Compilers and Compiler-based Tools for HPC

John Mellor-Crummey

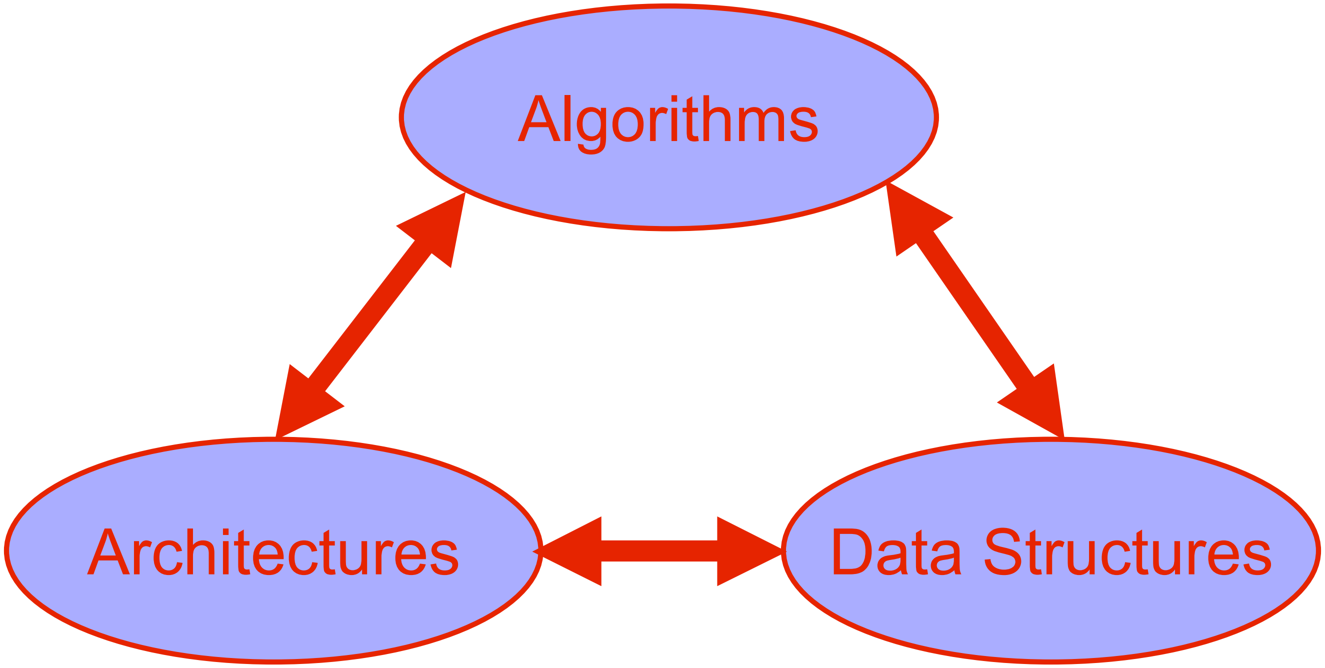
Department of Computer Science

Rice University

http://lacsi.rice.edu/review/2004/slides/Compilers-Tools.pdf

LACSI

High Performance Computing



- Good scalability requires effective parallelizations
- Good performance requires efficient node programs
- Compiler technology can reduce the application development burden

Some ASC HPC Challenges for CS

- Understand interplay between ASC applications and architectures
  - pinpoint performance bottlenecks to guide application tuning
  - evaluate the promise of future architectures
- Support tailoring of applications to target architectures of ASC interest (current and emerging systems)
- Achieve high efficiency on a range of modern processors
- Make it easier to write high-performance programs for scalable parallel systems

Summary: Impact Now and in Future

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 (UPC, CAF)

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

Outline

- Performance analysis
  - HPCToolkit for analyzing node program performance
- Performance modeling
  - understanding performance of node programs
- Compiler technology for parallel languages
  - Co-array Fortran
  - high-level data parallel languages (HPF)
- Open source platforms for delivering compiler technology
- Compiler technology for node programs

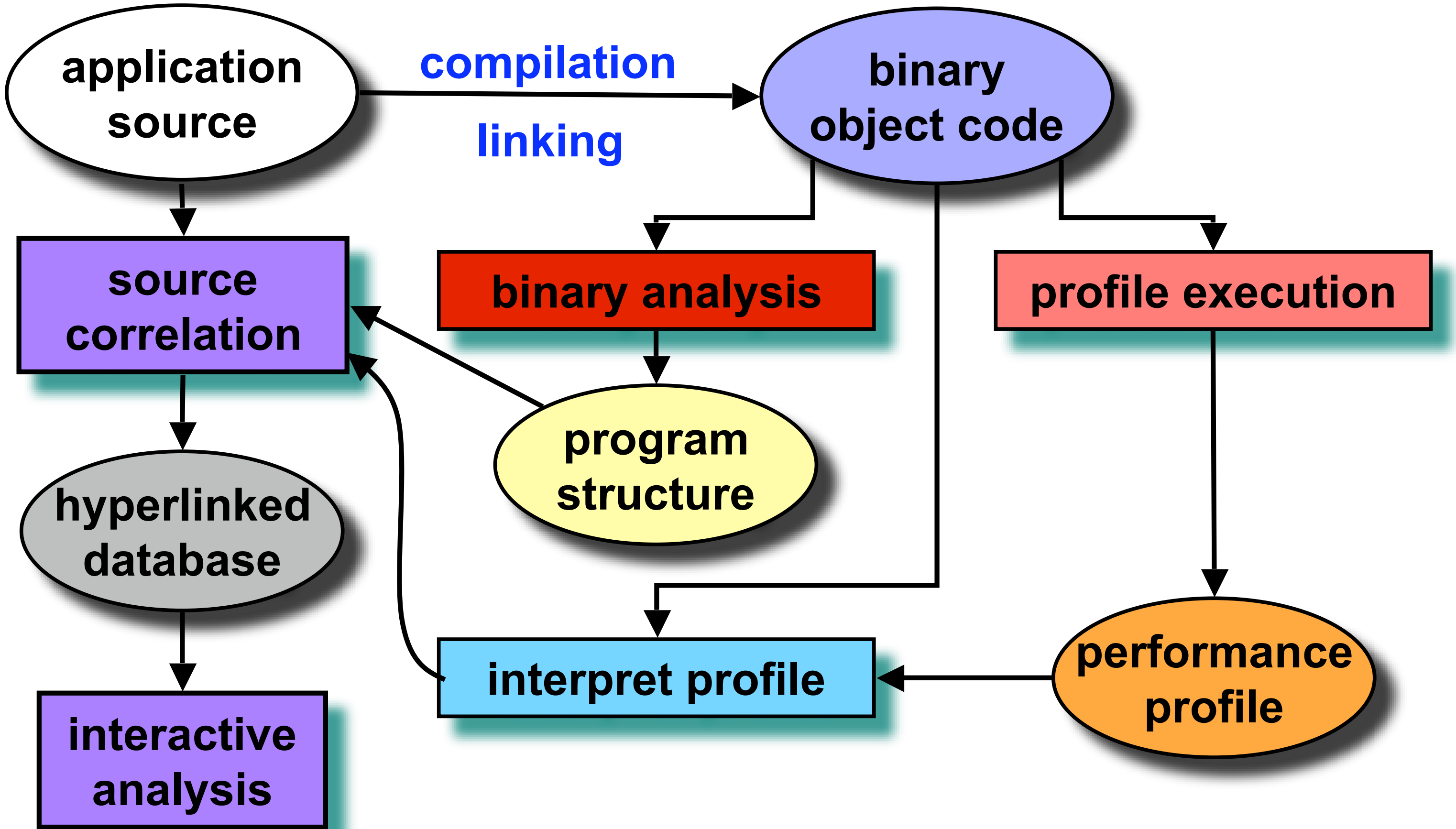
Performance Analysis

- 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

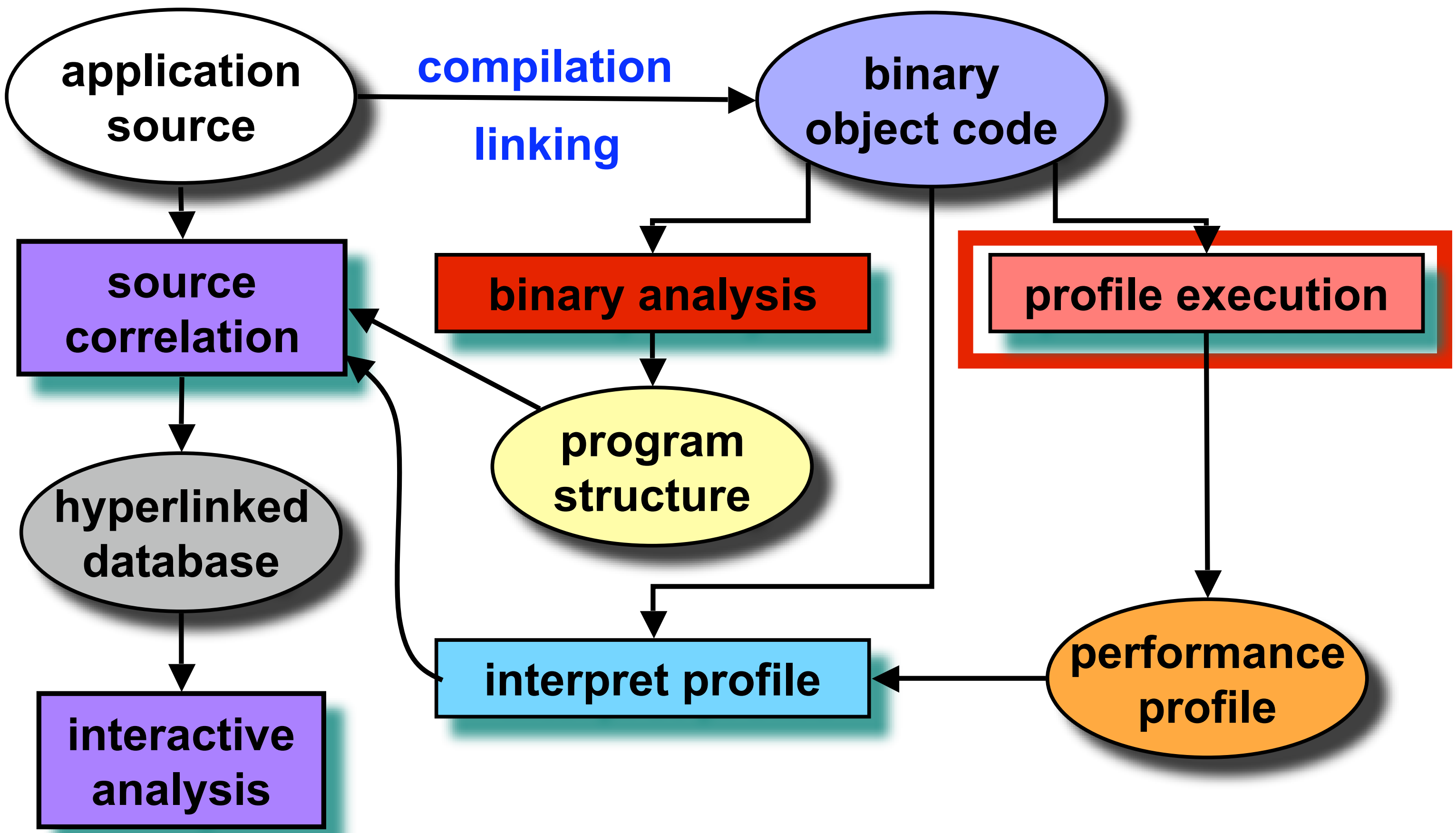
HPCToolkit

- Goal: Effective tools for performance analysis
  - intuitive top-down user interface
  - provide information crucial for analysis and tuning
- Platform, language and compiler independence
  - emphasis on LANL ASC platforms
  - multiple data sources enable cross-platform comparisons
  - extract hierarchical program structure from binaries
    - handle multi-module, multi-language (F77, F9x, C, C++, …)
- Eliminate manual labor from the analyze-tune-run cycle

HPCToolkit System Workflow

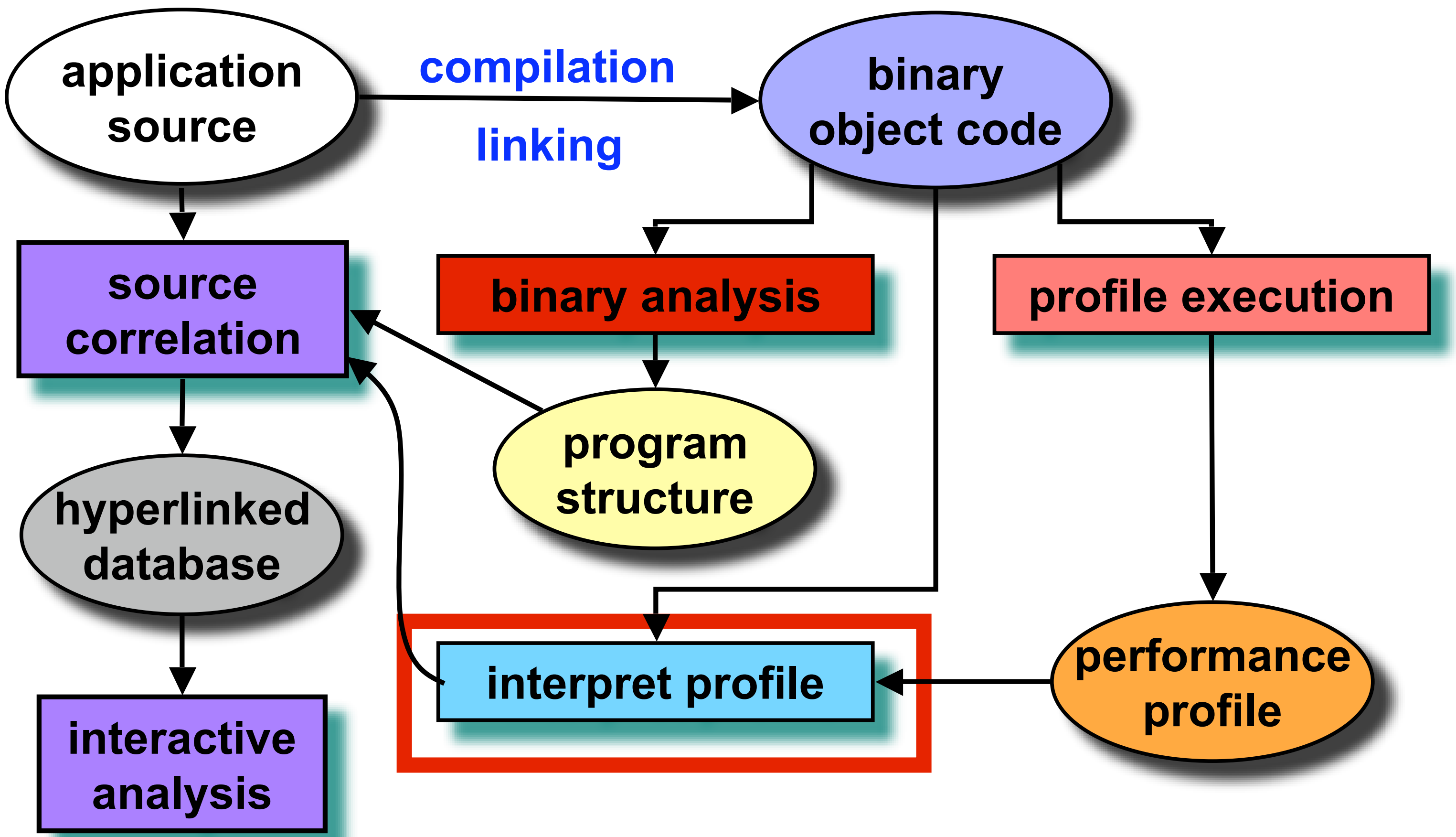


HPCToolkit System Workflow



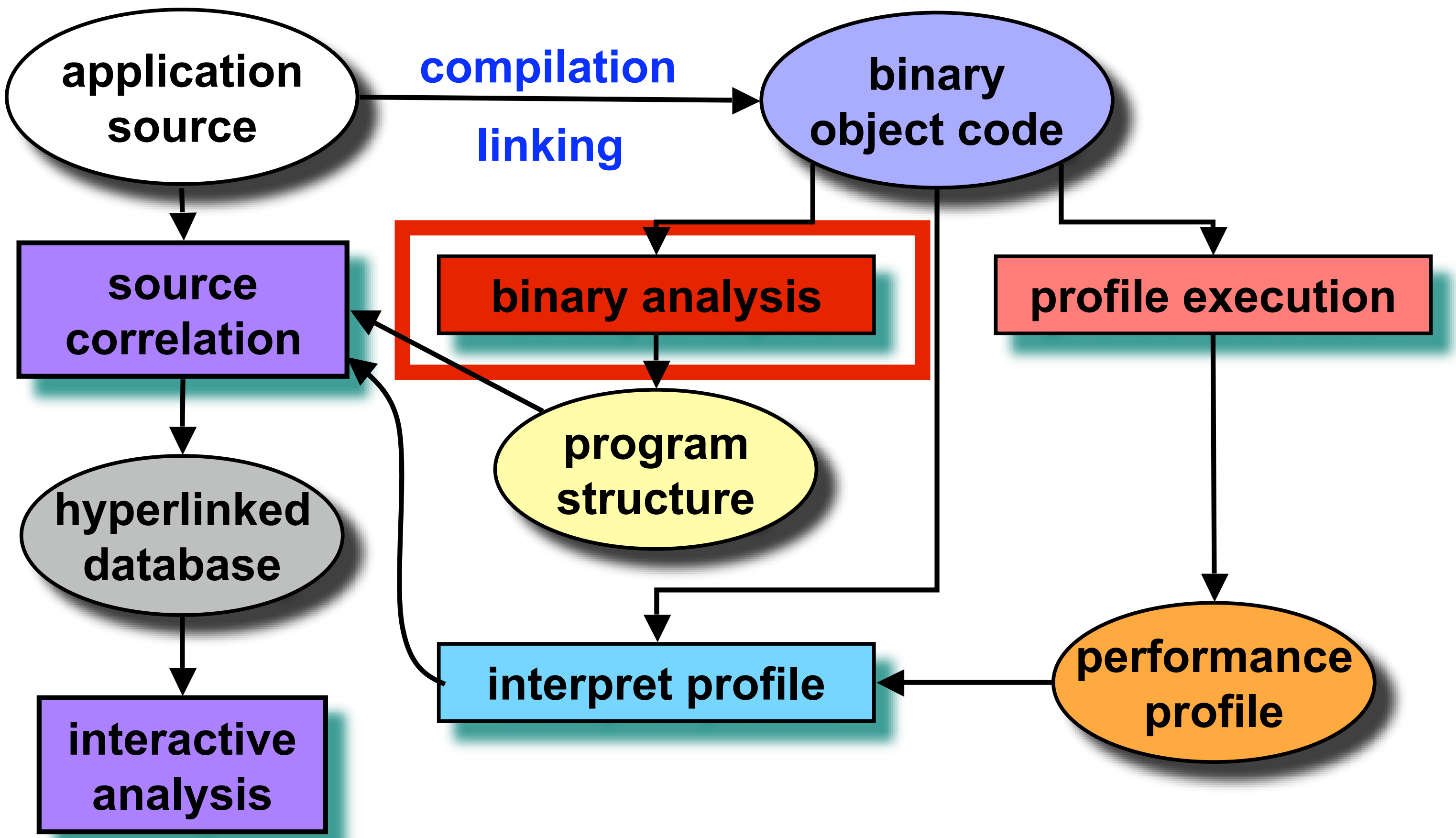
– launch unmodified, optimized application binaries
– collect statistical profiles of events of interest

HPCToolkit System Workflow



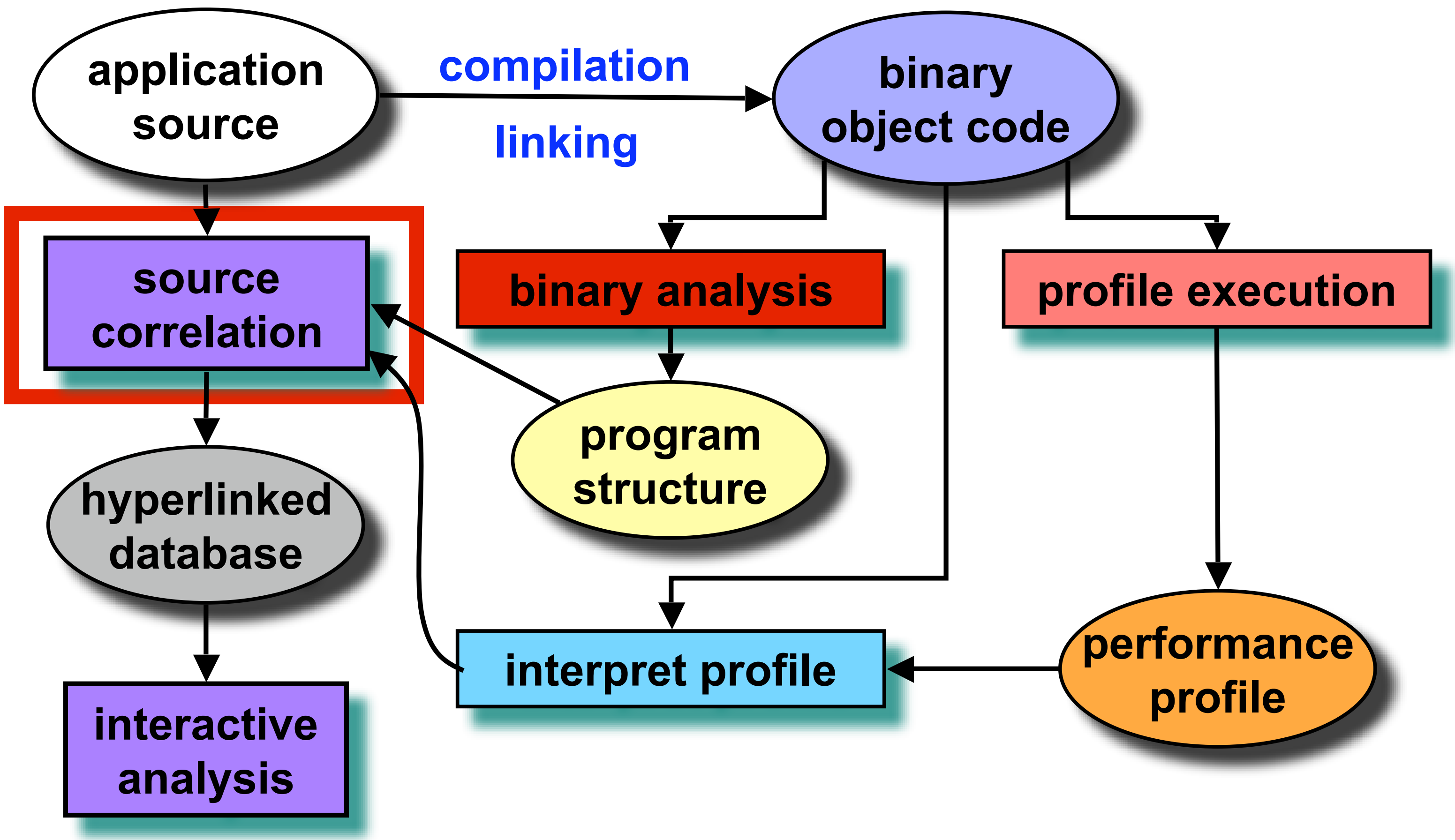
— decode instructions and combine with profile data

HPCToolkit System Workflow



— extract loop nesting information from executables

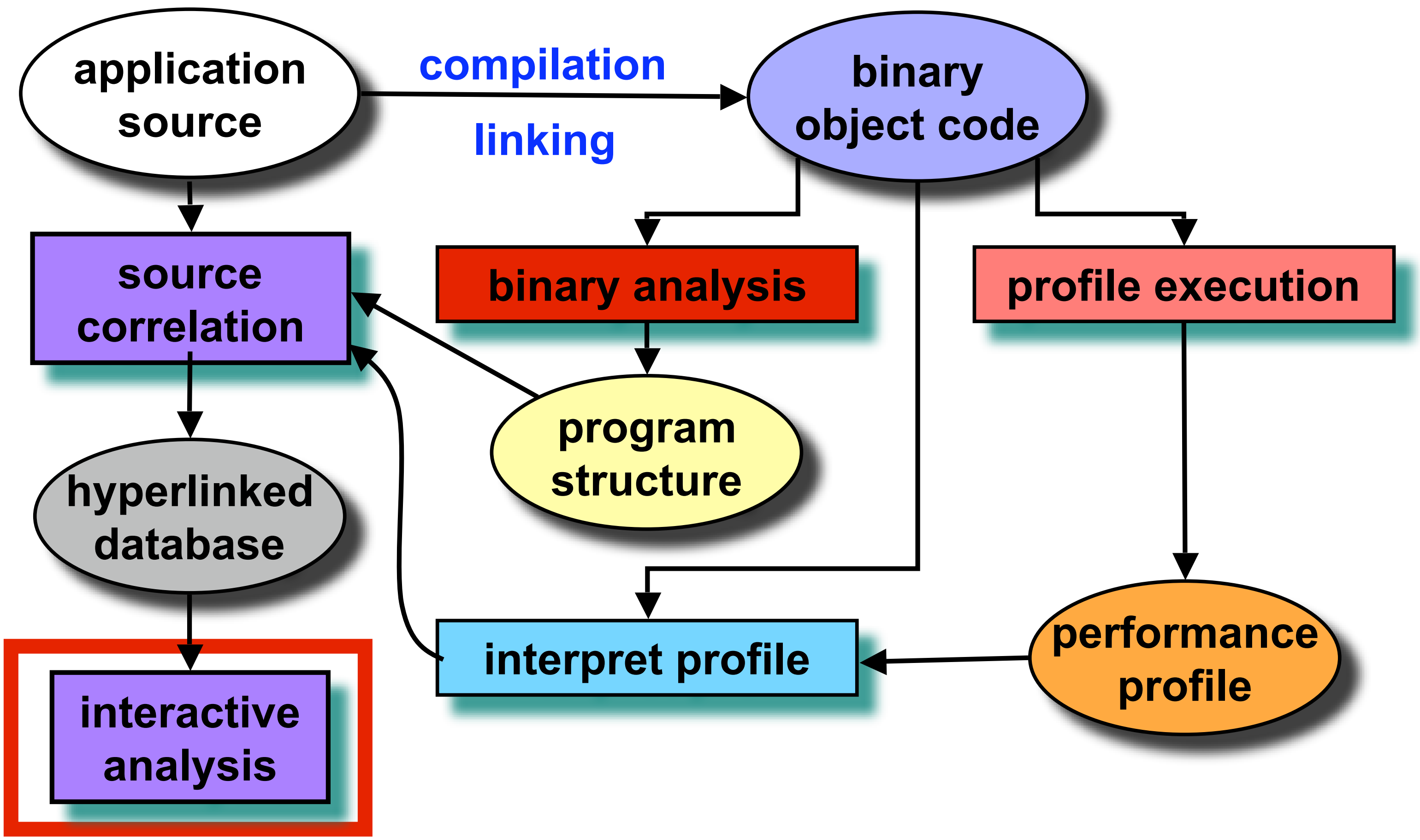
HPCToolkit System Workflow



— synthesize new metrics by combining metrics
— relate metrics, structure and program source

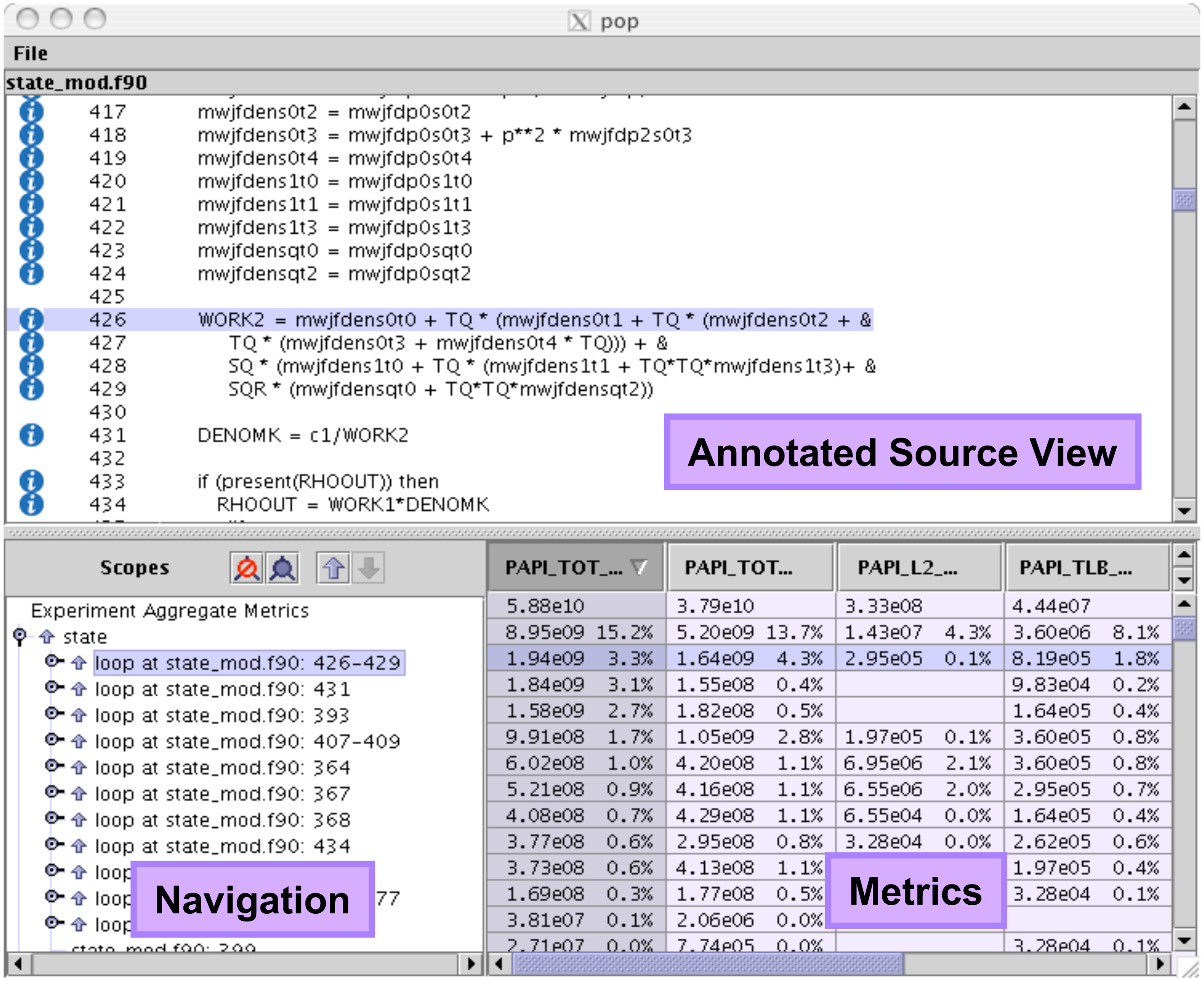
LACSI

HPCToolkit System Workflow



- support top-down analysis with interactive viewer
- analyze results anytime, anywhere

HPCToolkit on LANL's POP



LACSI

HPCToolkit ASC Impact

- HPCToolkit installed and used on ASC systems at LANL
  - past: Nirvana/Blue Mountain (MIPS+Irix)
  - today: Q (Alpha+Tru64)
  - emerging: Lightning (Opteron+Linux)
- Conducted performance tuning workshops at LANL
  - SAGE, FLAG, Truchas, MCNP, Blanca
- Used to pinpoint and tune ASC applications
  - assisted in analysis and tuning SAGE (factor of 2 improvement)
  - used to help gain additional factor of 2 on SAGE smvp performance
  - used to help gain a factor of 26 in Blanca AMR setup model code
- Used to assess FLAG for ASC burn code review in 2003

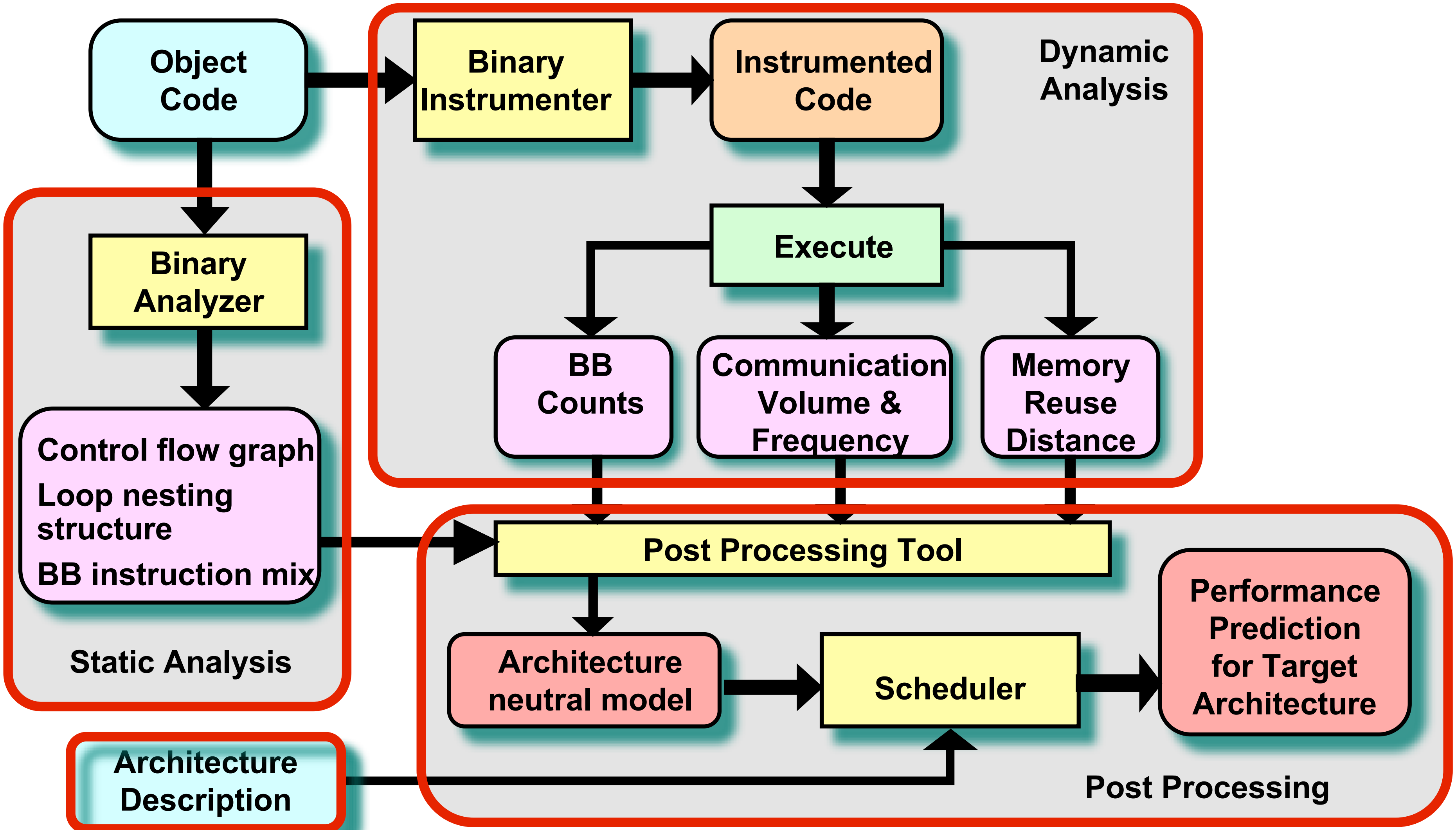
Nov 2004: HPC community tutorial at SC04

Performance Modeling

- LANL PAL modeling: scalable models of application performance
- The "missing link": detailed models of node performance
- Rice 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

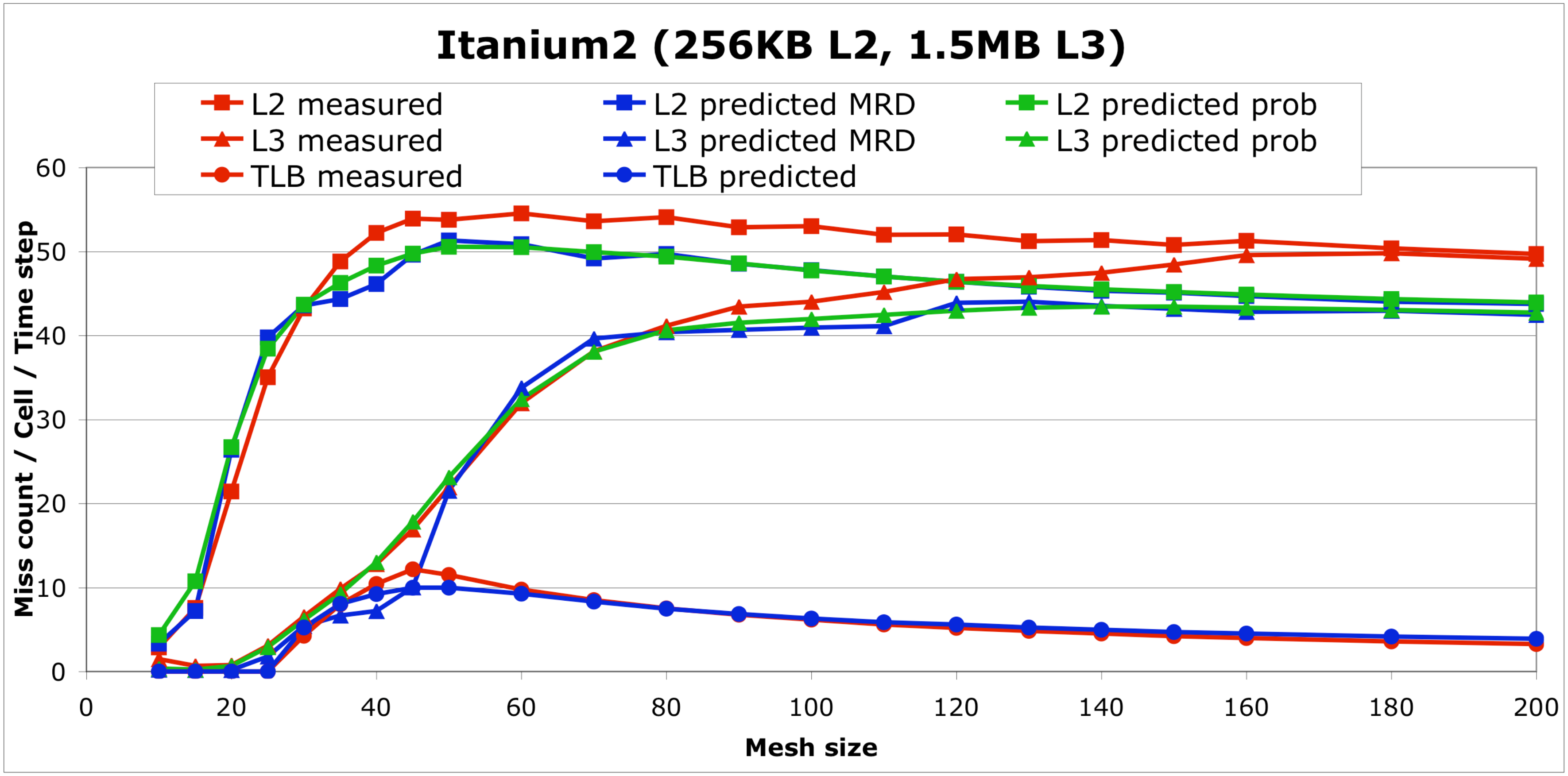
Marin & Mellor-Crummey: SIGMETRICS '04

Analysis and Modeling Toolkit



LACSI

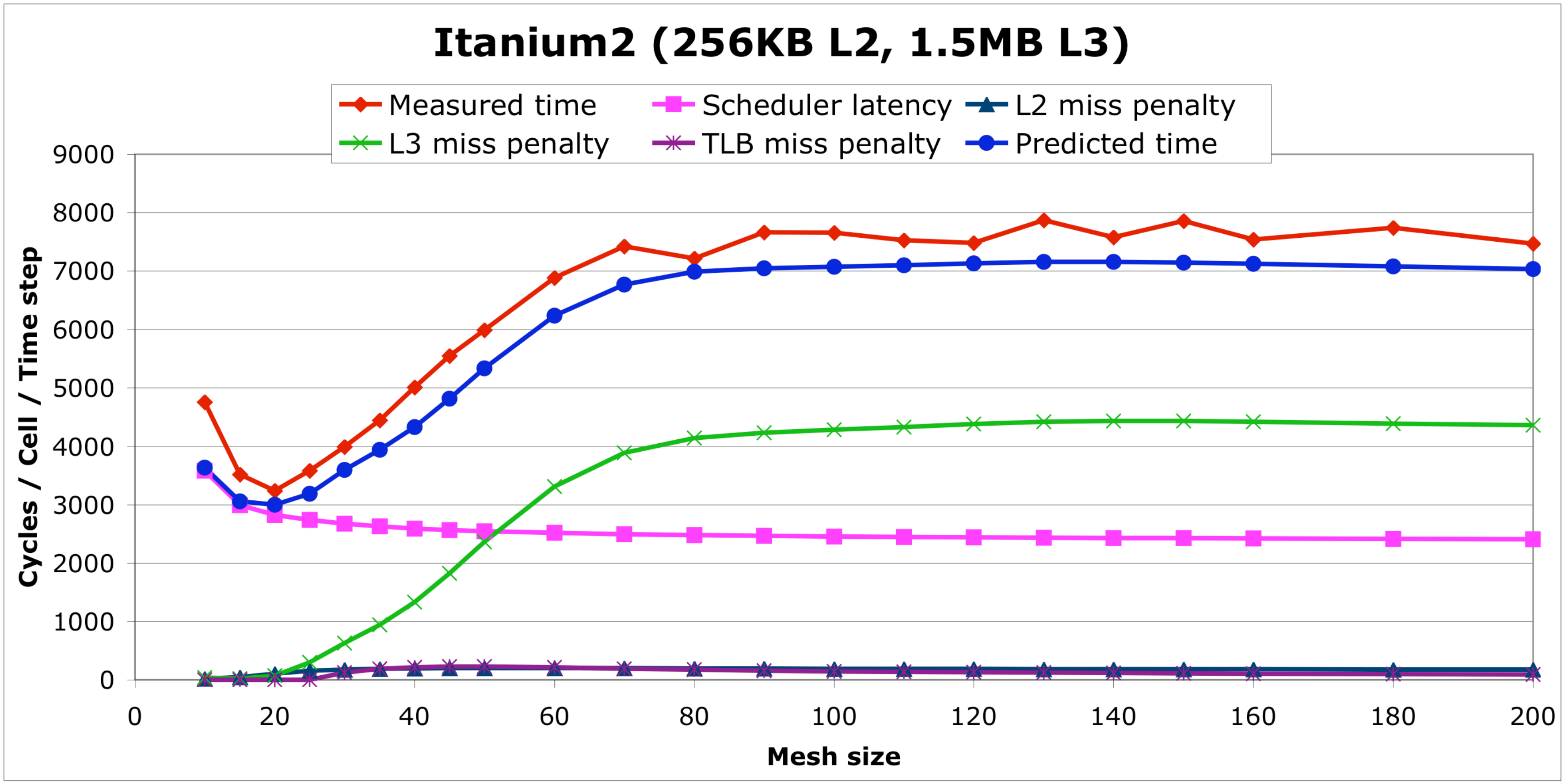
Memory Behavior: Sweep3D



Predicted from optimized SPARC binaries!

LACSI

Execution Behavior: Sweep3D



Predicted from optimized SPARC binaries!

LACSI

Compiler Technology for Parallel Languages

- Today: Computing on fragmented address spaces with MPI
- Enormous burden on application developer
  - Choose granularity of parallelism
  - Partition application data structures and computation
  - Add data movement and synchronization
  - Manage storage for non-local data
  - Developer responsible for all optimization of communication
    - latency tolerance: overlapping communication with computation
- Implications
  - Granularity choices are hard-coded into program
  - Hard to tailor for different architectures, e.g. vector vs. clusters

Higher-level Programming Models

Simplify programming of parallel systems

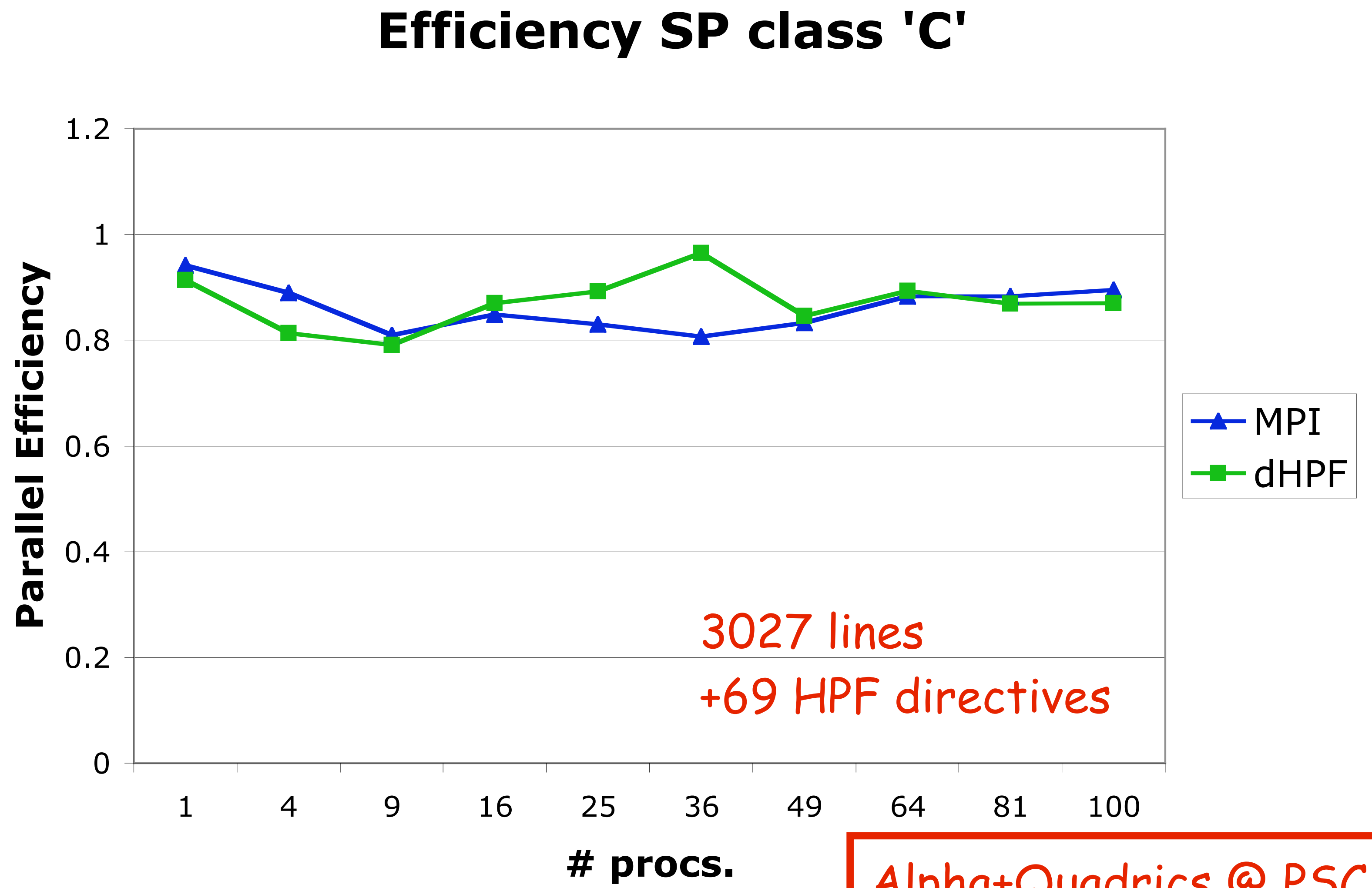
- Provide abstraction of global address space (GAS)
  - avoid tedious management of partitioned address spaces
- Separate concerns
  - algorithm specification vs. partitioning, mapping and synchronization
- Support flexible mappings of data and computation to “nodes”
  - high level management of locality
- Avoid target-dependent optimization by programmers
- Make applications more malleable
- Enhance performance portability

Compilers for High-Level Parallel Programming

- Near term
  - compiler technology for SPMD global address space languages results:
    - multiplatform Co-array Fortran compiler: (Alpha, Itanium, MIPS, Pentium) x (Quadrics, Myrinet, shared memory)
    - prototype compiler delivers performance comparable with MPI
  - Previous work on OpenMP compiler and tools
- Longer term
  - compiler technology for high-level data parallel languages, e.g. HPF
  - results: push the envelope for data parallel languages
    - new data and computation partitionings,
    - compiler optimizations for communication and computation
    - deliver performance competitive with hand-coded MPI

Two best paper awards in International Conferences

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



Alpha+Quadrics @ PSC

LACSI

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

1334 lines
+45 HPF directives

- Compiled with dHPF on PSC's Lemieux (Alpha+Quadrics)

# 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

LACSI

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

Led SC04 Workshop on Open Source Open64

- LLVM is another candidate
  - Newer compiler with fewer implemented optimizations
  - Good architecture; strong support for retargeting
  - Support for runtime reoptimization

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**
  - Scalarization
    - Asymptotically optimal scalarization of F90 array assignment
  - Blocking
    - Automating blocking of LU and QR factorization
  - Array contraction
- **Low Level**
  - Removing redundant memory operations
    - Discover and remove redundant pointer-based memory operations (up to 40% of all loads; 16% on average)
  - Object-to-object vectorizer for Pentium
    - Vectorize Pentium object code for SSE: 4x for some loops
  - Fast techniques for copy coalescing
    - New technique for modeling interferences
    - Speeds up copy coalescing and live-range identification

Knowledge Transfer

- Visitors at LANL
  - 2 LANL Rice summer interns
  - Extended visits (Fowler at LANL 6 weeks)
- Joint workshops with LANL
  - 2 performance tuning; 1 performance modeling
- Publications
  - topics: parallel compiler and runtime technology, improving memory hierarchy performance, algorithm implementation studies, performance analysis & tools, performance modeling, compiler algorithms
  - 3 best paper awards (IPDPS 01, IPDPS 02, PACT 02)
  - 14 journal articles
  - 52 conference and workshop papers
  - 3 technical reports
- 5 Rice PhD graduates (2 to national labs, 3 to academia)