Application and Systems Performance: Academic Research Overview

LACSI Priorities and Strategies Workshop 2005

John Mellor-Crummey

Department of Computer Science

Rice University

LACSI

LACSI Compiler and Tools Research Portfolio

Long Term Research Affecting Future HPC Systems

- High-level data-parallel programming systems
- Source-to-source transformation tools
- Open source compilers
- Compilation techniques for SPMD languages
- Low-overhead measurement of large-scale systems
- Performance diagnosis tools

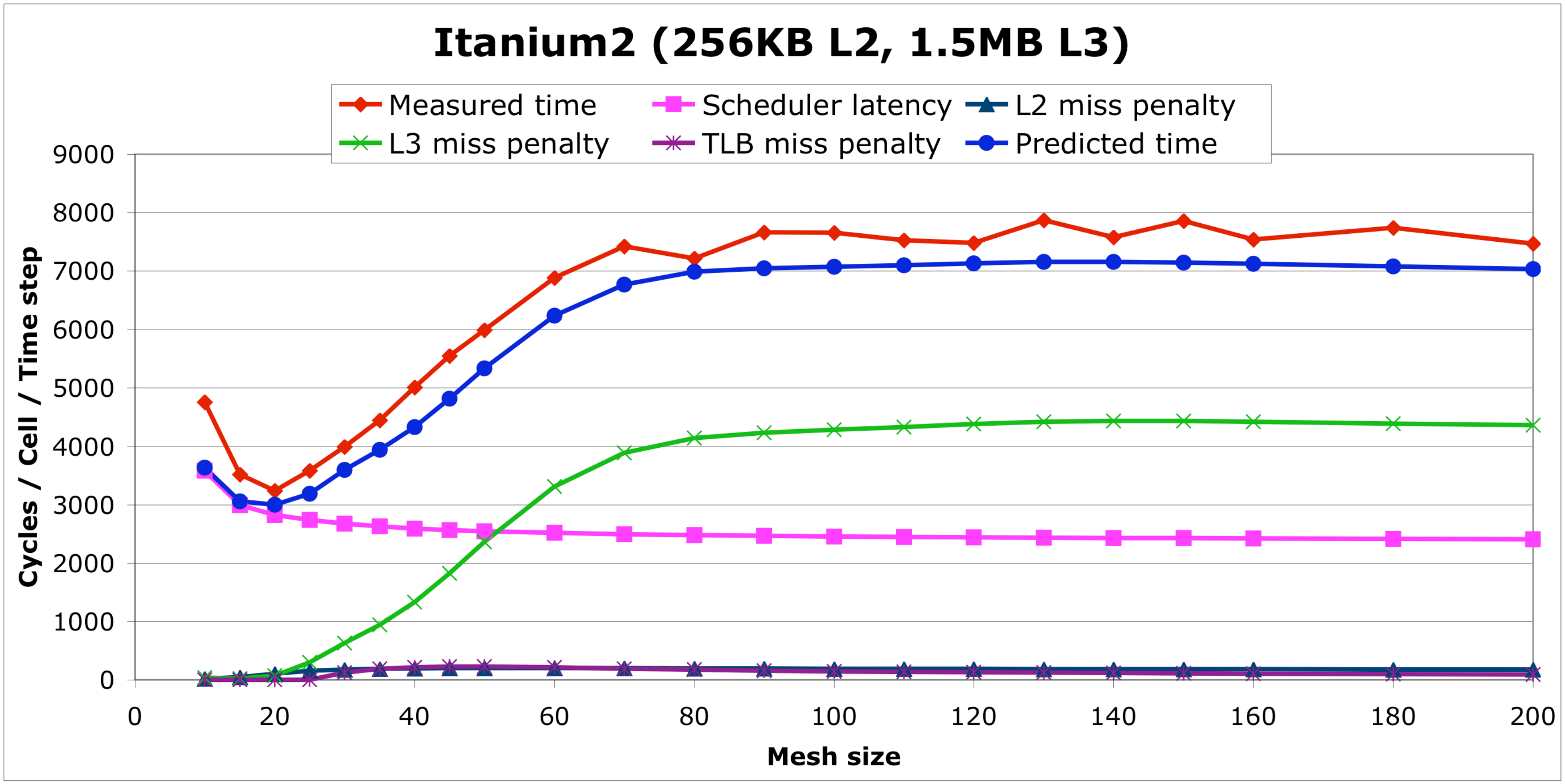
- Compilation for hybrid architectures
- Fundamental compiler algorithms
- Detailed performance modeling of applications
- Hand application of aggressive transformations to important codes

Immediate Impact in Support of ASC Mission Goals

Performance Modeling

- **Understand interplay between node program and microprocessor architecture**
  - measure application-specific factors
    - static analysis
    - dynamic analysis
  - construct models of computation and memory hierarchy performance
    - instruction dependences
    - memory hierarchy miss rates and latencies
  - identify impediments and enablers for high performance
- **Publications**
  - SIGMETRICS 2004: arch. independent modeling node performance
  - SC2004 poster: modeling memory hierarchy behavior of sci. appl.

Execution Behavior: Sweep3D



Predicted from optimized SPARC binaries!

LACSI

Collaboration

- **Joint workshops**
  - Performance and productivity of extreme-scale systems
    - (LACSI Symposium, October 2004)
  - Performance analysis and modeling (Rice, October 2004)
- **Ongoing interactions**
  - Who
    - Lubeck, Fowler, Kennedy, Marin, Mellor-Crummey
  - What
    - use hardware performance counters to measure the impact of memory hierarchy on application performance
    - explore regression and queuing models based on measured data
    - validate Rice predictive modeling of memory hierarchy

Performance Analysis

- Long-term compiler and architecture research requires detailed performance understanding
  - identify sources of performance bottlenecks in complex applications
  - discover automatic strategies for performance improvement
  - understand the mismatch between application needs and architecture capabilities
- Short-term result: Programmer-accessible tools for understanding application performance

Performance Analysis Tools

- **Flat profiling: HPCToolkit**
  - deployed new multiplatform release: Linux, Irix, Tru64
  - deployed process-based profiling on Lightning (bproc cluster)
  - improved binary analysis performance 70-80% for large Linux appl
  - SC04 tutorial: Application performance analysis on Linux
- **Call stack profiling**
  - collect information about where time is spent and call chain context
  - results: a sample-driven call-stack profiler for Alpha
    - data collection overhead proportional to sampling frequency
    - accurately attributes time spent to calling context

      avoids key assumption of gprof: all calls take uniform time
  - ongoing work: retarget to Opteron and x86
  - LACSI 2004 poster; forthcoming MS thesis

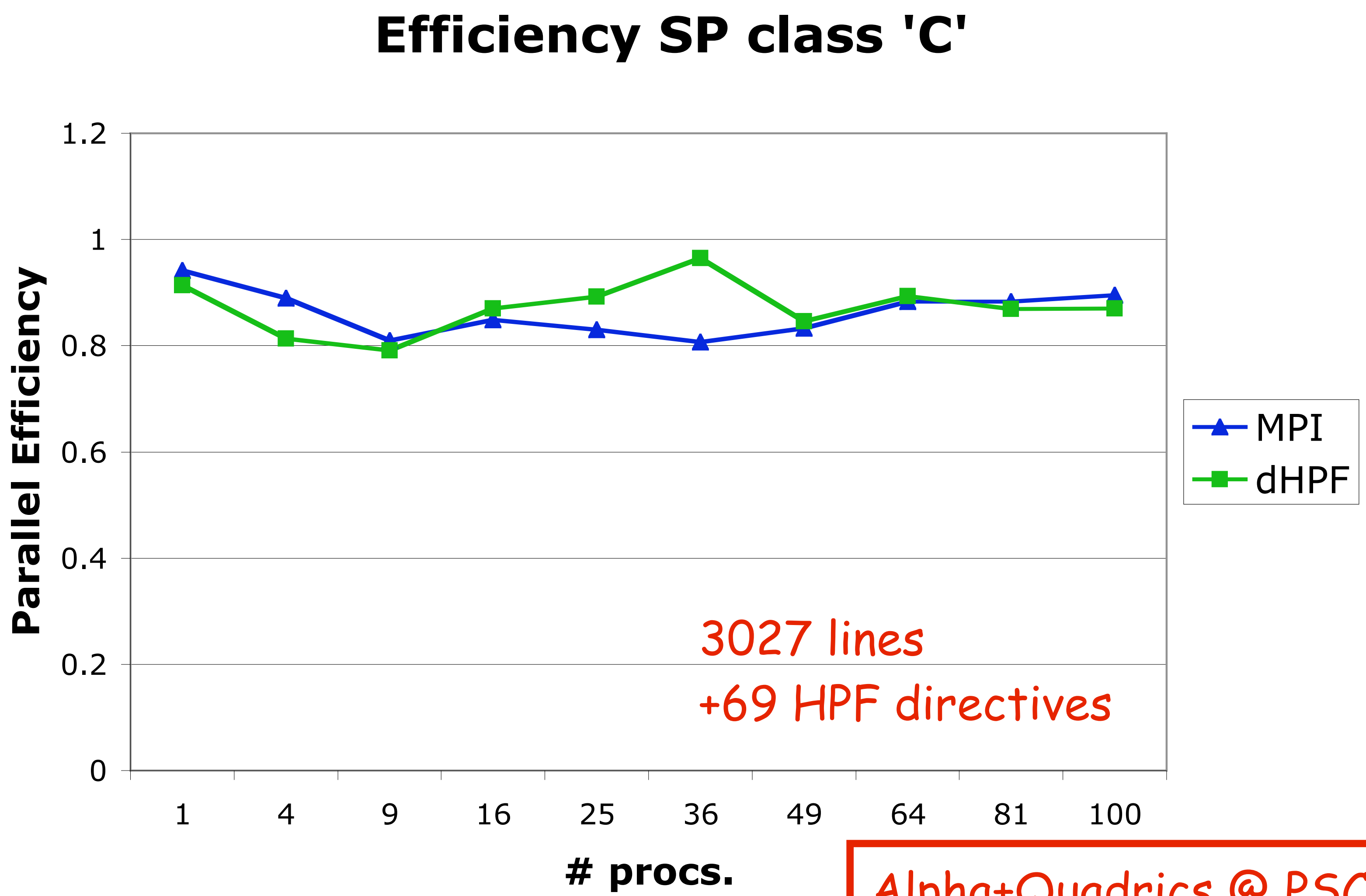
Large Systems and Statistical Sampling

- Want post-mortem analysis and spatio-temporal correlations
- But, for systems with thousands of nodes …
  - complete monitoring: expensive and inaccurate due to delays
  - the law of large numbers starts to apply
    - utilization, bandwidth, latency, and availability estimates
- Population sampling approach
  - select a statistically valid subset of the population/system
  - estimate properties of entire system based on analysis of subset
  - stratified sampling: identify equivalence classes and sample each
- Key sampling factors
  - sample size, represents the cost of the analysis
  - sampling accuracy (precision), where estimates may fail
  - sampling confidence, how often that precision is achieved

Compilers for High-Level Parallel Programming

- Near term
  - compiler technology for SPMD global address space languages
    - multiplatform Co-array Fortran compiler: (Alpha, Itanium, MIPS, Pentium) x (Quadrics, Myrinet, shared memory)
    - prototype compiler delivers performance comparable with MPI
  - papers
    - LCPC 2004 - CAF implementation strategies for shared memory
    - PACT 2004 - multi-platform CAF compiler
    - LACSI 2004 - evaluate CAF implementation of Sweep3D
    - PPOPP 2005 - compare CAF, UPC and MPI performance
- Longer term
  - compiler technology for high-level data parallel languages, e.g. HPF
    - enabling technology for high-productivity parallel programming
  - papers
    - PPOPP 2005 - compiler technology for optimizing communication
    - IPDPS 2005 - HPF study of IMPACT-3D on μproc cluster

HPF vs MPI Efficiency for NAS SP (162^3 size)



Alpha+Quadrics @ PSC

IMPACT-3D

HPF application: Simulate 3D Rayleigh-Taylor instabilities in plasma using TVD

- Problem size: 1024 x 1024 x 2048
- Compiled with HPF/ES compiler
  - 7.3 TFLOPS on 2048 ES processors ~ 45% peak
- Compiled with dHPF on PSC's Lemieux (Alpha+Quadrics)

1334 lines
+45 HPF directives

# procs	relative speedup	GFLOPS	% peak
128	1.0	46.4	18.1
256	1.94	89.9	17.6
512	3.78	175.5	17.4
1024	7.58	352.0	17.2

Open Source Compilers

Critical resources in scientific computing

- Big picture issues
  - GCC is a 1980s design and is showing its age
  - what will replace GCC?
  - will that compiler produce good code for scientific applications?
- Open64/ORC is a strong candidate
  - Well done suite of optimizations, including backend components
  - Full-blown dependence analyzer
  - Somewhat lacking in support for retargeting
- LLVM is another candidate
  - Newer compiler with fewer implemented optimizations
  - Good architecture; strong support for retargeting
  - Support for runtime reoptimization

Led SC04 Workshop on Open Source Open64

Improving Backend Optimization in LLVM

- **Register Allocation**
  - implemented two coloring allocators for LLVM, tested them across the range of supported architectures
  - both improve code performance with respect to the old allocator
  - we will distribute one or both of these allocators (legal issues)
  - both allocators move across architectures with almost no changes
- **Instruction Selection**
  - we intend to build an aggressive scheduler for LLVM
  - coupled with a new register allocator, should make LLVM competitive

Compiler Optimization Research

- High Level
  - automatic tuning
    - LACSI 2004 - automatic empirical tuning for memory hierarchy
  - loop restructuring
    - scalarization
    - vectorization
    - fusion and array contraction
- Low Level
  - LACSI 2004: adaptive compilation
  - removing redundant memory operations
    - discover and remove redundant pointer-based memory operations (up to 40% of all loads; 16% on average)
  - fast techniques for copy coalescing
    - new technique for modeling interferences

      speeds up copy coalescing and live-range identification

LACSI