Exposé oral et résumés écrits
INF7235
Hiver 2017

1 Objectifs

Un premier objectif de ce travail est de vous faire découvrir des aspects de la programmation parallèle que nous n'aurons pas le temps d'examiner ou d'approfondir en classe. Un autre objectif, puisqu'il s'agit d'un cours de cycle supérieur, est de vérifier que vous pouvez, *de façon autonome*, comprendre, synthétiser, expliquer et présenter un article de recherche portant la programmation parallèle, et ce tant de façon orale que de façon écrite.

2 Exposé oral

2.1 Ce que vous devez faire

L'exposé doit présenter, de façon claire et synthétique, un (1) article traitant de la programmation parallèle. (Voir section 5 pour quelques idées possibles de sujet — vous pouvez aussi proposer d'autres sujets.) Prenez note qu'il ne s'agit pas de faire une revue de la littérature sur un sujet, mais simplement de présenter un (1) article.

Note : Il faut faire approuver le choix de votre article au plus tard durant la semaine du **13 mars** — il ne pourra pas y avoir deux étudiants qui choisissent le même article. Pour indiquer votre choix d'article, vous devez utiliser le script à la page suivante :[1]

`http://www.labunix.uqam.ca/~tremblay/INF7235/Expose`

2.2 Dates et durée des exposés

Les exposés oraux auront lieu durant les semaines du **3 et 10 avril**.[2] Pour sélectionner le moment où vous voulez faire votre présentation, utilisez le script à la page suivante :

`http://www.labunix.uqam.ca/~tremblay/INF7235/Expose/choisir-date.cgi`

La durée (maximale!) des exposés devra être de 20–25 minutes, plus 5 minutes pour des questions — ce qui est donc semblable à une présentation faite dans une conférence.[3] Prenez note que je ferai respecter ces contraintes de temps de façon *stricte*.

Note : Une grille d'évaluation pour l'exposé orale est aussi disponible sur le même site. C'est cette grille que j'utiliserai pour évaluer votre présentation, grille qui sera aussi complétée par les autres étudiants.

[1] Un courriel indiquant votre choix m'est transmis automatiquement lorsque vous complétez ce formulaire.

[2] Lundi 17 avril : pas de cours (lundi de Pâques). Lundi 24 avril: examen final.

[3] Dans de nombreuses conférences, la durée d'un exposé est limitée à un maximum de 20 minutes plus 10 minutes pour les questions. Dans d'autres conférences, c'est 25 minutes plus 5 minutes pour des questions.

3 Résumés écrits

3.1 Ce que vous devez faire

Une compétence importante d'un chercheur est d'être capable de résumer (donc de **synthétiser**), au bon niveau de détail, les idées qu'il présente. Pour ce travail, vous allez donc devoir me remettre trois (3) résumés, de longueurs différentes :

1. Un résumé d'un paragraphe ($\leq$ 75–100 mots).
2. Un résumé d'environ une demi-page, typique d'un résumé (*abstract*) transmis lorsqu'un article est soumis à une conférence ($\leq$ 200–250 mots).
3. Un «résumé» plus détaillé, de 4–5 pages (l'article que vous lirez devrait généralement comporter 10–12 pages!).

Note : Pour les versions courtes, il n'est pas nécessaire (ni approprié) que vos résumés contiennent des sections. Par contre, pour la version longue, il est important de structurer votre résumé, entre autres, en utilisant explicitement des sections (numérotées).

3.2 Dates de remise

- Vous devrez me remettre, **au plus tard une semaine avant la date de votre présentation**, le résumé d'une demi-page. Il n'est pas nécessaire de fournir une page de présentation : identifiez simplement, en haut de la page, votre nom, l'article traité (auteur(s), titre, source), suivi de votre résumé. Utilisez un format raisonnable de mise en page (par exemple, la taille minimum des caractères devrait être de 10 points).
- Vous devrez me remettre au plus tard **mercredi 19 avril à 16h00** vos trois résumés écrits. Le cas échéant (nombreuses suggestions de correction), vous devrez aussi me remettre une version *révisée* de votre résumé d'une demi-page.

4 Remarques sur ce que doit être un *résumé*

Écrire un résumé synthèse consiste à lire un article, à le comprendre, et ensuite (après l'avoir mis de coté pendant quelques heures ☺) à l'expliquer *dans vos propres mots*, pour en décrire les éléments clés, les principales idées, conclusions, etc. — bref, décrire l'esprit et non la lettre. Il est donc possible de produire un résumé comportant n'importe quel nombre de pages (1/2, 2, 5 pages) : il s'agit de choisir le niveau de détail approprié, de déterminer ce qui est important et ce qui ne l'est pas étant donné les contraintes d'espace, donc d'aller *à l'essentiel*. En d'autres mots, il faut **synthétiser**.

Synthétiser signifie aussi **ne pas rester *collé* de façon trop étroite** à l'article, ni au niveau de la structure, ni au niveau des détails — par exemple, les sections (la structure) du résumé n'ont pas à être identiques à celles de l'article. Bref, il vaut mieux avoir trois (3) bonnes pages bien structurées et bien écrites, exprimant clairement ce que vous avez compris de l'article, que cinq (5) pages remplies de phrases et de détails que vous n'avez pas bien compris qui ont été simplement traduits de l'article.

Faire un résumé ou une synthèse d'un article *ne consiste donc pas* à extraire un certain nombre de phrases de l'article et à les traduire ou retranscrire pour en arriver à m mots ou n pages. En fait, toute phrase tirée, **ou traduite**, directement du texte original devrait être *citée* de façon explicite, c'est-à-dire en indiquant clairement qu'il s'agit d'une citation — par exemple, à l'aide de guillemets et en indiquant le numéro de page en référence.

> **Note :** L'utilisation textuelle (directe ou traduite) d'une partie de texte sans une référence appropriée constitue *une forme de plagiat*. Pour plus d'informations sur ce qui est considéré comme du plagiat, consultez l'URL suivant — qui explique aussi très bien ce qu'est une *reformulation* correcte :
>
> `http://www.bibliotheques.uqam.ca/plagiat`.

Remarques concernant l'exposé oral

Des remarques similaires quant «*au niveau de détail*» s'appliquent à l'exposé oral, qui ne devrait pas dépasser 20–25 minutes. Évitez l'erreur, fréquente, de donner trop de détails au début de la présentation, puis de vous dépêcher à la fin pour tenter de tout couvrir...

Deux suggestion de lecture pour la préparation de l'exposé oral :

- «*Research Experiences for Undergraduates (Colorado State University)*, pages 26–50, disponible via le site *Web* du cours :

  `http://www.labunix.uqam.ca/~tremblay/INF7235/Expose`

- «*Patrons et anti-patrons de présentation*», Guy Tremblay, séminaire Latece, 2014 :

  `http://www.labunix.uqam.ca/~tremblay/MGL7460/Materiel/anti-patrons-presentation.pdf`

5 Quelques suggestions de thématiques et d'articles

- Langages pour le traitement de données massives [14, 44, 45, 5]
- Parallélisme irrégulier [26, 32, 30, 23]
- Parallélisme de flux [13] ; modèle FastFlow [4, 3, 2]
- Traitement de graphes [33]
- Dégogage de programmes massivement parallèle [27]
- Programmation de GPU [29, 38]
- *Threads* déterministes [31]
- Mémoire transactionnelle [28, 24, 19, 36, 18]
- Primitives performantes de synchronisation [10]
- Modèles de mémoire partagée [1]
- Compilation pour machines parallèles [17]
- Langage de scripts [41]
- Comparaison des divers langages [20, 35]
- Développement de programmes parallèles [25]
- Loi d'Amdhal pour machines multi-coeurs [42].
- Transformation de programmes MPI [16]
- Programmation parallèle X10 [15]
- Langages PGAS : Chapel [12], Langage Titanium [43], Habanero Java [11].
- Mise en oeuvre des opérations collectives de MPI [40]
- Approche BSP [6].
- Échange de messages [21, 22]
- Parallélisme de données [34, 7]
- *Concurrent Collections* : Définition et mise en oeuvre [9, 8, 39, 46]
- Patrons pour la programmation parallèle : voir site web indiqué plus bas.

Vous pouvez choisir d'autres sujets ou articles que ceux indiqués plus haut. Pour des suggestions, venez me voir et, en fonction de vos intérêts, je *tenterai* de vous faire des propositions. Entre autres, vous pouvez trouver de nombreux articles, possiblement en lien avec des sujets plus spécifiquement liés à vos intérêts ou votre projet de recherche, dans les comptes-rendus des conférences suivantes, tous publiés dans la série «*Lecture Notes in Computer Science*» (Springer-Verlag) :

- *Euro-Par Parallel Processing Workshops*
- *Algorithms and Architectures for Parallel Processing*
- *Languages and Compilers for Parallel Computing*
- *ECOOP — Object-Oriented Programming*
- *Large-scale Scientific Computing*

6 Sites *Web* pour bibliographies et autres idées de sujet

Un site *Web* est disponible pour consulter (par l'intermédiaire d'un script `cgi`) ma liste de références bibliographiques sur la programmation parallèle et les architectures parallèles :
`http://www.labunix.uqam.ca/~tremblay/chercher-reference.cgi`

Vous pouvez aussi trouver de nombreuses références sur les sites *Web* suivants — plus spécifiquement, le deuxième site est dédié aux *Resources on Parallel Patterns* :

- `http://www.cs.rit.edu/~ncs/parallel.html`
- `http://www.cs.wustl.edu/~schmidt/patterns-ace.html`

Références

[1] S.V. Adve and H.-J. Boehm. Memory models: A case for rethinking parallel languages and hardware. *Communications of the ACM*, 53(8):90–101, August 2010.

[2] M. Aldinucci, M. Danelutto, P. Kilpatrick, and M. Torquati. Fastflow: High-level and efficient streaming on multi-core. In S. Pllana and F. Xhafa, editors, *Programming Multi-core and Many-core Computing Systems*. Wiley, 2014. `http://calvados.di.unipi.it/dokuwiki/doku.php/ffnamespace:papers`.

[3] Marco Aldinucci, Marco Danelutto, Lorenzo Anardu, Massimo Torquati, and Peter Kilpatrick. Parallel patterns + macro data flow for multi-core programming. In *Proc. of Intl. Euromicro PDP 2012*, pages 27–36, Garching, Germany, February 2012. IEEE.

[4] Marco Aldinucci, Marco Danelutto, Peter Kilpatrick, Massimiliano Meneghin, and Massimo Torquati. Accelerating code on multi-cores with FastFlow. In *Proc. of 17th Intl. Euro-Par 2011 Parallel Processing*, volume 6853 of *LNCS*, pages 170–181, Bordeaux, France, August 2011. Springer.

[5] Timo Bingmann, Michael Axtmann, Emanuel Jöbstl, Sebastian Lamm, Huyen Chau Nguyen, Alexander Noe, Sebastian Schlag, Matthias Stumpp, Tobias Sturm, and Peter Sanders. Thrill: High-performance algorithmic distributed batch data processing with C++. *CoRR*, abs/1608.05634, 2016.

[6] O. Bonorden, B. Juurlink, I. von Otte, and I. Rieping. The padeborn university BSP (PUB) library. *Parallel Computing*, 29:187–207, 2003.

[7] T. Braunl. Parallaxis-III: Architecture-independent data parallel processing. *IEEE Trans. on Soft. Eng.*, 26(3):227–243, 2000.

[8] Z. Budimlić, M. Burke, V. Cavé, K. Knobe, G. Lowney, R. Newton, J. Palsberg, D. Peixotto, V. Sarkar, F. Schlimbach, and S. Taşirlar. Concurrent collections. *Scientific Programming*, 18(3-4):203–217, August 2010.

[9] Z. Budimlic, A. Chandramowlishwaran, K. Knobe, G. Lowney, V. Sarkar, and L. Treggiari. Multi-core implementations of the concurrent collections programming model. In *CPC' 09: 14th International Workshop on Compilers for Parallel Computers*, 2009.

[10] D. Bueso. Scalability techniques for practical synchronization primitives. *Comm. of the ACM*, 58(1):66–74, 2015.

[11] V. Cavé, J. Zhao, J. Shirako, and V. Sarkar. Habanero-Java: The new adventures of old X10. In *Proc. of the 9th International Conference on Principles and Practice of Programming in Java*, PPPJ '11, pages 51–61, 2011.

[12] B.L. Chamberlain, D. Callahan, and H.P. Zima. Parallel programmability and the Chapel language. *International Journal of High Performance Computing Applications*, 21(3):291–312, August 2007.

[13] R.D. et al. Chamberlain. Auto-pipe: Streaming applications on architecturally diverse systems. *IEEE Computer*, 43(3):42–49, March 2010.

[14] Craig Chambers, Ashish Raniwala, Frances Perry, Stephen Adams, Robert Henry, Robert Bradshaw, and Nathan. FlumeJava: Easy, efficient data-parallel pipelines. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 363–375, 2 Penn Plaza, Suite 701 New York, NY 10121-0701, 2010.

[15] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. von Praun, and V. Sarkar. X10: an object-oriented approach to non-uniform cluster computing. In *OOPSLA '05: Proceedings of the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 519–538, New York, NY, USA, 2005. ACM.

[16] A. Danalis, L. Pollock, and M. Swany. Automatic MPI application transformation with ASPhALT. In *Parallel and Distributed Processing Symposium (IPDPS 2007)*, pages 1–8, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[17] C. Dave, H. Bae, S.-J. Min, Se. Lee, R. Eigenmann, and S. Midkiff. Cetus: A source-to-source compiler infrastructure for multicores. *Computer*, 42(12):36–42, 2009.

[18] Aleksandar Dragojević, Pascal Felber, Vincent Gramoli, and Rachid Guerraoui. Why STM can be more than a research toy. *Commun. ACM*, 54:70–77, April 2011.

[19] U. Drepper. Parallel programming with transactional memory. *CACM*, 52(2):38–43, Feb. 2009.

[20] Lusk. E. and K. Yelick. Languages for high-productivity computing: The DARPA HPCS language project. *Parallel Processing Letters*, 17(1):89–102, 2007.

[21] S. Gorlatch. Toward formally-based design of message passing programs. *IEEE Trans. on Soft. Eng.*, 26(3):276–288, 2000.

[22] S. Gorlatch. Send-Recv considered harmful? myths and truths about parallel programming. In *Parallel Computing Technologies*, pages 243–257. Springer-Verlag, LNCS-2127, 2001.

[23] Mahantesh Halappanavar, Alex Pothen, Ariful Azad, Fredrik Manne, Johannes Langguth, and Arif Khan. Codesign lessons learned from implementing graph matching on multithreaded architectures. *IEEE Computer*, 48(8):46–55, Aug. 2015.

[24] T. Harris, S. Marlow, S. Peyton-Jones, and M. Herlihy. Composable memory transactions. *CACM*, 51(8):91–100, August 2008.

[25] L. Hochstein and V.R. Basili. The ASC-alliance projects: A case study of large-scale parallel scientific code development. *IEEE Computer*, 41(3):50–58, March 2008.

[26] M. Kulkarni, K. Pingali, B. Walter, G. Ramanarayanam, K. Bala, and P.L. Chew. Optimistic parallelism requires abstractions. *CACM*, 52(39):89–97, Sept. 2009.

[27] I. Laguna, D.H. Ahn, and B.R. *et al.* De Supinski. Debugging high performance computing applications at massive scales. *Comm. of the ACM*, 58(9):72–81, 2015.

[28] J. Larus and C. Kozyrakis. Transactional memory. *CACM*, 51(5):80–88, July 2008.

[29] V.W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A.D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey. Debunking the 100X GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, pages 451–460. ACM, 2010.

[30] Andrew Lenharth and Keshav Pingali. Scaling runtimes for irregular algorithms to large-scale NUMA systems. *IEEE Computer*, 48(8):35–44, Aug. 2015.

[31] T. Liu, C. Curtsinger, and E.D. Berger. Dthreads: efficient deterministic multithreading. In *Proc. of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 327–336. ACM, 2011.

[32] Scott Lloyd and Maya Gokhale. In-memory data rearrangement for irregular, data-intensive computing. *IEEE Computer*, 48(8):18–25, Aug. 2015.

[33] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A System for Large-scale Graph Processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.

[34] L.S. Nyland, J.F. Prins, A. Goldberg, and P.H. Mills. A design methodology for data-parallel applications. *IEEE Trans. on Soft. Eng*, 26(4):293–314, 2000.

[35] V. Pankatrius, A. Jannesari, and W.F. Tichy. Parallelizing Bzip2: A case study in multicore software engineering. *IEEE Software*, 26(6):70–77, November 2009.

[36] C.J. Rossbach, H.E. Ramadan, O.S. Hofmann, D.E. Porter, A. Bhandari, and E. Witchel. TxLinux and MetaTM: Transactional memory and the operating system. *Communications of the ACM*, 51(9):83–91, Sept. 2008.

[37] D.B. Skillicorn and D. Talia. *Programming Languages for Parallel Processing*. IEEE Computer Society Press, 1995.

[38] J.A. Stratton, C. Rogrigues, R. Sung, L.-W. Chang, N. Anssari, G. Liu, W.W. Hwu, and N. Obeid. Algorithms and data optimization techniques for scaling to massively threaded systems. *IEEE Computer*, 44(8):26–32, 2012.

[39] S. Tasirlar and V. Sarkar. Data-driven tasks and their implementation. In *Proceedings of the 2011 International Conference on Parallel Processing*, ICPP '11, pages 652–661, Washington, DC, USA, 2011. IEEE Computer Society.

[40] R. Thakur and W. Gropp. Improving the performance of collective operations in MPICH. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 257–267. Springer-Verlag, LNCS-2840, 2003.

[41] M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford, and I. Raicu. Parallel scripting for applications at the petascale and beyond. *Computer*, 42(11):50–60, 2009.

[42] D.H. Woo and H.-H.S. Lee. Extending amdhal's law for energy-efficient computing in the many-core era. *IEEE Computer*, 41(12):24–31, Dec. 2008.

[43] K. Yelick, P. Hilfinger, S. Graham, D. Banochea, J. Su, A. Kamil, K. Datta, P. Colelle, and T. Wen. Parallel languages and compilers: Perspective fro the Titanium experience. *International Journal of High Performance Computing Applications*, 21(3):266–290, August 2007.

[44] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

[45] Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proc. of the 24th ACM Symposium on Operating Systems Principles*, SOSP, pages 423–438, New York, NY, USA, 2013. ACM.

[46] P. Zaichenkov, B. Gijsbers, C. Grelck, O. Tveretina, and A. Shafarenko. A case study in coordination programming: Performance evaluation of Concurrent Collections vs. S-Net. In *28th IEEE International Parallel and Distributed Processing Symposium (IPDPS'14) Workshops, Phoenix, USA*. IEEE Computer Society, 2014.