Projet PPA
Le crible d'Eratosthène

13 décembre 2013

1 Présentation du sujet

Les nombres premiers sont les entiers naturels qui ne sont divisibles que par 1 et par eux-même (1 est exclu). Ces nombres premiers sont fondamentaux en arithmétique et beaucoup d'applications (cryptographie, tables de hachage, générateur de nombres aléatoires...) reposent sur ces nombres et sur la difficulté des algorithmes correspondants.

Nous considérons ici le crible d'Eratosthène pour calculer les nombres premiers entre 2 et N. Le principe de ce crible est donné en Algorithme 1.

Entrées : N, entier
Sorties : L'ensemble des nombres premiers entre 2 et N
début
  Créer une "liste" des entiers naturels allant de 2 à N et les mettre à l'état *non marqué* ;
  $k \leftarrow 2$, le premier nombre non marqué dans la liste ;
  répéter
    Marquer tous les multiples de k entre k^2 et N ;
    $k \leftarrow$ le plus petit nombre plus grand que k qui est non marqué ;
  jusqu'à $k^2 > N$;
  Les nombres premiers sont les nombres non marqués ;
fin

Algorithme 1: Le crible d'Eratosthène

On s'intéresse ici aux grandes valeurs de N : par exemple $N = 1\,000\,000\,000$. On souhaite de plus récupérer l'ensemble des nombres premiers stockés de façon contiguë dans un tableau en mémoire vive (côté CPU) à la fin du calcul. Pour information, il y a 50 847 534 nombres premiers entre 2 and 1 000 000 000.

Quelques remarques :
- il est algorithmiquement plus efficace de ne marquer que les multiples de k, plutôt que de vérifier pour chaque nombre s'il est divisible par k ;
- pour les grandes valeurs de N qui nous intéressent, $\sqrt{N}$ est "très petit" devant N.

2 Travail à effectuer

Dans votre travail, vous vous intéresserez aux points suivants.

1. – Réaliser un code séquentiel (C ou C++) qui implémente le crible d'Eratosthène ainsi que la sauvegarde contiguë des nombres premiers en mémoire.
   – Quel temps de traitement obtenez-vous sur un cœur CPU pour $N = 1\,000\,000\,000$?

2. – Déployer le crible d'Eratosthène sur GPU.
   – Comment obtenez-vous le stockage contigu des nombres premiers sur GPU ? Si nécessaire, proposez une version plus efficace.
   – Comment vérifiez-vous que votre calcul sur GPU est correct ?
   – Quel temps de traitement obtenez-vous pour $N = 1\,000\,000\,000$?
   N.B. : pour les temps GPU, on ne prendra pas en compte l'initialisation du GPU, ni les allocations mémoire. On pourra par contre distinguer les temps des différents kernels et les temps des transferts sur le bus PCI, ainsi que le temps global.

3. – Comment optimiser votre code séquentiel sur CPU ?
   – Comment mettre en place du parallélisme de tâches pour ce code sur CPU multicœur ?
   – Comparer votre version par tâches (OpenMP) avec une parallélisation multi-thread OpenMP.

Les tests de performance devront être réalisés sur une des trois machines : `ari-gpu-1`, `ari-gpu-2`, `ari-gpu-3`. Vous préciserez la machine utilisée pour vos tests de performance. Pour la partie CPU multicœur, on préférera une des machines de la salle SAR (version de `gcc`/`OpenMP` plus récente).

3 Travail à remettre

Vous devrez remettre les documents suivants :
– le code source, sous la forme d'une archive `tar` compressée et nommée suivant le modèle `projet_PPA_nom1_nom2.tar.gz`. L'archive ne doit contenir aucun exécutable, et elle devra contenir un fichier `Makefile`. Veillez à ce que le code compile sans avertissement et surtout sans aucune erreur.
– un rapport (au format PDF, de 5 à 10 pages, nommé suivant le modèle `rapport_PPA_nom1_nom2.pdf`) présentant vos choix, vos implémentations (sans code source), vos résultats et vos conclusions.
Une comparaison des différentes implémentations, une analyse la plus approfondie possible de tous vos résultats accompagnée d'une réflexion sur le comportement de votre programme seront particulièrement appréciées.
Tout phénomène d'accélération (ou d'absence d'accélération) inattendu et toute optimisation supplémentaire doivent être analysés dans le rapport.

4 Quelques précisions importantes

– Le projet est à réaliser par binôme. Aucun trinôme ne sera accepté.
– Veillez à bien compiler avec l'option de compilation `-O3`.
– Le projet devra être remis au plus tard le lundi 27 janvier 2014 à 23h59 (heure de Paris) par courriel à : `mathias.bourgoin@lip6.fr` et `pierre.fortin@lip6.fr`
– En cas d'imprévu ou de problème technique commun, n'hésitez pas à nous contacter pour que nous puissions vous proposer une solution ou une alternative.