

Tutorial Proposal to Micro-36

Title:

=====

Open Research Compiler (ORC): Proliferation of Technologies and Tools

Abstract:

=====

Open Research Compiler (ORC) has been well adopted by the research community for compiler-related research since its inception in Jan 2002. Continuing the drive to outstanding performance and functionality, a new release, ORC 2.1, is planned in the summer of 2003, and the tutorial will provide an overview of the status and features in this release. Different from the past ORC tutorials, however, this tutorial will emphasize on a number of projects and tools proliferated from ORC. These tools and experiences contributed by different user groups can benefit the community by enabling exploration and quick prototyping of new technologies.

A set of alias and data dependence profiling tool allows ORC to overcome the traditional conservatism of static alias and data dependence analyses and to augment such analysis results with probabilistic measures. This technology has led to the development of innovative speculative analysis and optimization techniques. Pin, a tool for user-defined dynamic instrumentation, will also be available along with ORC. Similar to the functionality of ATOM, Pin, which is compiler independent, instruments an Itanium/Linux program while it is running. Pin provides an excellent mechanism to characterize program behaviors and to enable many post-link time optimizations.

Speculative Parallel Threading (SPT) is a compiler-driven approach to exploit speculative thread-level parallelism in single threaded applications. A compiler framework to automatically identify, create, and optimize speculative parallel threads has been built on ORC and much of the framework is applicable to various models of speculative parallel thread execution. Unified Parallel C (UPC) is an important extension to C for scientific computing. UPC has drawn increasing attention in the research community, and it has been implemented on ORC. The discussion on SPT and UPC will provide the insights on extending the compiler for various types of research projects.

Proposed outline of the tutorial:

=====

- I. Overview of ORC
 - Summary of key ORC features

- The latest release: ORC 2.1
 - Status
 - Functionality
 - Performance of ORC on Itanium and Itanium 2
 - Upcoming new features
 - Recent research activities on Open64/ORC

II. Enabling Tools

- Alias and data dependence profiling
 - Motivation
 - Approach
 - Application
 - Speculative analysis and optimizations
 - Experimental results
- PIN (tool for user-defined dynamic instrumentation of IPF/Linux programs)
 - Overview and the underlying mechanism
 - Instrumentation API
 - Applications
 - Demo

III. Proliferation of ORC

- Speculative Parallel Threading (SPT)
 - Execution models
 - Compiler framework for speculative thread parallelization
 - Experimental results
- Unified Parallel C (UPC): parallel extension to C for scientific computing
 - Language extensions
 - Overview: challenges of src to src translation and the system
 - Optimizations and preliminary results

Overview of ORC:

=====

The objective of the Open Research Compiler (ORC) project is to provide a leading open source Itanium compiler infrastructure to the compiler and architecture research community. This common and open infrastructure would encourage and facilitate compiler and architecture research. The design of ORC stresses the following: compatibility to other open source tools, robustness of the entire infrastructure, flexibility and modularity for quick prototyping of novel ideas, and leading performance among the Itanium open source compilers. ORC is currently delivering superior performance comparable to the best performing production compilers on Itanium/Linux.

Three versions of ORC have been released to the open source community

(<http://ipf-orc.sourceforge.net/>) since January, 2002. ORC inherits many state-of-the-art compilation and optimization technologies from the base of Open64 (Pro64) open source compiler. In addition, ORC has largely redesigned the code generation phase in the two aspects: new optimizations to better utilize the Itanium architectural features and new features to facilitate future research. The new Itanium optimizations include global instruction scheduler integrated with a finite-state-automaton-based resource management, control and data speculation with recovery code generation, if-conversion, and predicate analysis. The research-enabling features include region-based compilation, a rich set of profiling feedback and support, and parameterized machine descriptions.

The ORC project would like to invite researchers to explore all aspects of compiler and architecture research to base on ORC, e.g. performance-driven optimizations, thread-level parallelism, co-design of software and hardware features, power management, language-based security, co-design of static and dynamic compilation, optimizations for memory hierarchies, etc.

Preferred length:

=====

Half day (about 4 hours)

List of co-organizers:

=====

- Roy Ju (MRL, Intel)
- Pen-Chung Yew (Univ. of Minnesota)
- Ruiqi Lian (ICT, CAS)
- Lixia Liu (ICRC, Intel)
- Tin-fook Ngai (MRL, Intel)
- Robert Cohn (MMDC, Intel)
- Costin Iancu (Lawrence Berkeley Lab)

Corresponding person:

=====

Roy Ju (roy.ju@intel.com, 408-765-8401)

Bio:

Roy Ju is a senior researcher at the Programming System Lab in the Microprocessor Research Labs, Intel Corp. He is currently the manager and

architect of an IA-64 open source research compiler, which aims at providing an infrastructure for compiler and architecture research on IA-64 to the research and open source communities. His primary research interests include compiler optimization, optimization for memory hierarchy, software power management, program analysis, computer architecture, and parallel processing. Prior to joining in Intel, he was with the Hewlett-Packard Company from 1994 to 1999, and he was a project lead in designing and developing an optimizing compiler for IA-64. He worked at IBM from 1992 to 1994 in developing a then state-of-the-art Fortran 90 optimizing compiler. He received his B.S. degree in Electrical Engineering from National Taiwan University in 1984. He received his M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Texas at Austin in 1988 and 1992, respectively. He currently holds 8 U.S. patents and has published more than 40 journal and conference papers in various areas, including array language optimizations, compilation for instruction-level parallelism, cache optimization, coarse-grained parallelization, etc. He has served on the program committees of a number of conferences, such as MICRO-33, MICRO-35, PLDI '01, and CGO '03.

Pen-Chung Yew is the William Norris Land-Grant Chair Professor and the Head of the Department of Computer Science and Engineering, University of Minnesota. Previously, he was an Associate Director of the Center for Supercomputing Research and Development at the University of Illinois at Urbana-Champaign. From 1991 to 1992, he served as the Program Director of the Microelectronic Systems Architecture Program in the Division of Microelectronic Information Processing Systems at the National Science Foundation, Washington, D.C.

Pen-Chung Yew is an IEEE Fellow. He is currently the Editor-in-Chief of the IEEE Trans. on Parallel and Distributed Systems. He has served on the program committee of various conferences. He also served as a co-chair of the 1990 International Conference on Parallel Processing, a general co-chair of the 1994 International Symposium on Computer Architecture, the program chair of the 1996 International Conference on Supercomputing, and a program co-chair of the 2002 International Conference on High Performance Computer Architecture.

He received his PhD from the University of Illinois at Urbana-Champaign, MS from University of Massachusetts at Amherst, and BS from National Taiwan University. His research interests include high-performance multiprocessor system design, parallelizing compilers, computer architecture, runtime dynamic compilation, and performance evaluation.

Ruiqi Lian is an associate professor in the Advanced Compiler Technology Lab at the Division of Computer System and Architecture, Institute of Computing

Technology, Chinese Academy of Sciences. She is currently working in a project of developing the code generator for network processing systems. Prior to that, she was a lead developer of an IA-64 open source research compiler. She received her Ph.D. in Computer Sciences from the Institute of Computing Technology, Beijing, P. R. China in 2001 and her B.S. in Computer Sciences from the Northern Jiaotong University, Beijing, P. R. China in 1996. Her research interests include instruction-level-parallelism, machine-dependent optimization and common compiler infrastructure.

Lixia Liu is a researcher at Intel China Research Center. She is currently working on an IA64 open source research compiler. She received her B.S. Degree in Computer Communication from Beijing University of Posts and Telecommunications in 1999 and her M.S. Degree in Computer Science from Peking University in 2002. From 2000 to 2002, she was a developer for the open research compiler at Institute of Computing Technology, Chinese Academy of Sciences. Her current research interests include computer architecture and compiler optimization.

Tin-Fook Ngai is a senior researcher at the Programming Systems Lab in the Microprocessor Research Labs, Intel Corp. He is currently leading a team doing research work in speculative parallel threading compilation. He received his Ph.D. Degree in Electrical Engineering from Stanford University in 1992. Prior to joining Intel, he was a member of technical staff at Hewlett-Packard Labs from 1991 to 1993, where he participated in the architecture definition studies and the corresponding advanced compiler design and prototyping for the SuperWorkstation project whose architecture later evolved into the Intel-HP IA64 architecture. From 1994 to 1999, he was an assistant professor in Computer Science at the Hong Kong University of Science and Technology, where he taught compilers, computer architectures and operating systems, and led his students to develop a real-time multimedia operating system and a real-time parallel video server system. His current research interests include speculative thread-level parallelization and optimization, speculation support in compilers, new computer architectures/systems, and run-time systems.

Robert Cohn is a Principal Engineer at Intel, where he works on just in time compilation, dynamic instrumentation, and post link optimization. Previously, he worked for Digital and Compaq where he implemented profile guided optimization in the product compiler and Om post link optimizer for Alpha. He was a lead developer for the Spike post link optimizer. Robert received a Ph.D. in Computer Science from Carnegie Mellon in 1992.

Costin Iancu is a researcher working for the Future Technologies Group at Lawrence Berkeley National Laboratory. He is currently working on compiler and run-time optimization techniques for the implementation of Global Address

Space Languages (UPC). Prior to that he worked for Green Hills Software where he developed code generation optimizations for the ARM/Thumb processor family. He is also an ex-member of the Object Oriented SUIF (OSUIF) compiler research effort at University of California Santa Barbara. He received his Ph.D. in Computer Science from UC Santa Barbara and a M.S. in Computer Science from Politehnica University of Bucharest. His current research interests include program analysis and optimizations for parallel languages and using re-configurable computing elements (FPGAs) as application specific hardware accelerators.