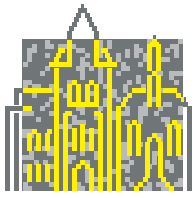
J. Knoop (TU Wien, AT), J. Lee (Seoul National Univ., ROK), S. Midkiff (Purdue Univ., US), D. Padua (Univ. of Illinois, US)
(Editors)

Hardware and Software Consistency Models: Programmability and Performance

Dagstuhl Seminar 03431 – October 19 to October 24, 2003
Dagstuhl-Seminar-Report No. 399



SCHLOSS DAGSTUHL

INTERNATIONALES
BEGEGNUNGS-
UND FORSCHUNGSZENTRUM
FÜR INFORMATIK

ISSN 0940-1121

Herausgegeben von IBFI gem. GmbH, Schloss Dagstuhl, 66687 Wadern, Germany.

Das Internationale Begegnungs- und Forschungszentrum für Informatik (IBFI) Schloss Dagstuhl ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Gesellschafter:

- Gesellschaft für Informatik e.V. – Bonn
- TH Darmstadt
- Universität Frankfurt
- Universität Kaiserslautern
- Universität Karlsruhe
- Universität Stuttgart
- Universität Trier
- Universität des Saarlandes

Motivation

Hardware consistency models define the order that events that occur on one processor, or memory subsystem, appear to occur to other processors or memory subsystems. We use Memory model to refer to the equivalent software concept. A memory model can be defined as part of the semantics of the programming language. The memory model defines the order that memory references in thread of a program, written in the language, should appear to other threads, written in the same language. A memory model defines the order that memory references in a thread of a computer program are mandated by the semantics of a language or other piece of system software to appear to occur in other threads in the computer program. Until recently, these issues were largely the province of specialists who designed memory subsystems and processor cache protocols, implementors of operating systems, and database architects. The design of consistency and memory models was skewed towards providing high performance at the expense of usability or programmability. There are at least two contributing factors for this. First, processors were expensive, and never quite fast enough, requiring performance be maximized. Second, multithreaded programming was used almost exclusively in the design of widely used components such as database systems and operating systems. Thus very labor intensive approaches to programming these consistency models was acceptable. Most ordinary programmers never had to deal with memory consistency issues.

The widespread availability of explicitly parallel programming targeting shared memory systems has changed this equation. In particular, Java, OpenMP, C#, P-Threads, and distributed shared memory systems have forced programmers to be aware of the underlying semantics of the memory model. And, in all of these systems, poor performance, incorrect programs and lack of portability can result from an improper understanding of the underlying model. Thus knowledge that was formerly required of a relatively small number of specialists is now required of large numbers of programmers in fact, required of the typical programmer. Given that the systems written by these typical programmers are not as widely disseminated as the systems written by the specialists, the cost of coping with the vagaries of consistency models is relatively much higher. Moreover, as the complexity of operating systems and middleware grows, the complexity of hardware and consistency models and software memory models leads to subtle errors in the code, degrading software reliability.

These changes in the tradeoffs between programmability and performance in memory models have sparked renewed research into how to design both consistency and memory models. Topics of intense interest include

- What are the trends in hardware and software consistency models?
- What is the performance loss associated with moving towards simpler consistency and memory models? How much loss is acceptable?
- How can hardware consistency models be made simpler for programmers with acceptable losses in performance?

- What compiler techniques can be used to mask the complexity of hardware consistency models, or mask the performance costs of simpler hardware consistency models?
- How can memory models be designed to allow programmers to more easily write correct programs? What are the costs of doing this in terms of missed compiler optimization opportunities and additional synchronization overhead in modern out-of-order processors?
- Can compile-time analyses and optimizations mitigate some of these costs, and if so how?
- Are heuristic approximations to expensive compile-time analyses sufficient?
- What idioms and software engineering tools can be used to increase programmability in the face of complex memory models?

We have two large goals for the seminar. First, we would like to foster discussions about the usability and performance requirements of consistency models in the different areas where these are important issues (architecture and hardware, databases, and programming languages) and give knowledgeable members of the fields the opportunity to learn from the experiences of their colleagues in different fields. From these discussions, we hope to come to a better understanding of the tradeoffs and possibilities thatcan be exploited by researchers and practitioners in each of these areas, and to come up with important research questions that will yield broadly applicable results. Because of Dagstuhl's schedule allowing for mix of unstructured discussion in a congenial environment and more formal presentations, we see it as an ideal setting for bringing together members of these different communities to tackle these diffcult issues.

Participants

- Adve, Sarita (University of Illinois – Urbana)
- Antoniu, Gabriel (IRISA – Rennes)
- Arvind, (MIT – Cambridge)
- Boehm, Hans J. (HP Labs – Palo Alto)
- Bougé, Luc (IRISA – Rennes)
- Chatterjee, Siddhartha (IBM TJ Watson Research Center – Yorktown Heights)
- Choi, Jong-Deok (IBM TJ Watson Research Center)
- Cintra, Marcelo (University of Edinburgh)
- Cohen, Albert (UPMC – Paris)
- Falsafi, Babak (Carnegie Mellon University – Pittsburgh)
- Hill, Mark D. (University of Wisconsin – Madison)
- Horspool, Nigel (University of Victoria)
- Kessler, Christoph W. (Linköping University)
- Knoop, Jens (TU Wien)
- Lee, Jaejin (Seoul National University)
- Manson, Jeremy (University of Maryland – College Park)
- Midkiff, Samuel P. (Purdue University)
- Moreira, José (IBM Systems and Technology Group)
- Padua, David (University of Illinois – Urbana)
- Petersen, Paul (Intel Corporation – Champaign)
- Pugh, William (University of Maryland – College Park)
- Puntigam, Franz (TU Wien)
- Rauchwerger, Lawrence (Texas A&M University)
- Schuster, Assaf (Technion – Haifa)
- Sura, Zehra N. (University of Illinois – Urbana)
- von Praun, Christoph (IBM TJ Watson Research Center – Yorktown Heights)
- Wong, Chi-Leung David (University of Illinois – Urbana)
- Yelick, Katherine (University of California – Berkeley)