PACT 2006 Tutorial Submission:

A Case Study of a Physically Based Visual Simulation Targeting Emerging Architectures and Programming Models

March 31, 2006

Yahya H. Mirza

Aurora Borealis Software LLC

15817 NE 90th ST Suite E340
Redmond WA, 98052
yahya@aurorasoft.net
(425)-749-7033

Tutorial Title

A Case Study of a Physically Based Visual Simulation Targeting Emerging Architectures and Programming Models

Tutorial Organizer

Yahya H. Mirza
Aurora Borealis Software LLC
15817 NE 90th St. Suite E340
Redmond WA 98052
yahya@aurorasoft.net

Tutorial Abstract

When developing new multicore architectures, parallel system software, multi-level parallel programming languages, or their tools, it's imperative to have a clear understanding of how these technologies will be used in concert to solve a particular application problem. Often times solutions are built based on incremental advancements on existing solutions without a coherent understanding of any particular domain in which these solutions will be used. Consequently, we believe a path to creating useful infrastructure technology is by "spiking" a concrete application domain.

The objective of this half day tutorial is to use a non-trivial example from the computer graphics / special effects domain to illustrate the challenges when utilizing a modern high-performance multicore processor architecture and a new multi-level parallel programming language. Throughout our tutorial, a single physically accurate visual simulation will be used as a point of context. First, a manually parallelized and tuned C implementation of the visual simulation will be illustrated for the CELL Broadband Engine Architecture. Next, a discussion will be initiated on how that same visual simulation can be implemented using Sun Microsystems's Fortress language.

By using a single coherent example, it is our objective to illustrate the patterns that emerge in application domains, and how these patterns can be mapped to new multicore processor architecture and succinctly expressed using a new emerging explicitly parallel language funded by the DARPA HPCS effort. Our proposed application utilizes a large varied number of algorithms. These algorithms include a coupled physics simulation incorporating a stochastic path simulation with a multigrid-based Poisson solver, volume visualization, ray casting, antialiasing filters, and real-time user interaction. Through our tutorial, we hope to provide future multicore processor architects, parallel programming model designers and tool developers a fresh look into the applicability of their work through practical insight from a particular domain.

Outline of Tutorial Content

Time	Presenter	Title
12:55 PM – 1:00 PM	Yahya H. Mirza	Welcome and Tutorial Introduction.
1:00 PM – 2:00 PM	Yahya H. Mirza	A C Based Visual Simulation Targeting the CELL Broadband Engine.
2:00 PM – 3:00 PM	Yahya H. Mirza	A Fortress Based Visual Simulation In Fortress.
3:00 PM – 3:15 PM	***Break***	
3:15 PM – 4:15 PM	George Bosworth	Runtime System Issues for an Applications Perspective
4:15 PM – 5:15 PM	George Bosworth	SIMD, Memory Hierarchy, Stall and Branch Elimination Optimizations.
5:15 PM – 5:30 PM	Mirza / Bosworth	Question / Answer / Discussion.

Presentation Overviews

The Sony, Toshiba, IBM CELL Broadband Engine architecture is a 64 bit PowerPC based, multi-threaded, multi-core, open architecture, supporting multiple synergistic processor units (with SIMD capability), as well as on-chip broadband communication capabilities. The CELL is an early member of a new generation of scalable multicore processor architectures that will soon be ubiquitous from vendors across the industry. Through our example, we hope to provide a holistic understanding of what makes multicore processors like the CELL, different from conventional processors, and how these differences impact software algorithm design and optimization in general.

With respect to emerging programming models, a modern object oriented parallel programming language such as Fortress embodies many different levels of abstractions. These levels of abstractions include for example, the Sun multi-language Portable Runtime, the Fortress programming language, as well as the Fortress runtime libraries. In order for programmers to truly take advantage of Fortress, they must be able to understand and traverse up and down this hierarchy of abstractions. By explaining the "design motifs" embodied in a concrete example or hope is to more clearly illustrate technical innovations embodied in Fortress.

Finally, from a software optimization perspective, when programming multicores processors like the CELL, some software optimizations that are not needed for conventional processor architectures are required for the CELL. For example, since the SPE's local memory is of a modest size (256KB) and is not cache coherent with shared memory, data locality and memory hierarchy optimization techniques become extremely important to writing high performance CELL code. Additionally, the SPE supports a dual-issue pipeline with seven execution units. Therefore, to more effectively utilize the SPE's dual issue pipeline, optimizations such as loop unrolling, instruction reordering, and no-op placement within SPE code sequences can be utilized. Finally, unlike conventional streaming architectures, the SPE can support branching code. Although the SPE does not explicitly support branch prediction hardware, compiler and software supported branch elimination techniques become important optimizations, which can be applied by the CELL programmer to enable efficient SPE code execution. Through our tutorial, we hope to intertwine a discussion on architecture, programming models and how they are utilized through a practical example.

Expected Tutorial Duration

We would like to request a half-day tutorial.

Target Audience

Our tutorial is aimed at individuals interested in developing multicore architectures, runtime systems, and their compilers for emerging parallel programming models better understand how their respective technologies are used. Attendees explicitly interested in the CELL architecture or the Fortress parallel programming language will find our tutorial especially useful. Our tutorial will not be an introduction to physically based visual simulation, although software optimization issues that pertain as to how these simulations might be more efficiently implemented on emerging multicore architectures will be covered in depth. Furthermore, this course is not meant as a primer on parallel processing per say. Rather, our aim is to enable attendees to effectively understand and utilize the parallel processing capabilities introduced by emerging multicore architectures and their new programming models.

Expected Number of Attendees

Due to the timeliness and practicality of our tutorial, we expect 30 attendees, although many more are possible.

Short Biography of Each Speaker / Organizer

Presentation 2: Yahya H. Mirza

Yahya H. Mirza
CEO / CTO
Aurora Borealis Software LLC
yahya@aurorasoft.net

Yahya Mirza's original background was aeronautical engineering and was initially employed by Battelle Labs in Columbus Ohio. After transitioning into the software industry, Yahya spent three years as a visiting researcher in the UIUC Smalltalk Research Group. Through his company, Aurora Borealis Software LLC, Yahya has worked on system software projects for Microsoft, CatDaddy Games, Source Dynamics and Pixar Animation Studios. Since 2002, Yahya has been organizing the Language Runtimes workshops, held at the OOPSLA and Supercomputing conferences. Yahya is currently working on programming models and applications for multi-core processor architectures like the CELL for entertainment, and engineering modeling applications.

Presentation 1: George Bosworth

George Bosworth
.NET CLI Architect
Independent Consultant / Microsoft Retired
georgebosworth@acm.org

George Bosworth has worked on VMs since 1977, most of them for Smalltalk/V while at Digitalk, which he co-founded in 1982. For the last 5 yrs George has been one of the many architects at Microsoft behind the CLR, .Net runtime, .CLI, COM+, COM 2.0 (or whatever it's current name is today). Prior to retiring from Microsoft, George was involved with the Phoenix compiler technology as it relates to the CLR.