INF7440 : Travail de fin de session
Automne 2007

(À remettre au plus tard mardi, 18 décembre, à 16h30.)

1 Objectif du travail

L'objectif de ce travail est de montrer que vous pouvez utiliser et mettre en pratique les notions vues dans le cours pour un problème nouveau, c'est-à-dire, un problème qui n'a pas été vu durant la session (ni dans les notes de cours, ni dans les devoirs ou exercices).

2 Ce qui doit être fait

Le travail consiste à choisir un problème et à développer trois (3) algorithmes différents permettant de résoudre le problème en question. Ces algorithmes devront être tout d'abord présentés en pseudocode. Des programmes MPD réalisant ces algorithmes devront aussi être écrits, puis compilés et exécutés.

1. Un algorithme séquentiel asymptotiquement efficace (appelons-le A_S).
2. Un algorithme parallèle asymptotiquement efficace en termes de temps d'exécution (en supposant un modèle abstrait et idéal d'exécution, style parallélisme itératif à granularité fine, parallélisme récursif ou PRAM) (appelons-le A_P^∞). Notez que cet algorithme *n'a pas* à être nécessairement optimal en termes de coût ou de travail.
3. Un algorithme parallèle qui soit relativement efficace lorsque mis en oeuvre sur une machine avec un nombre *limité* de processeurs (par ex., `arabica` avec six (6) processeurs) (appelons-le A_P^p).

Note : Vous devrez faire approuver le choix de votre problème en m'envoyant un courriel.

Chacun des algorithmes devra être analysé en termes des caractéristiques suivantes (et en fonction de la taille du problème, définie de façon appropriée pour le problème en question) et vos réponses (analyses) devront être (brièvement) justifiées et expliquées :

- Nombre de processeurs requis par l'algorithme.
- Temps d'exécution (complexité asymptotique).
- Coût (complexité asymptotique).

Finalement, vous devrez aussi déterminer, de façon *empirique*, l'accélération obtenue par la version parallèle de votre algorithme conçu pour un nombre limité de processeurs *relativement*

à la mise en oeuvre de l'algorithme séquentiel (c'est-à-dire, A_P^p vs. A_S). Vous devrez aussi expliquer brièvement les résultats obtenus.

Les versions séquentielle et parallèle pour un nombre limité de processeurs (A_S et A_P^p) devront évidemment être exécutées sur une même machine (plus précisément, `arabica`, à moins que vous ayez accès à une autre (?) machine *multi-processeurs* pouvant exécuter du code MPD).

3 Ce que vous devez remettre (barème de correction)

Pour ce travail, le barème de correction pour les différents items que vous devez remettre sera le suivant :

- [10 %] Une brève description du problème que vous avez choisi.
- [30 %] Vos trois algorithmes en pseudocode (A_S, A_P^∞ et A_P^p) ainsi qu'une courte explication du fonctionnement de chacun d'entre eux — en d'autres mots, vous devez expliquer brièvement de quelle façon l'algorithme fonctionne.

  Le code MPD réalisant ces algorithmes — une copie papier (listage) devra être jointe au document que vous me remettrez. Vous devrez aussi m'envoyer, *par courriel*, le code MPD de l'ensemble de votre programme.
- [20 %] L'analyse asymptotique de vos trois algorithmes (résultats et justifications).
- [20 %] Une preuve du bon fonctionnement des programmes conçus pour A_S, A_P^∞ et A_P^p.

  Plus précisément, vous devez montrer que les trois (3) versions produisent *exactement les mêmes résultats*. La meilleure stratégie consiste à vérifier que le résultat de la version séquentielle est correct, puis à vérifier, dans le programme lui-même, que le résultat produit par chacune des versions parallèles est bien le même que celui produit par la version séquentielle.
- [10 %] Les résultats empiriques (temps d'exécution, accélération) obtenus pour la version séquentielle et la version parallèle avec un nombre limité de processeurs (A_S et A_P^p) et leurs justifications.

  Plus précisément, vous devez donner les temps d'exécution de la version avec un nombre limité de processeurs pour $p = 1, 2, 4$ et 6 processeurs, et ce en utilisant des instances du problème de taille suffisamment grandes.
- [10 %] Qualité du français.
- [**Bonus 10** %] Un bonus (jusqu'à 10 %) pourra être accordé si le problème choisi est particulièrement difficile.

Note importante : Lorsque vous expliquez votre problème ou vos algorithmes, il est important, le cas échéant, de bien citer vos sources (références bibliographiques). En d'autres mots, l'inclusion d'un bout de texte tiré textuellement d'un livre ou d'un article n'est permise que si vous indiquez explicitement la référence appropriée — dans le cas contraire, il s'agit alors d'une forme de *plagiat* (infraction de nature académique).

4 Autres informations

- Ce travail peut être fait seul ou avec une (1) autre personne.
- Le travail compte pour 25 % de la note finale du cours, et ce peu importe qu'il soit fait seul ou en équipe.
- La qualité du français écrit sera évaluée.
- Les travaux doivent être remis, sous forme de document *papier*, dans la boîte de remise des travaux devant l'entrée du secrétariat du département d'informatique (PK-4150).
- La pénalite pour un travail remis en retard sera de 10 % par jour (complet ou partiel). La pénalité s'applique dès la date limite de remise indiquée plus haut. Les travaux seront ramassés le mardi en fin d'après-midi et leur correction débutera mercredi matin.

5 Idées de sujets

Des solutions (séquentielle et/ou parallèle) à ces divers problèmes sont présentées, en partie, dans les références indiquées après chaque sujet.

- Identifier les composantes connexes d'un graphe non dirigé [MB00] [JaJ92]
- Résoudre un système d'équations linéaires (tridiagonale ou non) [MB00] [JaJ92]
- Déterminer si un ensemble de segments de ligne couvre un intervalle [MB00]
- Calculer l'enveloppe convexe d'une série de points (*convex hull*) [MB00] [JaJ92]
- Calculer la transformée discrète de Fourier d'une série de valeurs [JaJ92] [XI98]
- Identifier les régions de *pixels* dans une image (traitement d'images) [AO93] [MB00]
- Déterminer la triangulation de coût minimum d'un polygone régulier [GGKK03]
- Simuler un groupe de particules (problème des N-corps) à l'aide de l'algorithme de Barnes-Hut [KKT01]
- Rechercher les occurrences d'un motif dans un arbre à l'aide d'un arbre des suffixes [Cha02]
- *Haploptype block partitioning* [ZDT+02]
- Alignement de séquences multiples [Cha04]

Vous pouvez aussi choisir un problème qui n'est pas indiqué dans cette liste. Dans tous les cas, vous devez me consulter (`tremblay.guy@uqam.ca`) pour faire *approuver* le choix de votre problème.

Références

[AO93] G.R. Andrews and R.A. Olsson. *The SR Programming Language.* Benjamin/Cummings Publishing, 1993. [QA76.73S68A54].

[Cha04] K.-M. Chao. Dynamic programming strategies for analyzing biomolecular sequences. In *Selected Topics in Post-genome Knowledge Discovery.* Singapore University Press, 2004. `http://www.lacim.uqam.ca/~chauve/Enseignement/INF7440/A04/COURS4/DOC-Biomolecular-Sequences.pdf`.

[Cha02] C. Chauve. Tree pattern matching for linear static terms. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval*, pages 160-169. Springer-Verlag, *Lecture Notes in Computer Science*, Volume 2476, 2002. `http://www.lacim.uqam.ca/~chauve/publications.html`

[CLR94] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction à l'algorithmique.* Dunod, 1994. [QA76.6C65614].

[GGKK03] A. Grama, A. Gupta, G. Karypis, and V. Kumar. *Introduction to Parallel Computing (Second Edition).* Addison-Wesley, 2003.

[JaJ92] J. JaJa. *An Introduction to Parallel Algorithms.* Addison-Wesley Publishing Company, 1992. [QA76.58J34].

[KGGK94] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing — Design and Analysis of Algorithms.* The Benjamin/Cummings Publishing Company, 1994. [QA76.58I58].

[KKT01] J. Keller, C. Kessler, and J. Traff. *Practical PRAM Programming.* John Wiley & Sons, Inc., 2001.

[MB00] R. Miller and L. Boxer. *Algorithms Sequential & Parallel.* Prentice-Hall, 2000. [QA76.9A43M55].

[NN04] R. Neapolitan and K. Naimipour. *Foundations of Algorithms Using C++ Pseudocode (Third edition).* Jones and Bartlett Publishers, 2004.

[XI98] C. Xavier and C. Iyengar. *Introduction to Parallel Algorithms.* John Wiley & Sons, 1998.

[ZDT+02] K. Zhang, M. Deng, T.Chen, M.S. Waterman, and F. Sun. A dynamic programming algorithm for haplotype block partitioning. In *Proc. Natl. Acad. Sci. USA*, volume 99, pages 7335–7339, 2002. `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=124231`.

Note : Un certain nombre de ces volumes sont disponibles à la réserve de la bibliothèque. Pour certains autres, vous pouvez me contacter.