces distributions ont en commun un comportement particulier au niveau de leurs valeurs extrêmes. Ces valeurs extrêmes constituent la queue de distribution de la distribution considérée qui caractérise notamment le comportement extrême et même pire cas d'un système [PM08]. La valeur pire cas de la distribution F est donnée par le point de terminaison à droite $x_F = \sup\{x \in \mathbb{R} : F(x) < 1\}$ qui peut être fini ou non.

La queue de distribution est donnée par les valeurs extrêmes de la distribution initiale i.e. celles qui sont au dessus d'un seuil élevé u relativement à l'ensemble des valeurs que peut prendre la variable aléatoire. $\mathcal{Y} = \mathcal{X} - u$ est la variable aléatoire des pics de $\mathcal{X}$ au dessus du seuil u. En ne gardant que les pics de dépassement du seuil u i.e. $\mathcal{X} > u$, on obtient la fonction de répartition des dépassements F_u définie comme :

$$F_u(x) \overset{def}{=} P(\mathcal{Y} = \mathcal{X} - u \leq x | \mathcal{X} > u) = \frac{F(x+u) - F(u)}{1 - F(x)}, \tag{2.7}$$

pour $0 < x < x_F - u$. Le théorème de Pickands [EKM97] fournit alors l'équivalence suivante :

$$\lim_{u \to x_F} \sup_{0<x<x_F-u} \mid F_u(x) - P_\xi(x) \mid = 0 \longleftrightarrow F \in MDA(H_\xi), \tag{2.8}$$

où P_ξ est la distribution de Pareto généralisée (GPD Generalized Pareto Distribution).

Définition 2.2 (*Distribution de Pareto généralisée*)
La fonction de répartition de $\mathcal{Y}$ au dessus du seuil u conditionnellement à $\mathcal{X} > u$ suit une GPD P_ξ :

$$P_\xi(y) = \begin{cases} 1 - (1 + \xi y/\alpha_u)^{-1/\xi} & \xi \neq 0 \\ 1 - \exp(-y/\alpha_u) & \xi = 0 \end{cases} \tag{2.9}$$

avec α_u un facteur d'échelle, et telle que $\{y \in \mathbb{R},\ 1 + \xi y/\alpha_u > 0\}$.

La variable aléatoire $\mathcal{X}$ est distribuée selon F avec $F \in MDA(H_\xi)$ ssi la fonction de répartition des seuls pics $\mathcal{Y}$ suit une GPD P_ξ. Il y a bien un lien direct entre la distribution des maxima d'une variable aléatoire et la distribution de ses pics de dépassement. En partculier, pour des paramètres déterminés formellement, la valeur du paramètre de forme ξ reste constante par passage aux pics.

2.4.3 Application de l'EVT pour l'estimation du pWCET

Il existe deux approches statistiques pour appliquer l'EVT dans des études pratiques comme la climatologie, la finance ou ici pour l'estimation du pWCET. Ces deux approches sont l'approche des maxima par bloc (Block Maxima BM) et l'approche par les dépassements d'un seuil élevé (Peaks Over Threshold POT) [EKM97]. Les approches BM et POT sont respectivement basées sur le théorème de Fisher-Tippett et le théorème de Pickands.

Dans les deux cas, une suite de mesures $X_1, \ldots, X_n$ iid du paramètre d'intérêt est considérée, ici le temps d'exécution de la tâche étudiée.

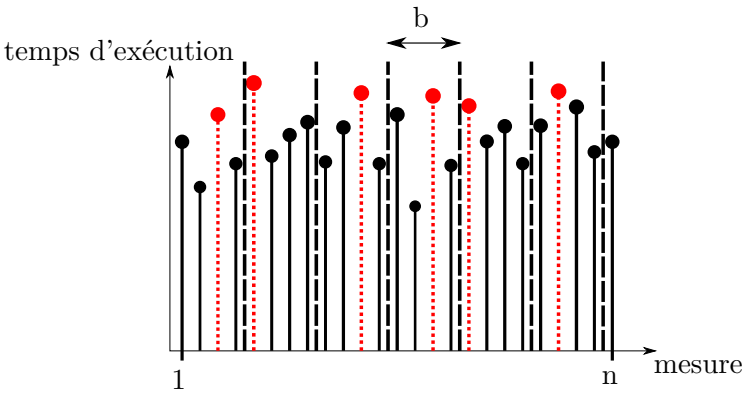


FIGURE 2.8 – Division de la suite de mesures en blocs de taille b, ici égale à 4, selon l'approche BM. Les blocs maxima sont les mesures indiquées en pointillés rouges.

Approche BM L'approche BM considère la division de la suite de mesures en blocs de taille b. La suite $X_1, \ldots, X_n$ est divisée en B blocs, du premier bloc de $X_1, \ldots, X_b$ au dernier bloc $X_{(B-1)b+1}, \ldots, X_n$. La taille de bloc b est une période de mesure. Ainsi, dans les études climatologiques, on considère une période temporelle, comme une journée, ou bien une année. On extrait de chaque bloc sa valeur maximale donnant la suite des maxima $M_1, \ldots, M_B$. La suite expérimentale des maxima obtenue ne permet pas d'observer toutes les valeurs possibles issues de la distribution théorique des maxima. Seule la loi limite théorique des maxima permet d'inférer l'ensemble des maxima possibles. Or, le théorème de Fisher-Tippett donne la loi limite des maxima qui est une distribution GEV de la forme :

$$H_\xi(x) = \begin{cases} \exp\left(-\left(1+\xi\frac{x-\mu}{\sigma}\right)^{-1/\xi}\right) & \xi \neq 0 \\ \exp\left(-\exp\left(-\frac{x-\mu}{\sigma}\right)\right) & \xi = 0, \end{cases} \tag{2.10}$$

définie pour $\{x \in \mathbb{R}, 1+\xi\frac{x-\mu}{\sigma} > 0\}$ l'ensemble des maxima, $\xi \in \mathbb{R}$ le paramètre de forme, $\sigma > 0$ le paramètre d'échelle des maxima, et $\mu \in \mathbb{R}$ la paramètre de localisation des maxima. Dans ce cas, si $\xi < 0$ il s'agit d'une distribution de Weibull, si $\xi > 0$ c'est une distribution de Fréchet, enfin si $\xi = 0$ alors c'est une distribution de Gumbel. Les paramètres $\xi,\ \mu,\ \sigma$ sont estimés tels que la loi théorique corresponde au mieux à la distribution expérimentale de la suite des maxima $M_1, \ldots, M_B$. Dans ce but, on utilise des algorithmes comme l'estimateur du maximum de vraisemblance (Maximum Likelihood Estimator MLE) [Smi85] ou celui des moments pondérés [HW87]. La distribution GEV dont on a estimé les paramètres est le pWCET cherché. L'approche BM est résumée dans la Figure 2.8.

L'approche BM est la plus largement utilisée dans les approches MBPTA [HHM09]

Approche POT L'approche POT considère les pics de la suite de mesures au dessus d'un seuil élevé u. Les p pics $Y_1, \ldots, Y_p$ sont extraits de la suite de mesures $X_1, \ldots, X_n$.

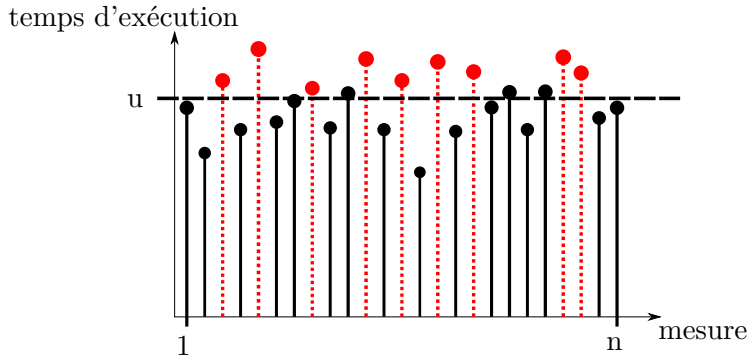


FIGURE 2.9 – Extraction des pics au dessus du seuil u indiqués en pointillés rouges selon l'approche POT.

D'après le théorème de Pickands, les pics sont distribués selon une GPD définie dans l'équation 5.1. Les paramètres ξ et α_u de la GPD sont estimés telle que la distribution théorique corresponde au mieux à la distribution expérimentale des pics $Y_1, \ldots, Y_p$ comme dans l'approche BM. La GPD dont on a estimé les paramètres est le pWCET cherché. L'approche POT est résumée dans la Figure 2.9.

Les deux approches sont comparées dans la Table 2.2.

approche	BM	POT
distributions	distributions GEV	GPD
	3 paramètres inconnus (ξ, μ, σ)	2 paramètres inconnus $(\xi, \alpha_u (= \mu - \xi(u - \sigma)))$
données modélisées	maxima par bloc	pics
paramètre	taille de bloc b	seuil u
applications	mesures périodiques	performances des systèmes

TABLE 2.2 – Comparaison entre les deux approches statistiques basées sur l'EVT.

2.4.4 Evaluation de l'estimation du pWCET

Chacune des deux approches décrites ci-avant fournissent un pWCET sur la base d'une suite de mesures du temps d'exécution de la tâche étudiée. Néanmoins, la mesure sur des plateformes réelles et l'application de l'EVT sur ces mesures peut être discutable. En effet, l'EVT requiert en théorie trois hypothèses pour être appliquée.

Dans un premier temps, les mesures doivent être iid i.e. indépendantes les unes des autres et identiquement distribuées selon une même distribution F. L'indépendance des mesures est difficile à prouver pour des applications pratiques. Il est impossible de prouver théoriquement que les mesures sont identiquement distribuées puisque la distribution théorique initiale F est inconnue et ne peut être estimée seulement à partir des mesures.

Dans un deuxième temps, la distribution théorique initiale F des temps d'exécution doit être dans le MDA d'une distribution GEV pour garantir que la loi limite des maxima des temps d'exécution soit une distribution GEV. Dans ce cas encore, la distribution F étant inconnu on ne peut prouver théoriquement cette hypothèse.

Si les hypothèses :

-h_1 **:** mesures indépendantes.

-h_2 **:** mesures identiquement distribuées.

-h_3 **:** mesures dans le MDA d'une distribution GEV.

sont remplies, l'application de l'EVT sur les mesures considérées pour l'estimation du pWCET est fiable.

La plupart des travaux consacrés aux approches MBPTA cherchent à appliquer des tests statistiques i.e. basés sur les mesures, pour évaluer la fiabilité du pWCET estimé. Les différentes approches considérées pour appliquer l'EVT dans le but d'estimer le pWCET d'une tâche sont référencées dans les Tables 2.3, 2.4, 2.5.

	Hypothèses		
	indépendance	identiquement distribués	MDA
EDGAR 2000, [BE00]	non mentionnée	non mentionnée	non mentionnée
EDGAR 2001, [EB01]	assumée	assumée	non mentionnée
HANSEN 2009, [HHM09]	assumée	assumée	non mentionnée
LU 2011, [LNBCG11]	*indépendance artificielle* : entrées de la tâche aléatoires extraction des maxima par bloc bootstrapping	assumée	non mentionnée
CUCU 2012, [CGSH+12]	réinitialise états matériel runs test	test de Kolmogorov-Smirnov	non mentionnée
LU 2012, [LNBC12]	[LNBCG11]	assumée	non mentionnée
LIU 2013, [LBN13b]	[LNBCG11]	assumée	non mentionnée
LIU 2013, [LBN13a]	BM distribués selon $H_{\xi'}$	assumée	non mentionnée
BEREZOVSKYI 2014, [BSBT14]	*dépendance à court terme* : runs test *indépendance des extrêmes* : extrêmogramme blocks test	test de Kolmogorov-Smirnov *cas stationnaire* : autocorrélation modèle autorégressif test Ljung-Box	non mentionnée
SANTINELLI 2014, [SMDJ14]	[BSBT14]	[BSBT14]	assumée
ABELLA 2015, [AdCPC15]	test de Ljung-Box	assumée	non mentionnée

TABLE 2.3 – Hypothèses requises pour appliquer l'EVT.

Ces tables décrivent de manière chronologique l'évolution de l'utilisation de l'EVT dans les approches MBPTA. Alors que l'application de l'EVT à une suite de mesures est directe et simple, garantir les résultats qui en découlent n'est pas trivial dans le cas

	Application			
	approche	distribution	estimation des paramètres	paramètre de forme ξ
EDGAR 2000, [BE00]	toute la suite	GEV	MLE	$\xi \in \mathbb{R}$
EDGAR 2001, [EB01]	toute la suite	GEV	moments	$\xi = 0$
HANSEN 2009, [HHM09]	BM	GEV	régression linéaire (QQ-plot)	$\xi = 0$
LU 2011, [LNBCG11]	BM	GEV	régression linéaire (QQ-plot)	$\xi = 0$
CUCU 2012, [CGSH$^+$12]	BM	GEV	régression linéaire (QQ-plot)	$\xi = 0$
LU 2012, [LNBC12]	BM	GEV	algorithme de recherche	$\xi = 0$
LIU 2013, [LBN13b]	BM	GEV	régression linéaire (QQ-plot)	$\xi = 0$
LIU 2013, [LBN13a]	POT	$P_{\xi'}$	non mentionnée	$\xi \in \mathbb{R}$
BEREZOVSKYI 2014, [BSBT14]	BM	GEV	MLE	$\xi = 0$
SANTINELLI 2014, [SMDJ14]	BM & POT	GEV & GPD	MLE	$\xi = 0$ & $\xi \in \mathbb{R}$
ABELLA 2015, [AdCPC15]	POT	GPD	MLE	$\xi = 0$

TABLE 2.4 – Application de l'EVT à une suite de mesures.

d'applications pratiques comme ici sur des systèmes temps réel. Cette difficulté est notamment due à l'application d'une théorie mathématique au cadre applicatif de mesures sur des systèmes temps réel. De plus, l'existence de plusieurs tests statistiques concernant la même hypothèse, la difficulté d'interpréter les résultats de ces tests et un manque de systématisme pour appliquer l'EVT rendent l'évaluation des pWCETs estimés par l'EVT très complexe.

Historiquement, l'EVT dans les approches MBPTA a longtemps considéré l'approche BM en contraignant la valeur du paramètre de forme ξ à 0. La théorie a d'abord été considérée selon l'approche BM par [BE00] pour estimer un WCET relativement à une probabilité de dépassement. Dans cette étude, les auteurs cherchent à capturer la variabilité du temps d'exécution due à la complexité des composants constitutifs des processeurs sur lesquels ils exécutaient des tâches. La distribution empirique des temps d'exécution exhibe une queue de distribution longue et suggère l'utilisation d'une loi de Gumbel pour modéliser les mesures de temps d'exécution. En sélectionnant l'approche BM, il est possible d'appliquer l'estimateur MLE pour déterminer les valeurs des deux paramètres restant (μ et σ) de la distribution GEV au lieu de trois. L'estimation de trois paramètres reste difficile [Hir96]. Ainsi, la distribution de Gumbel et l'approche BM ont été utilisées dans de nombreuses études par la suite [HHM09, CGSH$^+$12, LBN13b, BSBT14]. L'approche POT n'a été que très peu usitée pour l'estimation du pWCET [LBN13a, SMDJ14, AdCPC15]. [SMDJ14] compare les deux approches en utilisant différentes tailles de bloc et différentes valeurs de seuil. La contrainte de la distribution Gumbel dans le cas BM rend la comparaison difficile puisque le paramètre de forme ξ n'est pas contraint dans le cas POT.

Néanmoins, les estimations de WCET fournies par l'application de l'EVT manquent

	Evaluation		
	nombre de mesures	fiabilité	nombre de maxima
EDGAR 2000, [BE00]	non mentionné	graphique	toute la suite
EDGAR 2001, [EB01]	100	test du χ^2	toute la suite
HANSEN 2009, [HHM09]	50000	test du χ^2	$\geq 30 \cap$ χ^2-test
LU 2011, [LNBCG11]	non mentionné	test du χ^2	$\geq 30 \cap$ χ^2-test
CUCU 2012, [CGSH$^+$12]	critère de convergence	test de la queue exponentielle	non mentionné
LU 2012, [LNBC12]	précision requise niveau de confiance (confidence level cl) degré de variabilité	test du χ^2 probabilité divisée par les cls bootstrapping	$\geq 30 \cap$ χ^2-test
LIU 2013, [LBN13b]	cl marge d'erreur	test du χ^2 produit des cls	$\geq 30 \cap$ χ^2-test
LIU 2013, [LBN13a]	cl	test du χ^2	$\geq 30 \cap$ χ^2-test stabilité du seuil paramètre d'échelle modifié
BEREZOVSKYI 2014, [BSBT14]	non mentionné	tests statistiques	$b = 25$
SANTINELLI 2014, [SMDJ14]	non mentionné	tests statistiques erreur relative	différentes tailles de bloc
ABELLA 2015, [AdCPC15]	1000	tests statistiques	CV-plot statistique de contraste

Table 2.5 – Evaluation de la fiabilité des pWCET estimés par l'EVT.

de fiabilité. En effet, l'application de l'EVT à une suite de mesures retourne toujours, avec un nombre suffisant de mesures et une variabilité suffisante, un pWCET i.e. une distribution GEV qui modélise les BM. Cependant, cette modélisation peut être erronnée car l'application de l'EVT peut ne pas être acceptable dans certains cas de mesure. Autrement dit, il s'agit de vérifier que la distribution des mesures du temps d'exécution est dans le MDA de cette distribution GEV. Ainsi, les estimations de WCET à partir d'un tel pWCET sont inutilisables. Ce cas a bien été repéré par [EB01] en appliquant le test d'adéquation des mesures au modèle théorique du χ^2. [HHM09] intègre le test du χ^2 dans une boucle d'itération sur la taille de bloc pour explorer différentes tailles de BM afin d'obtenir des BM distribués en adéquation avec la distribution de Gumbel i.e. qui vérifient le test du χ^2. La plupart des travaux suivant s'efforcent de poursuivre la vérification de l'adéquation du pWCET estimé avec les mesures selon le test du χ^2 [LNBC12, LBN13b]. [CGSH$^+$12] propose d'utiliser le test de la queue exponentielle (Exponantial Tail test) issu de [Gar02].

Il reste également difficile d'assumer l'hypothèse d'indépendance des mesures pour les architectures considérées. Garantir l'indépendance des mesures du temps d'exécution pour une plateforme COTS n'est pas immédiat. En effet, les exécutions de la tâche étudiée sont dépendantes de l'historique d'exécution et notamment des données qui sont

présentes dans les mémoires caches [CGSH+12]. [LNBCG11] propose des méthodes pour considérer des entrées aléatoires de la tâche lors des mesures et une randomisation de la suite de mesures pour satisfaire l'hypothèse. Randomiser les entrées provoque de la variabilité fonctionnelle et ne permet plus à certain degré d'approcher la variabilité due aux composants du processeur. Pour s'en assurer, il faudrait comparer le nombre d'entrées différentes et le nombre de temps d'exécution différents. Si l'ensemble des temps d'exécution est plus grand alors de la variabilité système a été capturée. Des opérations sur la suite de mesures peuvent mener à la perte de mesures et donc d'informations qui pourrait donner des estimations non sûres. [CGSH+12] réinitialise les états matériels avant chaque nouvelle mesure et applique le *runs test* pour vérifier l'indépendance des mesures. La réinitialisation des états matériels est une condition de mesures particulière et peut être même du pire cas. Les auteurs proposent également de vérifier que les mesures sont identiquement distribuées avec le test de Kolmogorov-Smirnov. Cependant, sans la connaissance de la vraie distribution initiale des temps d'exécution la tentative de vérifier cette hypothèse reste hasardeuse. [BSBT14] introduit une batterie de tests statistiques pour vérifier l'indépendance et la distribution identique des mesures.

Dans ces travaux, la conclusion à propos de la fiabilité des estimations, évaluées avec les tests statistiques présentés reste floue. De plus, chaque étude, pour la plupart, fait usage de tests différents et les approches proposées demeurent isolées par rapport à leurs auteurs. Ainsi, il devient difficile d'exhiber une démarche systématique de l'évaluation du pWCET estimé.

En statistiques, l'hypothèse d'indépendance des mesures, et leur degré variabilité, est primordiale pour appliquer une modélisation comme celle basée sur l'EVT. Cependant, les architectures d'ordinateur sont construites de manière à fournir un comportement déterministe de ses composants. Considérant une tâche logicielle comme la suite de ces instructions, pour une même entrée de la tâche, la suite de ces instructions reste la même. Ainsi, pour des exécutions dont les états des composants restent les mêmes, la suite d'instructions devrait s'opérer de la même manière pour toutes les exécutions. En pratique, dans certains cas, le temps d'exécution mesuré montre une certaine variabilité mais qui peut ne pas être suffisante pour appliquer une modélisation statistique. En conséquence, plusieurs travaux visent à randomiser les architectures matérielles pour que les mesures de temps d'exécution soient indépendantes.

2.4.5 Randomiser le matériel

Cette approche a pour objectif de rendre les architectures matérielles directement compatibles avec l'application de l'EVT en cassant la dépendance des exécutions. L'approche a été largement développée dans le cadre du projet PROXIMA [DVA+14] qui étudie l'utilisation des approches MBPTA dans le cadre du développement de systèmes temps réel à criticité mixte. L'étude porte notamment sur la réduction des dépendances dans des architectures multicœur.

Une première étude de la randomisation des architectures a été réalisée dans [WKL+13] pour un système avionique industriel. Dans le but d'appliquer l'EVT et de garantir l'hypothèse d'indépendance, les auteurs modifient les politiques des caches

de données et d'instructions conformément à l'étude de [KAQC13a]. Ce faisant, les auteurs garantissent la conformité de l'architecture avec les exigences de l'EVT. Ainsi, les auteurs estiment des pWCETs fiables au regard des conditions de l'EVT. Néanmoins, cette randomisation est au prix d'une légère perte de performance en temps d'exécution moyen.

De ces travaux, et toujours dans le cadre du projet PROXIMA, s'ensuivent plusieurs études qui visent à formaliser et à garantir les estimations du pWCET sur ce type d'architectures randomisées. En particulier, les composants qui peuvent être randomisés sont étudiés afin d'assurer la conformité de la plateforme avec l'approche MBPTA [KQA+14]. La randomisation a débuté avec la modification des mémoires caches [KAQC13b, SKA+14, MZV+15]. Des travaux portent également sur la modification des pipelines d'instructions [KVAF+13]. Enfin, la randomisation a aussi été étudiée au niveau des bus d'accès mémoire [JKA+14, PAH+15].

De plus, il faut garantir l'exploration de tous les états des composants pour inférer l'improbable avec l'EVT, avec des architectures randomisées. Ce sont les problèmes de couverture et de représentativité des résultats produits par les approches MBPTA. La couverture des résultats caractérise la garantie d'avoir exploré ou inféré tous les états matériels. La représentativité des résultats évalue les conditions d'exécution qui vont mener à un pWCET confiant, i.e. un pWCET qui modélise effectivement le pire cas. Dans [CVQA13], les auteurs introduisent la notion de représentativité et énumèrent les différents niveaux impactant la confiance dans le pWCET. Ces différents niveaux sont les entrées de la tâche, les états de tâche et ceux du matériel. Le but étant de satisfaire l'exigence iid et de fournir les mesures les plus élevées pour garantir la représentativité du pWCET estimé avec l'EVT. Les auteurs décrivent les sources de variabilité du temps d'exécution et particulièrement celles qu'il est possible de modifier pour fournir une *bonne* suite de mesures pour l'EVT. Dans [AQW+14], une méthode pour modifier la taille des mémoires caches est proposée pour favoriser l'apparition d'évènements rares et ainsi augmenter la confiance dans le pWCET estimé. Enfin, dans [ZMV+15], une méthode pour garantir la couverture des chemins de la tâche est étudiée dans le cas de mémoires caches avec une politique de remplacement aléatoire. Cette méthode a été appliquée dans [MFB+17].

2.4.6 Motivations pour cette thèse

L'objectif de cette étude est l'estimation du WCET pour des applications temps réel s'exécutant sur des plateformes COTS. La conclusion sur les méthodes d'estimation du WCET existantes a été de considérer, dans cet objectif, les approches MBPTA. Basées mesures, les approches MBPTA évitent de devoir modéliser la plateforme COTS, étape très chronophage et voire même impossible pour des raisons industrielles. De plus, les plateformes COTS cherchent à optimiser en moyenne le temps d'exécution au détriment de la difficulté d'estimer le WCET de la tâche étudiée. Les approches probabilistes permettent de faire un compromis entre sûreté et performance en fonction de la probabilité désirée du WCET en estimant un pWCET.

Depuis une quinzaine d'années maintenant, plusieurs approches MBPTA basées sur

l'EVT ont été étudiées. Alors que l'application de l'EVT, selon l'approche BM ou POT, est plutôt directe, l'évaluation des estimations qu'elle fournit est moins triviale dans le cas d'une suite de mesures sur des plateformes COTS. Beaucoup de tests pour valider l'application de l'EVT ont été considérés mais les approches proposées manquent souvent de systématisme et ne concluent pas toujours clairement sur la fiabilité finale du pWCET estimé.

D'autres études ont cherché à randomiser le matériel pour appliquer directement l'EVT, mais ces plateformes ne sont pas vraiment représentatives de celles utilisées dans l'industrie et obligent à modifier le matériel.

En conséquence, dans cette thèse nous nous efforçons de rendre systématique l'application de l'EVT pour l'estimation du pWCET. En particulier, nous souhaitons faire en sorte que l'évaluation des estimations doit être plus facile et directe à interpréter. Ainsi que pour toutes les méthodes d'estimation du WCET, la méthode proposée doit être sûre. Dans le cas de l'application de l'EVT, les WCETs inférés à partir du pWCET estimé sont sûrs d'un point de vue statistique si les conditions d'application de l'EVT sont respectés et si le pWCET estimé modélise bien les mesures de temps d'exécution. Comme nous considérons des plateformes COTS nous n'allons pas concentrer nos efforts sur les architectures randomisées mais plutôt étudier en quoi les plateformes COTS, peuvent être, à un certain degré, compatibles avec les approches MBPTA. Enfin, nous étudierons les moyens d'obtenir un pWCET représentatif du pire cas ou du moins avoir une approche empirique du pire cas au moyen d'heuristiques.

Troisième partie

Matériels et Méthodes

Chapitre 3

Pire temps d'exécution probabiliste basé mesures

Nous abordons dans ce chapitre le cadre de travail sur lequel repose l'estimation du pWCET selon les approches MBPTA. L'analyse aborde les méthodologies associées à la mesure du temps d'exécution (Section 3.1). La mesure peut être réalisée tant bien sur des processeurs réels que sur la base de simulations. Nous nous intéressons particulièrement à la mesure sur des architectures réelles et explicitons les difficultés associées. Nous définissons les notations et formalismes utilisés dans le reste de cette thèse. L'approche MBPTA proposée est basée sur la collection de mesures du temps d'exécution appelée trace (Section 3.2). A partir de cette trace on peut construire le profil de temps d'exécution de la tâche étudiée (Section 3.3). Enfin, nous nous intéressons au pWCET de la tâche considérée et ses propriétés (Section 3.4).

3.1 Mesure du temps d'exécution

Plusieurs méthodes sont envisageables quant à la mesure du temps d'exécution d'une tâche logicielle. Deux méthodes sont principalement considérées :

-directe : la tâche mesurée intègre le code utile à la mesure.

-indirecte : le temps d'exécution de la tâche est mesuré lors de son exécution à l'aide d'une application tierce.

La méthode directe a pour intérêt de pouvoir sélectionner précisément la portion de code à mesurer dans la tâche. La méthode indirecte permet de ne pas intégrer des éléments de code non nécessaires à l'exécution de la tâche. Chaque méthode retourne un temps d'exécution de la tâche mesurée C. Nous n'avons pas considéré les méthodes de mesures physiques électroniques généralement réalisées à l'aide d'un oscilloscope.

Exemple 3.1 (*Méthode de mesure directe*)
Dans cet exemple on souhaite mesurer le temps d'exécution de la fonction C foo().

```
time=rdtsc_64bit();
result=args->foo();
time=rdtsc_64bit()-time;
```

Dans l'exemple de code C ci-dessus, le compteur de temps RDTSC est un registre 64 bits présent sur tous les processeurs x86 et permet d'accéder à l'horloge CPU avec un très faible coût temporel. Pour obtenir des résultats fiables dans le cas de processeurs multicœur, il faut forcer l'exécution de la tâche sur un seul cœur. A la fin de l'exécution de la tâche à mesurer, on appelle de nouveau le compteur de temps pour y soustraire le résultat du premier appel afin d'obtenir le temps d'exécution time.

Dans les deux cas, l'exécution de la tâche au moment de la mesure est *différente* de l'exécution au moment de son déploiement sur le système final. En effet, que ce soit du code intégré dans la tâche mesurée ou bien une application de mesure externe qui s'exécute en parallèle de la tâche mesurée, le code de mesure entraîne des états S_m des composants du processeur différents des états S_d en absence d'outil de mesure i.e. au déploiement. En général, la réalisation d'une mesure aura tendance à augmenter le temps d'exécution de la tâche mesurée car on fait appel à davantage de ressources et l'exécution de la tâche mesurée est perturbée en amont. En notant le temps d'exécution de la tâche mesurée C_m (respectivement C_d) provenant de l'état S_m (respectivement S_d), on a intuitivement

$$C_m > C_d. \tag{3.1}$$

L'inégalité précédente peut être démontrée en comparant une estimation statique du temps d'exécution de la tâche mesurée avec le code de mesure, avec celle du temps d'exécution de la tâche mesurée sans le code de mesure.

Exemple 3.2 (*Gestion du surcoût temporel*)
En reprenant l'exemple 3.1, il est possible de gérer le surcoût temporel de la mesure comme ci-après :

```
time=rdtsc_64bit();
result=args->foo();
time=rdtsc_64bit()-time;
time=time-2*overhead;
```

On soustrait du temps d'exécution les deux surcoûts temporels dûs à l'appel du compteur RDTSC. Le surcout temporel est contenu dans la variable overhead. *La valeur de l'*overhead *a été calculée au préalable en estimant le temps moyen d'appel du compteur RDTSC. En pratique, on mesure la durée moyenne de la fonction* rdtsc_64bit() *pour plusieurs exécutions de celle-ci.*

S'il s'avère que le surcoût temporel est négligeable devant le temps d'exécution de la tâche à mesurer, il est possible de ne pas le considérer [Hil02]. Plusieurs outils existant permettent de mesurer les performances temporelles des tâches logicielles. Parmi ces outils, on peut citer LTTng [FDD09] et PAPI [BDG+00]. Ces outils sont plus ou moins faciles à intégrer et permettent d'accéder à différents registres comme les compteurs de cache, de prédiction de branchements, etc... Il est nécessaire de quantifier le surcout temporel que ces outils introduisent pour évaluer le rapport entre les possibilités qu'offre l'outil considéré et la dégradation des performances qu'il engendre. Dans notre cas, le code utilisé dans l'exemple 3.2 a été suffisant pour mesurer le temps d'exécution de certaines tâches logicielles alors que nous avons utilisé LTTng pour une étude de l'impact des mémoires caches sur le pWCET [GSM16b].

En conclusion, la mesure du temps d'exécution de tâches logicielles amenées à être intégrées dans une application temps réel n'est pas triviale [Hil02]. La mesure engendre un surcout temporel et une exécution de la tâche mesurée différente de son exécution lorsqu'elle sera déployée.

Les temps d'exécution de tâches logicielles sont mesurés en cycles CPU ou en unité de temps. Si des unités de temps sont utilisés, les temps d'exécution de tâche d'application temps ont généralement des ordres de grandeur allant de la nanoseconde à la milliseconde. Dans le cas d'applications robotiques, il est possible de mesurer des temps d'exécution de l'ordre de la seconde. C'est le cas notamment d'algorithmes de navigation autonome telle que ceux de recherche de positionnement dans l'espace. Dans cette thèse, les temps d'exécution sont majoritairement mesurés en cycles CPU, les temps d'exécution sont donc des valeurs entières.

3.2 Trace de temps d'exécution

Si la mesure du temps d'exécution ne varie pas pour des conditions d'exécution données S, alors le processeur étudié est qualifié de temporellement déterministe. Les paramètres d'entrée doivent demeurer inchangés car les opérations arithmétiques peuvent prendre des temps d'exécution différents selon les valeurs considérées, comme dans le cas d'applications multimédia ou de traitement du signal [HKA+01].

Exemple 3.3 (*Exemples de conditions d'exécution*)
Les paramètres d'entrée de la tâche constituent autant de conditions d'exécution que le nombre de valeurs que peuvent prendre ces paramètres.

Dans le cas d'une architecture multicœur, les tâches pouvant s'exécuter en parallèle de la tâche à mesurer, sont autant de conditions d'exécution qu'il y a de combinaisons d'exécution des tâches en parallèle.

La Définition 3.1 introduit la notion de déterminisme temporel associée à un processeur.

Définition 3.1 (*Déterminisme temporel*)
Un processeur est temporellement déterministe si pour toute tâche logicielle s'exécutant sur ce même processeur, toutes les mesures du temps d'exécution d'une tâche réalisées dans les mêmes conditions S donne la même valeur C. Un processeur temporellement déterministe est noté Φ_D sinon il est noté $\Phi_{\bar{D}}$ un processeur temporellement non déterministe.

Or, dans le cas des processeurs COTS, ces derniers sont temporellement non-déterministes [EB01, M.13, WTM13, NYP16]. Les mécanismes d'optimisation des processeurs COTS permettent de réduire en moyenne le temps d'exécution d'une tâche mais introduisent une variabilité temporelle et donc du non déterminisme temporel. L'utilisation croissante de tels processeurs COTS pour les systèmes temps réel justifie cette thèse.

Ainsi, pour une tâche à étudier s'exécutant sur un processeur temporellement non-déterministe, l'approche des pires temps d'exécution probabilistes basés mesures (MBPTA : Measurement-Based Probabilistic Timing Analysis) considère des suites de mesures du temps d'exécution de la tâche à mesurer appelées *trace*.

Définition 3.2 (*Trace de temps d'exécution*)
Une trace de temps d'exécution $\mathcal{T}$ est une suite de mesures du temps d'exécution de la tâche τ s'exécutant sur le processeur $\Phi_{\bar{D}}$ dans les conditions S. Pour une suite de n mesures alors

$$\mathcal{T} = (C_i, i \in [\![1; n]\!]). \tag{3.2}$$

La longueur d'une trace (ainsi que pour toute liste ou ensemble) est donnée par l'opérateur $|.|$ tel que dans ce cas $|\mathcal{T}| = n$. Dans le cas de mesures sur un processeur Φ_D, la trace $\mathcal{T}$ est constante égale à C_1 pour tout i.

Exemple 3.4
La Figure 3.1 montre une trace de temps d'exécution contenant 66 mesures. Cette trace résulte de la mesure d'une tâche provenant d'une application robotique avec des contraintes temps réel [GGD+16, Gob16]. La trace contient des temps d'exécution différents pour chaque instant de mesure. On en conclut que le processeur faisant tourner l'application temps réel est temporellement non-déterministe.

Définition 3.3 (*Trace de temps d'exécution ordonnée croissante*)
La trace de temps d'exécution ordonnée croissante $\mathcal{T}^{\uparrow}$ de la trace $\mathcal{T}$ est définie comme la suite des temps d'exécution ordonnés par valeurs croissantes :

$$\mathcal{T}^{\uparrow} = (C_{(k)} \in \mathcal{T}, k \in [\![1; N]\!]) \tag{3.3}$$

et telle que

$$C_{(1)} < \ldots < C_{(N)}.$$

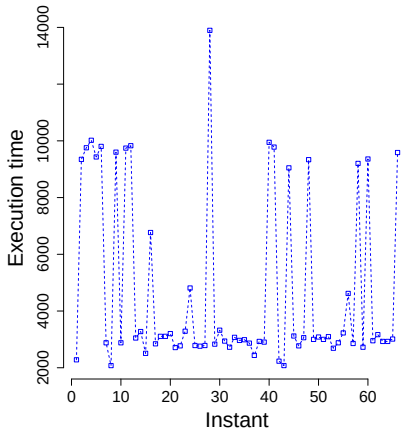


FIGURE 3.1 – Trace de temps d'exécution.

Dans d'autres domaines, une trace est aussi appelée série temporelle. Elle est ainsi dénommée car c'est une suite de mesures dans le temps.

Définition 3.4 (*Ensemble des temps d'exécution d'une trace*)
L'ensemble des temps d'exécution est l'ensemble des valeurs prises par le temps d'exécution de la tâche mesurée contenues dans la trace $\mathcal{T}$ et est noté $\Omega_{\mathcal{T}}$ tel que

$$\Omega_{\mathcal{T}} = \left\{ C_{(k)}, k \in [\![1; N]\!] \right\}. \tag{3.4}$$

3.3 Modélisation probabiliste

Dans le cas d'un processeur temporellement non-déterministe $\Phi_{\bar{D}}$, le cardinal de l'ensemble des temps d'exécution d'une tâche $\Omega_{\mathcal{T}}$ est différente de 1. Comme le montre la Figure 3.1, le temps d'exécution d'une tâche peut prendre différentes valeurs et ce de manière incertaine. Nous modélisons la variabilité du temps d'exécution avec des probabilités. Au lieu de considérer un temps d'exécution, une variable aléatoire sera définie pour représenter le comportement temporel de la tâche mesurée.

L'utilisation de variables aléatoires dans les systèmes temps réel apparaît dans [M.13]. Dans [M.13], la notion de système temps réel probabiliste réfère à l'existence d'au moins un paramètre du système défini par une variable aléatoire. Ce paramètre peut être le temps d'exécution, les temps d'arrivée ou les dates d'échéance. Dans notre cas, on utilise la notion de variable aléatoire pour rendre particulièrement compte du temps d'exécution de tâches s'exécutant sur des processeurs définis comme temporellement non-déterministes.

Une variable aléatoire peut être à valeurs discrètes ou continues. Dans le cas d'une variable aléatoire discrète, cette dernière est définie sur un ensemble discret. La variable aléatoire qui va représenter le comportement temporel de la tâche mesurée est l'ETP de

la tâche, noté $\mathcal{C}$.

Définition 3.5 (*Profil de temps d'exécution*)
Le profil de temps d'exécution $\mathcal{C}$ d'une tâche mesurée est la variable aléatoire définie sur l'ensemble des valeurs de temps d'exécution $\Omega_{\mathcal{T}}$.

En pratique, une mesure du temps d'exécution d'une tâche s'exécutant sur un processeur temporellement non-déterministe retourne de manière aléatoire une valeur de temps d'exécution. On modélise donc le temps d'exécution de la tâche mesurée par une variable aléatoire définie sur l'ensemble des temps d'exécution possibles. En outre, une variable aléatoire associe à chaque valeur possible une probabilité selon une loi ou distribution de probabilité. La distribution de probabilité d'occurrence sur l'ensemble des valeurs de temps d'exécution du profil de temps d'exécution est donnée par la distribution de probabilité empirique associée.

Définition 3.6 (*Distribution de probabilité empirique*)
La distribution de probabilité empirique du profil de temps d'exécution $\mathcal{C}$ est la fonction $f_{n,\mathcal{C}}$, n étant le nombre de mesures dans $\mathcal{T}$, définie sur $\Omega_{\mathcal{T}}$ vers les probabilité d'occurrence de chaque élément de $\Omega_{\mathcal{T}}$:

$$f_{n,\mathcal{C}} = \begin{pmatrix} C_{(1)} & \dots & C_{(N)} \\ f_1 & \dots & f_N \end{pmatrix}, \tag{3.5}$$

avec f_i la probabilité d'occurrence de la valeur de temps d'exécution $C_{(i)}$, $\forall i \in [\![1; N]\!]$.

Ainsi que le profil de temps d'exécution est relatif à une trace $\mathcal{T}$, les probabilités d'occurrence découlent de $\mathcal{T}$. Une estimation de la probabilité d'occurrence d'une valeur de temps d'exécution correspond au nombre de fois que cette valeur est dans $\mathcal{T}$ divisée par $|\mathcal{T}|$. Ainsi, $\forall i \in [\![1; N]\!]$

$$f_{n,\mathcal{C}}\left(C_{(i)}\right) = f_i = \frac{1}{|\mathcal{T}|} \sum_{1 \leq j \leq |\mathcal{T}|} \mathbf{1}_{C_j = C_{(i)}} = P\left(\mathcal{C} = C_{(i)}\right) \in [0; 1], \tag{3.6}$$

avec $\mathbf{1}_{C_j = C_{(i)}}$ la fonction indicatrice qui vaut 1 si C_j est égal à $C_{(i)}$ sinon 0. En outre, la somme des probabilités d'occurrence vaut 1, $\sum_{i=1}^{N} f_i = 1$.

Remarque 3.1
Il est possible de grouper les valeurs de temps d'exécution. Au lieu de déterminer la probabilité d'occurrence d'une valeur on estime le nombre de mesures qui appartiennent à chaque groupe. Les groupes de temps d'exécution sont des intervalles de taille fixe.

Exemple 3.5
La trace de l'exemple 3.4 donne l'ETP de la Figure 3.2. Pour plus de lisibilité, le profil de temps d'exécution de la figure 3.2 donne les probabilités d'occurrence (Probability mass) par groupe de valeurs de temps d'exécution. Dans cet exemple, chaque groupe est de taille 2000. Par exemple, une estimation de f_1 à partir de cette trace de mesure vaut $f_1 = \frac{1}{66}\sum_{j\leq 66} \mathbf{1}_{2000\leq C_j<4000}$. Graphiquement, le profil de temps d'exécution nous renseigne sur le comportement temporel moyen de la tâche. Ainsi, le temps d'exécution de la tâche est majoritairement entre 2000 et 4000 cycles CPU. Il se peut que le temps d'exécution de la tâche mesurée diverge vers 14000 cycles CPU.

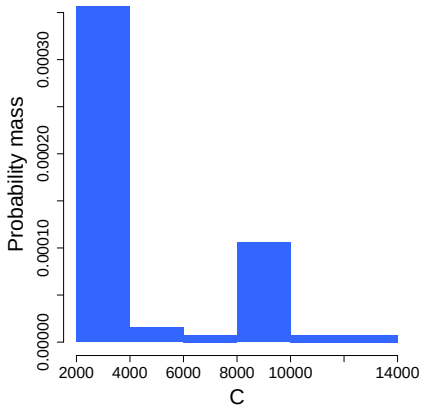


FIGURE 3.2 – Profil de temps d'exécution.

De manière équivalente, on peut s'intéresser à la probabilité de ne pas dépasser un temps d'exécution avec la distribution cumulative empirique $F_{n,\mathcal{C}}$. Inversement, la probabilité de dépassement d'un temps d'exécution est donnée par la distribution cumulative inverse empirique de fonction $\bar{F}_{n,\mathcal{C}}$ (Inverse Cumulative Distribution Function ICDF). L'indice renvoie à la taille de la trace $\mathcal{T}$ considérée et ainsi à la notion d'empirique contrairement aux fonctions théoriques. Une estimation pour chacune de ces probabilités pour une trace $\mathcal{T}$ est $\forall i \in [\![1; N]\!]$

$$F_{n,\mathcal{C}}\left(C_{(i)}\right) = \frac{1}{|\mathcal{T}|}\sum_{1\leq j\leq|\mathcal{T}|} \mathbf{1}_{C_j\leq C_{(i)}} = P\left(\mathcal{C}\leq C_{(i)}\right) \in [0;1], \tag{3.7}$$

$$\bar{F}_{n,\mathcal{C}}\left(C_{(i)}\right) = 1 - F_{\mathcal{C}}\left(C_{(i)}\right) = \frac{1}{|\mathcal{T}|}\sum_{1\leq j\leq|\mathcal{T}|} \mathbf{1}_{C_j>C_{(i)}} = P\left(\mathcal{C}> C_{(i)}\right) \in [0;1]. \tag{3.8}$$

La probabilité de dépassement d'un temps d'exécution est souvent considérée comme la confiance sur le choix de ce temps d'exécution comme borne supérieure. En effet, si l'on choisit un temps d'exécution $C_{(m)}$ dans l'ensemble $\Omega_{\mathcal{T}}$ comme borne supérieure sur le temps d'exécution de la tâche mesurée alors avec $\bar{F}_{\mathcal{C}}\left(C_{(m)}\right) = 0.1$, par exemple, il

y a une chance sur dix de dépasser cette borne supérieure. Il est donc très dangereux de prendre $C_{(m)}$ comme borne supérieure sur la variable aléatoire $\mathcal{C}$. Dans la suite on se référera principalement à la distribution cumulative inverse. Par ailleurs, on rappelle que la fonction de distribution cumulative est strictement croissante sur son ensemble de définition et la fonction de distribution cumulative inverse est strictement décroissante sur son ensemble de définition.

3.4 Pire temps d'exécution probabiliste

Le modèle probabiliste sera également appliqué au pire temps d'exécution. Initialement, le pire temps d'exécution est une borne supérieure déterministe sur le temps d'exécution de la tâche étudiée. Le pire temps d'exécution ne peut donc être qu'une unique valeur et non une distribution de valeurs.

A une trace de temps d'exécution $\mathcal{T}$ correspond un ensemble de valeurs de temps d'exécution $\Omega_{\mathcal{T}}$. Cet ensemble n'est généralement pas exhaustif lorsque les mesures sont réalisées sur un processeur temporellement non-déterministe i.e. il est impossible de vérifier que tous les temps d'exécution possibles aient été mesurés. D'une part, les mesures dépendent des conditions d'exécution S. Pour différentes conditions S_1 et S_2 on pourrait avoir deux ensembles différents. Dans un deuxième temps, pour les mêmes conditions d'exécution, il reste difficile de prouver l'exhaustivité des états des composants du processeur expérimentés. Dans la suite, on considérera toujours les même conditions d'exécution, et telles que l'ensemble de valeurs de temps d'exécution exhaustif correspondant serait donné par Ω^* [1] et tel que $\Omega_{\mathcal{T}} \subset \Omega^*$. Le profil de temps d'exécution $\mathcal{C}$ défini sur l'ensemble exhaustif Ω^* est distribué selon la densité de probabilité théorique continue $f_{\mathcal{C}}$ et telle que $f_{n,\mathcal{C}} \underset{n\infty}{\longrightarrow} f_{\mathcal{C}}$ i.e. pour une infinité de mesures n.

Les systèmes temps réel s'intéressent aux valeurs extrêmes de manière supérieure du temps d'exécution d'une tâche. Il s'agit donc d'un sous ensemble de Ω^* que l'on noterait Ω^{*+} [2]. Il est possible d'admettre, pour le moment, que les valeurs de Ω^{*+} correspondent à des évènements dont la probabilité d'occurrence est très faible par rapport aux valeurs en dehors de Ω^{*+}. En interprétant physiquement ce phénomène, on peut dire qu'il est de plus en plus difficile, et donc plus rare, d'obtenir une mesure qui dépasse les mesures déjà réalisées.

En probabilités, ces valeurs correspondent aux quantiles extrêmes d'une distribution. Les quantiles sont donnés par la fonction quantile q définie sur l'ensemble des probabilités. Pour une probabilité p, $q(p)$ est la valeur de temps d'exécution telle que la probabilitité de ne pas dépasser cette valeur vaut p i.e. pour la variable aléatoire $\mathcal{C}$, on a donc $P(\mathcal{C} \leq q(p)) = p$. Les quantiles extrêmes sont les valeurs données par $q(1-\alpha)$ avec α qui tend vers zero. Ces valeurs correspondent au comportement pire cas de la tâche étudiée. Or, la mesure seule du temps d'exécution ne permet pas d'accéder de manière sûre au comportement pire cas de la tâche ou moyennant des mesures complexes et coûteuses en temps. L'estimation sûre du quantile à 10^{-9} par la seule mesure

1. L'exposant '*' ne réfère pas à l'ensemble privé de zéro dans ce cas.
2. L'exposant '+' ne désigne pas l'ensemble privé des éléments négatifs dans ce cas.

nécessiterait environ 10^{11} échantillons [RC13]. On considère donc une distribution de temps d'exécution définie sur l'ensemble Ω^{*+} et qui rend compte des quantiles extrêmes désignée par le pire temps d'exécution probabiliste noté $\mathcal{C}^*$. En particulier, le pire temps d'exécution probabiliste contient le pire temps d'exécution.

Définition 3.7 (*Pire temps d'exécution probabiliste*)
Le pire temps d'exécution probabiliste (pWCET : probabilistic Worst Case Execution Time) $\mathcal{C}^*$ est la variable aléatoire continue qui rend compte du comportement pire cas par valeurs supérieures de la tâche mesurée.

Tel qu'il est défini le pWCET correspond à une variable aléatoire qui possède une loi de probabilité et des propriétés.

Remarque 3.2 (*Loi de probabilité du pWCET*)
La loi de probabilité du pWCET est donnée par sa fonction de densité de probabilité $f_{\mathcal{C}^*}$. On note la distribution de probabilité cumulative du pWCET $F_{\mathcal{C}^*}$ et inverse $\bar{F}_{\mathcal{C}^*}$.

Pour aborder les propriétés du pWCET nous introduisons les notations de comparaison entre variables aléatoires [MNS].

Définition 3.8 (*Comparaison de variables aléatoires*)
Soient deux variables aléatoires $\mathcal{X}_1$ et $\mathcal{X}_2$ de distribution de probabilité cumulative f_1 et f_2 respectivement définies sur les ensemble Ω_1 et Ω_2. $\mathcal{X}_1$ est supérieure à $\mathcal{X}_2$ si $\forall x \in \Omega_1 \cap \Omega_2$ $F_1(x) \leq F_2(x)$ et est noté

$$\mathcal{X}_1 \succeq \mathcal{X}_2. \tag{3.9}$$

Proposition 3.1 (*Propriétés du pWCET*)
*Le pWCET d'une tâche est supérieur à toutes les variables aléatoires représentant le comportement temporel de cette même tâche i.e. pour toutes conditions d'exécution et pour tous les états du processeur considéré ; le pWCET est donc **sûr**.*

*Le pWCET d'une tâche représente exactement son comportement temporel pire cas par valeurs supérieures ; le pWCET est donc **précis**.*

En conséquence, le pWCET d'une tâche est le plus petit majorant sur l'ensemble des variables aléatoires caractérisant cette même tâche.

3.4.1 Estimation du pire temps d'exécution probabiliste

Le pWCET est un formalisme mathématique pour caractériser le comportement temporel pire cas d'une tâche en y associant des probabilités. En effet, le pWCET résulterait de la modélisation d'un nombre infini de mesures du temps d'exécution de la tâche

étudiée en considérant de surcroît les conditions qui favoriseraient les temps d'exécution extrêmes. De telles mesures sont trop coûteuses et sont irréalisables.

L'objectif de cette thèse est d'estimer le pWCET pour qu'il soit utilisé dans les analyses d'ordonnancement lors du développement de systèmes temps réel. On désire en outre que l'estimation du pWCET soit simple, rapide et fiable. On réfère au pWCET estimé par la variable aléatoire continue $\mathcal{C}^{\lambda}$, avec λ un vecteur de paramètres. Comme le pWCET, le pWCET estimé doit être sûr et précis. Cependant, comme le pWCET est le plus petit majorant sur l'ensemble des variables aléatoires caractérisant le temps d'exécution d'une tâche, alors le pWCET estimé ne peut lui être inférieur :

$$\mathcal{C}^{\lambda} \succeq \mathcal{C}^{*}. \tag{3.10}$$

3.4.2 Seuils de pire temps d'exécution

Le pWCET est une distribution de temps d'exécution qui représente le comportement temporel probabiliste pire cas de la tâche étudiée. Ainsi, le pWCET est la pire des distributions de temps d'exécution de la tâche étudiée qui associe aux valeurs de temps d'exécution sur lesquelles elle est définie une probabilité d'être dépassée, dans le cas de la distribution de probabilité cumulative inverse.

Majoritairement, les algorithmes d'ordonnancement ne se réfèrent qu'à une valeur de pire temps d'exécution par tâche composant l'application temps réel. Le pWCET ne peut donc être utilisé directement pour la conception d'un système temps réel. On définit à partir du pWCET, théorique et estimé, les seuils de pire temps d'exécution qui associent un pire temps d'exécution dans l'ensemble de définition du pWCET à sa probabilité de dépassement p : $\langle WCET; p\rangle$ (WCET seuil à p).

Pour de faibles probabilités d'être dépassé p, $p \to 0$, le pire temps d'exécution associé sera d'autant plus pessimiste. En effet, pour $\langle WCET_1; p_1\rangle$ et $\langle WCET_2; p_2\rangle$ tels que $WCET_1 > WCET_2$ alors $p_1 < p_2$. Il y a plus de confiance en $WCET_1$ en tant que borne supérieure sur le temps d'exécution de la tâche car sa probabilité d'être dépassée p_1 est plus faible, mais le pire temps d'exécution borné est plus pessimiste.

3.5 Discussion

On résume l'approche proposée des pires temps d'exécution probabilistes basés mesures par le diagramme en Figure 3.3. Pour un système temps réel donné composé de tâches logicielles exécutées sur un processeur, l'objectif est d'estimer le pWCET $\mathcal{C}^{\lambda}$ à partir d'une trace de mesures de temps d'exécution de la tâche étudiée $\mathcal{T}$. Des WCET seuils à p $\langle WCET; p\rangle$ sont inférés du pWCET pour être utilisés dans les analyses d'ordonnancement du système temps réel.

En comparaison avec les méthodes d'estimation existantes et spécialement statiques du pire temps d'exécution, l'approche proposée se veut simple, rapide, fiable et précise. L'approche s'intègre facilement dans un cycle de développement, ou de conception, d'un système temps réel, car elle considère directement le système. Outre le temps d'intégration de la tâche sur la plateforme considérée et le nombre de mesures à effectuer,

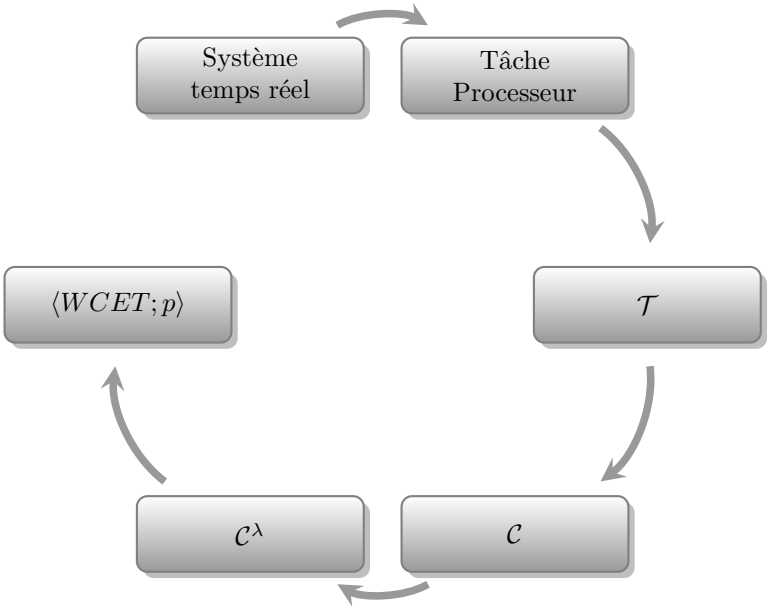


FIGURE 3.3 – Diagramme de l'approche des pires temps d'exécution probabilistes basés mesures.

l'approche est rapide. Pour garantir la fiabilité et la précision d'une telle approche, il reste à montrer comment nous pouvons approximer la queue de distribution empirique des temps d'exécution $f_{n,\mathcal{C}}$ par une fonction de distribution continue du pWCET $\mathcal{C}^\lambda$ i.e. $f_{\mathcal{C}^\lambda}$. Enfin, grâce aux mesures réalisées il est possible de reboucler entre le système temps réel et le couple tâche, processeur, pour améliorer les mesures relativement aux exigences des méthodes d'estimation du pWCET.

Chapitre 4

Architectures étudiées

Au cours de cette thèse, nous avons disposé de différentes architectures réelles COTS sur lesquelles nous avons réalisé des mesures et appliquer les analyses statistiques que nous proposons. Cela a permis de mettre en évidence la possibilité d'utiliser facilement et rapidement l'outil DIAGXTRM, développé durant la thèse, pour caractériser l'influence des différentes architectures différentes sur le temps d'exécution de tâches. Nous avons aussi pu observer certains cas limites. L'étape prenant le plus de temps est d'intégrer la tâche étudiée sur la plateforme considérée et de mettre en place un moyen de mesurer de la manière la moins intrusive possible son temps d'exécution. Dans cette partie, nous listerons les plateformes considérées dans cette étude. Nous détaillerons également les principales caractéristiques de chaque plateforme et expliquerons leur intérêt pour les systèmes temps réel. Les tâches logicielles que nous avons exécutées sur les architectures respectives seront présentées. Enfin, nous allons référencer les traces de mesure du temps d'exécution qui seront utilisées dans la présentation des contributions.

4.1 Système embarqué critique industriel

La première architecture considérée provient d'un cas industriel développé chez AIRBUS. Il s'agit d'un système embarqué avionique intégré sur une architecture multicœur devant répondre à des contraintes temps réel dur.

Un premier cas d'étude porte sur une tâche considérée comme longue et dont les entrées sont invariantes. La tâche mesurée est exécutée sur un cœur alors qu'un autre cœur exécute en parallèle des activités liées à l'Input/Output (I/O). Les performances de la tâche étudiée est contrainte par la DDR3 [1] SDRAM [2] alors que l'autre tâche responsable des I/Os présente une utilisation moindre de la DDR3 utilisant davantage les caches privées du cœur sur lequel la tâche est exécutée. En particulier, les activités I/Os incluent 100 Mbps de controlleurs Ethernet, le stockage et la récupération d'images dans

1. La DDR3 est un standard de mémoire vive.
2. La SDRAM est une mémoire vive (RAM) synchrone (S) dynamique (D).

et depuis la DDR3. Ainsi, la tâche exécutant des des activités I/Os interfère avec la tâche étudiée.

De même, dans un deuxième cas d'étude, la tâche mesurée est exécutée sur un cœur tandis qu'un deuxième cœur exécute des activités concurrentes. La tâche mesurée rapatrie du code et des données depuis la SDRAM. Les tâches en concurrence accèdent également à la SDRAM mais à des banques mémoires différentes. Dans le cas d'une mémoire vive dynamique, les données doivent être réactualisées par un rafraîchissement. Nous étudions donc ici l'influence du rafraîchissement de le SDRAM à l'aide de mesures.

Les traces suivantes sont choisies pour être étudiées dans cette thèse.

-trace1, trace2, trace3 : une trace de mesures réalisée selon le premier cas d'étude.

-trace4 : une trace de mesures réalisée selon le second cas d'étude sans rafraîchissement.

-trace5 : trace4 avec un offset aléatoire ajouté à toutes les mesures.

-trace6 : une trace de mesures réalisée selon le second cas d'étude avec rafraîchissement.

-trace7 : trace6 avec un offset aléatoire ajouté à toutes les mesures.

4.2 Architecture multicœur

Une autre architecture étudiée est une plateforme COTS composée de deux processeurs Intel®Xeon®E5620 chacun étant cadencé à 2.4 GHz. Chaque processeur est un multicœur contenant au total quatre cœurs ainsi que trois niveaux de cache. Les deux premiers niveaux (L1 et L2) sont privés à chaque cœur et le dernier niveau (LLC) est partagé entre tous les cœurs. L2 et LLC sont de taille respective 256 KB et 12 MB, et unifient données et instructions. Au contraire, L1 est partitionné entre le cache d'instructions et le cache de données, toutes deux de taille 32 KB. Chaque mémoire cache est 8-A associatif par ensemble avec une politique de remplacement dite pseudo-LRU. La politique pseudo-LRU est une politique propriétaire implémentée par le fabricant qui fonctionne sur la base de la politique LRU mais dont certains détails relèvent du secret industriel.

Sur la base de cette architecture, nous étudions les interférences dues aux exécutions de tâches en parallèle sur le multicœur. On considère les tâches du benchmark Mälardelen dans une exécution dualcœur i.e. la tâche mesurée s'exécute sur un cœur et une autre tâche s'exécute en parallèle sur un autre cœur du même processeur. En particulier, on considère les tâches *bs*, *ns*, *nsichneu* et *statemate*.

-*fibcall* **:** calcule la suite de Fibonacci.

-*bs* **:** effectue une recherche d'un élément dans une liste en divisant la liste successivement par deux. C'est une recherche binaire (binary search *bs*).

-*ns* **:** cherche un élément dans tableau de dimension quatre. La tâche itère selon quatre boucles *for* imbriquées. Pour simplifier, nous ne considérons que quatre éléments, ce qui donne quatre chemins possibles de temps d'exécution différents.

-*nsichneu* **:** simule un réseau de Petri. Cette tâche est intéressante car elle contient un nombre important d'instructions et fait beaucoup usage du cache d'instructions.

-*statemate* **:** simule une machine à états.

-*cnt* **:** compte les nombres non négatifs dans une matrice de taille paramétrable. La matrice peut remplir les caches selon sa taille.

Pour imposer de telles conditions expérimentales, les tâches sont écrites en langages C et compilées sous la forme d'une bibliothèque partagée. Le processeur tournant sous Linux, nous utilisons la programmation système C pour Linux. Nous avons développé un simple ordonnanceur qui exécute les tâches sur les cœurs spécifiés pendant un nombre donné de fois. Nous mesurons le temps d'exécution de la tâche qui nous intéresse grâce au registre *rdtsc* du cœur correspondant comme dans l'Exemple 3.1. Des mesures d'autres paramètres de performance ont été réalisées dans cette configuration, comme le nombre de *hits* et de *misses* au niveau du cache L1, grâce à l'outil LTTng. De plus, Linux n'étant pas un système opératoire temps réel, nous devons forcer l'exécution temps réel des tâches en imposant la politique d'ordonnancement Linux temps réel `SCHED_FIFO`. La politique `SCHED_FIFO` assure l'exécution temps réel des tâches puisqu'une tâche prête se met dans la file d'attente et lorsque le cœur est libre, alors la tâche peut s'exécuter. Nous avons également fait attention à limiter le nombre d'interruptions extérieures.

Les traces suivantes sont choisies pour être étudiées dans cette thèse. Elles sont constituées de mesures du temps d'exécution d'une tâche exécutée en parallèle de deux tâches *cnt*, chacune exécutée sur un cœur.

-trace8 : trace de mesures du temps d'exécution de la tâche *bs*.

-trace9 : trace de mesures du temps d'exécution de la tâche *ns*.

-trace10 : trace de mesures du temps d'exécution de la tâche *nsichneu*.

-trace11 : trace de mesures du temps d'exécution de la tâche *statemate*.

4.3 Système embarqué robotique

Dans le cadre de l'étude d'analyses temps réel dans le domaine de la robotique [GGD+16], nous avons eu l'opportunité d'analyser des traces de mesure du temps d'exécution de tâches provenant d'une architecture logicielle robotique. Les auteurs proposent un déploiement des tâches pour la réalisation de la mission, effectuent des mesures du temps d'exécution des tâches qui les intéressent, et utilisent une approche MBPTA pour estimer le WCET de chaque tâche. Connaissant le WCET de chaque tâche, les auteurs effectuent une analyse de faisabilité d'ordonnancement basée sur le temps de réponse. Si le déploiement proposé est faisable, il est conservé sinon les auteurs proposent un nouveau déploiement.

Le robot considéré dans cette étude est le Pioneer 3DX robot mobile de Adept Mobile Robots. Le robot intègre un processeur COTS Intel Core i5 contenant 4 cœurs cadencés

à 2,7 GHz. Le processeur fait tourner le système opératoire Linux Ubuntu 14.04 configuré pour utiliser l'ordonnanceur temps réel Linux `SCHED_FIFO` comme pour le processeur précédent. Les auteurs utilisent l'outil LTTng pour effectuer des mesures du temps d'exécution des tâches qui les intéressent. Dans ce contexte embarqué, la problématique de stockage des mesures est critique car la mémoire est limitée.

Les tâches mesurées par les auteurs correspondent aux tâches soumises à des contraintes temps réel et dont le WCET va servir à l'analyse d'ordonnançabilité. Les tâches font parties de composants de l'architecture robotique. On trouve les composants relatifs au contrôle du chemin à suivre comme *guidance, control, safety_switch*. Par exemple, la tâche en charge d'éviter des collisions, `avoid_collision`, fait partie du composant *guidance*.

Les traces suivantes sont choisies pour être étudiées dans la thèse.

-trace12 : trace de mesures du temps d'exécution d'une tâche du composant relative au *SLAM*, algorithme de cartographie.

-trace13 : trace de mesures du temps d'exécution d'une tâche du composant *navigation*.

-trace14 : trace de mesures du temps d'exécution d'une tâche du composant *mapping*.

-trace15 : trace de mesures du temps d'exécution de la tâche de la trace14 sans préchargement.

-trace16 : trace de mesures du temps d'exécution d'une tâche du composant relatif au driver $p3dx$.

-trace17 : trace de mesures du temps d'exécution d'une tâche du composant *exploration*.

4.4 Graphical Processor Unit (GPU)

Les GPUs sont des processeurs utilisés pour paralléliser massivement des applications. La majorité des applications concernent le traitement d'images. Les GPUs offrent une puissance de calcul immense et des modèles de programmation, comme le modèle Compute Unified Device Architecture (CUDA), qui facilitent le développement d'applications logicielles sur ce type de plateformes. En conséquence, les GPUs sont de plus en plus considérés pour des systèmes embarquées. Néanmoins, il existe peu de méthodes pour garantir la prédictibilité d'applications temps réel sur ce type de plateforme. L'auteur de [Ber16] étudient différentes méthodes d'estimation du WCET pour des tâches s'exécutant sur des GPUs. Une de ces méthodes est une approche MBPTA [BSBT14, BGS+16].

Les GPUs sont composés de plusieurs *Streaming Multiprocessor* (SM) ressemblant à des processeurs manycœur. Dans cette étude, on considère le NVIDIA Kepler GK104 avec 8 SMs et un cache partagé L2 de 1,5 MB. Chaque SM a 192 cœurs sous CUDA et une mémoire dédiée de 64 kB divisée en une mémoire partagée et un cache L1. Les programmes CUDA sont appelés noyaux. Les tâches sous CUDA sont regroupés dans

des blocs et chaque bloc est déployé sur un SM. Les blocs ne peuvent pas migrer entre des SMs. Le moteur CUDA assure un déploiement tel que les SMs restent occupés mais un tel déploiement n'est pas documenté.

Les auteurs effectuent des mesures de la tâche du générateur de diagramme de Voronoï. Le diagramme de Voronoï est un découpage du plan en cellules à partir d'un ensemble discret de points appelés germes. Chaque cellule ne renferme qu'un seul germe et forme l'ensemble des points les plus proches de ce germe. Le générateur considéré prend une matrice en entrée qui constitue le plan. Certains éléments de la matrice contiennent un germe et le générateur découpe la matrice comme décrit ci-avant. Pour cette étude, les auteurs font varier le nombre de blocs de tâches selon quatre scénarios : 64/256/512/1024 blocs. Comme le nombre de tâches ne varie pas, les quatre scénarios correspondent aux quatre scénarios : 1024/512/256/64 tâches par bloc. De plus, il est possible de faire varier la répartition de la mémoire dédiée de chaque SM divisée en une mémoire partagée et un cache privé L1. Trois options sont offertes : 25%/75%, 50%/50%, 75%/25%. La variation de ces deux paramètres permettent d'obtenir $4 \times 3 = 12$ scenarios de mesure.

Les traces suivantes sont choisies pour être étudiées dans cette thèse. Seul le cas de 64 blocs de tâches est considéré.

-trace18 : trace de mesures du temps d'exécution de la tâche étudiée dans le cas 25%/75%.

-trace19 : trace de mesures du temps d'exécution de la tâche étudiée dans le cas 50%/50%.

-trace20 : trace de mesures du temps d'exécution de la tâche étudiée dans le cas 75%/25%.

4.5 Field-Programmable Gate Array (FPGA)

Les FPGAs sont des processeurs qu'il est possible de "reprogrammer" au sens de la reconfiguration de ses composants logiques. A partir d'un modèle de processeur écrit en langage VHDL, la plateforme FPGA est flashée et présente les caractéristiques du modèle. Dans cette étude, on considère diverses configurations d'un multiprocesseur composé de deux processeurs LEON3. Le LEON3 est un processeur 32 bits cadencé à 50 MHz couramment utilisé dans des applications spatiales. Le modèle sera flashé sur deux plateformes comportant un FPGA Xilinx : la plateforme Gaisler GR-XC6S-LX75 et la Xilinx ML605. Le processeur possède un cache d'instructions et un cache de données séparés de 16 KB chacun.

Dans cette étude, l'impact de certains composants du processeur sur l'estimation du pWCET par les approches MBPTA est inspecté. Les mémoires caches, l'unité de prédiction de branchement, ou encore l'unité de calcul des nombres flottants (Floating Point Unit FPU) peut être activées ou non. Quatre politiques de remplacement sont possibles à la configuration du processeur : aléatoire, direct, le bloc le moins récemment remplacé (Least Recently Replaced LRR), le bloc le moins récemment utilisé (LRU). Lorsque l'unité de calcul en nombre flottant est désactivée, elle est alors simulée et

le calcul est plus long que pour une unité matérielle. En modifiant ces paramètres, 9 architectures différentes sont considérées sur la base des deux plateformes disponibles.

Chaque processeur est équipé d'une unité statistique appelée L3STAT composée de 8 compteurs. Un de ces compteurs permet de mesurer le temps d'exécution de la tâche sur le processeur correspondant. De nouveau, des tâches du benchmark Mälardelen sont utilisées :

-*cnt* et *matmult* : remplissent le cache de données. *matmult* opère une multiplication de matrices de taille paramétrable.

-*nsichneu* :

-$jfdcint$: ne sature aucune mémoire cache. La tâche opère une transformée en cosinus discrète sur un bloc de 8×8 pixels. La transformée en cosinus discrète est utilisée pour compresser des images. C'est une application typique d'un système embarqué.

-*lms* : est utilisée pour comparer les performances du FPU. La tâche opère un filtre basé sur la minimisation des moindres carrés pour reconnaître un signal sinusoïdal bruité.

Deux conditions d'exécution sont observées sur la base des 9 architectures proposées. La première exécute la tâche mesurée sans interférence i.e. aucune autre tâche ne s'exécute. La deuxième condition considère le système opératoire temps réel RTEMS pour exécuter la tâche mesurée en présence d'interférences. Des interférences intracœur et intercœur sont introduites en exécutant respectivement une boucle infinie de plus haute priorité que la tâche mesurée et la même boucle infinie sur le deuxième cœur de la plateforme.

Les trace suivantes sont choisies pour être étudiées dans cette thèse.

-trace21 : trace de mesures du temps d'exécution de la tâche *nsichneu* selon la deuxième condition d'exécution.

-trace22 : trace de mesures du temps d'exécution de la tâche $jfdctint$ selon la deuxième condition d'exécution.

On considère également des traces issues de la simulation du comportement d'une architecture randomisée. Pour ce faire, des temps d'exécution sont tirés de manière aléatoire depuis une fonction de répartition donnée. Les traces suivantes sont choisies pour être étudiées.

-trace23, trace24 : trace de temps d'exécution issus d'une loi normale.

-trace25 : trace de temps d'exécution issus d'une loi exponentielle.

4.6 Architecture manycœur

Les architectures manycœur ont déjà été abordées dans la Section 1.1. Les auteurs de [NYP16] étudient l'utilisation du Kalray MPPA-256, un processeur manycœur composé de 256 cœurs d'exécution. L'intérêt d'une telle architecture est de pouvoir paralléliser massivement des tâches comme les GPUs. Néanmoins la surutilisation de la

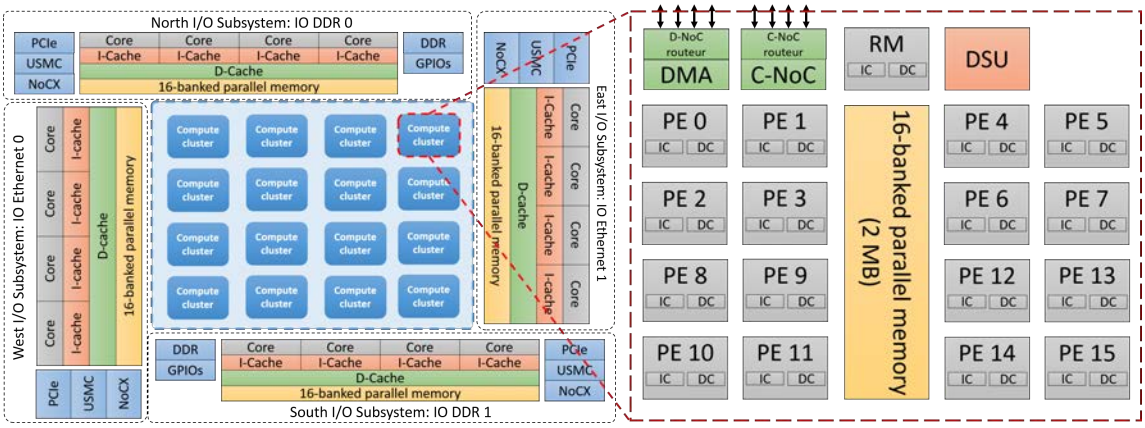


FIGURE 4.1 – Schéma de l'architecture du KALRAY MPPA-256.

plateforme peut mener à des interférences entre les tâches et les contraintes temps réel peuvent être mises à mal. Les cœurs sont répartis parmi 16 tuiles de 16 cœurs comme sur la Figure 4.1.

Chaque tuile possède une mémoire partagée de 2 MB et les cœurs ont un niveau de cache privé L1 partitionné entre un cache d'instructions et un cache de données. La mémoire partagée est divisée en 16 banques mémoires de 128 KB et telle que son espace d'adressage peut être divisé séquentiellement ou non. Ne pas diviser séquentiellement l'espace d'adressage, espace d'adressage "entrelacé", se révèle utile pour des applications hautement parallélisables. Les cœurs peuvent accéder à la mémoire partagée selon un tourniquet temporel (Round Robin RR) i.e. à chaque cœur est attribué un créneau pendant lequel il peut, et lui seul, accéder à la mémoire. Les tuiles sont reliées entre elles par un réseau (Network On Chip NoC). Le NoC permet d'échanger des données entre les tuiles ou d'accéder à des banques mémoires partagées entre les tuiles. L'intérêt des manycœur est de pouvoir expliciter lors de l'implémentation les accès au NoC et de pouvoir calculer les délais de traverser du NoC selon les méthodes de calcul réseau.

Les auteurs étudient les effets temporels de la parallélisation de tâches dans une tuile. Pour ce faire, les auteurs ont sélectionné 13 benchmarks issus de la suite TACLE-bench [TAC16]. Deux conditions d'exécution sont étudiées :

-mode isolation : la tâche benchmark étudiée est exécutée seulement sur un cœur.

-mode interférences : la tâche benchmark est exécutée sur un cœur en parallèle de 15 tâches identiques produisant des interférences (Interference Generator IG) au niveau du cache partagé de la tuile.

Chaque tâche IG initialise une liste de 1024 entiers, vide le cache de données privé puis lit chaque élément de la liste 8 par 8. La taille de 8 entiers correspond à la taille d'un bloc mémoire de telle manière que chaque lecture provoque un *miss*. Ainsi, la donnée demandée est chargée depuis la mémoire partagée créant du trafic sur le bus d'accès mémoire.

Les auteurs considèrent également un deuxième cas d'étude portant sur l'analyse temporelle d'un algorithme de traitement d'images exécuté sur le Kalray MPPA-256. L'intérêt est de pouvoir paralléliser les différentes opérations de l'algorithme. Néanmoins, la parallélisation introduit des interférences au niveau des ressources partagées entraînant la variabilité du temps d'exécution de l'algorithme. Cette variabilité rend difficile l'estimation du WCET de la tâche. L'algorithme étudié est une application de traitement d'images pour de la détection infrarouge développée par Airbus Defence and Space. L'application a pour objectif d'être embarquée dans la navette Euclid en cours de développement par l'Agence Spatiale Européenne (European Space Agency ESA). En résumé, la tâche prend en entrée une image infrarouge haute définition de l'espace et applique un algorithme de traitement composé de 7 étages. Le code est source est composé d'environ 2000 lignes de code C.

Les traces suivantes sont choisies pour être étudiées dans cette thèse.

-trace26 : trace de mesures de la tâche *statemate* réalisée selon le premier cas d'étude avec contentions et le caches privés activés.

-trace27 : même cas sans contention.

-trace28 : même cas avec contentions et caches privés désactivés.

-trace29 : meme cas sans contention et caches privés désactivés.

-trace30 : trace de mesures de la tâche *detectCosmicRay* du second cas d'étude.

-trace31 : trace de mesures de la tâche *detectSaturation* du second cas d'étude.

-trace32 : trace de mesures de la tâche *substractSuperBias* du second cas d'étude.

Quatrième partie

Contributions

Chapitre 5

Evaluation de la fiabilité du pWCET estimé à partir de l'application de l'EVT à une trace de mesures

Sommaire

Résumé 1
L'étude des méthodes d'estimation existantes du WCET a montré leur difficile généralisation et leur coût élevé notamment dans le cas de plateformes COTS. L'objectif dans ce chapitre est d'évaluer la fiabilité des pWCETs estimés à partir des approches MBPTA basées sur l'EVT. Cette évaluation doit être systématique, simple à interpréter, complète du point de vue des exigences de l'EVT et adaptée au cas de plateformes COTS. Ainsi, nous proposons un ensemble de conditions pour garantir l'estimation du pWCET. L'ensemble orgnisé des conditions sous forme de diagramme de décision est la base de l'outil développé de DIAGnostic de l'applicabilité de la théorie des valeurs eXTRêMes DIAGXTRM *qui permet d'évaluer la fiabilité des pWCETs estimés. Pour estimer de manière plus précise le pWCET, nous utilisons l'approche POT qui évite de contraindre le modèle au cas d'une loi de Gumbel. Néanmoins, l'estimation du pWCET reste sensible à certains paramètres notamment à cause du nombre limité de mesures. En conséquence, l'outil* DIAGXTRM *intègre des algorithmes pour encadrer ces incertitudes et évaluer la confiance dans l'estimation proposée.*

5.1 Application de l'approche POT pour l'estimation du pWCET dans le cas de mesures stationnaires

Dans la section 2.4.3, nous avons comparé les approches statistiques pour appliquer l'EVT à une trace de mesures. L'analyse bibliographique des approches MBPTA existantes montre que l'approche BM dans le cas de mesures iid a principalement été considérée pour l'estimation du pWCET. En conséquence, le pWCET a été modélisé par une distribution de Gumbel ($\xi = 0$) et le choix de la taille des blocs b reste arbitraire. Néanmoins, le cas iid n'est pas adapté au cas de plateformes COTS car l'indépendance stricte ne peut être justifié. En effet, les mesures ne sont pas strictement indépendantes car les mécanismes des mémoires caches entraînent des dépendances. Enfin, il n'est pas possible de prouver que les temps d'exécution sont tous issus d'une même distribution, puisque cette dernière est inconnue.

Afin de pouvoir considérer l'application de l'EVT à des traces de mesures, nous étudions l'approche POT dans le cas de mesures stationnaires. L'approche POT utilise

la GPD pour modéliser le pWCET. Ainsi que dans l'Equation 5.1, la GPD est déterminée par deux paramètres ξ et α_u à estimer contre trois dans le cas d'une distribution GEV si ξ n'est pas contraint à 0. De ce fait, nous pouvons utiliser des algorithmes courants d'estimation des paramètres pour estimer le pWCET sans avoir à contraindre le paramètre de forme ξ. Ainsi, le pWCET ne sera pas contraint au cas d'une distribution de Gumbel et pourra modéliser au mieux les mesures i.e. être plus précis. Le pWCET estimé permet d'inférer des WCETs seuil à une probabilité p tel que :

$$\langle WCET; p\rangle = \begin{cases} u + \frac{\alpha_u}{\xi}\left(\frac{n}{k}p^{-\xi} - 1\right) & \xi \neq 0 \\ u - \alpha_u \log\left(\frac{n}{k}p\right) & \xi = 0 \end{cases} \tag{5.1}$$

avec k le nombre de pics de dépassement au dessus de u.

Dans les approches MBPTA, l'EVT est appliquée à une trace de mesures ou série temporelle. L'aspect temporel est très important dans cette thèse car celui-ci nous permet de dégager des propriétés sur les mesures. En outre, l'application de l'EVT dans le cas d'une trace de mesures stationnaires est connu dans la littérature EVT mais n'a pas été considéré dans les estimations du pWCET. L'analyse requiert ainsi des mesures stationnaires répondant à deux conditions de dépendances liées à l'aspect temporel. Dans ce cadre, il n'est pas nécessaire de connaître la distribution initiale des temps d'exécution. De plus, les conditions de dépendances, que nous allons préciser, sont plus représentatives de mesures sur des plateformes COTS. En considérant un cadre d'analyse plus relâché, la sûreté des estimations s'en trouve renforcée.

5.1.1 Mesures stationnaires

L'hypothèse de stationnarité est essentielle et courante dans les analyses statistiques, mais elle est souvent admise. Il n'existe ainsi pas de manière systématique pour étudier la stationnarité de mesures. Celle-ci est souvent basée sur des analyses subjectives [MS96]. La définition de la stationnarité stricte d'une trace est la suivantes :

Définition 5.1 (*Trace strictement stationnaire*)
Une trace $\mathcal{T} = (C_1, C_2, \ldots)$ est strictement stationnaire si pour tout entier j, k, l, la suite de temps d'exécution $C_j, \ldots, C_{j+k}$ a la même distribution de probabilité que la suite $C_{l+j}, \ldots, C_{l+j+k}$.

Si les temps d'exécution de la trace $\mathcal{T}$ sont tels qu'ils respectent la Définition 5.1, alors les mesures sont identiquement distribuées selon une distribution à estimer.

Comme la distribution initiale théorique est la GPD continue, l'analyse de stationnarité pose le problème de la continuité des mesures. En effet, les distributions théoriques sont des lois continues sur leur ensemble de définition. Les mesures de temps d'exécution sont des multiples de l'horloge du cœur du processeur, ce sont donc des entiers. Pour des mesures *proches* les unes des autres, il est acceptable d'accepter la continuité. Néanmoins, pour des mesures éloignées, alors l'hypothèse de continuité n'est plus valable. Par exemple, pour des mesures d'une même trace issues d'une part d'un chemin

rapide et d'autre part d'un chemin lent, cela donnera un écart conséquent entre ces deux ensembles de mesures [GB10]. La continuité des mesures n'est donc pas acceptable dans un tel cas.

5.1.2 Dépendance à court terme

Une hypothèse de l'EVT est l'indépendance des mesures. Or dans un système réel, réaliser des mesures indépendantes entre elles est pratiquement impossible car il existe des phénomènes de dépendance à plus ou moins court terme. Par exemple, en climatologie, une journée de basse température sera probablement suivie d'une journée de basse température également car la masse atmosphérique évolue lentement. Dans le cas des systèmes temps réel, les composants impactant la dépendance sont les mémoires caches. La persistance des données utiles à une tâche dans les mémoires caches entraîne une dépendance entre les mesures des différentes exécutions de la tâche. Néanmoins pour deux mesures très éloignées dans le temps, on peut considérer, sous certaines conditions, que cette persistance n'a plus lieu d'être et que les mesures sont indépendantes.

Ce phénomène est formulé sous la notion de dépendance à court terme et se trouve formalisé sous la condition de Berman ou encore condition D [LLR78, Lea91] :

Définition 5.2 (*Condition* $D(u_n)$)
Pour tout entier m et n : $1 \leq i_1 < \ldots < i_m < j_1 < \ldots < j_m \leq n$ tel que $j_1 - i_m \geq l$, on a

$$|P\left(\{C_i, i \in A_1 \cup A_2\} \leq u_n\right) - P\left(\{C_i, i \in A_1\} \leq u_n\right) P\left(\{C_i, i \in A_2\} \leq u_n\right)| \leq \alpha_{n,l}, \tag{5.2}$$

où $A_1 = \{i_1, \ldots, i_m\}$, $A_2 = \{j_1, \ldots, j_m\}$ et $\alpha_{n,l} \to 0$ quand $n \to \infty$ pour une certaine suite $l = l_n = o(n)$.

En particulier, pour une suite u_n dans le théorème de Fisher-Tippett, des temps d'exécution issus de deux blocs A_1 et A_2 distants de l sont indépendants. Au lieu de chercher l'indépendance des mesures une à une, on va vérifier que les blocs de temps d'exécution de taille m sont indépendants.

5.1.3 Indépendance extrême

L'approche POT s'intéresse aux pics de temps d'exécution au dessus d'un seuil u. La condition D précédente concerne des blocs de mesures indépendants. L'important pour la modélisation POT est d'avoir des pics indépendants pour ne pas biaiser l'estimation. Par exemple, si un ensemble de pics sont proches dans le temps, ils sont probablement tous dûs à un même phénomène. Un tel phénomène peut être l'activation d'une tâche sur un cœur qui cherche à accéder à la mémoire partagée de la tâche étudiée créant du trafic et interférant avec la tâche mesurée. Les pics dans cette zone temporelle de la trace sont dépendants d'un même phénomène. Considérer l'ensemble des pics dans

l'approche POT biaise l'estimation des paramètres de la GPD. Une telle zone de pics de dépassement est appelée un cluster de pics de temps d'exécution.

En conséquence, une deuxième condition est nécessaire pour garantir l'application de l'EVT dans le cas stationnaire. Elle est notée condition D' [LLR78, Lea91] :

Définition 5.3 (*Condition* $D'(u_n)$)
La relation

$$\lim_{i\to\infty} \limsup_{n\to\infty} n \sum_{j=1}^{[n/i]} P(C_1 > u_n, C_j > u_n) = 0, \tag{5.3}$$

doit être vérifiée.

La Condition D' signifie que la probabilité que les temps d'exécution suivants un pic de temps d'exécution soient au-dessus du seuil spécifié doit tendre vers 0. Si la relation est vérifiée pour la trace de temps d'exécution, alors les pics au dessus du seuil sont considérés comme indépendants. En conséquence, le choix du seuil u est très important puisque pour un seuil trop bas, les pics peuvent se révéler trop nombreux et donc dépendants.

5.1.4 Adéquation du modèle

Si les conditions D et D' sont vérifiées pour une trace de temps d'exécution stationnaire, alors l'EVT peut être appliquée [Lea91, EKM97]. Ainsi, les paramètres de la GPD ξ et α_u, formant le couple de paramètres λ, sont estimés relativement aux pics de mesure de temps d'exécution au dessus du seuil u. La GPD estimée modélise le pWCET $\mathcal{C}^\lambda$ mais pour être fiable, il faut que la distribution des temps d'exécution $F_\mathcal{C}$ soit dans le MDA d'une distribution GEV. De manière équivalente, par le Théorème 2.2, la distribution expérimentale discrète des pics de temps d'exécution F_u doit tendre vers $F_{\mathcal{C}^\lambda}$. Une distribution discrète F_u est modélisée par une distribution continue $F_{\mathcal{C}^\lambda}$ afin d'inférer des temps d'exécution non mesurés car requérant des mesures trop coûteuses. Pour valider le modèle estimé, l'écart entre les distributions F_u et $F_{\mathcal{C}^\lambda}$est évalué en comparant pour tous les pics de dépassement la probabilité retournée par F_u et celle retournée par $F_{\mathcal{C}^\lambda}$. Si cet écart est proche de 0, on peut conclure que les pics de temps d'exécution suivent bien une GPD. Dans ce cas, le pWCET estimé est fiable par rapport aux exigences de l'EVT.

5.2 Diagramme de décision pour l'évaluation de la fiabilité de l'estimation du pWCET

L'enjeu principal des études MBPTA est de définir une approche systématique pour fournir des estimations fiables de pWCET. La fiabilité de l'approche repose sur la bonne identification et définition des hypothèses nécessaires à son application. Dans le cas de

l'estimation du pWCET, la fiabilité de ce dernier est garantie si chaque hypothèse de l'EVT est vérifiée pour la trace de mesures considérée.

L'étude du cas de mesures stationnaires révèle qu'il est possible d'appliquer l'EVT pour des conditions expérimentales dans le cas de plateformes COTS. Pour simplifier, nous référençons chaque hypothèse à vérifier. Les hypothèses à vérifier dans ce cas pour une trace de mesures sont :

- $h'_{1.1}$ mesures stationnaires.

- $h'_{2.1}$ indépendance à court terme.

- $h'_{1.2}$ pics de dépassement stationnaires.

- $h'_{2.2}$ indépendance extrême.

- h'_{3} adéquation du modèle aux mesures.

L'ensemble des hypothèses notées $h'_{1.*}$ porte sur la stationnarité des mesures i.e. au fait que les mesures soient identiquement distribuées (hypothèse classique h_1). L'ensemble des hypothèses notées $h'_{2.*}$ est relatif à l'indépendance des mesures (hypothèse classique h_2). Il est également intéressant de vérifier la stationnarité des pics, hypothèse $h'_{1.2}$, pour renforcer l'idée que les pics sont bien issus d'une même distribution qui serait la GPD estimée à partir de ces pics.

Pour systématiser l'approche MBPTA proposée, nous ordonnons les différentes étapes de diagnostic du pWCET estimé comme dans la Figure 5.1. Ce processus de diagnostic est à la base de l'outil de DIAGnostic de l'applicabilité de la théorie des valeurs eXTRêMes à une trace de mesures : DIAGXTRM.

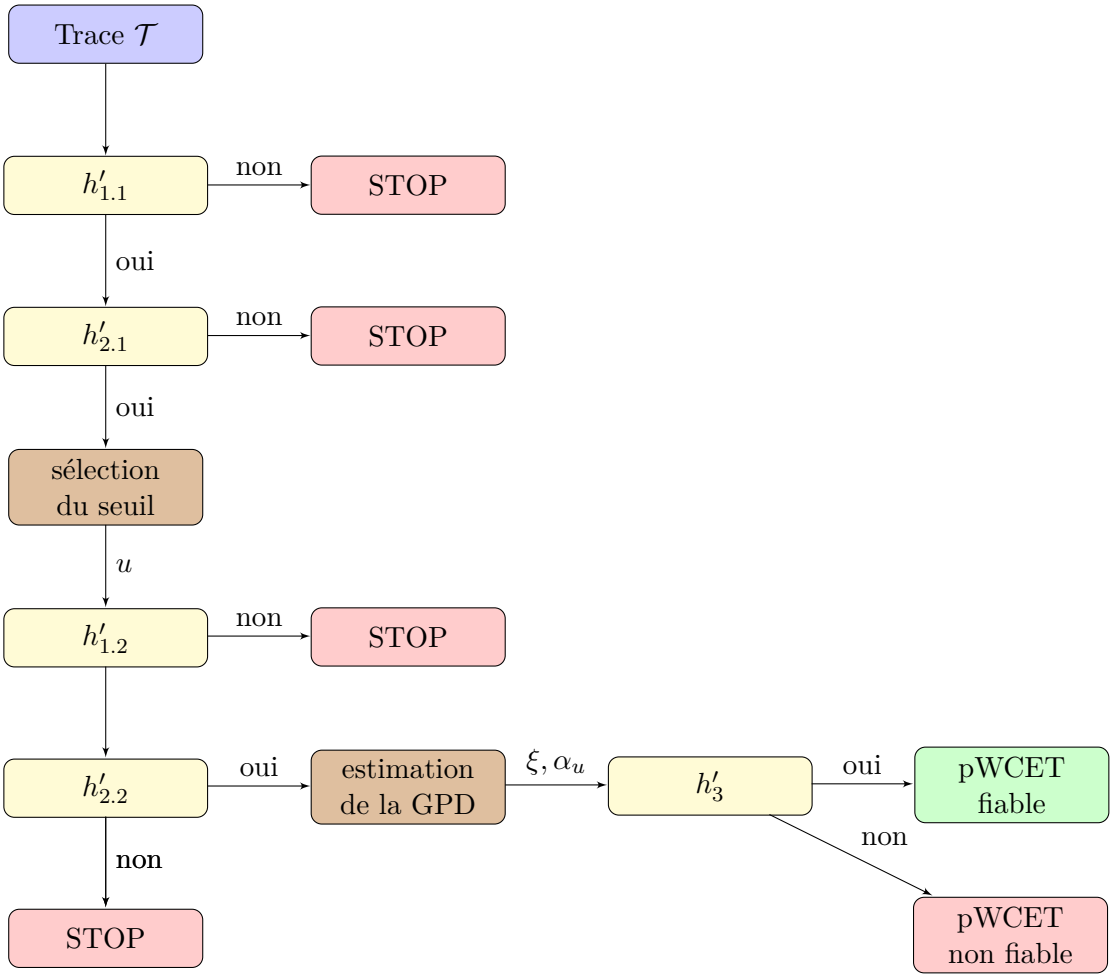


FIGURE 5.1 – Diagramme de décision implémenté dans DIAGXTRM pour évaluer la fiabilité de l'estimation du pWCET selon une approche MBPTA.

La majorité des étapes du diagramme de décision concernent les hypothèses de l'EVT à vérifier. Les étapes *sélection du seuil* et *estimation de la GPD* sont les étapes classiques de l'approche POT.

Sélection de seuil Dans un premier temps, nous utilisons différentes valeurs de seuil u qui correspondent aux quantiles à 50, 60, 70, 80 et 90% de la trace $\mathcal{T}$. Les quantiles sont calculés grâce à la fonction quantile $q(p, \mathcal{T})$ de probabilité p de la trace $\mathcal{T}$.

Estimation de la GPD Les paramètres de la GPD sont estimés par des estimateurs comme le maximum de vraisemblance, la méthode des moments pondérés ou encore l'estimateur de Hill [PM08]. Dans notre processus d'estimation, nous avons décidé de ne

retenir que la méthode du maximum de vraisemblance (Maximum Likelihood Estimator MLE), car étant la plus répandue.

Estimateur du maximum de vraisemblance Les pics de dépassement sont utilisés pour estimer la GPD qui va modéliser le pWCET $\mathcal{C}^{\lambda}$. La notation λ fait référence au couple de paramètres de la GPD i.e. $\lambda = \{\xi, \alpha_u\}$. Si la distribution théorique des temps d'exécution $F_{\mathcal{C}}$ était connue, on aurait pu déduire de manière formelle les *vrais* paramètres de la GPD. Néanmoins, cette distribution est inconnue et nous ne disposons que de mesures. La méthode MLE basée sur les pics de dépassement mesurés *peaks* consiste à maximiser une fonction objectif qui dépend du couple de paramètres λ [PM08] :

$$\ell(\lambda, peaks) = \begin{cases} \ln \prod_{i=1}^{k} \frac{1}{\alpha_u} \left(\xi \times \frac{peaks[i]-u}{\alpha_u} + 1\right)^{-\frac{\xi+1}{\xi}} & \xi \neq 0, \\ \ln \prod_{i=1}^{k} \frac{1}{\alpha_u} \exp\left(-\frac{(peaks[i]-u)}{\alpha_u}\right) & \xi = 0. \end{cases} \tag{5.4}$$

Un algorithme d'optimisation détermine les paramètres optimaux ξ et α_u qui maximisent la fonction ℓ.

Pour le moment, l'approche propose d'arrêter le processus si une hypothèse n'est pas vérifiée. Comme il est possible de le voir sur le diagramme, l'ensemble des hypothèses $h'_{*.1}$ sont appliquées à la trace $\mathcal{T}$. L'ensemble des hypothèses $h'_{*.2}$ concernent les pics de temps d'exécution.

Enfin, pour garantir la fiabilité des pWCETs estimés, il ne suffit pas d'identifier les hypothèses à vérifier. Il est également nécessaire d'implémenter et de sélectionner des tests adéquats.

5.3 Application de tests d'hypothèse pour évaluer la fiabilité du pWCET estimé

Dans cette section, nous étudions les tests d'hypothèse afin d'évaluer chaque hypothèse h'. L'objectif est ainsi de développer un outil automatique de diagnostic.

5.3.1 Tests d'hypothèse

Les tests d'hypothèse étudient le rejet d'une hypothèse nulle H_0. Si H_0 n'est pas rejetée, c'est une condition nécessaire mais non suffisante pour approuver H_0. Nous faisons le choix d'utiliser ce type de test car il nous est impossible de prouver formellement l'hypothèse à vérifier.

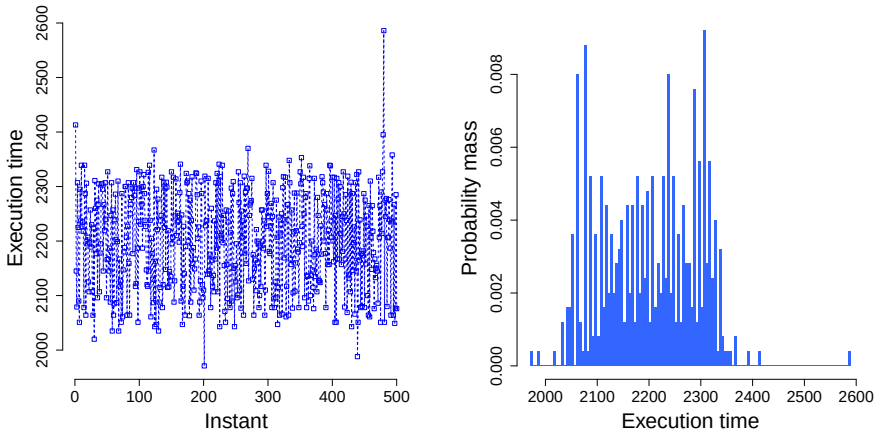
Pour décider du rejet de l'hypothèse nulle H_0, nous nous intéressons à la valeur $p-value$ retournée par le test. La $p-value$ est calculée en fonction du résultat du test $result$ par rapport à une métrique ou des valeurs établies dans une table. Enfin, il reste à choisir un seuil α tel que pour une $p-value$ inférieure strictement à ce seuil, l'hypothèse nulle soit rejetée. Le seuil α choisi par l'utilisateur correspond à la probabilité de rejeter à tort H_0 i.e. $P(\overline{H_0}|H_0)$. La valeur de α est généralement choisie de manière arbitraire parmi les valeurs 0.01, 0.05 et 0.1, et correspondent à autant de valeurs critiques cvs. Si

result est supérieur à la valeur critique *cv* fixée, alors H_0 doit être rejetée avec un taux d'erreur de α.

Ce cadre de travail est inspiré à la fois de l'approche classique de Fisher qui considère la $p-value$ comme une mesure de la présomption contre l'hypothèse nulle, mais aussi du cadre alternatif de Neyman-Pearson qui introduit le taux d'erreur de première espèce α. La seule valeur qui n'est pas considérée dans cette thèse est le taux d'erreur de seconde espèce β qui concerne l'hypothèse alternative à l'hypothèse nulle qui n'existe pas forcément pour tous les tests. Nous considérons les deux cadres dans cette thèse car les décisions sont prises en fonction des valeurs de $p-value$ comparées à différentes valeurs de α.

Nous décrivons dans la suite les tests sélectionnés pour les différentes hypothèses h' de l'EVT.

Exemple 5.1
Pour illustrer les étapes de DIAGXTRM*, nous sélectionnons une trace de mesures de temps d'exécution. Il s'agit d'une trace issue d'un cas d'étude de l'application fibcall exécutée sur l'architecture multicœur (Section 4.2). La tâche est exécutée 500 fois sur un cœur du processeur. Pour chaque mesure, le paramètre d'entrée de la tâche est modifié de manière aléatoire parmi 28 valeurs possibles. En conséquence, on observe de la variabilité fonctionnelle due aux paramètres d'entrée et de la variabilité système due à l'architecture sur laquelle est exécutée la tâche. La figure de la trace relative à ce cas d'étude est fournie dans la Figure 5.2(a) et confirme la présence des deux types de variabilité. La trace présente des temps d'exécution variables d'une mesure à l'autre avec un maximum à l'instant 480 qui vaut 2586 cycles. Le profil de temps d'exécution ETP empirique relatif à la trace est donné dans la figure 5.2(b). L'ETP montre la distribution des temps d'exécution entre le minimum mesuré et le maximum mesuré. La queue de distribution de l'ETP, à droite du profil, montre que certaines mesures dévient des valeurs moyennes. Toutes ces observations empiriques sont des motivations pour l'application de l'EVT à la trace de mesures.*



(a) Trace de mesures du temps d'exécution de l'Exemple 5.1.

(b) ETP empirique de l'Exemple 5.1.

FIGURE 5.2 – Graphes représentatifs de l'exécution moyenne de l'Exemple 5.1.

5.3.2 Stationnarité $h'_{1.1}$ et $h'_{1.2}$

Le test de Kwiatowski Philips Schmidt Shin (KPSS) [KPSS92] a été choisi pour l'analyse de stationnarité. Le test KPSS étudie l'hypothèse nulle de stationnarité. C'est le test le plus puissant concernant l'étude de l'hypothèse nulle de stationnarité i.e. le test accepte le moins à tort l'hypothèse de stationnarité comparé à d'autres tests.

Le test KPSS peut être appliqué dans les cas d'une trace avec ou sans tendance linéaire. En notant un instant de mesure t, une trace peut s'écrire en fonction de t, $\mathcal{T}(t)$. Une trace de mesures avec une tendance linéaire peut se décomposer de la manière suivante $\mathcal{T}(t) = a \times t + b + C_t$, avec C_t une mesure de temps d'exécution issue de la distribution $F_{\mathcal{C}}$. La tendance linéaire est donnée par la fonction de $t : a \times t + b$, avec a et b deux réels. L'analyse de stationnarité d'une trace avec une tendance linéaire considère la trace retirée de cette tendance. Ce cas ne sera pas abordé dans cette étude car cette tendance donne une trace non stationnaire si elle n'est pas retirée. La tendance est aussi difficile à prendre en compte au niveau de l'EVT car cette dernière peut tendre vers l'infini pour t vers l'infini. Ce cas n'a donc que très peu de sens en terme de temps d'exécution d'une tâche issue d'une application temps réel.

Exemple 5.2
Graphiquement, la trace de mesures de l'Exemple 5.1 exhibe des mesures variables autour d'une moyenne constante des temps d'exécution. Les propriétés moyennes du système temps réel étudié ne varient pas en fonction du temps i.e. il n'y a pas de forte discontinuité entre les mesures ni de bloc isolé de mesures en fonction du temps. Le système temps réel ne subit pas de variation discontinue tant au niveau fonctionnel, entrées de la tâche, que du système, interférences, interruptions ou surcharges de calcul. Ainsi, la stationnarité de la trace ne devrait pas être rejetée même si le critère d'une moyenne constante n'est pas suffisant. Le test KPSS pour vérifier l'hypothèse $h'_{1.1}$ est appliqué à la trace de l'Exemple 5.1. L'application du test retourne une $p-value$ supérieure à 0.1. L'hypothèse $h'_{1.1}$ ne peut donc a priori pas être rejetée. Le test KPSS pour vérifier l'hypothèse $h'_{1.2}$ nécessite de sélectionner un seuil. Il sera donc appliqué après l'étude de l'hypothèse d'indépendance extrême $h'_{2.2}$, comme sur la Figure 5.1.

5.3.3 Dépendance à court terme $h'_{2.1}$

Pour l'étude de la dépendance à court terme, nous avons choisi d'appliquer le test de Brock Dechert Scheinkman (BDS) [BSDL96] qui mesure le degré de corrélation entre les blocs de temps d'exécution. Le test BDS étudie l'hypothèse nulle : la trace est iid. Certains effets de non stationnarité sont dus à des phénomènes de dépendance c'est pourquoi le test étudie ici l'hypothèse iid même s'il est plus spécifique de l'étude de l'indépendance d'une trace. Il est basé sur l'intégrale de corrélation donnée dans la Définition 5.4.

Définition 5.4 (*Intégrale de corrélation*)
L'intégrale de corrélation $CI_{m,n}(\epsilon)$ pour la dimension de plongement m et une distance ϵ est

$$CI_{m,n}(\epsilon) = \frac{1}{\binom{n}{2}} \sum_{1 \leq s < t \leq n} \sum \chi_\epsilon(||C_s^m - C_t^m||), \tag{5.5}$$

où C_s^m est une m-histoire i.e. la séquence de mesures de C_s à C_{s+m-1}, et $\chi_\epsilon(||C_s^m - C_t^m||) = \prod_{i=0}^{m-1} \chi_\epsilon(|C_{s+i} - C_{t+i}|)$ et χ_ϵ est la fonction caractéristique sur l'ensemble $[0, \epsilon)$.

Soit $CI_m(\epsilon) = \lim_{n\to\infty} CI_{m,n}$. Pour une trace iid i.e. qui respecte le test BDS :

$$\forall\ m,\ \epsilon,\ CI_m - (CI_1)^m \simeq 0.$$

L'intégrale de corrélation mesure le degré de corrélation entre des blocs de mesures (C_s^m et C_t^m) de différentes dimensions m, m la taille des blocs de la condition D de la Définition 5.2. Cette mesure dépend de la distance ϵ et si l'égalité ci-dessus est vérifiée pour la trace de mesures, alors les blocs de mesures sont indépendants. Le choix des valeurs de m et de ϵ reste arbitraire.

La métrique du test BDS suit une distribution de loi normale de moyenne nulle et de déviation standard 1 sous l'hypothèse nulle. La connaissance de cette distribution permet de calculer les $p-values$ en fonction du résultat retourné par le test.

Exemple 5.3
Graphiquement, la figure de la trace de l'Exemple 5.1 exhibe une variabilité suffisante et aucun déterminisme entre les mesures. La complexité du système temps réel introduit de la variabilité aléatoire dans le temps d'exécution de la tâche étudié. En conséquence, l'hypothèse d'indépendance ne devrait pas être rejetée dans ce cas. Le test BDS est appliqué à la trace de l'Exemple 5.1 selon les recommandations de [BM11]. En particulier, le test est appliqué pour ϵ égal à 0.5, 1 et 1.5 fois la déviation standard de la trace notée $sd(\mathcal{T})$. De plus, les auteurs recommandent d'appliquer le test pour des dimensions de plongement de 2 à $|\mathcal{T}|/200$. L'application du test BDS selon ces recommandations donne les $p-values$ de la Table 5.1. Pour $\epsilon = 0.5 \times sd(\mathcal{T})$, les $p-values$ sont nulles et donc inférieures à 0.01, l'hypothèse doit être rejetée. Néanmoins, dans les autres cas, les $p-values$ sont supérieures à 0.1 et ainsi l'hypothèse ne peut a priori pas être rejetée. Cet exemple montre la difficulté de prendre une décision sur les résultats du test BDS.

$p-value$	$\epsilon = 45.66$	91.32	136.98
$m = 2$	0	0.93	0.51
3	0	0.96	0.97

TABLE 5.1 – $p-values$ issues du test BDS appliqué à la trace de l'Exemple 5.1.

5.3.4 Indépendance extrême $h'_{2.2}$

Pour cette hypothèse, nous n'utilisons pas un test d'hypothèse au sens usuel. Nous considérons l'indice extrémal (Extremal Index EI) θ [EKM97] qui mesure le degré de regroupement des pics de dépassement. L'EI est une valeur réelle dans l'intervalle $]0;1]$. L'indice renvoie à la probabilité d'avoir un pic assez distant d'un autre pic pour qu'ils soient considérés comme indépendants. En effet, si un pic est trop proche d'un autre pic, la probabilité pour qu'ils résultent d'un même phénomène (interférences soudaines, paramètres d'entrée de la tâche différents) est grande. Ainsi ils ne peuvent pas être indépendants ce qui biaise l'estimation du pWCET. En présence de groupements de pics, cette probabilité décroit en fonction de la taille des groupements. Plus les pics seront distants, et indépendants, plus la probabilité et donc θ est proche de 1.

L'estimateur de θ a été défini sur l'idée que la probabilité d'occurrence d'un pic décroit de manière exponentielle après l'occurrence d'un pic [FS02]. Pour estimer θ, nous considérons l'ensemble des temps d'arrivée intermédiaires T_i définis par le nombre de mesures entre deux pics consécutifs. Cet ensemble de T_i donne un estimateur non

biaisé de θ :

$$\hat{\theta} = \frac{2\left(\sum_{i=1}^{k-1}(T_i - 1)\right)^2}{(k-1)\sum_{i=1}^{k-1}(T_i - 1)(T_i - 2)}, \tag{5.6}$$

avec k le nombre de pics dans la trace.

Pour vérifier la condition D' de la Définition 5.3, la valeur de $\hat{\theta}$ doit être proche de 1.

Exemple 5.4
Pour vérifier les hypothèses $h'_{1.2}$ (stationnarité des pics de dépassement) et $h'_{2.2}$, nous sélectionnons des seuils u conformément à l'approche POT. La trace de l'Exemple 5.1, n'exhibe pas de groupements de pics de temps d'exécution qui pourraient être dus à une surcharge soudaine du système. De plus, les remarques sur la stationnarité de la trace entière semble pouvoir s'étendre au niveau des pics de dépassement. Les deux hypothèses extrêmes devraient donc ne pas être rejetées. Nous appliquons ensuite respectivement le test de KPSS et l'estimation de l'EI aux différents ensembles de pics de dépassement. Les résultats de l'application des tests sont donnés dans la Table 5.2. Les résultats des tests montrent que l'hypothèse $h'_{1.2}$ ne peut pas être rejetée pour tous les seuils u, sauf pour $u = 2291$ où la $p - value$ est de 0.06 mais reste supérieure à 0.05 ce qui indique une faible présomption contre l'hypothèse. Enfin, les EIs sont tous proches de 1 donc on ne peut rejeter l'hypothèse $h'_{2.2}$ pour tous les seuils.

	$u = 2205.00$	2236.40	2265.00	2291.00	2315.00
KPSS.$p - value$	> 0.1	> 0.1	> 0.1	0.06	> 0.1
EI	0.97	1	0.93	1	1

TABLE 5.2 – Résultats des tests de vérification des hypothèses $h'_{*.2}$ pour différents seuils u.

5.3.5 Adéquation du modèle

Pour valider le modèle du pWCET estimé, donné par la fonction de répartition $F_{\mathcal{C}^\lambda}$, nous mesurons l'écart entre cette fonction et la répartition expérimentale des pics de dépassement F_u. Nous considérons donc le test d'adéquation de Cramer Von Mises (CVM) qui mesure la distance quadratique entre $F_{\mathcal{C}^\lambda}$ et F_u. Nous choisissons le test CVM car il fonctionne bien dans le cas des distributions de l'EVT [Ste77, Lai04] comme la GPD dans notre cas.

Le résultat du test CVM est une mesure de la *distance* entre les deux distributions. Plus la *distance* est proche de 0, plus le probabilité que les pics de dépassement soient issus d'une GPD est grande. Pour appliquer le test CVM, les pics de dépassement sont triés de manière croissante dans une liste *uom* (upper ordered measurements). La

distance est calculée selon :

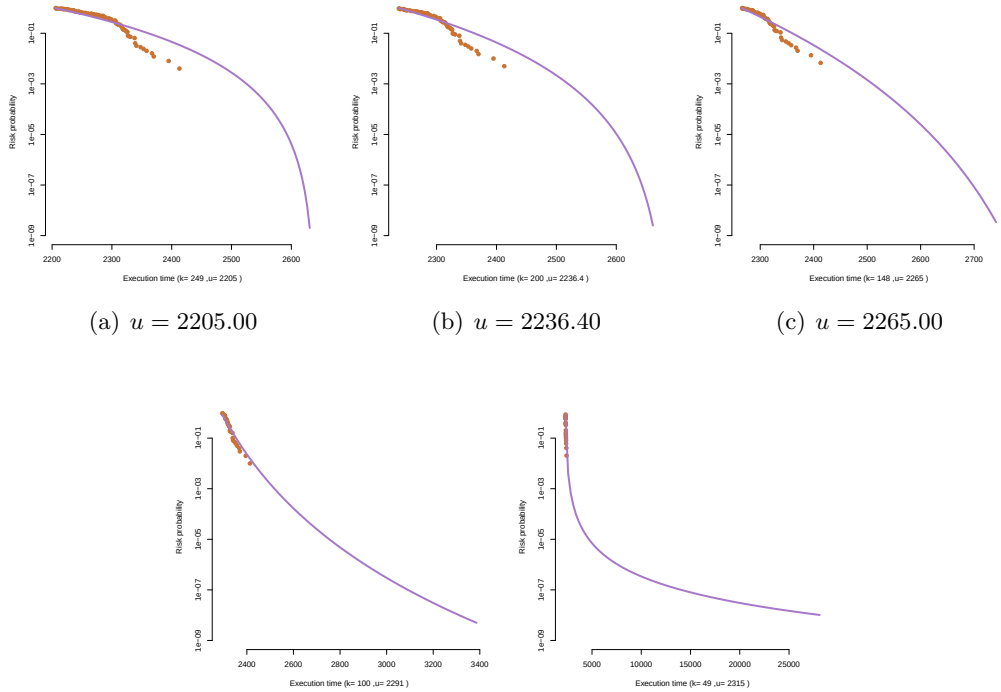
$$distance = \sum_{i=1}^{k} \left(F_{\mathcal{C}^\lambda}(uom[i]) - \frac{2i-1}{2 \times k} \right)^2 + \frac{1}{12 \times k}. \tag{5.7}$$

Les *distance*s sont associées à des valeurs critiques *cv*s qui sont données dans [CF96].

Exemple 5.5
Ainsi que nous l'avions remarqué sur l'ETP de la trace dans la Figure 5.2(b), la queue de distribution de l'ETP est constituée de quelques pics qui semblent être distribués selon une loi exponentielle. Par conséquent, approcher la queue de distribution de l'ETP discret par une distribution GPD est cohérent. Nous appliquons le test CVM sur la trace de l'Exemple 5.1 pour vérifier l'hypothèse h'_3 selon les différentes valeurs de seuil u. Les résultats du test CVM pour chaque seuil ainsi que les estimations des paramètres de la GPD ξ et α_u retournées par la méthode MLE sont donnés dans la Table 5.3. Les résultats montrent que seule la distance du pWCET pour le seuil $u = 2315.00$ est suffisamment faible pour ne pas rejeter l'hypothèse h'_3. Enfin, les estimations des paramètres de la GPD pour différents seuils montre la forte variabilité des estimations. Ainsi, le choix d'un seuil, donc d'un couple de paramètres et donc d'un pWCET devient incertain. Dans ce cas, il fait sens de ne retenir que le dernier seuil car c'est le seuil qui ne rejette pas h'_3. Néanmoins, dans le cas où plusieurs seuils u vérifient cette hypothèse le choix du seuil devient plus difficile. De plus, on donne dans la Figure 5.3 les tracés des pWCETs issus de la phase d'estimation des paramètres. Les pWCETs sont représentés selon leur ICDF respective $\bar{F}_{\mathcal{C}^\lambda}$ en fonction des probabilités de dépassement en échelle logarithmique (Risk Probability). Graphiquement, on remarque bien que le pWCET colle bien aux pics de dépassement seulement dans le dernier cas en accord avec les valeurs de cl_3. Conformément aux valeurs des paramètres de forme ξ, dans ce cas, plus le seuil u augmente plus la valeur de ξ augmente et ainsi la fonction converge de plus en plus lentement vers 0. En conséquence, les WCETs seuil à 10^{-9} sont respectivement de plus en plus pessimiste, de 2631 cycles pour $u = 2205$ à 28110 cycles pour $u = 2315$.

	$u = 2205.00$	2236.40	2265.00	2291.00	2315.00
distance	1.99	1.33	1.51	1.39	0.12
ξ	-0.19	-0.14	-0.07	0.08	0.34
α_u	83.88	64.54	44.72	24.8	15.83

TABLE 5.3 – Résultats du test CVM de vérification de l'hypothèse h'_3 ainsi que les estimations des paramètres de la GPD selon MLE pour différentes valeurs de seuil u.



(a) $u = 2205.00$ (b) $u = 2236.40$ (c) $u = 2265.00$

(d) $u = 2291.00$ (e) $u = 2315.00$

FIGURE 5.3 – Tracé des pWCETs issus de la trace de l'Exemple 5.1 pour les différentes valeurs de seuil u. Les points représentent la distribution empirique des pics de dépassement au dessus de u et la ligne pleine le pWCET estimé à partir de ces pics.

5.4 Application de la logique floue pour la quantification de la fiabilité du pWCET estimé

Après avoir sélectionné les tests adéquats pour tester les hypothèses de l'EVT, nous souhaitons mettre en place un cadre pour prendre une décision sur l'applicabilité de l'EVT sur une trace donnée $\mathcal{T}$. Pour ce faire nous appliquons la logique floue pour quantifier l'applicabilité de l'EVT [KTOY00].

La logique floue est utilisée dans le développement de processus de prise de décision et réfère à la statistique robuste [Buc05, GM08, KTOY00] quand elle est appliquée aux statistiques pour quantifier les incertitudes associées aux approches classiques de test d'hypothèse.

En conséquence, au lieu de décider si la trace est stationnaire ou non, la logique floue quantifie la distance de la trace au modèle stationnaire. Si la trace est proche du modèle de chaque hypothèse h', alors la confiance dans l'application de l'EVT est grande. Au lieu de ne choisir qu'une seule valeur de seuil arbitraire α, quatre valeurs sont sélectionnées pour qu'il soit possible soit de rejeter, soit d'avoir une forte, faible,

très faible, aucune présomption contre l'applicabilité de l'EVT correspondant aux seuils α 0,01, 0,025, 0,05 et 0,1. De cette manière, l'approche de diagnostic gagne en facilité d'interprétation pour déclarer le pWCET estimé comme fiable ou non grâce à ces niveaux de confiance (confidence level cl). Cette opération de clarification correspond à déflouter (defuzzification) le résultat du test statistique comme résumé dans la Figure 5.4.

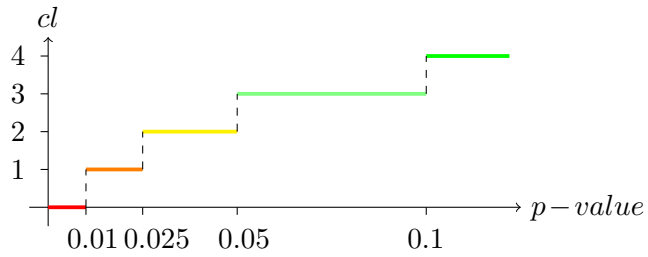


FIGURE 5.4 – La fonction de DEFUZZIFICATION est définie sur les valeurs floues de $p-value$ (ou de manière équivalente leur cv correspondante) vers les niveaux de confiance cls dans $\{0; 1; 2; 3; 4\}$. Pour des cls croissants l'hypothèse nulle testée H_0 est rejetée ou il y a une forte, faible, très faible, aucune présomption contre H_0.

5.4.1 Processus de prise de décision

Chaque bloc de test d'hypothèse, blocs h' dans la Figure 5.1, fournit une $p - value$ sur la trace. La $p - value$ reste difficile à interpréter et est propre à chaque test. Elle est donc évaluée selon un niveau de confiance cl grâce au processus DEFUZZIFICATION décrit sur la Figure 5.4. Le principal avantage des niveaux de confiance est de définir une échelle commune pour tous les tests afin de les agréger et obtenir un niveau de confiance global sur l'application de l'EVT à une trace. Cette agrégation doit se faire de manière automatique comme nous l'avons spécifié pour notre outil. Il existe plusieurs méthodes pour agréger des niveaux de confiance, mais la méthode choisie doit être en accord avec les exigences de l'EVT.

Dans le processus d'évaluation proposé, on retrouve cinq hypothèses à vérifier $h'_{1.1}$, $h'_{2.1}$, $h'_{1.2}$, $h'_{2.2}$, h'_3. En conséquence, les niveaux de confiance associés à chaque hypothèse sont notés $cl_{1.1}$, $cl_{2.1}$, $cl_{1.2}$, $cl_{2.2}$, cl_3. Le niveau de confiance final est noté $cl_{reliability}$. Pour satisfaire les exigences de l'EVT, la règle d'agrégation doit être telle que si un des niveaux de confiance est nul, alors le niveau de confiance final l'est aussi. Sinon, le niveau de confiance est le barycentre des niveaux de confiance comme décrit dans l'Algorithme 1
.

L'Algorithme 1 nous permet de définir formellement la fiabilité d'un pWCET selon les conditions d'application de l'EVT.

Algorithm 1 Algorithme d'agrégation AGGREGATION des niveaux de confiance

```
1: confidence_levels ← [cl_1.1, cl_2.1, cl_1.2, cl_2.2, cl_3]
2: procedure AGGREGATION(confidence_levels)
3:     if min(confidence_levels) ≥ 1 then
4:         cl_reliability ← (cl_1.1 + cl_2.1 + cl_1.2 + cl_2.2 + cl_3) /5
5:     else
6:         cl_reliability ← 0
```

Définition 5.5 (*Fiabilité du pWCET*)
Soit une trace de mesures de temps d'exécution $\mathcal{T}$, la fiabilité du pWCET est évaluée selon le niveau de confiance $cl_{reliability}$ tel que :

$$cl_{reliability} = \wedge_i cl_i \tag{5.8}$$

avec cl_i les niveaux de confiance associés aux hypothèses de l'EVT.

Conformément à l'Algorithme 1, soit l'application de l'EVT est non valide pour $cl_{reliability} = 0$, soit elle est valide et plus $cl_{reliability}$ est élevé, plus la confiance dans le pWCET estimé est élevée. Le processus de diagnostic proposé dépend aussi fortement des tests sélectionnés pour évaluer la fiabilité des pWCETs estimés.

De plus, nous avons fait le choix de définir la fiabilité du pWCET estimé comme l'intersection de tous les niveaux de confiance. Si dans ce cas d'application systématique un pWCET est diagnostiqué comme fiable, la confiance dans ce résultat est très élevée. Néanmoins, l'ajout de différents tests rend l'évaluation plus contraignante notamment parce que certaines hypothèses auraient tendance à se recouvrir. C'est le cas entre les hypothèses moyennes $h'_{*.1}$ et les hypothèses extrêmes $h'_{*.2}$. Des études et des discussions autour de traces de mesures pourraient aider à déterminer la nécessité de telle ou telle hypothèse pour relâcher l'évaluation dans certains cas.

5.4.2 Application aux tests d'hypothèse de l'EVT

Stationnarité et Adéquation du modèle Les tests KPSS et CVM, respectivement pour la stationnarité et l'adéquation du modèle, sont des tests où aucun paramètre n'a besoin d'être fixé. Dans ce cas, la $p - value$ retournée par chaque test est donnée à la fonction DEFUZZIFICATION et cette dernière renvoie le niveau de confiance respectif.

Dépendance à court terme Pour le test BDS, nous avons mis en évidence le besoin de fixer la dimension m et la distance ϵ. Pour éviter de fixer ces paramètres à une seule valeur, nous réalisons le test BDS pour plusieurs valeurs. L'application de la logique floue permet en outre de fusionner les résultats des tests BDS pour chaque valeur des paramètres et ainsi d'avoir un niveau $cl_{2.1}$ final représentatif. Sans le cadre de la logique floue, ce genre de décision aurait dû se faire par une évaluation humaine.

Le test BDS est appliqué en pratique comme dans l'Algorithme 2 de la procédure d'analyse de dépendance à court terme ShortTermDependence($\mathcal{T}$).

Algorithm 2 Application du test BDS [BM11]

1: **procedure** ShortTermDependence($\mathcal{T}$)
2: **for** $\epsilon \in \{\frac{1}{2}sd(\mathcal{T}), sd(\mathcal{T}), \frac{3}{2}sd(\mathcal{T})\}$ **do**
3: **for** m from 2 to $\frac{length(\mathcal{T})}{200}$ by 1 **do**
4: $results.append($Defuzzification $(BDS(\mathcal{T}, \epsilon, l)))$ ▷ $results$ est une liste des résultats du test BDS pour les différents ϵ et l
5: $cl_{2.1} \leftarrow \frac{sum(results)}{length(results)}$ ▷ règle de calcul de $cl_{2.1}$

Indépendance Extrême Enfin, le calcul du niveau de confiance pour l'analyse de l'indépendance extrême est basé sur l'estimation de l'EI θ. Dans ce cas, le calcul des niveaux de confiance n'est pas basé sur une $p - value$. Néanmoins, il est possible d'appliquer les mêmes niveaux de confiance tels que les pics sont indépendants pour une valeur de θ proche de 1. En conséquence, le niveau $cl_{2.2}$ est sur la même échelle que les niveaux de confiance issus des autres tests ce qui ne perturbe pas la fusion des niveaux de confiance pour le calcul de $cl_{reliability}$. Le niveau $cl_{2.2}$ est calculé en fonction de la valeur de θ tel qu'il est spécifié dans la table de correspondance de la Table 5.4.

$\theta \in$	[1.00 ;0.95[	[0.95 ;0.90[	[0.90 ;0.85[	[0.85 ;0.80[	[0.80 ;0.00[
$cl_{2.2}$	4	3	2	1	0

Table 5.4 – Niveaux de confiance de l'Indice Extrême.

Exemple 5.6
Nous appliquons tous les tests interprétés par la logique floue à la trace de l'Exemple 5.1 et pour toutes les valeurs de seuil u sélectionnées. Les résultats de l'analyse de fiabilité sont donnés dans la Table 5.5. Les premiers indices $cl_{*.1}$ sont indépendants du seuil u et sont donc identiques pour toutes les valeurs. Les indices concernant les pics de dépassement $cl_{*.2}$ ne sont pas très sensibles au différentes valeurs de seuil car elles sont quasi similaires. Enfin, l'adéquation du pWCET aux mesures est seulement acceptée pour la valeur de seuil $u = 2315.00$. En définitive, seul le pWCET issu à partir des pics de dépassement supérieurs au seuil le plus élevé peut être considéré comme fiable. Graphiquement, on représente la fiabilité du pWCET selon un diagramme radar comme dans la Figure 5.5. Si les niveaux de confiance vérifient l'Equation (5.8), alors le tracé du diagramme est vert, sinon il est rouge.

cl_i	$u = 2205.00$	2236.40	2265.00	2291.00	2315.00
$cl_{1.1}$	4	=	=	=	=
$cl_{2.1}$	2.67	=	=	=	=
$cl_{1.2}$	4	4	4	3	4
$cl_{2.2}$	4	4	3	4	4
cl_3	0	0	0	0	4
$cl_{reliability}$	0	0	0	0	4

TABLE 5.5 – Résultats des niveaux de confiance de l'analyse de fiabilité DIAGXTRM.

5.4.3 Discussion

Dans l'existant, les auteurs imposaient un modèle de pWCET basé sur la loi de Gumbel ($\xi = 0$). Dans ce cas d'étude, les pWCETs estimés suivent tous des lois de Fréchet ($\xi > 0$). En comparaison, imposer un modèle basé sur la loi de Gumbel donnerait des WCETs seuils plus précis car moins élevés, respectivement à une probabilité p, et donc plus proches des mesures. Nous distinguons la notion de précision dans le cas du WCET et la précision dans le cas du pWCET.

On mesure le pessimisme du WCET seuil de probabilité p par rapport au temps d'exécution maximum mesuré avec la fonction $Pess(p)$:

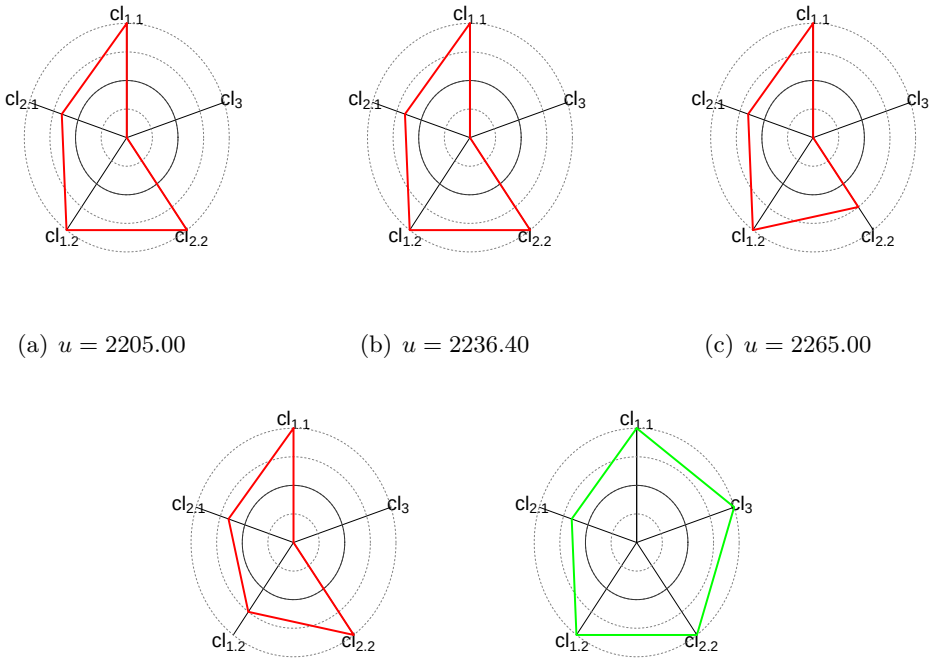
$$Pess(p) = \frac{\langle WCET; p\rangle - \max(\mathcal{T})}{\max(\mathcal{T})} \times 100(\%). \tag{5.9}$$

Plus le pessisme est faible et plus le WCET seuil est précis ainsi que représenté dans la Figure 2.1. Le WCET estimé doit être supérieur au WCET exact qui est inconnu la plupart du temps. Ainsi, nous comparons le WCET seuil au maximum mesuré. Le pessimisme des WCETs seuil inférés réfère à leur précision et non pas la confiance dans la sûreté de l'estimation. La sûreté de l'estimation est garantie par l'application de l'EVT évaluée avec l'indice $cl_{reliability}$. Si le pWCET estimé est fiable alors le WCET seuil à p est statistiquement sûr avec un niveau de confiance p.

La précision du pWCET estimé est donnée par le niveau de confiance sur l'adéquation du modèle estimé avec les mesures i.e. cl_3. Plus ce niveau est élevé, plus la précision du pWCET est élevée. Ainsi, les pics de dépassement mesurés sont modélisés avec confiance et les WCETs seuil extraits sont fiables.

Nous comparons selon les deux métriques précédentes le pWCET obtenu ci-avant par DIAGXTRM et le pWCET dans le cas d'une loi de Gumbel *gumbel* pour le seuil $u = 2315.00$. Les résultats de la comparaison sont donnés dans la Table 5.6.

Les résultats de la comparaison confirment, dans ce cas, que le modèle Gumbel donne des WCETs seuil plus précis car les valeurs de $Pess\left(10^{-9}\right)$ sont plus faibles. Cependant, le niveau de confiance sur l'adéquation du modèle cl_3 est plus faible. En conséquence, les pWCETs estimés modélisent moins précisément les mesures lorsqu'ils sont contraints à la seule loi de Gumbel ce qui entraîne une dégradation de la fiabilité du pWCET estimé voire même d'obtenir des pWCETs estimés abbérants. Dans ce cas, le modèle *gumbel*



(a) $u = 2205.00$ (b) $u = 2236.40$ (c) $u = 2265.00$

(d) $u = 2291.00$ (e) $u = 2315.00$

FIGURE 5.5 – Diagrammes radar d'évaluation de la fiabilité du pWCET issu de la trace de l'Exemple 5.1 pour les différentes valeurs de seuil u.

	$\langle WCET; 10^{-9} \rangle$	$Pess\,(10^{-9})$	cl_3
DIAGXTRM	28110	987.00	4
gumbel	2774	7.27	2

TABLE 5.6 – Précision du pWCET estimé et pessimisme des WCETs seuils résultants. Comparaison entre le cas *diagXtrm* où le pWCET n'est pas contraint à une loi de Gumbel et le cas *gumbel* où le pWCET est contraint par une loi de Gumbel.

peut même être accepté mais au prix d'une dégradation de la fiabilité et du systématisme recherché.

5.5 Algorithme de sélection du seuil u pour respecter le critère de biais/variance

Le choix du seuil u impacte directement la sélection des pics de dépassement et donc le pWCET estimé comme le montre l'analyse des résultats de la Table 5.5. Il existe plusieurs méthodes de sélection du seuil u dans la littérature EVT [SM12] et le choix de la méthode n'est pas systématique. Les méthodes existantes les plus connues sont d'ordre graphique [EKM97] (Mean Excess plot, Hill plot), et reposent sur une interprétation subjective. Aucune méthode n'arrive à fournir une valeur exacte dans le cas d'une trace de mesures.

Le critère le plus important pour le choix du seuil u est le critère de *biais/variance*.

5.5.1 Méthode proposée

Il existe différentes méthodes automatiques dans la littérature EVT comme le test de la queue exponentielle [Gar02] ou encore des "règles du pouce" peu généralisables [SM12]. Néanmoins, nous avons choisi d'implémenter une méthode automatique de sélection de seuil basée sur les niveaux de confiance. En particulier, nous implémentons un processus d'itération basé sur le niveau de confiance cl_3, qui s'intègre dans le diagramme de décision comme sur la Figure 5.6. Pour un même jeu de paramètres d'entrée de l'algorithme proposé, ce dernier retourne la même valeur de seuil u pour la trace $\mathcal{T}$.

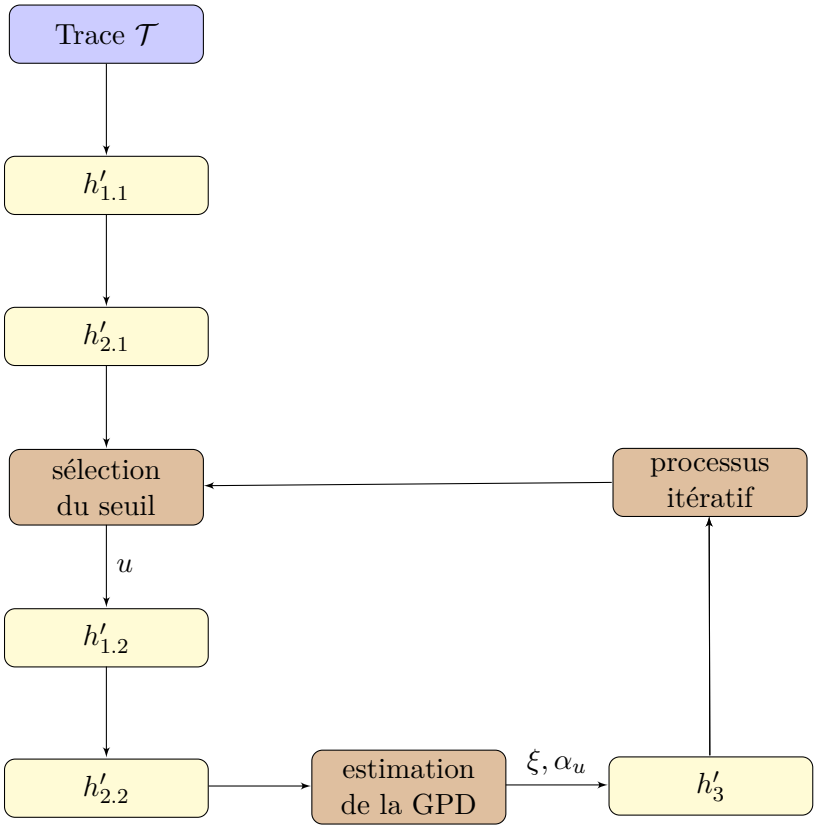


FIGURE 5.6 – Intégration du processus itératif de sélection du seuil dans le diagramme de décision de l'estimation du pWCET selon une approche MBPTA.

A l'implémentation :

Sélection du seuil $u = q(\mathcal{T}, 1 - k/n)$,

Processus itératif $for\ k \in [25; n]\ if\ cl_3$ nul depuis $n \times \epsilon_{stop}$ itérations $then\ break\ else\ k+ = 1$.

Le processus de sélection du seuil est basé sur le nombre de pics de dépassement k. La fonction $q(\mathcal{T}, p)$ est la fonction quantile $q(p)$ pour la trace $\mathcal{T}$. Avec $p = 1 - k/n$, la fonction quantile renvoie le seuil u tel qu'il y ait k pics de dépassement.

Le processus itératif fournit donc un ensemble de seuils possibles ordonnés de manière décroissante U. Pour ne conserver que des pWCETs fiables, seuls les seuils correspondant à des niveaux de confiance $cl_{2.2} > 0$ et $cl_3 > 0$ sont conservés dans U. Chaque valeur de seuil dans U donne un ensemble de pics de dépassement à partir desquels un pWCET a été estimé. Ainsi, l'ensemble des seuils U correspond à la liste des couples des paramètres de la GPD $[\xi(u), \alpha_u(u), \forall u \in U]$. L'ensemble des paramètres ξ, α_u et u est suffisant pour caractériser complètement le pWCET et ainsi fournir une estimation du WCET seuil $\langle WCET; p \rangle$. En conséquence, la liste des paramètres de la GPD en fonction de

l'ensemble des seuils U donne la liste des WCETs seuil à la probabilité p notée $\mathcal{L} = [\langle WCET; p\rangle(u), \forall u \in U]$.

Le problème de sélection de seuil repose sur le compromis entre biais et variance des estimations [PM08, SM12]. Pour des seuils élevés, le paramètre de forme varie beaucoup d'une valeur de seuil à l'autre. Pour de faibles seuils, le paramètre de forme devient presque constant en fonction du seuil, mais le nombre plus élevé de pics de dépassement biaise l'estimation. Nous implémentons alors une heuristique simple pour choisir un seuil qui se situe après la zone de variance des valeurs de WCET seuil de $\mathcal{L}$. Cette méthode permet de prendre en compte l'évolution des valeurs de ξ et de α_u dans la résolution du problème de biais/variance mais dépend de la probabilité p du WCET seuil.

Pour implémenter cette méthode nous nous basons sur la métrique qui évalue la variance empirique de $\mathcal{L}$:

Variance $\quad var(\mathcal{L}) = \frac{1}{|\mathcal{L}|}\sum_{i=1}^{|\mathcal{L}|}(\mathcal{L}[i]-\mu)^2$, avec $\mu = \frac{1}{|\mathcal{L}|}\sum_{i=1}^{|\mathcal{L}|}\mathcal{L}[i]$.

Cette métrique mesure la variabilité de la liste $\mathcal{L}$. En conséquence, si la liste converge vers une valeur en fonction de u, alors la variabilité décroit en fonction de u.

Pour choisir une valeur de $u \in U$ respectant le critère de biais/variance, on compare la variance de $\mathcal{L}$ pour différentes tailles de la liste. On note la liste $\mathcal{L}$ de taille i comme $\mathcal{L}[1:i]$. Dans le cas ou $i = |U|$ alors $\mathcal{L}[1:i] = \mathcal{L}$. La variabilité de référence de $\mathcal{L}$ est $var(\mathcal{L})$. Algorithmiquement,

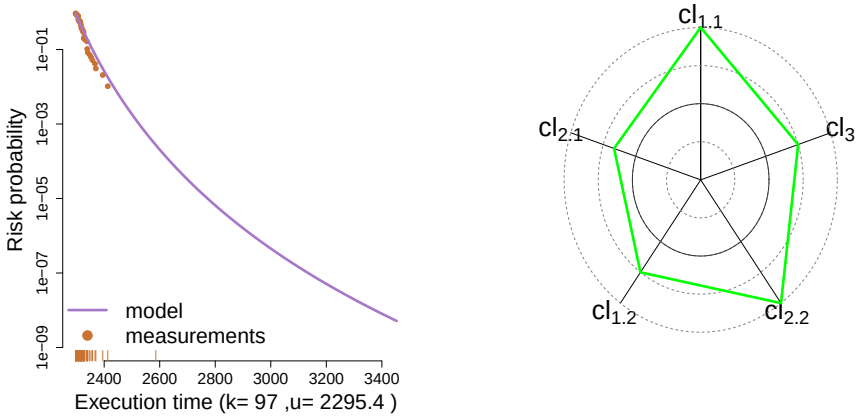
1: **for** $i \in [\![|U|, 2]\!]$ **do**
2: **if** $\frac{abs(var(\mathcal{L})-var(\mathcal{L}[1:i]))}{var(\mathcal{L})} < \epsilon_{var}$ **then**
3: $break$
4: **else**
5: $i-=1$

Avec *abs* la fonction valeur absolue. De cette manière, dès lors que le niveau de variabilité relatif descend en dessous d'un certain seuil, on considère que la valeur du WCET seuil commence à converger alors on choisit la valeur du seuil correspondant.

L'intérêt de l'heuristique que nous proposons pour la sélection du seuil basée sur le respect du critère de biais/variance permet de systématiser la sélection sans interprétation subjective. Enfin, il reste à choisir convenablement les valeurs des paramètres p, ϵ_{stop} et ϵ_{var}. Par la pratique, nous fixons $\epsilon_{stop} = \epsilon_{var} = 0.01$. Néanmoins, un expert a la liberté de faire varier ces degrés de convergence selon les résultats obtenus. De plus, il est toujours possible d'extraire un seuil u de manière graphique à partir du graphe du WCET seuil à p en fonction de u, comme pour les méthodes basées sur le graphe ME ou le graphe de Hill.

5.5.2 Application

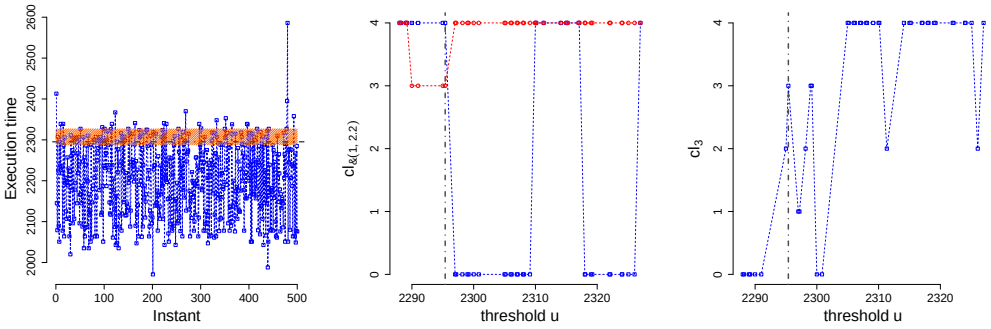
Exemple 5.7
L'algorithme de sélection de seuil proposé est appliqué à la trace de l'Exemple 5.1. Le processus itératif est illustré dans la Figure 5.8. La Figure 5.8(a) montre la zone U, zone hachurée, de temps d'exécution testés pour des potentiels seuils u. La ligne en pointillés indique le seuil u sélectionné qui vaut 2295.39 cycles. La Figure 5.8(b) montre l'évolution de $cl_{1.2}$ et de $cl_{2.2}$ en fonction du seuil u. Les deux niveaux de confiance restent élevés pour toutes les valeurs de seuil explorées. La Figure 5.8(c) montre l'évolution de cl_3 en fonction du seuil u. Le niveau cl_3 commence à chuter à 0 pour des seuils vers 2290 cycles ce qui stoppe le processus itératif. La Figure 5.8(d) montre l'évolution des WCETs seuil à 10^{-9} en fonction du seuil u. On retrouve bien le problème de biais/variance i.e. pour des seuils élevés les valeurs varient fortement d'un seuil à l'autre, à droite du graphe, alors que pour des seuils faibles les WCETs seuil convergent vers une même valeur mais peut être biaisée si le nombre de pics de dépassement est trop élevé. Dans la Figure 5.7, on donne le tracé du pWCET pour le seuil sélectionné $u = 2295.39$ ainsi que le diagramme radar de la fiabilité du pWCET estimé. Le pWCET estimé permet d'inférer un WCET seuil à 10^{-9} qui vaut 3453 cycles. Le pessimisme de ce nouveau WCET seuil à 10^{-9} est de 33% ce qui est bien plus intéressant que dans le cas où le seuil u avait été imposé arbitrairement.



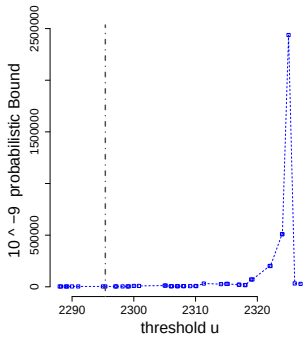
(a) pWCET pour le seuil sélectionné.

(b) Diagramme radar pour le seuil sélectionné.

FIGURE 5.7 – Résultats de DIAGXTRM avec l'analyse de sélection de seuil proposée pour la trace de l'Exemple 5.1.



(a) Zone d'exploration U. (b) Niveaux $cl_{*.2}$ en fonction de u. (c) Niveau cl_3 en fonction de u.



(d) $\langle WCET; 10^{-9} \rangle$

FIGURE 5.8 – Résultats de l'analyse de sélection de seuil proposée pour la trace de l'Exemple 5.1.

5.6 Construction d'un intervalle de confiance fiable sur les estimations de WCET seuil

L'application de l'EVT est basée sur un théorème asymptotique (Théorème 2.2) i.e. la GPD est la loi limite de la distribution des pics de dépassement pour une infinité de mesures. Pour une infinité de mesures, les paramètres estimés par la méthode MLE convergent en distribution vers une loi normale dont la moyenne est donnée par les *vrais* paramètres si $\xi > -0.5$ [Smi85]. Dans le cas où $\xi \leq -0.5$ le pWCET estimé ne peut être fiable par MLE. Ce cas correspond à des queues de distribution très courtes et est rarement rencontré dans des applications pratiques [Col01].

Cependant, en pratique, nous ne disposons pas d'une infinité de mesures ce qui introduit une incertitude autour de l'estimation des paramètres avec MLE ξ^{MLE} et α_u^{MLE}. En conséquence, les paramètres estimés sont normalement distribués autour des vrais paramètres donnés par leur moyenne et leur erreur standard (Standard Error SE)

ξ^{SE} et α_u^{SE} :

$$\xi^{MLE} \sim \mathcal{N}\left(\xi, \frac{1}{k}(1+\xi)^2 = \left(\xi^{SE}\right)^2\right);\ \alpha_u^{MLE} \sim \mathcal{N}\left(\alpha_u, \frac{1}{k}2(1+\xi)(\alpha_u)^2 = \left(\alpha_u^{SE}\right)^2\right) \tag{5.10}$$

La connaissance de la distribution des paramètres estimés permet de construire un intervalle de confiance sur les WCETs seuil. Pour ce faire il faut se référer à la méthode du delta [Col01]. Néanmoins, nous souhaitons déterminer une enveloppe de pWCETs fiables, relativement aux niveaux de confiance définis, autour du pWCET estimé $F_{\mathcal{C}^\lambda}$. Dans ce cas, nous proposons une méthode basée sur le niveau de confiance d'adéquation du modèle cl_3.

5.6.1 Construction d'un intervalle de confiance fiable sur le WCET seuil de probabilité p

Pour un WCET seuil de probabilité p $\langle WCET; p\rangle$ issu du pWCET de référence $\mathcal{C}^\lambda$, nous désirons estimer un intervalle de confiance $\left[WCET^{-SE}; WCET^{+SE}\right]$. Les bornes de l'intervalle de confiance doivent être issues de pWCETs fiables et l'intervalle doit être le plus grand possible. Ainsi, le plus grand des minorants $WCET^{-SE}$ est le WCET seuil fiable le plus précis, et le plus petit des majorants $WCET^{+SE}$ est le WCET seuil fiable le plus sûr. Néanmoins, si la différence relative entre les deux bornes est trop importante la confiance dans l'estimation du WCET seuil à la probabilité p est amoindrie compte tenu du nombre de mesures. En effet, comme donné dans l'Equation (5.10), l'erreur standard sur les paramètres de la GPD est inversement proportionnelle au nombre de pics de dépassement k.

L'intervalle $\left[WCET^{-SE}; WCET^{+SE}\right]$ est appelé intervalle de confiance fiable car sa construction repose à la fois sur l'erreur standard sur les paramètres estimés par la méthode MLE et à la fois sur le cadre de la logique floue de diagXtrm qui évalue la fiabilité de l'application de l'EVT. En ce sens, l'intervalle de confiance fiable diffère de l'intervalle de confiance classique relatif à un niveau de confiance e.g. ; 95%, donné par la méthode du delta [Col01].

Pour obtenir un ensemble de pWCETs, nous tirons de manière aléatoire $N_{samplings}$ couples de paramètres λ dans la loi de distribution des paramètres du pWCET donnée dans l'Equation (5.10). Chaque couple de paramètres correspond à un pWCET dont on déduit un WCET seuil $\langle WCET; p\rangle$. Il y a donc $N_{samplings}$ pWCETs.

A ce stade, la fiabilité des pWCETs ne dépend plus que du niveau de confiance cl_3. Ainsi, pour respecter le critère de fiabilité de l'intervalle de confiance, nous ne conservons que les pWCETs dont le niveau cl_3 est égal au niveau cl_3 du pWCET estimé de référence i.e. pour les valeurs ξ^{MLE} et α_u^{MLE}. Si aucun pWCET ne répond à ce critère, nous choisissons de conserver les pWCETs dont le niveau cl_3 est strictement positif. Néanmoins, si aucun pWCET ne répond à ce critère, l'estimation de l'intervalle de confiance fiable est stoppée et nous utilisons la méthode du delta.

Les valeurs minimales et maximales de WCET seuil à p, respectivement $WCET^{-SE}$ et $WCET^{+SE}$, issues des pWCETs fiables tirés aléatoirement sont utilisées pour déterminer l'intervalle de confiance fiable sur le WCET seuil à p. Les pWCETs retenus pour l'enveloppe du pWCET de référence dépendent de la probabilité du WCET seuil dont on souhaite un encadrement. De plus, pour s'assurer d'obtenir des bornes sûres, seuls les pWCETs fournissant des WCETs seuil supérieurs au maximum de la trace sont considérés.

Enfin, comme les bornes sont issues d'un tirage aléatoire sur le couple des paramètres du pWCET, il est possible que pour deux tirages différents les bornes varient. Cette variation dépend du nombre de tirages $N_{samplings}$. Dans les cas présentés le nombre de tirages est de 1000 et les bornes obtenues pour différents tirages sont stables. En pratique, on observe que pour des nombres de tirages inférieurs e.g., 100, les bornes varient sensiblement.

5.6.2 Application

Exemple 5.8
Nous appliquons l'algorithme de construction d'un intervalle de confiance fiable proposé à la trace de l'Exemple 5.1 de ce Chapitre. Nous nous intéressons à l'intervalle de confiance de l'estimation du WCET seuil à 10^{-9} *pour* $N_{samplings} = 1000$*. L'enveloppe de confiance résultante autour du pWCET estimé est donnée dans la Figure 5.9. Cette analyse nous fournit l'intervalle de confiance fiable autour du WCET seuil à* 10^{-9} *qui vaut 3453 cycles. L'intervalle de confiance fiable obtenu vaut* $[2590; 5083]$*. Comme observé sur la Figure 5.9, le pWCET inférieur de l'envelope a un* cl_3 *égal à 4 alors que le pWCET supérieur a un niveau égal à 3. La confiance sur les bornes de l'intervalle de confiance fiable est élevée. De plus, les deux bornes de l'intervalle sont plutôt proches ce qui renforce la confiance dans l'estimation du WCET seuil à* 10^{-9}*. En effet, ainsi que le montre la figure de l'enveloppe, cette dernière s'écarte à mesure que la probabilité du WCET seuil tend vers 0. Ceci s'explique par le nombre limité de mesures qui ne permet pas de faire des estimations de WCET seuil assez confiantes pour des probabilités qui tendent davantage vers 0.*

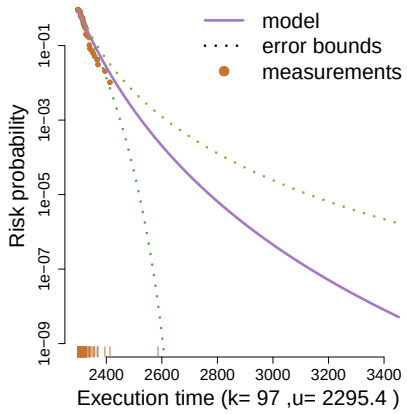


FIGURE 5.9 – Tracé de l'enveloppe autour du pWCET estimé pour un WCET seuil à 10^{-9}.

5.7 Métrique pour déterminer la convergence des mesures de temps d'exécution

Le nombre de mesures à réaliser est un choix critique pour reproduire l'estimation du pWCET. L'application de l'EVT permet de réduire le nombre de mesures à réaliser pour estimer des quantiles extrêmes i.e. des valeurs rares. Néanmoins, pour deux traces de même ordre de grandeur réalisées dans les mêmes conditions, il serait possible d'obtenir différents pWCETs. Il est donc nécessaire de réaliser un nombre de mesures suffisant pour assurer la reproductibilité de l'estimation.

Un des critères retenu pour déterminer le nombre de mesures à réaliser est de déterminer la convergence des pWCETs estimés pour différentes tailles de trace de mesures. Si le pWCET estimé ne varie plus par rapport au nombre de mesures alors on considère que le pWCET estimé a convergé vers le pWCET final. Le nombre de mesures final réfère à la taille de la trace utilisée n.

Des méthodes pour s'assurer de la convergence des mesures ont été appliquées dans [CGSH+12] et [LNBC12]. Cependant, elles ne dépendent pas des critères de fiabilité introduits précédemment.

5.7.1 Méthode proposée

La méthode proposée dans cette section utilise le cadre de travail basé sur la logique floue afin de déterminer un niveau de confiance sur la convergence du pWCET estimé en fonction du nombre de mesures. Le pWCET estimé repose sur ses deux paramètres, le paramètre de forme ξ et le paramètre d'échelle α_u. Le pWCET estimé est localisé selon la valeur de seuil u. Pour étudier la convergence du pWCET, on utilise une trace réduite $\mathcal{T}_{x\%}$ définie à partir de la trace $\mathcal{T}$ telle que : $\mathcal{T}_{x\%} = \mathcal{T}\,[1 : \lceil n - x \times n/100 \rceil]$. Le seuil u sélectionné à partir de la trace final $\mathcal{T}$ est un bon candidat pour la trace réduite $\mathcal{T}_{x\%}$.

Ainsi, le seuil u est inchangé entre les deux pWCETs. Les pics de dépassement de la trace $\mathcal{T}_{x\%}$ au dessus du seuil u sont utilisés pour estimer le pWCET $\mathcal{T}_{x\%}$ de paramètres ξ' et α'_u formant le couple de paramètres. On peut considérer que le pWCET a convergé si les deux pWCETs sont proches i.e. si la différence entre les paramètres ξ et ξ', et α_u et α'_u est négligeable. Une autre condition importante est le niveau d'adéquation cl_3 du pWCET issu de la trace réduite qui doit être supérieur à 1.

En pratique, on calcule la différence relative entre les deux couples de paramètres. La différence est interprétée selon des niveaux de confiance $cl_{4.1}$ et $cl_{4.2}$ comme dans la Table 5.7.

différence	< 0.01	$\in[0.01\,;0.02[$	$\in[0.02\,;0.05[$	$\in[0.05\,;0.1[$	≥ 0.1
$cl_{4.1\|2}$	4	3	2	1	0

TABLE 5.7 – Niveaux de confiance de la différence relative entre les deux couples de paramètres.

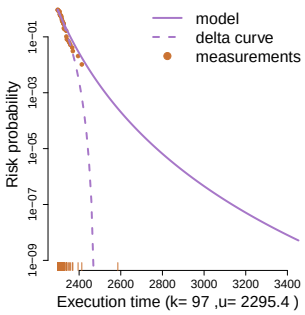
Dans le cas particulier où ξ et ξ' sont de signes différents alors le niveau de confiance $cl_{4.1}$ est nul. On construit un niveau de confiance cl_4 basé sur les valeurs de $cl_{4.1}$, $cl_{4.2}$ et le niveau d'adéquation cl_3 du pWCET issu de la trace réduite. Le niveau cl_4 est le centre de gravité entre ces trois niveaux s'ils sont tous supérieurs à 1 sinon cl_4 vaut 0.

5.7.2 Application

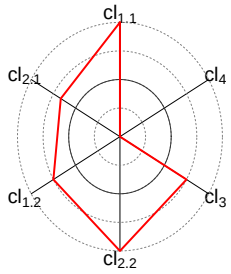
Exemple 5.9
On applique la métrique d'évaluation de la convergence pour la trace de l'Exemple 5.1. On choisit de comparer le pWCET estimé de la trace de l'Exemple 5.1 au pWCET issu de la trace réduite de 10% $\mathcal{T}_{10\%}$. La comparaison des deux pWCETs est donnée dans la Figure 5.10(a). Le résultat de l'évaluation de la convergence cl_4 est donné dans le digramme radar complet d'évaluation de la fiabilité du pWCET estimé de la Figure 5.10(b). La comparaison des tracés montre une grande différence entre les deux pWCETs, confirmé par le niveau cl_4 qui vaut 0. La différence entre les deux traces provient de la présence du maximum de temps d'éxécution dans la trace initiale à l'instant 480. On choisit de déplacer ce temps d'exécution maximum dans la trace pour qu'il n'influence pas l'analyse de convergence. Les nouveaux résultats sont donnés dans les Figures 5.10(c) et 5.10(d). Dans ce cas, la convergence des mesures est acceptée. En pratique la trace initiale nécessiterait davantage de mesures pour accepter la convergence et assurer la reproductibilité du pWCET.

5.8 Conclusion

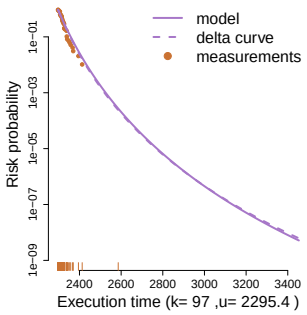
La méthode d'évaluation DIAGXTRM proposée dans ce chapitre permet de caractériser la fiabilité des pWCETs estimés selon l'EVT. Les pWCETs sont estimés selon l'approche POT dans le cas de mesures stationnaires. L'approche POT permet de ne pas



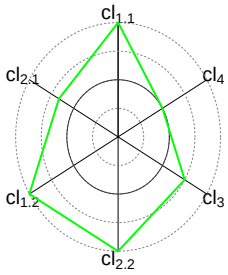
(a) Comparaison des pWCETs. Le pWCET estimé est en trait plein et le pWCET estimé de la trace réduite est en pointillés.



(b) Diagramme radar de la fiabilité du pWCET estimé avec l'analyse de convergence cl_4.



(c) Comparaison des pWCETs avec le maximum déplacé aléatoirement.



(d) Diagramme radar de la fiabilité du pWCET estimé dans le cas du maximum déplacé aléatoirement.

FIGURE 5.10 – Résultats de l'analyse de convergence des mesures pour la reproductibilité du pWCET pour la trace de l'Exemple 5.1.

contraindre le paramètre de forme et ainsi d'obtenir des pWCETs plus précis. Le cas de mesures stationnaires est plus adapté pour des mesures sur des plateformes COTS. C'est une méthode qui se veut systématique et complète du point de vue de l'ensemble des hypothèses requises pour appliquer l'EVT à une trace de mesures. Le choix d'être le plus complet possible entraîne une dépendance au niveau des tests d'hypothèse, notamment des tests de stationnarité et d'indépendance, car les notions sont très liées. Néanmoins, cette dépendance permet d'avoir un cadre complet de l'étude de l'application de l'EVT à des traces de mesures et ainsi de garantir la sûreté des estimations. L'application de tests d'hypothèse et de la logique floue permet d'établir une méthode de diagnostic automatique et facile à interpréter pour un concepteur de système temps réel.

De plus, la méthode d'évaluation de la robustesse du pWCET estimé également proposée dans ce chapitre permet de pallier aux incertitudes d'estimation des paramètres

du pWCET dues au nombre limité de mesures dans la trace $\mathcal{T}$. Nous avons développé trois algorithmes pour augmenter et quantifier la robustesse du pWCET estimé respectivement par rapport à la sélection du seuil u, à l'estimation des paramètres ξ et α_u avec la méthode du maximum de vraisemblance et au nombre de mesures n nécessaires à la reproductibilité du pWCET. L'outil résultant du diagramme de décision et des algorithmes de robustification est nommé DIAGXTRM en référence au DIAGnostique de l'application de la théorie des valeurs eXTRêMes à une trace de mesures.

Chapitre 6

Application de DIAGXTRM à des traces de mesures de temps d'exécution issues de différents processeurs COTS

Sommaire

Résumé 2
Le chapitre précédent a permis de mettre en place l'outil DIAGXTRM *d'évaluation de la fiabilité des pWCETs estimés à partir d'une trace de mesures. Ce chapitre traite de l'application de l'outil à des traces de mesures du temps d'exécution de tâches pour différentes plateformes COTS. L'objectif est d'étudier les capacités et les limites de l'outil proposé relativement au besoin de mettre en place une approche MBPTA d'estimation du pWCET fiable et robuste. La fiabilité des pWCETs estimés dépend directement de la qualité des traces et du nombre de mesures. La pertinence de l'application de l'EVT à certaines traces peut être discutable et même révoquée en raison du type de tâche ou de plateforme considérée. La discussion des résultats par rapport aux traces et aux systèmes, permet de mettre en évidence les conditions qui favorisent l'application de l'EVT, ainsi que les conditions pour estimer des pWCETs effectivement représentatifs du pire cas.*

On donne dans l'Annexe 8.9 l'ensemble des résultats fournis par DIAGXTRM appliqué aux traces décrites dans le Chapitre 4. Dans ce chapitre, les traces sont analysées suivant des catégories de résultat et enfin par rapport aux architectures dont elles sont issues. Ce chapitre d'application permet de tester la robustesse de l'outil développé et d'identifier des cas limites de l'application de l'EVT. Respectivement aux exigences de l'outil, des résultats systématiques et rapides sont obtenus. On considère dans cette thèse des traces de taille inférieure à 1000 échantillons pour évaluer la puissance de l'EVT avec peu de mesures.

6.1 Application de DIAGXTRM

6.1.1 Application fiable

Les résultats de DIAGXTRM appliqué aux traces de la Table 6.1 concernent les cas de pWCETs diagnostiqués comme fiables. En effet, les indices de confiance sont tous supérieurs à 1. De plus, les pWCETs estimés fournissent des WCETs seuil précis. Le pessimisme des WCETs seuil à 10^{-9}, pour cette catégorie de traces, va de 10^{-6} à 10^{0}. Enfin, les bornes des intervalles de confiance fiables autour des WCETs seuil sont également précises et proches l'une de l'autre. Graphiquement, les traces qui amènent à des pWCETs estimés fiables ressemblent aux traces de la Figure 6.2. Il existe donc un profil de trace particulier qui respecte les exigences de l'EVT. En définitive, les pWCETs estimés dans cette catégorie sont issus de traces qui respectent les conditions de l'EVT, les modèles approchent correctement les mesures, et les intervalles de confiance fiables autour des WCETs seuils sont resserrés. En conséquence, les WCET seuils issus de cette catégorie de traces sont statistiquement sûrs relativement aux conditions de mesure de leur trace respective. Cette catégorie de traces regroupe près de la moitié des traces étudiées.

$\mathcal{T}$	$cl_{1.1}$	$cl_{2.1}$	$cl_{1.2}$	$cl_{2.2}$	cl_3	cl_4	$cl_{reliability}$	$Pess\,(10^{-9})$	$WCET^{-SE}$	$WCET^{+SE}$	écart
trace1	4	3.92	4	4	4	3.67	3.93	1.99E-02	1.03E-04	4.41E-02	4.40E-02
trace2	4	3.58	4	4	4	3.67	3.88	2.55E-02	4.21E-04	1.20E-01	1.20E-01
trace3	1	1.08	4	1	4	3.00	2.35	5.17E-02	7.32E-04	6.61E-01	6.60E-01
trace7	4	3.50	4	4	4	3.67	3.86	2.17E-01	2.48E-02	5.09E-01	4.85E-01
trace13	4	4.00	4	2	1	2.67	2.95	4.84E+00	6.07E-02	2.42E+02	2.42E+02
trace18	4	3.08	4	4	4	4.00	3.85	1.63E+00	9.06E-03	7.09E+00	7.08E+00
trace19	4	2.83	4	3	3	2.33	3.19	1.28E-01	6.83E-03	3.25E+00	3.24E+00
trace21	4	3.58	4	4	1	3.00	3.26	5.63E-05	4.02E-07	2.03E-04	2.02E-04
trace25	4	4.00	4	4	4	3.67	3.94	2.90E-01	1.36E-03	1.30E+00	1.30E+00
trace28	4	4.00	4	3	4	3.33	3.72	4.24E-01	2.51E-04	2.86E+00	2.86E+00
trace30	4	4.00	4	4	4	3.67	3.95	6.67E-03	3.63E-05	1.96E-02	1.96E-02
trace32	1	4.00	4	4	1	2.67	2.78	5.47E-06	2.14E-06	3.19E-04	3.17E-04

TABLE 6.1 – Résultats de DIAGXTRM dans le cas des pWCETs diagnostiqués fiables. Dans cette table les valeurs de $WCET^{-SE}$ et de $WCET^{+SE}$ correspondent à leur pessimisme respectif ; écart est la différence entre le pessimisme des bornes pour mesurer la taille de l'intervalle de confiance fiable.

6.1.2 Application fiable mais dont le pessimisme des WCETs seuil est trop important

La trace8, dont les résultats de l'application de DIAGXTRM sont donnés dans la Table 6.2, fournit un pWCET fiable mais dont le pessimisme des WCETs seuil associé est trop élevé. En effet, le pessimisme du WCET seuil à 10^{-9} dans ce cas est de l'ordre de 10^5. Le WCET seuil calculé à partir du pWCET estimé est certainement sûr mais la précision est tellement faible qu'il est inutilisable. La trace8 est donnée dans la Figure 6.1.

La trace8 exhibe des pics peu fréquents qui divergent fortement des valeurs moyennes de la trace ce qui donne une valeur positive du paramètre de forme ξ du pWCET estimé. Ainsi, le pWCET converge très lentement vers 0 et les WCETs seuil sont éloignés des mesures. Pour diminuer le pessimisme des WCETs seuil, la trace nécessiterait davantage de mesures proches des pics. Si ces pics sont dus à une condition particulière il faudrait réaliser davantage de mesures dans ces conditions. Sinon, le nombre de mesures à réaliser doit être plus important pour avoir la chance d'observer des pics similaires. De plus, la taille de l'intervalle de confiance fiable autour du WCET seuil est de l'ordre de 10^9. En conséquence, l'incertitude d'estimation du pWCET est trop grande pour qu'il soit considéré dans une analyse d'ordonnancement.

6.1.3 EVT inapplicable

Dans cette section, DIAGXTRM a conclu l'EVT comme étant inapplicable aux traces étudiées.

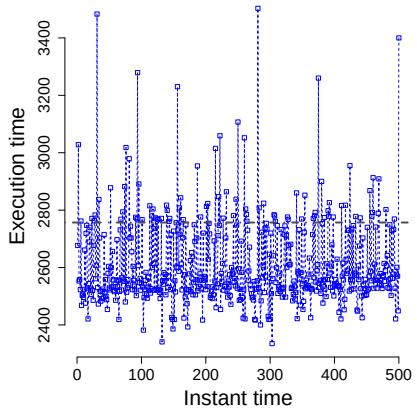


FIGURE 6.1 – trace8

$\mathcal{T}$	$cl_{1.1}$	$cl_{2.1}$	$cl_{1.2}$	$cl_{2.2}$	cl_3	cl_4	$cl_{reliability}$	$Pess\,(10^{-9})$	$WCET^{-SE}$	$WCET^{+SE}$	écart
trace8	4	3.50	4	4	4	2.67	3.70	2.81E+05	4.40E+01	1.76E+09	1.76E+09

TABLE 6.2 – Résultats de DIAGXTRM dans le cas des pWCETs diagnostiqués fiables mais dont le pessimisme des WCETs seuil est trop important.

Stationnarité

Les traces de cette catégorie donnent des pWCETs non fiables car elles ne respectent pas le critère de stationnarité. Ainsi, la confiance dans la stationnarité des traces de cette section $cl_{1.1}$ est nulle comme il est possible de la voir dans la Table 6.3. Le cas de la trace15 montre que le test d'indépendance n'est pas redondant avec le test de stationnarité. La trace15 ne contient que 35 mesures qui semblent augmenter avec le temps. Le test étant sensible à la taille de la trace, il semblerait logique de réaliser davantage de mesures pour espérer obtenir un pWCET fiable. Les cas des traces 12 et 17 sont donnés dans la Figure 6.5. La trace12 exhibe une tendance linéaire et la trace17 montre deux comportements bien distincts au cours du temps. La trace17 modifiée ne prend en compte que le pire comportement et le pWCET estimé résultant est diagnostiqué comme fiable. Ces deux cas de trace sont représentatifs de traces non stationnaires. Ainsi, le test de stationnarité a correctement identifié ces cas. Les mesures étant dépendantes du temps il n'est pas possible de garantir la fiabilité des pWCETs estimés correspondants.

Variabilité

Pour les traces de cette section, il est impossible de sélectionner un seuil u. Ainsi, que le montrent les résultats de l'application de DIAGXTRM à ces trace donnés dans la Table 6.4, plusieurs indices ne sont pas disponibles (Not Available NA). En effet, les traces sont similaires aux cas des traces de la Figure 6.3 ou même sont constantes à une

$\mathcal{T}$	$cl_{1.1}$	$cl_{2.1}$	$cl_{1.2}$	$cl_{2.2}$	cl_3	cl_4	$cl_{reliability}$	$Pess\,(10^{-9})$	$WCET^{-SE}$	$WCET^{+SE}$	écart
trace12	0	0.00	NA	NA	NA	NA	NA	NA	NA	NA	NA
trace15	0	2.67	4	0	4	3.33	0.00	3.93E+00	4.76E-02	7.82E+02	7.82E+02
trace17	0	0.00	4	3	4	0.00	0.00	4.61E+02	2.66E-01	1.87E+06	1.87E+06
trace17	4	3.67	4	3	4	4.00	3.78	6.92E+02	3.43E-02	2.26E+07	2.26E+07

TABLE 6.3 – Résultats de DIAGXTRM dans le cas des pWCETs diagnostiqués non fiables car la confiance dans la stationnarité des mesures $cl_{1.1}$ est nulle.

valeur. Or, pour estimer un pWCET il est nécessaire d'avoir au moins une vingtaine de valeurs différentes. En conséquence, les traces de cette catégorie exhibent une variabilité du temps d'exécution insuffisante pour être en mesure d'estimer un pWCET. Ce type de trace entraîne l'arrêt de DIAGXTRM au niveau de la sélection du seuil et donne un niveau de confiance sur l'indépendance des mesures $cl_{2.1}$ nul. Le cas de la trace26 est différent puisque la trace exhibe une variabilité suffisante et est similaire aux cas de trace de la première catégorie qui donnent des pWCETs fiables. Le test BDS utilisé pour tester l'indépendance des mesures est peut être trop restrictif. La trace26, donnée dans la Figure 6.8(a), peut être discutée graphiquement et à la lumière des conditions de mesure.

$\mathcal{T}$	$cl_{1.1}$	$cl_{2.1}$	$cl_{1.2}$	$cl_{2.2}$	cl_3	cl_4	$cl_{reliability}$	$Pess\,(10^{-9})$	$WCET^{-SE}$	$WCET^{+SE}$	écart
trace4	4	0.00	NA	NA	NA	NA	NA	NA	NA	NA	NA
trace6	4	1.67	NA	NA	NA	NA	NA	NA	NA	NA	NA
trace26	4	0.00	4	4	4	3.33	0.00	2.26E-01	1.82E-03	8.07E-01	8.05E-01
trace27	4	0.00	NA	NA	NA	NA	NA	NA	NA	NA	NA
trace29	0	0.00	NA	NA	NA	NA	NA	NA	NA	NA	NA
trace31	4	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

TABLE 6.4 – Résultats de DIAGXTRM dans le cas des pWCETs diagnostiqués non fiables car la confiance dans l'indépendance des mesures $cl_{2.1}$ est nulle.

Autres indices de confiance

Les résultats donnés par les indices $cl_{*.2}$ qui concernent la confiance des propriétés extrêmes de la trace sont plus difficiles à analyser graphiquement. Dans les cas précédents, les traces ne sont pas assez variables pour sélectionner un seuil u, ainsi les tests des hypothèses $h'_{*.2}$ n'ont pu être appliqués. Ainsi, les tests des hypothèses $h'_{*.1}$ sont assez efficaces pour discriminer ces cas limites pour lesquels l'application de l'EVT n'est pas acceptable. Dans les cas où la variabilité est suffisante, les indices $cl_{*.2}$ retournés par DIAGXTRM sont rarement nuls. En effet, la procédure de sélection du seuil u ne considère que les seuils pour lesquels ces indices sont supérieurs à 1.

Enfin, l'indice de confiance sur la pertinence du modèle GPD cl_3 est presque le plus

important pour la décision finale sur l'application de l'EVT à la trace donnée. Les indices cl_3 sont également rarement nuls puisque la procédure de sélection du seuil ne conserve que les seuils qui donne un niveau de confiance cl_3 supérieur à 1. Pour toutes les traces considérées, exceptées celles qui exhibent une faible variabilité, les indices de confiance cl_3 sont en majorité égaux à 4. Ce résultat montre la pertinence du cadre d'application de l'EVT à des traces de mesures de temps d'exécution.

6.1.4 Application à discuter

Dans ces cas de trace, la majorité des indices de confiance sont supérieurs à 1. Néanmoins, la systématisation de l'approche DIAGXTRM proposée rejette les pWCETs estimés issus des traces de cette section. Nous proposons de discuter les résultats associés à ces traces pour éventuellement obtenir des pWCETs fiables.

Manque de mesures

Les traces de cette section, dont les résultats de l'application de DIAGXTRM sont donnés dans la Table 6.5, manquent de mesures pour fournir un pWCET fiable et notamment reproductible. La confiance dans la reproductibilité du pWCET cl_4 est donc nulle. En plus du potentiel de ne pouvoir reproduire le pWCET estimé, les WCETs seuil fournis par ces pWCETs ont en moyenne un pessimisme important d'un ordre de grandeur supérieur à 10^2. Or, comme nous l'avons remarqué dans le cas de la trace8, la faible précision provient du fait d'avoir des pWCETs dont le paramètre de forme ξ est positif. La présence de pics fortement divergents et peu fréquents favorisent l'estimation de paramètres de forme positifs. En conséquence, dans ces cas de trace, il est nécessaire de réaliser davantage de mesures pour assurer la reproductibilité du pWCET, ainsi que de réduire le pessimisme des WCETs seuil qui en découlent.

Pour justifier ce propos, nous avons décider de réaliser davantage de mesures dans le cas de la trace20. La seconde trace20 dans la Table 6.5, est une trace de mesures de taille 2000 réalisée dans les mêmes conditions de mesure que la trace20 initiale. Dans ce cas, 1000 mesures de plus ont été suffisantes pour passer le critère de reproductibilité cl_4. De plus, le pessimisme du WCET seuil à 10^{-9} a été diminuée passant de 10^0 à 10^{-1}.

Opérations sur la trace

Dans cette section, les résultats de la Table 6.6 indiquent que les traces correspondantes ne donnent pas de pWCET estimé fiable. Néanmoins, une discussion basée sur l'analyse graphique des traces a permis d'obtenir des pWCETs fiables moyennant des modifications sur la trace. La trace10 et la trace10 modifiée (trace10 privée de ses 100 premières mesures) sont données dans la Figure 6.4. Le cas de la trace16 et de la trace16 modifiée (trace16 privée de ses 500 premières mesures) est similaire. Dans ces cas, on note la présence de pics de temps d'exécution au début de la trace et un comportement stable dans la suite de l'exécution. Nous supposons que ces pics sont à l'origine de la non

$\mathcal{T}$	$cl_{1.1}$	$cl_{2.1}$	$cl_{1.2}$	$cl_{2.2}$	cl_3	cl_4	$cl_{reliability}$	$Pess\,(10^{-9})$	$WCET^{-SE}$	$WCET^{+SE}$	écart
trace5	4	3.92	4	4	4	0.00	0.00	5.83E-01	2.19E-03	2.10E+01	2.10E+01
trace9	4	1.67	4	1	4	0.00	0.00	1.73E+02	1.04E-01	3.51E+04	3.51E+04
trace11	4	3.00	4	4	4	0.00	0.00	8.01E+05	8.00E-01	7.16E+11	7.16E+11
trace14	4	1.00	1	0	4	0.00	0.00	3.33E+05	2.71E+00	1.56E+11	1.56E+11
trace20	4	4.00	4	0	4	0.00	0.00	8.68E+00	3.97E-02	3.19E+06	3.19E+06
trace20	4	2.59	4	4	2	3.00	3.27	2.97E-01	6.09E-04	7.43E-01	7.42E-01
trace22	2	3.92	4	4	4	0.00	0.00	3.98E-02	1.15E-04	5.37E+00	5.37E+00
trace23	3	3.83	4	4	4	0.00	0.00	5.18E+01	1.18E-01	1.36E+03	1.36E+03
trace24	4	4.00	4	4	4	0.00	0.00	1.39E+02	2.81E-01	2.25E+03	2.25E+03

TABLE 6.5 – Résultats à discuter de DIAGXTRM dans le cas des pWCETs diagnostiqués non fiables par manque de mesures.

fiabilité des pWCETs estimés dans ces cas. De plus, le pessimisme du WCET seuil de la trace16 de l'ordre de 10^{13} est complètement irréaliste. Ainsi, nous considérons ces pics comme relevant de conditions de mesure différentes du reste des mesures et décidons de modifier les traces en supprimant ces pics. Ce faisant, les pWCETs estimés à partir des traces modifiées sont fiables et les WCETs seuil qui en découlent sont précis. A noter que la confiance dans l'indépendance des mesures de la trace16 est nul. Néanmoins, la trace est similaire aux cas de traces de la première catégorie qui donnent des pWCETs fiables. En conséquence, nous pouvons encore discuter le résultat du test BDS.

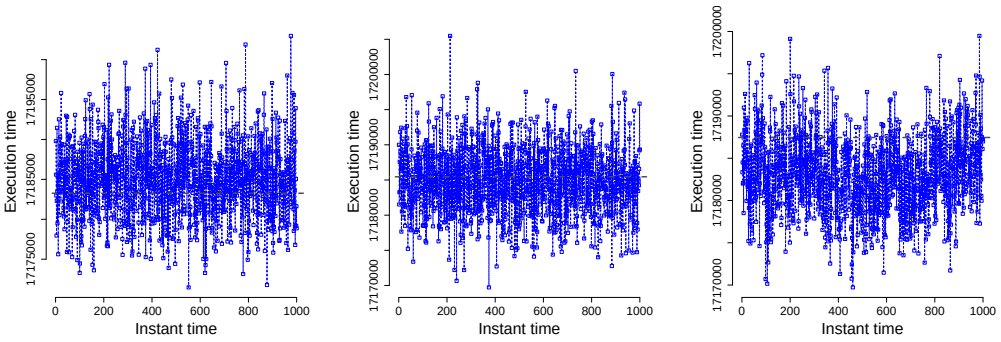
$\mathcal{T}$	$cl_{1.1}$	$cl_{2.1}$	$cl_{1.2}$	$cl_{2.2}$	cl_3	cl_4	$cl_{reliability}$	$Pess\,(10^{-9})$	$WCET^{-SE}$	$WCET^{+SE}$	écart
trace10	4	0.00	1	2	4	2.33	0.00	4.83E+02	5.67E-01	1.61E+04	1.61E+04
trace10	4	3.67	3	3	4	3.67	3.56	1.37E+00	2.60E-03	1.67E+01	1.67E+01
trace16	3	4.00	4	0	4	2.33	0.00	5.96E+13	2.22E+02	4.88E+25	4.88E+25
trace16	4	0.00	4	1	4	3.33	0.00	1.34E+00	1.86E-02	2.55E+01	2.55E+01

TABLE 6.6 – Résultats à discuter de DIAGXTRM dans le cas des pWCETs diagnostiqués non fiables en opérant sur la trace.

6.2 Analyse des résultats par architecture

6.2.1 Système embarqué critique industriel

Les pWCETs issus des traces 1 à 3 du premier cas d'étude considéré sur cette architecture sont diagnostiqués comme fiables. Comme le montre la Figure 6.2, les traces présentent assez de variabilité pour appliquer l'EVT. Les pics de dépassement au dessus du seuil u, indiqué en pointillés sur les traces, sont considérés pour modéliser le pWCET. Les pWCETs sont modélisés avec des GPDs dont le paramètre de forme ξ est négatif



(a) trace1 (b) trace2 (c) trace3

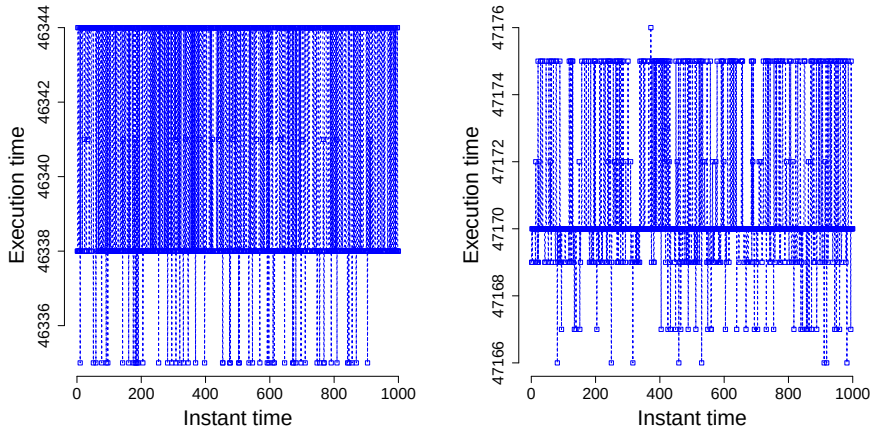
FIGURE 6.2 – Traces du premier cas d'étude du cas du système embarqué critique industriel.

pour les trois cas. Il en résulte que les WCETs seuils sont précis car leur pessimisme est de l'ordre de 10^{-2}%. Un pessimisme si bas est plutôt inhabituel dans l'industrie. Pour garantir la sûreté du WCET des tâches, les industriels ajoutent une marge 20% au temps d'exécution maximum mesuré ou à l'estimation fournie par un outil. Le WCET seuil fourni est inféré sur la base de l'EVT et est diagnostiqué comme fiable. Rien n'empêche d'ajouter une marge arbitraire au WCET seuil en dépit de la performance. Enfin, les bornes des intervalles de confiance fiables sont proches du WCET seuil inféré ce qui renforce la confiance de l'estimation pour la probabilité 10^{-9}.

Au contraire, les pWCETs issus des traces du deuxième cas d'étude, trace4 et trace6, ne sont pas diagnostiqués comme fiables. Graphiquement, la Figure 6.3 montre que les traces 4 et 6 exhibent très peu de variabilité, 4 valeurs différentes dans le premier cas et 6 dans le deuxième. En conséquence, la confiance dans l'indépendance des mesures $cl_{2.1}$ est basse voire nulle dans le premier cas. Les indices suivants ne sont pas disponibles car le choix d'un seuil ne retourne que trop peu de valeurs différentes. Les traces 4 et 6 sont clairement des cas limites de l'application de l'EVT. En conséquence, on explore la possibilité d'ajouter un offset aléatoire aux mesures de temps d'exécution. On obtient alors les traces 5 et 7. Dans ce cas, les pWCETs des traces sont diagnostiqués comme fiables car assez de variabilité a été ajoutée. Néanmoins, les paramètres de l'offset aléatoire sont choisis arbitrairement et n'ont pas plus de pertinence que d'ajouter une marge empirique de 20%. Dans le cas de la trace5, davantage de mesures sont nécessaires pour passer l'indice cl_4. De plus, les WCETs seuils sont précis.

6.2.2 Architecture multicœur

Les pWCETs issus des traces réalisées à partir de l'architecture multicœur sont fiables dans le cas 8 et requièrent seulement de nouvelles mesures dans les cas 9 et 11. Le cas de la trace10 n'est pas fiable car l'indice d'indépendance $cl_{2.1}$ est nul alors que les autres traces réalisées dans les mêmes conditions sont indépendantes. La trace10 dans la Figure 6.4 montre deux mesures qui divergent fortement des valeurs moyennes.



(a) trace4 (b) trace6

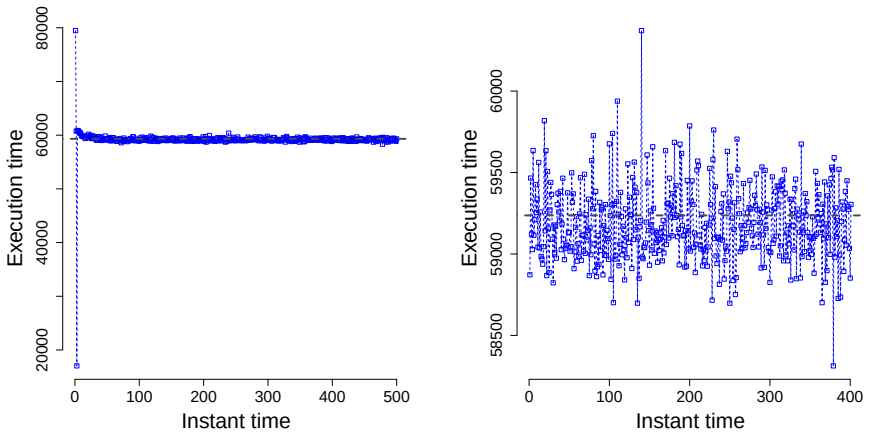
FIGURE 6.3 – Traces du second cas d'étude du cas du système embarqué critique industriel.

On applique de nouveau DIAGXTRM à la trace10 privée des 100 premières mesures. Dans ce cas, le pWCET estimé est diagnostiqué comme fiable. Cependant, les WCETs seuil ne sont pas très précis et les intervalles de confiance fiables sont très grands. Par conséquent, la confiance dans les estimations dans ce cas d'architecture est moindre que dans le cas précédent. La plateforme apporte beaucoup de variabilité au niveau du temps d'exécution, néanmoins, le nombre de mesures est trop restreint pour avoir entièrement confiance dans les estimations. Seul le cas de la trace10 modifiée permet d'obtenir un WCET seuil précis. En revanche, le WCET seuil est inférieur au maximum mesuré de la trace10 initiale. En analysant la trace initiale, le maximum mesuré correspond à la première mesure i.e. avec les données non présentes dans le cache. Ce maximum correspond au cas de mesures réalisées alors que les caches sont vidés avant chaque exécution de la tâche. Pour obtenir des mesures représentatives de ce cas, il faudrait vider les caches avant chaque exécution. En conclusion, pour cette architecture il est nécessaire de réaliser davantage de mesures pour estimer un WCET seuil à 10^{-9}.

6.2.3 Système embarqué robotique

Les traces issues de cette plateforme sont très variées. La difficulté dans ce contexte embarqué est l'évolution contrainte du robot en fonction du temps et de son environnement. Ainsi, certaines traces sont très courtes comme les traces 13, 14 et 15. Une autre contrainte est la mémoire embarquée disponible pour stocker les traces au cours de l'exécution du robot. En conséquence, il manque des mesures pour appliquer de manière fiable l'EVT sur les traces 13, 14 et 15.

D'autre part, la trace12 ne passe pas les hypothèses de stationnarité et d'indépendance. La Figure 6.5 de la trace12 montre que les mesures suivent une tendance linéaire. Cette tendance linéaire est caractéristique de non stationnarité et de



(a) trace10 (b) trace10 modifiée.

FIGURE 6.4 – Les traces trace10 et trace10 modifiée.

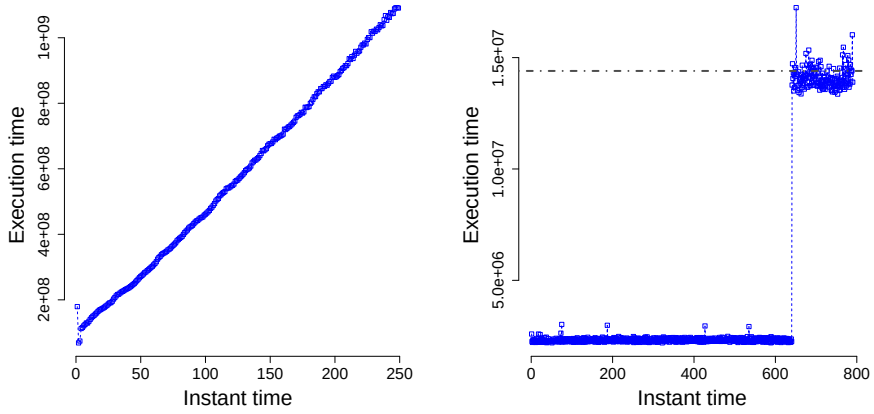
dépendance. La tâche associée à cette trace est celle responsable du SLAM, une tâche de cartograhie. Par conséquent, plus le robot parcourt de distance, plus la carte s'agrandit et la tâche de SLAM prend de temps pour s'exécuter. Le temps d'exécution de la tâche étant directement proportionnel à la taille de la carte, tant que la taille maximale n'est pas connue le temps d'exécution maximal ne peut être observé. En définitive, l'EVT n'est pas applicable dans ce cas.

Le cas de la trace16 ressemble au cas de la trace10 dans laquelle quelques pics divergent largement des temps moyens. La présence de ces pics rend l'EVT inapplicable et le WCET seuil résultant est très pessimiste, de l'ordre de 10^{13}%. On étudie alors la trace16 modifiée en ôtant ces 500 premières mesures. Dans ce cas l'hypothèse d'indépendance n'est pas vérifiée contrairement au cas de la trace16 non modifiée. Le WCET seuil dans le cas modifié est précis.

Enfin, la trace17, donnée dans la Figure 6.5, exhibe deux comportements bien différents avec des temps d'exécution bas dans un premier temps et des temps plus larges à la fin de la trace. En conséquence, l'hypothèse de stationnarité n'est pas vérifiée. On décide de tronquer la trace pour ne conserver que les temps d'exécution les plus larges i.e. le deuxième comportement de la tâche. L'application de l'EVT n'est pas différente du cas non tronqué et donne des pWCETs similaires. En revanche, le cas tronqué donne une trace fiable au regard des hypothèses de l'EVT. Ce cas de trace pose le problème du choix des conditions de mesure pour garantir la représentativité du pWCET.

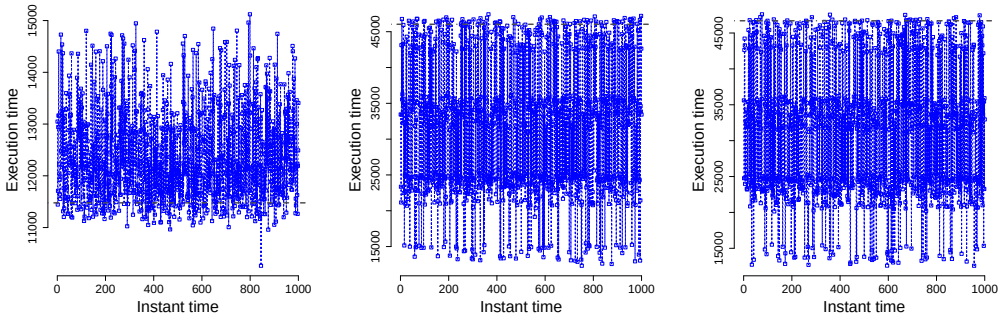
6.2.4 Graphical Processor Unit

Les statistiques moyennes des traces montrent que le cas où les données sont davantage dans le cache privé, trace18 (25%/75%), les temps d'exécution sont en moyenne de moitié plus petits que dans les deux autres cas qui ont des moyennes similaires. Comme le montre la Figure 6.6, excepté le cas de la trace18, les traces du cas de l'architec-



(a) trace12 (b) trace17

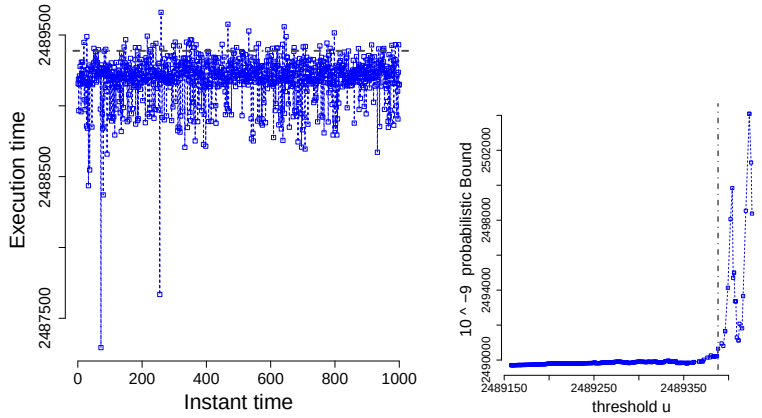
FIGURE 6.5 – Les cas limites des traces issues du cas du système embarqué robotique.



(a) trace18 (b) trace19 (c) trace20

FIGURE 6.6 – Traces issues du cas du GPU.

ture GPU exhibent des groupes de mesures avec des écarts assez conséquents entre les groupes. Par exemple, la trace19 montre un groupe de mesures autour de 45000 cycles, un autre autour de 35000 cycles, un autre autour de 25000 cycles et un dernier autour de 15000 cycles. Ces différents groupes de mesures peuvent provenir de quatre cas de diagrammes générés et l'exécution de la tâche exhibe un comportement différent selon la répartition des données en mémoire. Ainsi, dans ce cas encore, le choix des paramètres d'entrée de la tâche affecte la continuité des mesures et la représentativité du pWCET estimé résultant. Dans tous les cas, les pWCETs estimés sont considérés comme fiables. Seul le cas de la trace20 nécessiterait davantage de mesures pour passer l'hypothèse de convergence. Mille nouvelles mesures ont été réalisées permettant de passer l'hypothèse de convergence. Dans les cas des traces18 et 19, dans lesquels des groupes de temps d'exécution apparaissent, seules les mesures issues du groupe le plus élevé sont utilisées pour l'estimation du pWCET. Par conséquent, des mesures selon un paramètre d'entrée,



(a) trace22

(b) Variation du WCET seuil à 10^{-9} en fonction du seuil u.

FIGURE 6.7 – Analyse des résultats de la procédure d'exploration du seuil u dans le cas de la trace22.

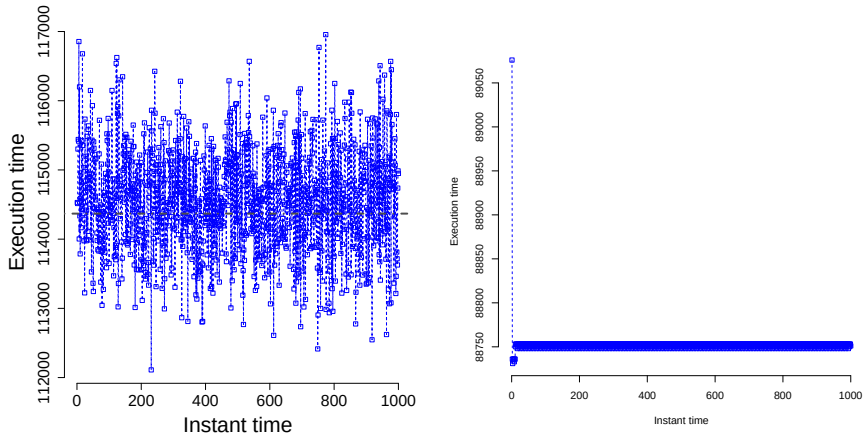
celui qui a conduit à ce groupe de valeurs, est suffisant pour garantir la représentativité et la fiabilité du pWCET. De plus, les WCETs seuil inférés sont précis dans les trois cas car les pessimismes sont de l'ordre de 1 et de 0.1%.

6.2.5 Field-Programmable Gate Array

Ce cas d'architecture donne des traces de mesures adaptées à l'application de l'EVT. Les traces 21 et 22 sont assez variables pour appliquer une modélisation probabiliste. Les pics de temps d'exécution sont assez nombreux dans le cas de la trace21 et ne divergent pas trop des valeurs moyennes. Comme le montre la Figure 6.7, l'algorithme de sélection de seuil dans le cas de la trace22 s'est arrêté assez rapidement retournant un seuil élevé et donc peu de pics sont considérés.

Néanmoins, le WCET seuil inféré avec ce seuil est proche de la valeur de convergence. Il en résulte que les WCETs seuil inférés dans les deux cas sont précis, surtout pour la trace21, et les intervalles de confiance fiables sont resserrés autour de leur WCET seuil respectif. Enfin, seul le pWCET de la trace21 est diagnostiqué comme fiable. Le cas de la trace22 requiert des mesures supplémentaires pour passer l'hypothèse de convergence.

Dans le cas des traces simulées, traces 23, 24 et 25, pour approcher le comportement d'un processeur randomisé, ces dernières passent toutes les hypothèses sauf celle de convergence. Alors que les traces sont de taille 1000, il manque des mesures pour obtenir des pWCETs fiables. La variabilité des mesures dans ces cas étant plus importante que dans les autres traces issues d'architectures COTS, il est nécessaire de réaliser davantage de mesures pour garantir la convergence des pWCETs estimés.



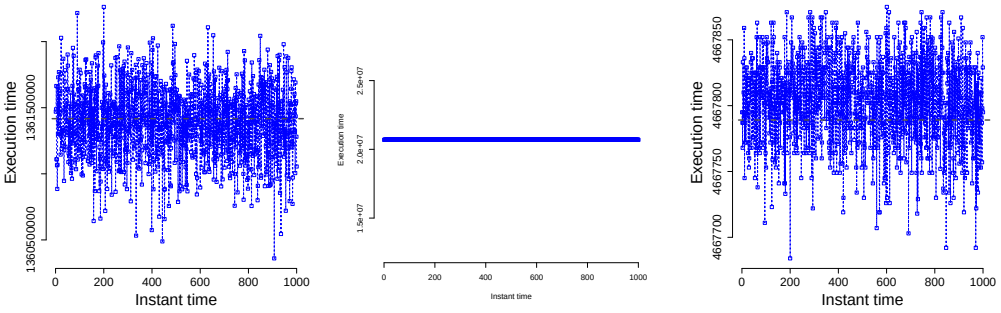
(a) trace26 (b) trace27

FIGURE 6.8 – Traces issues du cas de l'architecture manycœur avec les caches privés activés.

6.2.6 Architecture manycœur

Le premier cas d'étude porte sur les contentions introduites par les exécutions en parallèle de tâches. Les traces 26 et 28 représentent les cas avec contentions et les traces 27 et 29 sans. De plus, dans les traces 26 et 27 le cache privé de chaque cœur est activé alors qu'il est désactivé pour les traces 28 et 29. La Figure 6.8 montre bien la différence de variabilité du temps d'exécution avec ou sans contention. Les temps d'exécution moyens avec contentions sont plus élevés d'environ 1.5 fois par rapport au cas d'activation du cache respectif sans contentions. Les temps d'exécution moyens sans les caches privés sont environ 5 fois plus élevés que dans les cas de contention respectifs avec les caches privés activés. Les niveaux de variabilité dans les cas avec contentions sont similaires. En conséquence, l'EVT est inapplicable dans les cas sans contention et fiable dans le cas avec contentions de la trace28. La trace26 ne donne pas un pWCET fiable car la trace ne passe pas les hypothèses d'indépendance et de convergence. Ce résultat est à discuter car la trace26 ressemble sensiblement à la trace28 qui est indépendante. Enfin, il resterait à réaliser davantage de mesures dans le cas de la trace26 pour obtenir un pWCET fiable. Les traces 26 et 28 ne montrent pas de pics de temps d'exécution trop divergents des valeurs moyennes, et en conséquence les WCETs seuil inférés sont précis, leur pessimisme est de l'ordre de 0.1%. L'architecture manycœur considérée est sensible aux interférences entre les tâches mais limite la divergence des temps d'exécution.

Le deuxième cas d'étude réalisé à partir de l'architecture manycœur produit des traces avec des comportements bien différents. Il s'agit de traces de mesures de tâches correspondant à différents étages d'un algorithme de traitement d'image. Pour chaque nouvelle mesure une nouvelle image à analyser est générée de manière aléatoire. L'image est la même pour chaque étage de l'algorithme. Les traces de la Figure 6.9 montrent que des étages sont sensibles à la variabilité de l'image d'entrée et d'autres non. Ainsi,



(a) trace30 (b) trace31 (c) trace32

FIGURE 6.9 – Traces issues du cas d'étude d'un algorithme de traitement d'image exécuté sur l'architecture manycœur.

les temps d'exécution des traces 30, tâche *detectCosmicRay*, et 32, *substractSuperBias* sont variables alors que le temps d'exécution de la trace 31, tâche *detectSaturation*, est constant. En conséquence, les pWCETs issus des traces 30 et 32 sont fiables et l'EVT est inapplicable dans le cas de la trace31. Les WCETs seuils dans les deux cas sont très précis et les intervalles, de confiance fiables resserrés. Néanmoins, la queue de distribution empirique de la trace32 est très courte et fournit un pWCET avec un paramètre de forme $\xi = -0.53$. Or, la procédure MLE ne converge pas vers les vrais paramètres pour $\xi > -0.5$. Etant donné que le paramètre de forme estimé est très proche de 0.5, on décide de garder le pWCET estimé. Pour le cas de la trace31 qui est constante par rapport aux paramètres d'entrée, sachant que le nombre d'entrées différentes testées est élevé, considérer la valeur mesurée comme le WCET de la tâche est statistiquement fiable.

6.3 Discussion

L'application de DIAGXTRM à divers traces issues de processeurs COTS différents a permis de montrer l'intérêt d'utiliser un tel cadre de travail. La variabilité des temps d'exécution favorisée par la complexité des processeurs actuels rend l'estimation d'un WCET difficile. L'application de l'EVT utilise cette variabilité pour estimer un pWCET duquel on infère des WCETs seuil relativement à une probabilité. A travers ce chapitre d'application, nous avons mis en évidence divers cas limites de l'application de DIAGXTRM. Certaines traces n'exhibaient pas assez de variabilité, il était donc impossible d'estimer un pWCET. Ces cas sont repérés par un niveau d'indépendance $cl_{2.1}$ nul. D'autres traces présentaient assez de variabilité, mais soit elles suivaient une tendance linéaire, soit plusieurs comportements temporels de la tâche étaient visibles. Dans ces cas, le niveau de stationnarité $cl_{1.1}$ est nul. Des traces étaient trop variables et nécessitaient davantage de mesures pour passer l'hypothèse de convergence donnée par le niveau cl_4. Enfin, si une trace ne présente aucune de ces caractéristiques, les niveaux de confiance

restant $cl_{1.2}$, $cl_{2.2}$ et cl_3 sont rarement nuls.

Le degré de variabilité des traces est différent en fonction des architectures et des conditions de mesure. Parmi les conditions de mesure possibles, on trouve les interférences matérielles subies par la tâche étudiée. Comme dans le cas des traces 26 et 28 où des contentions sont introduites et les mesures sont variables contrairement aux traces 27 et 29 sans contention. D'autres conditions de mesure concernent les paramètres d'entrée de la tâche. Si les paramètres d'entrée sont modifiés au cours des mesures, la trace exhibe de la variabilité fonctionnelle. Pour certaines tâches, comme des étages de l'algorithme de traitement d'image, les tâches peuvent ne pas être sensibles aux paramètres d'entrée. Ces conditions peuvent être imposées par un utilisateur ou dépendantes de l'environnement d'exécution comme dans le cas du système robotique. Enfin, les architectures isolent de manière différente les interférences au niveau matériel et les temps d'exécution extrêmes peuvent être davantage contenus comme dans le cas de l'architecture manycœur au contraire de l'architecture multicœur.

Par conséquent, le choix des conditions de mesure est déterminant pour garantir la représentativité du pWCET estimé. En effet, le pWCET modélise les conditions pires cas de la trace de mesures dont il est issu. Certaines conditions, soit fonctionnelles, entrées de la tâche étudiée, soit systèmes, interférences matérielles, peuvent faire varier sensiblement les statistiques moyennes d'une trace de mesures. Ainsi, pour garantir la sûreté et la modélisation du pire cas, le pWCET doit être représentatif des pires conditions de mesure. Certains travaux ont choisi de randomiser le matériel [WKL+13] afin d'assurer d'avoir exploré le plus d'états matériels. Or, nous venons d'expérimenter la variabilité du temps d'exécution sans la randomisation du matériel. Les processeurs randomisés représentent un sous ensemble de processeurs. Nous faisons alors le choix d'étudier l'influence des interférences et des paramètres d'entrée comme conditions de mesure. Les conditions de mesure choisies pour réaliser une trace constituent un scénario S. La détermination d'un tel scénario S est critique pour la représentativité du pWCET estimé par DIAGXTRM comme illustré dans la Figure 6.10. Les chapitres suivants vont étudier la détermination d'un scénario S^{worst} qui doit mener aux temps d'exécution les plus longs de la tâche étudiée et à un pWCET représentatif.

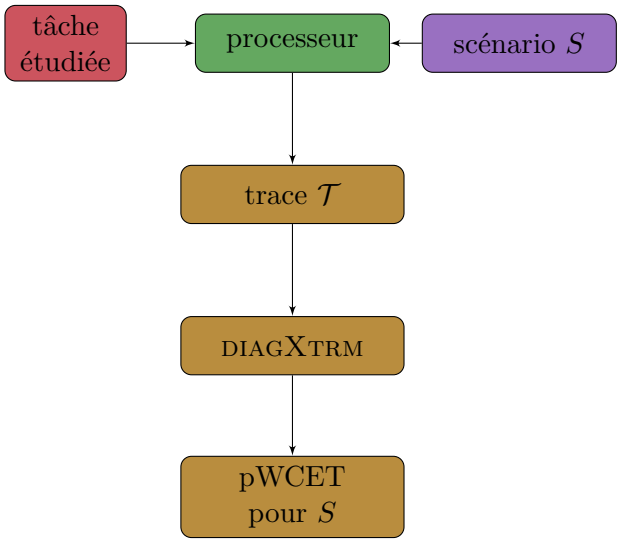


Figure 6.10 – Approche MBPTA diagXtrm.

Chapitre 7

Conditions d'exécution d'une trace pour l'estimation de pire temps d'exécution probabiliste robuste aux exécutions en parallèle

Sommaire

Résumé 3
Le chapitre précédent a mis en évidence certains cas limites de systèmes pour lesquels l'application de l'EVT n'est pas acceptable car elle retourne des pWCETs non fiables ou alors l'estimation n'est pas possible par manque de variabilité du temps d'exécution. Néanmoins, l'introduction d'interférences matérielles peut contribuer à favoriser la variabilité du temps d'exécution. Ces interférences peuvent être discrètes et peu fréquentes comme des rafraichissments mémoire ou des interruptions, et elles entraînent la nécessité d'effectuer beaucoup de mesures. Un autre type d'interférences vient de l'exécution en parallèle de tâches sur des plateformes contenant plusieurs cœurs. Les interférences issues de la parallélisation massive d'application génèrent des mesures de temps d'exécution variables et relativement continues et contribuent à ralentir l'exécution des tâches. Ainsi, pour un niveau élevé d'interférences matérielles le temps d'exécutione est en moyenne plus élevé et plus variable ce qui favorise l'application de l'EVT et l'obtention d'un pWCET plus représentatif du pire cas. Ce chapitre a donc pour objectif de déterminer le scénario de mesure qui donne le plus haut niveau de contentions dues aux exécutions en parallèle et répond aux besoins de fiabilité et de représentativité du pWCET.

7.1 Introduction

Les processeurs multi-manycœur actuels du commerce fournissent une ressource de calcul inimaginable encore il y a une dizaine d'années pour les systèmes temps réel. Plusieurs cœurs sont disponibles pour des exécutions de tâches en parallèle au lieu de les exécuter séquentiellement. En plus de pouvoir exécuter les tâches en parallèle, les composants d'optimisation comme les mémoires caches et les exécution pipelinées possibles avec ce type de processeurs augmentent davantage les performances de calcul mais au prix d'une prédictibilité difficile à garantir.

En particulier, les tâches s'exécutant en parallèle interfèrent entre elles via les ressources matérielles partagées comme les mémoires caches et les bus de communication. En conséquence, l'exécution des tâches est ralentie due aux accès concurrents aux ressources partagées pouvant créer des nœuds d'étranglement. Des techniques d'isolation et des politiques d'exécution déterministiques sont possibles, comme la politique d'arbitration d'accès au bus partagé Round Robin, mais elles sont au prix d'implémentations coûteuses et de performances dégradées. Une alternative est de prendre en compte les interférences dans l'estimation du pWCET.

Ce type d'interférences était généralement pris en compte dans les analyses d'ordonnançabilité (Section 1.3.7) car les interférences sont trop complexes à considérer dans l'estimation du WCET. Les interférences et les contentions ne ralentissent pas seulement l'exécution des tâches en parallèle mais introduisent également davantage de variabi-

lité au niveau du temps d'exécution. Les processeurs multi et manycœur, en offrant la possibilité de paralléliser les exécutions des tâches, introduisent du non déterminisme temporel. Ainsi, le modèle probabiliste du pWCET permet de modéliser la variabilité du temps d'exécution due aux interférences et ainsi de réduire la complexité des analyses d'ordonnancement pour ce genre de processeurs.

Outre la réduction de la complexité des analyses d'ordonnancement, la prise en compte des interférences dans l'estimation du pWCET permet de garantir la représentativité du pWCET. En effet, puisque les interférences ralentissent les exécutions, elles constituent des conditions d'exécution à considérer pour exhiber un pire scénario de mesure. Dans ce chapitre, nous traitons le cas du ralentissement de l'exécution de tâches exécutées en parallèle. En particulier, une méthodologie pour déterminer un pire scénario S^{worst} défini par un ensemble de tâches qui vont potentiellement interférer avec la tâche dont on souhaite estimer le pWCET et va favoriser les temps d'exécution les plus longs.

En particulier, les tâches étudiées sont monochemin et l'analyse proposée se veut applicable à des processeurs multicœur avec une hierarchie mémoire. Le processeur étudié dans ce chapitre présente un niveau de cache partagé avec un bus d'accès dont la politique d'arbitration est basée sur la politique First In First Out FIFO. Chaque tâche est attribuée à un cœur et ne peut pas migrer comme pour un ordonnancement multiprocesseur partitionné. Les tâches ne peuvent pas être préemptées lors de leur exécution.

7.2 Analyse de graphe pour déterminer les tâches pouvant s'exécuter en parallèle de la tâche étudiée

7.2.1 Graphes dirigés acycliques

Dans le cas de contraintes de précédence entre les tâches d'une application temps réel, l'application peut être représentée par un *graphe dirigé acyclique* (Directed Acyclic Graph DAG) $G(V, E)$, [ABS13, PMN+16]. V est l'ensemble des N *nœuds* et E est l'ensemble des *arcs dirigés*. Un nœud n_i de V est une tâche et tous les travaux qui lui sont associés. Le nœud peut être pondéré par $w(n_i)$ qui peut être le pWCET ou le WCET de la tâche n_i.

Les arcs représentent l'ordre d'exécution entre les tâches. Un arc $e_{i,j}$ connecte les deux nœuds n_i et n_j, avec n_i précédant n_j ; n_i *est le parent de* n_j *et donc* n_j *est le fils de* n_i. Un nœud sans parent est un *nœud d'entrée*, et un nœud sans fils est un *nœud de sortie*.

Les contraintes de précédence encodées dans les DAGs imposent que le nœud ne peut commencer son exécution avant que tous ses parents n'aient fini leur exécution. Les arcs peuvent également être pondérés par $w(e_{i,j})$ représentant le délai de communication entre les exécutions des tâches n_i et n_j. Un *chemin* depuis n_i vers n_j dans un DAG existe si et seulement s'il est possible d'atteindre n_j depuis n_i. Le chemin est l'ensemble des nœuds depuis n_i vers n_j et la suite des arcs, $\{\{n_i, n_k, n_r, \dots n_s, n_j\}, \{e_{i,k}, e_{k,r}, \dots e_{s,j}\}\}$.

Exemple 7.1
Un exemple de DAG avec dix tâches est donné dans la Figure 7.1. Le nœud d'entrée est n_0, qui est le parent de n_1, n_2 et n_3. Le nœud de sortie est n_{10}, qui est le fils de n_6, n_7 et n_8. Trois chemins possibles à partir de n_0 vers n_9 sont par exemple : $\{\{n_0, n_1, n_6, n_9\}, \{e_{0,1}, e_{1,6}, e_{6,9}\}\}$, $\{\{n_0, n_2, n_4, n_7, n_9\}, \{e_{0,2}, e_{2,4}, e_{4,7}, e_{7,9}\}\}$ *et* $\{\{n_0, n_3, n_4, n_7, n_9\}, \{e_{0,3}, e_{3,4}, e_{4,7}, e_{7,9}\}\}$.

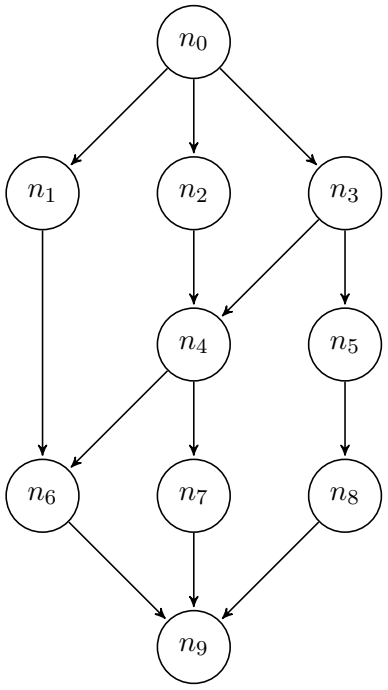


FIGURE 7.1 – Exemple d'un DAG G [ABS13].

7.2.2 Contentions dues aux exécutions parallèles

Connaissant les contraintes de précédence décrites dans le DAG de l'application temps réel, les tâches qui peuvent s'exécuter en parallèle sur différents cœurs sont indépendantes. Avec le DAG $G(V, E)$, deux tâches n_i et n_j sont indépendantes $n_i \nabla n_j$ si et seulement s'il n'existe pas de chemin depuis n_i vers n_j dans G. A l'exécution, les tâches indépendantes exécutées en parallèle n_i et n_j interfèrent entre elles en introduisant des contentions ; n_i et n_j sont contendantes.

A partir du DAG G, il est possible de déterminer l'ensemble des tâches potentiellement contendantes avec n_i : $\Gamma(n_i)$. Ces tâches peuvent s'exécuter en parallèle de n_i ou de manière équivalente sont indépendantes à n_i :

$$\Gamma(n_i) \stackrel{def}{=} \{n_j \in G \mid n_i \nabla n_j\}.$$

Rechercher les tâches contendantes est équivalent à appliquer le complément de la clôture transitive non dirigée du DAG G. En effet, la clôture transitive consiste à ajouter des arcs entre deux nœuds s'ils ne sont pas indépendants i.e. s'il existe un chemin depuis l'un vers l'autre. En prenant le graphe complémentaire de la clôture transitive non dirigée du DAG, alors seules les tâches indépendantes partagent un arc. Le graphe résultant de cette opération est appelé le graphe d'indépendance $\overline{G}$, comparé au graphe initial de précédence G.

Exemple 7.2
La procédure pour déterminer le graphe d'indépendance est appliqué à l'exemple d'application parallèle de l'Exemple 7.1 ; le graphe résultant est donné dans la Figure 7.2. Dans le graphe d'indépendance, n_1 est indépendant de $n_2, n_3, n_4, n_5, n_7, n_8$. L'ensemble des tâches potentiellement contendantes de n_1 est l'ensemble $\Gamma(n_1) = \{n_2, n_3, n_4, n_5, n_7, n_8\}$, signifiant que ces tâches peuvent être exécutées en parallèle de n_1 et vont potentiellement interférer avec n_1 pendant son exécution.

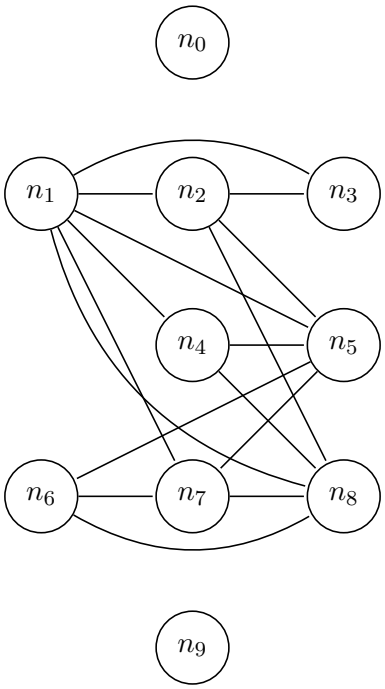


FIGURE 7.2 – Graphe d'indépendance $\overline{G}$.

7.2.3 Liste des tâches contendantes

L'analyse d'indépendance précédente fournit l'ensemble des tâches qui peuvent s'exécuter en parallèle de n_i. Néanmoins, avec un processeur à M cœurs, jusqu'à $M-1$ tâches indépendantes deux à deux peuvent s'exécuter en parallèle de n_i. Un ensemble

de $c-1$ tâches $n_{j_1}, \ldots, n_{j_{c-1}}$ sont indépendantes deux à deux et indépendantes de n_i si et seulement si $\forall k, l \in [\![1; c-1]\!],\ k \neq l,\ n_i \nabla n_{j_k},\ n_{j_k} \nabla n_{j_l}$. Au niveau du graphe $\overline{G}$, un ensemble de tâches indépendantes deux à deux signifie que tous les couples de tâches de l'ensemble partagent un arc.

Un ensemble de c tâches, incluant n_i, dans lequel chaque couple de tâches partage un arc est une clique $c - clique(n_i)$; la clique relative à n_i est de taille $c = |c - clique(n_i)|$. Pour chaque tâche n_i, il existe une taille maximale de clique. $clique^{\max}(n_i)$ est la clique maximale de n_i dans $\overline{G}$; $clique^{\min}(n_i)$ est la plus petite clique. La clique maximale et la clique minimale ne sont généralement pas uniques dans $\overline{G}$.

Exemple 7.3
A partir de l'application temps réel de l'Exemple 7.1 et son graphe d'indépendance $\overline{G}$ donné dans la Figure 7.2, n_1, n_2 et n_3 partagent un arc entre chacun et composent donc une $3 - clique(n_1) = \{n_1, n_2, n_3\}$; n_1, n_2 et n_3 pourraient s'exécuter en parallèle sur un processeur avec 3 cœurs. C'est une clique maximale de n_1 mais elle n'est pas unique car il en existe d'autres de même taille comme $\{n_1, n_2, n_5\}$. Au contraire, l'ensemble $\{n_1, n_2, n_4\}$ ne forme pas une clique car tous les couples de tâches ne partagent pas un arc dans $\overline{G}$.

La $c - clique(n_i)$ donne les $c-1$ tâches $c - clique(n_i)/n_i$ i.e. $c - clique(n_i)$ sans n_i, qui sont les tâches potentiellement contendantes avec n_i. En conséquence, tous les $c - cliques(n_i)/n_i$ pour $c \in [\![clique^{\min}(n_i)|, |clique^{\max}(n_i)|]\!]$ donne l'ensemble complet des ensembles de tâches contendantes de n_i.

Néanmoins, pour un processeur à M cœurs seulement $M-1$ tâches peuvent s'exécuter en parallèle de n_i. Alors, on ne considère pas les cliques de taille supérieure à M.

Il est raisonnable d'affirmer que si plus de tâches s'exécutent en parallèle, plus d'interférences auront lieu au niveau du matériel. Afin de considérer le pire scénario de mesure S^{worst} par rapport aux interférences matérielles dues aux exécutions en parallèle, la liste des ensembles de tâches contendantes de n_i ne doit être composée que des ensemble de tâches les plus grands. Pour n_i, une telle liste est la liste des tâches contendantes de n_i notée $ContenderList(n_i)$ définie comme :

$$ContenderList(n_i) \stackrel{def}{=} \{c - clique(n_i)/n_i \ : \\ c = \min(|clique^{\max}(n_i)|, M)\}.$$

Les ensembles de tâches dans $ContenderList(n_i)$ sont tous de taille $\min(|clique^{\max}(n_i)| - 1, M-1)$.

La recherche d'une telle liste pour chaque tâche $n_i \in G$ correspond à la procédure nommée ContenderListSearch.

Lemme 7.1 (ContenderListSearch)
Considérant une application temps réel avec des contraintes de précédence données par le DAG $G(V, E)$ s'exécutant sur un processeur à M cœurs. La recherche de la liste de l'ensemble des tâches contendantes de n_i a une complexité pire cas en $O(3^{N/3})$, où N est le nombre de nœuds dans G.

Preuve 1. *Comme exposé précédemment, la recherche de la liste des ensembles de tâches contendantes pour toutes les tâches de G est équivalente à rechercher toutes les cliques maximales dans $\overline{G}$. La complexité pire cas d'une telle recherche est en $O(3^{N/3})$, [TTT06].* ■

Exemple 7.4
La clôture transitive du DAG de l'Exemple 7.1 donne le graphe d'indépendance de la Figure 7.2. Pour n_1 la taille de la clique maximale est $|clique^{\max}(n_1)| = 3$ et $clique^{\max}(n_1)$ n'est pas unique. La liste des 3−cliques contenant n_1 est : $\{\{n_1, n_2, n_3\}$, $\{n_1, n_2, n_5\}$, $\{n_1, n_4, n_5\}$, $\{n_1, n_4, n_8\}$, $\{n_1, n_5, n_7\}$, $\{n_1, n_7, n_8\}\}$. Pour $M = 3$, la $ContenderList(n_1)$, liste des cliques de taille 3 sans n_1, est : $\{\{n_2, n_3\}$, $\{n_2, n_5\}$, $\{n_4, n_5\}$, $\{n_4, n_8\}$, $\{n_5, n_7\}$, $\{n_7, n_8\}\}$. Dans le cas où $M = 4$, comme la clique maximale est de taille 3, la $ContenderList(n_1)$ est la même que dans le cas où $M = 3$.

7.3 Heuristique pour déterminer le scénario de mesures rendant compte du niveau d'interférences matérielles le plus élevé

La procédure précédente permet de déterminer l'ensemble des ensembles de tâches contendantes avec la tâche étudiée n_i $ContenderList(n_i)$. Chaque ensemble de tâche de $ContenderList(n_i)$ exécuté en parallèle de n_i représente un scénario de mesure. Néanmoins, la $ContenderList(n_i)$ peut être de grande taille et tester tous les scénarios de mesure de la liste peut relever de l'explosion combinatoire. L'objectif est donc de déterminer l'ensemble de tâches dans $ContenderList(n_i)$ qui va introduire le plus d'interférences matérielles et ainsi favoriser l'observation des temps d'exécution de n_i les plus larges. Ce scénario de mesure représentera notre pire scénario de mesure S^{worst}.

7.3.1 Classification

Pour résoudre ce problème, nous introduisons une procédure d'apprentissage pour classer les ensemble de tâches de la $ContenderList(n_i)$ selon leur degré d'interférence. Cette procédure de classification est appelée ContenderClassification. La classification des tâches est basée sur leur degré de contention. Plusieurs approches existantes ont tenté de caractériser le degré de contention de tâches logicielles [MTS11, TMS11]. Elles représentent le niveau de contention d'une tâche selon ses accès aux ressources partagées comme les bus mémoire et les mémoires caches. Dans cette étude, nous nous

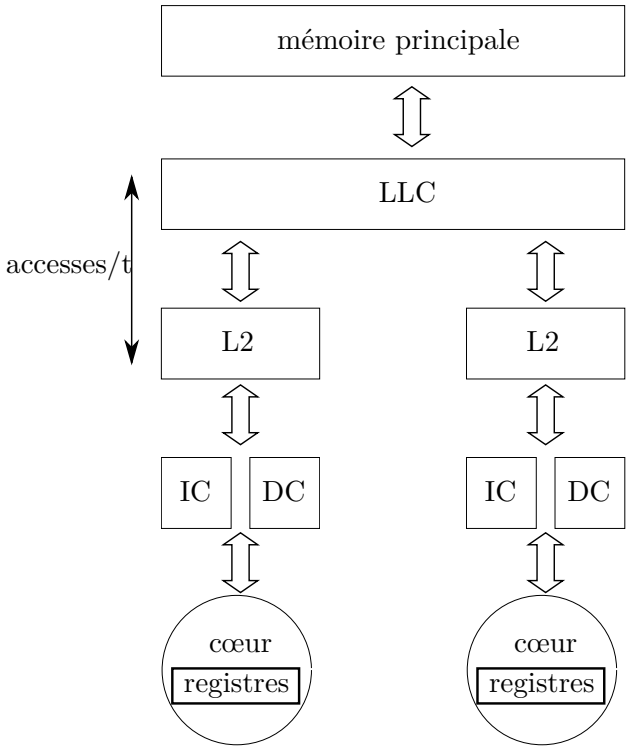


FIGURE 7.3 – Mesure du débit d'accès mémoire dans le cas de hiérarchie mémoire de la Figure 1.2.

concentrons sur les contentions au niveau de la hiérarchie mémoire et des bus pour des applications temps réel parallélisables.

Inspiré par ces approches [TMS11], nous mesurons le nombre d'accès à la mémoire partagée *accesses* par unité de temps t, t en nombre d'instructions ou en unité de temps usuelle, afin de caractériser le niveau d'interférence d'une tâche. Ces accès peuvent être des blocs mémoire chargés depuis le cache partagé ou bien des écritures vers le cache partagé. Cette métrique correspond au débit d'accès mémoire *mem_band* (pour memory bandwidth usage) :

$$mem\_band \stackrel{def}{=} accesses/t.$$

La mesure du débit d'accès mémoire est illustrée dans la Figure 7.3.

Pour des tâches introduisant des interférences au niveau des ressources matérielles partagées, le débit d'accès mémoire de la tâche *mem_band* sera élevé. Les seules mesures de cache *misses* ne sont pas suffisants car une tâche peut bénéficier de mécanisme de préchargement de ses données et instructions en cache de manière à ce que la tâche n'accède qu'une seule fois à la mémoire partagée pendant son exécution. Ainsi, son débit

d'accès mémoire sera presque nul i.e. la tâche n'interfère pas avec les autres tâches.

Les tâches peuvent avoir des temps d'exécution moyens différents et seront ainsi impactées différemment par les contentions. Considérons une tâche de longue durée s'exécutant en parallèle de tâches de courte durée. Si les tâches sont démarrées en même temps, alors la plus longue tâche s'exécutera seule vers la fin de son exécution et ne sera plus interférée. Pour résoudre ce problème, de manière non réaliste, les tâches sont exécutées en continue pendant un nombre d'itérations donné de la tâche étudiée n_i. Cette condition d'exécution force de manière pessimiste un grand nombre d'interférences et assure la *sûreté* de l'analyse proposée.

Analyse de complexité Une exécution par tâche est suffisante pour caractériser son degré de contention. Grâce à la connaissance du débit d'accès mémoire de chaque tâche, chaque ensemble de tâches de la $ContenderList(n_i)$ $\forall i$ est classé de manière croissante selon la somme des débits d'accès de chaque tâche. Il est possible de sommer les débits car les tâches sont indépendantes. La caractérisation des tâches et la classification des ensembles pour toutes les N tâches de G est en $\sum_{i=1}^{N} t_{n_i}$, où t_{n_i} est le temps d'exécution de n_i en isolation. Comme $\sum_{i=1}^{N} t_{n_i} < N \times T$, avec T le temps d'exécution le plus grand pour toutes les tâches de G, la complexité de la procédure ContenderClassification pour toutes les tâches de G est en $O(N)$.

Cette opération de classification permet de déterminer l'ensemble de tâches le plus contendant de la $ContenderList(n_i)$ $\forall i$, et ainsi d'exhiber un pire scénario S^{worst} pour chaque tâche n_i. Une trace de mesures pour tâche n_i est réalisée conformément aux conditions d'exécution spécifiées par le scénario S et les exigences de l'EVT décrites aux Chapitres 5 et 6. La complexité d'exhiber un pire scénario de contention S^{worst} est de réaliser des mesures dans ces conditions pour toutes les tâches de l'application est en $O(N)$.

7.3.2 Analyse du pire scénario de contention

Pour faciliter la lecture nous introduisons quelques notations relatives à l'exécution de tâches sur un processeur. Une tâche n_i s'exécutant sur un processeur composé de M cœurs Φ^M est noté $\Phi^M(n_i)$. Deux tâches n_i et n_j s'exécutant en parallèle sur Φ^M est noté $\Phi^M(n_i \parallel n_j)$; n_i et n_j sont exécutées sur deux cœurs différents et ne migrent pas. La notation $\parallel$ est également applicable pour q tâches $n_1 \parallel \ldots \parallel n_q$.

Nous proposons une analyse du pire scénario de contention dont les étapes sont :

1. ContenderListSearch : $\forall n_i \in G$, $ContenderList(n_i)$ est le résultat de la recherche qui donne l'ensemble de tâches qui peuvent s'exécuter en parallèle de n_i et dont la complexité est donnée dans le Lemme 7.1.
2. *Mesure du degré de contention* : $\forall n_i \in G$, $mem\_band(n_i)$.
3. ContenderClassification : $\forall n_i \in G$, $sort(ContenderList(n_i))$ ordonne par ordre décroissant les ensembles de tâches selon leur somme des débits d'accès.
4. *Trace selon le pire scénario de contention* : $\forall n_i \in G$ la trace $\mathcal{T}(n_i)$ est réalisée selon le pire scénario de contention S^{worst} : $\Phi^M\ (n_i || sort(ContenderList(n_i))[1])$.

7.4 Simulateur d'exécution en parallèle de tâches pour valider l'algorithme proposé

7.4.1 Application parallèle

Le cas d'étude de l'application avionique FAS [SPD+14] est utilisé pour illustrer l'approche proposée. L'application est composée de 19 tâches et de 21 contraintes de précédence. Le DAG de l'application $G(V, E)$ est donné dans la Figure 7.4 avec une description détaillée des contraintes de précédence entre les tâches.

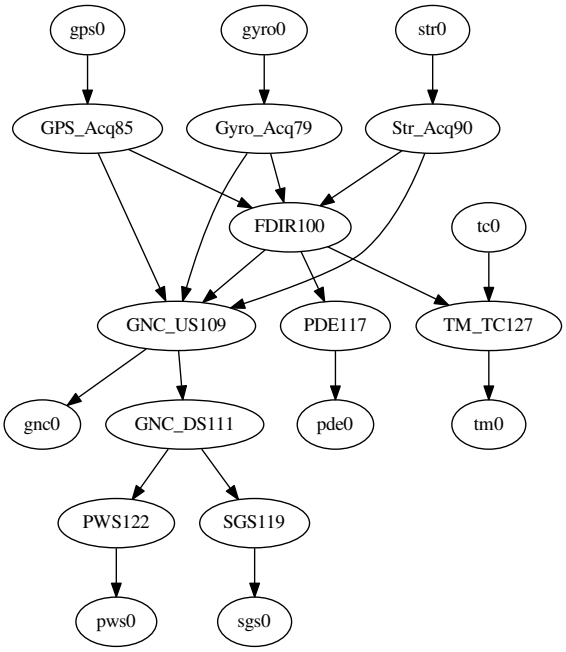


FIGURE 7.4 – DAG du cas d'étude FAS.

7.4.2 Simulateur de plateforme multicœur

Les approches MBPTA ont été appliquées avec succès dans des études précédentes [GSM16a, BGS+16, SGM17], mais elles ont montré la difficulté d'isoler certains mécanismes du fait de la complexité des plateformes. A l'inverse, des exécutions basées sur des simulateurs permettent d'avoir le contrôle de la plateforme et de ses paramètres.

Nous implémentons un simulateur de processeur multicœur avec une hiérarchie mémoire qui soit assez réaliste mais aussi simple à exécuter pour étudier les effets des contentions. Les caractéristiques techniques du simulateur, comme le fonctionnement des mémoires caches, ont principalement été implémentées selon [PH96]. Il permet d'obser-

ver des ressources spécifiques comme les mémoires caches et le bus mémoire qui nous intéressent dans ce chapitre.

L'architecture utilisée dans ce cas d'étude est composée de 4 cœurs avec deux niveaux de cache, L1 et L2. Le cache L2 est partagé entre les 4 cœurs et est accédé via un bus. Les mémoires caches utilisent la politique de remplacement LRU et la politique d'écriture des données *write-through*. Le cache L1 est 4-A et est composé de 8 blocs par ensemble i.e. 32 blocs au total, et dont la pénalité d'accès est de 1 cycle. Le cache L2 est également 4-A avec 32 blocs par ensemble i.e. 128 blocs au total, et dont la pénalité d'accès est de 4 cycles. Le bus mémoire utilise la politique FIFO. Cette architecture est inspirée du processeur LEON4 [AGW10].

L'implémentation du simulateur permet de choisir d'autres paramètres pour les ressources précédentes. Le simulateur est principalement utilisé pour inspecter et isoler plus facilement les effets des contentions au niveau des mémoires partagées et du bus d'accès. Tout aussi simple que ce simulateur puisse l'être, l'exécution des tâches ne diffère que de très peu par rapport à un processeur réel.

7.4.3 Modèle d'exécution du processeur

Le simulateur de processeur multicœur utilisé dans ce cas d'étude est basé sur l'exécution en parallèle de threads correspondant à l'exécution d'un cœur. La machine à états de la Figure 7.5 décrit le modèle d'exécution de chaque thread cœur. Les threads cœur fonctionnent de manière synchronisée par un thread en charge de l'horloge du processeur. Un cycle d'horloge correspond à une action des cœurs en fonction de son état. Le premier cycle d'horloge de chaque thread cœur correspond à un accès à une adresse mémoire en lecture ou en écriture. Si l'adresse accédée n'est pas en mémoire, il faut la charger depuis les mémoires supérieures ce qui entraîne des pénalités d'accès aux mémoires et au bus d'accès mémoire. Lorsqu'il ne reste plus d'instruction à exécuter, les threads entrent dans une phase de terminaison notamment pour nettoyer les blocs de mémoire cache i.e. actualiser la valeur contenue dans un bloc de cache inférieur dans les niveaux supérieurs. Comme tous les threads sont synchronisés par une horloge commune, les pénalités de cache et de bus de chaque thread sont répercutées sur les autres threads i.e. les threads interfèrent entre eux.

De plus, des statistiques des exécutions sont disponibles dans des fichiers de trace, un fichier par cœur pour plus de simplicité. Dans le fichier de trace sont enregistrés pour chaque exécution de la tâche sur le cœur correspondant : le temps d'exécution de la tâche, les caches *hits* et *misses* à tous niveaux, le nombre d'accès au bus mémoire. Pour 500 exécutions de la tâche, le fichier de trace comportera 500 lignes. Le débit d'accès mémoire dans ce cas correspond au nombre d'accès au bus *accesses* divisé par le nombre d'instructions i.e. $mem\_band = \frac{accesses}{instructions}$. Dans le cas du simulateur proposé, le calcul du débit d'accès mémoire ne nécessite pas de tâche de fond pour monitorer en temps réel le nombre d'accès mémoire car le processeur simulé ne dispose pas de mécanismes de préchargement. Des développements futurs du simulateur pourraient intégrer de tels mécanismes pour coller au mieux avec les processeurs réels.

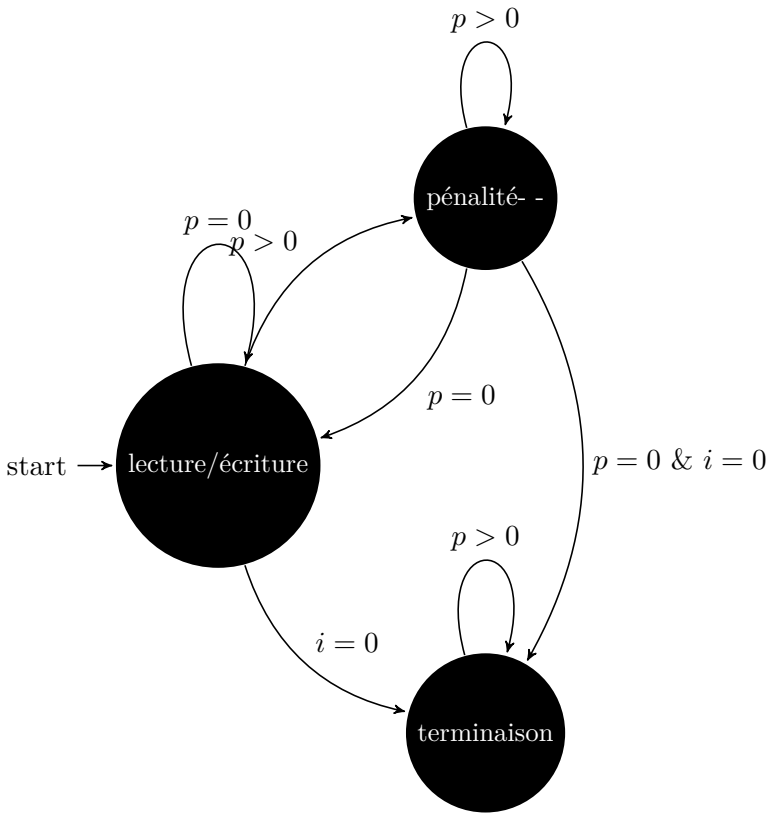


FIGURE 7.5 – Machine à états décrivant le modèle d'exécution d'un cœur du simulateur de processeur multicœur (p : pénalité due au cache/bus, i : nombre d'instructions restantes).

7.4.4 Simulation de tâche

Les tâches exécutées dans le simulateur sont abstraites selon une trace d'accès aux adresses mémoires en lecture ou en écriture. Chaque accès est associé à un bit pour indiquer si c'est un accès en lecture ou en écriture. Le simulateur prend en entrée un fichier par tâche.

Il est possible de récupérer de telles traces d'accès aux adresses mémoires à partir du code assembleur de la tâche [CSHB16]. Dans ce cas d'étude, nous avons été dans l'impossibilité d'utiliser une telle technique. En conséquence, les listes d'adresses sont générées de manière aléatoire selon une fonction de répartition D parmi l'ensemble D={loi Uniforme, loi Normale, loi de Pareto} La fonction de répartition joue sur la dispersion des adresses ainsi que sur leur fréquence d'accès. Les trois lois permettent de produire des comportements de tâche bien différents selon leurs accès mémoire. Chaque fonction est paramétrisée par un couple (a, b) où a est la localisation et b la dispersion des adresses dans la liste des instructions. Pour une loi uniforme a et b sont respectivement le borne inférieure et la borne supérieure de l'intervalle de distribution ; pour la loi Normale ce

sont la moyenne et la déviation standard ; pour la loi de Pareto ce sont le facteur d'échelle et la localisation. Une dispersion grande produit un grand nombre de cache *misses* et d'accès au bus. Ainsi, cela aboutit à une tâche qui interfère davantage avec les tâches s'exécutant en parallèle.

Le nombre d'instructions par tâche *instructions* est le quatrième paramètre qui décrit une tâche. Par exemple, pour une tâche dont les adresses sont distribuées selon une fonction D suivant une loi Normale de moyenne $a = 100$ et de déviation standard $b = 10$, et comportant $instructions = 100$ signifie que la tâche a 100 instructions à exécuter. Pour un nombre plus réduit d'instructions, la dispersion des adresses risque d'être moindre et ainsi son degré d'interférence.

Le générateur de tâche selon ce modèle permet de simuler rapidement divers comportements de tâche. La manière dont les tâches sont générées sous forme d'une séquence d'accès mémoire est proche des applications réelles. Néanmoins, les suites d'adresses attribuées par le compilateur dans le cas réel sont généralement plus déterministes i.e. l'adresse de l'instruction suivante a de fortes chances d'être l'adresse précédente plus un. Dans le futur, nous souhaiterions disposer d'un outil permettant de générer la trace des adresses mémoires accédées à partir du fichier source de la tâche et selon un compilateur donné.

7.5 Application de l'algorithme proposé en utilisant un cas d'étude industriel

Le cas d'étude FAS a été exécuté sur le simulateur de processeur multicœur présentée avec un processeur contenant 4 cœurs. Le cas d'étude et le simulateur ont été exécutés sur un processeur avec quatre Intel Core i3-4005U CPU @ 1.70GHz.

7.5.1 ContenderListSearch

Le résultat de la procédure ContenderListSearch appliquée au cas d'étude FAS est présenté dans la Figure 7.6.

La recherche de toutes les cliques pour toutes les tâches a pris 1700 μs. Comme la plupart des $ContenderList(n_i)$ pour tout i sont longues, nous donnons dans la Table 7.1 la taille des cliques maximales et la taille des $ContenderList(n_i)$ pour tout i. Comme le nombre de processeurs disponibles est de 4, les cliques de taille supérieure ne sont pas considérées. La plupart des tâches n_i possède une $ContenderList(n_i)$ contenant de nombreux ensembles de tâches, jusqu'à 50 pour la tâche gnc0. De telles tailles d'ensemble motivent le besoin de la procédure ContenderClassification afin d'éviter de tester tous les cas.

7.5.2 Analyse de contention basée mesures

Nous choisissons la tâche GPS_Acq85 comme tâche à étudier, étant un exemple représentatif de l'analyse de contention proposée.

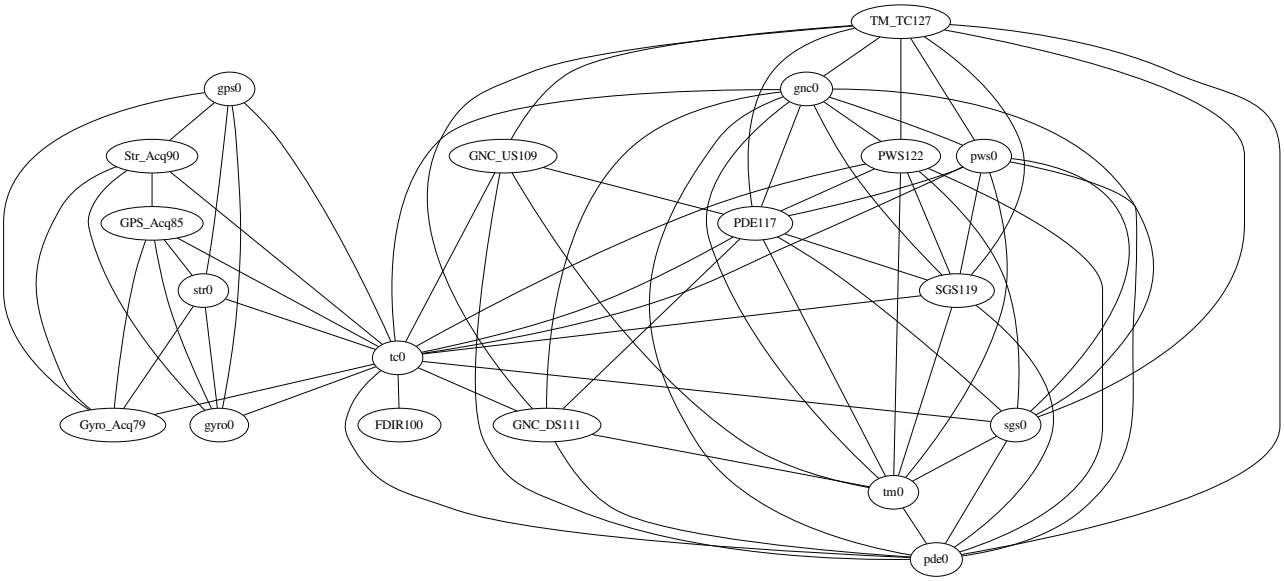


FIGURE 7.6 – Graphe des tâches indépendantes $\overline{G}$ du cas d'étude FAS.

Simulation des tâches

A partir de l'analyse d'indépendance (Figure 7.6) et du résultat de la procédure CONTENDERLISTSEARCH (Table 7.1), la *ContenderList*(GPS_Acq85) est {{Str_Acq90,tc0,gyro0}, {Str_Acq90,tc0,Gyro_Acq79}, {str0,tc0,gyro0}, {str0,tc0,Gyro_Acq79}}. Six tâches différentes se retrouvent dans la *ContenderList*(GPS_Acq85), et donc six tâches sont à simuler. Les six tâches sont simulées selon les paramètres donnés dans la Table 7.2. Les paramètres sont choisis de manière arbitraire afin d'illustrer l'analyse proposée. Le nombre d'instructions de chaque tâche est choisi en fonction du temps d'exécution des tâches donnés dans [SPD+14]. On établit une relation directe de proportionnalité entre le nombre d'instructions et le temps d'exécution spécifié pour chaque tâche.

Mesure du degré de contention

Toutes les tâches sont d'abord exécutées en isolation pour mesurer leur degré de contention i.e. le débit d'accès aux ressources matérielles partagées. Les résultats des mesures en isolation sont données dans la Table 7.3. Conformément au simulateur, les temps d'exécution sont en cycle. Un rang est attribué à chaque tâche selon leur débit d'accès mémoire ; le rang 1 étant pour le débit le plus élevé. Le classement montre que la tâche la plus contendante est gyro0 bien que son nombre d'instructions soit bas.

Le classement des tâches permet de procéder au CONTENDERCLASSIFICATION i.e. le classement des ensembles de tâches dans la *ContenderList*(GPS_Acq85). Le clas-

tâche	taille de la (des) clique(s) maximale(s)	nombre d'ensembles de tâches
gps0	4	4
gyro0	4	4
str0	4	4
GPS_Acq85	4	4
Gyro_Acq79	4	4
Str_Acq90	4	4
FDIR100	2	1
tc0	4	26
GNC_US109	3	6
PDE117	4	31
TM_TC127	4	22
gnc0	4	50
GNC_DS111	4	6
pde0	4	31
tm0	4	22
PWS122	4	28
SGS119	4	28
pws0	4	28
sgs0	4	28

TABLE 7.1 – Résultats de la procédure CONTENDERLISTSEARCH appliquée au cas d'étude FAS.

task	D	(a,b)	n_{instr}	address range
GPS_Acq85	normale	(320,20)	300	[260 ;390]
Str_Acq90	normale	(50,9)	300	[21 ;81]
tc0	pareto	(1.5,90)	100	[91 ;174]
gyro0	uniforme	(200,250)	100	[200 ;249]
Gyro_Acq79	uniforme	(5,20)	300	[5 ;19]
str0	pareto	(2,400)	100	[401 ;409]

TABLE 7.2 – Paramètres de simulation des tâches pour l'analyse de contention de GPS_Acq85.

sement ordonne les ensembles de tâches du plus contendant ts1 (task set 1) au moins contendant ts4 : $sort(ContenderList$(GPS_Acq85) ={ts1={Str_Acq90,tc0,gyro0}, ts2={str0,tc0,gyro0}, ts3={Str_Acq90,tc0,Gyro_Acq79}, ts4={str0,tc0,Gyro_Acq79}}. Le classement est tel que ts1 a la plus petite somme de rang de tâche $2+3+1=6$, etc.

Chaque ensemble de tâche ts$i$, est un scénario Si de mesure telle que la tâche GPS_Acq85 est exécutée en parallèle des tâches de l'ensemble : Φ^4 (GPS_Acq85||ts$i$). Les statistiques moyennes des traces issues de chaque scénario sont données dans la Table 7.4.

Les traces de temps d'exécution Si proviennent de l'exécution consécutive de GPS_Acq85 pendant un certain nombre d'itérations selon Φ^4 (GPS_Acq85||ts$i$). Pour l'analyse du pire scénario de contention, il n'est nécessaire que de considérer le premier scénario $S1$ i.e. Φ^4 (GPS_Acq85||Str_Acq90||tc0||gyro0). Néanmoins, nous avons également évalué les autres scénarios S_i à titre de comparaison.

tâche	temps d'exécution	L1 *hits/misses*	L2 *hits/misses*	*accesses*	*mem_band*	rank
GPS_Acq85	1025	123/177	93/84	204	0.68	-
Str_Acq90	811	225/75	28/47	118	0.40	2
tc0	271	89/11	0/11	34	0.34	3
gyro0	374	53/47	3/44	57	0.57	1
Gyro_Acq79	641	285/15	0/15	68	0.23	5
str0	261	94/6	0/6	32	0.32	4

TABLE 7.3 – Statistiques issues des mesures du degré de contention des tâches.

Mesures selon le pire scénario de contention

Pour garantir le niveau de contention le plus élevé, les tâches du scénario de mesure sont démarrées en même temps. Les tâches sont exécutées en continue sans synchronisation entre les redémarrages. Les exécutions s'arrêtent lorsque la tâche étudiée a été exécutée un nombre donné de fois. On impose un nombre initial de 500 itérations en référence aux tailles des traces étudiées dans le Chapitre 6. Cela permet de maintenir un niveau d'interférence continue même dans le cas de tâches de longueurs différentes. Entre deux exécutions d'une même tâche, les données associées au cœur dont la tâche redémarre son exécution sont vidées des caches. Vider les caches a pour effet de générer de nouveaux accès et ainsi davantage d'interférences. Ceci permet de considérer le cas de tâches s'exécutant entre les exécutions de la tâche du même cœur.

Pour valider l'analyse du pire scénario de contention, les quatre scénarios de mesure possibles sont exécutés et comparés. Le classement des ensembles de tâches selon l'heuristique proposée est comparé à l'aide des ETPs et des pWCETs de chaque scénario. Les traces résultant de chaque scénario selon les spécifications de mesure données Si sont évaluées par l'outil de diagnostique DIAGXTRM.

7.5.3 Analyse MBPTA

Les résultats en moyenne de l'analyse du pire scénario de contention sont donnés dans les diagrammes boîte à moustache de la Figure 7.7 et dans la Table 7.4 Les résultats montrent que l'ensemble de tâches ts1 favorise bien les temps d'exécution empiriques les plus larges. Le classement donné par l'analyse du pire scénario de contention est validé par les mesures puisque :

$$\max(\text{S1}) > \max(\text{S2}) > \max(\text{S3}) \simeq \max(\text{S4}).$$

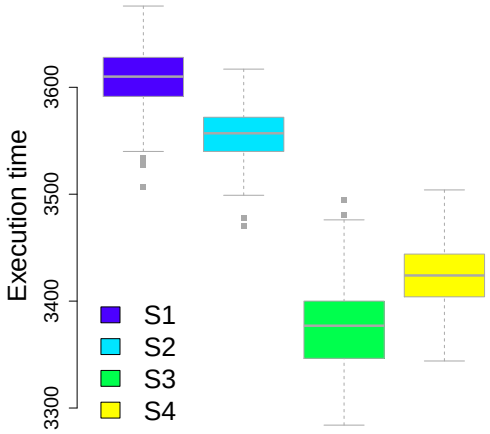


FIGURE 7.7 – Comparaison de la distribution empirique des temps d'exécution de la tâche GPS_Acq85 entre les quatre scénarios.

Néanmoins, on observe que max(S4) > max(S3), mais la différence entre les deux profils est moindre comparée aux profils des scénarios $S1$ et $S2$. De plus, les diagrammes montrent que le profil de $S3$ a une plus grande variabilité que le profil de $S4$. Cette variabilité vient du degré d'interférence supérieur du scénario $S3$ qui favorise davantage de cas d'interférence entre les tâches au niveau matériel, et donc de temps d'exécution différents.

En particulier, la variabilité du profil de $S3$ est amplifiée au niveau des mesures extrêmes i.e. les mesures au dessus de la moustache supérieure donnée par le quantile à 75%, $q(0.75, \mathcal{T})$. L'incertitude de modélisation du profil de $S3$ est plus grande que dans le cas de $S4$ ce qui se traduit par une queue de distribution plus longue. Rappelons que les pWCETs visent à modéliser les queues de distribution des ETPs et sont donc plus représentatifs pour la comparaison des scénarios en terme de pire cas.

$\mathcal{T}$	minimum	median	mean	$q(0.75, \mathcal{T})$	maximum
S1	3507	3610	3609	3628	3676
S2	3470	3557	3555	3572	3617
S3	3284	3377	3374	3400	3495
S4	3344	3424	3424	3444	3504

TABLE 7.4 – Résultats en moyenne issus des traces de temps d'exécution pour chaque scénario.

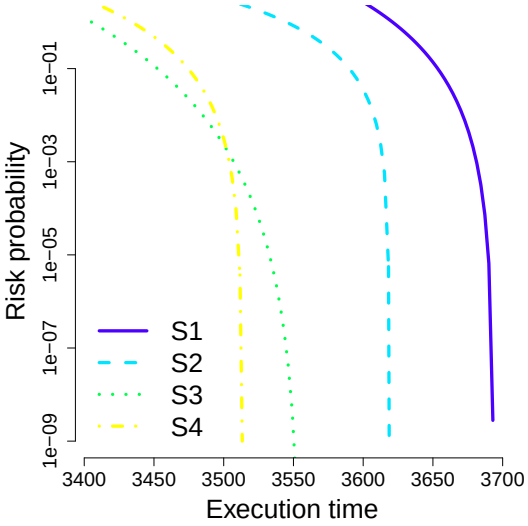


FIGURE 7.8 – Comparaison des pWCETs en échelle logarithmique entre les quatre scénarios.

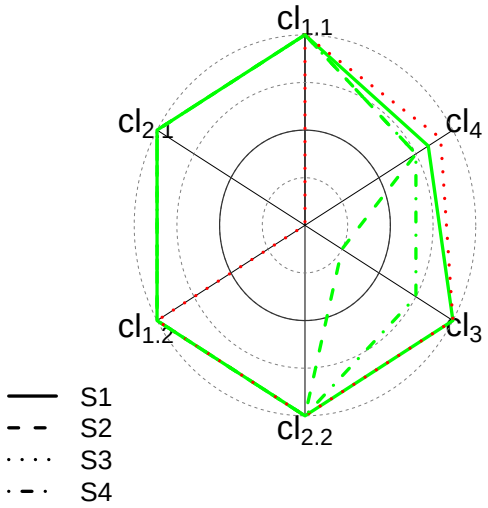


FIGURE 7.9 – Comparaison des résultats de DIAGXTRM entre les quatre scénarios.

Estimation des pWCETs

L'application de DIAGXTRM donne une estimation de pWCET pour tous les scénarios. Les pWCETs estimés sont tracés dans la Figure 7.8. Les paramètres des pWCETs et leur évaluation selon DIAGXTRM sont donnés dans la Table 7.5. Les niveaux de confiance de l'évaluation des quatre traces sont détaillés dans la Figure 7.9. En conclusion, l'EVT est applicable pour tous les cas sauf $S3$. Dans le cas de $S3$, le pWCET n'est

τ	length	$cl_{1.1}$	$cl_{2.1}$	$cl_{1.2}$	$cl_{2.2}$	cl_3	cl_4	u	ξ	ξ^{SE}	α_u	α_u^{SE}	$\langle WCET; 10^{-9} \rangle$	$Pess(10^{-9})$
S1	500	4	4	4	4	4	3.33	3621	-0.26	0.07	18.70	1.84	3693	0.46
S2	500	4	4	4	4	4	3.00	3545	-0.41	0.03	30.53	1.73	3618	0.05
S3	500	4	0	4	4	4	3.67	3405	-0.16	0.10	24.67	3.59	3548	1.54
S4	500	4	4	4	4	4	3.00	3434	-0.31	0.05	24.58	2.25	3513	0.26

TABLE 7.5 – Résultats de l'application de DIAGXTRM aux quatre scénarios de mesure.

pas fiable car le niveau $cl_{2.1}$ est nul. Ce résultat est à discuter puisque ce niveau de confiance est supérieure à un dans tous les autres cas réalisés dans les mêmes conditions. Ainsi, le test sélectionné d'hypothèse d'indépendance $h'_{2.1}$ est peut être trop discriminant ou sensible aux mesures.

La comparaison des pWCETs montre le même ordre entre les scénarios que la comparaison en moyenne. De plus, dans les quantiles extrêmes, à partir de 10^{-3}, le pWCET de $S3$ devient plus grand que le pWCET de $S4$. Ce dépassement confirme notre analyse graphique sur les queues de distribution des profils de temps d'exécution. De plus, cela vérifie que le scénario $S3$ génère plus d'interférences et favorise des temps d'exécution potentiellement plus longs que le scénario $S4$ comme donné par la CONTENDERCLASSIFICATION.

Les paramètres des pWCETs révèlent des paramètres de forme ξ tous négatifs, les pWCETs convergent vers un pire cas i.e. un WCET seuil à 0. La convergence des pWCETs montre que les temps d'exécution extrêmes divergent peu des temps d'exécution moyens. Cela confirme que les conditions d'exécution du simulateur sont plutôt stables pour toutes les itérations de la tâche étudiée : aucun mécanisme imprédictible ni surcharge imprévue du système n'a occurré.

Le pessimisme des WCETs seuils est très faible, moins de 1%. Ces ordres de grandeur de pessimisme sont largement dus aux paramètres de forme négatifs des pWCETs. Les WCETs seuils sont inférés sur la base des pWCETs estimés relativement à l'EVT et sont sûrs car les hypothèses sont respectées. Néanmoins, rien n'empêche d'ajouter une marge de sécurité e.g., de 20%, à la manière des pratiques industrielles. La comparaison des pWCETs confirme l'efficacité de la métrique de débit d'accès mémoire *mem_band* pour déterminer le pire scénario de contention.

Dans la Table 7.6, sont donnés les résultats concernant l'analyse de robustesse, décrite dans le Chapitre 5, appliquée aux traces issues des quatre scénarios de mesure.

Les traces ont toutes convergé, $cl_4 \geq 1$, comparées à leur trace réduite à 10% respective. Les bornes des intervalles de confiance fiables $[WCET^{-SE}; WCET^{+SE}]$ sont toutes confiantes et la taille des intervalles est très faible, de l'ordre de la centaine de cycles pour $S1$, $S2$ et $S4$. La borne supérieure de l'intervalle de confiance fiable dans le cas de $S3$ est la plus élevée, 5048 cycles, confirmant la queue de distribution du profil correspondant qui est plus longue. Ce résultat montre que le scénario $S3$ présente davantage de variabilité au niveau de ses temps d'exécution extrêmes, ainsi l'estimation est plus incertaine d'où un intervalle de confiance plus grand.

$\mathcal{T}$	cl_4	$\langle WCET; 10^{-9} \rangle$	$WCET^{-SE}$	cl_3^{-SE}	$WCET^{+SE}$	cl_3^{+SE}
S1	3.33	3693	3676	4	3790	4
S2	3.00	3618	3617	1	3638	1
S3	3.67	3548	3495	4	5048	4
S4	3.00	3513	3504	3	3568	3

TABLE 7.6 – Résultats de l'analyse de robustesse de DIAGXTRM pour les quatre scénarios de mesure.

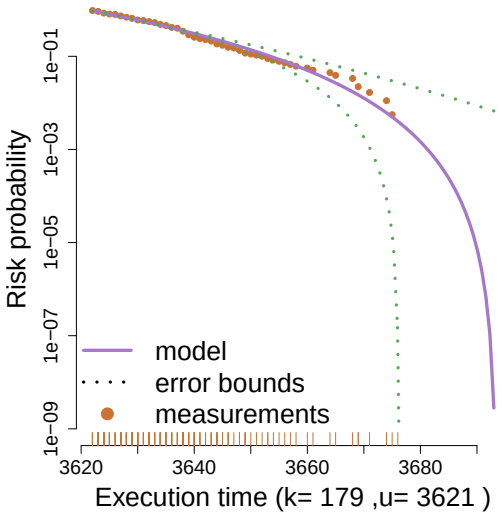


FIGURE 7.10 – Enveloppe de confiance sur le pWCET du pire scénario de contention $S1$.

pWCET de la trace issue du pire scénario de contention

Le tracé du pire pWCET issu du pire scénario de contention $S^{worst} = S1$ est donné dans la Figure 7.10. Le pWCET estimé modélise précisément la distribution empirique des pics de dépassement. Dans la figure est également tracée l'enveloppe de confiance, en pointillés, pour borner l'erreur d'estimation des paramètres du pWCET. Les courbes de l'enveloppe modélisent bien la distribution empirique puisque leur niveau d'adéquation cl_3 vaut 4. L'intervalle de confiance fiable fournit est resserré autour du WCET seuil à 10^{-9}. En conclusion, le pWCET estimé issu de la trace réalisée selon les conditions du pire scénario de contention $S1$ est fiable et robuste. Il est également représentatif des contentions les plus élevées du système car le pWCET estimé majore tous les autres scénarios de contention possibles. Finalement, le WCET seuil à 10^{-9} issu du pWCET estimé est sûr et précis puisque la différence absolue entre le maximum de la trace et le WCET seuil est de 0.46%.

7.5.4 Discussion

Analyse du reste de l'application

Dans ce chapitre, nous avons considéré la seule tâche GPS_Acq85 comme cas d'étude alors que l'application FAS est composée de 19 tâches. L'approche proposée reste applicable pour toutes les tâches restantes en choisissant des paramètres $(instructions, D, a, b)$ pour simuler chaque tâche. L'analyse proposée permet de déterminer le pire scénario de contention pour chaque tâche et de réaliser une trace de mesures selon les conditions du scénario i.e. les tâches à exécuter en parallèle. Enfin, DIAGXTRM est appliqué à chaque trace de mesures et un pWCET est estimé. Celui-ci est représentatif du pire cas d'exécution en parallèle i.e. des interférences les plus élevées au niveau des ressources matérielles. En utilisant l'approche proposée de classification CONTENDERCLASSIFICATION, seulement 19 traces doivent être réalisées pour toute l'application. Sans l'heuristique proposée, 303 traces devraient être réalisées pour déterminer un pWCET représentatif des pires interférences pour chaque tâche. Le gain en terme de temps avec l'heuristique proposée est considérable pusique pour des traces de longueur minimale de 500 mesures, l'approche permet d'éviter au minimum 142000 mesures. Parmi ce nombre élevé de mesures, la majorité n'est pas représentative du pire niveau de contention.

Nouvelles tâches

L'approche proposée est relativement composable. Si une nouvelle tâche est introduite dans l'application, seules quelques étapes sont à refaire. La nouvelle tâche entraîne un nouveau graphe G et donc un nouveau graphe d'indépendance $\bar{G}$. La CONTENDERLISTSEARCH doit être relancée sur le nouveau graphe d'indépendance. Toutes les étapes suivantes, de la classification aux mesures et l'estimation du pWCET doivent être effectués pour la nouvelle tâche. Enfin, l'étape de classification doit être faite pour les tâches n_i contenant la nouvelle tâche dans leur $ContenderList(n_i)$. Pour chaque tâche, si la nouvelle tâche ne fait pas partie de l'ensemble de tâche le plus contendant, le pWCET estimé sans la nouvelle tâche est conservé. Sinon, le pWCET de la tâche est à nouveau estimé.

Caches privés

La méthodologie de mesure proposée est pessimiste car les caches privés sont toujours vidés avant une nouvelle exécution. Puisque le placement des tâches n'est pas connu il est impossible de savoir quelles tâches vont partager les caches privés dans la suite de l'exécution de l'application. Il serait possible de diminuer le pessimisme de l'approche si le placement était connu.

Blocs de mémoire cache partagés

Pour des tâches s'exécutant en parallèle, si les tâches partagent les mêmes blocs mémoires i.e. données et instructions placées dans des blocs partagés, les tâches auront

tendance à expérimenter davantage de *misses*. Ainsi, l'analyse du placement mémoires des données et instructions de chaque tâche peut permettre d'affiner le classement des scénarios de mesure et améliorer la représentativité du pire scénario de contention. De plus, nous n'avons pas considéré le cas de tâches partageant de blocs mémoire alors la cohérence de cache n'était pas nécessaire.

Ordre des tâches

Alors que la question des caches privés pose le problème d'un pessimisme d'ordre spatial, l'ordre des tâches dans le DAG pose le problème d'un pessimisme temporel. En effet, selon le DAG de la Figure 7.4, les tâches gps0, gyro0, str0 et tc0 sont démarrées en même temps et si leur temps d'exécution sont proches, alors l'exécution en parallèle la plus réaliste est Φ^4(gps0 ‖ gyro0 ‖ str0 ‖ tc0). Alors que ces quatre tâches ont un nombre d'ensembles de tâches contendantes supérieur à 4, l'approche proposée pourrait fournir un scénario plus pessismiste que le scénario le plus réaliste i.e. au déploiement de l'application. De plus, si la tâche tc0 est démarrée en décalage, le scénario le plus réaliste est Φ^4(gps0 ‖ gyro0 ‖ str0). Sans la tâche tc0, le niveau d'interférence est moindre et ainsi augmente davantage le pessimisme du scénario de pire contention par rapport au scénario réaliste.

Pessimisme du scénario de mesure

Au déploiement de l'application temps réel, la tâche étudiée va expérimenter le scénario de mesure S^{real}. Le scénario réaliste S^{real} va potentiellement être différent du pire scénario proposé S^{worst}. Ainsi, le pire scénario de mesure est représentatif du pire cas mais pas du cas en opération. Si l'on note le pWCET issu de la trace de mesure selon les conditions du scénario S $pWCET(S)$, alors on devrait avoir l'ordre :

$$pWCET(S^{iso}) \leq pWCET(S^{real}) < pWCET(S^{worst}),$$

où S^{iso} est le scénario de mesure selon lequel la tâche est exécutée en isolation, sans interférence.

L'analyse du pire scénario de contention propose d'estimer un pWCET représentatif d'un grand niveau d'interférence même si ce niveau n'est pas réaliste, il garantit la *sûreté* de l'analyse. Pour se rapprocher du scénario réaliste, l'approche doit considérer davantage d'informations comme la répartition des tâches sur les cœurs et l'ordre d'exécution des tâches dans l'application.

Degré de contention de la tâche étudiée

Pour une tâche étudiée qui dépend très peu des ressources matérielles et donc dont le débit d'accès mémoire est faible, l'analyse du pire scénario de contention peut se révéler inutile. En effet, si une tâche accède très peu aux ressources matérielles, elle sera très peu affectée par les autres tâches. Ainsi, son temps d'exécution ne sera pas influencé par les contentions. Pour éviter une analyse de pire scénario contention, il faudrait définir une

limite sur le débit d'accès mémoire telle que pour une tâche dont le débit est inférieur à cette limite l'analyse de pire scénario de contention est inutile.

7.6 Conclusion

Nous proposons une analyse systématique de graphe d'application temps réel pour déterminer un scénario de mesures qui favorise les temps d'exécution les plus longs. Ce scénario de mesure est représentatif du niveau de contention dans les ressources matérielles partagées le plus élevé. L'application de DIAGXTRM à la trace permet d'estimer le pWCET issu de ce pire scénario de contention. Ce pWCET est représentatif du pire cas d'exécution de tâches en parallèle pour la tâche étudiée relativement à l'heuristique définie qui caractérise le niveau de contention d'une tâche.

L'analyse du pire scénario de contention a été appliquée et validée dans le cas d'un simulateur d'exécution de tâches en parallèle. Une perspective de ce travail est de pouvoir appliquer cette analyse à un cas d'étude réel sur une plateforme donnée. Pour ce faire, la plateforme doit être composée de plusieurs cœurs avec des caches partagés. La difficulté est l'exécution stricte des tâches sur les cœurs et la mesure du débit d'accès mémoire de chaque tâche.

Chapitre 8

Développement d'un algorithme pour déterminer les conditions d'exécution d'une trace pour l'estimation de pire temps d'exécution probabiliste robuste aux paramètres d'entrée de la tâche étudiée

Sommaire

Résumé 4
Le chapitre d'application de diagXtrm *a montré que les paramètres d'entrée d'une tâche ont un effet sur la stationnarité de la trace et la continuité relative des mesures. Ceci est notamment dû à la présence de plusieurs chemins dans la tâche dont les exécutions dépendent des paramètres d'entrée. Ce chapitre a pour objectif de déterminer les paramètres d'entrée à exécuter et comment les exécuter pour garantir la fiabilité et la représentativité du pWCET estimé. En effet, à cause du non déterminisme temporel de la plateforme considérée et la sensibilité différente des chemins aux interférences, ne considérer que le pire chemin peut fournir un pWCET non représentatif et donc non sûr. De plus, pour faciliter l'analyse du comportement temporel des tâches multichemin, une représentation spectrale de tâche est proposée. Cette représentation permet également de simuler le comportement temporel de tâches multichemin en fonction de ses paramêtres d'entrée.*

8.1 Etude de l'impact des paramètres d'entrée sur l'estimation du pWCET

Nous avons fait le choix dans le Chapitre 5 d'appliquer l'EVT dans le cas de mesures stationnaires. L'évaluation de la stationnarité d'une trace n'est pas systématique. Pour les besoins de diagXtrm, nous avons sélectionné le test de KPSS pour étudier la stationnarité de la trace d'entrée $\mathcal{T}$. De plus, la propriété de stationnarité peut être intimement liée à la propriété de continuité des mesures car les fonctions de répartition de l'EVT sont continues.

8.1.1 Impact sur la stationnarité des mesures

Au cours de cette thèse, nous avons pu remarquer l'impact des paramètres d'entrée, comme condition d'exécution imposable, sur la stationnarité des traces dans le cas de

tâches multichemin i.e. les entrées correspondant à des chemins différents. Pour illustrer ce propos, considérons une tâche *loop* composée d'une simple boucle *for* dont le code est donné dans l'Exemple 8.1. Le paramètre d'entrée de la tâche n est la borne supérieure sur le nombre d'itérations de la boucle de *loop*. A chaque entrée n correspond un nombre d'itérations n de la boucle et donc à un chemin de la tâche.

Exemple 8.1 (*Code source de la tâche loop*)
Exemple de code de la tâche loop.

```
void loop(int n){
  int i;
  for ( i = 2;
  i<= n;
  i++ ){
    //Do something
  }
}
```

On suppose que, le temps d'exécution du test de la boucle *for* et du bloc à l'intérieur restent constants. Considérons une fonction f de l'entier t vers la borne entière n, paramètre d'entrée de la tâche *loop* : $\forall t \in \mathbb{N}$, $f(t) = n$. Le temps t est un entier qui représente un instant de mesure. A $t = 1$, le temps d'exécution mesuré est C_1. Pour t quelconque on a C_t. Considérons trois cas de traces non stationnaires :

1. f est une fonction déterministe de t quelconque e.g., une fonction linéaire de paramètres entiers a et b telle que $f(t) = a \times t + b$. Ainsi, le temps d'exécution C_t suit également une fonction déterministe puisque le temps d'exécution de *loop* est directement proportionnel à n, déterminé par f. La série temporelle donnée par la trace $\mathcal{T}$ des mesures C_t suit donc une *tendance déterministe* qui est un cas de trace non stationnaire.
2. f est une suite de n_{t-1} i.e. la borne supérieure à l'instant précédent. Soit la fonction f la suite $f(t) = \min(n_{t-1} + r_t, 2)$, où r_t est un relatif aléatoire suivant une loi Normale centrée en 0. Ainsi, la mesure C_t est une fonction de la mesure précédente C_{t-1} et d'une variable aléatoire. Dans ce cas, $\mathcal{T}$ suit une *marche aléatoire* qui est également un cas de trace non stationnaire.
3. f est définie par une fonction continue par morceaux. Par exemple, pour deux bornes n_1 et n_2 et pour $t \in [\![0; 500]\!]$, $f(t) = n_1$, si $t < 250$ et $f(t) = n_2$ pour $t \geq 250$. La suite de mesure $\mathcal{T}$ est proportionnelle à n_1 pendant la première moitié de mesures, puis à n_2. C'est un cas de *saisonnalité*, la trace n'est alors pas stationnaire.

Un cas de trace stationnaire pourrait être pour n, paramètre d'entrée de *loop*, une variable aléatoire distribuée selon la fonction f. Le paramètre n comme variable aléatoire peut suivre une loi Normale, une loi de Cauchy ou de Gumbel. Comme le

temps d'exécution de *loop* est seulement fonction de la borne n, la distribution de la borne est la même que la distribution des mesures de temps d'exécution à un facteur près. En conséquence, la distribution initiale $f_{\mathcal{C}}$ des temps d'exécution est connue et il est possible de déterminer formellement le pWCET de la tâche, à un facteur près, à partir de f. La connaissance du code de *loop* et de la distribution des entrées de la tâche permet de déterminer formellement le pWCET de *loop* dans le cas d'une trace iid selon une loi proportionnelle à f. Néanmoins, le pWCET ne rend compte que de la variabilité fonctionnelle de *loop*. De plus, il suffirait de prendre n le plus grand possible pour déterminer le WCET de *loop*. Pour une exécution sur une plateforme dite temporellement non déterministe, les temps d'exécution de la boucle seule ne restent pas constants et ainsi du bruit dû à la variabilité du système e.g., interférences, est présent dans la trace de mesures et seul le cas de mesures stationnaires est applicable pour l'analyse MBPTA. Dans ce cas, la détermination du WCET par une analyse statique devient plus compliquée.

8.1.2 Conditions sur les chemins pour des mesures stationnaires

Considérons une tâche τ multichemin dont les chemins sont notés π_i avec $i \in [\![1; \Pi]\!]$ où Π est le nombre possible de chemins dans τ. Pour un chemin exécuté π_i, l'ensemble des temps d'exécution non observés est $\overline{\Omega_i}$. Pour tous les temps d'exécution $C \in \overline{\Omega_i}$, la probabilité de mesurer C sachant que le chemin π_i est exécuté est $P(C|\pi_i) = 0$. Si un chemin est un imposé de manière déterministe, ici π_i, pour une ou plusieurs mesures, puis un chemin différent est imposé, alors les mesures de temps d'exécution de la trace résultante ne sont pas stationnaires.

Cette réflexion peut être élargie à n'importe quelle condition d'exécution, que ce soit un chemin imposé ou des tâches en parallèle.

Une manière simple de prévenir la non stationnarité dans le cas d'une tâche multichemin est d'exécuter un seul chemin par trace. Ainsi, la variabilité modélisée par le pWCET est celle introduite par la complexité du processeur. Néanmoins, il reste à déterminer quel chemin exécuter pour déterminer un pWCET représentatif du pire cas.

Il se peut que nous n'ayons pas accès au code de la tâche, ou que la tâche soient composée d'un nombre trop important de lignes de code. Dans ce cas, la détermination d'un tel chemin est trop complexe et nous considérons alors un scénario de mesures S composé de différents chemins à exécuter. Quelles sont les conditions d'exécution des chemins pour garantir la stationnarité de la trace ? Soient les chemins π_i $\forall i \in \Pi_S$ composant le scénario S, avec Π_S un sous-ensemble de $[\![1; \Pi]\!]$. Pour assurer la stationnarité de la trace, $\forall t$ instant de mesure, la probabilité d'occurrence de chaque chemin $P(\pi_i)$ $i \in \Pi_S$ doit rester constante. Ainsi, dans le cas d'une plateforme temporellement non déterministe, la distribution des temps d'exécution issus du scénario S est une distribution de mélange à partir de la distribution des temps d'exécution de chaque chemin comme schématisé dans la Figure 8.1. Dans le cas d'une plateforme temporellement déterministe, la distribution des temps d'exécution issus du scénario serait une distribution proportionnelle à la distribution $P(\pi_i)$ des chemins considérés par le scénario S.

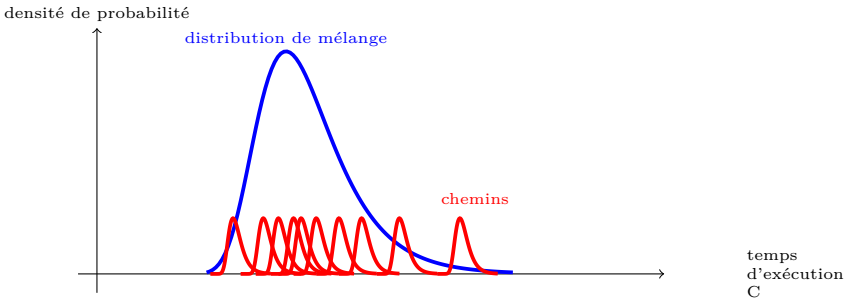


FIGURE 8.1 – Profil de temps d'exécution d'une trace issue d'un scénario de mesure composé de 10 chemins.

Exemple 8.2
On applique ce raisonnement à la tâche loop de l'Exemple 8.1. Si l'on considère un scénario avec dix entrées possibles de n_1 à n_{10} dans l'ordre croissant, alors le profil de temps d'exécution de loop ressemble à celui de la Figure 8.1.

Dans la suite, nous allons formaliser cette réflexion par la définition d'une analyse temporelle de tâche multichemin. Dans un second temps, un algorithme basé sur les probabilités d'occurrence des chemins sera proposé pour déterminer quels chemins exécuter pour obtenir un pWCET représentatif du pire cas mais aussi des conditions de l'EVT.

8.2 Définition de l'analyse spectrale de tâche multichemin

Pour formaliser l'analyse temporelle de tâches multichemin, nous introduisons un formalisme nommé *analyse spectrale*. L'analyse spectrale réfère au spectre fréquentiel d'un signal qui peut être modélisé par la somme de ses fréquences propres. L'objectif en théorie du signal est d'interpréter de manière simple un phénomène physique dépendant du temps décrit par un ou plusieurs signaux. L'intérêt d'une telle décomposition est de pouvoir déterminer la forme d'un signal de sortie à partir d'un signal d'entrée connu à travers un système décrit par une équation différentielle. Dans notre cas, l'analyse spectrale que nous proposons permet d'interpréter spatiallement le comportement temporel d'une tâche multichemin comme étant une composition des temps d'exécution de ses chemins.

8.2.1 Représentation spectrale d'une tâche mutlichemin

La tâche multichemin étudiée τ est vue comme la somme de ses modes propres qui sont donnés par les temps d'exécution caractéristiques des différents chemins composant la tâche.

Les temps caractéristiques de la tâche τ peuvent être déduits par des mesures selon un algorithme similaire à celui présenté dans la Section 2.1.2. La difficulté est de

prouver l'exhaustivité des chemins parcourus. Pour décrire les temps caractéristiques, on effectue la distinction entre une plateforme temporellement déterministe et une non déterministe. Pour la tâche τ s'exécutant sur une plateforme déterministe, alors pour chaque chemin π_i $i \in [\![1;\Pi]\!]$ l'ensemble des temps d'exécution mesurés Ω_i est de taille 1 i.e. $|\Omega_i| = 1$. Ainsi, $\Omega_i = T_i$ où T_i est le temps caractéristique du chemin π_i. Pour une plateforme temporellement non déterministe $|\Omega_i| > 1$ par définition. Dans ce cas, T_i est la localisation des temps d'exécution issus de π_i e.g., la moyenne de Ω_i.

A partir des temps caractéristiques de τ, nous proposons une représentation temporelle complète de la tâche :

Définition 8.1 (*Représentation spectrale d'une tâche multichemin (Task Spectral Representation TSR)*)
Soit une tâche τ dont les chemins sont notés $\pi_i, \forall i \in [\![1;\Pi]\!]$ et Π le nombre total de chemins. Soit Ω_i l'ensemble des temps d'exécution possibles que π_i peut prendre. La représentation spectrale de la tâche τ est $|\tau|(C)$ est une fonction du temps d'exécution $C \in \mathbb{R}^+$ telle que :

$$|\tau|(C) = \frac{1}{\Pi} \sum_{i=1}^{\Pi} \delta_i(C), \tag{8.1}$$

avec δ_i la fonction Dirac qui vaut 1 si $C = T_i$ et 0 sinon, où T_i est le temps d'exécution caractéristique du chemin π_i de τ.

Pour des mesures exhaustives, si la plateforme est temporellement déterministe, l'ETP de la trace est égal à la TSR de τ. Sinon, les temps d'exécutions mesurés sont dispersés autour des temps caractéristiques des chemins de τ.

Exemple 8.3
L'application de la TSR à la tâche loop n'est pas directe. En effet, si la borne maximale sur le nombre d'itérations n'est pas connue alors le spectre de la tâche est défini sur un ensemble non borné à droite. Le temps caractéristique d'un chemin dépend du nombre d'itérations de la boucle for. Soit le plus petit chemin π_1 de loop dont le temps caractéristique est noté T_1. Alors T_1 est de l'ordre de grandeur de deux fois les opérations effectuées dans la boucle for. Le chemin suivant π_2 est celui qui correspond à trois itérations soit son temps caractéristique $T_2 = 1.5T_1$. On déduit les temps caractéristiques des chemins suivants de la même manière soit pour le chemin π_n alors $T_n = T_1 + (n-1) \times \frac{T_1}{2}$.

La TSR permet de comparer temporellement deux chemins de τ :

Définition 8.2 (*Comparaison de chemins*)
Deux chemins π_i et π_j dont les temps caractéristiques sont respectivement T_i et T_j, π_i est plus petit que π_j, $\pi_i < \pi_j$, si $T_i < T_j$. Inversement, π_j est plus grand que π_j, $\pi_j > \pi_i$, $T_i > T_j$.

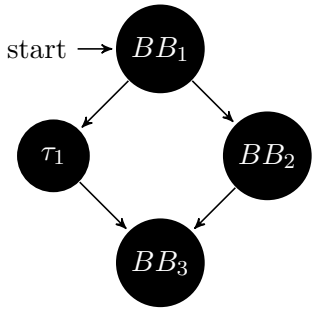


FIGURE 8.2 – Graphe de flôt de contrôle de τ.

Pour un processeur temporellement déterministe, la comparaison empirique des chemins est directe. Pour un processeur temporellement non déterministe, la comparaison empirique est moins évidente du fait de la variabilité système des temps d'exécution. Pour ce faire, il faut estimer le temps caractéristique T_i de chaque chemin π_i.

Exemple 8.4
Dans le cas de la tâche loop, les chemins sont ordonnés selon le nombre respectif d'itérations de la boucle for. Ainsi, $\pi_1 < \pi_2 < \ldots < \pi_\Pi$ *où* π_Π *est le chemin pour lequel le nombre d'itérations maximal est atteint.*

8.2.2 La composition de TSRs

L'une des difficultés de l'analyse temporelle des tâches multichemin est de garantir l'exhaustivité des chemins observés. La difficulté à garantir l'exhaustivité est notamment due à la taille importante des tâches et à leur nombre de chemins. Il serait donc judicieux d'analyser des sous-parties de tâche i.e. composés de moins de chemins que la tâche entière, pour déduire l'ensemble des chemins de la plus grande tâche initiale. Connaissant les TSRs des plus petites tâches, il est possible de composer ces TSRs pour obtenir la TSR final.

Par exemple, pour une tâche τ composée de Π_1+1 chemins, la tâche peut exécuter les Π_1 chemins de la tâche connue τ_1 ou un bloc de base BB_2 comme dans la Figure 8.2. Alors la TSR de τ peut s'exprimer comme $|\tau|(C) = |\tau_1|(C-\Delta C)+\delta_2(C-\Delta C)$. Les contraintes de précédence entre BB_1, τ_1 et BB_3 entraînent une relation de dépendance temporelle qui oblige à translater la TSR de τ_1 de ΔC correspondant au temps d'exécution de BB_1 et BB_3. Pour prendre en compte le bloc de base BB_2 dans la TSR de τ, il suffit d'ajouter son temps caractéristique également translaté de ΔC.

La règle de composition pour deux TSRs est :

Proposition 8.1 (*Composition de deux TSRs*)
Soit une tâche multichemin τ composée de deux tâches τ_1 et τ_2 telles que τ_1 soit composée de Π_1 chemins, et τ_2 de Π_2 chemins. Les TSRs des deux tâches τ_1 et τ_2 sont connus, alors la TSR $|\tau|$ est

- $|\tau_1| + |\tau_2|$ *pour des tâches sans contrainte de précédence,*
- $|\tau_1| \circ |\tau_2| = \frac{1}{\Pi_1+\Pi_2} \sum_{i=1}^{\Pi_1} \sum_{j=1}^{\Pi_2} \delta_{1_i}(t - T_{2_j}) \left(= \frac{1}{\Pi_1+\Pi_2} \sum_{i=1}^{\Pi_2} \sum_{j=1}^{\Pi_1} \delta_{2_i}(t - T_{1_j}) \right)$ *pour des tâches avec des contraintes de précédence, avec T_{2_j} tel que $\delta_{2_j}(T_{2_j}) = 1$, le temps caractéristiques des chemins de la tâche τ_2 (T_{1_j} le temps caractéristique de la tâche τ_1). $\circ$ est l'opérateur de convolution symétrique.*

Pour garantir un tel résultat au niveau de l'exécution, τ_1 et τ_2 ne doivent pas partager de donnée ou d'instruction. En effet, si τ_2 est exécutée après τ_1, τ_2 pourrait bénéficier des chargements mémoires effectués par τ_1. Dans le cas où le seul chemin pire de τ_2 bénéficierait de tels chargements, le chemin pire pourrait ne pas contribuer au pire chemin de la tâche globale τ. Un tel cas pourrait être détecté lors des mesures si les chemins sont tracés.

De plus, la composition des TSRs pourrait produire des chemins infaisables puisque la composition considère tous les chemins sans tester leur faisabilité. L'analyse reste sûre dans le sens où la composition ne peut que donner un nombre supérieur ou égal au nombre de chemins faisables de τ.

Exemple 8.5
Pour un nombre maximal d'itérations, la TSR de loop peut être construite par composition à partir de la TSR du plus petit chemin π_1 de temps caractéristique T_1 et qui correspond à deux itérations de la boucle for. Ainsi, on obtient la TSR de la tâche loop, $|loop|$, en translatant δ_1 de $\frac{T_1}{2}$ pour tous les chemins de la tâche.

8.2.3 Distance entre les chemins

L'application de l'EVT à une trace de mesures requiert une continuité suffisante entre les mesures. La fonction de répartition utilisée pour l'estimation du pWCET étant une distribution de Pareto, une fonction continue, si les mesures exhibent une discontinuité trop forte entre elles, un tel modèle ne peut être applicable [GB10].

Les temps d'exécution mesurés étant des multiples de l'horloge, ils sont par nature discrets. Mais relativement à l'ensemble des mesures possibles, si les temps d'exécution sont suffisamment proches i.e. distants de quelques cycles d'horloge, la condition de continuité peut être acceptée. La même notion de distance peut être appliquée aux chemins de la tâche :

Définition 8.3 (*Distance spectrale*)
La distance spectrale $\|.\|$ est une norme de dimension 1 sur l'espace de la représentation spectrale. La distance entre deux chemins π_i et π_j $\|\pi_i\pi_j\|$, dont les temps caractéristiques respectifs sont T_i et T_j, est telle que

$$\|\pi_i\pi_j\| = \|\pi_j\pi_i\| = \begin{cases} T_i - T_j & \text{if } T_i > T_j \\ T_j - T_i & \text{if } T_j > T_i \\ 0 & \text{if } T_i = T_j \end{cases}. \tag{8.2}$$

$\|.\|$ vérifie les propriétés usuelles des normes.

Exemple 8.6
Dans la tâche loop, tous les chemins sont distants deux à deux de $\frac{T_1}{2}$ soit la durée d'une itération de la boucle for.

Considérant la notion de distance spectrale, on peut caractériser la continuité d'une trace de mesures de temps d'exécution :

Remarque 8.1 (*Continuité d'une trace*)
Pour un scénario de mesure S composé de $\Pi_S > 2$ chemins ordonnés depuis le plus petit au plus grand $\pi_{(1)} < \ldots < \pi_{(i)} < \pi_{(i+1)} < \ldots < \pi_{(\Pi_S)}$. La trace de temps d'exécution issue de S est continue si

$$\forall i \in [\![1; \Pi_S - 1]\!] \quad \frac{\|\pi_{(i)}\pi_{(i+1)}\|}{\|\pi_{(1)}\pi_{(\Pi_S)}\|} \approx 0. \tag{8.3}$$

La Remarque 8.1 stipule que pour des chemins considérés dans un scénario de mesure S consécutivement proches deux à deux dans la TSR de la tâche étudiée, alors la trace qui est issue de S est continue.

Il est important de différencier la distance entre deux paramètres d'entrée et la distance entre deux chemins. En effet, supposons une condition sur un entier telle que pour un entier strictement supérieur à 1, la tâche exécute un chemin avec un temps caractéristique élevé, sinon la tâche exécute un chemin avec un temps caractéristique bas. Les deux chemins sont éloignés dans la TSR de la tâche. Néanmoins, les deux paramètres d'entrée 1 et 2 sont relativement proches dans l'ensemble des entiers mais correspondent à deux chemins éloignés temporellement.

Exemple 8.7
Considérons un scénario de mesure de la tâche loop tels que les paramètres d'entrée possibles soient $n_1 = 2$ et $n_2 = 21$. Ainsi, le temps caractéristique du chemin correspond au cas de l'entrée n_2 est 10 fois plus grand que dans le cas de n_1. Pour respecter les conditions de stationnarité, chaque paramètre d'entrée à une chance sur deux d'être tiré pour chaque mesure du temps d'exécution de la tâche. En conséquence, la trace exhibe un delta significatif entre les mesures dans le cas de n_1 et celles dans le cas de n_2. La continuité ne peut être stipulée dans ce cas. Néanmoins, pour $n_2 = 3$, selon le temps d'exécution des opérations de la boucle for, il y a davantage de chances pour satisfaire la continuité de la trace. Enfin, la continuité peut également être satisfaite dans le cas de plusieurs entrées dont les chemins ont des temps caractéristiques proches e.g., les entrées n_i $\forall i \in [\![2; 21]\!]$.

8.2.4 Application

Le benchmark *ns* cherche une clé dans un tableau de dimension quatre chacune contenant 5 éléments. Il s'agit d'une recherche linéaire à travers le tableau, il y a donc autant de chemin que de clés à chercher. Dans cette étude, nous ne considérons que quatre clés et donc quatre chemins. Les clés à chercher dans le tableau sont les éléments qui se trouvent le plus loin de la dernière dimension. La première clé consiste à itérer sur 2×5^3 éléments, la deuxième sur 3×5^3, la troisième sur 4×5^3 éléments et enfin la quatrième clé sur 5^4 éléments. Cela donne donc quatre chemins de longueurs bien différentes car les nombres d'éléments à explorer sont significativement différents. La TSR de la tâche *ns* peut s'écrire

$$|ns|(C) = \frac{1}{4} \sum_{i=1}^{4} \delta_i(C) \tag{8.4}$$

où $\forall i \in [\![1; 4]\!]$, δ_i est défini par les paramètres d'entrée de *ns* précédemment décrits. Comme les chemins sont distants entre eux par un nombre d'itérations constant on peut écrire la TSR de *ns* avec la composabilité à partir du premier chemin :

$$|ns|(C) = \frac{1}{4} \left[\delta_1(C) + \delta_1(C - \Delta C) + \delta_1(C - 2\Delta C) + \delta_1(C - 3\Delta C)\right], \tag{8.5}$$

avec δ_1, la fonction du premier chemin et ΔC la distance spectrale entre des chemins consécutifs.

Conditions d'exécution

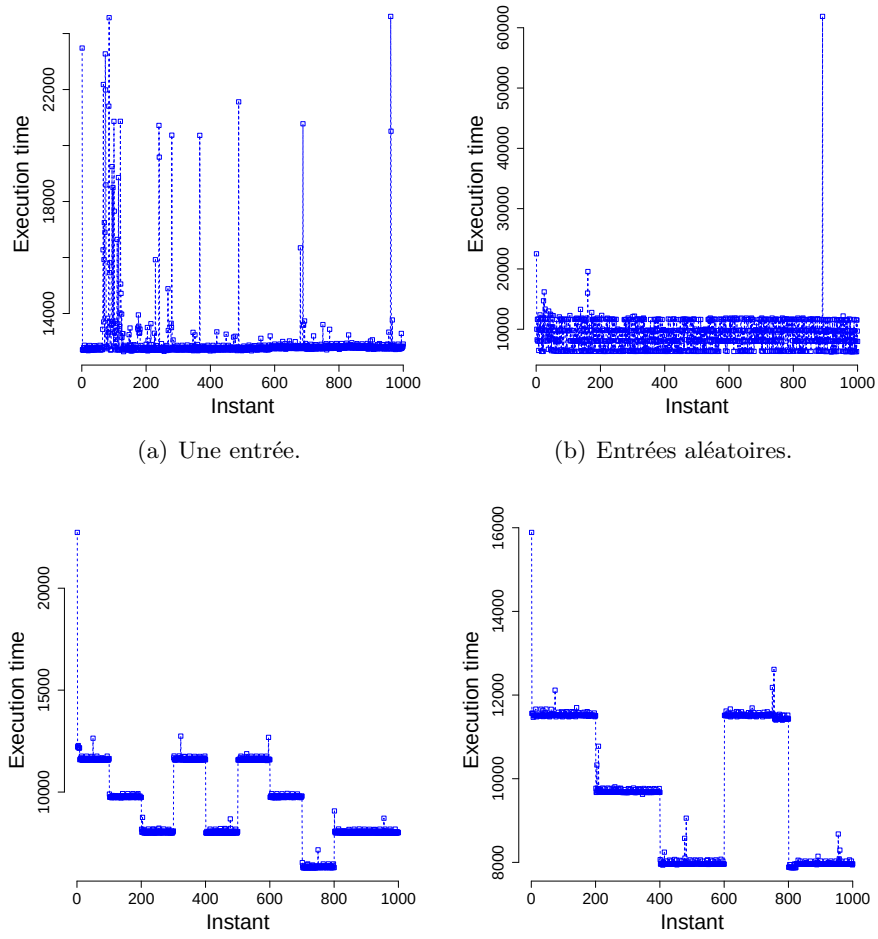
La tâche est exécutée 1000 fois sur la plateforme décrite dans la Section 4.2, sur un seul cœur et en isolation. On considère différents scénarios de mesure :

— Une seule entrée : la même clé pour toutes les mesures.
— Entrées aléatoires : chacune des quatre clés ont la même probabilité d'être exécutée pour toutes les mesures.

— Entrées périodiques : La clé est modifiée de manière aléatoire toutes les 100 et 200 mesures.

Résultats

Les traces pour toutes les conditions d'exécution sont présentées dans la Figure 8.3.



(a) Une entrée.

(b) Entrées aléatoires.

(c) Entrées périodiques : 100 itérations.

(d) Entrées périodiques : 200 itérations.

FIGURE 8.3 – Traces de temps d'exécution pour toutes les conditions d'exécution.

Les temps d'exécution sont en nombre de cycles. Le non déterminisme des temps d'exécution est particulièrement visible dans la Figure 8.3(a) i.e. pour un paramètre d'entrée fixe. Dans ce cas, c'est bien la variabilité due à la plateforme qui est mesurée. Le cas de la Figure 8.3(b) montre un recouvrement possible des temps d'exécution issus de chaque chemin. Néanmoins, les cas d'entrées périodiques des Figure 8.3(c) et 8.3(d) montrent que le temps d'exécution le plus élevé de chaque chemin est inférieur au temps

d'exécution minimum du chemin le plus grand. Le cas de la Figure 8.3(c) montre bien que les entrées changent toutes les 100 mesures alors que dans le cas de la Figure 8.3(d) il s'agit de toutes les 200 mesures. Par ailleurs, pour le même nombre total de mesures, dans le cas où la période est de 200, seulement trois chemins sont exécutés contre quatre au total. Ainsi, tous les chemins n'ont pas été explorés ce qui pose le problème de l'exhaustivité et donc de la sûreté de l'estimation finale car le chemin manquant aurait pu être le pire chemin de la tâche étudiée. Enfin, certains pics de temps d'exécution fortement divergents sont mesurés comme dans les cas d'une entrée et d'entrées périodiques alors qu'ils ne figurent pas dans les cas d'entrées périodiques. Cela pose le problème du nombre minimal de mesures par chemin qui sont à réaliser pour observer de tels pics et ainsi ne pas obtenir une estimation non sûre. De plus, cela pose la question de savoir quels sont les mécanismes du système qui peuvent amener à de telles divergences comme une surcharge de calcul ou un rafraîchissement des mémoires. Néanmoins, ces mécanismes sont trop discontinus et pourraient faire diverger l'estimation du pWCET selon l'EVT.

Les profils ETPs de la Figure 8.4 supportent l'idée de modéliser le temps d'exécution de la tâche avec une variable aléatoire. De plus, les ETPs montrent le nombre de chemins différents exécutés, un seul dans le premier cas, quatre dans le deuxième et troisième et trois dans le dernier cas. En ne conservant qu'un temps d'exécution caractéristique par chemin i.e. en filtrant les temps d'exécution dont les probabilités sont faibles, on déduit de manière expérimentale la TSR de ns.

Le cas aléatoire exhibe tous les chemins possibles et permet de déduire l'ETP respectif des temps caractéristiques de ns qui sont présentés dans la Table 8.1.

chemin π_i	**temps caractéristique** T_i	$\|\pi_i\pi_{i+1}\| \simeq \Delta C$
1	6300	0
2	8000	1700
3	9750	1750
4	11600	1850

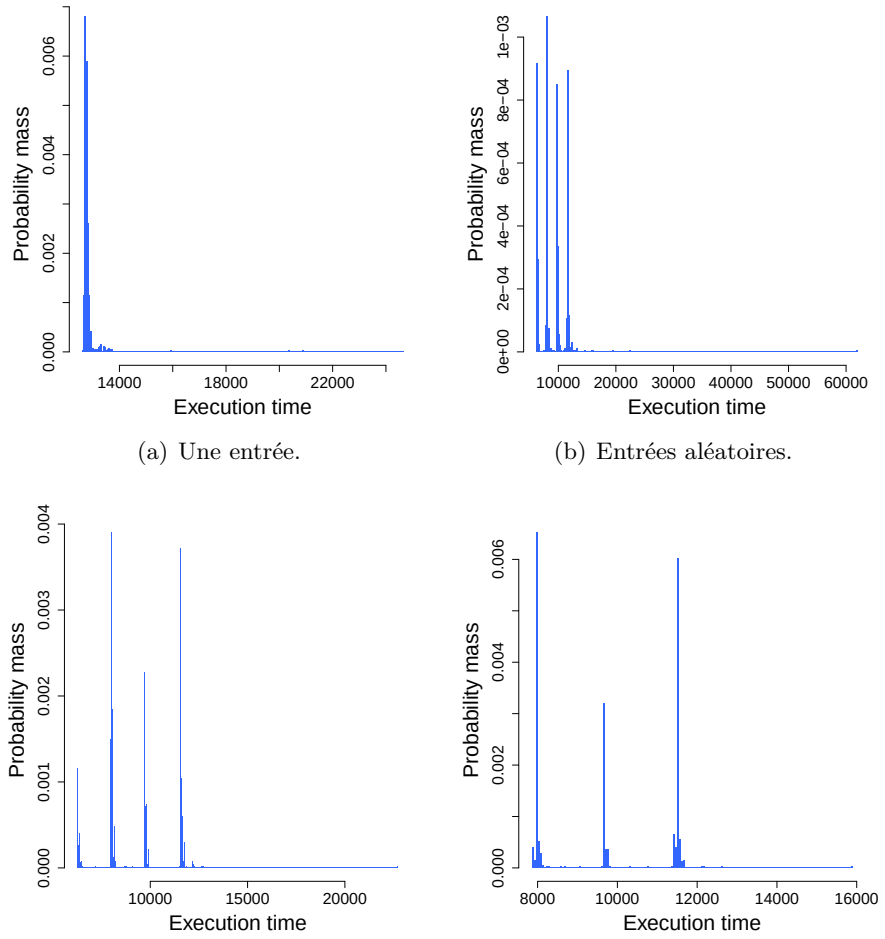
TABLE 8.1 – Calcul des temps caractéristiques des chemins de ns dans le cas d'entrées aléatoires.

On peut également calculer la distance entre les chemins consécutifs qui vaut environ ΔC.

L'hypothèse d'un ΔC constant entre les quatre chemins est confirmée avec un taux d'erreur de 9%.

Les résultats de l'analyse de stationnarité, à partir du test KPSS, sont présentés dans la Table 8.2. Ainsi que nous l'avions supposé, la stationnarité est validée dans les deux premiers cas : le cas d'une seule entrée, et le cas d'entrées aléatoires.

Enfin, les figures 8.3(b), 8.3(c) et 8.3(d) montrent que les mesures respectives de chaque chemin sont relativement éloignées les unes des autres. Il est donc difficile d'admettre que les mesures soient continues entre les différents chemins. De plus, cette dis-



(a) Une entrée.

(b) Entrées aléatoires.

(c) Entrées périodiques : 100 itérations.

(d) Entrées périodiques : 200 itérations.

FIGURE 8.4 – Profils de temps d'exécution pour toutes les conditions d'exécution.

$\mathcal{T}$	une entrée	aléatoire	100 itérations	200 itérations
KPSS	0.187	0.243	3.933	3.008

TABLE 8.2 – Résultats du test KPSS pour les différentes traces de temps d'exécution.

tance entre les chemins est donnée dans la Table 8.1 et vaut $\Delta C \simeq 1750$ cycles. La distance entre les chemins est du même ordre de grandeur que le temps caractéristique des chemins. En définitive, on ne peut admettre que les temps d'exécution entre les chemins soient continus.

8.3 Algorithme pour favoriser les temps d'exécution extrêmes dans le cas d'une tâche composée de plusieurs chemins

8.3.1 Modèle de simulation d'une tâche multichemin

Afin de développer l'algorithme proposé, nous considérons un modèle de simulation de tâche. Ce modèle est basé sur les réflexions menées lors de l'analyse spectrale de tâche multichemin. Dans l'analyse spectrale, chaque chemin π_i qui compose une tâche est modélisé par son temps caractéristique T_i. Pour une exécution sur une plateforme temporellement non déterministe, plusieurs temps d'exécution sont distribués autour du temps caractéristique des chemins de la tâche formant l'ensemble Ω_i.

Les approches MBPTA peuvent considérer un ou plusieurs chemins dans un scénario de mesure. De telles approches reposent sur le postulat que l'ensemble de temps d'exécution mesurés Ω_i de chaque chemin est non exhaustif, spécialement dans le cas des temps d'exécution extrêmes qui nous intéressent pour l'estimation du pWCET. La distribution des temps d'exécution mesurés associés à chaque chemin π_i peut être modélisée par une densité de probabilité f_i. Selon le chemin et ses caractéristiques, les fonctions de répartition f_i ne sont pas forcément identiques. Alors que l'ensemble des temps d'exécution doit être fini, borné par le BCET et le WCET de la tâche, les fonctions de répartition peuvent être définies sur des ensembles infinies comme la loi Normale.

A l'exécution, une mesure de temps d'exécution correspond à tirer de manière aléatoire un temps d'exécution dans l'ensemble Ω_i du chemin exécuté π_i. Pour simuler une tâche, lorsqu'un chemin π_i est exécuté, un temps d'exécution est tiré de manière aléatoire selon la densité de probabilité f_i. Le choix de la fonction f_i est dimensionnant pour coller au mieux à l'ETP de la tâche étudiée. La détermination d'une telle fonction ne sera pas abordée.

Le modèle proposé de simulation du temps d'exécution d'une tâche multichemin pour un processeur temporellement non déterministe est schématisé dans la Figure 8.5.

8.3.2 Approche boîte noire

Les paramètres d'entrée correspondant à l'exécution des chemins π_i sont supposés connus i.e. qu'il est possible d'imposer l'exécution d'un chemin particulier à partir de son ensemble de paramètres d'entrée correspondant. Comme dans les approches MBPTA, nous ne connaissons pas le comportement temporel de chaque chemin, donné par leur densité de probabilité inconnue f_i. De plus, nous ne savons pas comment les chemins sont ordonnés les uns par rapport aux autres i.e. leur temps caractéristique et leur position dans la TSR de la tâche. L'objectif est de privilégier l'exécution des chemins qui favorisent les temps d'exécution les plus longs avec les contraintes énoncées ci-avant.

Le fait de ne pas connaître les fonctions de répartition attribuées à chaque chemin permet d'assimiler la tâche étudiée τ à une boîte noire ainsi schématisée dans la Figure 8.6.

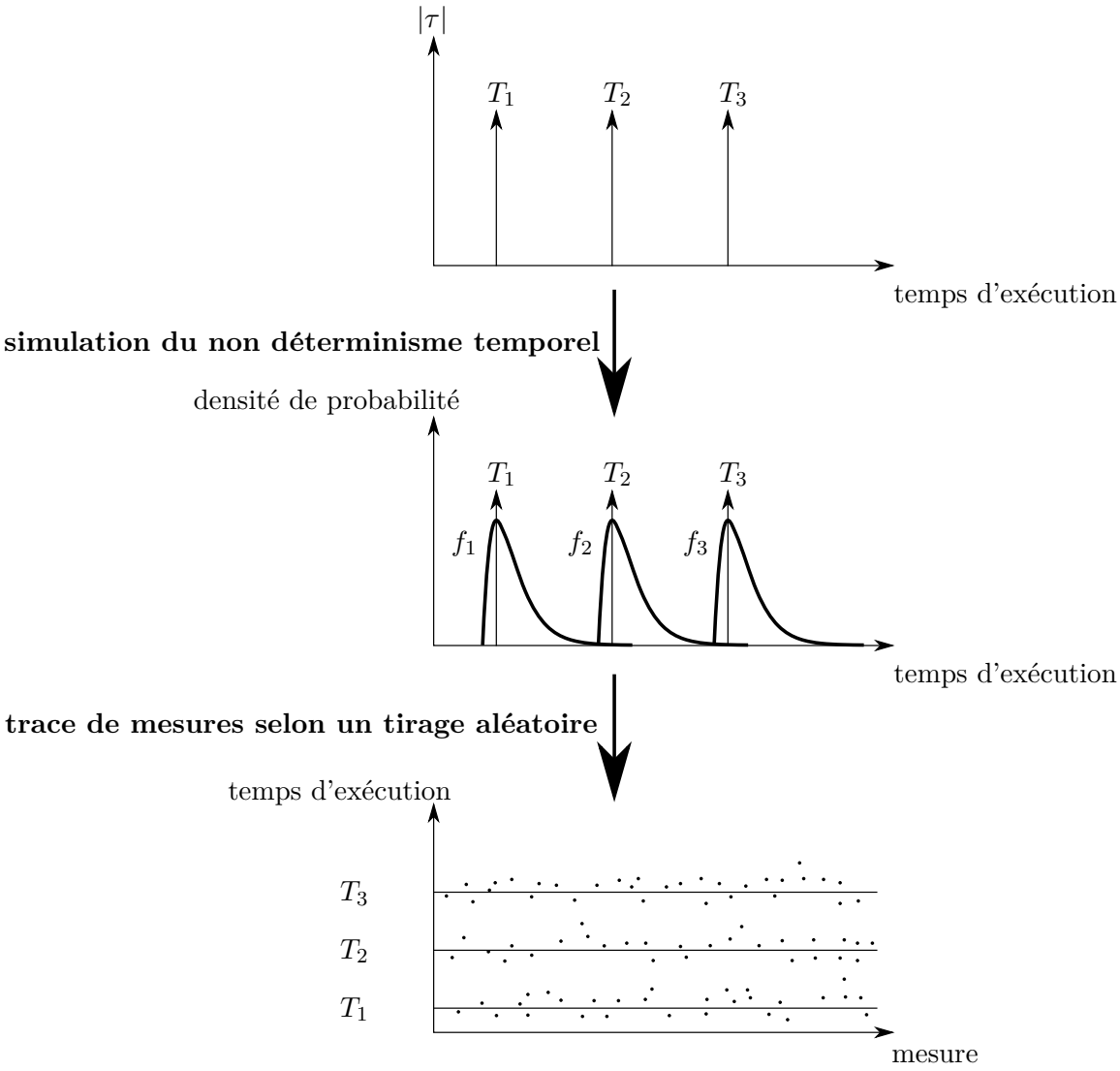


FIGURE 8.5 – Schématisation du modèle de simulation des temps d'exécution d'une tâche τ composée de trois chemins.

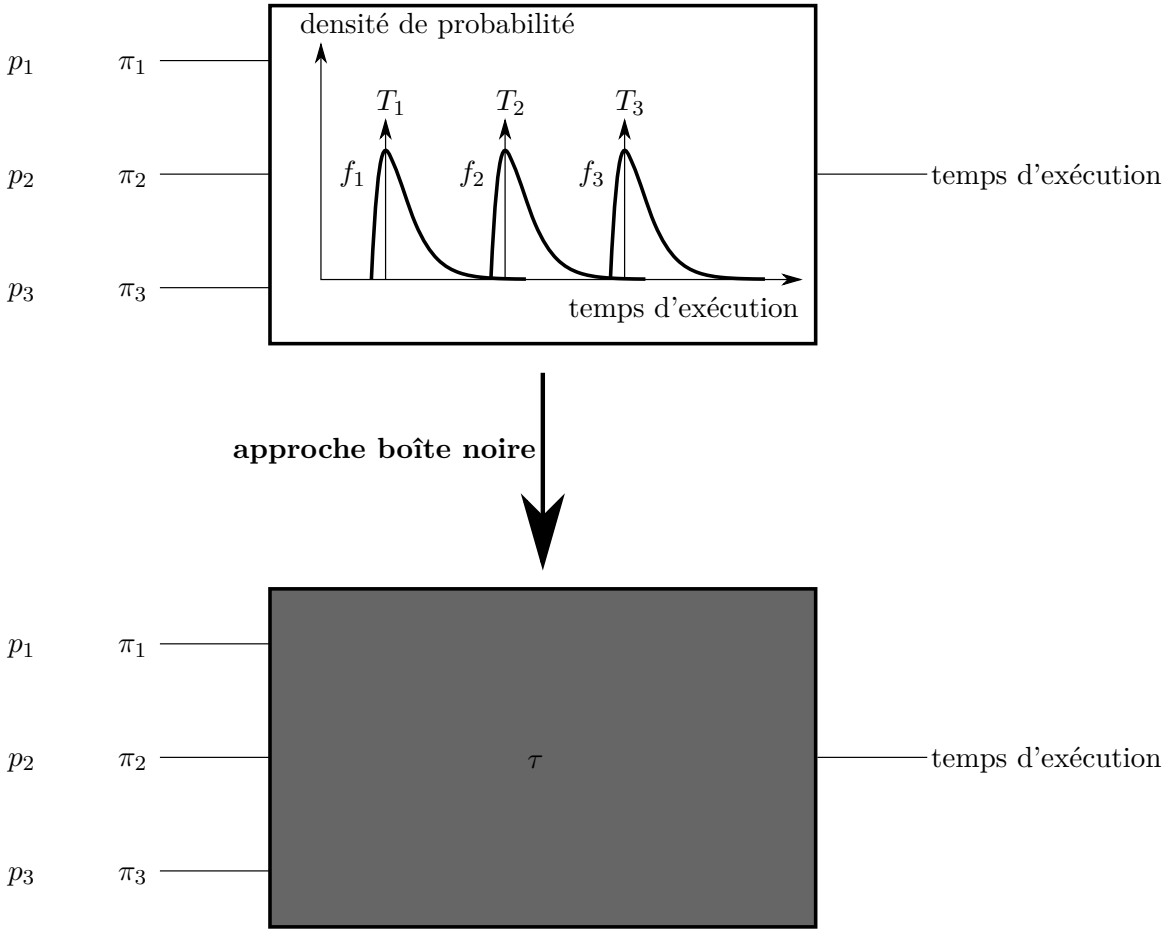


FIGURE 8.6 – Schématisation de l'approche boîte noire de simulation des temps d'exécution d'une tâche τ composée de trois chemins.

Seules les entrées de la boîte noire sont connues i.e. les paramètres d'entrée de la tâche associés à leur chemin π_i. La sortie mesurable de la boîte noire est le temps d'exécution de τ. Le temps d'exécution est la fonction objectif à maximiser. Par exemple, un chemin π_i est exécuté en choisissant d'imposer un de ses paramètres d'entrée correspondant. Ainsi, un temps d'exécution est tiré aléatoirement depuis la fonction f_i inconnue. Le temps d'exécution tiré aléatoirement est retourné par la boîte noire. Etant donné que les chemins sont caractérisés par une distribution de temps d'exécution, une seule mesure de temps d'exécution n'est pas suffisante pour déterminer le comportement temporel d'un chemin.

Au déploiement de l'application, chaque chemin de τ a une probabilité p_i d'être exécuté. Dans une trace, les probabilités p_i doivent être constantes pour toutes les mesures afin d'assurer la stationnarité de la trace. Ces probabilités impactent directement les chances de tirer un temps d'exécution depuis une densité de probabilité donnée. Pour une probabilité p_i les chances de tirer les temps d'exécution extrêmes de la densité de probabilité f_i sont plus faibles.

8.3.3 Favoriser les temps d'exécution les plus longs d'une tâche multi-chemin

Pour estimer un pWCET représentatif avec l'approche MBPTA DIAGXTRM, il est nécessaire de favoriser les temps d'exécution les plus longs. Selon le modèle de simulation de tâche proposé, les temps d'exécution sont directement donnés par les fonctions de répartition f_i. Néanmoins, par construction de la boîte noire, les temps d'exécution sont inconnus et variables. En conséquence, plusieurs temps d'exécution peuvent être obtenus à partir de l'exécution d'un même chemin et en particulier les temps d'exécution les plus longs.

Méthode d'entropie croisée Notre approche est basée sur le tirage aléatoire d'échantillons de temps d'exécution depuis les densités de probabilité f_i. Pour augmenter les chances de mesurer les temps d'exécution les plus longs, il faut modifier les probabilités p_i d'exécution des chemins π_i qui fournissent les temps d'exécution les plus longs. Cette démarche est similaire à la méthode d'entropie croisée (Cross Entropy CE) qui est basée sur la méthode de Monte-Carlo et utilisée afin d'estimer précisément la probabilité d'un évènement rare [Den06].

La méthode CE est également appliquée à des problèmes d'optimisation combinatoire. Cette famille de problèmes cherche à déterminer dans un ensemble discret une solution parmi les meilleures solutions réalisables. Dans notre cas, notre ensemble discret correspond à l'ensemble des probabilités de chemin. La méthode CE vient de l'objectif de minimiser la divergence de Kullback-Leibler D_{KL} entre deux distributions de Q à P : $D_{KL}(P \parallel Q)$ [Kul97, Mac03]. P correspond à la distribution objectif théorique des chemins à exécuter qui maximise les temps d'exécution, alors que Q est la distribution courrante des cemins. En conséquence, la méthode CE va modifier les paramètres de la distribution Q telle qu'elle minimise $D_{KL}(P \parallel Q)$. La distribution objectif P est définie

selon les critères d'optimisation désirés. Dans notre problème on cherche à modifier la distribution d'exécution des chemins de la tâche τ de manière à maximiser le nombre de temps d'exécution les plus longs.

La méthode CE utilise des éléments d'échantillonnage préférentiel [MB15]. Les méthodes d'échantillonnage préférentiel cherchent une distribution qui "encourage" les temps d'exécution les plus importants pour réduire la variance de l'estimation de la probabilité d'évènements rares i.e. réduire l'incertitude d'estimation. Cette distribution étant différente de la "vraie" distribution, elle est biaisée. Les échantillons tirés selon la distribution biaisée sont pondérés de manière à débiaiser l'estimation. La méthode CE est appliquée selon deux étapes :

— Générer une trace de temps d'exécution à partir du modèle boîte noire i.e. selon les probabilités d'exécution des chemins initiales.
— Mettre à jour les probabilités en fonction de la trace obtenue pour mesurer des temps d'exécution plus longs dans une nouvelle trace.

A la fin de l'algorithme, les probabilités d'exécution des chemins sont optimales de manière à maximiser les temps d'exécution. La trace finale est constituée en moyenne des temps d'exécution les plus longs et permet de réduire l'incertitude d'estimation du pWCET.

L'utilisation de la méthode CE est particulièrement intéressante dans notre cadre où les paramètres d'entrée sont discrets : les probabilités d'exécution des chemins. Dans le cas où les chemins et leur probabilité ne sont pas connus et seulement un espace continue d'entrées possibles de la tâche est connu e.g., pour une fonction $foo(x, y, z)$ avec x,y et z des entiers réels, il serait possible de considérer des algorithmes d'optimisation comme le recuit simulé [AvL85], les algorithmes génériques [Mit98] ou la recherche tabou [GL13].

Exemple d'application de la méthode CE Dans cet exemple, on cherche à estimer précisément la probabilité $P_\theta(S(x) > \gamma)$ avec S, la fonction de x : $S(x) = \exp^{(-(x-2)^2)} + 0.8 \times \exp^{(-(x+2)^2)}$. Le vecteur θ est le couple de paramètres de la densité de probabilité f_θ des entrées x de S. La valeur de x suit une loi Normale de moyenne μ et de variance σ^2 tels que $\theta = (\mu; \sigma^2)$. Pour estimer précisément la probabilité cherchée, il faut de nombreux échantillons de S au dessus de γ pour réduire la variance de l'estimation. Pour γ un seuil élevé, la précision peut requérir beaucoup trop d'échantillons suivant une simple approche Monte-Carlo. La méthode CE a pour but de déterminer les paramètres θ qui encourage les échantillons x qui maximisent $S(x)$ et donc minimiser $D_{KL}\left(\mathbf{1}_{S(x)>\gamma}\ P_\theta(S(x) > \gamma) \parallel f_\theta\right)$. Pour modifier les paramètres de θ, des échantillons de x sont générés suivant f_θ. On calcule les valeurs correspondantes de S. On ne retient que les valeurs de x qui donnent les valeurs de S les plus importantes. La moyenne et la variance de ces valeurs de x retenues sont affectées aux paramètres de θ. L'algorithme répète ces deux étapes jusqu'à ce que la variance de l'estimation de $P_\theta(S(x) > \gamma)$ ait convergé vers la précision souhaitée.

8.3.4 La méthode CE en action

Soit une tâche τ représentée par une boîte noire composée de densités de probabilité f_i $i \in [\![1;\Pi]\!]$ simulant le profil de temps d'exécution du chemin respectif π_i. La fonction objectif ou sortie de la boîte noire H est donnée par le temps d'exécution de τ. Chaque chemin a une probabilité p_i d'être exécuté. On note $\hat{p}$ le vecteur de probabilité d'exécution des chemins de τ i.e. $\hat{p} = [p_1, \ldots, p_\Pi]$. Le vecteur de probabilité $\hat{p}$ est une distribution de probabilités multinoulli telle que $\sum_{i=1}^{\Pi} p_i = 1$.

La méthode CE repose sur des itérations pour mettre à jour les probabilités p_i. A la fin des itérations $\hat{p} = \hat{p}^*$, avec $\hat{p}^*$ le vecteur de probabilités qui maximise la fonction objectif H. Le vecteur de probabilités initial est $\hat{p}_0$. Le vecteur de probabilités à la i-ème itération est noté $\hat{p}_i$. Un échantillon X tiré de $\hat{p}$ est un vecteur de zéros sauf le r-ème élément qui vaut 1 donnant le r-ème chemin π_r exécuté. Pour tout vecteur V, $V_{.,j}$ retourne le j-ème élément du vecteur. Alors, pour l'échantillon X, $X_{.,j} = 0$, $\forall j \neq r$ et $X_{.,r} = 1$. Ainsi, $H(X)$ est une variable aléatoire distribuée selon la densité de probabilité f_r.

Algorithme 1 (Méthode d'entropie croisée)**.**

1 Soit $\hat{p}_0$ et $\mathcal{S}$ la suite de temps d'exécution pondérés. Poser $t = 1$ (compteur).

2 Générer N échantillons $X_1, \ldots, X_N$ selon $\hat{p}_{t-1}$. Déduire les temps d'exécution $C_i = H(X_i)$ pour tout i. Classer pas ordre croissant les temps d'exécution $C_{(1)} \leq \ldots \leq C_{(N)}$. Soit γ_t le $1-\rho$ quantile $q(1-\rho)$ des temps d'exécution i.e. $\gamma_t = X_{(\lceil (1-\rho)N \rceil)}$ SI $X_{(\lceil (1-\rho)N \rceil)} > \gamma_{t-1}$ SINON $\gamma_t = \gamma_{t-1}$.

3 Utiliser les échantillons $X_1, \ldots, X_N$ pour calculer $j = 1, \ldots, \Pi$ [DBKMR05],

$$\hat{p}_{t,j} = \frac{\sum_{i=1}^{N} \mathbf{1}_{X_i > \gamma_t} W\left(X_i, \hat{p}_0, \hat{p}_{t-1}\right) X_{i,j}}{\sum_{i=1}^{N} \mathbf{1}_{X_i > \gamma_t} W\left(X_i, \hat{p}_0, \hat{p}_{t-1}\right)}, \tag{8.6}$$

où

$$W\left(X_i, \hat{p}_0, \hat{p}_{t-1}\right) = \sum_{j=1}^{\Pi} \frac{\hat{p}_{0,j}}{\hat{p}_{t-1,j} X_{i,j}}. \tag{8.7}$$

Alors

$$\hat{p}_{t,j} = \frac{\sum_{i=1}^{N} \mathbf{1}_{X_i > \gamma_t} \frac{\hat{p}_{0,j}}{\hat{p}_{t-1,j}} X_{i,j}}{\sum_{i=1}^{N} \mathbf{1}_{X_i > \gamma_t} \frac{\hat{p}_{0,j}}{\hat{p}_{t-1,j}}}. \tag{8.8}$$

4 Ajouter les échantillons dans la séquence des temps d'exécution pondérés i.e. pour tout i $\mathcal{S}.append([W\left(X_i, \hat{p}_0, \hat{p}_{t-1}\right), C_i])$.

5 SI (pour tout j $\frac{|\hat{p}_{(t-1),j} - \hat{p}_{t,j}|}{\hat{p}_{(t-1),j}} < \epsilon$) OU ($\gamma_t = \gamma_{t-1} = \gamma_{t-2}$) ALORS passer à l'étape 6 ; SINON poser $t = t + 1$ et réitérer depuis l'étape 2.

6 Appliquer un tirage aléatoire pondéré par W avec remplacement Nt fois depuis $\mathcal{S}$. Les Nt temps d'exécution tirés donnent la trace $\mathcal{T}$.

7 Appliquer DIAGXTRM à la trace $\mathcal{T}$.

8 Extraire dans $\mathcal{S}_u$ les temps d'exécution pondérés de $\mathcal{S}$ au-dessus du seuil u de la méthode POT retourné par DIAGXTRM à l'étape 7.

9 Appliquer un tirage aléatoire pondéré avec remise depuis $\mathcal{S}_u$. Le tirage aléatoire fournit une trace $\mathcal{T}_u$ de même taille que la séquence $\mathcal{S}_u$.

10 Appliquer DIAGXTRM à $\mathcal{T}_u$.

8.3.5 Détails de la méthode CE

A l'étape 2, ρ est usuellement compris entre 0.1 et 0.01. Les résultats de la méthode CE sont très sensibles à la valeur du paramètre ρ [DBKMR05]. A l'étape 3, il est possible de mettre à jour les probabilités des chemins grâce à la formule donnée cas le vecteur de probabilité suit une distribution multinoulli qui appartient à la famille des distributions exponentielles [DBKMR05]. Aux étapes 6 et 9, un tirage aléatoire consiste à tirer de manière aléatoire un échantillon depuis la séquence spécifiée. Si le tirage est avec remise il est possible de tirer le même échantillon plusieurs fois. Si les échantillons sont pondérés alors tous les échantillons n'ont pas la même probabilité d'être tirés. Pour un poids plus important, un échantillon à plus de chances d'être tiré.

Un tirage aléatoire pondéré est appliqué pour débiaiser les traces de temps d'exécution $\mathcal{S}$ et $\mathcal{S}_u$ issues de la méthode CE. Les traces sont biaisées car les temps d'exécution sont obtenus selon des probabilités de chemin différentes des probabilités initiales. De plus, les traces ne sont pas stationnaires car elles compilent les temps d'exécution issues des tirages pour toutes les itérations i.e. pour des probabilités de chemin différentes entre les itérations. En conséquence, les traces issues du tirage aléatoire pondéré avec remise $\mathcal{T}$ et $\mathcal{T}_u$ sont non biaisées et stationnaires. L'EVT est directement applicable sur ces traces.

A cause du tirage aléatoire pondéré avec remise, il se peut que certains des temps d'exécution les plus longs de $\mathcal{S}$ ne se retrouvent pas dans $\mathcal{T}$, alors que ce sont les valeurs qui nous intéressent le plus. A l'étape 7, DIAGXTRM est appliqué à $\mathcal{T}$ pour déterminer un seuil u à partir duquel les mesures se comportent selon l'EVT. Seules les valeurs de $\mathcal{S}$ au-dessus du seuil u sont conservées dans la séquence $\mathcal{S}_u$. Pour avoir une trace non biaisée et stationnaire, un nouveau tirage aléatoire pondéré avec remise est appliqué à $\mathcal{S}_u$ donnant la trace $\mathcal{T}_u$ qui présente davantage de temps d'exécution élevés que dans $\mathcal{T}$.

Les résultats de DIAGXTRM appliqué à $\mathcal{T}_u$ sont les résultats définitifs pour la tâche étudiée τ. Les WCET seuils n'ont pas besoin d'être modifiés car nous avons appliqué l'EVT sur une trace non biaisée. Les probabilités des événements rares sont donc correctes.

L'application de la méthode CE pour l'étude des mesures de temps d'exécution d'une tâche soulève deux enjeux. Le premier enjeu est l'utilisation de la méthode dans le cas d'une fonction objectif non déterministe. En effet, pour les problèmes d'optimisation usuels d'application de la méthode CE, la fonction objectif est généralement déterministe telle que pour un même ensemble de paramètres d'entrée, la fonction objectif retourne la même sortie. Dans notre cas, pour un même ensemble de paramètres d'entrée, la fonction objectif retourne une valeur tirée aléatoirement selon la densité de probabilité du chemin

exécuté. Dans le cas déterministe, il existe davantage de combinaisons de paramètres d'entrée. Alors que dans notre cas, il existe autant de combinaisons que de chemins existants. Le deuxième enjeu, est l'application de l'EVT sur une trace biaisée. Pour obtenir des estimations en fonction des probabilités initiales des chemins, il est nécessaire de débiaiser la trace. Cette procédure correspond aux étapes 9 et 10 de l'algorithme proposé.

8.4 Evaluation expérimentale de la méthode CE

8.4.1 Cas d'étude

Pour illustrer la méthode CE, nous considérons un cas d'étude expérimentale basé sur le modèle de simulation de tâche proposé. Conformément au modèle de simulation, le comportement temporel de chaque chemin π_i est donné par une densité de probabilité f_i. Pour ce cas d'étude nous nous restreignons au cas d'une densité de probabilité possible :

gumbel(a, b) **:** densité de probabilité de la loi de Gumbel de localisation a et de forme b. La densité de probabilité est définie sur un support infinie.

Les fonctions de répartition sont définies sur l'ensemble des temps d'exécution x qui font partie des réels positifs. Le paramètre de localisation a du chemin π_i correspond dans ce cas au temps caractéristique respectif T_i comme défini dans l'analyse spectrale de tâche. Comme cette étude n'est pas basée sur l'étude de tâches réelles, chaque couple de paramètres (a, b) peut être choisi de manière arbitraire.

8.4.2 Exemple pas à pas

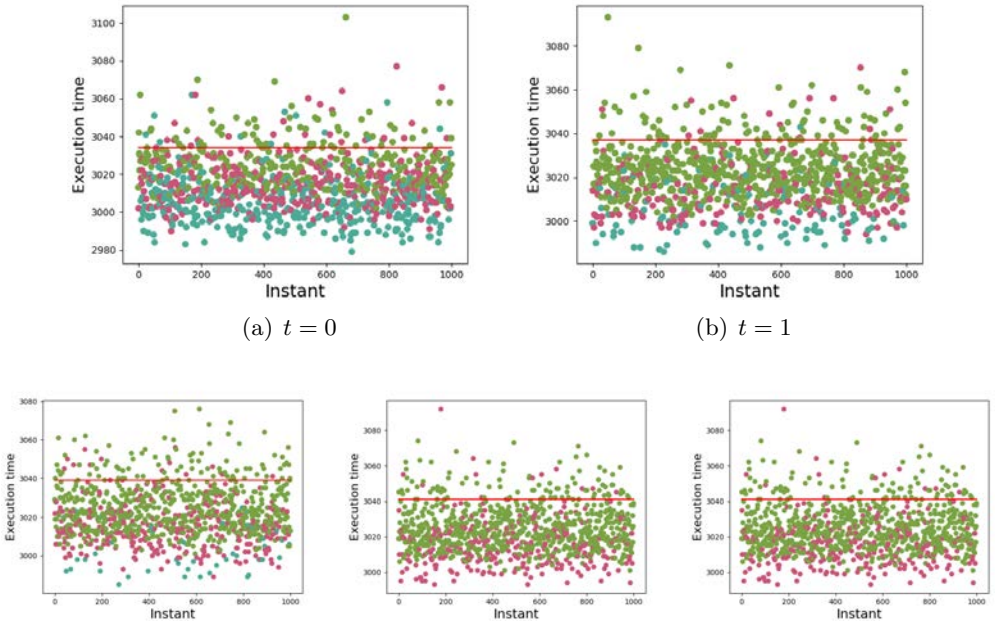
Pour cette étude, nous fixons les paramètres de la méthode CE tels que $N = 1000$, $\rho = 0.1$, $\epsilon = 0.1$. Nous appliquons la méthode CE pour la tâche τ composée de trois chemins π_1, π_2, π_3. Les probabilités initiales d'exécution des chemins sont toutes fixées à un tiers i.e. $\hat{p}_0 = [1/3, 1/3, 1/3]$ où $p_i = 1/3$ pour tout i de 1 à 3. Les fonctions de répartition associées aux chemins sont imposées comme $f_1(x) = \text{gumbel}(3000, 10)(x)$, $f_2(x) = \text{gumbel}(3010, 10)(x)$ et $f_3(x) = \text{gumbel}(3020, 10)(x)$. Les mises à jour des probabilités d'exécution des chemins selon le pas t de la méthode CE sont données dans la Table 8.3.

t	p_1	p_2	p_3	γ_t
0	1/3	1/3	1/3	-
1	0.1	0.2	0.7	3034
2	0.0	0.3	0.7	3037
3	0.0	0.3	0.7	3039
4	0.0	0.3	0.7	3041

TABLE 8.3 – Probabilités d'exécution des chemins selon le pas t de la méthode CE appliquée à τ.

Les trois chemins possèdent théoriquement le même comportement temporel car ils sont définis par la même densité de probabilité. Les chemins diffèrent entre eux par leur temps caractéristiques qui donnent π_3 comme le chemin le plus long. En conséquence, la méthode CE devrait converger vers un vecteur de probabilités qui favorise l'exécution du chemin π_3.

La Table 8.3 montre bien que le chemin π_3 est favorisé à l'issue de la méthode CE. Dans cet exemple, la procédure a convergé en 4 itérations vers le vecteur de probabilités $\hat{p}_4 = [0, 0.3, 0.7]$, le troisième chemin est bien le chemin favorisé. Les traces issues de chaque itération sont données dans la Figure 8.7.



(a) $t = 0$ (b) $t = 1$

(c) $t = 2$ (d) $t = 3$ (e) $t = 4$

FIGURE 8.7 – Traces de mesures pour chaque itération de la méthode CE appliquée à la tâche exemple. Les mesures en bleues sont les temps d'exécution issus de l'exécution du chemin π_1, en rose du chemin π_2 et en vert du chemin π_3.

Les différentes figures incluent le seuil γ_t de chaque itération qui augmente en fonction des itérations. En conséquence, la moyenne des temps d'exécution est plus importante au fur et à mesure des itérations. Pour avoir un pWCET sûr et représentatif, on choisit un scénario de mesure tel que les probabilités d'exécution des chemins soient comme celles du vecteur $\hat{p}_4$. De plus, la méthode CE maximise en moyenne les mesures de temps d'exécution puisque le seuil γ_t augmente à mesure que le vecteur de probabilités est mis à jour.

Pour comparer, nous considérons la tâche τ' caractérisé par les mêmes chemins que τ excepté π_2 dont la densité de probabilité est désormais $f_2(x) = \text{gumbel}(3010, 20)(x)$.

La distribution des temps d'exécution de π_2 est plus dispersée car la déviation standard de f_2 est plus grande. Pour les mêmes paramètres de la méthode CE, nous obtenons le vecteur de probabilités $\hat{p}_4 = [0.0, 0.8, 0.2]$. Dans ce cas, les deux chemins π_2 et π_3 participent à l'obtention de temps d'exécution plus longs pour un pWCET représentatif. Toujours dans le cas de τ', ρ est maintenant fixé à 0.01 et ainsi nous obtenons comme vecteur de probabilités final $\hat{p}_4 = [0.0, 1.0, 0.0]$. Ainsi, la valeur de ρ discrimine davantage les temps d'exécution moyens fournis par π_2.

8.4.3 Etude de l'application de la méthode CE

Dans le cas d'étude précédent, nous avons mis en évidence les bénéfices de la méthode CE. En particulier, dans ce cas nous n'avons aucune information sur le comportement temporel de la tâche en fonction de ses chemins et pour une plateforme inconnue, la méthode CE arrive à isoler le pire chemin pour favoriser le nombre et la valeur des temps d'exécution les plus longs. Dans cette étude nous considérons des tâches qui diffèrent par leurs chemins i.e. les probabilités d'exécution, les localisations et les déviations standards. Nous étudions également l'influence de certains paramètres de la méthode CE comme le nombre de mesures par itération N et le ρ-quantile.

Dans tous les cas, la tâche étudiée est composée de deux chemins i.e. le pire chemin et un ensemble de chemins moins pires i.e. qui fournissent en moyenne des temps d'exécution inférieurs, regroupés sous un seul autre chemin. Ce cas est analogue à un cas où la tâche serait composée de davantage de chemins, le moins pire regroupant un ensemble de chemins. Les différentes tâches considérées sont données dans la Table 8.4. Les tâches d'indice impair diffèrent par leur pire chemin. Dans les cas 1 et 3, le pire chemin est le chemin π_2, et dans le cas 5 il s'agit de π_1. Dans ces trois cas, les probabilités des chemins sont uniformément distribuées. Les tâches d'indice pair reprennent les chemins des tâches d'indice impair, et diffèrent par leurs probabilités d'exécution des chemins tel que le pire chemin soit largement défavorisé i.e. probabilité de 0.1.

tâche	π_1	π_2
τ_1	$f_1(x) = \text{gumbel}(3000, 100)(x)$, $p_1 = 1/2$	$f_2(x) = \text{gumbel}(4000, 100)(x)$, $p_2 = 1/2$
τ_2	$f_1(x) = \text{gumbel}(3000, 100)(x)$, $p_1 = 0.9$	$f_2(x) = \text{gumbel}(4000, 100)(x)$, $p_2 = 0.1$
τ_3	$f_1(x) = \text{gumbel}(3000, 100)(x)$, $p_1 = 1/2$	$f_2(x) = \text{gumbel}(3100, 100)(x)$, $p_2 = 1/2$
τ_4	$f_1(x) = \text{gumbel}(3000, 100)(x)$, $p_1 = 0.9$	$f_2(x) = \text{gumbel}(3100, 100)(x)$, $p_2 = 0.1$
τ_5	$f_1(x) = \text{gumbel}(3000, 200)(x)$, $p_1 = 1/2$	$f_2(x) = \text{gumbel}(3100, 100)(x)$, $p_2 = 1/2$
τ_6	$f_1(x) = \text{gumbel}(3000, 200)(x)$, $p_1 = 0.1$	$f_2(x) = \text{gumbel}(3100, 100)(x)$, $p_2 = 0.9$

TABLE 8.4 – Tâches pour étudier les bénéfices de la méthode CE.

Pour étudier les bénéfices de l'utilisation de la méthode CE, nous allons comparer certaines métriques de la trace issue de la méthode CE i.e. jusqu'à l'étape 5 de l'algorithme proposé, à une trace dite naïve issue d'un tirage aléatoire selon le vecteur de probabilités initial $\hat{p}_0$ et de même taille que la trace issue de la méthode CE. L'intérêt de la méthode CE est de favoriser le nombre de temps d'exécution les plus longs. Pour

discuter cet argument nous allons comparer les quantiles à 70% des deux traces $q(0.7, \mathcal{T})$, les nombres de pics de dépassement k au dessus du quantile à 70% de la trace issue de la méthode CE, et les maximums de chaque trace. Comme nous adressons le problème d'une fonction objectif non déterministe, la comparaison sera faite par rapport à la moyenne, fonction μ, et au maximum relevé, fonction max de chaque métrique sur 100 itérations. Dans un premier temps, nous appliquons la méthode CE selon les paramètres $N = 1000$, $\rho = 0.01$, et $\epsilon = 0.1$. Les résultats de cette première étude sont donnés dans la Table 8.5.

tâche	méthode	t	$\mu(q(0.7, \mathcal{T}))$	$\max(q(0.7, \mathcal{T}))$	$\mu(k)$	$\max(k)$	$\mu(\max(\mathcal{T}))$	$\max(\max(\mathcal{T}))$
τ_1	CE	2.00	4066.47(+0.29)	4075.00(+0.28)	597.70(+0.50)	600.00(+0.32)	4791.86(+0.00)	5297.00(+0.00)
-	naïve	.	3147.45	3174.30	397.22	453.00	4750.44	5293.00
τ_2	CE	2.01	4021.90(+0.33)	4032.00(+0.33)	597.83(+4.39)	600.00(+3.35)	4756.96(+0.04)	5303.00(+0.06)
-	naïve	.	3020.54	3030.00	110.88	138.00	4572.27	5002.00
τ_3	CE	3.60	3179.65(+0.04)	3192.00(+0.04)	1014.13(+0.17)	1794.00(+0.05)	3948.99(+0.00)	4508.00(+0.01)
-	naïve	.	3059.05	3064.00	869.31	1702.00	3932.88	4463.00
τ_4	CE	4.07	3126.77(+0.04)	3168.00(+0.05)	1195.25(+0.11)	3000.00(+0.06)	3900.53(-0.00)	4441.00(+0.04)
-	naïve	.	3017.32	3026.00	1081.02	2817.00	3918.62	4264.00
τ_5	CE	2.34	3204.89(+0.01)	3222.30(+0.04)	719.19(+0.00)	900.00(-0.09)	4592.93(+0.02)	5519.00(+0.03)
-	naïve	.	3078.78	3089.00	717.79	988.00	4523.14	5339.00
τ_6	CE	3.62	3203.65(+0.03)	3213.00(+0.03)	1097.25(+0.00)	2390.00(+0.02)	4594.83(+0.06)	5370.00(+0.05)
-	naïve	.	3103.35	3110.00	1092.46	2351.00	4316.60	5124.00

TABLE 8.5 – Bénéfices de la méthode CE. Les valeurs issues de la méthode CE sont comparées en pourcentage (±%) aux valeurs correspondantes de la méthode naïve.

Les résultats de la Table 8.5 confirment bien les bénéfices attendus de la méthode CE par rapport aux nombres de temps d'exécution longs plus importants. L'application de la méthode apporte davantage de bénéfices dans les cas où le pire chemin est nettement différent du chemin moyen i.e. τ_1 et τ_2. Le bénéfice est encore plus important dans le cas où le pire chemin est défavorisé i.e. τ_2. Dans les autres cas où les chemins sont plus rapprochés, les bénéfices de la méthode CE sont moins nets mais non négligeables. Ce résultat fait sens puisque les deux chemins participent aux pires mesures. La méthode CE permet dans tous les cas d'obtenir des temps d'exécution plus représentatifs du pire cas pour l'estimation du pWCET.

Les paramètres de la méthode CE, comme le nombre de tirages par itération N, et le ρ-quantile influencent sensiblement les résultats. Dans le cas de la tâche τ_6 où les deux chemins participent aux temps d'exécution les plus, nous étudions l'effet du paramètre N. Les résultats sont donnés dans la Table 8.6. La valeur de ρ est fixée à 0.01.

Les performances de la méthode CE restent stables de manière globale. Dans tous les cas, le pire chemin de τ_6 est π_1 et est favorisé avec une probabilité modifiée de 0.1 à 0.7 en moyenne. Les différences apparaissent entre les cas où N vaut 1000 et 2000, et les cas où N vaut 5000 et 10000. Les deux derniers cas nécessitent en moyenne une itération de plus pour converger. Avec un nombre de tirages plus grand, il devient plus difficile d'identifier le pire chemin lorsque les deux chemins sont proches. De plus, les deux derniers cas arrivent à favoriser un maximum plus grand de presque 0.2 fois le maximum dans le cas naïf, contre une augmentation de 0.05 dans les deux premiers cas.

N	méthode	t	$\mu(q(0.7,\mathcal{T}))$	$\max(q(0.7,\mathcal{T}))$	$\mu(k)$	$\max(k)$	$\mu(\max(\mathcal{T}))$	$\max(\max(\mathcal{T}))$
1000	CE	3.62	3203.65(+0.03)	3213.00(+0.03)	1097.25(+0.00)	2390.00(+0.02)	4594.83(+0.06)	5370.00(+0.05)
-	naïve	.	3103.35	3110.00	1092.46	2351.00	4316.60	5124.00
2000	CE	3.63	3203.50(+0.03)	3216.00(+0.03)	2169.29(+0.01)	4195.00(+0.02)	4755.08(+0.06)	5927.00(+0.07)
-	naïve	.	3103.70	3109.00	2158.17	4118.00	4496.80	5527.00
5000	CE	4.37	3203.70(+0.03)	3208.00(+0.03)	6524.72(+0.01)	11953.00(+0.01)	4981.38(+0.07)	6154.00(+0.12)
-	naïve	.	3103.53	3107.00	6470.22	11793.00	4652.56	5475.00
10000	CE	4.89	31203.55(+0.03)	3206.00(+0.03)	14622.14(+0.01)	26953.00(+0.00)	5129.44(+0.07)	6379.00(+0.19)
-	naïve	.	3103.65	3105.00	14503.16	26874.00	4794.12	5374.00

TABLE 8.6 – Influence du nombre de tirages par itération N sur la méthode CE dans le cas de la tâche 6 τ_6.

Avec un nombre de tirages plus important, les chances de tirer une valeur plus grande sont également plus importantes. Pour un nombre de tirages $N = 20000$, la méthode CE converge en 5.54 itérations en moyenne, pour un gain en moyenne sur le maximum tiré de 0.09. Davantage de valeurs sont tirées dans le cas naïf, et il devient de plus en plus difficile de tirer une valeur supérieure avec le méthode CE.

Toujours dans le cas de la tâche 6, on étudie l'influence de la valeur du ρ-quantile sur les performances de la méthode CE dans la Table 8.7. La valeur de N est fixée à 1000.

ρ	méthode	t	p_1	$\mu(q(0.7,\mathcal{T}))$	$\max(q(0.7,\mathcal{T}))$	$\mu(k)$	$\max(k)$	$\mu(\max(\mathcal{T}))$	$\max(\max(\mathcal{T}))$
0.1	CE	3.66	0.18	3203.01(+0.03)	3212.3(+0.03)	1092.71(+0.00)	2694.00(-0.03)	4340.33(+0.01)	5092.00(+0.04)
.	naïve	.	.	3103.38	3110.00	1088.98	2783.00	4287.14	4912.00
0.05	CE	3.90	0.27	3202.69(+0.03)	3215.30(+0.03)	1165.78(+0.00)	2998.00(+0.03)	4418.65(+0.02)	5127.00(+0.01)
.	naïve	.	.	3103.32	3109.00	1163.30	2910.00	4314.01	5061.00
0.02	CE	4.33	0.60	3203.22(+0.03)	3212.00(+0.03)	1295.96(+0.00)	2699.00(-0.02)	4617.72(+0.07)	5609.00(+0.07)
.	naïve	.	.	3103.56	3108.00	1291.56	2756.00	4327.44	5219.00
0.01	CE	3.62	0.75	3202.97(+0.03)	3215.30(+0.03)	1081.72(+0.00)	2099.00(-0.02)	4593.52(+0.07)	5507.00(+0.08)
.	naïve	.	.	3103.53	3110.00	1078.86	2155.00	4268.34	5121.00

TABLE 8.7 – Influence de la valeur du ρ-quantile sur la méthode CE dans le cas de la tâche 6 τ_6.

Dans ce cas particulier de la tâche 6, les chemins de la tâche sont proches et tels que le chemin avec le plus grand temps caractéristique n'est pas le pire chemin, la valeur de ρ impacte le vecteur de probabilité final. En effet, dans les deux derniers cas seulement, le pire chemin avéré est favorisé. La valeur du ρ-quantile est bien plus discriminante dans ces deux derniers. Ainsi, l'algorithme privilégie les valeurs extrêmes aux valeurs fréquentes, puisque le pire chemin était défavorisé initialement. Néanmoins, les performances de la méthode CE restent plutôt stables par rapport à la valeur du ρ-quantile. Ceci est dû au fait que les deux chemins participent aux temps d'exécution les pus longs. Enfin, la valeur de ρ dépend également de N puisque ρ ne peut être inférieur à $10 \times 1/N$ sinon aucune valeur ne serait conservée à l'étape 2 de l'algorithme.

8.5 Discussion

8.5.1 Application de DIAGXTRM

L'objectif de ce chapitre a consisté a choisir les chemins à mesurer pour estimer un pWCET sûr et représentatif avec DIAGXTRM. Pour ce faire, nous avons mis en place la méthode CE dans l'Algorithme 1 pour favoriser les temps d'exécution les plus longs à partir de la probabilité d'exécution des chemins de la tâche. Néanmoins, l'application de DIAGXTRM à la trace issue de la méthode CE n'est pas directe. En effet, il est nécessaire, à l'étape 7 de l'algorithme 1, de débiaiser la trace issue de la méthode CE pour estimer le pWCET avec l'EVT. Enfin, pour ne pas perdre les bénéfices de l'application de la méthode CE i.e. ne pas perdre les temps d'exécution favorisés par la méthode, il faut appliquer les étapes 8 et 9 avant d'appliquer de nouveau DIAGXTRM et conclure sur le pWCET estimé. Les auteurs de [TE15] étudient l'application de l'EVT à une trace d'échantillons biaisés. De futurs travaux pourraient considérer cette étude pour appliquer l'EVT directement à la trace biaisée.

8.5.2 Nombre de chemins

Dans la section d'étude de la méthode CE, nous avons considéré des cas avec seulement deux chemins. Le plus petit chemin pouvant être une distribution de mélange de différents chemins, les résultats seraient sensiblement similaires. Le nombre de chemins a un impact direct sur le nombre N de mesures par itération. Pour une tâche composée de 1000 chemins différents distribués de manière uniforme il faut au moins $N = 10000$ mesures pour réussir à appliquer la méthode CE et espérer converger vers le pire chemin. Pour $N = 10000$, la méthode naïve de mesure ne donnerait qu'en moyenne 10 mesures selon le pire chemin. La représentativité du pWCET estimé dans ce cas est donc limité. L'intérêt de la méthode CE dans le cas d'une tâche composée d'un nombre important de chemins est encore plus grand puisqu'elle permet d'éviter de mesurer tous les chemins non représentatifs par la suite.

8.5.3 Connaissance des entrées de la tâche

L'application de la méthode CE comme nous le proposons repose sur la base de connaissance des entrées de chaque chemin pour appliquer la méthode CE. Pour garantir la sûreté de l'estimation du pWCET il est nécessaire d'effectuer une recherche exhaustive des chemins de la tâche étudiée. Plusieurs études de recherche des chemins de la tâche existent en dehors de la considération de l'application de l'EVT [PN98, WMMR05, WR09, BZTK11] Néanmoins, comme les méthodes d'estimation du WCET basées sur l'EVT emploient des mesures, il est nécessaire de garantir la couverture des chemins [LB16] pour un pWCET estimé sûr. Les auteurs de [LB16] utilisent un algorithme de recuit simulé et différentes fonctions basées sur les exigences des analyses MBDTA pour produire une couverture des chemins fiable. Ces fonctions visent à générer les temps d'exécution les plus larges, à exécuter toutes les branches du code et maximiser le nombre d'itération des boucles. Cependant, l'EVT n'a pas été

appliquée à la sortie de cette procédure de couverture. De futurs travaux pourraient s'intéresser à récupérer les informations issues de la procédure de [LB16] pour les utiliser dans la méthode CE proposée. Par ailleurs, des méthodes de programmation par contraintes pourraient être considérées pour obtenir la liste exhaustive des entrées possibles d'une tâche multichemin. Enfin, une étude de l'environnement du système permettrait de déterminer les probabilités d'exécution des chemins i.e. la fréquence des entrées différentes de la tâche.

8.5.4 Application de la méthode CE à *ns*

L'application de la méthode CE au cas d'étude *ns* est assez directe. En effet, nous avons montré que la distance moyenne entre les quatre chemins de la tâche était de l'ordre de 1700 cycles. La tâche est une première fois exécutée avec des entrées aléatoires, cas de la Figure 8.3(b). Seuls les temps d'exécution du pire chemin π_4 sont conservés du fait de la distance significative entre les chemins. Ainsi, seul le chemin π_4 est exécuté à la deuxième itération, cas de la Figure 8.3(a). Ce cas est de nouveau exécuté, et l'algorithme converge vers le vecteur de probabilités $\hat{p}_3 = [0.0, 0.0, 0.0, 1.0]$. Dans ce cas où les chemins sont significativement distants les uns des autres, l'algorithme converge vers le cas où seul le pire chemin est exécuté.

8.5.5 Automatisation de la chaîne de travail

Dans ce chapitre, nous n'avons pu considérer que le cas de la simulation du comportement des tâches pour étudier la méthode CE proposée. La discussion précédente montre le potentiel d'application de la méthode CE à des cas réels. Pour ce faire, il faut intégrer dans la méthode de mesure les étapes 3, 4 et 5 de l'algorithme 1. En effet les étapes 1 et 2 sont classiques dans les méthodes de mesure considérées jusqu'ici.

Une autre méthode pourrait consister à déterminer la TSR de la tâche étudiée avec des mesures exhaustives des chemins de la tâche. A partir de la TSR, il faudrait modéliser précisément le comportement moyen de chaque chemin i.e. déterminer les densités de probabilité f_i. Enfin, il sera possible d'utiliser les moyens de simulation développés pour appliquer la méthode CE à la tâche étudiée et faire de nouvelles mesures selon les nouvelles probabilités d'exécution des chemins.

8.6 Conclusion

Nous proposons dans ce chapitre une représentation spatiale du comportement temporel de tâches multichemin pour faciliter l'analyse temporelle. La TSR de la tâche est utilisée pour caractériser les chemins de la tâche et pour simuler son comportement temporel. Pour ce faire, chaque chemin est caractérisé par un temps caractéristique et une densité de probabilité qui permettent de simuler la variabilité du temps d'exécution due au système. Afin de déterminer un scénario de mesure représentatif du cas multichemin et sûr, nous considérons l'application de la méthode CE. La méthode CE étant utilisée

dans des problèmes d'optimisation combinatoire, elle satisfait nos besoins pour favoriser les chemins qui vont amener aux temps d'exécution les plus longs.

L'analyse du pire scénario multichemin fournit par la méthode CE est appliqué dans le cas de tâche dont le comportement temporel est simulé. L'étude considère différentes valeurs des paramètres de la méthode CE pour mettre en évidence la robustesse de la technique. En définitive, la méthode CE arrive en moyenne à converger vers un pire scénario de mesure représentatif du cas multichemin. La procédure retourne un vecteur de probabilités d'exécution des chemins tel que les chemins qui génèrent les temps d'exécution les plus longs soient favorisés.

Cependant, la méthode proposée reste dépendante de la connaissance exhaustive des chemins et des paramètres d'entrées respectifs de la tâche.

Cinquième partie

Conclusions

8.7 Bilan

L'analyse de l'état de l'art des méthodes d'estimation du pire temps d'exécution nous a amené à divers constats : (i) les approches sont difficilement généralisables aux différentes architectures matérielles existantes ; les méthodes statiques peuvent être généralisées mais au prix d'une modélisation fine et coûteuse du système temps réel au niveau matériel et logiciel et pour chaque cas ; les méthodes basées mesure fournissent des résultats rapidement et sont facilement généralisables mais souffrent d'un manque de confiance dans l'observation du pire cas ; (ii) les approches probabilistes basées mesures qui appliquent la théorie des valeurs extrêmes se posent comme étant rapides, simples et fiables, et permettent de tirer parti des optimisations des processeurs jusqu'à un certain niveau ; néanmoins les différentes approches existantes manquent de systématisme et considèrent le cas de mesures indépendantes et identiquement distribuées qui n'est pas forcément adapté ; (iii) les approches probabilistes basées mesures cherchent à inférer le pire cas du système à partir de ce qui est mesurable ; cependant certaines conditions ne peuvent être inférées (conditions matérielles, paramètres d'entrée), et les approches probabilistes basées mesures souffrent ainsi d'un manque de confiance par la communauté temps réel.

Les travaux de cette thèse tentent de répondre à ces constats en proposant un ensemble de méthodologies pour systématiser l'évaluation de la fiabilité des estimations de pire temps d'exécution probabiliste à partir de l'application de la théorie des valeurs extrêmes à une suite de mesures, et déterminer des conditions de mesure pour des estimations robustes et représentatives du pire cas. Une partie applicative des ces travaux a pu montrer la simplicité de l'approche développée mais également ses limites.

En premier lieu, nous proposons d'étudier l'application de la théorie des valeurs extrêmes à une suite de mesures stationnaires. Ce cadre d'étude pose des conditions plus relâchées que le cas de mesures indépendantes et identiquement distribuées, mais surtout plus adaptées au cas de l'étude de processeurs du commerce. L'étude de l'applicabilité de la théorie a débouché sur un ensemble de conditions à assurer et la mise en place d'autant de tests d'hypothèse. Pour systématiser l'évaluation des traces de mesures, nous proposons un outil appelé DIAGXTRM pour le DIAGnostic de l'application de la théorie des valeurs eXTRêMes. Les résultats des tests d'hypothèse sont traduits sous la forme de la logique floue. De cette manière, pour une trace donnée à DIAGXTRM, l'outil retourne une estimation de pire temps d'exécution probabiliste associé à un niveau de confiance sur la pertinence de l'application de la théorie.

De plus, l'estimation du pire temps d'exécution probabiliste est sensible à certains paramètres qui découlent de l'application de la théorie des valeurs extrêmes selon l'approche des pics de dépassement. En effet, l'approche étant basée sur des mesures, elle ne permet pas de déterminer formellement les paramètres du pire temps d'exécution probabiliste. Elle consiste à approcher un modèle continue à partir de mesures discrètes et le nombre de mesures limité met en danger la reproductibilité de l'estimation dans

les mêmes conditions. C'est pourquoi, nous avons développé des algorithmes, intégrés dans DIAGXTRM, qui ont pour but de pallier à ces problèmes d'estimation dus aux incertitudes de mesure. Ces nouveaux algorithmes permettent notamment de gagner en confiance sur l'estimation du pire temps d'exécution probabiliste.

L'outil DIAGXTRM a été appliqué à un ensemble de traces de mesures issues de différents cas d'étude qui considèrent des architectures matérielles différentes. Ce cadre applicatif a permis de montrer que l'estimation fournie du pire temps d'exécution probabiliste modélise mieux les mesures que dans les études existantes dans lesquels le modèle était contraint à la loi de Gumbel. Ainsi, les estimations fournies par DIAGXTRM sont plus précises. La majorité des algorithmes d'ordonnancement ne considèrent qu'une valeur déterministe du pire temps d'exécution. Il est donc nécessaire de déterminer une probabilité de dépassement pour inférer un pire temps d'exécution à partir du pire temps d'exécution probabiliste estimé. Le choix de cette probabilité relève d'un compromis entre précision et sûreté car plus la probabilité est proche de 0, plus le pire temps d'exécution inféré est élevé. Dans cette thèse, la probabilité est fixée à 10^{-9} relativement aux probabilités de défaillance requises généralement dans l'aéronautique. Ce niveau requis challenge davantage les estimations puisque nous considérons des traces de taille d'ordre de grandeur 10^3. Ainsi, dans certains cas de l'étude applicative, le pire temps d'exécution inféré est fiable, précis et les incertitudes dues aux mesures faibles. Néanmoins, dans d'autres cas, les valeurs inférées sont trop pessimistes ou alors elles ne sont pas fiables. Si les valeurs sont trop pessimistes, alors dans la plupart des cas c'est par manque de mesures, parce que certains états matériels sont difficiles à reproduire et peu fréquents. Ces comportements matériels difficiles à reproduire peuvent être des rafraichissements peu fréquents des mémoires qui donnent des pics de temps d'exécution discontinues et donc difficiles à considérer dans le cas d'un modèle continue. Pour les cas où l'estimation n'est pas fiable, il se peut que la trace manque de valeurs i.e. qu'elle ne soit pas assez variable ou que la trace exhibe un comportement temporel qui n'est pas en accord avec les exigences de la théorie des valeurs extrêmes. Le manque de variabilité vient du processeur qui isole temporellement l'exécution de la tâche mesurée et ainsi son temps d'exécution demeure constant. Avec l'introduction d'interférences matérielles, la trace exhibe alors un comportement variable qui permet l'application fiable de la théorie des valeurs extrêmes. D'autres comportements non désirés sont dûs à la variation des paramètres d'entrée de la tâche étudiée qui change le mode d'exécution de la tâche. Si ces changements de mode sont déterministes alors l'estimation ne peut pas être fiable. En conséquence, il est nécessaire de connaître les entrées possibles de la tâche et les exécuter en accord avec les exigences de la théorie des valeurs extrêmes. Enfin, il apparaît que certains tests peuvent s'avérer quelque peu restrictifs alors que le reste des hypothèses est validé et que l'observation de la trace ne montre pas de comportement spécifique. Nous avons fait le choix d'inclure la totalité des conditions dans DIAGXTRM pour avoir le plus de confiance dans les estimations diagnostiquées comme fiables de manière systématique et rapide. Cependant, il peut s'avérer utile de discuter les résultats à la lumière d'analyses graphiques mais ces analyses peuvent être difficiles pour un concepteur de système temps réel.

Cette étude applicative nous a amené à considérer les besoins de moyens pour déterminer les conditions de mesures qui seraient en accord mais aussi favoriseraient l'application de la théorie des valeurs extrêmes. Pour déterminer de telles conditions, il est nécessaire d'avoir une connaissance, jusqu'à un certain degré, du matériel et du logiciel composant le système temps réel étudié. Le degré de connaissance doit rester faible pour conserver les avantages d'utiliser une approche d'estimation du pire temps d'exécution basée mesures, mais suffisant pour donner confiance dans l'estimation.

Nos observations sur les interférences matérielles nous ont mené à l'élaboration d'un algorithme pour déterminer le scénario de mesures qui contiendrait le plus haut niveau d'interférences. Cette étude s'inscrit dans le cadre de tâches monochemin qui s'exécutent sur un processeur multicœur i.e. qui permet l'exécution de tâches en parallèle. En ce sens, le but est de créer davantage de variabilité et générer des temps d'exécution représentatifs du pire cas d'exécution de tâches en parallèle. L'algorithme considère en entrée l'application temps réel qui donne sous forme de graphe dirigé acyclique l'ordre d'exécution des tâches composant l'application temps réel. A partir de ce graphe il est possible d'exhiber pour chaque tâche les tâches qu'il est possible d'exécuter en même temps et qui peuvent potentiellement interférer au déploiement du système temps réel. La caractérisation du niveau d'interférence de chaque tâche permet de déterminer un scénario représentatif du plus haut niveau d'interférence. L'application de l'algorithme a permis de valider la nécessité de considérer les contentions dans les approches probabilistes basées mesures. La prise en compte des contentions est plutôt réservée aux analyses d'ordonnancement car trop complexes à modéliser. Cependant, les contentions peuvent être facilement mesurées et ainsi pouvoir soulager les analyses d'ordonnancement particulièrement complexes dans le cas multicœur.

Les paramètres d'entrée de la tâche ont un impact sur le comportement temporel dans le cas de tâches multichemin. Nous avons proposé une représentation spatiale appelée la représentation spectrale qui permet de modéliser le comportement temporel d'une tâche en fonction de ses chemins. Ces considérations ont permis de déterminer les conditions d'exécution des paramètres d'entrée pour assurer l'application de la théorie des valeurs extrêmes notamment par rapport à la stationnarité et la continuité relative des mesures. Enfin, la représentation spectrale a permis de considérer l'application de la méthode d'entropie croisée pour déterminer les chemins à exécuter en accord avec l'application de la théorie des valeurs extrêmes. En se basant sur la connaissance des conditions d'exécution des chemins de la tâche, et non leur comportement temporel, et la probabilité d'exécution des chemins, la méthode de l'entropie croisée modifie les probabilités d'exécution des chemins pour obtenir des mesures afin d'estimer un pire temps d'exécution représentatif du pire cas. L'application de la méthode d'entropie croisée permet d'éliminer les chemins non qui ne tendent pas vers le pire cas et de générer des temps d'exécution plus longs en moyenne. Cependant, la méthode développée reste dépendante de la connaissance exhaustive des conditions d'exécution des chemins de la tâche i.e. déterminer les paramètres d'entrée associés à chaque chemin. Accéder à un tel niveau de connaissance reste coûteux en fonction de la taille de la tâche.

8.8 Perspectives

Les contributions méthodologiques à l'issue des travaux de cette thèse concernent : (i) la mise en place d'un outil de diagnostic de l'application de la théorie des valeurs extrêmes à une suite de mesures DIAGXTRM ; (ii) une méthode pour considérer les contentions matérielles dans l'estimation du pire temps d'exécution probabiliste ; (iii) une méthode pour déterminer les paramètres d'entrée de la tâche étudiée à exécuter pour l'estimation du pire temps d'exécution probabiliste. La robustesse et le cadre d'application réel de DIAGXTRM a été largement étudié dans le Chapitre 6 de cette thèse, mais certains points de l'outil restent à être développés. Dans le cas des deux contributions suivantes, ces dernières n'ont pu être appliquées que dans le cas de simulations du temps d'exécution et il serait intéressant d'approfondir ces deux approches au travers d'un cas d'étude réel.

Diffusion et développement de DIAGXTRM

L'outil DIAGXTRM développé dans cette thèse est disponible sous licence libre à l'adresse `https://forge.onera.fr/projects/diagxtrm2`. Il a été développé dans le langage R, utilisable sous forme de ligne de commande ou sous forme d'interface graphique en exploitant la bibliothèque Shiny. L'outil prend en entrée un fichier texte (.txt, .dat, .csv) contenant la suite de mesures à analyser. Après avoir fixé certains paramètres comme décrits dans ces travaux, l'outil retourne les paramètres estimés du pire temps d'exécution probabiliste ainsi que les niveaux de confiance associés à chaque hypothèse de la théorie des valeurs extrêmes.

Le temps d'exécution de l'application DIAGXTRM est relativement élevé pour des traces de taille de l'ordre de 10^4 mesures. Ceci est dû à la procédure itérative de recherche du meilleur seuil. Une solution serait de développer l'outil en langage C qui optimise davantage les opérations de boucle. Néanmoins, un certain nombre de méthodes relatives aux analyses statistiques, comme l'estimateur du maximum de vraisemblance, et les tests d'hypothèse, ne seraient plus disponibles et seraient donc à implémenter. Une autre solution consisterait à rechercher le meilleur seuil sur une trace de taille plus réduite et de l'appliquer à la trace totale.

L'outil a principalement été développé dans le cadre de l'estimation du pire temps d'exécution probabiliste, mais il est possible de fournir une suite de mesures de paramètres de domaines différents comme la finance ou la climatologie. En ce sens, il serait intéressant d'avoir le retour d'expérience des experts de ces domaines pour améliorer les analyses de DIAGXTRM.

Les travaux d'application de l'outil à divers traces a montré que les tests renvoyaient des niveaux de confiance qui n'étaient pas toujours satisfaisants car l'observation de la suite de mesures ne semblait pas indiquer de comportement contradictoire. L'outil DIAGXTRM est développé sur la base des diagrammes des Figures 5.1 et 5.6 donnent l'ensemble des exigences de la théorie des valeurs extrêmes. Nous avons exhibé dans cette thèse certains tests et algorithmes pour remplir les boites du diagramme. Néanmoins, il est possible de remplacer ces tests et algorithmes par d'autres plus satisfaisants en terme de pertinence et de performance. De plus, l'autre intérêt de DIAGXTRM est l'uti-

lisation de la logique floue pour fusionner les résultats des tests. Ainsi, il est possible de considérer plus qu'un test par hypothèse pour augmenter la confiance dans la fiabilité finale. Cependant, davantage de tests signifie aussi plus de chance de rejeter une hypothèse et donc de ne plus obtenir de résultats exploitables. Enfin, les niveaux de confiance ont été fixés sur la base des seuils α de chaque test. Une manière de diminuer le nombre de résultats insatisfaisants et de gagner en pertinence de résultat est d'apprendre des seuils à partir d'une base de données de traces.

Application de l'ensemble des méthodes développées à un cas d'étude

Une perspective à cette étude consisterait à appliquer l'ensemble des méthodes dans un ordre précis à un cas d'étude réel. L'objectif est d'estimer, pour chaque tâche de l'application temps réel considérée, un pire temps d'exécution probabiliste fiable et robuste mais qui dépend également des exigences du système i.e. quelle probabilité de dépassement du pire temps d'exécution seuil choisir.

Définir un cas d'étude de système temps réel La première étape de cette application est de déterminer un système temps réel : l'application temps réel et l'architecture matérielle. L'application est de préférence parallélisable pour étudier les effets des contentions dans la plateforme. Les tâches de l'application peuvent être multichemin, ou au moins une l'est pour étudier l'application de la méthode d'entropie croisée proposée. Pour paralléliser les exécutions, l'architecture matérielle doit être un processeur multicœur, ou un multiprocesseur.

Mettre en place un outil de mesure du temps d'exécution Pour récupérer les suites de mesure, il faut implémenter un ordonnanceur de tâches qui consiste à exécuter en continue la tâche étudiée et sauvegarder les mesures. Il est également nécessaire de vider les caches avant chaque nouvelle exécution. Dans le cas d'exécutions en parallèle, l'ordonnanceur doit pouvoir mapper les tâches sur les cœurs respectifs.

Pour effectuer les mesures de temps d'exécution ou d'autres paramètres, l'outil de mesure doit être le moins intrusif possible. Une solution consisterait à accéder aux registres correspondants s'ils sont disponibles. Dans le cas d'un système embarqué, il faut faire attention à la taille des suites de mesures dans l'espace mémoire disponible. La taille de l'espace mémoire disponible peut dimensionner le nombre de mesures par suite réalisables. Enfin, pour la mesure du niveau de contention de chaque tâche, il faut implémenter une tâche qui mesure le débit d'accès mémoire de la tâche étudiée sans trop interférer avec.

Appliquer les différentes étapes de l'approche probabiliste basées mesures Pour obtenir des pires temps d'exécution probabilistes robustes, il faut appliquer l'heuristique pour déterminer le pire scénario de contention et l'entropie croisée pour exécuter les chemins qui tendent vers le pire cas des tâches multichemin.

Pour déterminer le pire scénario de contention, il suffit d'implémenter correctement la tâche qui mesure en temps réel le débit d'accès mémoire.

Pour appliquer l'entropie croisée, il faut déterminer les paramètres d'entrée associés aux chemins de la tâche. Certains algorithmes ont été cités dans l'analyse bibliographique et les discussions de cette thèse. Ensuite, il faut déterminer le chemin qui introduit le plus d'interférences dans le système i.e. mesurer le débit d'accès mémoire de chaque chemin.

Il reste alors à effectuer des suites de mesures du temps d'exécution de chaque tâche selon les résultats des procédures précédentes et appliquer DIAGXTRM à toutes les suites de mesures.

Trois cas sont ensuite possibles :

-estimations fiables : il suffit d'inférer un pire temps d'exécution seuil à la probabilité exigée.

-estimation discutables : il faut analyser les traces, et éventuellement effectuer davantage de mesures.

-estimations pas fiables : la théorie des valeurs extrêmes est inapplicable pour ce cas d'étude e.g., la plateforme est temporellement déterministe et/ou isole les tâches des interférences. Dans ce cas, soit on considère le maximum mesuré comme pire temps d'exécution, soit on décide d'appliquer une approche d'estimation statique parce que la plateforme est assez simple. La confiance autour du maximum mesuré est élevée puisque les mesures sont représentatives d'un haut niveau de contention, nous connaissons les chemins des tâches, et la faible ou absence de variabilité indique l'absence de mécanisme matériel imprédictible influent sur le temps d'exécution.

La suite des opérations concerne le cas d'estimations fiables.

Appliquer l'analyse d'ordonnancement Un pire temps d'exécution seuil est inféré à partir de chaque pire temps d'exécution probabiliste. La probabilité des pires temps d'exécution seuil dépend du niveau de sûreté exigé pour chaque tâche mais aussi de son pessimisme. En effet, pour des applications temps réel dur, les probabilités des pire temps d'exécution seuil pourraient être de l'ordre de 10^{-9}. Si peu de mesures ont été réalisées il se peut que les valeurs inférées soient trop pessimistes et pourraient dégrader les performances du systèmes. Il est alors possible soit de réaliser davantage de mesures, si les contraintes du projet le permettent, et éventuellement de réduire le pessimisme ; soit de diminuer la probabilité pour effectuer un compromis.

Enfin, l'analyse d'ordonnancement est effectuée avec les pire temps d'exécution seuil de chaque tâche. Comme nous sommes dans un contexte multiprocesseur il faut considérer le placement des tâches sur chaque cœur. Néanmoins, comme les estimations sont représentatives du plus haut niveau de contentions dues aux exécutions en parallèle, il n'est pas nécessaire de prendre en compte les interférences dans l'analyse d'ordonnancement. Par ailleurs, certaines analyses d'ordonnancement récentes prennent directement en compte les pires temps d'exécution probabilistes [DLG+04, CG09, MCG13, SCG15].

Déployer et valider/certifier le système A la fin de toutes ces étapes il est désormais possible de déployer l'application temps réel sur la cible matérielle. Chaque étape de la procédure permet d'obtenir in fine un pire temps d'exécution probabiliste pour chaque tâche dans lequel le concepteur a confiance par rapport aux critères cités. Une manière de valider/certifier le système pourrait être statistique en faisant tourner plusieurs fois le système temps réel dans des conditions d'essai et vérifier que les tâches respectent leur contrainte temporelle.

Comparaison des résultats obtenus avec d'autres approches

L'avantage des méthodes basées mesures (déterministes et probabilistes) est de pouvoir fournir une estimation provenant de l'exécution réelle de la tâche étudiée. La mesure donne davantage de confiance dans l'estimation, au contraire d'une estimation issue d'un modèle statique d'un ensemble d'objets complexes. Néanmoins, la confiance dans l'observation du pire cas dans le cas de mesures est faible contrairement aux modèles statiques. Que ce soit par une approche basée mesures ou statique, le problème reste celui de l'estimation d'un pire cas car sa connaissance exacte est impossible ou trop coûteuse. Il serait intéressant de pouvoir comparer les estimations issues de l'approche proposée dans cette thèse et celles provenant d'une approche statique. La comparaison fournirait des résultats quantitatifs par rapport à la sûreté et à la précision des estimations, ainsi que des résultats qualitatifs en terme de retour d'information d'une approche pour l'autre. Par exemple, considérer certains mécanismes matériels dans l'approche basée mesures détectés dans l'analyse statique, ou ajuster le modèle statique grâce aux mesures.

8.9 Dissémination scientifique

Les travaux de cette thèse ont donné lieu à différentes publications dans des conférences, des congrès et des workshops.

Communication dans des conférences

K. Berezovskyi, F. Guet, L. Santinelli, K. Bletsas, and E. Tovar. Measurement-based probabilistic timing analysis for graphics processor units. In *Architecture of Computing Systems–ARCS 2016*, pages 223–236. Springer, 2016.

C. Damman, G. Edison, F. Guet, E. Noulard, L. Santinelli, and J. Hugues. Architectural performance analysis of fpga synthesized leon processors. In *Proceedings of the 27th International Symposium on Rapid System Prototyping : Shortening the Path from Specification to Prototype*, pages 33–40. ACM, 2016.

N. Gobillot, F. Guet, D. Doose, C. Grand, C. Lesire, and L. Santinelli. Measurement-based real-time analysis of robotic software architectures. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3306–3311. IEEE, 2016.

F. Guet, L. Santinelli, and J. Morio. Probabilistic analysis of cache memories and cache memories impacts on multi-core embedded systems. In *Industrial Embedded Systems (SIES), 2016 11th IEEE Symposium on*, pages 1–10. IEEE, 2016.

L. Santinelli, F. Guet, and J. Morio. Revising measurement-based probabilistic timing analysis. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017 IEEE*, pages 199–208. IEEE, 2017.

Communication dans des congrès

F. Guet, L. Santinelli, and J. Morio. On the reliability of the probabilistic worst-case execution time estimates. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

Communication dans des workshops

F. Guet, L. Santinelli, and J. Morio. Discussion on the spectral analysis of real-time multi-path tasks. In *9 th Junior Researcher Workshop on Real-Time Computing JRWRTC 2015*, page 5, 2015.

F. Guet, L. Santinelli, and J. Morio. On the representativity of execution time measurements : Studying dependence and multi-mode tasks. In *OASIcs-OpenAccess Series in Informatics*, volume 57. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

Annexe A : Résultats de l'application de diagXtrm

L'outil développé de diagnostic de l'application de l'EVT diagXtrm est appliqué à l'ensemble des traces présentées dans le Chapitre 4. Dans un premier temps, on fournit les statistiques classiques des traces et les résultats sont compilés dans la Table 8. A noter que certaines traces apparaissent deux fois de suite. La première est la trace initiale, et la seconde est une trace modifiée à la lumière de discussions détaillées dans les sections respectives. La table renseigne le nombre de mesures utilisé n, et les statistiques moyennes de chaque trace.

Dans un deuxième temps, diagXtrm est appliqué à l'ensemble des traces. Les résultats relatifs aux paramètres du pWCET sont présentés dans la Table 9. La table donne le nombre de pics de dépassement k, le seuil u, les paramètres estimés de la GPD ξ et α_u. La GPD estimée permet d'inférer un WCET seuil à 10^{-9} $\langle WCET; 10^{-9} \rangle$ et son intervalle de confiance fiable $\left[WCET^{-SE}; WCET^{+SE}\right]$. Le pessimisme du WCET seuil est donné par la fonction $Pess$ d'une probabilité p.

Les résultats concernant l'applicabilité de l'EVT sont dans la Table 10. La Table donne les indices de confiance cl_i ainsi que la conclusion relative à propos de l'estimation du pWCET.

Cette application a permis de mettre en évidence divers cas limites lorsque la conclusion est inapplicable, et des cas pour lesquels la conclusion débouche sur une discussion et un retour sur la trace initiale. En conséquence, nous allons discuter les résultats des traces architecture par architecture.

$\mathcal{T}$	n	minimum	$q(0.25, \mathcal{T})$	médiane	moyenne	$q(0.75, \mathcal{T})$	maximum
trace1	1000	1.72E+07	1.72E+07	1.72E+07	1.72E+07	1.72E+07	1.72E+07
trace2	1000	1.72E+07	1.72E+07	1.72E+07	1.72E+07	1.72E+07	1.72E+07
trace3	1000	1.72E+07	1.72E+07	1.72E+07	1.72E+07	1.72E+07	1.72E+07
trace4	1000	4.63E+04	4.63E+04	4.63E+04	4.63E+04	4.63E+04	4.63E+04
trace5	1000	5.09E+04	5.09E+04	5.09E+04	5.09E+04	5.09E+04	5.11E+04
trace6	1000	4.72E+04	4.72E+04	4.72E+04	4.72E+04	4.72E+04	4.72E+04
trace7	1000	5.18E+04	5.18E+04	5.18E+04	5.19E+04	5.19E+04	5.19E+04
trace8	500	2.34E+03	2.52E+03	2.56E+03	2.61E+03	2.71E+03	3.50E+03
trace9	500	8.48E+03	9.19E+03	1.23E+04	1.34E+04	1.54E+04	2.25E+04
trace10	500	1.70E+04	5.90E+04	5.92E+04	5.92E+04	5.94E+04	7.95E+04
trace10	401	5.83E+04	5.90E+04	5.92E+04	5.92E+04	5.93E+04	6.04E+04
trace11	500	4.14E+03	8.37E+03	8.44E+03	8.47E+03	8.57E+03	1.06E+04
trace12	249	6.86E+07	3.24E+08	5.66E+08	5.78E+08	8.35E+08	1.09E+09
trace13	66	2.08E+03	2.85E+03	3.03E+03	4.68E+03	6.28E+03	1.39E+04
trace14	37	1.19E+04	1.29E+04	1.38E+04	1.45E+04	1.43E+04	3.19E+04
trace15	35	1.19E+04	1.29E+04	1.37E+04	1.36E+04	1.43E+04	1.64E+04
trace16	1000	1.41E+04	1.57E+04	1.61E+04	1.62E+04	1.64E+04	1.70E+05
trace16	1000	1.44E+04	1.56E+04	1.61E+04	1.60E+04	1.64E+04	1.73E+04
trace17	790	2.19E+06	2.27E+06	2.33E+06	4.56E+06	2.42E+06	1.72E+07
trace17	149	1.34E+07	1.37E+07	1.40E+07	1.41E+07	1.43E+07	1.72E+07
trace18	1000	1.03E+04	1.17E+04	1.22E+04	1.24E+04	1.29E+04	1.51E+04
trace19	1000	1.23E+04	2.39E+04	3.24E+04	3.14E+04	3.61E+04	4.74E+04
trace20	1000	1.26E+04	2.34E+04	3.20E+04	3.12E+04	3.58E+04	4.76E+04
trace20	2000	1.22E+04	2.36E+04	3.22E+04	3.15E+04	4.22E+04	4.76E+04
trace21	1000	2.48E+06	2.48E+06	2.48E+06	2.48E+06	2.48E+06	2.48E+06
trace22	1000	2.49E+06	2.49E+06	2.49E+06	2.49E+06	2.49E+06	2.49E+06
trace23	1000	1.37E+05	4.39E+05	5.03E+05	5.06E+05	5.75E+05	8.96E+05
trace24	996	1.00E+00	3.00E+00	7.00E+00	1.07E+01	1.50E+01	8.90E+01
trace25	1000	9.85E+04	9.97E+04	1.00E+05	1.00E+05	1.00E+05	1.02E+05
trace26	1000	1.12E+05	1.14E+05	1.15E+05	1.15E+05	1.15E+05	1.17E+05
trace27	1000	8.87E+04	8.88E+04	8.88E+04	8.88E+04	8.88E+04	8.91E+04
trace28	1000	5.16E+05	5.23E+05	5.24E+05	5.24E+05	5.26E+05	5.33E+05
trace29	1000	3.68E+05	3.68E+05	3.68E+05	3.68E+05	3.68E+05	3.68E+05
trace30	1000	1.36E+09	1.36E+09	1.36E+09	1.36E+09	1.36E+09	1.36E+09
trace31	1000	2.07E+07	2.07E+07	2.07E+07	2.07E+07	2.07E+07	2.07E+07
trace32	1000	4.67E+06	4.67E+06	4.67E+06	4.67E+06	4.67E+06	4.67E+06

TABLE 8 – Statistiques des traces.

$\mathcal{T}$	k	u	ξ	α_u	$\langle WCET; 10^{-9} \rangle$	$WCET^{-SE}$	$WCET^{+SE}$	$Pess\,(10^{-9})$
trace1	650	1.72E+07	-0.23	5360.95	1.72E+07	1.72E+07	1.72E+07	1.99E-02
trace2	439	1.72E+07	-0.16	4159.24	1.72E+07	1.72E+07	1.72E+07	2.55E-02
trace3	165	1.72E+07	-0.13	3056.33	1.72E+07	1.72E+07	1.73E+07	5.17E-02
trace4	NA	NA	NA	NA	NA	NA	NA	NA
trace5	230	5.09E+04	0.03	16.19	5.14E+04	5.11E+04	6.18E+04	5.83E-01
trace6	NA	NA	NA	NA	NA	NA	NA	NA
trace7	825	5.18E+04	-0.07	19.94	5.21E+04	5.20E+04	5.22E+04	2.17E-01
trace8	84	2.76E+03	0.62	50.83	9.84E+06	5.04E+03	6.17E+10	2.81E+05
trace9	97	1.82E+04	0.21	157	6.14E+04	2.25E+04	7.91E+06	1.73E+02
trace10	149	5.93E+04	0.33	210.4	4.63E+05	7.99E+04	1.29E+07	4.83E+02
trace10	155	5.92E+04	-0.08	192.43	6.12E+04	6.04E+04	7.05E+04	1.37E+00
trace11	34	8.72E+03	0.78	53.05	8.52E+07	1.07E+04	7.61E+13	8.01E+05
trace12	NA	NA	NA	NA	NA	NA	NA	NA
trace13	30	3.08E+03	-0.56	6400.03	1.46E+04	1.39E+04	4.76E+04	4.84E+00
trace14	25	1.33E+04	0.54	1017.62	1.06E+08	3.28E+04	4.97E+13	3.33E+05
trace15	25	1.30E+04	-0.36	1436.18	1.70E+04	1.64E+04	1.44E+05	3.93E+00
trace16	25	1.70E+04	2.09	77.42	1.01E+17	5.46E+05	8.28E+28	5.96E+13
trace16	90	1.66E+04	-0.27	252.05	1.75E+04	1.73E+04	2.17E+04	1.34E+00
trace17	31	1.44E+07	0.23	331443.22	9.68E+07	1.73E+07	3.23E+11	4.61E+02
trace17	31	1.44E+07	0.23	331443.22	1.37E+08	1.73E+07	3.90E+12	6.92E+02
trace18	860	1.15E+04	-0.39	1503.51	1.54E+04	1.51E+04	1.62E+04	1.63E+00
trace19	74	4.60E+04	-0.44	645.65	4.75E+04	4.74E+04	4.90E+04	1.28E-01
trace20	25	4.67E+04	0.01	278.08	5.17E+04	4.76E+04	1.52E+09	8.68E+00
trace20	146	4.61E+04	-0.41	669.65	4.77E+04	4.76E+04	4.80E+04	2.97E-01
trace21	638	2.48E+06	-0.29	7.67	2.48E+06	2.48E+06	2.48E+06	5.63E-05
trace22	53	2.49E+06	0.05	45.77	2.49E+06	2.49E+06	2.62E+06	3.98E-02
trace23	104	6.32E+05	-0.01	41921.18	1.36E+06	8.97E+05	1.30E+07	5.18E+01
trace24	262	1.43E+01	0	10.18	2.13E+02	8.93E+01	2.09E+03	1.39E+02
trace25	444	1.00E+05	-0.23	467.85	1.02E+05	1.02E+05	1.03E+05	2.90E-01
trace26	601	1.14E+05	-0.3	862.7	1.17E+05	1.17E+05	1.18E+05	2.26E-01
trace27	NA	NA	NA	NA	NA	NA	NA	NA
trace28	333	5.25E+05	-0.16	1703.06	5.35E+05	5.33E+05	5.48E+05	4.24E-01
trace29	NA	NA	NA	NA	NA	NA	NA	NA
trace30	459	1.36E+09	-0.27	253764.59	1.36E+09	1.36E+09	1.36E+09	6.67E-03
trace31	NA	NA	NA	NA	NA	NA	NA	NA
trace32	696	4.67E+06	-0.53	45.5	4.67E+06	4.67E+06	4.67E+06	5.47E-06

TABLE 9 – Résultats de la modélisation GPD issue de DIAGXTRM.

$\mathcal{T}$	$cl_{1.1}$	$cl_{2.1}$	$cl_{1.2}$	$cl_{2.2}$	cl_3	cl_4	$cl_{reliability}$	conclusion
trace1	4	3.92	4	4	4	3.67	3.93	fiable
trace2	4	3.58	4	4	4	3.67	3.88	fiable
trace3	1	1.08	4	1	4	3.00	2.35	fiable
trace4	4	0.00	NA	NA	NA	NA	0.00	inapplicable
trace5	4	3.92	4	4	4	0.00	0.00	manque mesures
trace6	4	1.67	NA	NA	NA	NA	0.00	inapplicable
trace7	4	3.50	4	4	4	3.67	3.86	fiable
trace8	4	3.50	4	4	4	2.67	3.70	fiable
trace9	4	1.67	4	1	4	0.00	0.00	manque mesures
trace10	4	0.00	1	2	4	2.33	0.00	à discuter
trace10	4	3.67	3	3	4	3.67	3.56	fiable
trace11	4	3.00	4	4	4	0.00	0.00	manque mesures
trace12	0	0.00	NA	NA	NA	NA	0.00	inapplicable
trace13	4	4.00	4	2	1	2.67	2.95	fiable
trace14	4	1.00	1	0	4	0.00	0.00	manque mesures
trace15	0	2.67	4	0	4	3.33	0.00	manque mesures
trace16	3	4.00	4	0	4	2.33	0.00	à discuter
trace16	4	0.00	4	1	4	3.33	0.00	fiable
trace17	0	0.00	4	3	4	0.00	0.00	à discuter
trace17	4	3.67	4	3	4	4.00	3.78	fiable
trace18	4	3.08	4	4	4	4.00	3.85	fiable
trace19	4	2.83	4	3	3	2.33	3.19	fiable
trace20	4	4.00	4	0	4	0.00	0.00	à discuter
trace20	4	2.59	4	4	2	3.00	3.27	fiable
trace21	4	3.58	4	4	1	3.00	3.26	fiable
trace22	2	3.92	4	4	4	0.00	0.00	manque mesures
trace23	3	3.83	4	4	4	0.00	0.00	manque mesures
trace24	4	4.00	4	4	4	0.00	0.00	manque mesures
trace25	4	4.00	4	4	4	3.67	3.94	fiable
trace26	4	0.00	4	4	4	3.33	0.00	à discuter
trace27	4	0.00	NA	NA	NA	NA	0.00	inapplicable
trace28	4	4.00	4	3	4	3.33	3.72	fiable
trace29	0	0.00	NA	NA	NA	NA	0.00	inapplicable
trace30	4	4.00	4	4	4	3.67	3.95	fiable
trace31	4	NA	NA	NA	NA	NA	0.00	inapplicable
trace32	1	4.00	4	4	1	2.67	2.78	fiable

TABLE 10 – Evaluation de la fiabilité des pWCETs issus de DIAGXTRM.

Bibliographie

[AB09] S. Altmeyer and C. Burguiere. A new notion of useful cache block to improve the bounds of cache-related preemption delay. In *Real-Time Systems, 2009. ECRTS'09. 21st Euromicro Conference on*, pages 109–118. IEEE, 2009.

[ABS13] A. Al Badawi and A. Shatnawi. Static scheduling of directed acyclic data flow graphs onto multiprocessors using particle swarm optimization. *Computers & Operations Research*, 40(10) :2322–2328, 2013.

[AD14] S. Altmeyer and R. I. Davis. On the correctness, optimality and precision of static probabilistic timing analysis. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6. IEEE, 2014.

[AdCPC15] J. Abella, J. del Castillo, M. Padilla, and F. J. Cazorla. Extreme value theory in computer sciences : The case of embedded safety-critical systems. In *6th International Conference on Risk Analysis ICRA*, 2015.

[AGW10] J. Andersson, J. Gaisler, and R. Weigand. Next generation multipurpose microprocessor. In *Int. Conf. on Data Systems in Aerospace (DASIA), Hungary*, 2010.

[AQW+14] J. Abella, E Quiñones, F. Wartel, T. Vardanega, and F. J. Cazorla. Heart of gold : Making the improbable happen to increase confidence in mbpta. In *2014 26th Euromicro Conference on Real-Time Systems*, pages 255–265. IEEE, 2014.

[AvL85] E. H. L. Aarts and P. J. M. van Laarhoven. Statistical cooling : A general approach to combinatorial optimization problems. 1985.

[BCP03] G. Bernat, A. Colin, and S. Petters. pWCET : A tool for probabilistic worst-case execution time analysis of real-time systems. Technical report, 2003.

[BCPV96] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress : A notion of fairness in resource allocation. *Algorithmica*, 15(6) :600–625, 1996.

[BDG+00] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors.

The international journal of high performance computing applications, 14(3) :189–204, 2000.

[BE00] Alan Burns and Stewart Edgar. Predicting computation time for advanced processor architectures. In *12th Euromicro Conference on Real-Time Systems ECRTS, Proceedings*, pages 89–96, 2000.

[Ber16] K. Berezovskyi. Timing analysis of general-purpose graphics processing uni ts for real-time systems : Models and analyses. 2016.

[BGS+16] K. Berezovskyi, F. Guet, L. Santinelli, K. Bletsas, and E. Tovar. Measurement-based probabilistic timing analysis for graphics processor units. In *Architecture of Computing Systems–ARCS 2016*, pages 223–236. Springer, 2016.

[BM11] A. Bonache and K. Moris. Chaos dans les ventes de biens à la mode et implication pour le contrôle de gestion. Post-print, HAL, 2011.

[BMSO+96] J. V. Busquets-Mataix, J. J. Serrano, R. Ors, P. Gil, and A. Wellings. Adding instruction cache effect to schedulability analysis of preemptive real-time systems. In *Real-Time Technology and Applications Symposium, 1996. Proceedings., 1996 IEEE*, pages 204–212. IEEE, 1996.

[BSBT14] K. Berezovskyi, L. Santinelli, K. Bletsas, and E. Tovar. WCET measurement-based and extreme value theory characterisation of CUDA kernels. In *22nd International Conference on Real-Time Networks and Systems, RTNS*, 2014.

[BSDL96] W. A. Brock, J. A. Scheinkman, W. D. Dechert, and B. LeBaron. A Test for Independence based on the Correlation Dimension. *Econometric Reviews*, 15(3) :197–235, 1996.

[Buc05] J. J. Buckley. Fuzzy statistics : hypothesis testing. *Soft Comput.*, 9(7) :512–518, 2005.

[BZTK11] S. Bunte, M. Zolda, M. Tautschnig, and R. Kirner. Improving the confidence in measurement-based timing analysis. In *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2011 14th IEEE International Symposium on*, pages 144–151. IEEE, 2011.

[CC77] P. Cousot and R. Cousot. Abstract interpretation : a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 238–252. ACM, 1977.

[CF96] S. Csorgo and J. J. Faraway. The exact and asymptotic distributions of cramér-von mises statistics. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 221–234, 1996.

[CG09] L. Cucu-Grosjean. Probabilistic real-time schedulability analysis : from uniprocessor to multiprocessor when the execution times are uncertain. *RR-INRIA, May*, 2009.

[CGSH+12] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzeti, E. Quinones, and F. J. Cazorla. Measurement-

Based Probabilistic Timing Analysis for Multi-path Programs. In *the 24th Euromicro Conference on Real-Time Systems*, 2012.

[Col01] S. Coles. *An introduction to statistical modeling of extreme values*. Springer Series in Statistics. Springer-Verlag, London, 2001.

[CPRS03] A. Colin, I. Puaut, C. Rochange, and P. Sainrat. Calcul de majorants de pire temps d'exécution : état de l'art. *Technique et Science Informatiques*, 22(5) :651–677, 2003.

[CSHB16] C. Chen, L. Santinelli, J. Hugues, and G. Beltrame. Static probabilistic timing analysis in presence of faults. In *Industrial Embedded Systems (SIES), 2016 11th IEEE Symposium on*, pages 1–10. IEEE, 2016.

[CVQA13] F. J. Cazorla, T. Vardanega, E. Quiñones, and J. Abella. Upper-bounding program execution time with extreme value theory. In *OASIcs-OpenAccess Series in Informatics*, volume 30. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[DB11] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4) :35, 2011.

[DBKMR05] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1) :19–67, 2005.

[Den06] L.-Y. Deng. The cross-entropy method : a unified approach to combinatorial optimization, monte-carlo simulation, and machine learning, 2006.

[DLG+04] J. L. Diaz, J. M. Lopez, M. Garcia, A. M. Campos, K. Kim, and L. L. Bello. Pessimism in the stochastic analysis of real-time systems : Concept and applications. In *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*, pages 197–207. IEEE, 2004.

[DVA+14] R. I. Davis, T. Vardanega, J. Andersson, F. Vatrinet, M. Pearce, I. Broster, M. Azkarate-Askasua, F. Wartel, L. Cucu-Grosjean, M. Patte, G. Farrall, and F. J. Cazrola. Proxima : A probabilistic approach to timing behaviour of mixed-criticality systems. *Ada User Journal*, 35(2), June 2014.

[EB01] S. Edgar and A. Burns. Statistical Analysis of WCET for Scheduling. In *Real-Time Systems Symposium RTSS*, pages 215–224. IEEE Computer Society, 2001.

[EES+03] J. Engblom, A. Ermedahl, M. Sjödin, J. Gustafsson, and H. Hansson. Worst-case execution-time analysis for embedded real-time systems. *International Journal on Software Tools for Technology Transfer (STTT)*, 4(4) :437–455, 2003.

[EKM97] P. Embrechts, C. Klüppelberg, and T. Mikosch. *Modelling extremal events for insurance and finance*. Applications of mathematics. Springer, Berlin, Heidelberg, New York, 1997.

[ENNT15] A. Esper, G. Nelissen, V. Nelis, and E. Tovar. How realistic is the mixed-criticality real-time system model ? In *the 23st International Conference on Real-Time and Networked Systems (RTNS'15)*, 2015.

[FDD09] P.-M. Fournier, M. Desnoyers, and M. R. Dagenais. Combined Tracing of

the Kernel and Applications with LTTng. In *Proceedings of the 2009 Linux Symposium*, July 2009.

[FS02] C.A.T. Ferro and J. Segers. Automatic Declustering of Extreme Values Via an Estimator for the Extremal Index. 2002.

[Gar02] M. Garrido. *Modelling of rare events and estimation of extreme quantiles, model selecting methods for distribution tails.* PhD thesis, Université Joseph-Fourier - Grenoble I, June 2002.

[GB10] D. Griffin and A. Burns. Realism in statistical analysis of worst case execution times. In *OASIcs-OpenAccess Series in Informatics*, volume 15. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

[GGD+16] N. Gobillot, F. Guet, D. Doose, C. Grand, C. Lesire, and L. Santinelli. Measurement-based real-time analysis of robotic software architectures. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3306–3311. IEEE, 2016.

[GL13] F. Glover and M. Laguna. Tabu search∗. In *Handbook of Combinatorial Optimization*, pages 3261–3362. Springer, 2013.

[GM08] J. C. F. García and J. J. S. Méndez. A fuzzy logic approach to test statistical hypothesis on means. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, 4th International Conference on Intelligent Computing, ICIC, Proceedings*, pages 316–325, 2008.

[Gob16] N. Gobillot. *Validation d'architectures temps-réel pour la robotique autonome*. PhD thesis, Toulouse, ISAE, 2016.

[GSM16a] F. Guet, L. Santinelli, and J. Morio. On the reliability of the probabilistic worst-case execution time estimates. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[GSM16b] F. Guet, L. Santinelli, and J. Morio. Probabilistic analysis of cache memories and cache memories impacts on multi-core embedded systems. In *Industrial Embedded Systems (SIES), 2016 11th IEEE Symposium on*, pages 1–10. IEEE, 2016.

[HHM09] J. Hansen, S. Hissam, and G. A. Moreno. Statistical-Based WCET Estimation and Validation. In *9th International Workshop on Worst-Case Execution Time Analysis WCET*, pages 1–11, 2009.

[Hil02] M. Hillary. You can't control what you can't measure, or why it's close to impossible to guarantee real-time software performance on a cpu with on-chip cache. Technical report, Applied Microsystems Corp, 2002.

[Hir96] H. Hirose. Maximum likelihood estimation in the 3-parameter weibull distribution. a look through the generalized extreme-value distribution. *IEEE transactions on dielectrics and electrical insulation*, 3(1) :43–55, 1996.

[HKA+01] C. J. Hughes, P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan. Variability in the execution of multimedia applications and implications for architecture. In *Computer Architecture, 2001. Proceedings. 28th Annual International Symposium on*, pages 254–265. IEEE, 2001.

[HW87] J. R. M. Hosking and J. R. Wallis. Parameter and quantile estimation for the generalized pareto distribution. *Technometrics*, 29(3) :339–349, 1987.

[JKA+14] J. Jalle, L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla. Bus designs for time-probabilistic multicore processors. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2014.

[KAQC13a] L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla. A cache design for probabilistically analysable real-time systems. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 513–518. EDA Consortium, 2013.

[KAQC13b] L. Kosmidis, J. Abella, E. Quinones, and F. J. Cazorla. Multi-level unified caches for probabilistically time analysable real-time systems. In *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, pages 360–371. IEEE, 2013.

[KPN02] R. W. Katz, M. B. Parlange, and P. Naveau. Statistics of extremes in hydrology. *Advances in Water Resources*, 25 :1287–1304, August-December 2002.

[KPSS92] D. Kwiatkowski, P. C. B. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root : How sure are we that economic time series have a unit root ? *Journal of Econometrics*, 54(1-3) :159–178, 00 1992.

[KQA+14] L. Kosmidis, E. Quiñones, J. Abella, T. Vardanega, I. Broster, and F. J. Cazorla. Measurement-based probabilistic timing analysis and its impact on processor architecture. In *Digital System Design (DSD), 2014 17th Euromicro Conference on*, pages 401–410. IEEE, 2014.

[KTOY00] Y. Kato, M. Takahashi, R. Ohtsuki, and S. Yamaguchi. A proposal of fuzzy test for statistical hypothesis. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 4, pages 2929–2934 vol.4, 2000.

[Kul97] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[KVAF+13] L. Kosmidis, T. Vardanega, J. Abella Ferrer, E. Quiñones Moreno, and F. J. Cazorla Almeida. Applying measurement-based probabilistic timing analysis to buffer resources. In *13th International Workshop on Worst-Case Execution Time Analysis : Paris, France, July 9 2013*, pages 97–108, 2013.

[KWH95] L. Ko, D. B. Whalley, and M. G. Harmon. Supporting user-friendly analysis of timing constraints. In *ACM SIGPLAN Notices*, volume 30, pages 99–107. ACM, 1995.

[Lai04] F. Laio. Cramer-von Mises and Anderson-Darling goodness of fit tests for extreme value distributions with unknown parameters. *Water Resources Research*, 40(W09308) :1–10, 2004.

[LB16] S. Law and I. Bate. Achieving appropriate test coverage for reliable measurement-based timing analysis. In *Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on*, pages 189–199. IEEE, 2016.

[LBN13a] M. Liu, M. Behnam, and T. Nolte. Applying the peak over thresholds

method on worst-case response time analysis of complex real-time systems. In *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA*, pages 22–31, 2013.

[LBN13b] M. Liu, M. Behnam, and T. Nolte. An evt-based worst-case response time analysis of complex real-time systems. In *8th IEEE International Symposium on Industrial Embedded Systems, SIES*, pages 249–258, 2013.

[Lea91] M. R. Leadbetter. On a basis for peaks over threshold modeling. *Statistics & Probability Letters*, 12(4) :357–362, 1991.

[LGAD15] B. Lesage, D. Griffin, S. Altmeyer, and R. I. Davis. Static probabilistic timing analysis for multi-path programs. In *Real-Time Systems Symposium, 2015 IEEE*, pages 361–372. IEEE, 2015.

[LHT00] M. Lindgren, H. Hansson, and H. Thane. Using measurements to derive the worst-case execution time. In *Real-Time Computing Systems and Applications, 2000. Proceedings. Seventh International Conference on*, pages 15–22. IEEE, 2000.

[LL73] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1) :46–61, 1973.

[LL85] C. C. Lee and D.-T. Lee. A simple on-line bin-packing algorithm. *Journal of the ACM (JACM)*, 32(3) :562–572, 1985.

[LLR78] M. R. Leadbetter, G. Lindgren, and H. Rootzén. Conditions for the convergence in distribution of maxima of stationary normal processes. *Stochastic Processes and their Applications*, 8(2) :131–139, 1978.

[LNBC12] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean. A statistical response-time analysis of real-time embedded systems. In *Proceedings of the 33rd IEEE Real-Time Systems Symposium, RTSS*, pages 351–362, 2012.

[LNBCG11] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean. A Trace-Based Statistical Worst-Case Execution Time Analysis of Component-Based Real-Time Embedded Systems. In *16th IEEE International Conference on Emerging Technology and Factory Automation ETFA, WiP session*, September 2011.

[LW82] J. Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance evaluation*, 2(4) :237–250, 1982.

[M.13] Dorin M. *Analyse probabiliste des systemes temps réel.* PhD thesis, Université de Lorraine, 2013.

[Mac03] D. J. C. MacKay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[MAL13] *WCET project/ Benchmarks*, 2013.

[MB15] J. Morio and M. Balesdent. *Estimation of Rare Event Probabilities in Complex Aerospace and Other Systems : A Practical Approach.* Woodhead Publishing, 2015.

[MBBF16] A. Mangean, J.-L. Béchennec, M. Briday, and S. Faucou. Best : a binary executable slicing tool. In *OASIcs-OpenAccess Series in Informatics*,

volume 55. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[MCG13] D. Maxim and L. Cucu-Grosjean. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, pages 224–235. IEEE, 2013.

[MFB+17] E. Mezzetti, M. Fernandez, A. Bardizbanyan, I. Agirre, J. Abella, T. Vardanega, and F. J. Cazorla. Epc enacted : Integration in an industrial toolbox and use against a railway application. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017 IEEE*, pages 163–174. IEEE, 2017.

[MGP16] S. M'Sirdi, W. Godard, and M. Pantel. A multi-core interference-aware schedulability test for ima systems, as a guide for sw/hw integration. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[Mit98] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[MNS] A. Melani, E. Noulard, and L. Santinelli. Learning from Probabilities : Dependences within Real-Time Systems. In *2013 ETFA*.

[MS96] R. Manuca and R. Savit. Stationarity and nonstationarity in time series analysis. *Phys. D*, 99(2-3) :134–161, December 1996.

[MTS11] J. Mars, L. Tang, and M. L. Soffa. Directly characterizing cross core interference through contention synthesis. In *Proceedings of the 6th International Conference on High Performance and Embedded Architectures and Compilers*, pages 167–176. ACM, 2011.

[Mul06] A. Muller. *Asymptotic behavior of extreme rainfall distribution in France*. Thèses, Université Montpellier II - Sciences et Techniques du Languedoc, November 2006.

[MZV+15] E. Mezzetti, M. Ziccardi, T. Vardanega, J. Abella, E. Quiñones, and F. J. Cazorla. Randomized caches can be pretty useful to hard real-time systems. *Leibniz Transactions on Embedded Systems*, 2(1) :01–1, 2015.

[NYP16] V. Nélis, P. M. Yomsi, and L. M. Pinho. The variability of application execution times on a multi-core platform. In *16th International Workshop on Worst-Case Execution Time Analysis*, 2016.

[PAH+15] M. Panic, J. Abella, C. Hernandez, E. Quiñones, T. Ungerer, and F. J. Cazorla. Enabling tdma arbitration in the context of mbpta. In *Digital System Design (DSD), 2015 Euromicro Conference on*, pages 462–469. IEEE, 2015.

[PH96] D. A. Patterson and J. L. Hennessy. Computer architecture a quantitative approach. *Morgan Kauffman*, 1996.

[PK89] P. Puschner and C. Koza. Calculating the maximum execution time of real-time programs. *Real-Time Systems*, 1(2) :159–176, 1989.

[PM08] M. Piera-Martinez. *Modélisation des comportements extrêmes en ingénierie*. Theses, Université Paris Sud - Paris XI, September 2008.

[PMB13] V.-A. Paun, B. Monsuez, and P. Baufreton. On the Determinism of Multicore Processors. In Christine Choppy and Jun Sun, editors, *1st French*

Singaporean Workshop on Formal Methods and Applications FSFMA, volume 31 of *OpenAccess Series in Informatics OASIcs*, pages 32–46, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[PMN+16] Quentin Perret, Pascal Maurere, Eric Noulard, Claire Pagetti, Pascal Sainrat, and Benoit Triquet. Temporal isolation of hard real-time applications on many-core processors. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–11. IEEE, 2016.

[PN98] P. Puschner and Roman. Nossal. Testing the results of static worst-case execution-time analysis. In *Real-Time Systems Symposium, 1998. Proceedings. The 19th IEEE*, pages 134–143. IEEE, 1998.

[Pot12] F. Pothon. Do-178c/ed-12c versus do-178b/ed-12b : Changes and improvements. *Frederic Pothon*, 2012.

[Pus02] P. Puschner. Worst-case execution-time analysis a non-problem ?–towards new software and hardware architectures, proc. 2nd euromicro international workshop on wcet analysis york yo10 5dd, united kingdom. In *Department of Computer Science, University of York*. Citeseer, 2002.

[RC13] C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[Res97] S. I. Resnick. Discussion of the danish data on large fire insurance losses. *Astin Bulletin*, 27(01) :139–151, 1997.

[RMMA15] H. Rihani, M. Moy, C. Maiza, and S. Altmeyer. Wcet analysis in shared resources real-time systems with tdma buses. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, pages 183–192. ACM, 2015.

[RWT+06] J. Reineke, B. Wachter, S. Thesing, R. Wilhelm, I. Polian, J. Eisinger, and B. Becker. A definition and classification of timing anomalies. In *OASIcs-OpenAccess Series in Informatics*, volume 4. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.

[SAA+15] M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann, et al. T-crest : Time-predictable multi-core architecture for embedded systems. *Journal of Systems Architecture*, 61(9) :449–471, 2015.

[SCG15] L. Santinelli and L. Cucu-Grosjean. A probabilistic calculus for probabilistic real-time systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(3) :52, 2015.

[SGM17] L. Santinelli, F. Guet, and J. Morio. Revising measurement-based probabilistic timing analysis. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2017 IEEE*, pages 199–208. IEEE, 2017.

[SKA+14] M. Slijepcevic, L. Kosmidis, J. Abella, E. Quiñones, and F. J. Cazorla. Time-analysable non-partitioned shared caches for real-time multicore systems. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.

[SM12] C. Scarrott and A. MacDonald. A review of extreme value threshold es-

timation and uncertainty quantification. *REVSTAT–Statistical Journal*, 10(1) :33–60, 2012.

[SMDJ14] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart. On the Sustainability of the Extreme Value Theory for WCET Estimation. In *14th International Workshop on Worst-Case Execution Time Analysis*, pages 21–30, 2014.

[Smi85] R. L. Smith. Maximum Likelihood Estimation in a Class of Nonregular Cases. *Biometrika*, 72(1) :pp. 67–90, 1985.

[SPD+14] L. Santinelli, W. Puffitsch, A. Dumerat, F. Boniol, C. Pagetti, and V. Jegu. A grouping approach to task scheduling with functional and non-functional requirements. In *Embedded real-time software and systems (ERTS^2 2014)*, 2014.

[SRL90] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols : An approach to real-time synchronization. *IEEE Transactions on computers*, 39(9) :1175–1185, 1990.

[Ste77] M. A. Stephens. Goodness-Of-Fit for the Extreme Value Distribution. *Biometrika*, 64(3) :583–8, 1977.

[TAC16] *Timing analysis on code-level*, 2016. `http//www.tacle.eu/index.php/activities/taclebench`.

[TE15] G. Tzavelas and P. Economou. Extreme value distributions for biased samples. *Probability in the Engineering and Informational Sciences*, 29(2) :277–290, 2015.

[TMS11] L. Tang, J. Mars, and M. L. Soffa. Contentiousness vs. sensitivity : improving contention aware runtime systems on multicore architectures. In *Proceedings of the 1st International Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era*, pages 12–21. ACM, 2011.

[TTT06] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1) :28–42, 2006.

[Ves07] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the 28th IEEE Real-Time Systems Symposium RTSS*, pages 239–243, 2007.

[WEE+08] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. B. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. P. Puschner, J. Staschulat, and P. Stenström. The worst-case execution-time problem - overview of methods and survey of tools. *ACM Trans. Embedded Comput. Syst.*, 7(3), 2008.

[WKL+13] F. Wartel, L. Kosmidis, C. Lo, B. Triquet, E. Quinones, J. Abella, A. Gogonel, A. Baldovin, E. Mezzetti, L. Cucu, et al. Measurement-based probabilistic timing analysis : Lessons from an integrated-modular avionics case study. In *Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on*, pages 241–248. IEEE, 2013.

[WMMR05] N. Williams, B. Marre, P. Mouy, and M. Roger. Pathcrawler : Automatic

generation of path tests by combining static and dynamic analysis. In *European Dependable Computing Conference*, pages 281–292. Springer, 2005.

[WR09] N. Williams and M. Roger. Test generation strategies to measure worst-case execution time. In *Automation of Software Test, 2009. AST'09. ICSE Workshop on*, pages 88–96. IEEE, 2009.

[WTM13] V. M. Weaver, D. Terpstra, and S. Moore. Non-determinism and overcount on modern hardware performance counter implementations. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 215–224. IEEE, 2013.

[Zho10] S. Zhou. An efficient simulation algorithm for cache of random replacement policy. In *IFIP International Conference on Network and Parallel Computing*, pages 144–154. Springer, 2010.

[ZMV+15] M. Ziccardi, E. Mezzetti, T. Vardanega, J. Abella, and F. J. Cazorla. Epc : Extended path coverage for measurement-based probabilistic timing analysis. In *Real-Time Systems Symposium, 2015 IEEE*, pages 338–349. IEEE, 2015.

Résumé

Dans les systèmes informatiques temps réel, les tâches logicielles sont contraintes par le temps. Pour garantir la sûreté du système critique controlé par le système temps réel, il est primordial d'estimer de manière sûre le pire temps d'exécution de chaque tâche. Les performances des processeurs actuels du commerce permettent de réduire en moyenne le temps d'exécution des tâches, mais la complexité des composants d'optimisation de la plateforme rendent difficile l'estimation du pire temps d'exécution.

Il existe différentes approches d'estimation du pire temps d'exécution, souvent ségréguées et difficilement généralisables ou au prix de modèles coûteux. Les approches probabilistes basées mesures existantes sont vues comme étant rapides et simples à mettre en œuvre, mais souffrent d'un manque de systématisme et de confiance dans les estimations qu'elles fournissent.

Les travaux de cette thèse étudient les conditions d'application de la théorie des valeurs extrêmes à une suite de mesures de temps d'exécution pour l'estimation du pire temps d'exécution probabiliste, et ont été implémentées dans l'outil diagXtrm. Les capacités et les limites de l'outil ont été étudiées grâce à diverses suites de mesures issues de systèmes temps réel différents. Enfin, des méthodes sont proposées pour déterminer les conditions de mesure propices à l'application de la théorie des valeurs extrêmes et donner davantage de confiance dans les estimations.

Mots clés : pire temps d'exécution, théorie des valeurs extrêmes, temps réel, série temporelle, multicœur

Abstract

Software tasks are time constrained in real time computing systems. To ensure the safety of the critical systems that embeds the real time system, it is of paramount importance to safely estimate the worst-case execution time of each task. Modern commercial processors optimisation components enable to reduce in average the task execution time at the cost of a hard to determine task worst-case execution time.

Many approaches for executing a task worst-case execution time exist but are usually segregated and hardly scalable, or by building very complex models. Measurement-based probabilistic timing analysis approaches are said to be easy and fast, but they suffer from a lack of systematism and confidence in their estimates.

This thesis studies the applicability of the extreme value theory to a sequence of execution time measurements for the estimation of the probabilistic worst-case execution time, leading to the development of the diagXtrm tool. Thanks to a large panel of sequences of measurements from different real time systems, capabilities and limits of the tool are enlightened. Finally, a couple of methods are provided for determining measurements conditions that foster the application of the theory and raise more confidence in the estimates.

Keywords : worst-case execution time, extreme value theory, real time, time series, multicore

ONERA

THE FRENCH AEROSPACE LAB