Département Informatique

Mémoire intermédiaire
Travaux d'étude et de recherche

Sujet :
Impact de la parallélisation d'algorithmes sur la consommation énergétique des architectures parallèles

Proposé et encadré par : Sylvain JUBERTIE

Réalisé par :
Youenn LEBRAS, Romain PROU, Issam RAIS
et Joris RUBAGOTTI

Année : 2013-2014

Table des matières

1 Résumé du projet

Notre projet, proposé par M. JUBERTIE, nous entraîne dans l'étude de l'impact de la parallélisation d'algorithmes sur la consommation énergétique des architectures parallèles.

Les machines actuelles, du smartphone aux supercalculateurs, possèdent des processeurs multi-cœurs. Afin d'augmenter les performances le développeur doit s'appliquer à développer des codes s'exécutant en parallèle. Cependant, la parallélisation n'est pas très répandue dans les périphériques nomades. Les nombreux cœurs servent généralement à exécuter plusieurs tâches distinctes.

Pour juger de l'impact du parallélisme sur consommation énergétique, certains choix s'offrent à nous : soit faire fonctionner un seul processeur à sa fréquence maximale, soit utiliser plusieurs cœurs à fréquence réduite.

Pour cela de nombreux algorithmes parallèles seront implémentés et exécutés sur différentes architectures. Une comparaison entre des codes écrits en Java et C/C++ pour Android pourra également être envisagée afin d'étudier l'impact du langage sur les performances et la consommation.

2 Introduction

2.1 Le domaine

Du smartphone aux supercalculateurs, nos outils technologiques ne cessent d'évoluer. Nous arrivons à stade où les appareils mobiles, ainsi que les machines de calculs, possèdent des processeurs multi-cœurs, leur but premier est de réduire le temps de calculs des différents algorithmes que l'on souhaite exécuter. En parallèle la consommation électrique de ces technologies augmente considérablement.
Ce qui met en avant certaines problématiques auxquelles nous tenterons d'apporter un élément de réponse.

2.2 La problèmatique

Est-il vraiment utile de paralléliser du code pour gagner du temps de calcul et ainsi réduire la consommation énergétique ? Autrement dit, le rapport temps / énergie consommée est-il négligeable, entraîne-t-il, au contraire, une surconsommation énergétique ?

Pour répondre à ces questions nous devrons d'abord nous demander comment exploiter les différents types d'architectures, afin de réaliser des calculs performants et peu intensifs.

Dans un premier temps, nous réaliserons des mesures de performances et de consommations sur des processeurs nomades en utilisant des codes séquentiels et parallèles (notamment grâce aux bibliothèques OpenMP [1] et Threads [2]). Nous pourrons ainsi donner une première approche du problème ; à savoir s'il est préférable d'utiliser pleinement un cœur avec sa fréquence maximal ou s'il faut privilégier un code réparti sur plusieurs cœurs avec des fréquences réduites. Et ainsi établir un ratio performance / gain énergétique.

Ensuite, nous ferons une analogie des résultats obtenus avec les tests effectués sur des architectures plus "traditionnelles" tels que des ordinateurs.

Pour finir, nous mettrons en relief les différentes corrélations que nous avons pu voir durant ce projet ; nous tenterons ainsi d'établir un modèle sur les coûts de la parallélisation et l'impact sur la consommation énergétique qui en découle.

1. OpenMP : Open Multi-Processing est un ensemble de modules permettant la réalisation d'applications utilisant le multi-threading sur C/C++ ou Fortran.
2. Un thread est similaire à un processus car tous deux représentent l'exécution d'un ensemble d'instructions du langage machine d'un processeur.

2.3 Les objectifs

Pour répondre à cette problématique nous devrons nous concentrer sur deux points :

— Premièrement, le développement de code de calculs simples et d'en produire des versions optimisées séquentielles et parallèles (grâce notamment aux bibliothèques OpenMP ou threads).

— Deuxièmement, effectuer des tests sur des tablettes/smartphones et également sur des PC traditionnels afin d'en déduire une correlation entre la parallélisation d'un code et les effets qui en découlent ; tels que le temps de calcul gagné et la consommation énergétique engendrée.

C'est pourquoi un état de l'art de ce domaine a été essentiel.

3 Analyse de l'existant

L'état de l'art

De nombreux articles parlent de près ou de loin du sujet que nous allons traiter.
La liste suivante est une liste non exhaustive des articles ainsi qu'un résumé de leurs contenus.

"Parallelism Level Impact on Energy Consumption in Reconfigurable Devices" :[1]
Impact de l'implémentation d'algorithmes parallèles sur la performance et la consommation énergétique. Il nous montre que la consommation énergétique diminue quand le parallélisme augmente.

"Analytical Modeling And Simulation Of The Energy Consumption Of Independent Tasks" :[2]
Comment la consommation d'energie peut être analysée, capturée par un modèle de consommation d'énergie analytique. Traite également de l'impact de la modification de la fréquence des processeurs sur la consomation d'énergie.

"Modeling the Energy Consumption for Concurrent Executions of Parallel Tasks" :[3]
Explore la consomation d'énergie des taches parralèles executé de façon concurentes.

"On the Interplay of Dynamic Voltage Scaling and Dynamic Power Management in Real-Time Embedded Applications" :[4]
Mise en evidence des differences entre deux techniques connues de gestion de l'energie pour les systèmes embarqués.

"Prediction Models for Multi-dimensional Power-Performance Optimization on Many Cores" :[5]
Présentation d'un modèle pour prédire les performances lorsque sont appliquées simultanement de multiples techniques de sauvegarde d'energie.

"An Analysis of Efficient Multi-Core Global Power Management Policies : Maximizing Performance for a Given Power Budget" :[6]
Ce papier évoque des techniques pour le contrôle et la gestion dynamique de l'énergie sur les systèmes multicoeurs.

"On the Interplay of Parallelization, Program Performance, and Energy Consumption" :[7]
Développe le lien etroit entre le parallelisme, la performane et la consomation énergetique d'un programme.

"Hybrid MPI/OpenMP Power-Aware Computing" :[8]

Présente un modèle hybrid MPI[1]/OpenMP de programmation économe en energie.

"Energy-Aware Computing for Android[2] Platforms" :[9]
Présentation de la gestion de l'alimentation énergetique des systemes android.

"Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones" : [10]
Présentation de PowerBooster et PowerTutor, techniques de construction de modèle energetique pour smartphone, developpées par Google.

"Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era" :[11]
Extension de la loi d'Amdahl[3] pour l'écoenergétique pour le développement multicoeurs.

"Advanced Android Power Management and Implementation of Wakelocks" :[12]
Explique comment android gère l'utilisation batterie et pourquoi Linux à été choisie comme base pour android.

"An Analysis of Power Consumption in a Smartphone" :[13]
Approche visant à améliorer la consomation et la gestion énergetique sur smartphone.

"Android Power Management : Current and Future Trends" :[14]
Présente de nombreuses recherches traitant de la diminution de la consomation énergetique

"Energy Consumption Modeling for Hybrid Computing" :[15]
Montre qu'un programme utilisant le parallelisme utilise moins d'énergie qu'un programme non parallèle.

"Power-Aware Speedup" :[16]
Présentation d'un modèle permettant de juger de la performance mais aussi de la consomation énergetique des algorithmes parallèles

1. MPI (The Message Passing Interface) est une norme définissant une bibliothèque de fonctions, utilisable avec les langages C, C++ et Fortran. Elle permet d'exploiter des ordinateurs distants ou multiprocesseur par passage de messages.
2. Android est un système d'exploitation pour smartphones
3. La loi d'Amdahl, énoncée par Gene Amdahl en 1967, exprime le gain de performance qu'on peut attendre d'un ordinateur en améliorant une composante de sa performance.

4 Besoins non fonctionnels

Notre domaine d'action se porte sur les appareils mobiles sous Android. Par extension notre hypothèse sera testée sur des ordinateurs.

Nous n'avons aucune fiabilité sur la réussite de notre projet. Ce dernier étant un domaine de recherche, le but est d'affirmer ou de réfuter l'hypothèse énoncée précédemment.

Les algorithmes implémentés :

— Un bassin versant[17] est une aire délimitée par des lignes de partage des eaux, à l'intérieur des quelles toutes les eaux tombées alimentent un même exutoire : cours d'eau, lac, mer, océan, etc. Une ligne de partage des eaux ce confond très souvent avec une ligne de crête.

Chaque bassin versant se subdivise en un certain nombre de bassins élémentaires (parfois appelés « sous-bassin versant ») correspondant à la surface d'alimentation des affluents se jetant dans le cours d'eau principal.

— Il s'agit de la façon la plus fréquente de multiplier des matrices[18] entre elles.
En algèbre linéaire, une matrice A de dimensions m lignes et n colonnes (matrice m×n) représente une application linéaire f d'un espace de dimension n vers un espace de dimension m. Une matrice colonne V de n lignes est une matrice n×1, et représente un vecteur v d'un espace vectoriel de dimension n. Le produit A×V représente f(v).
Si A et B représentent respectivement les applications linéaires f et g, alors A×B représente la composition des applications fog.
Cette opération est utilisée notamment en mécanique lors des calculs de torseur statique, ou en informatique pour la matrice d'adjacence d'un graphe.
Le produit de deux matrices ne peut se définir que si le nombre de colonnes de la première matrice est le même que le nombre de lignes de la deuxième matrice, c'est-à-dire lorsqu'elles sont de type compatible.

— On peut trouver une valeur approchée de π [19] de façon empirique, en traçant un cercle, puis en mesurant son diamètre et sa circonférence, puis en divisant la circonférence par le diamètre. Une autre approche géométrique, attribuée à Archimède, consiste à calculer le périmètre Pn d'un polygone régulier à n côtés et à mesurer le diamètre d de son cercle circonscrit, ou celui de son cercle inscrit. Plus le nombre de côtés du polygone est grand, meilleure est la précision obtenue pour la valeur de π.
Archimède a utilisé cette approche en comparant les résultats obtenus par la formule en utilisant deux polygones réguliers ayant le même nombre de côtés, pour

lesquels le cercle est pour l'un circonscrit et pour l'autre inscrit. Il a réussi, avec un polygone à 96 côtés, à déterminer que $3 + 10/71 < \pi < 3 + 1/7$.
On peut également obtenir des valeurs approchées de π en mettant en œuvre des méthodes plus modernes. La plupart des formules utilisées pour calculer π se basent sur la trigonométrie et le calcul intégral. Cependant, certaines sont particulièrement simples, comme la formule de Leibniz.

— l'histogramme représente la distribution des intensités d'une matrice.

Langage d'implémentation :

Pour effectuer nos tests nous avons opté pour écrire un code natif, c'est-à-dire du C/C++ pour que le code soit portable autant sur appareils mobile que sur nos ordinateurs, en changeant un minimum l'implémentation.

Tests de validation :

Nous ne pouvons pas réellement prévoir de tests de validation, étant donné que nous ne savons pas encore quel genre de résultat nous allons obtenir ; nous ne pourrons seulement valider nos tests que s'ils convergent vers le même résultat.

Description des problèmes associés :

Tout algorithme n'est pas parallélisable. En effet, induire du parallélisme peut ralentir le temps d'exécution d'un algorithme. En d'autres termes, un même algorithme peu être plus efficace en séquentielle qu'en parallèle.

Produire du code parallèle pousse le développeur à avoir une vision différente des algorithmes. Le but du développeur ici est d'avoir le meilleur gain possible. Autrement dis le temps d'exécution d'un programme parallèle doit avoir un réel facteur de division du temps en fonction du nombre de coeurs choisie.

5 Besoins fonctionnels

5.1 Le but

Le but de ce projet est d'établir l'utilité de la parallélisation dans un code, en vue de réduire la consommation énergétique d'une batterie d'un appareil mobile, par extension la consommation sur un ordinateur fixe ou portable.

Les batteries d'appareils mobiles sont de plus en plus mal menées par les nouvelles applications qui consomment de plus en plus de batterie, sans que ces dernières n'évoluent. Il faut donc trouver d'autres solutions pour limiter la consommation excessive d'énergie.

En parallèle les processeurs d'appareils mobiles évoluent pour avoir de plus grandes cadences de calculs et être de plus en plus nombreux dans nos derniers appareilles multimédia.

Par exemple, le Galaxy S4 de Samsung propose un Quad-core A15 (à 1.6GHz) plus un Quad-core A7 (à 1.2GHz). Il serait donc dommage de ne pas exploiter cette puissance de calcule pour nos applications, tout en gardant un oeil sur la consomation énèrgetique. Les mêmes conceptes et même problématiques s'appliquent aux ordinateurs .

5.2 L'application

L'idée est de déterminer si il vaut mieux effectuer un calcul sur un processeur avec une grande cadence de calcul ou de distribuer le calcul sur plusieurs processeurs plus lent, mais qui par conséquent consomme moins d'énergie.

En effet un processeur, cadencé à 4GHz, consommerait plus que quatre processeurs, cadencés à 1Ghz, pourtant le temps de calcul devrait être sensiblement le même.

Pour vérifier cette hypothèse nous allons réaliser différents tests avec différents calculs, sur plusieurs architectures ; et ainsi déterminer si nous consommerions moins d'énergie en réduisant la fréquence du processeur dynamiquement et en répartissant les calculs sur les différents processeurs.

En effet nous cherchons à déterminer si en distribuant un code la consommation énergétique diminue (sachant que la consommation énergétique augmente de façon exponentielle).

Pour moduler la fréquence et le voltage des appareils mobiles Android, nous nous sommes basé sur l'application AnTUTU[20], qui nous permet de régler de façon simple la

fréquence des différents processeurs disponibles sur nos appareils.

Des alternatives à AnTUTU existent, par exemple, Voltage Control[21] qui réalise la même chose.

Pour vérifier la consommation de batterie nous utilisons Battery Monitor[22], cette application trace une courbe de la consommation batterie en direct, mais nous oblige à garder l'application ouverte en tache de fond ce qui utilisera aussi un peu de batterie et donc nos résultat seront légérement faussé mais la consommation engendrée par cette application devrait être négligeable dans nos résultats.

Nous utiliserons aussi Battery Log[23] qui écrit dans un fichier les informations de la batterie à un instant T et l'intervalle entre deux temps est modulable.

En ce qui concerne nos tests sur ordinateur nous travaillons sous linux avec l'outil CPUFreq-selector[24] qui nous permet, comme AnTUTU sous Android, de moduler la fréquence des processeurs.

Ces outils nous permettrons d'évaluer les différents résultats obtenus durant nos tests.

6 Résultats de tests

Les outils / Code parallèle

Il n'existe pas beaucoup d'outils pour paralléliser du code. Nous avons dû faire une recherche sur ce qui existe et utilisable sur Android (cf : 5-Besoins fonctionnels).

Pour obtenir des gains de performance nous nous sommes penchés sur les librairies de parallèlisme classique. Parmi celles ci nos tests ont montrés qu'il était possible d'utiliser des technologies comme les Threads C++11, OpenMP et NEON [1] pour la partie Android. En plus de ces technologies, les ordinateurs classiques nous permettent d'utiliser des technologies comme SSE [2], AVX [3], ou encore CUDA [4].

L'estimation des performances énergétiques

Le but est de tester la consommation énergetique du parallélisme sur Android et sur PC avec les différentes technologies exploitables.

Nos tests consistent à exécuter des algorithmes où le parallélisme s'applique de façon optimale. Pour ce, des algorithmes où le parallélisme est maximal serait optimale, c'est-à-dire où chaque processeurs travailleraient sur des tailles de données équivalentes (cf : Non Fonctionnelle).

1. L'ARM NEON également appelé Advanced SIMD ou encore « MPE » (de l'anglais media processing engine, littéralement « moteur de calcul de médias ») est une unité de calcul de type SIMD(Single Instruction on Multiple Data), accélérant les calculs

2. Streaming SIMD Extensions, généralement abrégé SSE, est un jeu de 70 instructions supplémentaires pour microprocesseurs x86, le fonctionnement est de type SIMD.

3. Advanced Vector Extensions, est une extension des instructions SSE et s'appliquent sur des registres deux fois plus grand.

4. CUDA (Compute Unified Device Architecture) est une technologie de GPGPU (General-Purpose Computing on Graphics Processing Units), c'est-à-dire qu'un processeur graphique (GPU) est utilisé pour exécuter des calculs généraux habituellement exécutés par le processeur central (CPU).

7 Planning

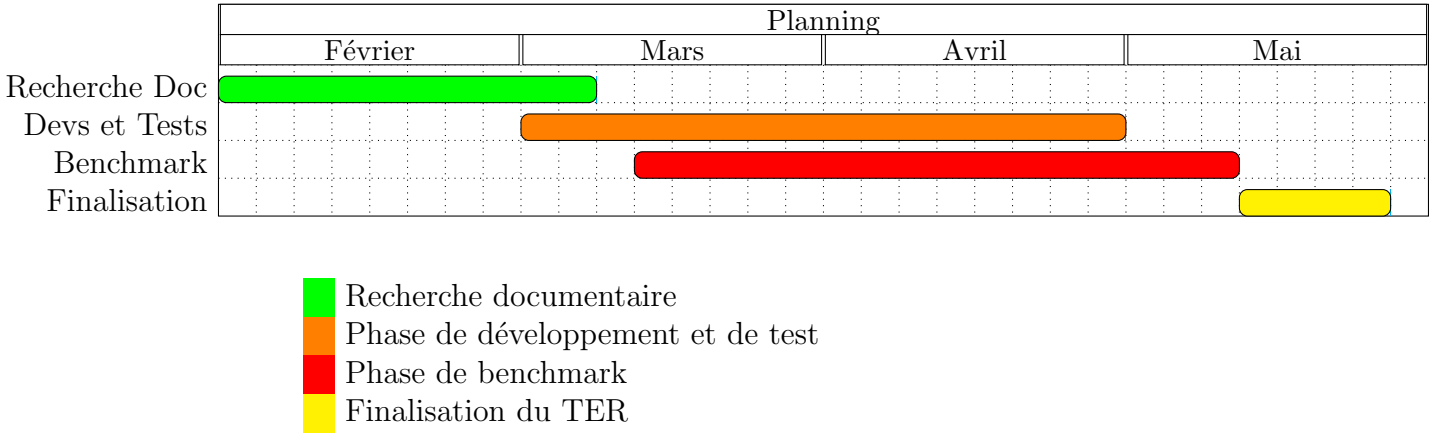

FIGURE 7.1 – Planning de réalisation du projet

Bibliographie

[1] R. Bonamy, D. Chillet, O. Sentieys, and S. Bilavarn, "Parallelism level impact on energy consumption in reconfigurable devices," *SIGARCH Comput. Archit. News*, vol. 39, pp. 104–105, Dec. 2011.

[2] T. Rauber and G. Rünger, "Analytical modeling and simulation of the energy consumption of independent tasks," in *Proceedings of the Winter Simulation Conference*, WSC '12, pp. 245 :1–245 :13, Winter Simulation Conference, 2012.

[3] T. Rauber and G. Rünger, "Modeling the energy consumption for concurrent executions of parallel tasks," in *Proceedings of the 14th Communications and Networking Symposium*, CNS '11, (San Diego, CA, USA), pp. 11–18, Society for Computer Simulation International, 2011.

[4] V. Devadas and H. Aydin, "On the interplay of dynamic voltage scaling and dynamic power management in real-time embedded applications," in *Proceedings of the 8th ACM International Conference on Embedded Software*, EMSOFT '08, (New York, NY, USA), pp. 99–108, ACM, 2008.

[5] M. Curtis-Maury, A. Shah, F. Blagojevic, D. S. Nikolopoulos, B. R. de Supinski, and M. Schulz, "Prediction models for multi-dimensional power-performance optimization on many cores," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, (New York, NY, USA), pp. 250–259, ACM, 2008.

[6] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies : Maximizing performance for a given power budget," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, (Washington, DC, USA), pp. 347–358, IEEE Computer Society, 2006.

[7] S. Cho and R. G. Melhem, "On the interplay of parallelization, program performance, and energy consumption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, pp. 342–353, Mar. 2010.

[8] D. Li, B. R. Supinski, M. Schulz, K. Cameron, and D. S. Nikolopoulos, "Hybrid mpi/openmp power-aware computing," in *In Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on (April 2010*, p. 12.

[9] H.-c. Chang, A. R. Agrawal, and K. W. Cameron, "Energy-Aware Computing for Android* Platforms," 2012.

[10] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES/ISSS '10, (New York, NY, USA), pp. 105–114, ACM, 2010.

[11] D. H. Woo and H.-H. S. Lee, "Extending amdahl's law for energy-efficient computing in the many-core era," *Computer*, vol. 41, no. 12, pp. 24–31, 2008.

[12] M. B. Motlhabi., "Advanced android power management and implementation of wakelocks," 2013.

[13] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, (Berkeley, CA, USA), pp. 21–21, USENIX Association, 2010.

[14] S. K. Datta, C. Bonnet, and N. Nikaein, "Android power management : Current and future trends," in *Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on*, pp. 48–53, IEEE, June 2012.

[15] A. Marowka, "Energy consumption modeling for hybrid computing," in *Proceedings of the 18th International Conference on Parallel Processing*, Euro-Par'12, (Berlin, Heidelberg), pp. 54–64, Springer-Verlag, 2012.

[16] R. Ge and K. W. Cameron, "Power-aware speedup.," in *IPDPS*, pp. 1–10, IEEE, 2007.

[17] "Bassin versant." http://fr.wikipedia.org/wiki/Bassin_versant.

[18] "Produit matriciel." http://fr.wikipedia.org/wiki/Multiplication_matricielle.

[19] "Approximation de pi." http://fr.wikipedia.org/wiki/Pi#Approximation_de_.CF.80.

[20] "Antutu." https://play.google.com/store/apps/details?id=com.antutu.CpuMasterFree&hl=fr.

[21] "Voltage control." https://play.google.com/store/apps/details?id=com.darekxan.voltagecontrol.

[22] "Battery monitor." https://play.google.com/store/apps/details?id=ccc71.bmw.

[23] "Battery log." https://play.google.com/store/apps/details?id=kr.hwangti.batterylog&hl=fr.

[24] "Cpufreq-selector." http://manpages.ubuntu.com/manpages/hardy/man1/cpufreq-selector1.html.