Ecole Doctorale STIC

Sciences & Technologies de l'Information et de la Communication

ED STIC - Proposition de Sujets de Thèse

pour la campagne d'Allocation de thèses 2017

Axe Sophi@Stic :	SystèmesRéseaux|
Titre du sujet :	Performance optimisation of parallel and distributed C++ applications
Mention de thèse :	Informatique
HDR Directeur de thèse inscrit à l'ED STIC :	Sid Touati

Co-encadrant de thèse éventuel :

Nom :	
Prénom :	
Email :	
Téléphone :	

Email de contact pour ce sujet :	sid.touati@inria.fr
Laboratoire d'accueil :	I3S/INRIA

Description du sujet :

Avec l'introduction massive des processeurs multi-cœurs, le parallélisme entre dans une nouvelle ère technologique. Il devient un domaine plus large, populaire, plus accessible en coût, et s'ouvre à de nouvelles applications qui étaient jusqu'à présent concentrées sur les calculs HPC.

Du point de vue langage de programmation et de modèles de calculs parallèles, les processeurs multi-cœurs sont vues comme étant des machines multi-processeurs classiques. Un cœur n'est qu'un processeur qui exécute un programme séquentiel. En revanche, pour optimiser les performances d'une application parallèle, il est indispensable de prendre en considération les caractéristiques micro–architecturales des processeurs multi-cœurs. En effet, les cœurs partagent des ressources matérielles invisibles au logiciel (différents niveaux de cache, réseau sur puce). Afin que le logiciel puisse s'exécuter le plus efficacement possible sur une architecture multi-cœur, il faut étudier de nouvelles méthodes d'optimisation des performances qui s'appuient

sur des connaissances fines de la micro-architecture. Sans cela, les performances affichées des programmes parallèles resteraient fictives et éloignées des capacités crêtes du processeur : même si le système d'exploitation affiche qye 100% des CPU sont utilisés à 100% de leur capacité visible, nous savons qu'en fait ces CPU ne sont utilisés qu'à une petite fraction de leur capacité maximale, car la micro-architecture n'est pas pleinement exploitée. Même si la scalabilité du speedup affiche de bons scores, nous savons aussi que les temps d'exécution obtenus sont assez éloignés des capacités des processeurs.
Le sujet de cette thèse aborde l'optimisation des performances des applications parallèles à divers niveaux :
1. Niveau compilation : étudier de nouvelles méthodes pour générer du code efficace d'appel de fonctions des librairies de threads.
2. Niveau système d'exploitation : étudier de nouvelles méthodes d'ordonnancement et de placement de processus légers.
3. Niveau librairie d'exécution : étudier la possibilité d'un mécanisme qui transmets des informations calculées statiquement par le compilateur aux librairies d'exécution afin de transmettre des informations sémantiques sur le parallélisme au niveau application.

Le langage adopté sera C++ (version 2011), car c'est un langage orienté objet très populaire et très convenable pour l'écriture d'applications parallèles hautes performances. Contrairement aux langages interprétés comme java et python, C++ est un langage qui peut être compilé et optimisé pour que le code binaire soit adapté au mieux aux caractéristiques matérielles des processeurs multi-cœurs. Cela fait de lui un excellent langage de programmation efficace et modulaire.

Ce sujet de thèse a beaucoup de débouchés :
- Sur le plan académique, il permet de faire le lien entre diverses couches et domaines, qui sont la compilation, le système d'exploitation, le parallélisme et la programmation orientée objet. Le doctorant se verra son expertise renforcée entre ces différents thèmes en informatique.
- Sur le plan industriel, ce sujet de thèse permettra de proposer des méthodes de programmation efficaces pour des programmes parallèles commerciaux orientés objets qui tirent profit pleinement des processeurs multi-cœurs disponibles dans quasiment tous les ordinateurs individuels. Grâce à cette programmation efficace, les logiciels produits seront plus compétitifs et de meilleure qualité.

English version:

Distributed computing applications are often programmed with a high level view of the underlying hardware platform. Usually, a distributed application encodes a parallel algorithm with distributed memory model, with an abstract view of the parallel machine. The used performance metrics of distributed applications are theoretical and high level, they have no direct relationship with the final observed execution times: high level parallel degree, CPU occupation rate, amount of communication and network transactions, scalability and speedup are all such performance metrics. These metrics are used to judge whether a distributed application or algorithm is efficient or not. However, it is very common in practice that theoretically "good"

distributed applications end with poor and disappointing measured performances. That is, the observed execution times of distributed applications are often far from the expected performance on high performance computers, especially on modern multicore processors. The reasons are known: distributed application programmers do neither write compiler friendly codes nor modern processor friendly codes. The consequences are that applications written with high-level languages are not well optimised to run in harmony with the modern processor architecture and micro-architectures. Even if CPU (core) occupancy as reported by the operating systems are high (let say 100%), the execution times may still be disappointing: the reason is that middleware and operating systems have an abstract view of the processors: all processor stalls are hidden micro-architectural phenomena that are not observed by the OS. Even if an OS or a middleware reports that a CPU is occupied in 100% of its time, in practice the processor functional units are maybe used at 20% of their peak capacity. The end user and the programmer get misleading conclusions and think that the distributed application uses the distributed machines at high occupancy rate, while in practice it is not: an important potential of performance improvement still remains even if CPU are used at 100%. The reasons are known from the high performance computing field: processor functional and data hazards (instruction level parallelism), memory hierarchy and shared caches between cores, branch prediction hazards, network traffic hazards, etc.

In this PhD thesis proposal, we will study new programming and coding methods for distributed object oriented applications to improve their performance. With the emergence of multicore processors, optimising the performance of distributed applications need new research efforts: multicore processors require finer performance tuning compared to historical parallel multi-processor systems.