Estimation de la consommation des systèmes logiciels sur des architectures multi-coeurs

Maxime Colmant, Romain Rouvoy, Lionel Seinturier

Compas'2015 : Parallélisme/ Architecture / Système
Lille, France, du 30 juin au 3 juillet 2015

Estimation de la consommation des systèmes logiciels sur des architectures multi-cœurs *

Maxime Colmant[1,2], Romain Rouvoy[2], Lionel Seinturier[2]
maxime.colmant@inria.fr, romain.rouvoy@inria.fr,
lionel.seinturier@inria.fr

[1] ADEME, France
[2] Université de Lille / Inria, France

Résumé
Les TIC représentent une part importante dans les émissions mondiales de CO_2. En effet, les dernières études montrent qu'elles y contribuent à hauteur de 2%. Leur efficience énergétique est donc un enjeu majeur pour notre société. Dans cet article, nous décrivons une solution capable d'estimer précisément la consommation énergétique des applications s'exécutant sur des processeurs multi-cœurs. Un modèle énergétique, indépendant des applications suivies, est appris automatiquement selon un processus décrit en détail dans cet article. Notre wattmètre logiciel, POWERAPI, basé sur un modèle d'acteurs, utilise ensuite ce modèle pour estimer la consommation des logiciels. À partir de métriques collectées en temps réel, POWERAPI permet d'obtenir des estimations énergétiques précises, sans imposer d'investissements matériels importants. Les expériences présentées dans cet article prouvent que POWERAPI est une solution fiable et non invasive.

Mots-clés : profils énergétiques, estimations en temps réel, wattmètre logiciel

1. Introduction

De nos jours, les *Technologies de l'Information et de la Communication* (TIC) (*e.g.*, ordinateurs, smartphones) ont une part non négligeable sur les rejets de dioxyde de carbone. Leur empreinte carbone a été estimée à 0.83 $GtCO_2$ en 2007 et est estimée à 1.43 $GtCO_2$ pour 2020, ce qui représentera 6% de nos émissions globales par année [16]. Leur efficience énergétique est donc un enjeu crucial pour les années à venir. L'estimation de la consommation énergétique des processus est la pierre angulaire dans la construction de systèmes plus efficients. Les processeurs modernes sont de plus en plus complexes et très peu de solutions sont proposées pour estimer leurs consommations énergétiques. Quelques constructeurs vont cependant dans cette direction : Intel, par exemple, développe la solution *running average power limit* (RAPL) [5, 19] mais elle reste limitée à certaines architectures. Il reste donc beaucoup de travail à accomplir pour proposer une solution précise et agnostique vis-à-vis du processeur.
Dans cet article, nous présentons notre solution modulable, POWERAPI [2], pour estimer la consom-

*. Article publié : Colmant (M.), Kurpicz (M.), Huertas (L.), Rouvoy (R.), Felber (P.) et Sobe (A.). – Process-level Power Estimation in VM-based Systems. – In *European Conference on Computer Systems (EuroSys)*, Bordeaux, France, avril 2015.

2. Disponible sous licence AGPL) : http://powerapi.org

mation énergétique des processus. Différents modèles énergétiques (*e.g.*, RAPL, ou n'importe quel wattmètre interfaçable), peuvent être intégrés à POWERAPI. Ici, nous proposons un modèle énergétique à grain fin ciblant les processeurs modernes utilisant différentes technologies pour améliorer leurs performances ou réduire leurs consommations, comme le *multi-cœur*, l'*Hyper-Threading*, le *Dynamic Voltage/Frequency Scaling* et le *TurboBoost*.
Notre solution, POWERAPI, a été développée en SCALA et est capable d'inférer le modèle énergétique d'un processeur par le biais d'une phase d'apprentissage (*c.f.*, figure 2). Ce modèle supporte les diverses technologies présentées précédemment et est ensuite utilisé sur le processeur cible pour produire des estimations en temps réel. Plusieurs expériences ont été effectuées sur diverses architectures pour prouver la fiabilité et la précision de notre solution en nous comparant aux valeurs mesurées par un wattmètre physique externe. Les résultats nous montrent que POWERAPI produit des estimations précises grâce au modèle énergétique décrit dans cet article.
De nos jours, la principale source observée de consommation énergétique est le processeur, représentant pour un peu plus d'un tiers la consommation énergétique global d'un ordinateur. C'est pour cette raison que, comme d'autres études [1, 9, 19] le font, notre modèle énergétique se concentre plus particulièrement sur le processeur. En effet, comme le montre [7], la modélisation de la consommation du disque dur nécessite de nouvelles études. C'est pour cela que nous avons sélectionné, dans un premier temps, des benchmarks de telle sorte que le surcoût de consommation induit par le disque dur ou la mémoire soit négligeable. D'autres études sur les data centers [12] ont montré que le réseau, dans le cas d'une connexion Ethernet, n'impactait que très peu la consommation énergétique d'un serveur. Nous limitons donc ainsi les interférences liées aux autres composants lorsque nous récoltons les valeurs provenant du wattmètre externe. Sur la base de ce premier travail, d'autres études approfondies nous permettront ensuite de compléter la cartographie de la consommation en prenant en compte d'autres composants matériels (*e.g.*, GPU, SSD).
La suite de l'article est organisée comme suit. Premièrement, nous présenterons l'état de l'art en Section 2. Nous introduirons ensuite l'architecture de POWERAPI dans la Section 3, puis nous décrirons notre modèle énergétique en Section 4. Nous conclurons en Section 5.

2. État de l'art

Les processeurs ne permettant pas d'obtenir des estimations énergétiques fines, McCullough *et. al* [10] montrent que l'utilisation de profils énergétiques est primordiale. À l'heure actuelle, la solution la plus connue a été développée par Intel (RAPL). Disponible sur certaines générations de processeurs, elle permet d'estimer en temps réel la consommation énergétique du processeur.
Cette solution n'étant disponible que sur certains processeurs, les modèles énergétiques sont le plus souvent utilisés et se basent sur un certain nombre de compteurs de performance. Par exemple, Li et John [8] utilisent 5 compteurs, dont celui permettant de récupérer le nombre d'instructions par cycle (IPC), et obtiennent un modèle énergétique par régression. Un travail similaire a été effectué par Contreras *et al.* [4] qui considère, en plus des travaux précédemment cités, les fréquences disponibles sur le processeur mais se limite aux architectures mono-cœur. Bircher et John [3] ont réussi à réduire le pourcentage d'erreur de leur modèle à 9% en utilisant les compteurs de performance et estiment la consommation des composants matériels. Les auteurs de l'article [13] décident d'utiliser l'ensemble des compteurs de performance disponibles et réduisent leur nombre en analysant les relations qu'il y a entre eux. Les modèles énergétiques sont, le plus souvent, validés en comparant les estimations produites aux mesures provenant

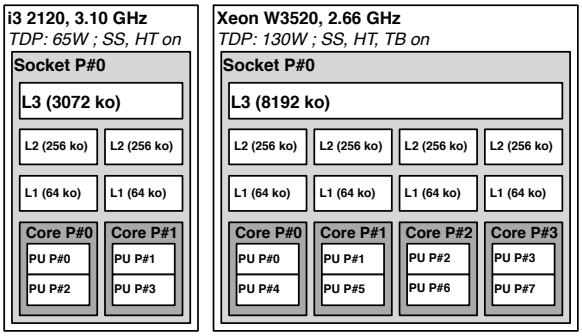

FIGURE 1 – Topologies du Core i3 & Xeon.

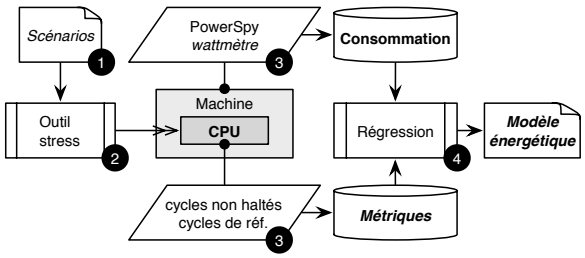

FIGURE 2 – Phase d'apprentissage du modèle énergétique.

d'un wattmètre externe [18].
Leurs modélisations nécessitent, la plupart du temps, des techniques automatiques d'apprentissage. L'une des techniques principalement utilisée est l'échantillonnage [1]. Elle assume que la consommation est liée à la charge du système. Les données récoltées à l'aide d'un wattmètre physique et l'ensemble normalisé des métriques récoltées sont alors injectés dans diverses techniques de régression, la plupart du temps linéaires [10].
Kansal *et al.* [6] et Versick *et al.* [17] démontrent que les modèles linéaires utilisant la charge du processeur ne sont pas assez précis et qu'il est nécessaire de considérer d'autres paramètres. McCullough *et al.* montrent que les modèles linéaires ont une erreur moyenne relative de 10 à 14% et qu'il est difficile d'obtenir de meilleurs résultats, même en appliquant des techniques plus complexes. Ces modèles se basent sur l'indépendance des variables, ce qui ne peut pas être mis à l'échelle des technologies mises en œuvre sur le processeur. Les régressions polynomiales ou exponentielles sont préférées pour modéliser ce type d'architecture. Comme présenté en [3], la solution quadratique modélise parfaitement les architectures multi-cœurs. Ces modèles doivent cependant isoler les fonctionnalités implémentées sur le processeur, comme l'*HyperThreading* ou le *TurboBoost*. HAPPY [19] introduit donc un modèle prenant en compte l'*HyperThreading*. Ce modèle permet de différencier les cas où un seul des deux *hyperthreads* d'un cœur est utilisé.
Shen *et al.* [14] propose des conteneurs d'énergie pour gérer la consommation énergétique des serveurs multi-cœurs et des plate-formes *cloud*. Les auteurs évaluent l'efficacité de leur solution en exécutant plusieurs applications en parallèle tout en considérant chacune d'entre-elles séparément. Lors de notre évaluation de POWERAPI, nous prenons également soin de vérifier que les estimations demeurent précises dans le cas d'exécutions concurrentielles.

3. POWERAPI, un watt-mètre logiciel efficace

POWERAPI a été développé dans le but de permettre aux utilisateurs d'assembler leurs wattmètres logiciels en fonction de leurs besoins. Les wattmètres logiciels sont des solutions modulables qui permettent de délivrer des estimations énergétiques à différents niveaux et à différentes fréquences.
Notre solution utilise le modèle par acteur pour assurer sa scalabilité, que ce soit sur la fréquence de rafraîchissement ou sur le nombre de processus suivis. Un acteur est une entité légère permettant de traiter des millions de messages à la seconde, indispensable pour supporter des estimations en temps réel. Notre outil, POWERAPI, définit plusieurs composants :

Clock représente l'horloge interne permettant à POWERAPI de fonctionner à différentes fréquences. Des messages sont publiés sur le bus d'événements interne en fonction des fréquences paramétrées ;

Monitor représente le suivi énergétique d'un ou plusieurs processus. Il réagit aux messages publiés par la Clock en fonction de la fréquence paramétrée pour le suivi. Des messages sont publiés sur le bus d'événements interne pour chaque processus suivi. Monitor est également en charge d'agréger les estimations en utilisant la fonction paramétrée par l'utilisateur (*e.g.*, somme, moyenne, maximum) ;

Sensor connecte le wattmètre logiciel au système pour récolter les métriques utiles à la représentation des activités des processus. Ces métriques peuvent représenter la consommation globale d'une machine reportée par un wattmètre externe, des sondes internes (*e.g.*, RAPL) ou des statistiques d'utilisation du processeur (PROCFS). Ce composant réagit aux messages publiés par les Monitor et publie les données récoltées à destination de la formule adéquate ;

Formula représente le modèle utilisé pour produire une estimation énergétique. Il réagit aux messages publiés par le Sensor et convertit les métriques envoyées en consommations. La granularité des estimations (machine, cœur, processus) dépend de la granularité des données reçues (*i.e.*, des données récoltées par les Sensor) ;

Reporter produit des rapports formatés permettant d'afficher les estimations énergétiques selon un format choisi. Ces rapports peuvent être affichés, par exemple, au sein d'une interface Web, d'un fichier ou en console.

Pour améliorer les solutions proposées dans l'état de l'art, nous avons créé un modèle énergétique basé sur libpfm [3] pour utiliser les compteurs de performance en tant que métriques (cf. Section 4).

POWERAPI est un outil puissant permettant d'assembler des wattmètres logiciels à la demande. Les résultats présentés dans les sections suivantes sont basés sur un ensemble de wattmètres créés avec POWERAPI pour suivre la consommation globale d'un système avec un wattmètre externe ou des sondes, apprendre le profil énergétique du processeur et délivrer des estimations énergétiques.

4. Modèle énergétique des processeurs multi-cœurs

POWERAPI utilise des modèles dédiés pour estimer la consommation énergétique des processus. L'estimation à bas niveau est cruciale pour identifier le processus le plus gourmand en énergie et ainsi prendre des contre-mesures. Nous allons voir dans cette section comment notre modèle supporte les architectures multi-cœurs modernes.

Des processeurs optimisés.

Pour contrôler leurs consommations énergétiques, les processeurs peuvent échelonner leurs fréquences, ou alors utiliser différents modes de sommeil pour ajuster leurs performances à la demande. En particulier, les processeurs multi-cœurs développés par Intel possèdent les optimisations suivantes :

Hyper-Threading (HT) est utilisé par différentes générations de processeur (*e.g.*, Pentium IV, Xeon) pour diviser chaque cœur en deux *threads* logiques. Cette technologie est basée sur le *Simultaneous Multi-Threading* (SMT) permettant aux processeurs d'augmenter leurs niveaux de parallélisation et de partager plus efficacement les ressources. Les gains

3. http://perfmon2.sourceforge.net

de performance dépendent de l'implémentation des logiciels, qui tirent ou non parti de ces niveaux de parallélisation. Néanmoins, il peut être plus efficace, dans certains cas, de désactiver cette technologie.

SpeedStep (SS) est l'implémentation d'Intel du *Dynamic Voltage/Frequency Scaling* (DVFS) permettant au processeur d'ajuster sa fréquence et sa tension en fonction des besoins. Le système d'exploitation peut ainsi augmenter la fréquence du processeur pour exécuter plus rapidement des opérations ou alors la diminuer pour minimiser l'énergie dissipée lorsque le processeur est sous-utilisé.

TurboBoost (TB) permet, selon le niveau d'activité du processeur, d'augmenter sa fréquence maximale. Elle peut être plus importante que le *Thermal Design Frequency* (TDP) durant un court laps de temps. Cette technologie permet donc au processeur d'exécuter plus d'instructions si nécessaire. Elle n'est cependant activée que lorsque certaines conditions sont vérifiées, notamment si le nombre de cœurs actifs est faible et que la température interne du processeur n'est pas trop importante. Son activation dépend également du système d'exploitation qui la requiert lorsqu'une application a besoin d'une puissance de calcul supplémentaire.

Deux familles de processeur Intel ont été utilisées lors de l'évaluation de POWERAPI. Leurs complexités ont été analysées par l'outil *Portable Hardware Locality* (`hwloc`[4]) et sont présentées dans la figure 1. Elles diffèrent notamment par le nombre de cœurs et de *threads* disponibles mais également par les technologies implémentées sur le processeur (TB) et qui peuvent être exploitées par le système d'exploitation.

Apprentissage du modèle.

Apprendre le modèle énergétique d'un processeur multi-cœur requiert la définition de scénarios capables de stresser minutieusement les technologies disponibles et activées sur le processeur. C'est pour cela qu'il est important d'isoler les bruits énergétiques produits par d'autres composants et ainsi capturer efficacement la consommation du processeur évalué. Nous avons, pour cela, choisi la commande `stress`[5], disponible sur la plupart des distributions UNIX, pour développer des scénarios spécifiques de charge. Ces scénarios nous permettent de stresser de manière incrémentale les différents composants matériels comme le processeur, la mémoire ou le disque.

Différents scénarios ont été créés grâce aux options de la commande `stress`. Dans un premier temps, le processeur est stimulé cœur par cœur à charge maximale dans le but de capturer sa fréquence maximale et d'observer les effets de l'*Hyper-Threading* et du *TurboBoost* sur la consommation énergétique. La charge du processeur est ensuite changée dynamiquement pour capturer les effets du *SpeedStep* sur la consommation. Les mêmes étapes sont ensuite appliquées pour chaque fréquence du processeur en utilisant `cpufreq-utils`. Plus de détails sur l'algorithme et les options des outils utilisés sont disponibles au sein de POWERAPI. Grâce à cela, nous sommes capable d'identifier les fréquences utilisées par le *TurboBoost*.

Pour inférer le modèle énergétique, il est nécessaire de collecter des données capables de représenter le plus grand nombre d'applications. Comme présenté par [6], la charge processeur ne représente pas correctement toutes les activités du processeur. C'est pour cette raison que notre modèle se base sur les compteurs de performance, métriques précises reflétant les différents types d'opérations exécutées par le processeur. Nous avons décidé d'utiliser la bibliothèque `libpfm4` pour accéder aux compteurs disponibles sur les processeurs. Les compteurs utili-

4. `http://www.open-mpi.org/projects/hwloc`
5. `http://linux.die.net/man/1/stress`

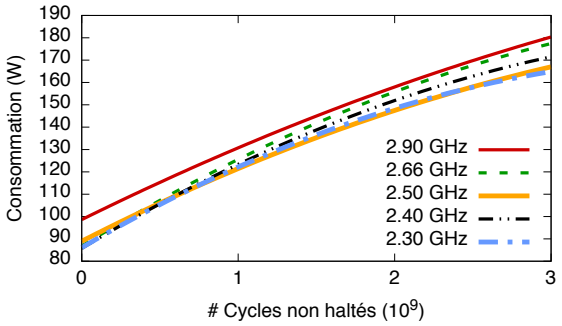

FIGURE 3 – Modèles énergétiques pour les plus hautes fréquences du processeur Xeon.

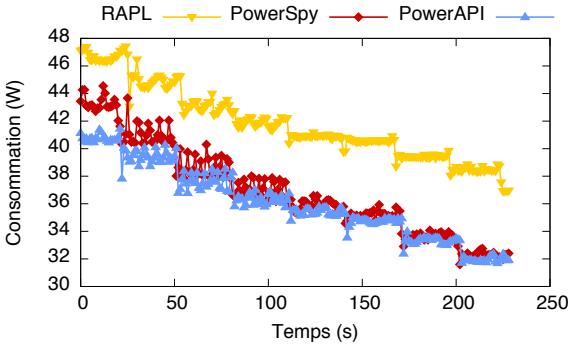

FIGURE 4 – Charge décroissante de la commande `stress` sur le processeur i3.

sés pour estimer la consommation énergétique doivent être sélectionnés en fonction de leurs disponibilités sur un grand nombre d'architectures et du coût imposé par leurs exploitations.
Notre principal objectif ici est de construire un modèle énergétique avec une faible empreinte énergétique. De part nos observations, nous avons choisi d'utiliser uniquement deux compteurs de performance : les cycles non haltés (uc) [6] [9, 18, 19] et les cycles de référence (rc) [7] pour représenter au mieux les différentes activés des architectures multi-cœurs modernes. Alors que le premier compteur représente le nombre cycles réellement exécutés et donc l'activité des cœurs, le second représente le nombre de cycles comptés à une fréquence de référence qui peut être différente de la fréquence actuelle du processeur. Le second est donc très utile pour estimer la fréquence d'un cœur.
La fréquence moyenne (f) est calculée en divisant le nombre de cycles non haltés par le nombre de cycles de référence ($f = uc/rc$). Cette fréquence moyenne est ensuite utilisée pour construire les modèles énergétiques et choisir la formule à utiliser durant l'exécution.
Un wattmètre externe, suivant la consommation de l'ordinateur dans sa globalité, est utilisé durant la phase d'apprentissage du modèle. Nous utilisons ici le wattmètre Bluetooth PowerSpy [8]. Ce wattmètre permet, en fonction du pays, de récupérer la consommation selon une fréquence comprise entre 45 et 65 Hz. La consommation calculée par le wattmètre est la résultante de la moyenne obtenue sur l'ensemble des valeurs d'une période. Dans le but d'améliorer nos modèles, la phase d'apprentissage et nos expériences sont exécutées plusieurs fois, réduisant ainsi les variations induites par les mesures physiques.

Inférence du modèle par régression.

Les compteurs de performance et les consommations récoltées durant la phase d'apprentissage sont alors corrélés à l'aide d'une régression polynomiale. Un modèle est construit par fréquence disponible sur le processeur et représente la consommation d'un seul cœur, permettant ainsi de modéliser efficacement l'*HyperThreading* [19]. Nous partons ensuite du principe que la consommation énergétique augmente linéairement en fonction du nombre de cœurs actifs. La figure 2 explique schématiquement la phase d'apprentissage.
En pratique, le modèle énergétique que nous obtenons pour un cœur à la fréquence f sur un

6. `CPU-CLK-UNHALTED:THREAD, event=0x003C`
7. `CPU-CLK-UNHALTED:REF, event=0x013c`
8. `http://www.alciom.com/en/products/powerspy2-en-gb-2.html`

processeur Intel Xeon sur une période peut être modélisé par cette équation :

$$P(f) = P_{idle}(f) + \sum_{pid \in PIDs} P_{cpu}(f, uc^1_{pid} ... uc^N_{pid})$$

Où $P_{idle}(f)$ correspond à la consommation statique (*i.e.*, la consommation totale au repos, induite par le matériel) de la machine pour la fréquence f qui a été inférée lors de la régression (les consommations au repos sont collectés durant quelques secondes à la fréquence fixée). $uc^1_{pid} ... uc^N_{pid}$ représente un vecteur de cycles non haltés collectés par identifiant de processus pid et par cœur 1..N. La consommation énergétique du processeur, P_{cpu}, est définie comme la somme des consommations par fréquence, P_f, pour chaque cœur n :

$$P_{cpu}(f, uc^1_{pid} ... uc^N_{pid}) = \sum_{n=1}^{N} P_f(uc^n_{pid})$$

Nous obtenons finalement un modèle énergétique par fréquence, TB inclus. L'une des formules est décrite comme suit pour l'une des fréquences du TB sur le processeur Intel Xeon (2.90 GHz) :

$$P_{2.90}(uc_{pid}) = \frac{8.64 \cdot uc_{pid}}{10^9} - \frac{6.10 \cdot uc^2_{pid}}{10^{18}}$$

La formule obtenue est un polynôme de degré 2 (illustré dans la figure 3).
Cette formule est conforme aux résultats publiés dans la littérature et montre l'impact de l'*HyperThreading* sur les modèles énergétiques [19]. Cette figure trace l'estimation des consommations énergétiques en fonction du nombre de cycles non haltés pour chacun des modèles inférés par fréquence. Pour des raisons de clarté, seules les fréquences au dessus de 2.30 GHz sont affichées. La consommation statique ($P_{idle}(f)$ quand $x = 0$) est calculée lors de la régression et est impactée par la fréquence courante du processeur. Nous pouvons observer que la courbe représentant la fréquence de 2.50 GHz est au dessus de celle de la fréquence de 2.40 GHz. Cela peut être expliqué par l'imprécision de `cpufreq-utils` qui ne nous assure pas que le processeur est constamment à la fréquence demandée car elle est contrôlée au niveau logiciel, et non pas matériel. Il est donc possible qu'il y ait certaines variations et imprécisions lors de la génération des modèles, surtout si les fréquences sont proches.

Validation du modèle.

Pour démontrer que POWERAPI est capable de suivre des processus ayant des charges non constantes, nous avons commencé par une expérience sur le processeur *i3*. Elle utilise les commandes `stress` et `cpulimit` pour contrôler la charge du processus sur un seul cœur. Ici, nous utilisons uniquement l'option `-c` de la première commande pour cibler uniquement un cœur, et les options `-cp` pour `cpulimit` dans le but de réduire la charge du processeur de 10% toutes les 30 secondes. De plus, nous nous assurons que le processus associé à la commande `stress` soit bien attaché au même cœur durant toute l'expérience. Lors de cette expérience, nous comparons les estimations de POWERAPI, de RAPL et des valeurs mesurées par PowerSpy. Pour rappel, RAPL permet de récupérer, par l'intermédiaire de registres matériels, l'estimation de la consommation énergétique du processeur sur certaines générations récentes. Nous avons également fixé la fréquence du processeur à 1.6 GHz pour éviter les instabilités des mesures. La figure 4 nous montre que RAPL suit la même tendance que les mesures produites par PowerSpy mais qu'il surestime toujours la consommation. Nous pouvons remarquer ici que le taux d'erreur de notre solution est plus important lorsque la charge appliquée est la plus élevée. Nous pouvons l'expliquer car notre modèle est inféré par régression, ce qui peut engendrer

un taux d'erreur plus élevé dans certains cas. Cependant, le taux d'erreur maximal relevé ici est inférieur à 1%. Comparé à RAPL, POWERAPI est plus précis et montre qu'il est capable d'estimer avec précision la consommation énergétique d'un processus, même si sa charge est fluctuante.
Dans les expériences suivantes, nous nous assurons que notre modèle énergétique est également valable pour des applications multi-cœurs. Nous utilisons, pour cela, la suite de benchmarks PARSEC [2] dans sa version 2.1. Elle a été créée dans le but de stimuler toutes les ressources des processeurs multi-cœurs. Nous présentons ici la consommation énergétique des benchmarks disponibles sur les deux configurations de processeur présentées précédemment. La figure 5 affiche les erreurs relatives entre les estimations et les consommations mesurées. Bien que la suite n'ait pas été utilisée lors de notre phase d'échantillonnage, nous observons que les estimations sont proches des mesures collectées par le wattmètre. La méthode la plus proche de la notre, décrite en [17], adopte une approche itérative pour affiner leurs estimations et a un taux d'erreur de 5%. Elle est cependant limitée car elle ne considère qu'une charge maximale constante sur le processeur et que le modèle est spécifique aux applications suivies. La solution que nous proposons ici est indépendante des applications suivies, peut suivre aussi bien celles qui utilisent le processeur, la mémoire du processeur intensivement ou tirent profit des différentes technologies disponibles.
La figure 6 illustre la capacité d'estimer et d'isoler la consommation énergétique de processus s'exécutant en parallèle sur le même processeur. En particulier, elle montre comment la consommation énergétique du processeur Xeon est répartie entre sa consommation statique (induite par le matériel) et deux benchmarks de la suite PARSEC (x264 et freqmine). Avec une fréquence de rafraîchissement de 4 Hz (une valeur toutes les 250 ms), notre solution possède un taux d'erreur moyen de 0.30% et un taux d'erreur maximal de 9.73%. La plus grande limitation de POWERAPI est la fréquence d'accès aux capteurs logiciels et matériels utilisés. Plus précisément, POWERAPI peut estimer la consommation des processus jusqu'à 40 Hz lorsqu'il est connecté à PowerSpy, et jusqu'à 10 Hz en utilisant la bibliothèque `libpfm4`. Nous pouvons remarquer que plus la fréquence de rafraîchissement est élevée, moins les estimations sont stables. POWERAPI est donc configuré avec une fréquence de 1 Hz dans le reste de l'article pour stabiliser les estimations présentées. La figure 6 montre également la consommation de POWERAPI durant l'exécution de l'expérience. Sa consommation est de 5.4W en moyenne (mesurée à une fréquence de 4 Hz), démontrant ainsi que son empreinte énergétique est faible. Celle-ci dépend fortement de la fréquence choisie et du nombre d'applications suivies. Ces expériences valident notre solution, POWERAPI, comme étant efficace et fiable pour la construction de wattmètres logiciels.

Vers un modèle générique ?

Alors que notre modèle énergétique multi-cœur n'est vérifié que sur des processeurs Intel (cf. figure 1), notre solution ne se base pas sur des extensions spécifiques aux processeurs Intel. En effet, notre modèle considère différentes technologies (HT, SS, TB) qui sont aussi disponibles sur d'autres marques de processeurs. Les processeurs AMD sont également une cible intéressante mais une limitation de la bibliothèque `libpfm4` nous empêche pour le moment d'accéder au compteur de cycles de référence, nous empêchant donc de calculer la fréquence courante. Une fois cette barrière franchie, nous nous attendons à démontrer la validité de POWERAPI sur ces processeurs.
Nous également très intéressé par les processeurs ARM, très utilisés et ayant des architectures très intéressantes. Nos premiers essais ont été infructueux car il n'y avait pas assez de mémoire

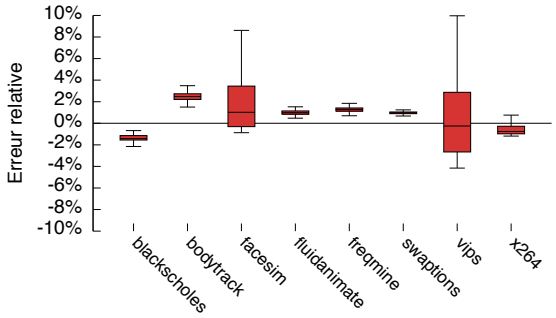

FIGURE 5 – Distribution des erreurs relatives pour PARSEC sur le processeur Xeon.

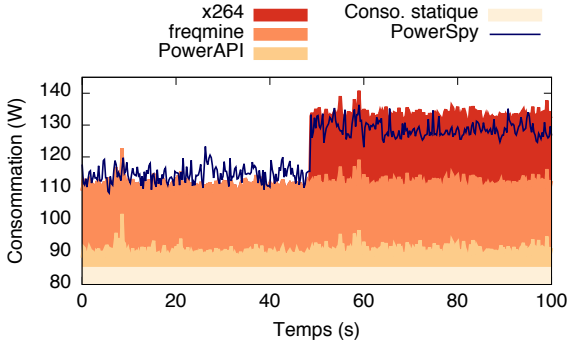

FIGURE 6 – Consommations énergétiques à grain fin de POWERAPI, freqmine et x264.

vive sur notre Raspberry Pi 1. Les compteurs utilisés au sein de notre modèle étant disponibles, nous sommes assez confiants quant à l'utilisation de notre solution sur des processeurs ARM.

5. Conclusion

Dans cet article, nous avons présenté POWERAPI, permettant de construire des wattmètres logiciels.
Un tel wattmètre est une alternative précise aux matériels dédiés et aux sondes embarquées pour estimer la consommation énergétique finement—*i.e.*, au niveau processus. POWERAPI se doit de délivrer des estimations précises pour n'importe quel type d'application. C'est pour cela que nous avons développé un modèle énergétique capable de considérer les complexités internes des processeurs, comme l'*Hyper-Threading*, le *Dynamic Voltage/Frequency Scaling* et le *TurboBoost*, impactant fortement leurs consommations. Ce modèle est utilisé dans POWERAPI, sans aucune modification du système, pour délivrer des estimations énergétiques avec un taux d'erreur moyen de 2%.
À notre connaissance, POWERAPI est la première solution à estimer si précisément la consommation avec un modèle énergétique générique, *i.e.*, agnostique vis-à-vis de l'application. Nous avons évalué la performance de notre solution sur deux architectures de processeurs configurées différemment. Nous avons montré que POWERAPI fonctionne très bien à différents niveaux.
La consommation énergétique étant liée aux applications exécutées, les développeurs commencent à prendre cet aspect énergétique en compte durant leurs développements. À vrai dire, un nombre croissant d'outils ont été créés pour afficher des informations concernant l'efficacité énergétique de leurs programmes aux développeurs [11, 15]. Nous espérons que POWERAPI représente une précieuse contribution pour les chercheurs, les développeurs et les ingénieurs. Le code est open-source et disponible sous licence AGPL sur GITHUB [9].

Bibliographie

1. Bertran (R.), Becerra (Y.), Carrera (D.), Beltran (V.), Gonzílez (M.), Martorell (X.), Navarro (N.), Torres (J.) et Ayguadé (E.). – Energy Accounting for Shared Virtualized Environments Under DVFS Using PMC-based Power Models. *Future Generation Computer Systems*, vol. 28, n2, 2012, pp. pp. 457–468.

9. http://powerapi.org

2. Bienia (C.). – *Benchmarking Modern Multiprocessors*. – Thèse de PhD, Princeton University, janvier 2011.
3. Bircher (W. L.) et John (L. K.). – Complete System Power Estimation : A Trickle-Down Approach Based on Performance Events. – In *Proc. of IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, pp. 158–168, avril 2007.
4. Contreras (G.) et Martonosi (M.). – Power prediction for Intel XScale processors using performance monitoring unit events. – In *Proc. of IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 221–226, août 2005.
5. Hähnel (M.), Döbel (B.), Völp (M.) et Härtig (H.). – Measuring Energy Consumption for Short Code Paths Using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, n3, décembre 2012, pp. pp. 13–17.
6. Kansal (A.), Zhao (F.), Liu (J.), Kothari (N.) et Bhattacharya (A. A.). – Virtual Machine Power Metering and Provisioning. – In *Proc. of ACM Symposium on Cloud Computing (SoCC)*, pp. 39–50, juin 2010.
7. Krevat (E.), Tucek (J.) et Ganger (G. R.). – Disks Are Like Snowflakes : No Two Are Alike. – In *Proc. of USENIX conference on Hot topics in Operating Systems (HotOS)*, pp. 14–14, mai 2011.
8. Li (T.) et John (L. K.). – Run-time Modeling and Estimation of Operating System Power Consumption. *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, n1, juin 2003, pp. pp. 160–171.
9. Lim (M. Y.), Porterfield (A.) et Fowler (R. J.). – SoftPower : Fine-Grain Power Estimations Using Performance Counters. – In *Proc. of ACM International Symposium on High Performance Distributed Computing (HPDC)*, pp. 308–311, juin 2010.
10. McCullough (J. C.), Agarwal (Y.), Chandrashekar (J.), Kuppuswamy (S.), Snoeren (A. C.) et Gupta (R. K.). – Evaluating the Effectiveness of Model-Based Power Characterization. – In *Proc. of USENIX Annual Technical Conference (ATC)*, pp. 12–12, juin 2011.
11. Noureddine (A.), Rouvoy (R.) et Seinturier (L.). – Unit Testing of Energy Consumption of Software Libraries. – In *Proc. of ACM Symposium On Applied Computing*, pp. 1200–1205, mars 2014.
12. Orgerie (A.-C.), Assuncao (M. D. d.) et Lefevre (L.). – A Survey on Techniques for Improving the Energy Efficiency of Large-scale Distributed Systems. *ACM Computing Surveys (CSUR)*, vol. 46, n4, avril 2014, pp. pp. 47 :1–47 :31.
13. Powell (M. D.), Biswas (A.), Emer (J. S.), Mukherjee (S. S.), Sheikh (B. R.) et Yardi (S.). – CAMP : A Technique to Estimate Per-Structure Power at Run-time using a Few Simple Parameters. – In *Proc. of IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 289–300, février 2009.
14. Shen (K.), Shriraman (A.), Dwarkadas (S.), Zhang (X.) et Chen (Z.). – Power Containers : An OS Facility for Fine-grained Power and Energy Management on Multicore Servers. – In *Proc. of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 65–76, avril 2013.
15. Sterling (C.). – Energy Consumption tool in Visual Studio 2013, juillet 2013.
16. The Climate Group. – SMART 2020 : Enabling the low carbon economy in the information age, 2008.
17. Versick (D.), Wassmann (I.) et Tavangarian (D.). – Power Consumption Estimation of CPU and Peripheral Components in Virtual Machines. *ACM SIGAPP Applied Computing Review*, vol. 13, n3, septembre 2013, pp. pp. 17–25.
18. Wang (S.), Chen (H.) et Shi (W.). – SPAN : A software power analyzer for multicore computer systems. *Sustainable Computing : Informatics and Systems*, vol. 1, n1, 2011, pp. pp. 23–34.
19. Zhai (Y.), Zhang (X.), Eranian (S.), Tang (L.) et Mars (J.). – HaPPy : Hyperthread-aware Power Profiling Dynamically. – In *Proc. of USENIX Annual Technical Conference (ATC)*, pp. 211–217, juin 2014.