Énergie ou performance : impact de l'implémentation sur la consommation

Hicham Nekt

ENSEIRB-MATMECA

Rapport de stage 2A
Mémoire Master 1

Énergie ou performance : impact de l'implémentation sur la consommation

Hicham Nekt

Encadrants : M. Abdou Guermouche MMe Amina Guermouche

Soutenu le : 26/10/2023

Table des matières

1 Introduction **2**

2 Contexte du stage **2**

3 État de l'art **3**

4 Architectures et outils **3**
4.1 Vectorisation : Instructions SIMD 4
4.2 Architectures 4
4.3 Logiciels 5
4.3.1 MKL 5
4.3.2 Chameleon 5
4.3.3 Likwid 5
4.4 Méthodologie 5

5 Expérimentations **7**
5.1 Application compute-bound : Chameleon 7
5.1.1 Impact de la vectorisation sur la consommation énergétique 7
5.2 Application memory-bound : SPMV 10

6 Conclusion Générale **11**

Résumé

L'objectif de mon stage à **Inria** de l'Université de Bordeaux était de réaliser et d'analyser le profil énergétique de plusieurs noyaux de calculs. Pour maximiser notre maîtrise de l'étude, nous avons choisi des noyaux effectuant des opérations de base, telles que le produit matriciel. Notre but était d'examiner l'impact des différentes implémentations sur à la fois les performances et la consommation d'énergie. Les différentes variantes ont porté sur des aspects tels que la vectorisation, la précision, etc. Pour généraliser les résultats, les tests ont été menés sur différentes machines équipées de processeurs **Intel**, mais de natures différentes.

Afin de répondre de manière approfondie à la problématique concernant la relation entre la vitesse et l'efficacité énergétique, notre étude s'est appuyée sur deux profiles d'application de calcul en HPC. Le premier correspondait à une application de calcul intensif appelée "compute-bound", tandis que le second se basait davantage sur le temps d'accès aux données que sur le temps de calcul, ce qui le qualifiait de "memory-bound". Cette approche nous a permis d'observer divers comportements énergétiques possibles.

1 Introduction

Inria a entamé ses activités le 3 janvier 1967 dans le cadre du plan Calcul , un plan gouvernemental français visant à garantir l'autonomie du pays dans le domaine informatique. **Inria** a pour mission de promouvoir la recherche en informatique, tant au niveau national qu'international, en explorant de nouvelles voies, souvent dans une perspective interdisciplinaire et en collaborant avec des partenaires industriels pour relever des défis ambitieux. Actuellement, **Inria** compte 10 centres de recherche situés au sein des principales universités de recherche, 215 équipes-projets, dont 80 % sont en partenariat avec d'autres acteurs, et 3 900 scientifiques.

J'ai eu l'opportunité d'effectuer mon stage au sein du centre **Inria** de Bordeaux, situé au 200 avenue de la Vieille Tour - 33405 TALENCE CEDEX. Ce centre regroupe 300 personnes, réparties en 20 équipes de recherche, travaillant dans quatre axes principaux : la simulation numérique et le calcul haute performance, l'optimisation et la quantification de l'incertitude, la santé numérique et l'interface homme-machine. Parmi ces 20 équipes, je faisais partie des équipes **TOPAL** et **STORM**.

L'équipe TOPAL se spécialise dans l'optimisation des algorithmes d'algèbre linéaire ainsi que l'intelligence artificielle. Parmi ses projets, on trouve PaStiX, un solveur linéaire pour les matrices creuses, et Chameleon, une bibliothèque d'algèbre linéaire dense.

L'équipe STORM se spécialise dans l'optimisation et les supports d'exécution.

2 Contexte du stage

Le HPC a connu d'incroyables avancées au cours des dernières années. En effet, le supercalculateur le plus puissant au monde **Frontier** atteint l'exaflop/s (un milliard de milliards d'opérations flottantes par seconde). Cependant, malgré l'accent mis sur l'optimisation des performances, un aspect fondamental a été négligé : l'efficacité énergétique, puisque **Frontier** consomme 21MW, c'est la conso d'une ville américaine d'environ 11700 foyers. Actuellement, la recherche en HPC se tourne vers des préoccupations environnementales, prenant en compte non seulement les performances, mais aussi l'efficacité énergétique des applications de calcul haute performance. Dans ce contexte, l'objectif de mon stage est d'étudier l'efficacité énergétique de différents types d'applications de calcul haute performance. En d'autres termes, je cherche à examiner la relation entre la consommation d'énergie et les performances, soulevant ainsi la question suivante : l'augmentation de la vitesse de calcul entraîne-t-elle systématiquement une réduction de la consommation énergétique, ou le compromis n'est-il pas si évident ?

Pour répondre à cette problématique, notre étude portera sur le profilage énergétique de quelques noyaux de calcul effectuant des opérations de base, telles qu'une multiplication matricielle, tout en variant des paramètres tels que la précision des calculs (simple ou double), le type de vectorisation utilisé, etc. Cette étude se concentrera sur deux profiles d'applications de calcul en HPC. Le premier correspond à une application de calcul intensif appelée "compute-bound", tandis que le second repose davantage sur le temps d'accès aux données que sur le temps de calcul, ce qui le qualifie de "memory-bound".

Afin d'approfondir la compréhension du contexte de notre recherche, nous proposons une structure en plusieurs étapes. Tout d'abord, dans la section 3, nous examinerons l'état de l'art, offrant ainsi un aperçu des développements existants dans le domaine de notre étude. Ensuite, nous détaillerons les expérimentations réalisées sur différentes architectures, ces dernières étant présentées en détail dans la section 4. Après avoir exposé les résultats obtenus dans les sections 5 et 5.2, nous tirerons des conclusions dans la section 6 pour récapituler les principales découvertes et implications de notre recherche.

3 État de l'art

De nombreuses recherches ont été menées par divers acteurs, notamment les institutions académiques et industrielles, visant à optimiser la consommation énergétique des supercalculateurs qui requièrent des quantités considérables d'énergie.

Pour atteindre cet objectif, les chercheurs se sont penchés sur l'évaluation de l'influence de divers facteurs sur le profil énergétique de ces systèmes. Parmi ces facteurs, dans [5], les auteurs se sont penchés sur l'impact des différents niveaux de vectorisation sur plusieurs applications. Cette étude complémente notre propre travail, car nous effectuons des tests similaires sur d'autres applications en tenant compte de la précision des calculs, un facteur qui peut influencer la consommation d'énergie. En parallèle, l'article [4] explore l'impact énergétique des différents niveaux de précision. Dans notre étude, nous avons limité notre analyse à la simple précision et à la double précision, en y ajoutant une deuxième variable, à savoir la vectorisation.

D'autres études ont été menées sur d'autres aspects, tels que le choix du langage de programmation [3] et le déploiement d'accélérateurs GPU [2]. Ces travaux sont plus éloignés de notre étude, car nous ne nous sommes concentrés sur les CPU sans changer de langage de programmation. Mais elles peuvent néanmoins ouvrir des pistes pour des travaux futurs.

L'objectif de cette étude est de contribuer à la compréhension de l'impact de la précision des calculs sur la consommation énergétique dans le domaine du HPC, en se concentrant sur des aspects spécifiques tels que la vectorisation. Dans la suite de ce rapport, nous présenterons les résultats de nos expérimentations visant à comprendre cette problématique cruciale pour l'optimisation des supercalculateurs et clusters HPC.

4 Architectures et outils

Dans cette section, nous allons fournir une description détaillée des architectures et des logiciels utilisés pour nos tests. Cela permettra d'établir un cadre solide pour la compréhension de nos expérimentations et de leurs résultats. Nous aborderons les caractéristiques clés de chaque architecture, y compris les spécifications matérielles, les configurations de processeur, et les détails pertinents sur les logiciels.

4.1 Vectorisation : Instructions SIMD

Comme expliqué dans l'article [5], Les instructions SIMD permettent l'exécution simultanée de la même opération sur différents éléments d'un vecteur ou sur différents points de données. Le nombre d'opérations simultanées dépend de la taille des registres fournie par les processeurs. Intel a implémenté des instructions SIMD en virgule flottante depuis la fin des années 90 avec l'introduction des Streaming SIMD Extensions (SSE). Des registres de 128 bits (16 octets) étaient utilisés pour stocker 2 nombres à virgule flottante en double précision ou 4 nombres à virgule flottante en simple précision. Les Advanced Vector Extensions (AVX) sont apparues en 2010. La taille des registres a été doublée (256 bits) pour les opérations en virgule flottante. Cependant, les instructions SIMD en virgule flottante de 128 bits n'ont pas été étendues. Enfin, depuis le processeur Haswell (2013), les extensions AVX2 ont été introduites. Elles étendent la plupart des instructions SIMD en virgule flottante de 128 bits à 256 bits et ajoutent des instructions de multiplication-addition fusionnée (FMA). AVX-512 est une extension de 512 bits des opérations de 256 bits (pour les opérations en virgule flottante et les opérations entières). Elles sont implémentées dans les processeurs Intel Xeon Phi et Skylake depuis 2015.

(a) Scalar processor : one operation at a time

(b) SIMD processor : several simultaneous operations

FIGURE 1 – Difference between SIMD and scalar operations. Example : addition of two vectors

4.2 Architectures

Les expérimentations ont été essentiellement conduites sur la plateforme PlaFRIM [7]. La plateforme PlaFRIM, qui signifie "Plateforme Aquitaine de Calcul Intensif et de données pour la Recherche, l'Industrie et la Médecine", est une infrastructure de calcul haute performance (HPC) située en France, plus précisément à Bordeaux, en Nouvelle-Aquitaine. PlaFRIM propose un environnement de calcul avancé qui permet aux chercheurs et aux ingénieurs de réaliser des simulations complexes, de traiter de grands ensembles de données et de mener des projets de recherche nécessitant une puissance de calcul importante. Cette plateforme offre un accès à des supercalculateurs de pointe, à des clusters informatiques, ainsi qu'à des ressources de stockage et de gestion de données.

Parmi les clusters utilisés, on trouve Bora et Miriel, qui présentent des caractéristiques distinctes. Vous trouverez un résumé de la configuration de ces deux machines dans la table 1 :

	bora	miriel
CPU	Intel Cascade Lake	Intel Haswell
nb CPUs	2	2
nb coeurs par CPU	18	12
Mémoire	192 GB	128 GB

TABLE 1 – Configuration des deux clusters de plafrim : bora et miriel

Une distinction majeure entre ces deux clusters réside dans les niveaux de vectorisation disponibles, pour lesquels nous avons effectué des tests sur quatre niveaux différents. Le tableau 2 résume les niveaux de vectorisation disponibles sur les deux machines, Bora et Miriel :

	bora	miriel
SSE4_2	✓	✓
AVX	✓	✓
AVX2	✓	✓
AVX512	✓	×

TABLE 2 – Types de vectorisations disponibles sur bora et miriel

4.3 Logiciels

Cette partie se consacrera à la description des outils logiciels que nous avons utilisés pour mener à bien nos calculs. Nous allons fournir des informations générales sur chaque outil, notamment leurs fonctionnalités.

4.3.1 MKL

La bibliothèque principale employée dans les tests est l'Intel MKL (Math Kernel Library), une bibliothèque qui offre des routines mathématiques séquentielles et parallèles hautement optimisées. Les éléments mathématiques fondamentaux inclus sont les routines standardisées de l'algèbre linéaire, telles que BLAS, LAPACK, ScaLAPACK, BLACS, des solveurs creux et FFT..

4.3.2 Chameleon

Chameleon [1] est une bibliothèque codée en langage C. Elle procure un ensemble d'algorithmes qui exécute des opérations BLAS/LAPACK en exploitant les architectures modernes. Parmi ces algorithmes, on utilise la factorisation de Cholesky qui, pour une matrice réelle définie positive A, permet de trouver une matrice triangulaire inférieur L telle que $A = LL^T$. Cette factorisation se décompose en 4 noyaux de calculs : **gemm**, **trsm**, **syrk**, **potrf**.

4.3.3 Likwid

likwid [6], acronyme de "Like I Knew What I'm Doing" (comme si je savais ce que je faisais), est un ensemble d'outils et de bibliothèques open source conçu pour la mesure et l'analyse des performances sur les architectures matérielles modernes, en particulier dans le domaine du calcul haute performance (HPC). **likwid** met à disposition plusieurs commandes qui effectuent différentes tâches, dans notre cas, on s'est servi de `likwid-perfctr` qui permet de mesurer divers aspects de performances des machines tels que le nombre de flops, l'énergie consommé, le temps écoulé, etc.

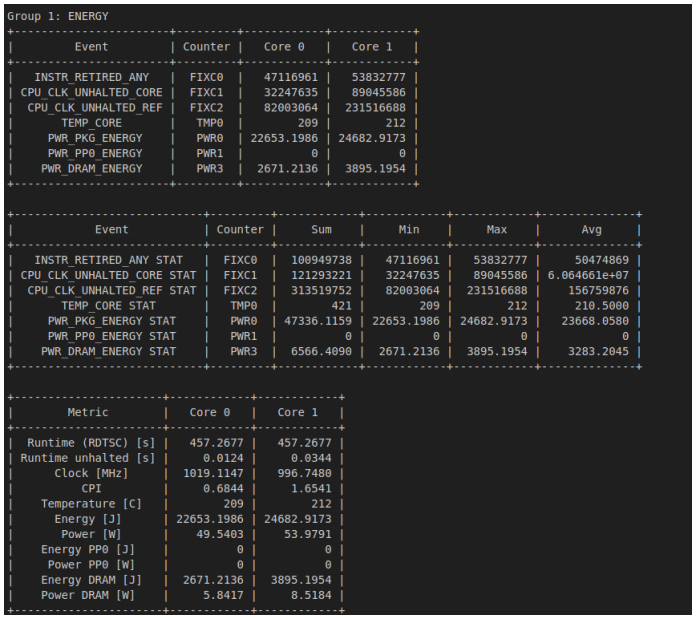
4.4 Méthodologie

Les tests que nous réalisons reposent sur l'outil LIKWID, plus précisément sur la commande « `likwid-perfctr` », qui nous permet d'effectuer un profilage de performance de la machine en collectant les résultats des compteurs des processeurs. Une ligne de commande typique pour lancer un test avec LIKWID ressemble à ceci :

```
likwid-perfctr -c 0,1 -g event1 -g event2 "./test arg1 arg2 ...."
```

Étant donné que nos tests s'effectuent sur des machines multiprocesseurs, nous utilisons l'option "-c" pour spécifier les processeurs sur lesquels nous souhaitons effectuer les mesures. L'option "-g" quant à elle nous permet de définir les événements que nous souhaitons capturer. Une liste complète des événements disponibles pour chaque type de processeur est disponible dans le dépôt officiel de LIKWID à l'adresse suivante : `https://github.com/RRZE-HPC/likwid/tree/master/groups`. Parmi les événements que nous utilisons pour nos tests, il y a notamment :

— ENERGY : c'est un groupe d'évènements qui permet de mesurer la puissance et l'énergie consommée par l'exécutable. Les résultats fournis sont présentés dans la figure 2. Comme



```
Group 1: ENERGY
+----------------------+---------+------------+------------+
|        Event         | Counter |   Core 0   |   Core 1   |
+----------------------+---------+------------+------------+
|   INSTR_RETIRED_ANY  |  FIXC0  |  47116961  |  53832777  |
| CPU_CLK_UNHALTED_CORE|  FIXC1  |  32247635  |  89045586  |
|  CPU_CLK_UNHALTED_REF|  FIXC2  |  82003064  | 231516688  |
|       TEMP_CORE      |   TMP0  |        209 |        212 |
|     PWR_PKG_ENERGY   |   PWR0  | 22653.1986 | 24682.9173 |
|     PWR_PP0_ENERGY   |   PWR1  |          0 |          0 |
|    PWR_DRAM_ENERGY   |   PWR3  |  2671.2136 |  3895.1954 |
+----------------------+---------+------------+------------+

+---------------------------+---------+-----------+------------+------------+-------------+
|           Event           | Counter |    Sum    |    Min     |    Max     |     Avg     |
+---------------------------+---------+-----------+------------+------------+-------------+
|   INSTR_RETIRED_ANY STAT  |  FIXC0  | 100949738 |   47116961 |   53832777 |     50474869 |
| CPU_CLK_UNHALTED_CORE STAT|  FIXC1  | 121293221 |   32247635 |   89045586 | 6.064661e+07 |
|  CPU_CLK_UNHALTED_REF STAT|  FIXC2  | 313519752 |   82003064 |  231516688 |    156759876 |
|       TEMP_CORE STAT      |   TMP0  |       421 |        209 |        212 |     210.5000 |
|     PWR_PKG_ENERGY STAT   |   PWR0  | 47336.1159| 22653.1986 | 24682.9173 |   23668.0580 |
|     PWR_PP0_ENERGY STAT   |   PWR1  |         0 |          0 |          0 |            0 |
|    PWR_DRAM_ENERGY STAT   |   PWR3  |  6566.4090|  2671.2136 |  3895.1954 |    3283.2045 |
+---------------------------+---------+-----------+------------+------------+-------------+

+----------------------+------------+------------+
|        Metric        |   Core 0   |   Core 1   |
+----------------------+------------+------------+
|  Runtime (RDTSC) [s] |   457.2677 |   457.2677 |
| Runtime unhalted [s] |     0.0124 |     0.0344 |
|      Clock [MHz]     |  1019.1147 |   996.7480 |
|          CPI         |     0.6844 |     1.6541 |
|    Temperature [C]   |        209 |        212 |
|      Energy [J]      | 22653.1986 | 24682.9173 |
|       Power [W]      |    49.5403 |    53.9791 |
|    Energy PP0 [J]    |          0 |          0 |
|     Power PP0 [W]    |          0 |          0 |
|    Energy DRAM [J]   |  2671.2136 |  3895.1954 |
|    Power DRAM [W]    |     5.8417 |     8.5184 |
+----------------------+------------+------------+
```

FIGURE 2 – Résultat de l'appel à likwid-perfctr avec le groupe d'événement ENERGY

illustré dans la figure 2. Le groupe d'événement ENERGY nous permet de déterminer plusieurs aspects énergétiques de l'exécution, notamment le temps d'exécution, l'énergie en joules ainsi que la puissance à la fois du processeur (package) et de la mémoire (DRAM) sur chaque socket ou processeur disponible sur la machine (sur les architectures mentionnées en section 4.2, il y en a deux).

— FLOPS_SP (simple précision) ou FLOPS_DP (double précision) : Ces ensembles d'événements permettent de compter le nombre d'opérations effectuées lors de l'exécution, que ce soit en simple ou en double précision. Un exemple de résultat est présenté dans la figure 3 : L'événement FLOPS_SP (ou FLOPS_DP) nous permet d'évaluer divers aspects de la performance, comme le nombre d'instructions effectuées et la vitesse en MFLOPS/s, que ce soit en mode séquentiel ou parallèle/vectorisé.

Chaque cas de test a été exécuté en boucle, avec un total de 20 itérations, pendant lesquelles nous avons collecté les résultats de chaque exécution. Par la suite, nous avons calculé une moyenne standard à partir des valeurs obtenues. Cette démarche nous a permis d'obtenir une mesure plus robuste et représentative des performances et des caractéristiques énergétiques de chaque cas de test, en réduisant l'impact des variations individuelles d'une exécution à l'autre.

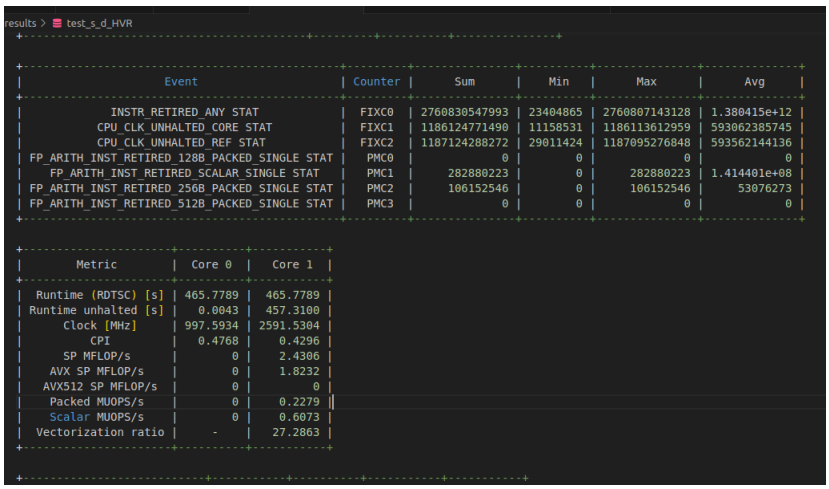


FIGURE 3 – Résultat de l'appel à likwid-perfctr avec l'événement FLOPS_SP

5 Expérimentations

Dans cette section, nous allons détailler les divers tests effectués, présenter les résultats obtenus, procéder à une analyse approfondie, et enfin, réaliser une synthèse des conclusions tirées.

5.1 Application compute-bound : Chameleon

Le premier cas de test consiste à réaliser un profil énergétique d'un ensemble de noyaux de calculs prédéfinis sur Chameleon. Les noyaux testés sont **gemm**, **trsm**, **syrk** et **potrf**. Chaque noyau ayant ces propres caractéristiques, on essaye de voir les différents comportements énergétiques en jouant sur des variantes de performances telles que la vectorisation et le nombre de cœurs.

5.1.1 Impact de la vectorisation sur la consommation énergétique

Afin d'évaluer l'impact de la vectorisation sur la consommation énergétique, notre approche consiste à comparer divers aspects énergétiques en variant le niveau de vectorisation, allant du moins intense SSE4_2 au plus avancé AVX512. Parmi ces aspects, deux se démarquent en tant que paramètres essentiels reflétant le rapport entre l'énergie consommée et les performances obtenues : l'efficacité énergétique en Gflops/j (gigaflops par joule), qui représente le nombre d'opérations effectuées par unité d'énergie consommée, et la puissance en watts (W). Pour déterminer le niveau de vectorisation lors des tests, nous utilisons la variable d'environnement de la bibliothèque mathématique Intel Math Kernel Library (MKL), MKL_ENABLE_INSTRUCTIONS.

Résultats

Les tests ont été effectués de manière séquentielle (utilisant un seul cœur) sur des matrices carrées denses de tailles allant de 800 à 12 800. Le choix de matrices de ces dimensions était crucial, car la mesure correcte ne pouvait être réalisée que si le temps d'exécution des tests était de l'ordre des secondes. En effet, la granularité des mesures d'énergie diffère de celle des mesures de temps, car les compteurs ne sont pas mis à jour assez fréquemment. On commence par étudier l'impact de la vectorisation sur la puissance consommé, les résultats obtenus, sur le cluster bora sont regroupés dans la figure 4. Il est important de noter que nous ne présentons que les résultats des calculs avec l'opération **gemm** et sur le cluster Bora, car la tendance est similaire pour les différents cas de tests, le reste des résultats sont présentés dans l'annexe.

textbf {gemm}ïĂžgraphe_double_W1.png

FIGURE 4 – Puissance consommée du noyau **gemm** : En double precision

Il est observable, dans la figure 4 que plus la vectorisation est intense, plus la puissance consommée par l'exécution des calculs augmente. Cette tendance met en évidence une corrélation directe entre le niveau de vectorisation et la consommation de puissance.

Pour évaluer l'incidence de la précision sur la consommation de puissance, une comparaison a été réalisée en effectuant le même test à la fois en simple et en double précision. Les résultats de cette comparaison sont présentés dans la figure 5.

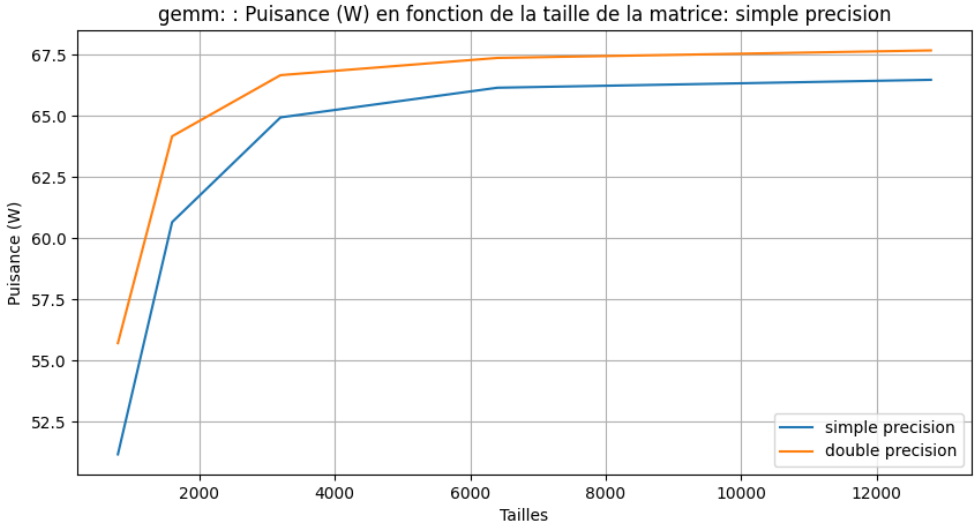


FIGURE 5 – Impact de la précision sur la puissance consommée

On remarque que la précision des calculs n'a pas d'impact significatif sur la puissance consommée, elle se stabilise à ~70 W pour les deux précisions. Cette observation suggère que, dans ce contexte particulier, le choix entre la précision simple et double n'influence pas de manière significative la consommation d'énergie.

Le deuxième aspect énergétique important dans notre étude est l'efficacité énergétique en GFLOPS/J, cette métrique nous permettra de décrire plus précisément la corrélation performance-énergie. La figure 6 résume les résultats obtenus sur bora pour le noyau de calcul **gemm**.

textbf {gemm}ïĂžgraphe_simple_Gflop_J_1.png

FIGURE 6 – Efficacité énergétique des calculs sur le noyau **gemm** : en simple precision

Comme illustré dans la figure 6, nous pouvons observer une tendance notable en ce qui concerne l'efficacité énergétique, avec une mise en évidence particulière de la vectorisation la plus puissante. Par exemple, avec AVX512, l'efficacité énergétique atteint un maximum de 1.8 Gflops/J, puis elle diminue à 0.9 Gflops/J pour la vectorisation AVX2, et ainsi de suite. Cette réduction d'efficacité énergétique de l'ordre de 2 entre les différents types de vectorisation s'explique principalement par la taille des registres associés à ces instructions SIMD. En effet, la taille des registres utilisés par AVX est de 128 bits, tandis qu'AVX2 utilise des registres de

256 bits, et AVX512 déploie des registres de 512 bits, donc on peut retenir que plus la taille des registres augmente, plus la performance est élevée.

Pour observer l'impact de la précision sur la consommation énergétique, on compare les résultats d'un même test, en simple et en double précision. La figure 7 présente une comparaison au niveau efficacité énergétique entre la double et la simple précision.

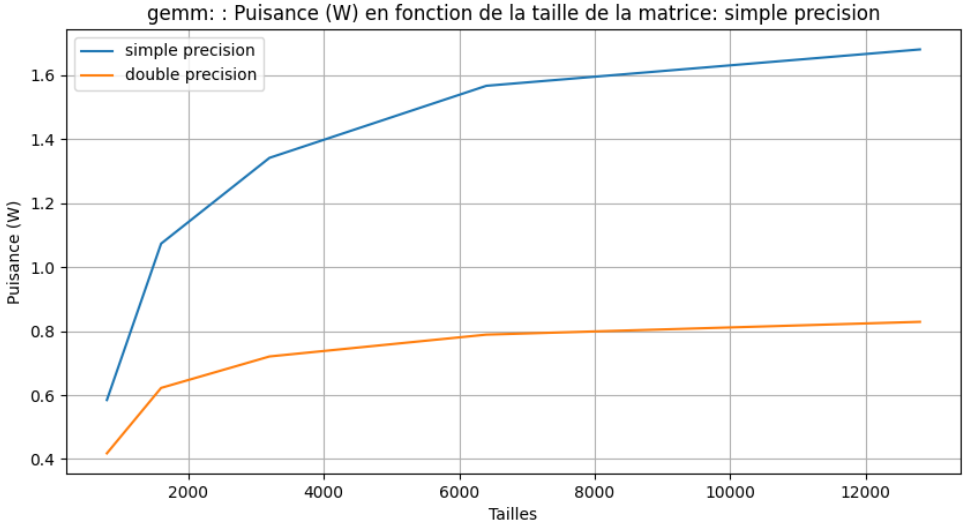


FIGURE 7 – Impact de la précision sur l'efficacité énergétique

Comme le montre clairement la figure 7, il est évident que l'efficacité énergétique est nettement plus élevée en ce qui concerne la précision simple par rapport à la précision double. Cette observation met en évidence une différence significative dans la relation entre les performances et la consommation d'énergie en fonction du niveau de précision des calculs. En optant pour la précision simple, il est possible d'obtenir un niveau de performance considérable tout en consommant moins d'énergie par opération par rapport à la précision double.

Discussion

En résumé, il est clair que le niveau de vectorisation utilisé exerce une influence significative sur les divers aspects énergétiques de l'exécution, en fonction de l'ordre croissant de parallélisme. Plus précisément, l'intensité de la vectorisation affecte considérablement des paramètres tels que l'efficacité énergétique, avec une tendance croissante à mesure que la vectorisation devient plus intense. D'autre part, il est à noter que le niveau de précision utilisé dans les calculs a un impact significatif sur l'efficacité énergétique, tout en maintenant la puissance consommée relativement stable.

Une conclusion qu'on peut tirer des résultats obtenus est que, dans le cadre de ce test spécifique, la consommation énergétique est étroitement liée au temps d'exécution. En d'autres termes, plus le temps nécessaire à l'exécution est réduit, plus l'efficacité énergétique est élevée. Cette relation découle en grande partie de l'optimisation poussée et de l'intensité des calculs effectués sous Chameleon. Cependant, il serait également pertinent d'explorer davantage cette corrélation en menant des tests sur une application axée sur les accès mémoire (memory-bound) plutôt que sur les calculs. Cela permettrait de déterminer si les résultats obtenus restent cohérents et si la relation entre la consommation énergétique et le temps d'exécution demeure constante dans des scénarios différents.

5.2 Application memory-bound : SPMV

La multiplication matrice-vecteur creuse (SPMV) est une opération fondamentale en informatique scientifique, largement utilisée dans des domaines tels que la simulation numérique, l'apprentissage automatique et la recherche en sciences des données. Cette opération est considérée comme "memory bound", ce qui signifie qu'elle est principalement limitée par la vitesse de la mémoire plutôt que par la puissance de calcul du processeur. Lors d'une opération SPMV, un vecteur est multiplié par une matrice creuse. Cette opération nécessite un accès fréquent à la mémoire pour récupérer les données de la matrice. Étant donné que l'on travaille avec des matrices creuses, l'accès à un élément de la matrice nécessite d'abord de déterminer sa position via les tableaux d'indices, puisque seuls les éléments non nuls de la matrice sont stockés. Cela se produit quel que soit le format de stockage utilisé. Par conséquent, le ratio entre les opérations de calcul et le débit mémoire est déséquilibré. Pour faciliter cette procédure, les matrices sont stockées sous des formes spécifiques, soit en COO (Coordinate Format) ou en CSR (Compressed Sparse Row), comme illustré dans la figure 8.

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 6 \\ 0 & 0 & 0 & 0 & 7 \end{bmatrix}$$

$$\text{CSR :} \quad \text{row\_ptr} = \begin{bmatrix} 0 & 2 & 4 & 4 & 6 & 7 \end{bmatrix}, \quad \text{col\_indices} = \begin{bmatrix} 0 & 1 & 3 & 4 & 3 & 4 \end{bmatrix}$$

$$\text{values} = \begin{bmatrix} 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

FIGURE 8 – Représentation condensée de la matrice au format CSR.

La figure présente une représentation matricielle condensée au format CSR (Compressed Sparse Row) pour une matrice creuse A. Dans cette forme, la matrice est décomposée en trois vecteurs distincts : row_ptr, col_indices et values. Le vecteur row_ptr indique le début de chaque ligne de la matrice, le vecteur col_indices stocke les indices des colonnes des éléments non nuls, et le vecteur **values** contient les valeurs correspondantes de ces éléments non nuls. Cette représentation est utilisée pour économiser de l'espace mémoire en stockant uniquement les éléments non nuls de la matrice, ce qui est essentiel dans le traitement de matrices creuses de grande taille couramment rencontrées dans des domaines tels que la simulation numérique et l'apprentissage automatique.

Nous avons réalisé les tests de l'opération SPMV en utilisant un code source disponible en ligne [1]. Par la suite, nous avons effectué des mises à jour des fonctions utilisées et modifié le code pour pouvoir extraire les données depuis des fichiers de test au format mtx (Matrix Market). Cela était nécessaire car les fonctions de la bibliothèque MKL utilisées dans le code initial étaient devenues obsolètes, et il manquait une partie du code pour la lecture et l'organisation des données provenant des fichiers de test. Il est important de noter que ces fichiers contiennent des matrices creuses de grande taille, pouvant comporter jusqu'à 265 millions d'éléments non nuls et atteindre des dimensions allant jusqu'à 2 millions.

1. `http://berenger.eu/blog/cc-sparse-mkl-examples-c00-csr-dia-bcsr-gemv-and-conversions/`

Pendant mon stage, nous avons rencontré quelques défis. Tout d'abord, la conversion du code a pris plus de temps que prévu. Ensuite, l'utilisation de **likwid** pour distinguer la simple précision de la double précision s'est avérée problématique, ce qui a généré des résultats incohérents. Actuellement, nous travaillons toujours à résoudre ces problèmes, ce qui a retardé la finalisation de nos travaux sur l'opération SPMV.

6 Conclusion Générale

La réalisation de ce stage a permis d'explorer en profondeur la relation complexe entre la performance et la consommation d'énergie dans le domaine du calcul haute performance (HPC). Les résultats de nos expérimentations ont montré que cette relation est fortement influencée par plusieurs facteurs, notamment la vectorisation, la précision des calculs et le type d'application (compute-bound ou memory-bound).

En ce qui concerne la vectorisation, nos tests ont démontré que, pour les applications cpu-intensive, l'intensité de la vectorisation a un impact significatif sur la consommation d'énergie et l'efficacité énergétique. Plus la vectorisation est intense, plus la puissance consommée augmente, mais l'efficacité énergétique (mesurée en GFLOPS/J) s'améliore également. Cela souligne l'importance de choisir judicieusement le niveau de vectorisation en fonction des exigences de performance et d'efficacité énergétique d'une application spécifique.

En ce qui concerne la précision des calculs, nos résultats ont montré que l'utilisation de la précision simple peut permettre d'obtenir des performances élevées tout en consommant moins d'énergie par opération par rapport à la précision double. Cependant, le choix de la précision doit être fait avec prudence en fonction des besoins réels de l'application, car la précision réduite peut entraîner une perte de précision dans les résultats.

Enfin, notre étude a également mis en évidence que la relation entre la consommation d'énergie et le temps d'exécution varie en fonction du type d'application. Dans le cas des applications orientées calcul intensif, où les opérations de calcul prédominent, une réduction du temps d'exécution a tendance à améliorer l'efficacité énergétique de manière générale. Cependant, dans le cas des applications centrées sur l'accès à la mémoire, la relation entre la consommation d'énergie et le temps d'exécution peut devenir plus complexe.

Il convient de noter que notre étude n'a pas permis de conclure de manière définitive sur les applications à forte demande mémoire, car les tests sur l'opération SPMV n'ont pas abouti. Des recherches futures viseront à poursuivre l'évaluation de ces applications memory-bound, en examinant leur comportement énergétique lorsque des paramètres tels que la vectorisation et la précision sont modifiés.

En conclusion, cette étude met en lumière l'importance de considérer à la fois les performances et l'efficacité énergétique lors de la conception et de l'optimisation d'applications HPC. Les décisions relatives à la vectorisation, à la précision des calculs et au type d'application doivent être prises en fonction des objectifs spécifiques de l'application et des contraintes énergétiques. En fin de compte, l'optimisation de la consommation d'énergie dans le domaine du HPC est un défi complexe qui nécessite une approche équilibrée pour atteindre un compromis optimal entre les performances et l'efficacité énergétique.

Remerciement

Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by **Inria**, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see `https://www.plafrim.fr`).

Références

[1] Emmanuel Agullo et al. “Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model”. In : *IEEE Transactions on Parallel and Distributed Systems* (2017). In press. DOI : 10.1109/TPDS.2017.2766064. URL : https://hal.archives-ouvertes.fr/hal-01618526.

[2] Emmanuel Agullo et al. “Faster, Cheaper, Better – a Hybridization Methodology to Develop Linear Algebra Software for GPUs”. In : *GPU Computing Gems*. Sous la dir. de Wen-mei W. Hwu. T. 2. ID : inria-00547847. Morgan Kaufmann, 2010.

[3] Cyrille Bonamy Laurent Lefèvre Laurent Bourgès. “Impact des langages de programmation sur la performance énergétique des applications de calcul scientifique : un challenge ?” In : (). Accessed on September 28, 2023. URL : https://unif.fr/wp-content/uploads/2022/09/IMPACT-langages-de-programmation-sur-performances-energetiques_1963_20220315_144218.pdf.

[4] Jack Dongarra, Laura Grigori et Nicholas J. Higham. “Numerical algorithms for high-performance computational science”. In : *Phil. Trans. R. Soc. A* 378 (2020), p. 20190066. DOI : 10.1098/rsta.2019.0066. URL : http://doi.org/10.1098/rsta.2019.0066.

[5] Amina Guermouche et Anne-Cécile Orgerie. “Thermal design power and vectorized instructions behavior”. In : *Concurrency and Computation : Practice and Experience* 34.2 (2022), e6261. DOI : https://doi.org/10.1002/cpe.6261. eprint : https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.6261. URL : https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.6261.

[6] *LIKWID*. Accessed on September 28, 2023. URL : https://hpc.fau.de/research/tools/likwid/.

[7] *plafrim*. URL : https://www.plafrim.fr/.

Annexe

Toutes les données brutes des tests ainsi que le code utilisé sont disponibles sur le lien GitHub suivant : https://github.com/gswn2/precision.

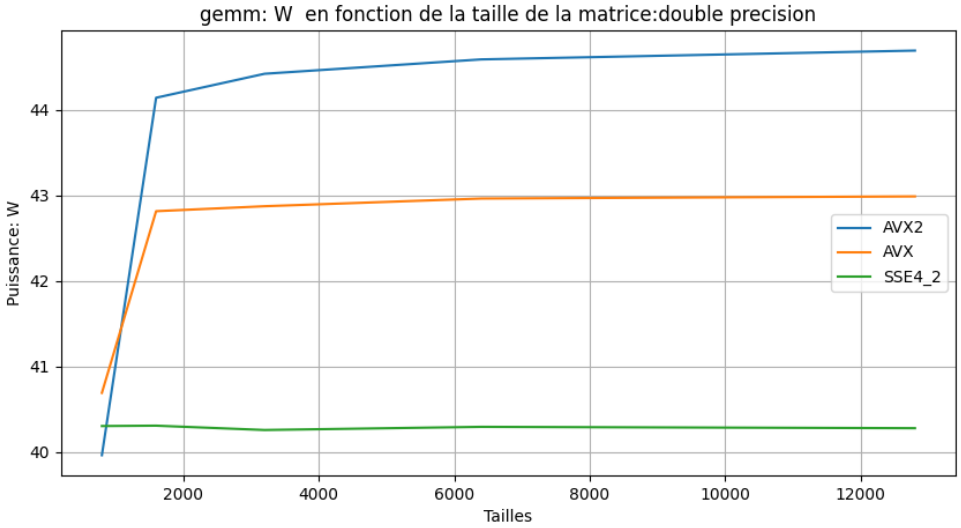


Figure 9 – Puissance consommée du noyau **gemm** sur miriel : En double precision

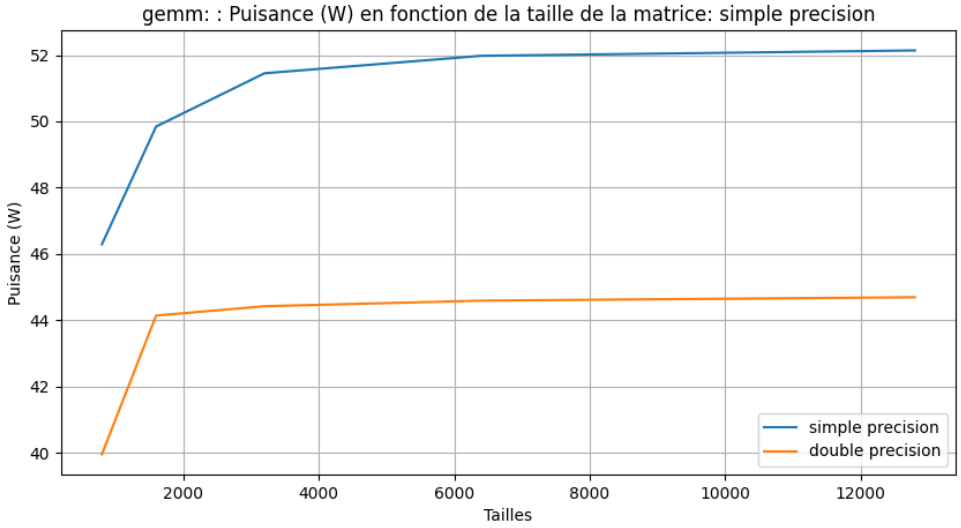


FIGURE 10 – Impact de la précision sur la puissance consommée sur miriel

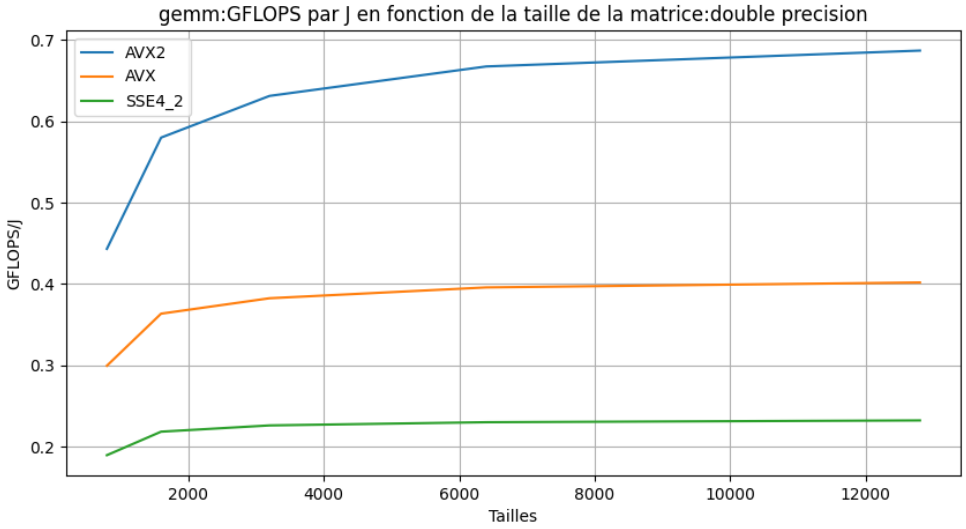


FIGURE 11 – Efficacité énergétique des calculs sur le noyau **gemm** sur miriel : en simple precision

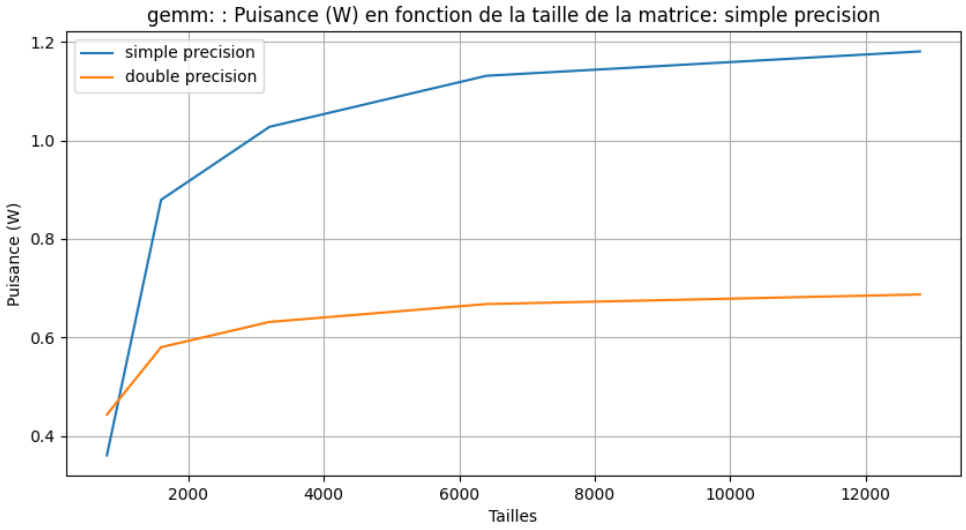


FIGURE 12 – Impact de la précision sur l'efficacité énergétique sur miriel