More information about this series at http://www.springer.com/series/7407
Languages and Compilers for Parallel Computing

26th International Workshop, LCPC 2013
San Jose, CA, USA, September 25–27, 2013
Revised Selected Papers
Message from the Chairs

The 26th Workshop on Languages and Compilers for Parallel Computing was held in September 2013 in San Jose, California, USA. More than 50 researchers from around the world gathered together to present their latest results and exchange ideas on topics ranging from parallel programming models, compiler analysis techniques, parallel data structures and parallel execution models, to GPGPU and other heterogeneous execution models, code generation for power efficiency on mobile platforms, and debugging and fault tolerance for parallel systems.

The Program Committee, with the help of external reviewers, selected 20 papers out of 44 submissions for presentation at the workshop. Each paper had at least three reviews and was extensively discussed in the committee meeting.

We were honored to have two outstanding keynote addresses at LCPC 2013. Prof. Katherine Yelick, from University of California Berkeley and Lawrence Berkeley National Laboratory, presented “Avoiding, Hiding and Managing Communication”. She discussed how new systems are constrained in terms of both power density and energy, and require new programming models as well as algorithmic work to reduce the amount of communication. She demonstrated how PGAS languages reduce the communication costs through overlap, caching and aggregation. As communication hierarchies are becoming more complex (memory systems and interconnect), new language and compiler techniques need to be developed. She also discussed work done in her team on new parallel algorithms that use structure to reduce the amount of communication, and discussed the challenges to automate such methods through compiler transformations and auto-tuning.

David Sehr from Google presented our second keynote. He discussed compilation technologies for executing native code in the browser. Browsers are currently available on all platforms, and thus provide a common environment to deploy applications. However, there are both performance and security considerations when executing arbitrary code off the web into a complex engine such as the browser. David presented Google’s design of Native Client (NaCL) and discussed the compiler challenges to produce efficient code capable of running games at native speed. He concluded the keynote with a live demonstration of the system.

In addition to the paper presentations, we were fortunate to have six invited speakers who provided insights into new technologies and challenging research directions. We would like to thank the speakers: Benedict Gaster (Qualcomm), Rastislav Bodik (UC Berkeley), Jaejin Lee (Seoul National University, Korea), Samuel P. Midkiff (Purdue University), Lawrence Rauchwerger (Texas A&M University), and George Almasi (IBM Research).

The success of the LCPC 2013 workshop would not have been possible without help from many people. We thank the Program Committee members for their time and effort in reviewing papers. We thank Nancy May and Pamela Millart from Qualcomm for their help with the local organization. We thank the Qualcomm admin staff for
providing support hosting the workshop. The LCPC Steering Committee, David Padua, Lawrence Rauchwerger, Alex Nicolau, and Rudi Eigenmann, provided continuous support and encouragement.

We also thank Intel Corp. for their generous support.

And finally, we would like to thank all the authors who submitted to LCPC 2013. They made the workshop enjoyable.

September 2013

Călin Cașcaval
Pablo Montesinos
Organization

Workshop Chairs

Călin Cașcaval Qualcomm Research, USA
Pablo Montesinos Qualcomm Research, USA

Program Committee

James Brodman Intel Corporation, USA
Călin Cașcaval Qualcomm Research, USA
Hironori Kasahara Waseda University, Japan
Keiji Kimura Waseda University, Japan
Pablo Montesinos Qualcomm Research, USA
Sanjay Rajophadhye Colorado State University, USA
Michelle Strout Colorado State University, USA
Peng Tu Intel Corporation, USA

Steering Committee

David Padua University of Illinois at Urbana-Champaign, USA
Lawrence Rauchwerger Texas A&M University, USA
Alex Nicolau University of California, Irvine, USA
Rudolf Eigenmann Purdue University, USA

Referees

Farhana Aleen Fredrik Berg Kjolstad Tomoaki Tsumura
Carmen Badea Masayoshi Mase Yasutaka Wada
Umeda Dan Makoto Nakayama Cheng Wang
Alexandre Duchateau Catherine Olschanowsky Youfeng Wu
Akihiro Hayashi Hideki Saito Koichiro Yamashita
Sunpyo Hong Jun Shirako Akimasa Yoshida
Guillaume Iooss Mamoru Shimaoka Tomofumi Yuki
Kazuhiisa Ishizaka Albert Sidelnik Yun Zou
Chi-Keung Luk Xinmin Tian Xing Zhou

Supporting Institution

Intel Corporation, Santa Clara, CA, USA
Invited Talks
The Good, the Bad, and the Ugly: Heterogeneous Programming Models for Performance and Power in a Thermally Constrained World

Benedict Gaster
Qualcomm

Abstract. Heterogeneous computing is being realized in modern SoCs, because power and performance are key to delivering multimedia experiences in mobile devices that are limited by physical thermal limits. Programming these devices is challenging - lack of shared virtual address spaces, different ISAs, different performance capabilities, are just some of the factors that complicate any programming model.

Heterogeneous programming can be pretty ugly in places and is not easy for the first time developer, however, it has also been proven on multiple devices, ranging from tiny mobile devices that fit a pocket, to supercomputers, providing its users access to huge possibilities. In this talk I will reflect on the good, the bad, and the ugly of design and feature capabilities of OpenCL 2.0 and its new formalized foundations for shared memory model programming and support for data- and irregular-parallel workloads. With a particular focus on OpenCL 2.0’s low-level features for high-level programming abstractions we will discuss capabilities of Qualcomm’s task-based programming model, MARE, and how it might be integrated into a heterogenous platform supporting OpenCL.

Bio

Benedict R. Gaster is an architect working at Qualcomm on next-generation heterogeneous processors. Benedict has contributed extensively to the OpenCL’s design. Benedict has a Ph.D in computer science for his work on type systems for extensible records and variants.
Why Parallel Web Browsers?

Rastislav Bodik
UC Berkeley

Abstract. The quality of web browsers is reaching the point where HTML5 apps begin to compete with native apps (see for example the Firefox OS or the sencha.com Facebook app written entirely in HTML5). Thanks to the portability of HTML5 apps, the adoption of the HTML5 would erode the domination of closed mobile app platforms. To secure adoption of HTML5 on mobile devices, mobile browsers may still need to improve, though — in responsiveness, energy efficiency, programmability, as well as usability.

Our research group has been working on browser technologies since 2007. We were the first to parallelize key browser components such as the lexer, parser, CSS selectors and the layout engine. This talk will describe our recent results and ongoing work. Our work divides into (i) technologies for performance improvements, via code synthesis and parallelization; and (ii) new techniques for programmability. From the performance category, I will demo our browser-based GPU framework for real-time visualization of large data. In the programmability category, I will show how we can equip the browser with programming by demonstration, which allows users to automate tasks, extract and visualize web data, and layout documents without knowledge of CSS/HTML.

Bio

Ras Bodik is a Professor of Computer Science at UC Berkeley. He works on various flavors of program synthesis, from programming by demonstration to compilers for declarative languages. His group has applied synthesis to high-performance computing, web browser construction, algorithm design, document layout, and biology. Their web browser project investigates how to run client application stacks on low-power devices. He also designed a course on programming languages in which students design learn small languages by constructing a small modern web browser.
SnuCL: A Unified Framework of OpenCL

Jaejin Lee
Seoul National University, Korea

Abstract. OpenCL is a programming model for heterogeneous parallel computing systems. OpenCL provides a common abstraction layer across different processor architectures, such as CPUs, GPUs, DSPs, FPGAs, and Xeon Phi processors. OpenCL ICD (installable client driver) enables OpenCL platforms from different vendors for different processors to coexist under a single operating system instance. Applications choose a platform and dispatch OpenCL API calls to the platform with ICD. However, current OpenCL has two major limitations. First, to use different processors from different vendors in a single application, programmers need to explicitly specify a specific OpenCL platform for each processor. Moreover, OpenCL objects (buffers, events, etc.) cannot be shared across different platforms. Second, OpenCL is restricted to a system running a single operating system instance. To target a cluster running multiple operating system instances, programmers must use an OpenCL framework together with a communication library, such as MPI. This talk introduces how to overcome the limitations of current OpenCL with SnuCL. SnuCL is an OpenCL framework that provides the programmer with an illusion of a single OpenCL platform image. It naturally extends the original OpenCL semantics to a cluster system running multiple operating system instances. Programmers do not need to explicitly specify a specific platform in SnuCL, and OpenCL objects can be shared across different platforms. SnuCL is open-source software developed at Seoul National University, Korea.

Bio

Jaejin Lee is the director of the Center for Manycore Programming and a professor in the Department of Computer Science and Engineering at Seoul National University (SNU), Korea. He received a PhD degree in Computer Science from the University of Illinois at Urbana-Champaign (UIUC) in 1999. He also received an MS degree in Computer Science from Stanford University in 1995 and a BS degree in Physics from SNU in 1991. After obtaining the PhD degree, he spent a half year at the UIUC as a visiting lecturer. He was an assistant professor in the Department of Computer Science and Engineering at Michigan State University until 2002 before joining SNU. His primary research focus in these days is on heterogeneous parallel programming models, and building efficient heterogeneous supercomputers.
Characterizing and Detecting Smartphone Energy Bugs

Samuel P. Midkiff
Purdue University

Abstract. Modern power constrained devices such as mobile phones require programmers to explicitly manage whether components (including the processor and display) are turned on or off. User defined defaults often give a period of time that a component (e.g., the processor or display) will be on, but after that time expires the device will shut down unless the program being executed actively and explicitly prevents it. Other components, such as the GPS unit, the GSM or wireless transmitters and receivers, and so forth may shut down at any time, even during a call or while establishing a network connection. The operating systems for these devices provide wake locks that allow library and application programmers to force a component to stay active until the wait lock is released. This talk will describe strategies for automatically detecting these bugs. We will finish with a short discussion of how this issue relates to parallelism and compiler infrastructure issues.

Bio

Samuel Midkiff is a Professor of Electrical and Computer Engineering at Purdue University, where he has been since 2002. He received his PhD degree from the University of Illinois at Urbana-Champaign in 1992 and worked at the IBM TJ Watson Research Center from 1991 until 2002. He has had the pleasure of working on the Parafrase, PTran, Ninja and Cetus compiler infrastructures. His research is currently focused on energy bugs in mobile devices, using semantic information in compiling and issues related to debugging parallel programs and web hubs.
KLA: A New Algorithmic Paradigm for Parallel Graph Computations

Lawrence Rauchwerger
Texas A&M University

Abstract. This paper proposes a new algorithmic paradigm for parallel graph computations, k-Level Asynchronous (KLA), that bridges the level-synchronous and asynchronous paradigms for processing graphs.

The KLA paradigm allows vertex-centric fine-grained expression of parallel graph algorithms, while enabling the level of asynchrony to be parametrically varied from none (level synchronous) to full (asynchronous). This enables improved efficiency by expeditiously using an appropriate combination of expensive global synchronizations, as in level-synchronous algorithms, and possibly more, but less expensive local synchronizations (and potentially redundant work), as in asynchronous algorithms.

To enable the expression of a wide variety of graph algorithms in the KLA paradigm, we classify algorithms into three categories based on the penalty paid for asynchrony (e.g., cost of correcting a wrong guess), which theoretically depends on the resilience (for correctness) of the algorithm to message ordering. We propose how to best apply the KLA paradigm for each category of algorithms and use it to implement several important classes of graph algorithms including breadth-first search type computations (e.g., single-source shortest paths), PageRank, k-core and pointer jumping.

Results of an implementation of KLA using the STAPL Graph Library show good scalability to more than 16,000 cores for a variety of important graph algorithms. Compared to traditional approaches, KLA improves performance of certain graph algorithms on real-world graphs by 6x or more.
Parallel Programming for the Cloud

George Almasi
IBM Research

Abstract. Love it or hate it, cloud computing is here to stay. It is the chief enabler of the race to bottom in cost of computing resources. Non-technology companies are relying on it to replace their IT infrastructure; many technology companies are betting their livelihoods on it; government[s] spy on it; in short, most everybody has a stake in cloud computing.

This talk focuses on the languages used for setting up and running a large cloud infrastructure compute farm. The cloud software infrastructure is a large parallel distributed computation similar to an HPC compute farm. The programming language approach taken is however fundamentally different. I will discuss how the words “performance”, “scalability” and “consistency” apply to a motley collection of 100K+ lines of Python, Ruby and shell scripts, and how the best programming practices in these languages are used to deal with issues familiar to “real” parallel programmers: threads, locks, mutual exclusion, producer/consumer paradigms, fault tolerance etc., meanwhile allowing the cloud architect to keep his/her sanity while deploying and maintaining such a system.
Keynotes
Avoiding, Hiding and Managing Communication

Katherine Yelick

EECS Department, University of California, Berkeley, CA, USA
Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract. Future computing system designs will be constrained by power density and total system energy, and will require new programming models and implementation strategies. Data movement in the memory system and interconnect will dominate running time and energy costs, making communication cost reduction the primary optimization criteria for compilers and programmers. Communication cost can be divided into latency costs, which are per communication event, and bandwidth costs, which grow with total communication volume. The trends show growing gaps for both of these relative to computation, with the additional problem that communication congestion can conspire to worsen both in practice.

In this talk I will discuss prior work an open problems in optimizing communication, starting with PGAS languages. This involves reducing the communication cost, through overlap, and the frequency through caching and aggregation. Much of the compile-time work in this area was done in the Titanium language, where strong typing and data abstraction aid in program analysis, while UPC compilers tend to use more dynamic optimizations. There are still many open questions on the design of languages and compilers, especially as the communication hierarchies become deeper and more complex.

Bandwidth reduction often requires more substantial algorithmic transformations, although some techniques, such as loop tiling, are well known. These can be applied as hand-optimizations, through code generation strategies in autotuned libraries, or as fully automatic compiler transformations. Less obvious techniques for communication avoidance have arisen in the so-called “2.5D” parallel algorithms, which I will describe more generally as “.5D” algorithms. These ideas are applicable to many domains, from scientific computations to database operations. In addition to having provable optimality properties, these algorithms also perform well on large-scale parallel machines. I will end by describing some recent work that lays the foundation for automating transformations to produce communication optimal code for arbitrary loop nests.

Bio

Katherine Yelick is the Associate Laboratory Director for Computing Sciences at Lawrence Berkeley National Laboratory and a Professor of Electrical Engineering and Computer Sciences at the University of California at Berkeley. She co-invented the UPC and Titanium languages as well as techniques for self-tuning sparse matrix
kernels. She earned her Ph.D. in EECS from MIT and has been a professor at UC Berkeley since 1991 with a joint appointment at LBNL since 1996. She has received multiple research and teaching awards, is an ACM Fellow and serves on numerous advising committee, including the California Council on Science and Technology and the National Academies Computer Science and Telecommunications Board.
Bringing Native Code to the Web

David Sehr
Google, Mountain View, CA, USA

Abstract. In this talk we discuss some of the security concerns we’ve faced when compiling for web applications and how compilers are used to build systems.

Bio

David Sehr is the lead of the Native Client project at Google. David co-founded this effort, which uses software fault isolation to run untrusted native code securely in a web browser or other environment. He has been at Google since 2007. Prior to that he was a Senior Principal Engineer in the compiler team at Intel Corporation. He received his Ph.D. from the University of Illinois at Urbana-Champaign in 1992.
## Contents

### Programming Models

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchical Computation in the SPMD Programming Model</td>
<td>3</td>
</tr>
<tr>
<td>Amir Kamil and Katherine Yelick</td>
<td></td>
</tr>
<tr>
<td>Porting Applications with OpenMP Using Similarity Analysis</td>
<td>20</td>
</tr>
<tr>
<td>Wei Ding, Oscar Hernandez, Tony Curtis, and Barbara Chapman</td>
<td></td>
</tr>
</tbody>
</table>

### Tasks

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task-Aware Optimization of Dynamic Fractional Permissions</td>
<td>39</td>
</tr>
<tr>
<td>Christoph M. Angerer</td>
<td></td>
</tr>
<tr>
<td>Near Optimal Work-Stealing Tree Scheduler for Highly Irregular Data-Parallel Workloads</td>
<td>55</td>
</tr>
<tr>
<td>Aleksandar Prokopec and Martin Odersky</td>
<td></td>
</tr>
<tr>
<td>OpenCL Task Partitioning in the Presence of GPU Contention</td>
<td>87</td>
</tr>
<tr>
<td>Dominik Grewe, Zheng Wang, and Michael F.P. O’Boyle</td>
<td></td>
</tr>
</tbody>
</table>

### Heterogeneous Computing

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiling a High-Level Directive-Based Programming Model for GPGPUs</td>
<td>105</td>
</tr>
<tr>
<td>Xiaonan Tian, Rengan Xu, Yonghong Yan, Zhifeng Yun,</td>
<td></td>
</tr>
<tr>
<td>Sunita Chandrasekaran, and Barbara Chapman</td>
<td></td>
</tr>
<tr>
<td>Separate Compilation in a Language-Integrated Heterogeneous Environment</td>
<td>121</td>
</tr>
<tr>
<td>Mike Murphy, Jaydeep Marathe, Girish Bharambe, Sean Lee,</td>
<td></td>
</tr>
<tr>
<td>and Vinod Grover</td>
<td></td>
</tr>
<tr>
<td>Parametric GPU Code Generation for Affine Loop Programs</td>
<td>136</td>
</tr>
<tr>
<td>Athanasios Konstantinidis, Paul H.J. Kelly, J. Ramanujam,</td>
<td></td>
</tr>
<tr>
<td>and P. Sadayappan</td>
<td></td>
</tr>
</tbody>
</table>

### Power

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSCAR Compiler Controlled Multicore Power Reduction on Android Platform</td>
<td>155</td>
</tr>
<tr>
<td>Hideo Yamamoto, Tomohiro Hirano, Kohei Muto, Hiroki Mikami,</td>
<td></td>
</tr>
<tr>
<td>Takashi Goto, Dominic Hillenbrand, Moriyuki Takamura,</td>
<td></td>
</tr>
<tr>
<td>Keiji Kimura, and Hironori Kasahara</td>
<td></td>
</tr>
</tbody>
</table>
Folklore Confirmed: Compiling for Speed = Compiling for Energy ....... 169
  Tomofumi Yuki and Sanjay Rajopadhye

Debugging

Effectively Recognize Ad hoc Synchronizations with Static Analysis....... 187
  Le Yin

AntSM: Efficient Debugging for Shared Memory Parallel Programs ....... 202
  Jae-Woo Lee and Samuel P. Midkiff

DRIFT: Decoupled CompileR-Based Instruction-Level Fault-Tolerance .... 217
  Konstantina Mitropoulou, Vasileios Porpodas, and Marcelo Cintra

Algorithms

Optimizing the LU Factorization for Energy Efficiency on a Many-Core
Architecture ................................................................. 237
  Elkin Garcia, Jaime Arteaga, Robert Pavel, and Guang R. Gao

An Input-Adaptive Algorithm for High Performance Sparse Fast
Fourier Transform .......................................................... 252
  Shuo Chen and Xiaoming Li

Caches

Aligned Scheduling: Cache-Efficient Instruction Scheduling
for VLIW Processors ..................................................... 275
  Vasileios Porpodas and Marcelo Cintra

Compile Time Modeling of Off-Chip Memory Bandwidth for Parallel Loops . 292
  Munara Tolubaeva, Yonghong Yan, and Barbara Chapman

Compiler Optimizations for Non-contiguous Remote Data Movement ....... 307
  Timo Schneider, Robert Gerstenberger, and Torsten Hoefler

Transactional Memory

Combining Lock Inference with Lock-Based Software Transactional Memory . 325
  Stefan Kempf, Ronald Veldema, and Michael Philippsen

Speculative Execution of Parallel Programs with Precise Exception Semantics
on GPUs ........................................................................... 342
  Akihiro Hayashi, Max Grossman, Jisheng Zhao, Jun Shirako,
  and Vivek Sarkar

Author Index ..................................................................... 357