FRACTAL – Fall 2007
Front Range Architecture Compilers Tools and Languages Workshop

Hosted by
University of Colorado at Boulder
Sunday, October 14, 2007

Reviewed by

John Michalakes
CSCI 7900-903: Intro to the PhD program

___

The 2nd Front Range Architecture Compilers Tools and Languages (FRACTAL) workshop (http://www.cs.du.edu/fractal) was held Sunday, October 14th, 2007, in the Bechtel Collaboratory in the Discovery Learning Center on the CU Boulder Campus. The series is held jointly by Colorado State University, The University of Denver, and The University of Colorado. The Fall 2007 workshop was sponsored by Google.

The purpose of the workshop series is to foster discussion of new ideas in the area of programming languages and architectures and to help establish closer ties between computer science and electrical engineering researchers in Front Range locale of Colorado which includes Denver, Boulder, and Fort Collins. The meeting also provided a student presenters with the opportunity to test out and hone thesis or conference material in front of a small, friendly audience.

All of the presentations were informative and interesting and it was a Sunday well spent. In this review I will focus on several of the talks that particularly caught my interest: GRAIL, OS support for multicore pipelining, smashing periodic domains in parallel codes, and managing perturbation.

The first talk, "GRAIL: A Generic Reconfigurable Affine Interconnection Lattice", was given by Prof. Sanjay Rojapadhye in the CS department at CSU. The purpose of the research is to support rapid, reconfigurable network interconnection between increasing numbers of processing elements being fabricated on single dies of silicon. He quoted Dr. Kathy Yellick of U.C. Berkeley: "the processor is the new transistor" as the unit of computing hardware on a chip. Another term he introduced, FPOAs, is a variation on Field Programmable Gate Arrays (FPGAs) except instead of "G" for "gate", O stands for "object." Thus, like FPGAs, FPOAs are large reconfigurable networks of arithmetic units on the chip – a "sea of ALUs". Prof. Rojapadhye then turned to the issue of programming FPOAs and how languages might help expose the large amounts of parallelism required to make efficient use of this architecture. He focused on a the Affine Control Loop (ACL) class of programs in which loop bounds are known statically and the access patterns into memory are affine functions that can be represented using linear

algebra. He then sketched some programming examples and their representation in as L.A. functions that can be transformed for computation on switched memory architectures. GRAIL, then (the topic of the talk) is a non-blocking circuit-switched multistage interconnection work that is reconfigurable to support temporally changing affine interconnections in ACL programs.

Fourth year PhD student John Giacomoni presented "Operating System Support for Pipeline Parallelism on Multicore Architectures," adapted from his refereed workshop publication in the *2007 Workshop on Operating System Support for Heterogeneous Multicore Architectures (OSHMA),* Brasov, Romania, September 2007.[1] The presentation first gave an overview of the FastForward work to use software pipelining between multiple cores of chip-multiprocessors to perform real-time filtering of network packages to detect and prevent distributed denial-of-service (DDoS) attacks. The issue, of course, is speed – in order to handle a real-time packet stream, processing for each packet must be done in under 672 nanoseconds. Unfortunately, just the pipeline communication overhead for system calls and pthread mutex's consumes more than half that time. Work previous to this presentation involved developing FastForward, a portable software-only framework, for reducing this to under 40 nanoseconds. However, to apply this on real servers, support in the operating system is also required. This talk presented ideas for providing "pipelinable" OS services. CMP hardware can support this by allowing multiple cores in a CMP to be gang scheduled, an idea Giacomoni termed "heterogeneous gang scheduling" so that every stage in the software pipeline, whether in user space or OS kernel space, can be executed with a "zero-stall" guarantee.

The intriguingly titled "Smashing Periodic Domains" was presented by Nissa Osheim and Dave Rostron, both graduate students in the department of Computer Science at Colorado State University. Osheim and Rostron are working with CSU faculty members Rajopadhye and Michelle Strout on methods for improving temporal and spatial locality in parallel programs by applying polyhedral transformations such as tiling and skewing that can reduce or eliminate the cost for communication and serial dependencies in parallel codes. Osheim and Rostron are focusing on data dependencies that result from periodic dependencies – for example, the dependencies between computations on the boundaries of a regular Cartesian grid wrapped around a cylinder or a sphere. One application for this work is the spherical icosahedral climate model being developed by David Randall's group in the Atmospheric Sciences department at CSU. The idea behind "smashing", a term coined by the presenters, is to fold a domain so that the cells representing the opposite edges of periodic boundary end up on the same processors, eliminating the need for communication. Osheim and Rostron presented ideas for smashing periodic boundaries for cylinders, spheres, and tori, as well as some initial ideas for smashing Randall's icosahedral domain (imagine his surprise). I chatted during the breaks with Osheim, Rostron, and Rajopadhye about their ideas and tradeoffs between communication and computation that effect whether to do smashing and other polyhedral code transformations to improve locality or simply pay the price for data movement and interprocessor communication. These offline discussions during FRACTAL led to my being invited to CSU to give an invited CS Department colloquium presentation

[1] Source for reference is John's CV on the CU web pages.

November 26, 2007 on the application of High Performance Computing to Atmospheric Simulation (http://www.cs.colostate.edu/BMAC).

The most free-wheeling and provocative presentation during the FRACTAL workshop came near the end of the day. Prof. Amer Diwan of the University of Colorado at Boulder Computer Science faculty presented "Managing Perturbation", a jeremiad on the pitfalls and perils of measuring and optimizing computer system performance. Misunderstanding the performance characteristics of a system leads to misdirected attempts at optimization that may actually degrade performance. However, measurement of those characteristics perturbs the system being measured. Two sources of perturbation are environmental perturbation – non-determinism, interference from other processes, and network activity – and the cost of the measurements themselves. Diwan took aim at conventional wisdom that the level of perturbation is proportional to the level of measurement effort. He presented results from some simple experiments showing that, depending on the metric, measured performance may be extremely sensitive to even very small amounts of instrumentation in a program. The sensitivities appear both for individual metrics – for example instruction and cycle counts – and for correlations between different factors. The argument was weakened somewhat by Diwan's occasional focus on strawmen in his data: for example, he pointed to extreme sensitivity of L1 miss rates to measurement perturbation but his data also showed that these had very little actual impact on overall program performance, which was considerably less sensitive. But overall, his analysis and conclusions – that "best effort guidelines" are necessary but often not sufficient for capturing the true behavior of a computer system in the face of perturbation – were well considered and important caveats for researchers doing quantitative computer systems performance analysis.