MONDAY, April 14, 2008

Workshops 1-13

TUESDAY, April 15, 2008

Opening Session - Chair Viktor Prasanna

Keynote Speech: Translational Research Design Templates, Grid Computing, and HPC
Joel Saltz
Department of Biomedical Informatics
The Ohio State University

Abstract: Design templates that involve discovery, analysis, and integration of information resources commonly occur in many scientific research projects. In this paper we present examples of design templates from the biomedical translational research domain and discuss the requirements imposed on Grid middleware infrastructures by them. Using caGrid, which is a Grid middleware system based on the model driven architecture (MDA) and the service oriented architecture (SOA) paradigms, as a starting point, we discuss architecture directions for MDA and SOA based systems like caGrid to support common design templates.

Session 1: Algorithms - Scheduling
chair Anne Benoit

Session 2: Applications - General Applications
chair Srinivas Aluru

Session 3: Architecture - Input/Output
chair Douglas Thain

Session 4: Software - Redundancy and Faults
chair Phuong Ha Hoai

Session 5: Algorithms - Numerical Algorithms
chair Padma Raghavan

Session 6: Applications - P2P Systems Architecture
chair Weikuan Yu

Session 7: Architecture - Multi-core
chair Allan Gottlieb

Session 8: Software - Implementing Message Passing
chair Jesper Larsson Traeff

Session 9: Algorithms - P2P and Overlay Networks
chair Eddy Caron

Session 10: Applications - Grids
chair Gagan Agrawal

Session 11: Architecture - Supercomputing/SIMD
chair Martin Schulz

All Symposium Tutorial: Open64 Compiler infrastructure for emerging multicore/manycore architecture
Presenters:
Sun C. Chan, SimpInano Inc.
Guang R. Gao, University of Delaware
Barbara Chapman, University of Houston
Tony Linthicum, Qualcomm Inc.
Anshuman Dasgupta, Qualcomm Inc.

Abstract: Open64 was originally developed by SGI and released as the MIPSpro compiler. It has been well recognized as an industrial-strength production compiler for high-performance computing. It includes advanced
interprocedural optimizations, loop nest optimizations, global scalar optimizations, and code generation with advanced global register allocation and software pipelining. It was open-sourced in 2000 after it was retargeted to the Itanium processor. Now, Open64 is accepted by many compiler researchers as a good infrastructure for research on new compiler optimizing technologies, especially the for the emerging multi-core/many-core architecture.

WEDNESDAY, April 16, 2008

Opening Session - Chair Paul Kelly

Keynote Speech: Programming Models for Petascale to Exascale
Katherine Yelick
NERSC Division Director, Lawrence Berkeley National Lab and EECS Department
University of California at Berkeley

Abstract: Multiple petascale systems will soon be available to the computational science community and will represent a variety of architectural models. These high-end systems, like all computing platforms, will have an increasing reliance on software-managed on-chip parallelism. These architectural trends bring into question the message-passing programming model that has dominated high-end programming for the past decade. In this talk I will describe some of the technology challenges that will drive the design of future systems and their implications for software tools, algorithm design, and application programming. In particular, I will show a need to consider models other than message passing as we move towards massive on-chip parallelism. I will talk about a class of Partitioned Global Address Space (PGAS) languages, which are an alternative to both message passing models like MPI and shared memory models like OpenMP. PGAS languages offer the possibility of a programming model that will work well across a wide range of shared memory, distributed memory, and hybrid platforms. Some of these languages, including UPC, CAF and Titanium, are based on a static model of parallelism, which gives programmers direct control over the underlying processor resources. The restricted nature of the static parallelism model in these languages has advantages in terms of implementation simplicity, analyzability, and performance transparency, but some applications demand a more dynamic execution model, similar to that of Charm++ or the recently developed HPCS languages (X10, Chapel, and Fortress). I will describe some of our experience working with both static and dynamically managed applications and some of the research challenges that I believe will be critical in developing viable programming techniques for future systems.

Plenary Session - Best Papers
chair Yves Robert

Session 12: Algorithms - Communication Algorithms
chair Marina Papatriantafilou

Session 13: Applications - P2P Structure
chair Ali Hurson

Session 14: Architecture - Power/SMT/ILP
chair David Kaeli

Session 15: Software - Tuning and Performance
chair Bronis de Supinski

Plenary Session - Panel Discussion
How to avoid making the same mistakes all over again or... how to make the experiences of the parallel processing communities useful for the multi/many-core generation
Moderator: Jesper Larsson Traeff, IT Research Division, NEC Laboratories Europe
Panelists:
• Hideharu Amano, Keio University
• Anwar Ghuloum, Intel
• John Gustavson, Clearspeed
• Keshav Pingali, University of Austin, Texas
• Vivek Sarkar, Rice University, Texas
• Uzi Vishkin, U of Maryland, Institute for Advanced Computer Studies
• Kathy Yelick, University of Berkeley, California
Abstract: It is timely for the parallel-processing community to take stock: What does the community have to offer the upcoming generation that will have to deal with parallelism for a much broader range of applications? What are the fundamental paradigms and techniques of the past? How can these be most effectively conveyed, and to whom? Which were the mistakes and wrong turns of the past? How can repetition be avoided? Which problems remain unsolved, and what are the major, new challenges?

Banquet and Invited Speech
Chair - Jie Wu
Speech: Computational Thinking and Thinking About Computing
Jeannette Wing
Carnegie Mellon University and National Science Foundation Directorate for Computer and Information Science and Engineering
Abstract: My vision for the 21st Century: Computational thinking will be a fundamental skill used by everyone in the world. To reading, writing, and arithmetic, let's add computational thinking to every child's analytical ability. Computational thinking has already influenced other disciplines, from the sciences to the arts. The new NSF Cyber-enabled Discovery and Innovation initiative in a nutshell is computational thinking for science and engineering. Realizing this vision gives the field of computing both exciting research opportunities and novel educational challenges. The field of computing is driven by technology innovation, societal demands, and scientific questions. We are often too easily swept up with the rapid progress in technology and the surprising uses by society of our technology, that we forget about the science that underlies our field. In thinking about computing, I have started a list of "Deep Questions in Computing," with the hope of encouraging the community to think about the scientific drivers of our field.

THURSDAY, April 17, 2008

Opening Session - Chair Cynthia Philips
Keynote Speech: Looking at Data
Dro Feitelson
School of Computer Science & Engineering
The Hebrew University of Jerusalem
Abstract: Collecting and analyzing data lies at the basis of the scientific method: findings about nature usher new ideas, and experimental results support or refute theories. All this is not very prevalent in computer science, possibly due to the fact that computer systems are man made, and not perceived as a natural phenomenon. But computer systems and their interactions with their users are actually complex enough to require objective observations and measurements. We'll survey several examples related to parallel and other systems, in which we attempt to further our understanding of architectural choices, system evaluation, and user behavior. In all the cases, the emphasis is not on heroic data collection efforts, but rather on a fresh look at existing data, and uncovering surprising, interesting, and useful information. Using such empirical information is necessary in order to ensure that systems and evaluations are relevant to the real world.

Session 16: Algorithms - Theory
chair Frédéric Vivien

Session 17: Applications - P2P Reliability and Trust
chair Loris Marchal

Session 18: Architecture - Networks
chair D.K. Panda

Session 19: Software - Language Features and Implementation
chair Xin Yuan

Session 20: Algorithms - Fault Tolerance
chair Stephen Scott

Session 21: Applications - Sensors
chair Ricky Kwok

Session 22: Applications - Web Applications
chair Sukwoo Kang

Session 23: Software - Resource Management and Scheduling
chair Xin Yuan
Commercial Tutorial – Automated Management of Hierarchical Memories on Multicore Processors

Presenter: William Lundgren, Gedae, Inc.

Abstract: Multicore processors present new programming challenges even to those with experience programming parallel and distributed systems. Gedae offers improved productivity and an expanded developer pool for these architectures by automating many of the difficult and tedious issues such as threading, deadlock avoidance, planning of memory use, and runtime observability. Gedae is able to address these issues will still creating highly efficient applications through the automatic incorporation of target-optimized compute kernels and minimal impact of the Gedae scheduler during runtime. The focus of this tutorial is one of those challenges heightened by the advent of multicores - the management of hierarchical memories. Because multiple cores are being brought together on the limited real estate of a single chip, there is limited room to provide core-specific memory, bringing about programming challenges for the software developer. An example is the Cell Broadband Engine (Cell/B.E.) processor. The Cell/B.E. processor combines 8 Synergistic Processing Elements (SPEs) with one Power Processing Element (PPE). The SPEs each have a small, 256 kB local storage, while the system has a larger monolithic memory available to all PEs. Accessing the system memory from the SPEs must utilize a single memory interface with limited bandwidth compared to the Element Interconnect Bus (EIB) between the SPEs. Programming this hierarchical memory involves manual management of the SPEs’ local storage, overlapping of memory puts and gets with computation, and special consideration of alignment issues to provide high performance. While the Cell/B.E. memory structure presents special programming considerations, other multicores also utilize hierarchical memory structures, such as the use of core-specific multilayered cache on Intel Core 2 and Tilera Tile64 processors. Gedae is a programming language, compiler, and analysis tools that provide a method for specifying the use of hierarchical memory and automating the use of these memories. This tutorial will introduce the concepts of managing hierarchical memories on multicore processors, discuss how those issues affect programming the processors, illustrate Gedae’s solution for programming these memories, and walk through example applications that show how Gedae automatically manages these issues.

FRIDAY, April 18, 2008

Workshops 14-22