Seminareinführung
Sprachen für Parallelprogrammierung

Andreas Zwinkau <zwinkau@kit.edu>

13. April 2010

1 Einführung

Die Grenzen der Taktraten von Prozessoren scheinen erreicht zu sein. Zur weiteren Leisterungssteigerung geht der Trend zu Multicoresystemen. Um diese effizient zu Programmieren sind neue Programmiermodelle und Programmiersprachen nötig. In diesem Seminar sollen aktuelle wissenschaftliche Erkenntnisse in der Programmierung nebenläufiger Anwendungen vorgestellt werden.

Ziel des Seminars ist es einen Überblick über die aktuelle Forschung im Bereich der Parallelprogrammierung (soweit relevant im Kontext unseres Lehrstuhls) zu erhalten. Weiter sollen die Teilnehmer Erfahrungen mit der Recherche, dem Zusammenfassen und Vortragen wissenschaftlicher Erkenntnisse sammeln. Durch Feedback von Kommilitonen und Mitarbeitern wird für Studien- und Diplomarbeit trainiert.

Jeder Teilnehmer erarbeitet einen Vortrag sowie eine Ausarbeitung. An den Vortragsterminen werden jeweils zwei Vorträge gehalten, wobei die Anwesenheit für alle Teilnehmer Pflicht ist.

2 Termine und Orte

Aktuelle Termine finden sich auf der Website! Die Vorbesprechung ist am 15. April um 15:45 Uhr im AVG (Gebäude 50.41) in Raum 207. Man beachte, dass die Vorbesprechung und die Vorträge *nicht* am selben Ort sind.

Die konkreten Termine werden kurz nach der Vorbesprechung bekannt gegeben. In der Vorbesprechung wird nur die Reihenfolge festgelegt.

3 Themen

Für fettgedruckte Themen muss es einen Vortrag geben und jedes Thema kann nur einmal vergeben werden.

1. **X10** [CGS+05, BCK+09] Was sind Eigenheiten der Sprache? Wofür ist sie geeignet und wofür nicht?
2. **Chapel** [CCZ07, CCZ] Was sind Eigenheiten der Sprache? Wofür ist sie geeignet und wofür nicht?
3. **Fortress** [ACH+08] Was sind Eigenheiten der Sprache? Wofür ist sie geeignet und wofür nicht?
4. Cilk [BJK+95] Was sind Eigenheiten der Sprache? Wofür ist sie geeignet und wofür nicht?
5. Erlang [Lar09, Arm07, Arm97] Was sind Eigenheiten der Sprache? Wofür ist sie geeignet und wofür nicht?
6. Concurrent Garbage Collection [PD01] Wie funktioniert es? Wo ist es sinnvoll?
7. Transactional Memory [HMJH05, FH07] Wie funktioniert es? Wo ist es sinnvoll?
8. Memory Models [Boe05, MPA05] Was ist das? Sind sie notwendig?
9. Polytope Model [GLW98] Was ist das?
10. Autoparallelization [LL97, BC86] Was ist das? Welche Ansätze gibt es? Wie gut funktionieren diese?
11. Google MapReduce[1] [DG08] Wie funktioniert es? Wo ist es sinnvoll?
12. Atomic Instructions[1] [HPS02] Wie funktioniert es? Wo ist es sinnvoll?
13. Architectures[1] [KL94, LAS+07] Welche Architekturarten gibt es? Wofür sind sie jeweils geeignet?
14. Synchronization primitives[1] [Her91] Welche gibt es? Wofür sind sie jeweils geeignet?

[1] Relativ einfaches Thema. Kandidat für den ersten Vortragstermin.

4 Vorgaben

Jeder Teilnehmer muss für eine erfolgreiche[2] Teilnahme leisten:

- Ein Vortrag von 30 Minuten zum vereinbarten Thema. Dazu ein entsprechender Foliensatz im PDF-Format[3].
- Eine Ausarbeitung von 6–10 Seiten als pdf bis spätestens zum letzten Vortragstermin. Formatvorgabe ist DIN A4.
- Anwesenheit bei den anderen Vorträgen.

Die fertige Versionen von Folien und Ausarbeitung sollten jeweils mindestens drei Tage vorher vom Betreuer begutachtet werden, um stilistische und inhaltliche Fehler zu vermeiden. Wir erwarten außerdem eine *selbstständige Literaturrecherche* über die hier angegebenen Quellen hinaus. Weiterhin sollen im Fall der Programmiersprachthemen *eigene praktische Erfahrungen* vorgestellt werden.

Viel Spaß!

Literatur

[ACH+08] Eric Allen, David Chase, Joe Hallett, Victor Luchangco, Jan-Willem Maessen, Sukyoung Ryu, Guy L. Steele, and Sam Tobin-Hochstadt. *The Fortress Language Specification.* 2008.

[Arm97] Joe Armstrong. The development of erlang. In *ICFP '97: Proceedings of the second ACM SIGPLAN international conference on Functional programming*, pages 196–203, New York, NY, USA, 1997. ACM.

[Arm07] Joe Armstrong. A history of erlang. In *HOPL III: Proceedings of the third ACM SIGPLAN conference on History of programming languages*, New York, NY, USA, 2007. ACM.

[BC86] Michael Burke and Ron Cytron. Interprocedural dependence analysis and parallelization. *SIGPLAN Not.*, 21(7):162–175, 1986.

[BCK+09] Ganesh Bikshandi, Jose G. Castanos, Sreedhar B. Kodali, V. Krishna Nandivada, Igor Peshansky, Vijay A. Saraswat, Sayantan Sur, Pradeep Varma, and Tong Wen. Efficient, portable implementation of asynchronous multi-place programs. In *PPoPP '09: Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 271–282, New York, NY, USA, 2009. ACM.

[2] Sollte ein benoteter Schein notwendig sein, bitte frühzeitig Bescheid sagen.

[3] Der Vortrag selbst kann auch mit eigenem Laptop, MS Powerpoint, Apple Keynote oder Ähnlichem durchgeführt werden. In diesem Fall ist ein pdf-Export notwendig.

[BJK+95] Robert D. Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, and Yuli Zhou. Cilk: an efficient multithreaded runtime system. *SIGPLAN Not.*, 30(8):207–216, 1995.

[Boe05] Hans J. Boehm. Threads cannot be implemented as a library. In *PLDI '05: Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, volume 40, pages 261–268, New York, NY, USA, June 2005. ACM.

[CCZ] David Callahant, Bradford L. Chamberlaint, and Hans P. Zimaj. The cascade high productivity language*.

[CCZ07] B. L. Chamberlain, D. Callahan, and H. P. Zima. Parallel programmability and the chapel language. *Int. J. High Perform. Comput. Appl.*, 21(3):291–312, 2007.

[CGS+05] Philippe Charles, Christian Grothoff, Vijay Saraswat, Christopher Donawa, Allan Kielstra, Kemal Ebcioglu, Christoph von Praun, and Vivek 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.

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[FH07] Keir Fraser and Tim Harris. Concurrent programming without locks. *ACM Trans. Comput. Syst.*, 25(2):5+, May 2007.

[GLW98] M. Griebl, C. Lengauer, and S. Wetzel. Code generation in the polytope model. pages 106–111, October 1998.

[Her91] Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, January 1991.

[HMJH05] Tim Harris, Simon Marlow, Simon P. Jones, and Maurice Herlihy. Composable memory transactions. In *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 48–60, New York, NY, USA, 2005. ACM.

[HPS02] David Hovemeyer, William Pugh, and Jaime Spacco. Atomic instructions in java. In *ECOOP 2002 — Object-Oriented Programming*, pages 5–16. 2002.

[KL94] A. C. Klaiber and H. M. Levy. A comparison of message passing and shared memory architectures for data parallel programs. *SIGARCH Comput. Archit. News*, 22(2):94–105, 1994.

[Lar09] Jim Larson. Erlang for concurrent programming. *Commun. ACM*, 52(3):48–56, 2009.

[LAS+07] Jacob Leverich, Hideho Arakida, Alex Solomatnikov, Amin Firoozshahian, Mark Horowitz, and Christos Kozyrakis. Comparing memory systems for chip multiprocessors. *SIGARCH Comput. Archit. News*, 35(2):358–368, May 2007.

[LL97] Amy W. Lim and Monica S. Lam. Maximizing parallelism and minimizing synchronization with affine transforms. In *POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 201–214, New York, NY, USA, 1997. ACM.

[MPA05] Jeremy Manson, William Pugh, and Sarita V. Adve. The java memory model. In *POPL '05: Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 378–391, New York, NY, USA, 2005. ACM.

[PD01] Tony Printezis and David Detlefs. A generational mostly-concurrent garbage collector. *SIGPLAN Not.*, 36(1):143–154, 2001.