Modélisation, simulation et validation expérimentale du greenup de la parallélisation idéale d'un programme sur architecture multicœur

Antoine Solcourt, Georges-André Silber

Centre de recherche en informatique, Mines Paris — PSL, France
antoine.solcourt@minesparis.psl.eu

Résumé
Nous présentons un modèle analytique conçu pour estimer le *greenup*, soit le gain en efficacité énergétique, réalisé lors de la parallélisation idéale d'un programme sur un processeur multicœur. Contrairement au gain de vitesse qui croît linéairement, l'amélioration énergétique est limitée par la consommation des ressources partagées, appelées *uncore*, par rapport à celle des cœurs individuels. Nous validons notre approche par une étude expérimentale sur une architecture Intel Haswell, utilisant un programme synthétique et des mesures via les compteurs RAPL. Les résultats confirment que le gain énergétique converge vers une borne asymptotique prévisible, offrant ainsi un outil simple pour optimiser le nombre de cœurs actifs. Cette décomposition entre composants *core* et *uncore* permet de mieux comprendre la scalabilité énergétique des systèmes informatiques modernes.

Mots-clés : greenup, parallélisation idéale, modèle analytique, énergie, architecture multicœur.

1. Introduction

Paralléliser un programme sur plusieurs cœurs ne modifie pas la quantité totale d'instructions à exécuter. En revanche, cette optimisation peut permettre de réduire l'énergie consommée pendant l'exécution sur un processeur multicœur. Néanmoins, même dans le cas d'un programme idéalement parallélisé, cette réduction admet une borne supérieure. Afin d'identifier cette borne, nous proposons un modèle simple permettant de déterminer le nombre de cœurs minimisant la consommation d'énergie, et se fondant sur la répartition de la consommation entre différentes parties du processeur.
Cette estimation statique ne nécessite pas l'exécution préalable du programme mais repose néanmoins sur la caractérisation en amont d'un ratio énergétique, reliant l'énergie dépendant de l'activité des cœurs exécutant du code, appelée *core*, à l'activité inter-cœurs pendant l'exécution du code, appelée *uncore*. Ce ratio dépend à la fois de l'architecture du processeur et du type de calcul considéré : *CPU bound* ou *memory bound*.

2. Modèle de consommation d'énergie d'un processeur multicœur homogène

Dans un processeur multicœur, il est possible de distinguer la consommation d'énergie des cœurs (consommation *core*, incluant les caches L1 et L2) de celle des ressources partagées (*un-*

core, incluant le cache L3 et les interconnexions) [10]. Dans le cadre d'une exécution parallèle, nous définissons le *travail*, exprimé en nombre de cycles et noté c, comme la somme de tous les cycles utilisés par les instructions exécutées sur tous les cœurs participant à l'exécution du programme. La formulation du modèle repose sur les hypothèses suivantes.

- **(i)** Le travail considéré est supposé invariant vis-à-vis du degré de parallélisation. Cette hypothèse de parallélisation idéale conduit à négliger les effets de contention, de déséquilibre de charge ainsi que les surcoûts de synchronisation.
- **(ii)** On note $\bar{e}_{core}$ l'énergie moyenne consommée par cycle et par cœur (en joules), et $\bar{e}_{uncore}$ l'énergie moyenne consommée par cycle par l'ensemble des ressources partagées [7]. Ces deux grandeurs sont supposées constantes tout au long de l'exécution d'un programme donné.
- **(iii)** Les cœurs sont considérés homogènes et décrits par des paramètres énergétiques identiques.
- **(iv)** La consommation des ressources partagées est supposée indépendante du nombre de cœurs actifs, traduisant une invariance du travail total (voir Section 4). Ainsi, l'activation de cœurs supplémentaires n'engendre ni surcoût énergétique au niveau *uncore*, ni contention ou phénomène de *stall* imputable à ces ressources.
- **(v)** dans ce cadre idéal, lors de l'exécution parallèle, n cœurs sont actifs pendant une durée de c/n cycles. Les cœurs inactifs sont supposés éteints et ne consomment aucune énergie.

Les cœurs actifs consomment alors une énergie cumulée égale à $n \cdot \bar{e}_{core}$ joules par cycle, tandis que les ressources partagées consomment une énergie $\bar{e}_{uncore}$ joules par cycle. L'énergie totale consommée par le processeur au cours de l'exécution du programme s'écrit donc :

$$E(n) = (n \cdot \bar{e}_{core} + \bar{e}_{uncore})\frac{c}{n} \tag{1}$$

$$= (\bar{e}_{core} + \frac{\bar{e}_{uncore}}{n})c\,. \tag{2}$$

En reprenant la notion de *speedup* telle que définie par Amdahl [6, §1.9], égale au degré de parallélisme n dans le cas idéal, nous utilisons l'appellation de « *greenup* » utilisée notamment par [1]. Le *greenup*, $G(n)$, est le rapport entre l'énergie consommée lors de l'exécution séquentielle du programme, $E(1)$, et celle consommée lors de son exécution parallèle sur n cœurs, $E(n)$ (équation 3). Une valeur plus élevée de ce *greenup* traduit une réduction plus importante de l'énergie totale consommée.

$$G(n) = \frac{E(1)}{E(n)} = \frac{\bar{e}_{core} + \bar{e}_{uncore}}{\bar{e}_{core} + \bar{e}_{uncore}/n}\,. \tag{3}$$

Ce ratio met en évidence que seule la composante énergétique *uncore* voit sa contribution réduite lorsque le nombre de cœurs utilisés augmente. En effet, la consommation liée aux cœurs évolue linéairement avec le degré de parallélisme, tandis que la contribution des ressources partagées décroît relativement, celles-ci étant sollicitées sur une durée d'exécution réduite. Le *greenup* peut alors se réécrire en introduisant un ratio $r = \bar{e}_{uncore}/\bar{e}_{core}$:

$$G(n) = \frac{\bar{e}_{core}/\bar{e}_{core} + \bar{e}_{uncore}/\bar{e}_{core}}{\bar{e}_{core}/\bar{e}_{core} + \frac{\bar{e}_{uncore}/n}{\bar{e}_{core}}} = \frac{1+r}{1+r/n}\,. \tag{4}$$

Dans ce modèle, même pour un travail parfaitement parallélisable, le gain énergétique demeure contraint par ce ratio r. Le *greenup* ne suit donc pas le *speedup* (n ici) : il dépend de la

part relative de la consommation *uncore*, et plus celle-ci est faible, plus le bénéfice énergétique associé au parallélisme est limité. Lorsque n tend vers l'infini, le *greenup* converge vers une borne asymptotique égale à $1+r$, indépendamment du parallélisme disponible. L'obtention du ratio r peut être réalisée en mesurant le *greenup* d'un programme donné pour deux degrés de parallélisation différents, par exemple n et $n+1$, amenant le système suivant :

$$\begin{cases} G(n) = \dfrac{1+r}{1+\frac{r}{n}}, \\ G(n+1) = \dfrac{1+r}{1+\frac{r}{n+1}}. \end{cases} \tag{5}$$

Ce système admet une solution unique en r, notée r_n, car elle dépend, a priori, du choix de n. Celle-ci peut être estimée numériquement à partir de deux mesures successives du *greenup*. En utilisant les valeurs expérimentales $G(n)$ et $G(n+1)$:

$$r_n = \frac{G(n+1) - G(n)}{\frac{G(n)}{n} - \frac{G(n+1)}{n+1}}. \tag{6}$$

3. Étude expérimentale

L'objectif de cette étude expérimentale est de valider le modèle analytique proposé pour prévoir le *greenup* d'un programme parallélisé. Plus précisément, nous cherchons à vérifier que l'évolution du gain énergétique en fonction du degré de parallélisme suit bien la loi théorique définie à l'équation 4, où le ratio r est estimé expérimentalement. Les expérimentations sont réalisées sur une machine cible utilisant un processeur d'architecture Haswell - Intel Xeon E5-2630 v3.

3.1. Protocole expérimental

L'étude expérimentale de la relation entre parallélisme et consommation énergétique est réalisée au moyen d'un programme synthétique simulant différents niveaux de parallélisation d'un *travail* utilisant exclusivement le CPU (voir annexe A.1). Ce choix permet de s'approcher du cadre théorique de parallélisation idéale retenu dans le modèle.
Le nombre de cœurs actifs est contrôlé explicitement, de 1 jusqu'au nombre maximal disponible sur la machine. Pour chaque configuration, le programme exécute une quantité fixe de travail c, répartie entre les n cœurs, conformément à l'hypothèse d'invariance du travail total.
L'énergie consommée est mesurée à l'aide des compteurs RAPL (*Running Average Power Limit*), décrits en annexe A.2. Les mesures correspondent à l'énergie totale consommée par le processeur sur la durée complète d'exécution. La tension et la fréquence sont maintenues constantes et les mesures sont restreintes à une température et un voltage fixe de 42 °C et 6,659 V.
Chaque configuration est exécutée 50 fois afin de limiter l'impact du bruit expérimental (variabilité thermique, fluctuations liées au système d'exploitation, granularité de mesure). Les valeurs reportées correspondent à la médiane des mesures purgées des valeurs aberrantes, ici les valeurs dont l'écart à la médiane dépasse 3 fois l'écart médian absolu.

3.2. Mesures

Les mesures de consommation d'énergie d'un *travail* parfaitement parallélisé, réalisées pour un nombre de cœurs variant de 1 jusqu'au nombre disponible sur la machine cible, permettent de tracer la figure 1. Elle illustre l'amélioration de la consommation d'énergie G obtenue grâce à la parallélisation de ce travail.

En première approximation, nous voyons que l'on peut supposer r indépendant de n ; nous posons, par la suite, $r = 5.5 \pm 1.5$. La courbe en trait plein de la figure 2, correspond au *greenup* estimé par le modèle analytique, calculé à partir du ratio r estimé expérimentalement. Les points représentent les valeurs mesurées, chaque point correspondant à la médiane des exécutions réalisées pour un même degré de parallélisation. La droite en tirets mixtes représente le *speedup* idéal (n), celle en pointillés l'asymptote du greenup théorique 6,663.

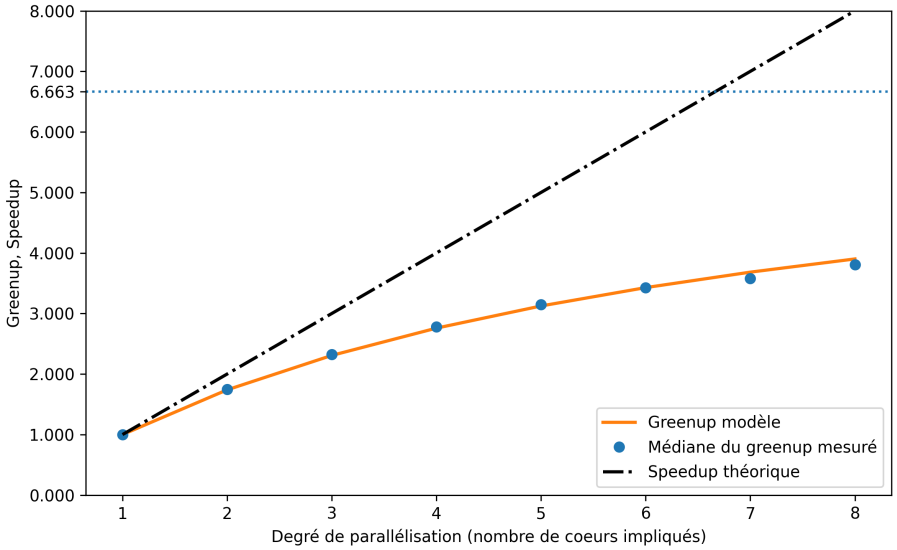

FIGURE 1 – Amélioration temporelle et énergétique en fonction du degré de parallélisation

La figure 1 permet d'observer que le *greenup* mesuré suit la tendance prévue par le modèle et présente la saturation asymptotique attendue. Contrairement au *speedup*, qui croît linéairement avec le nombre de cœurs dans une parallélisation idéale, le gain énergétique converge vers une borne supérieure déterminée par le ratio r, conformément à l'analyse théorique.

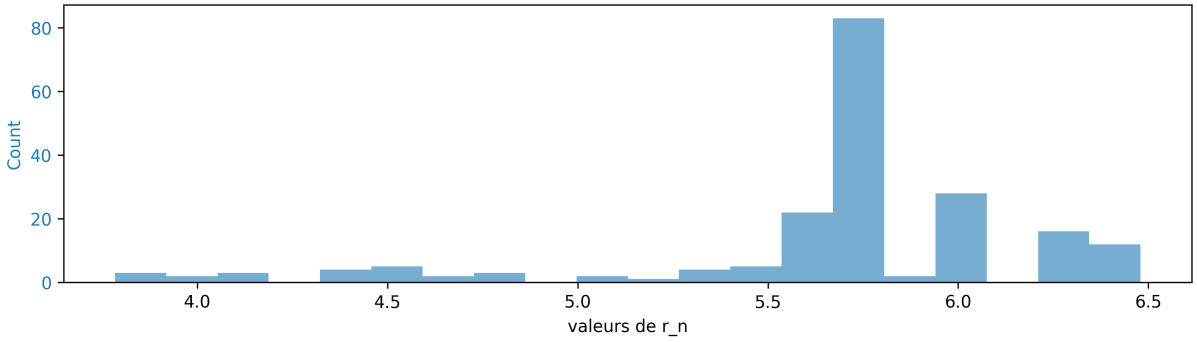

FIGURE 2 – Répartition des valeurs mesurées de r_n, en faisant varier n entre 1 et 8.

Dans ce cas idéal, l'erreur absolue entre le *greenup* mesuré et la valeur prévue par le modèle est

au maximum de 5,28 % (erreur quadratique moyenne). Cette proximité valide la pertinence de la décomposition énergétique entre *core* et *uncore*, ainsi que la capacité du modèle à capturer la diminution d'énergie consommée par un travail parallélisé.

4. Discussion

Comme nous l'avons vu, le ratio r, supposé théoriquement constant, présente en pratique certaines variations. À ce stade, ces écarts peuvent être attribués à plusieurs facteurs. Des phénomènes physiques propres aux transistors liés à une régulation imparfaite de la température lors des expériences sont susceptibles d'influencer les mesures. Par ailleurs, si le modèle considère la composante *uncore* comme indépendante du degré de parallélisme, il est vraisemblable qu'en pratique les ressources partagées soient davantage sollicitées à mesure que le nombre de cœurs actifs augmente. Il serait alors souhaitable de prendre en compte ces variations dans le modèle. Réduire la consommation d'énergie globale du processeur passe par la diminution conjointe des composantes $\bar{e}_{\text{core}}$ et $\bar{e}_{\text{uncore}}$. Toutefois, nous pouvons remarquer que la réduction de la composante $\bar{e}_{\text{core}}$, tout en conservant $\bar{e}_{\text{uncore}}$ constant, améliore directement la scalabilité énergétique.
Plusieurs autres mécanismes affectant la consommation d'énergie tels que le *power gating* [2] et les états d'éveil des cœurs (*ACPI*) [4] entrent également en compte dans la valeur du ratio r et nécessiteraient d'être caractérisés pour être intégrés au modèle. Une analyse plus fine des valeurs du ratio mesuré pour différents nombres de cœurs actifs, mettant en évidence divers régimes distincts, permettrait de conclure sur l'impact de la politique d'alimentation sur ces variations.

5. Travaux antérieurs

Parmi les premiers modèles de performance et de consommation énergétique dans les systèmes parallèles s'appuyant sur la loi d'Amdahl, Cho et Melhem [12] proposent une analyse des liens entre parallélisation, vitesse d'exécution et consommation d'énergie. Leur travail joue sur l'ajustement de la fréquence des processeurs entre les phases séquentielles et parallèles d'un programme. Dans ce modèle, la consommation dynamique est supposée proportionnelle à une puissance de la fréquence (f^{α} [1]), et la consommation statique est modélisée comme un terme constant par processeur. Les auteurs montrent notamment que, dans un régime énergétique optimal (borne inférieure de l'énergie consommée), l'énergie dynamique représente une fraction fixe de l'énergie statique, indépendante du nombre de processeurs ou du taux de parallélisme.
Cependant, l'approche de Cho et Melhem repose sur plusieurs hypothèses : (1) possibilité de faire varier la fréquence entre les régions séquentielles et parallèles, (2) cœurs homogènes et (3) une abstraction du matériel qui ne distingue pas explicitement les différentes composantes microarchitecturales du processeur. En particulier, l'énergie *uncore* des ressources partagées (interconnexions, LLC, contrôleurs mémoire) n'est pas traitée comme une entité distincte. Dans la continuité de ces travaux, plusieurs auteurs ont cherché à affiner la modélisation énergétique des systèmes parallèles. Korthikanti et Agha [8] ainsi que Ge et al. [3] étendent l'analyse fondée sur la loi d'Amdahl en introduisant des métriques énergétiques ou énergie-performance dans un contexte plus restreint de calculs haute performance. D'autres approches, telles que celles de Wang et al. [14], combinent extinction de cœurs et DVFS pour réduire la consommation, en s'appuyant sur des modèles proches du matériel intégrant tension, fréquence et

1. exposant de puissance dynamique

paramètres physiques de la technologie employée tels que la capacité des transistors. Sur le plan expérimental, plusieurs travaux antérieurs à RAPL se sont appuyés sur des simulateurs micro-architecturaux détaillés, comme Wattch [9], ou plus récemment sur des outils de simulation énergétique Intel permettant une décomposition fine des sous-domaines matériels [5]. Dans ce travail, nous privilégions l'utilisation des compteurs matériels RAPL, qui offrent un compromis entre précision, reproductibilité et simplicité de mise en œuvre.
Notre approche est volontairement simplifiée et directement exploitable. En complément de l'approche proposée par Cho et Melhem [12], qui met en évidence l'existence d'un optimum énergétique par l'ajustement dynamique de la fréquence entre les phases séquentielles et parallèles, nous adoptons une perspective orthogonale centrée sur le choix du degré de parallélisme à fréquence fixée. Notre modèle s'appuie sur une décomposition explicite entre la consommation des cœurs (*core*) et celle des ressources partagées (*uncore*), permettant de caractériser analytiquement les gains énergétiques observés expérimentalement. D'un autre côté si Cho et Melhem mettent en évidence un optimum via le réglage dynamique de la fréquence, nous montrons qu'un optimum énergétique (nombre maximal de cœurs actifs) apparaît également par le seul choix du nombre de cœurs actifs, en raison du poids plus ou moins prépondérant de la consommation *uncore* sur les architectures multicœur.

6. Conclusion et travaux futurs

Cet article a présenté un modèle analytique simple pour l'optimisation énergétique de codes parallèles exécutés sur une architecture multicœur. Nous avons formulé une expression de l'énergie consommée qui distingue les contributions des cœurs (*core*) et des ressources partagées (*uncore*). Le modèle repose sur des hypothèses simplificatrices justifiées expérimentalement, notamment la négligence de fluctuation des paramètres de consommation d'énergie par cycles, ce qui permet de se concentrer sur l'impact de la parallélisation.
Les résultats expérimentaux, obtenus sur une plateforme Intel Haswell, montrent que le modèle estime avec une précision satisfaisante (erreur inférieure à 5,28 %) l'énergie en fonction du nombre de cœurs actifs. La comparaison entre mesures et estimations valide indirectement les hypothèses retenues, en particulier l'approximation par une constante de l'énergie par cycle.
Ce travail ouvre des perspectives pour l'optimisation énergétique des applications parallèles, en fournissant un cadre simple pour guider le choix du degré de parallélisme. Des extensions pourraient intégrer explicitement les effets de la fréquence dynamique (DVFS) ou de l'hétérogénéité des cœurs, afin d'affiner l'estimation pour d'autres configurations matérielles.
Plusieurs extensions de ce travail peuvent être envisagées, comme l'introduction d'une fraction séquentielle du programme. La prise en compte des coûts de communication et de synchronisation permettrait également d'évaluer la robustesse du modèle au-delà du cas *CPU-bound* idéal. L'étude de l'impact de divers types de charges sur le ratio d'énergie $r = \bar{e}_{\mathrm{uncore}}/\bar{e}_{\mathrm{core}}$ sera étudié, notamment avec des applications bornées par la mémoire.
De plus, les mécanismes matériels de gestion d'énergie tels que le *power gating*, l'extinction dynamique de cœurs, le DVFS ou l'adaptation de fréquence du domaine *uncore* sur des architectures récentes présents par exemple sur Intel Xeon Platinum 8568Y+ (architecture Emerald Rapids) suggèrent que le ratio r, supposé constant, dépend en réalité du degré de parallélisme et des politiques d'alimentation. L'impact de la température et de la fréquence, jusqu'ici contrôlés expérimentalement, pourrait ainsi être intégré au modèle. Enfin, une analyse statistique plus approfondie des mesures énergétiques contribuerait à quantifier plus finement la variabilité des résultats expérimentaux . L'utilisation de mesures externes avec des capteurs matériels permettrait de dépasser la disparité des mesures fournies par RAPL.

Bibliographie

1. Abdulsalam (S.), Zong (Z.), Gu (Q.) et Meikang Qiu. – Using the Greenup, Powerup, and Speedup metrics to evaluate software energy efficiency. – In *2015 Sixth International Green and Sustainable Computing Conference (IGSC)*, 2015.
2. Arora (M.), Manne (S.), Paul (I.), Jayasena (N.) et Tullsen (D. M.). – Understanding idle behavior and power gating mechanisms in the context of modern benchmarks on CPU-GPU Integrated systems. – In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 366–377, Burlingame, CA, USA, février 2015. IEEE.
3. Ge (R.), Feng (X.) et Cameron (K. W.). – Modeling and evaluating energy-performance efficiency of parallel processing on multicore based power aware systems. – In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pp. 1–8, Rome, Italy, May 2009. IEEE.
4. Hackenberg (D.), Schöne (R.), Ilsche (T.), Molka (D.), Schuchart (J.) et Geyer (R.). – An energy efficiency feature survey of the intel haswell processor. – In *2015 IEEE international parallel and distributed processing symposium workshop*, pp. 896–904. IEEE, 2015.
5. Haj-Yahya (J.), Mendelson (A.), Ben Asher (Y.) et Chattopadhyay (A.). – Compiler-Directed Energy Efficiency. *In : Energy Efficient High Performance Processors*, pp. 107–133. – Singapore, Springer Singapore, 2018.
6. Hennessy (J. L.) et Patterson (D. A.). – *Computer Architecture, A Quantitative Approach*. – Morgan Kaufmann, 2019, sixth édition.
7. Intel. – *Intel® 64 and IA-32 Architectures Software Developer's Manual*. – Rapport technique, June 2024.
8. Korthikanti (V. A.) et Agha (G.). – Analysis of Parallel Algorithms for Energy Conservation in Scalable Multicore Architectures. – In *2009 International Conference on Parallel Processing*, pp. 212–219, Vienna, Austria, September 2009. IEEE.
9. Li (J.) et Martinez (J.). – Power-Performance Implications of Thread-level Parallelism on Chip Multiprocessors. – In *IEEE International Symposium on Performance Analysis of Systems and Software, 2005. ISPASS 2005.*, pp. 124–134, Austin, TX, USA, 2005. IEEE.
10. Mazouz (A.), Wong (D. C.), Kuck (D.) et Jalby (W.). – An Incremental Methodology for Energy Measurement and Modeling. – In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, pp. 15–26, L'Aquila Italy, April 2017. ACM.
11. Raffin (G.) et Trystram (D.). – Dissecting the software-based measurement of CPU energy consumption : a comparative analysis. *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, n1, November 2024, p. 96.
12. Sangyeun Cho et Melhem (R.). – Corollaries to Amdahl's Law for Energy. *IEEE Computer Architecture Letters*, vol. 7, n1, January 2008, pp. 25–28.
13. Vaddina (K. R.), Brandner (F.), Memmi (G.) et Jouvelot (P.). – Experimental Assessment and Biaffine Modeling of the Impact of Ambient Temperature on SoC Power Requirements. – In *Embedded Computer Systems : Architectures, Modeling, and Simulation : 24th International Conference, SAMOS 2024, Samos, Greece, June 29 – July 4, 2024, Proceedings, Part I*, pp. 18–33, Berlin, Heidelberg, July 2024. Springer-Verlag.
14. Wang (Z.), Wang (H.), Zhao (W.) et Cheng (L.). – Energy optimization of parallel programs in a heterogeneous system by combining processor core-shutdown and dynamic voltage scaling. *Future Generation Computer Systems*, vol. 92, March 2019, pp. 198–209.

A. Protocole de simulation et de mesure

A.1. Simulation

Le simulateur[2], écrit en langage C, implémente un *travail* synthétique avec un degré de parallélisme n paramétrable occupant uniquement le CPU (*CPU-bound*). Il exécute une boucle de calcul intensif (des multiplications flottantes, définies par l'équation 7) répartie entre n threads, tout en fixant la fréquence CPU et en contrôlant l'affinité des cœurs (exécution sur un seul CPU dans le cas d'un serveur multi-socket) pour garantir des conditions expérimentales répétables. Pour simuler la parallélisation de la charge de calcul, chaque cœur sollicité exécute alors N/n pas de boucles.

$$\begin{aligned} &\text{Initialisation :} && x_0 \\ &\text{Récurrence :} && x_{i+1} = x_i \cdot a + b, \quad i = 0, 1, \ldots, N-1 \\ &\text{Solution fermée :} && x_N = (a)^N x_0 + b\frac{(a)^N - 1}{a - 1} \end{aligned} \tag{7}$$

avec $x_0 = 1,23456789$, $a = 1,0000001$ et $b = 0,0000001$. Nous vérifions l'absence d'*overflow* avec le calcul de la solution fermée. Le résultat de la boucle d'accumulation est conservé pour empêcher que le compilateur n'optimise la boucle. En faisant varier n, nous pouvons obtenir un temps d'exécution remplissant les conditions citées dans la section A.2.

A.2. Mesures

Les expérimentations sont réalisées sur des serveur Intel Xeon E5-2630 v3, en particulier sur le nœud *parasilo* du réseau Grid5000. Les serveurs considérés sont munis de deux sockets : dans le cadre de nos essais, seul un des deux processeurs est utilisé. Ces processeurs possèdent une architecture Haswell avec 8 cœurs physiques et 2 threads par cœurs. Nous avons fait le choix de désactiver l'*hyperthreading* (1 thread par cœur), car le comportement énergétique dépend du nombre de threads actifs par cœur, un effet encore non modélisé.

Chaque processeur étant isolé, les tâches exécutées sont supposées ne pas subir d'interférences d'autres tâches excepté le fonctionnement de l'OS, dans notre cas : `Linux SMP Debian 5.10.244-1 (2025-09-29) x86_64 GNU/Linux`. D'autre part, la fréquence (2,4 GHz), la tension et la température sont fixées. Avant chaque mesure, une boucle de conditionnement thermique vérifie que la température du cœur 0 soit à ± 5°C de la valeur initiale lue au début de l'exécution. En effet la consommation énergétique des processeurs est impactée par les variations de température [13].

La mesure de la consommation d'énergie de l'exécution d'une tâche s'appuie, pour cette étude, sur les compteurs RAPL (Running Average Power Limit) exposés via l'interface `perf_event` du noyau Linux. Comme discuté dans [11], cette interface fournit des informations suffisamment détaillées pour notre usage : elle fournit une précision de 1×10^{-3} J et un rafraîchissement des valeurs de l'ordre de la milliseconde.

Les compteurs RAPL sont organisés en plusieurs domaines (*package*, cœurs, mémoire, etc.). Nous limitons les mesures au domaine *package* (*pkg*), qui regroupe l'énergie consommée par les cœurs (PP0) ainsi que par la logique de contrôle du processeur (*uncore*) (le serveur étudié ne possédant pas de domaine PP0). La dénomination *uncore* est décrite dans le manuel du développeur Intel [7].

Pour chaque exécution du benchmark, le script initie l'ouverture d'un compteur *perf* de type RAPL correspondant à l'architecture cible localisé sous linux dans le fichier `/sys/bus/event_`

2. `https://github.com/gabiansystems/parallel_simulator_energy/tree/compas_2026`

`source/devices/power/type` [7] (exemple, Haswell : type 34, Skylake : type 53) et au domaine de mesure de même dans le dossier `[...]/power/events/energy-pkg` (PKG pour le package, PP0 pour les cœurs).
La valeur du compteur est initialisée avant chaque début de mesures (`ioctl(fd, PERF_EVENT _IOC_RESET, 0);`). Une seule lecture de la valeur du compteur est effectuée au début et à la fin de chaque expérience ; les résultats sont enregistrés au format pour analyse ultérieure. Afin de limiter l'erreur induite par le quantum de mise à jour ($\approx$ 1 ms), la durée d'exécution de la charge de travail N (voir listing 7) est choisie au moins un ordre de grandeur au-dessus de cette période, ici 1×10^9. Le temps d'exécution est également fixé de manière à rendre négligeable le surcoût énergétique introduit par le *wrapper* d'instrumentation.
L'énergie consommée au cours de l'exécution est alors calculée à partir de la valeur du compteur RAPL via la formule $\texttt{Energy} = \text{count} \times 2.3283064365386962890625 \times 10^{-10}$ J. Le facteur d'échelle est disponible dans un fichier au même chemin que le domaine de mesure, en ajoutant `.scale`. Les mesures de tension et de température sont lues directement depuis les MSR (Model Specific Registers) `IA32_PERF_STATUS` (0x198) et `IA32_THERM_STATUS` (0x19C). Le temps d'exécution est quant à lui obtenu par une lecture du temps avec `clock_gettime(x@)`.
Cette chaîne de mesure fournit des données énergétiques reproductibles avec une précision adaptée à l'analyse des compromis énergie/parallélisme.