Application Performance Profiling on the Cray XD1 using HPCToolkit

John Mellor-Crummey[1], Nathan Tallent[1]
Michael Fagan[1], and Jan E. Odegard[2]

[1]Computer Science
[2]Computer & Information Technology Institute
Rice University

CITI@20

Computer and Information Technology

To build a community of scholars that engages in collaborative research and education covering virtually every aspect of information technology and computing

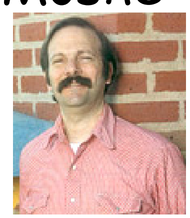
Directors:

Ken Kennedy (1986-1992)

Sidney Burrus (1992-1998)

Willy Zwaenepoel (1998-2001)

Moshe Vardi (2001-…)

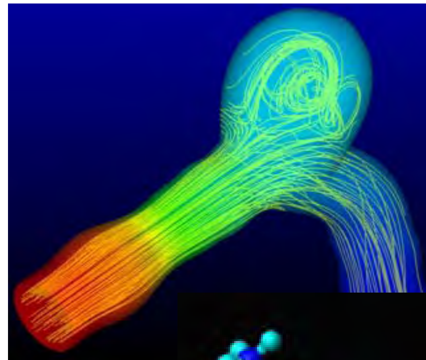
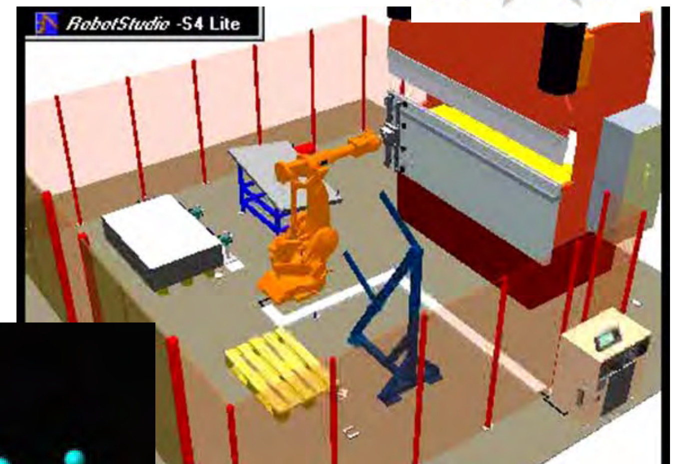


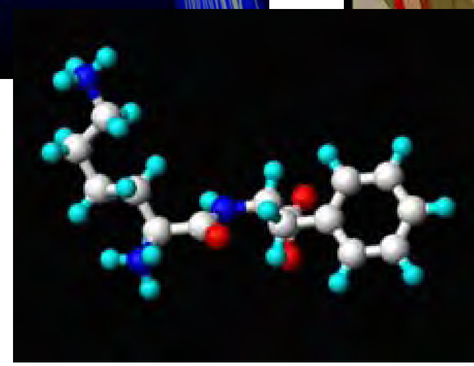
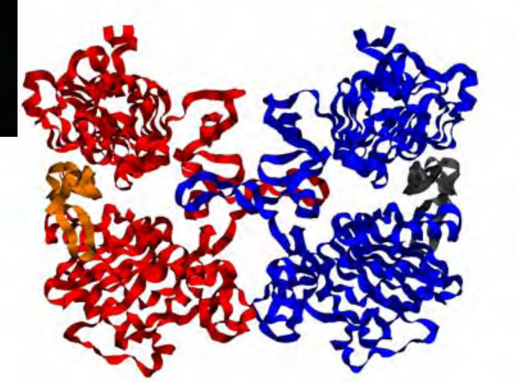
6 schools ←→ 20 departments ←→ 140 members
7 centers ←→ 12+ ad hoc research groups

Research Centers

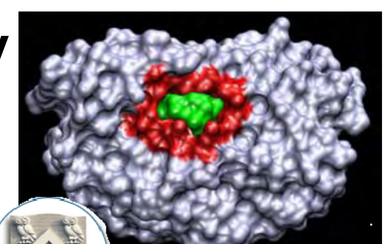
- **Center for High Performance Software (HiPerSoft)**
  - —**Director: TBN**
- **Center for Multimedia Communication (CMC)**
  - —**Director: Behnaam Aazhang, ECE**
- **Center for Computational Geophysics (CCG)**
  - —**Co-directors: Bill Symes, CAAM / Alan Levander, ES**
- **Center for Computational Finance & Economic Systems (CoFES)**
  - —**Director: Kathy Ensor, STAT**
- **LAboratory for NanoPhotonics (LANP)**
  - —**Director: Naomi Halas, ECE**
- **Center for Technology in Teaching and Learning (CTTL)**
  - —**Director: Tony Gorry, CS**
- **Center for Excellence and Equity in Education (CEEE)**
  - —**Director: Richard Tapia, CAAM**

Research Groups & Labs

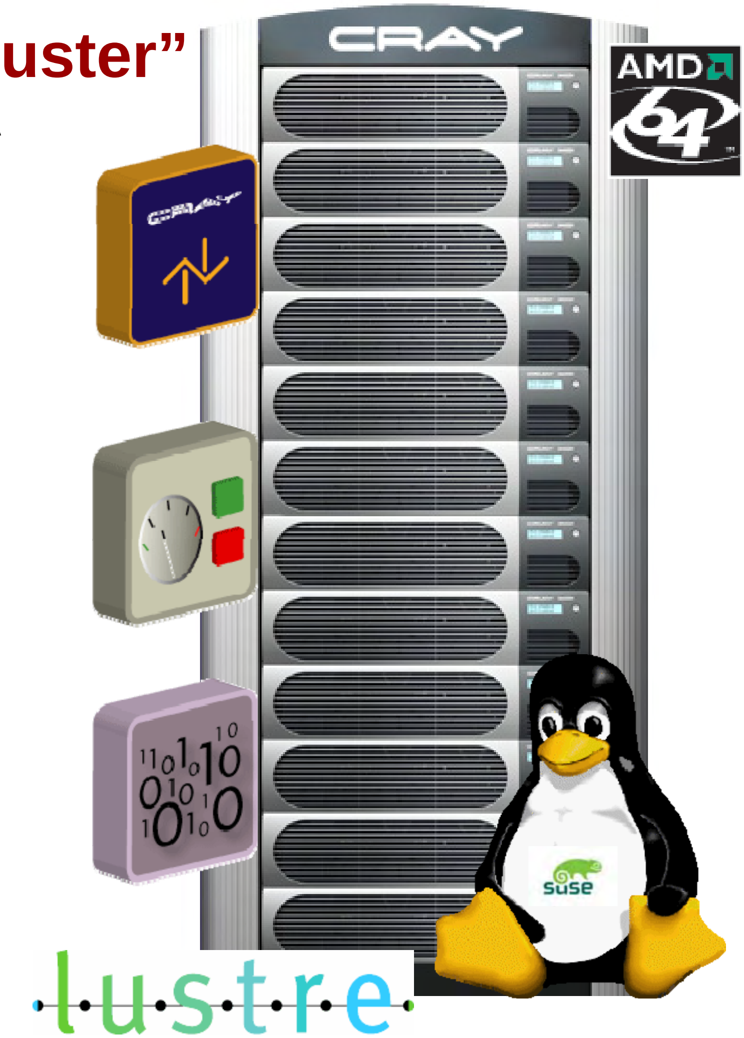
- Gaming Group
- Robotics Group
- Sensor Nets Group
- Bioinformatics Group
- Rice Networking Group
- Digital Signal Processing
- Dynamical Systems Group
- Statistical Consulting Lab
- Rice Computer Architecture Group
- Complex Flow of Complex Fluids Group
- Theoretical and Computational Neuroscience
- Connexions: Open content education repository
- Advanced Research Initiative on the Emerging Library
- …



Cray XD1 System
Dual-Core AMD Opteron™

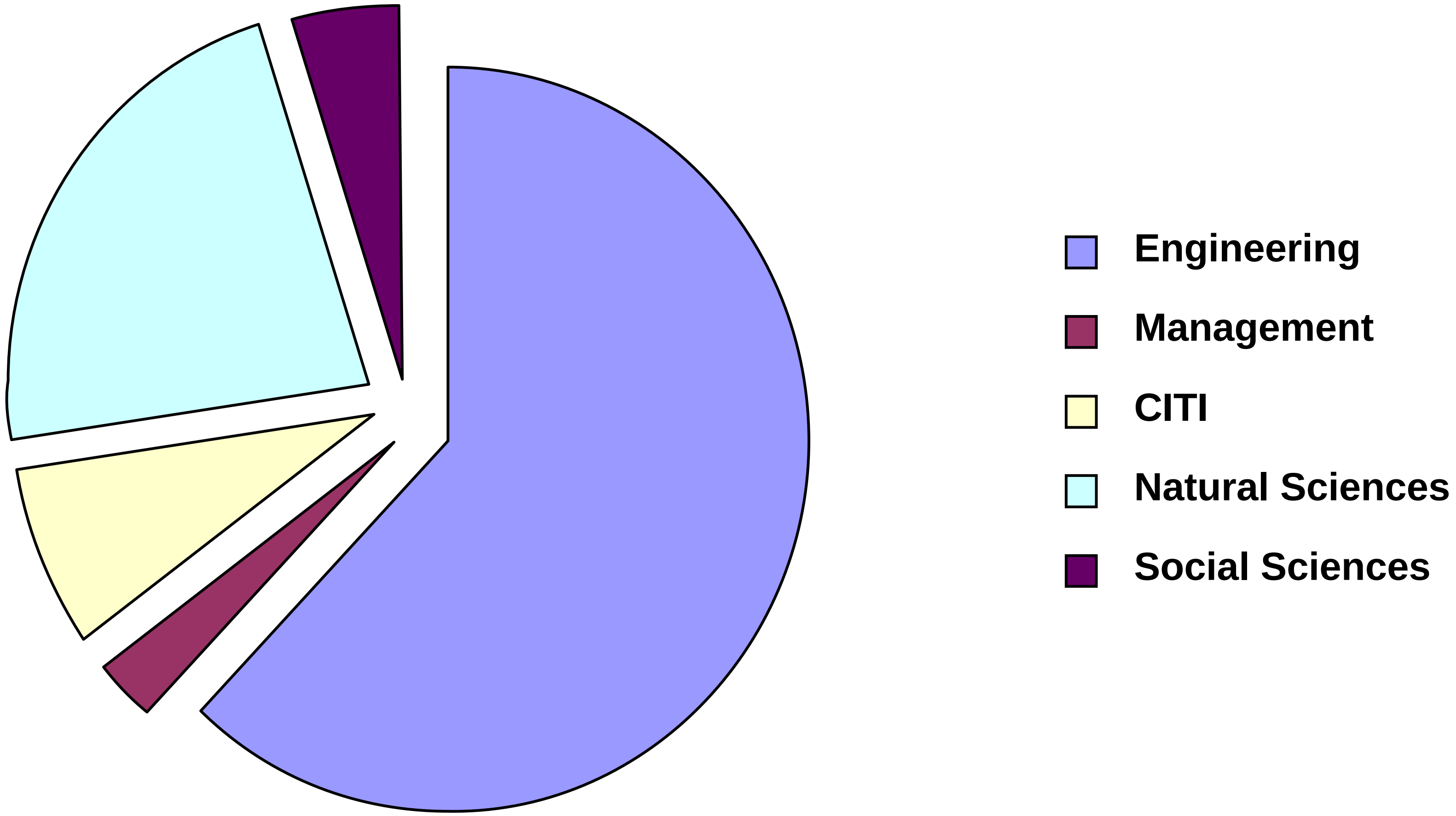
"Rice Computational Research Cluster"

- **~3 TeraFLOP Cray XD1 Linux cluster***
  - **336 Dual-Core AMD Opteron™ 275**
    - **2.2GHz, 1MB / Core**
    - **168 dual socket nodes (4 way SMP)**
    - **8 GB DDR 400 / compute SMP**
    - **16 GB DDR 333 / system SMP**
  - **Cray RapidArray (4x Infiniband)**
  - **1.4 TB DDR2 400**
  - **12 TB Local Disk**
  - **6 TB Lustre parallel file system**
  - **10 TB NFS file system**
  - **One XD1 Chassis with FPGA**
    - **6 Xilinx Virtex-4/LX160**



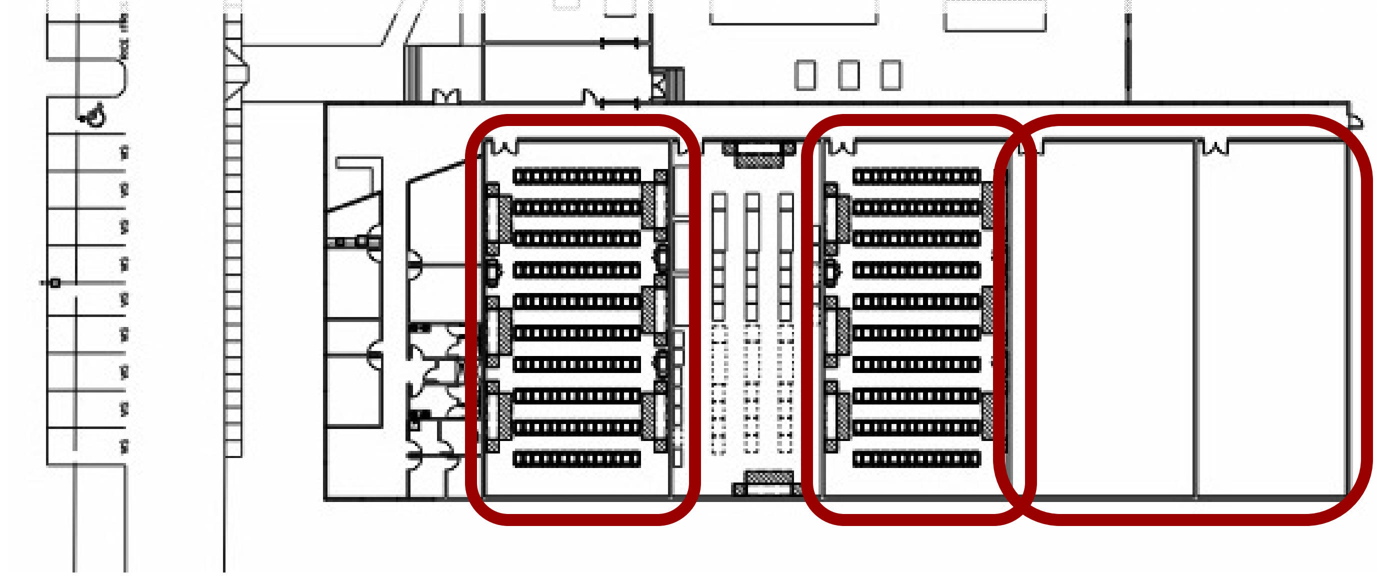
NSF MRI, Rice, AMD and Cray

250+ Active Users



New Datacenter (July 2007)

- 20,000 sq.ft. (48” raised floor)
- 6 MWatt inbound power
- Three pod configuration (3-15+ KW/rack)
- Three separate electrical systems (A, B & C)
- Mechanical: N+1 or better



Datacenter (~12-06)

Datacenter (~12-06)

The Challenge

Getting Science Done

To achieve acceptable (top) application performance scientists and engineers are required to tailor applications to effectively exploit the capabilities of a “bewildering” array of features offered by current and future architectures

Performance Analysis and Tuning

- **Increasingly necessary**
  - —gap between typical and peak performance is growing
- **Increasingly hard**
  - —complex architectures are harder to program effectively
    - – complex processors
      - deeply pipelined, out of order, superscalar
    - – complex memory hierarchy
      - non-blocking, multi-level caches, TLB, hw prefetching
  - —modern scientific applications pose challenges for tools
    - – multi-lingual programs
    - – many source files
    - – complex build process
    - – external libraries in binary-only form

HPCToolkit Goals

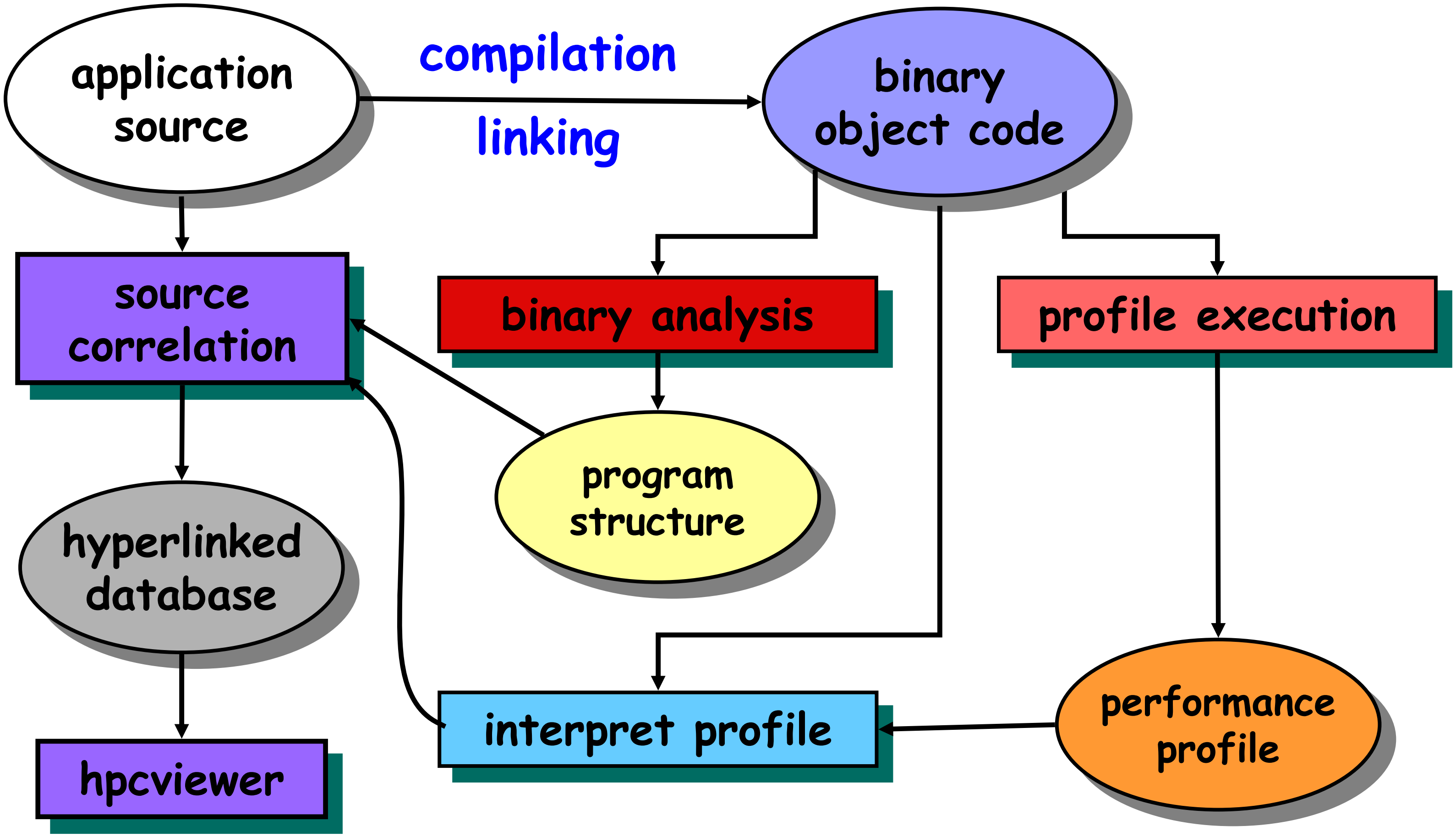
- Support large, multi-lingual applications
  - —a mix of of Fortran, C, C++
  - —multiple programming models (MPI, OpenMP, multi-threading)
  - —external libraries
  - —hundreds of procedures
  - —for ease of use, avoid
    - – manual instrumentation
    - – significantly altering the build process
    - – frequent recompilation
- Analysis of both serial and parallel codes
- Scalable data collection for parallel executions
- Effective presentation of analysis results
  - —intuitive enough for scientists and engineers to use
  - —detailed enough to meet the needs of compiler writers

 http://hipersoft.cs.rice.edu/hpctoolkit/

HPCToolkit Design Principles

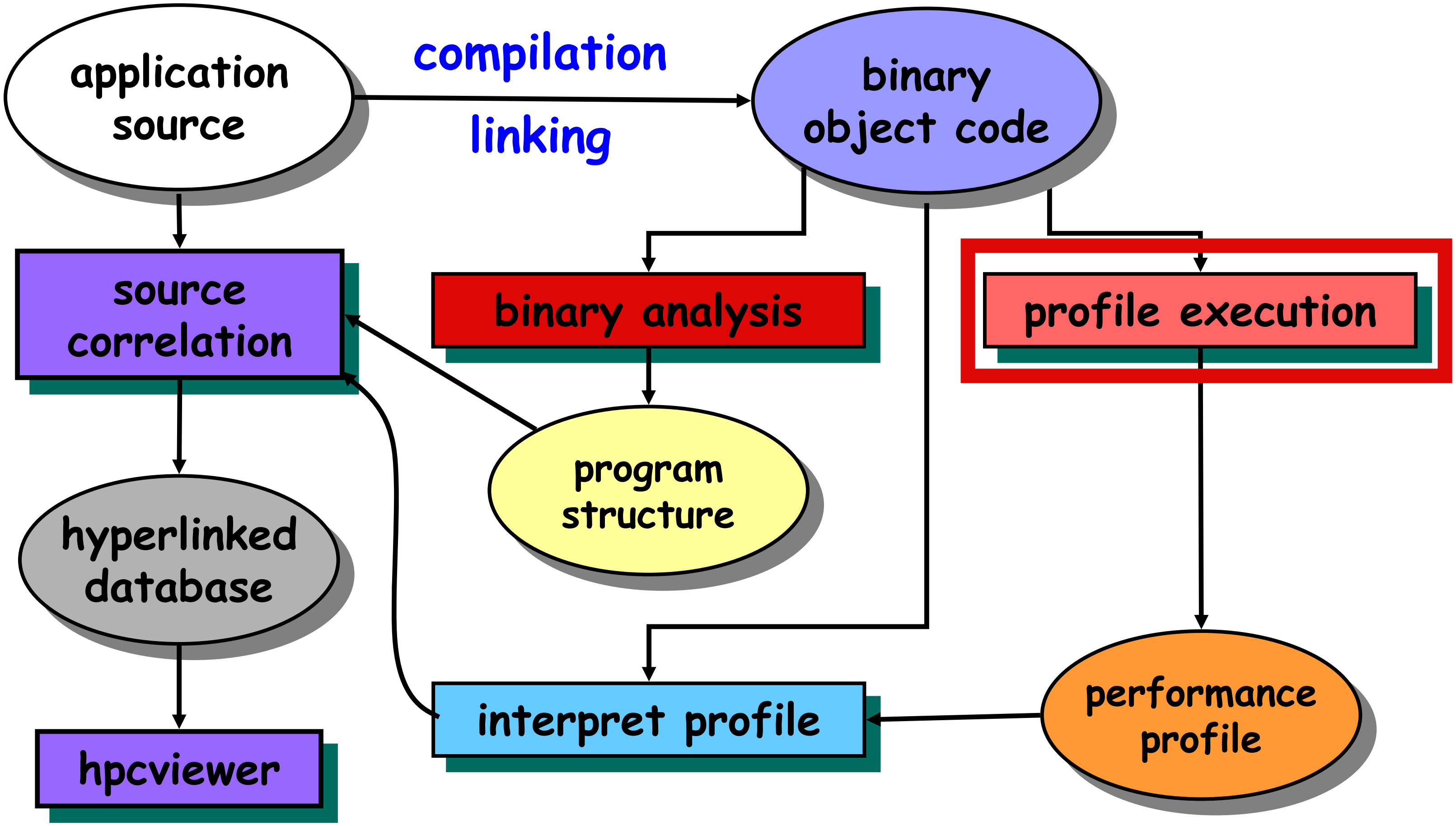
- Language independence: work at the binary level
  —supports multi-lingual codes with external binary-only libraries
- Avoid code instrumentation in each procedure
  —instrumentation adds overhead and distorts measurements
- Context is essential for understanding modern software
  —modular software often depends on layered libraries (e.g. MPI)
- Any one performance measure produces a myopic view
  — hard to diagnose a problem with only one species of event
- Derived metrics are essential for effective analysis
- Performance analysis should be top down
- Event aggregation for loops and procedures is important
  —accurate despite approximate event attribution from counters
  —loop-level info is more important than line-level info

HPCToolkit Workflow



 http://hipersoft.cs.rice.edu/hpctoolkit/

HPCToolkit Workflow

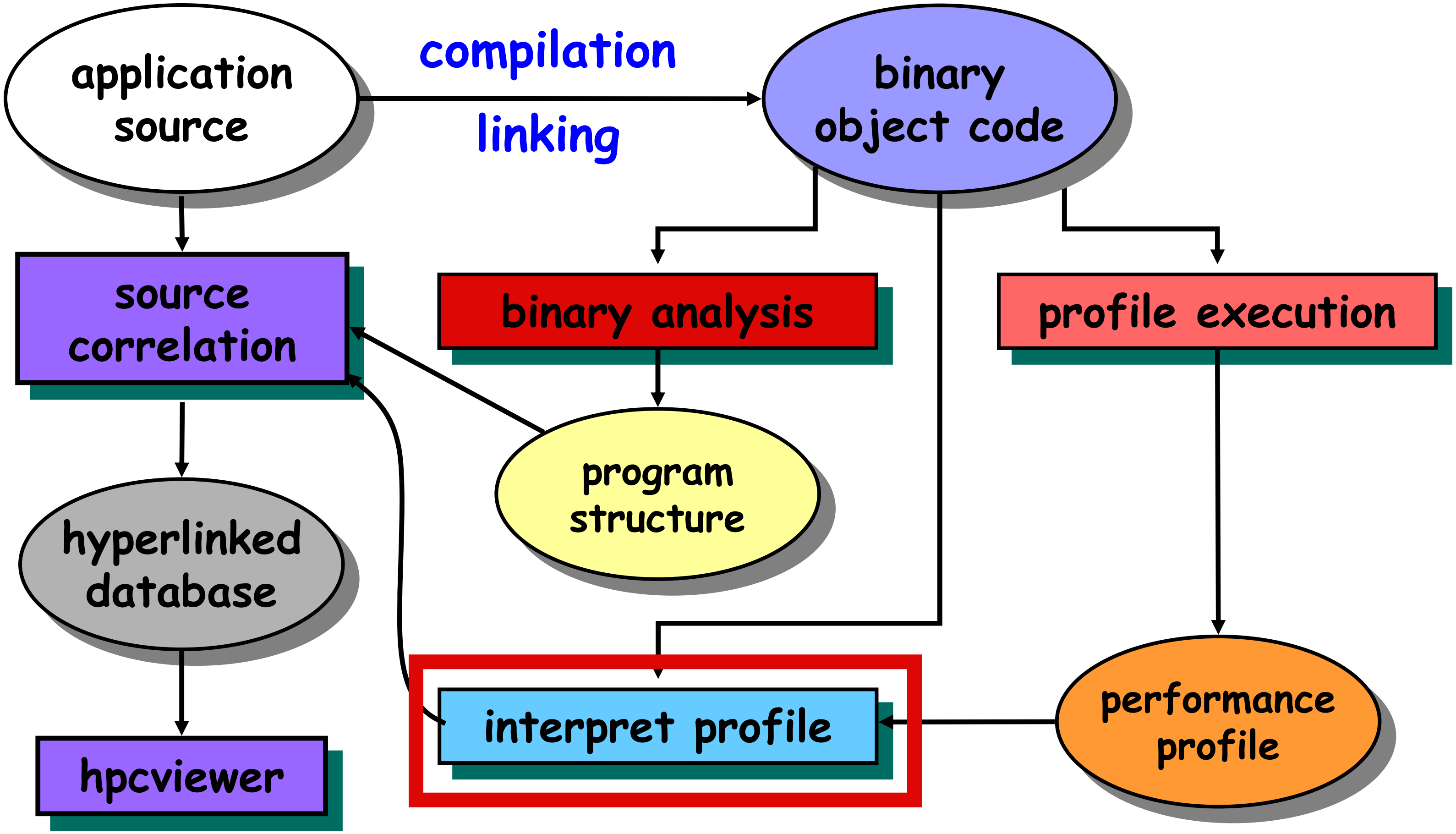


—launch unmodified, optimized application binaries

—collect statistical profiles of events of interest

http://hipersoft.cs.rice.edu/hpctoolkit/

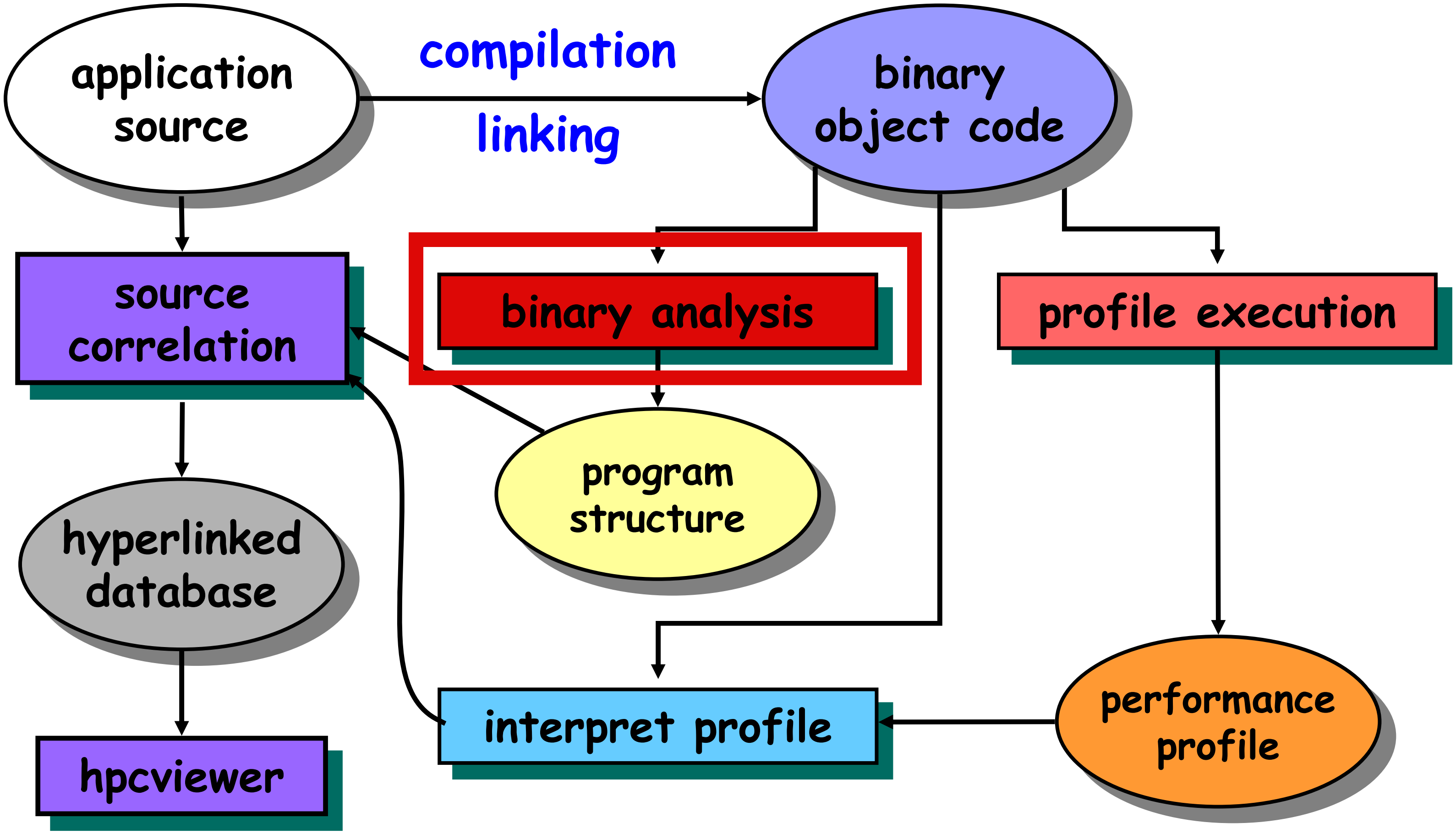
HPCToolkit Workflow



—decode instructions and combine with profile data

 http://hipersoft.cs.rice.edu/hpctoolkit/

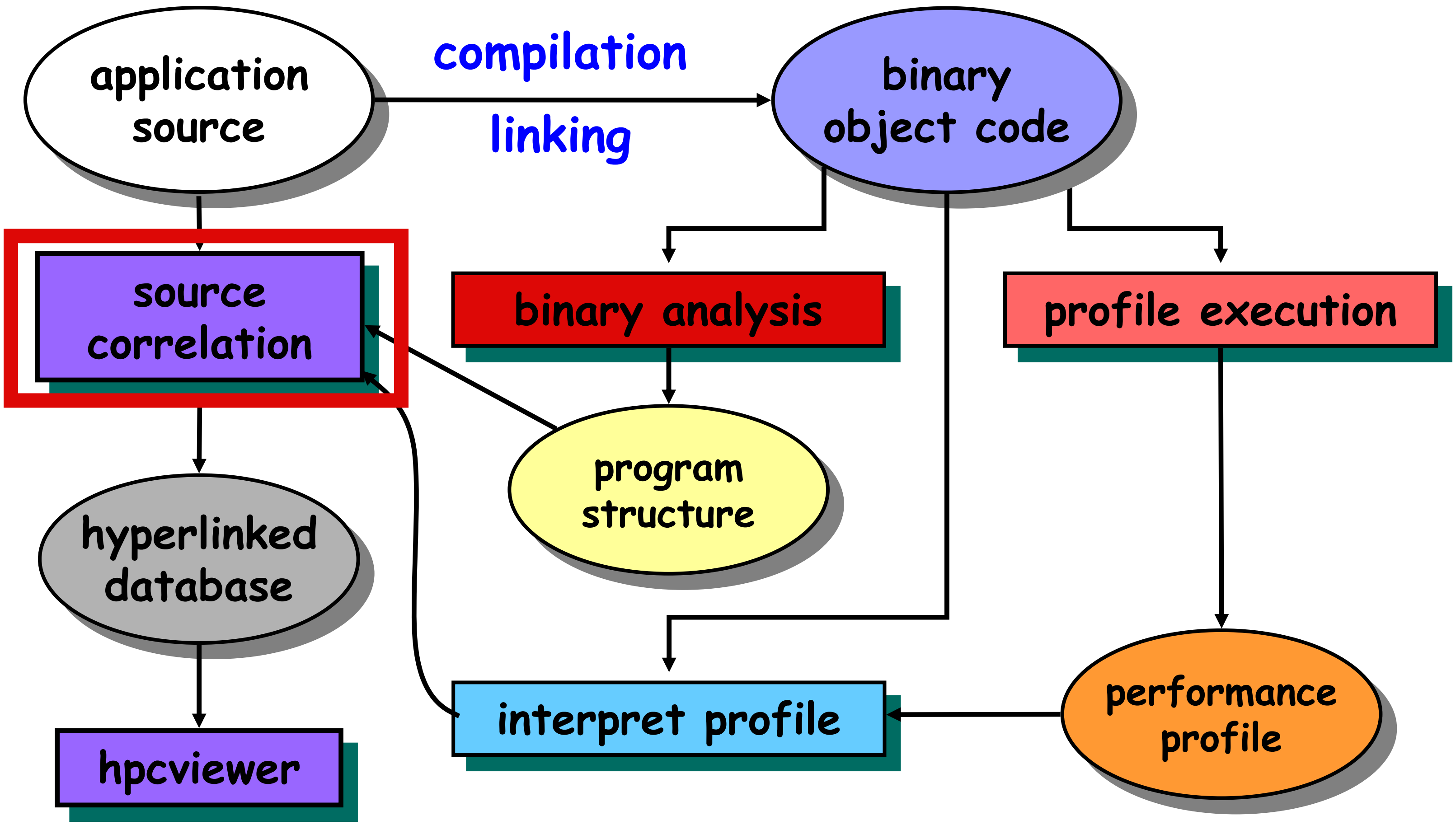
HPCToolkit Workflow



—extract loop nesting & inlining from executables

 http://hipersoft.cs.rice.edu/hpctoolkit/

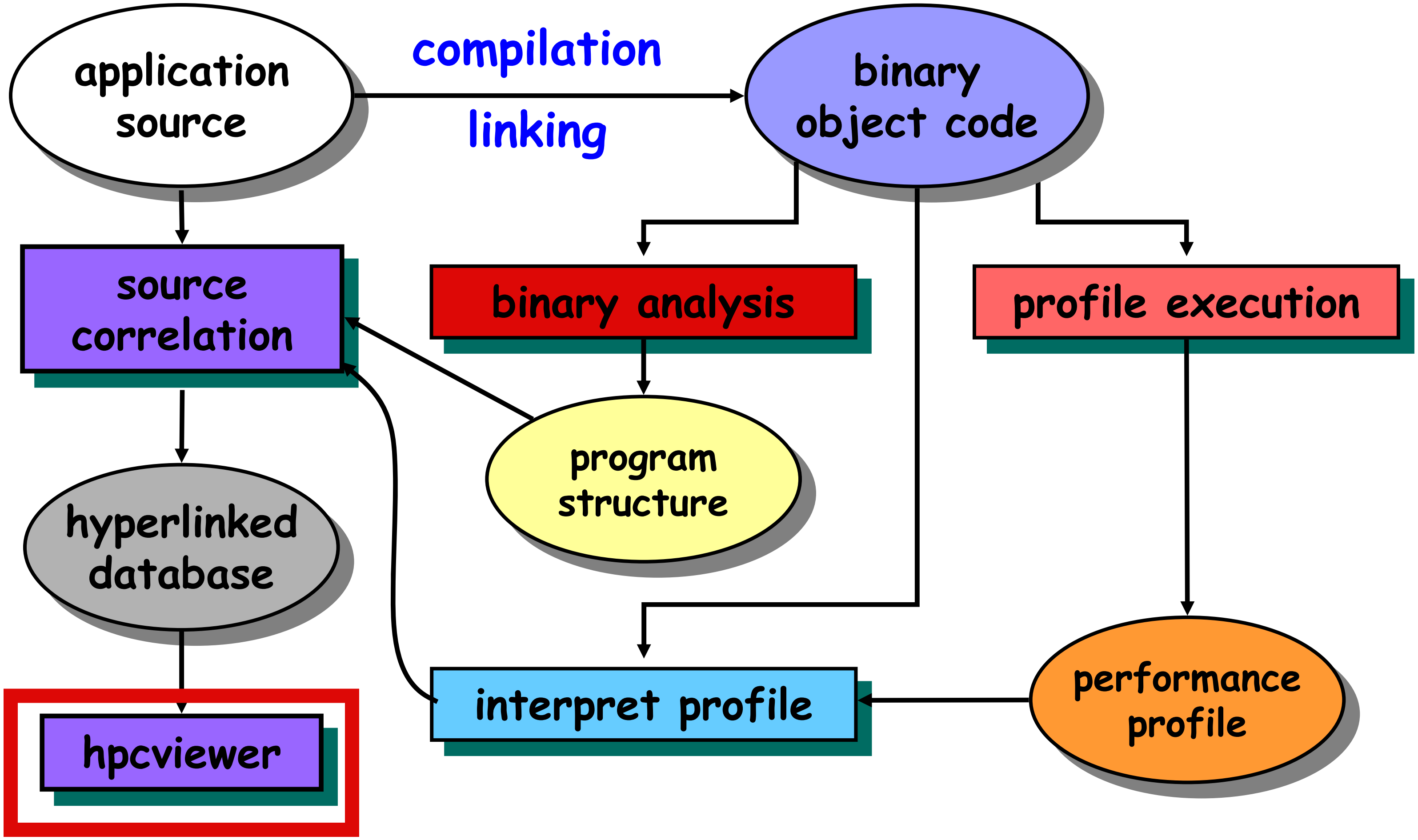
HPCToolkit Workflow



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

http://hipersoft.cs.rice.edu/hpctoolkit/

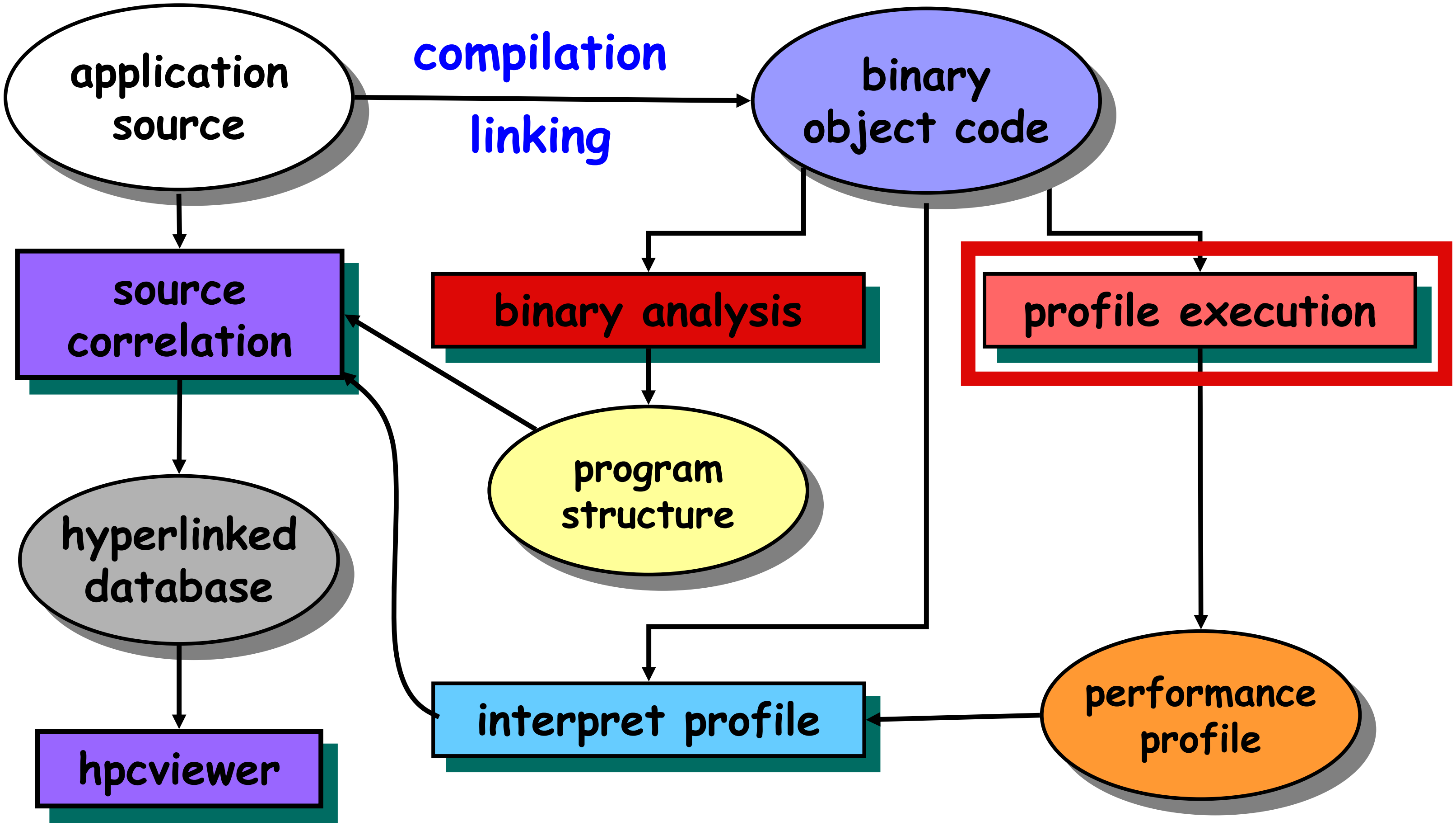
HPCToolkit Workflow



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

http://hipersoft.cs.rice.edu/hpctoolkit/

HPCToolkit System Overview



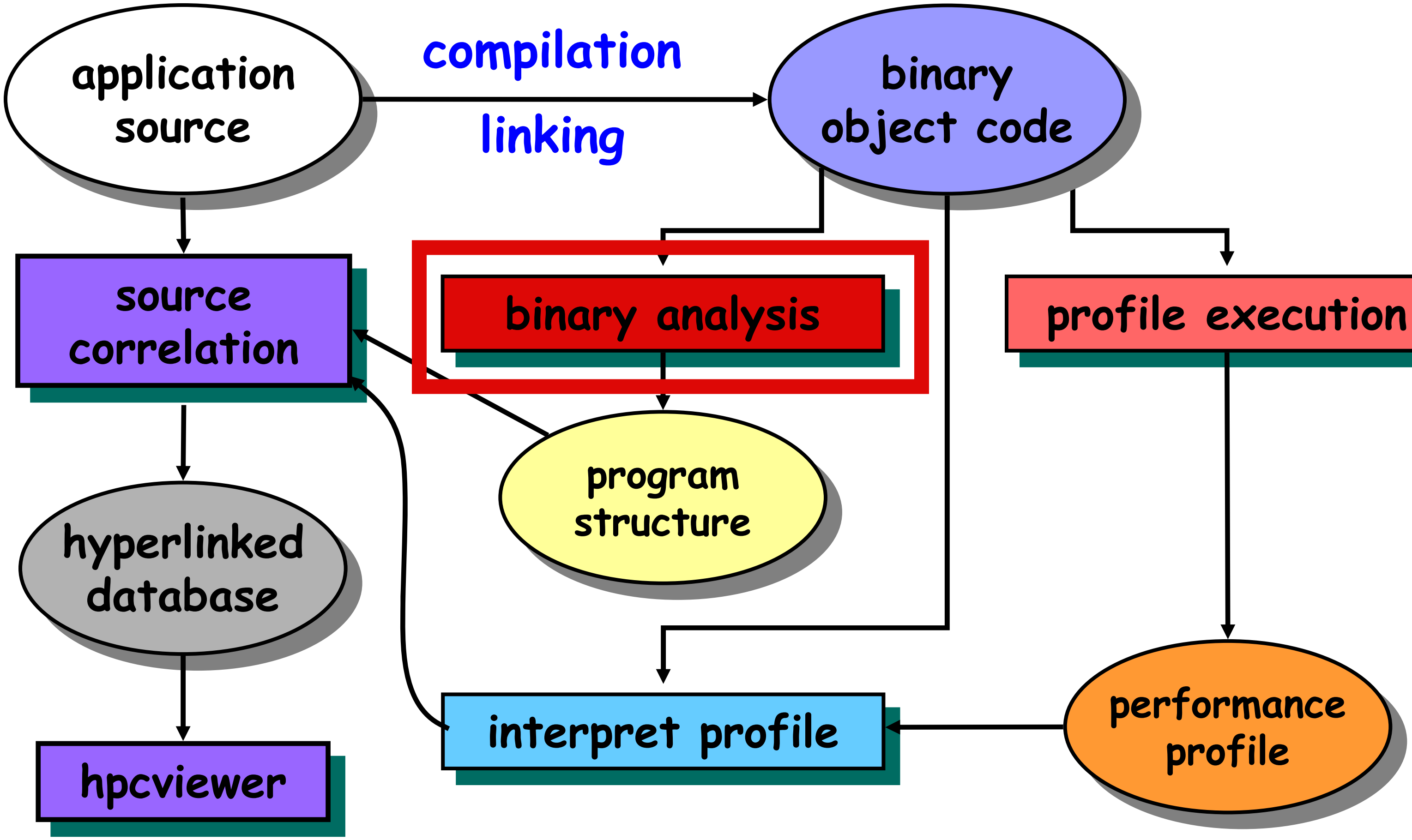
 http://hipersoft.cs.rice.edu/hpctoolkit/

Data Collection

Support analysis of unmodified, optimized binaries

- **Use statistical sampling to profile events**
  - **hardware performance counter overflows**
  - **interval timer events**
- **Tools**
  - **`hpcrun`: flat sampling yields PC histograms**
  - **`csprof`: attributes samples to calling context**

HPCToolkit System Overview

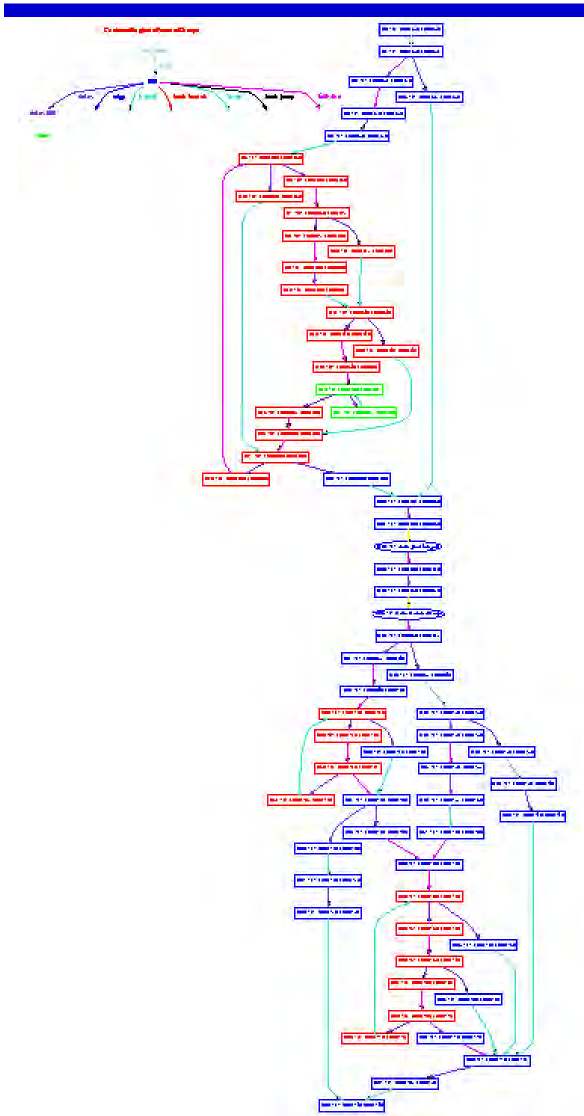


Program Structure Recovery with `bloop`

Analyze an application binary

- Construct control flow graph from branches
- Identify natural loop nests using interval analysis
- Map instructions to source lines, procedures
  —leverage line map + DWARF debugging information
- Discover inlined code
- Normalize output to recover source-level view

Sample Flowgraph from an Executable



Loop nesting structure

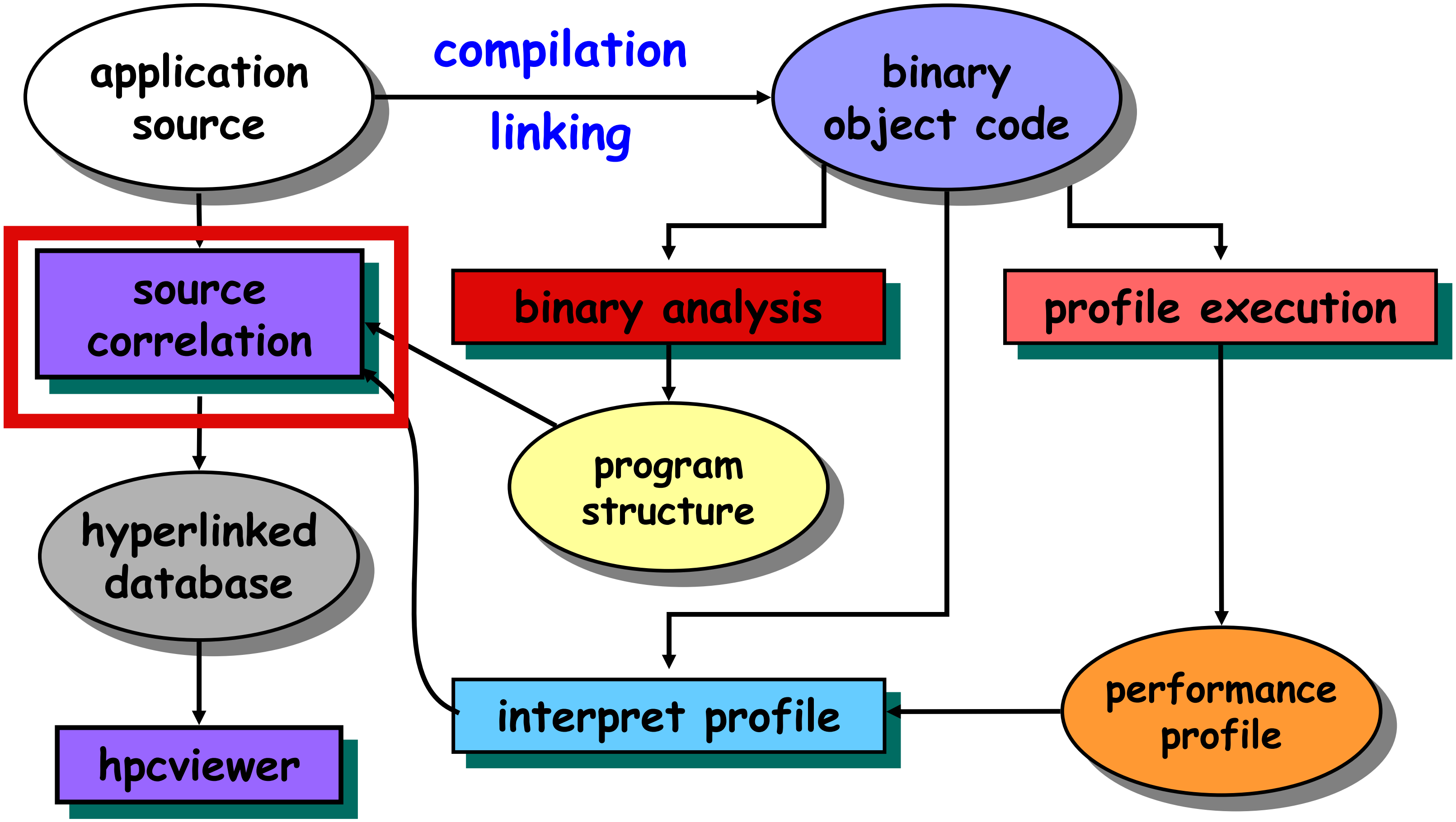
- —blue: outermost level
- —red: loop level 1
- —green loop level 2

Observation

optimization complicates program structure!

http://hipersoft.cs.rice.edu/hpctoolkit/

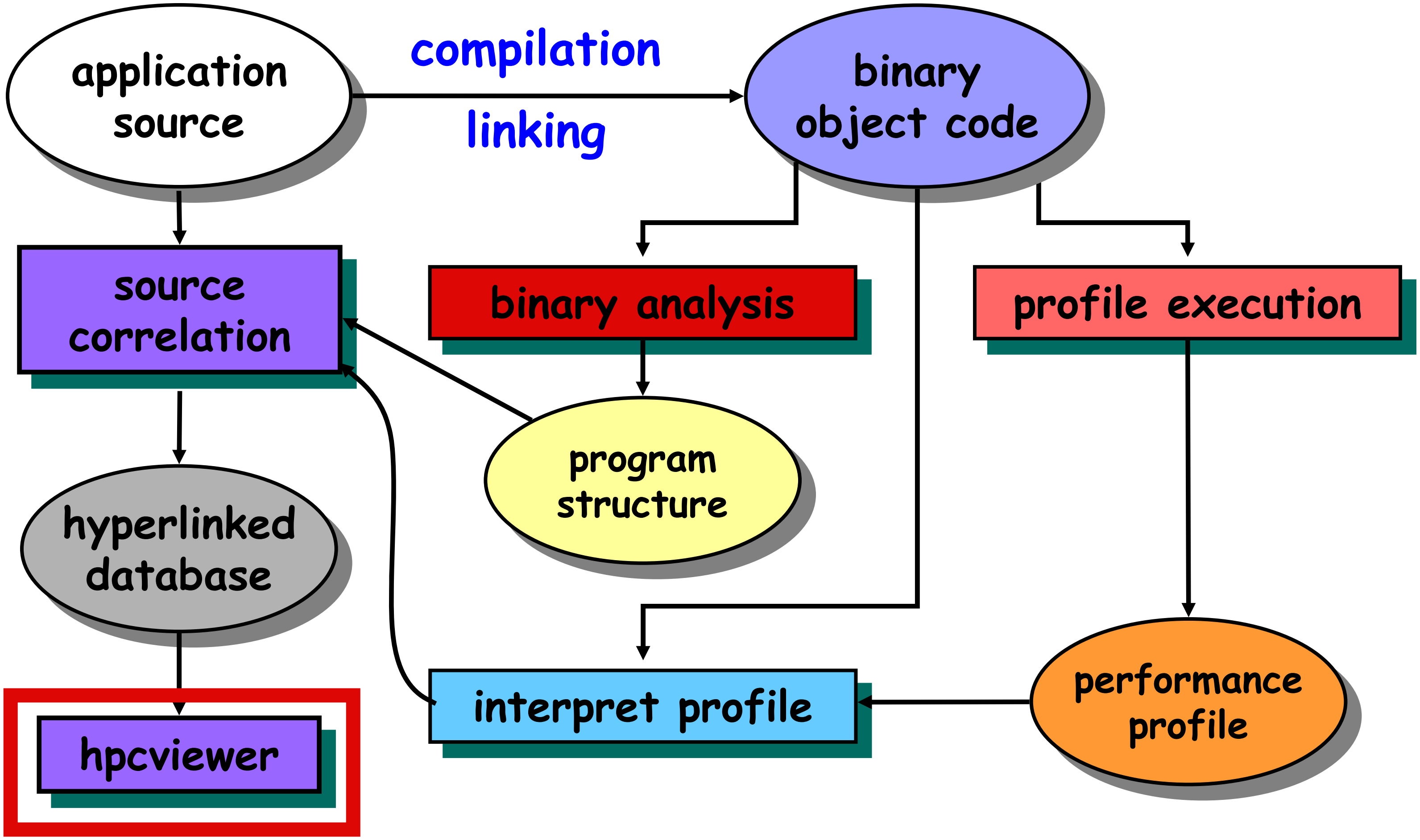
HPCToolkit System Overview



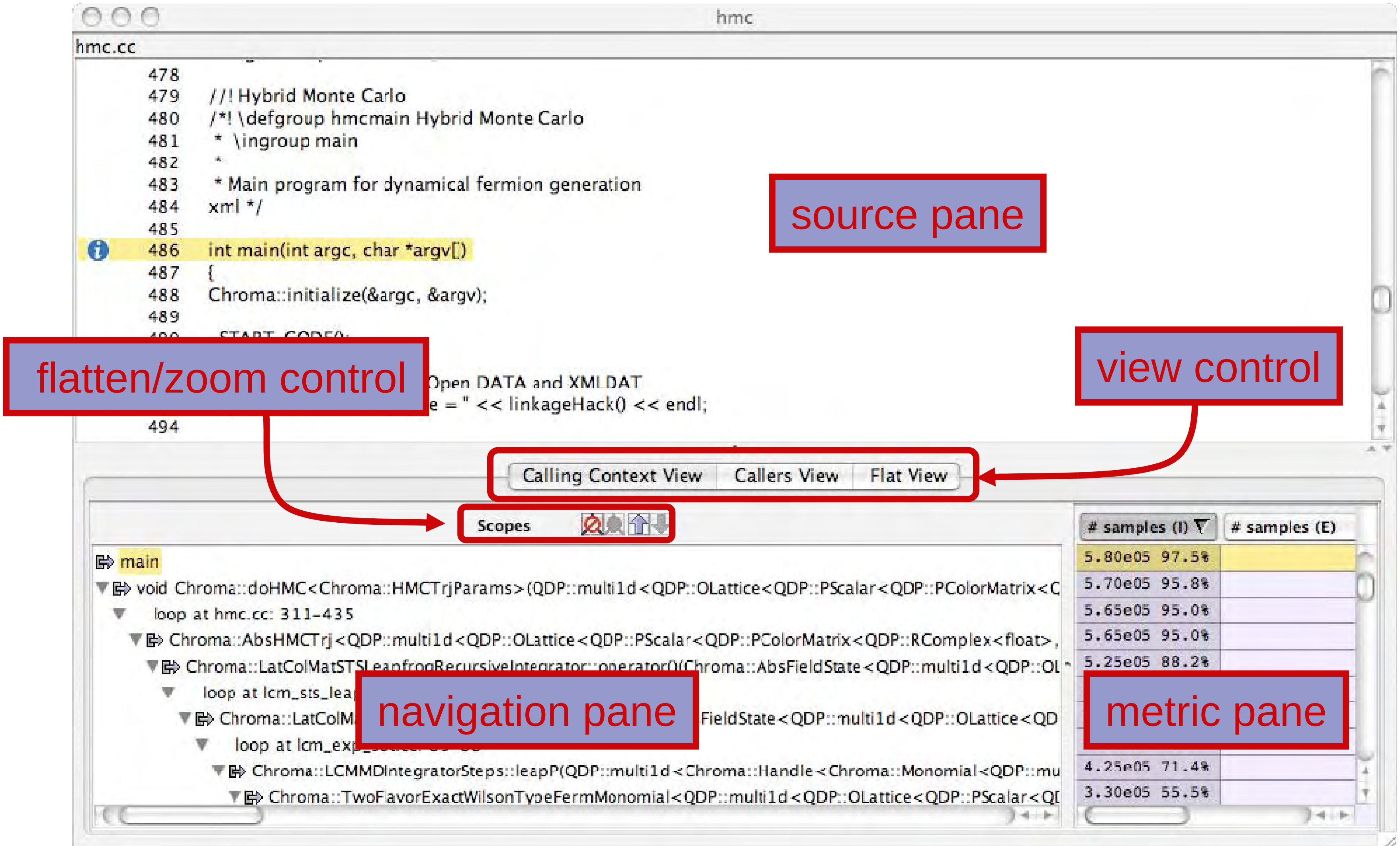
Data Correlation

- **Problem**
  - —**any one performance measure provides a myopic view**
    - – **some measure potential *causes* (e.g. cache misses)**
    - – **some measure *effects* (e.g. cycles)**
    - – **cache misses not always a problem**
  - —**event counter attribution is often inaccurate**
- **Approaches**
  - —**multiple metrics for each program line**
  - —**computed metrics (e.g. waste = peak FLOPs - actual FLOPS)**
    - – **eliminates mental arithmetic**
    - – **serves as a key for sorting**
  - —**hierarchical structure**
    - – **errors with line level attribution still yield good loop-level information**

HPCToolkit System Overview



hpcviewer User Interface



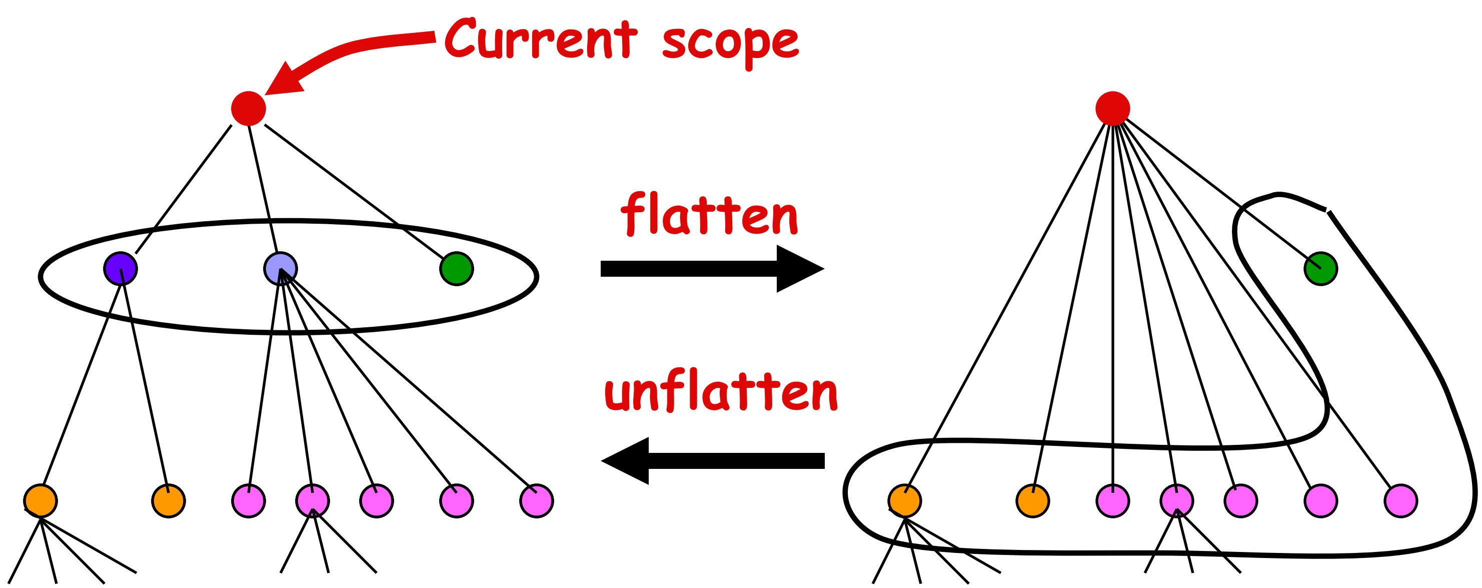
http://hipersoft.cs.rice.edu/hpctoolkit/

Principal `hpcviewer` Views

- **Calling context tree view**
  - —***top-down*** **view shows dynamic** ***calling contexts*** **in which costs were incurred**
- **Caller's view**
  - —***bottom-up*** **view apportions costs incurred in a routine to the routine's dynamic calling contexts**
- **Flat view**
  - —**aggregates all costs incurred by a routine** ***in any context*** **and shows the details of where they were incurred within the routine**

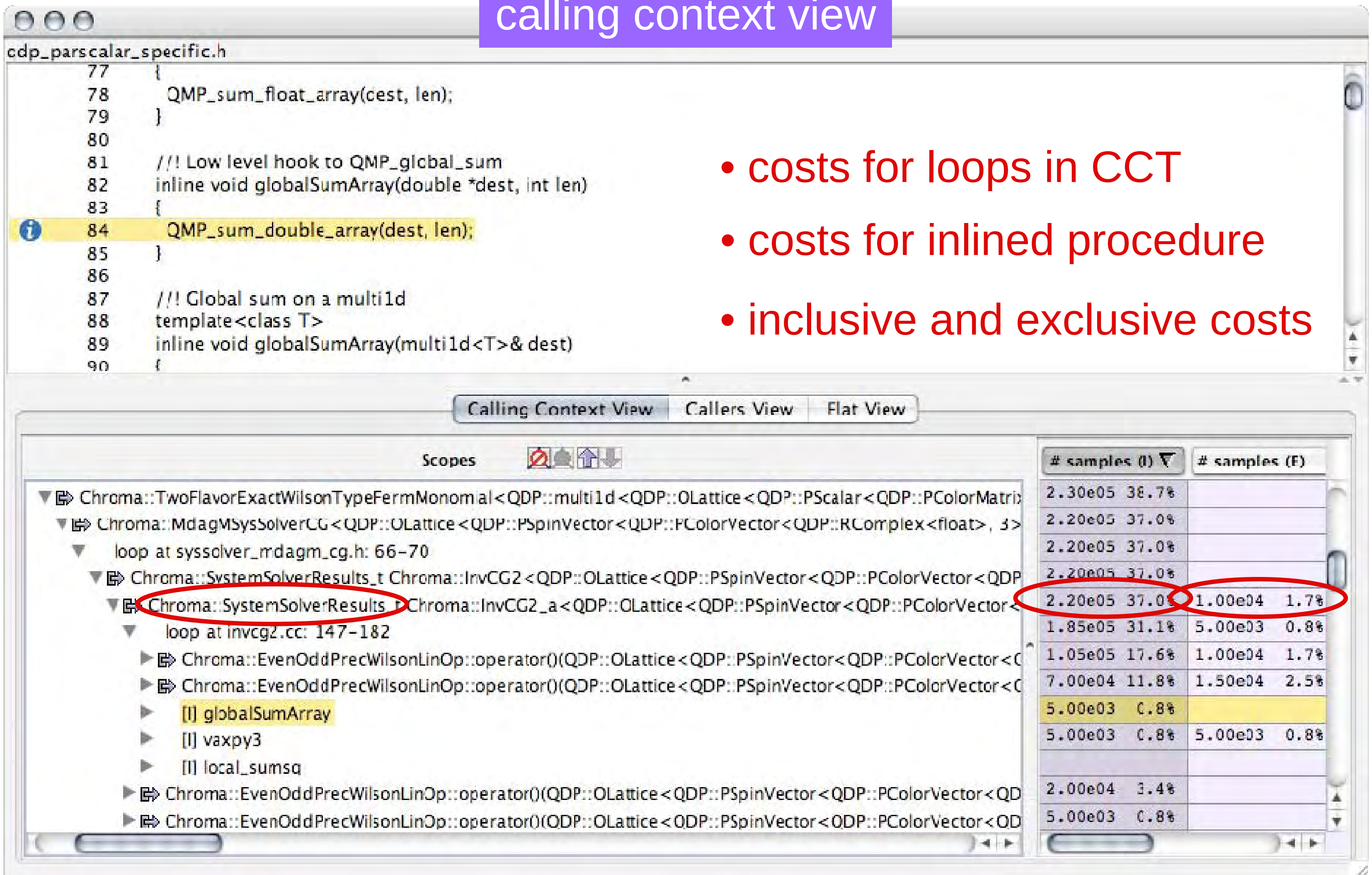
Flattening Static Hierarchies

- **Problem**
  - —hierarchical view of a program is too rigid
  - —sometimes want to compare children of different parents
    - – e.g. compare all loops, regardless of the routine they are inside
- **Solution**
  - —flattening elides a scope and shows its children instead



Chroma Lattice QCD Library

calling context view

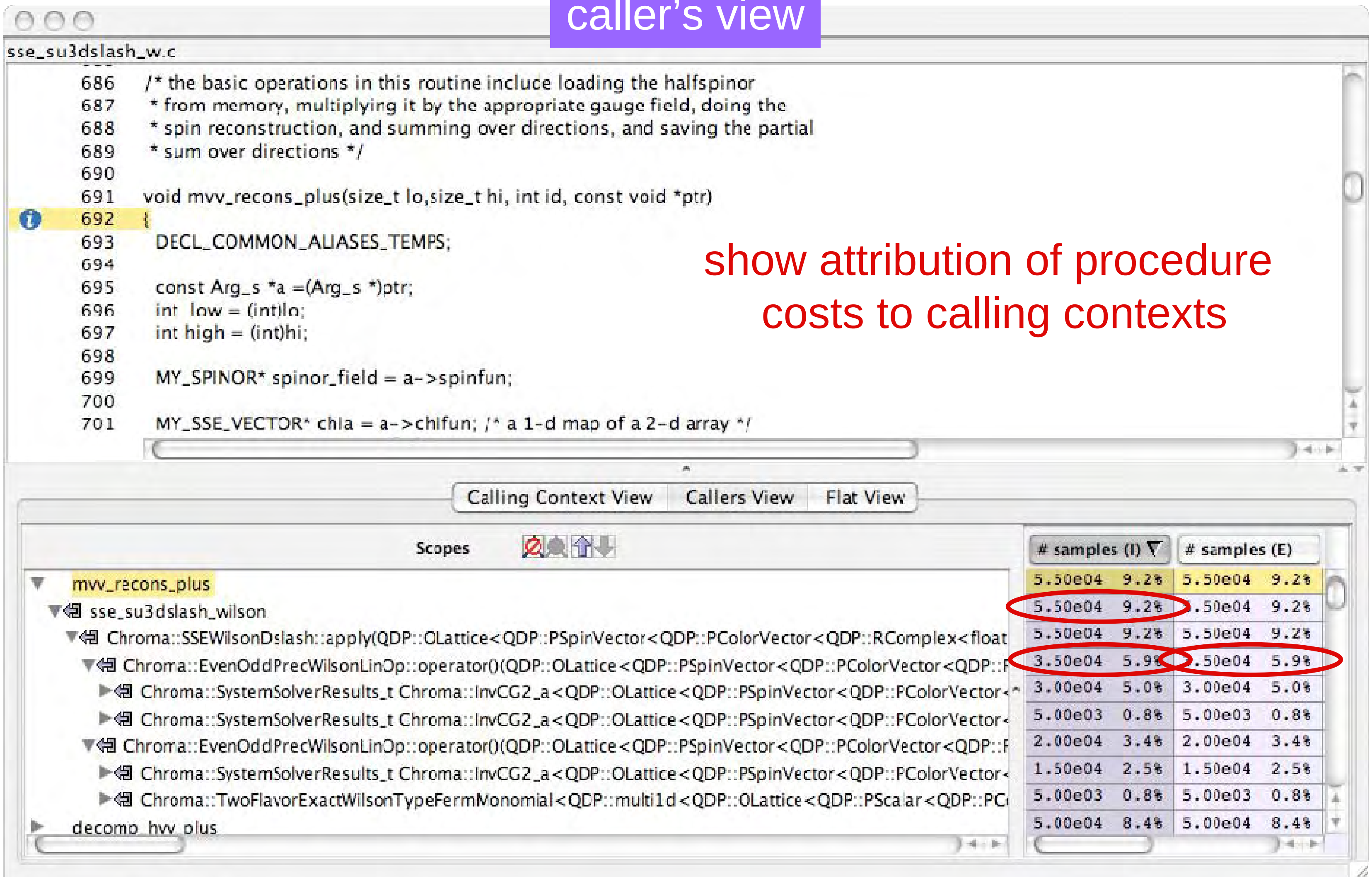


- costs for loops in CCT
- costs for inlined procedure
- inclusive and exclusive costs

http://hipersoft.cs.rice.edu/hpctoolkit/

Chroma Lattice QCD Library

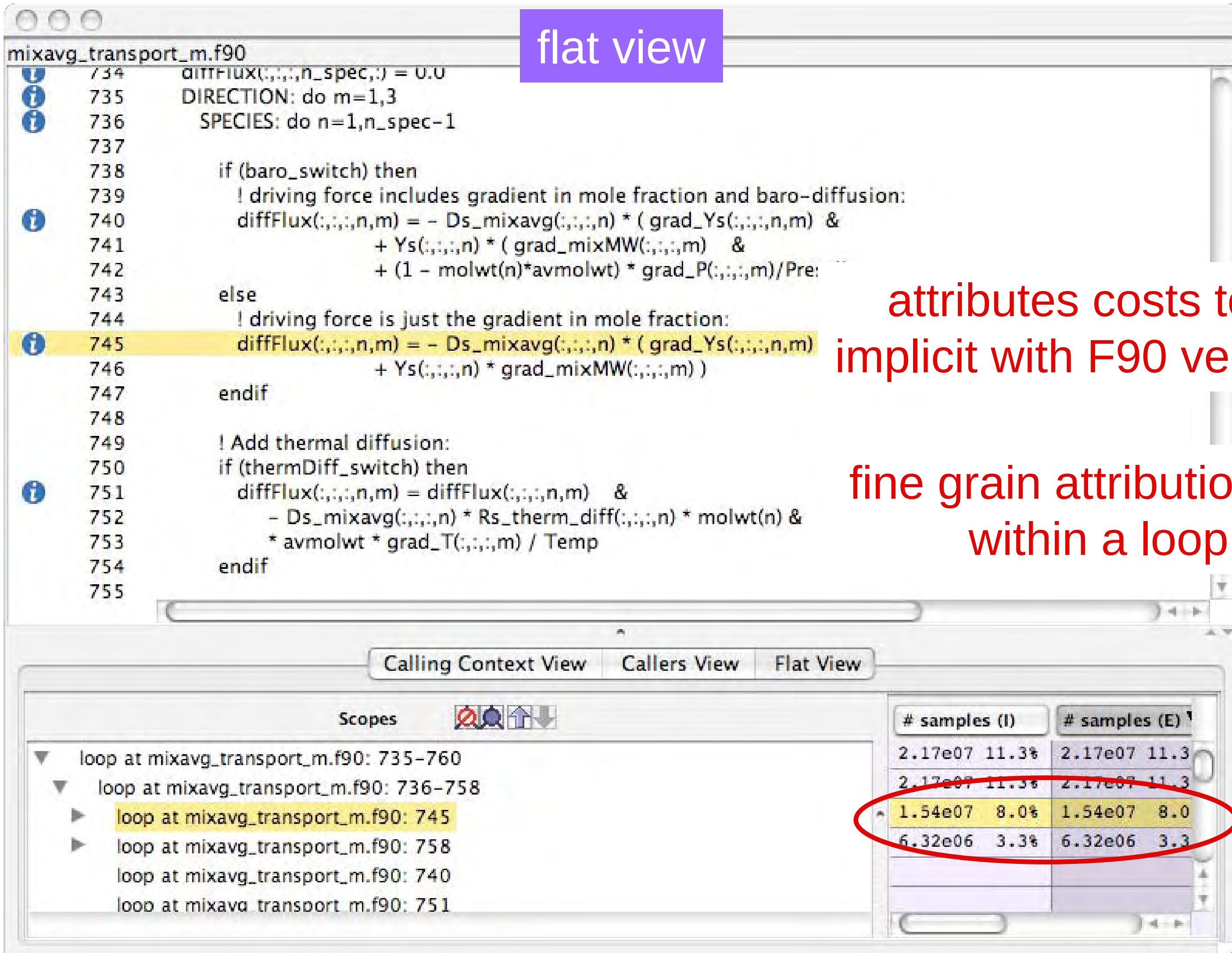
caller's view



show attribution of procedure costs to calling contexts

http://hipersoft.cs.rice.edu/hpctoolkit/

S3D Solver for Turbulent, Reacting Flows

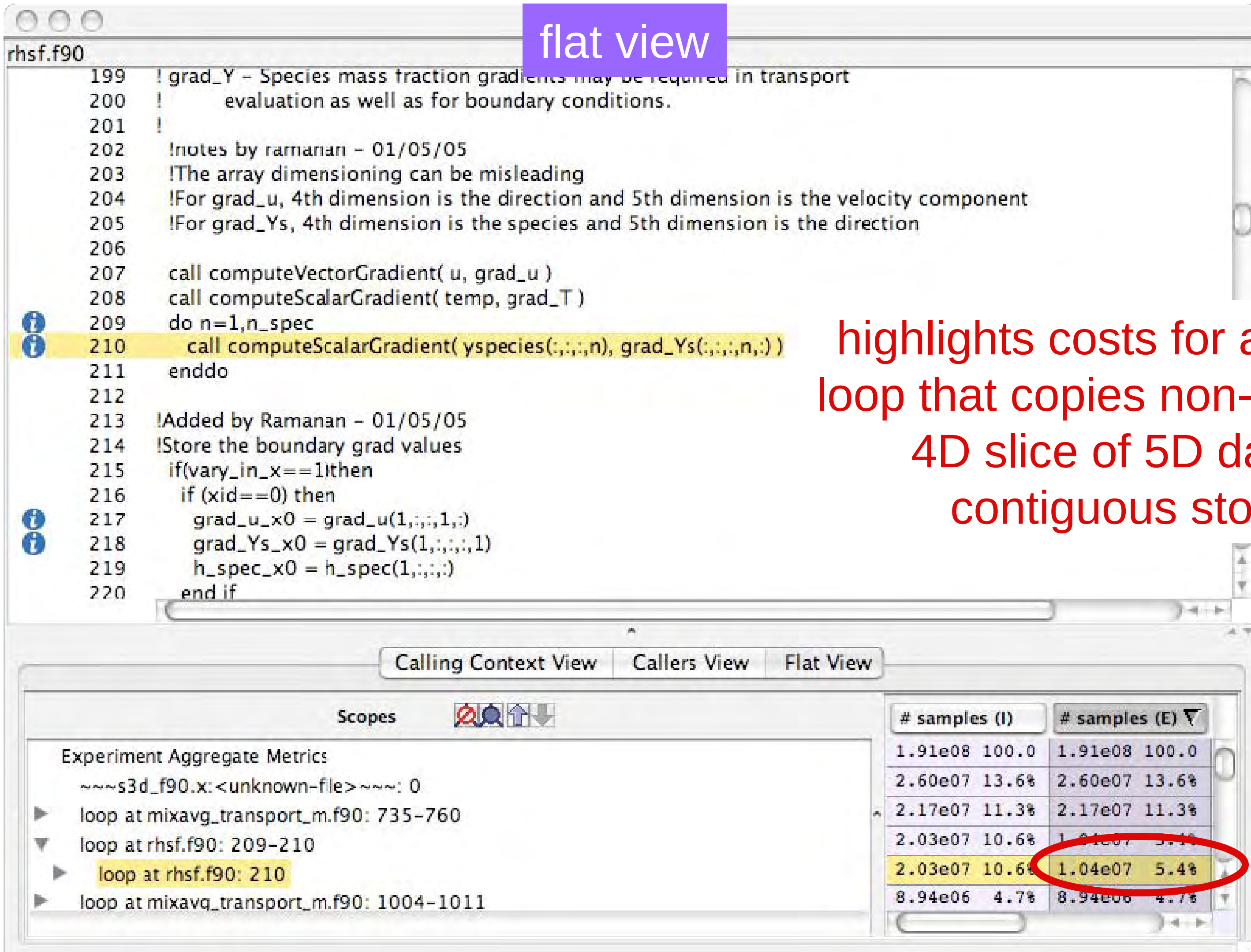


attributes costs to loops implicit with F90 vector syntax

fine grain attribution to loops within a loop nest

http://hipersoft.cs.rice.edu/hpctoolkit/

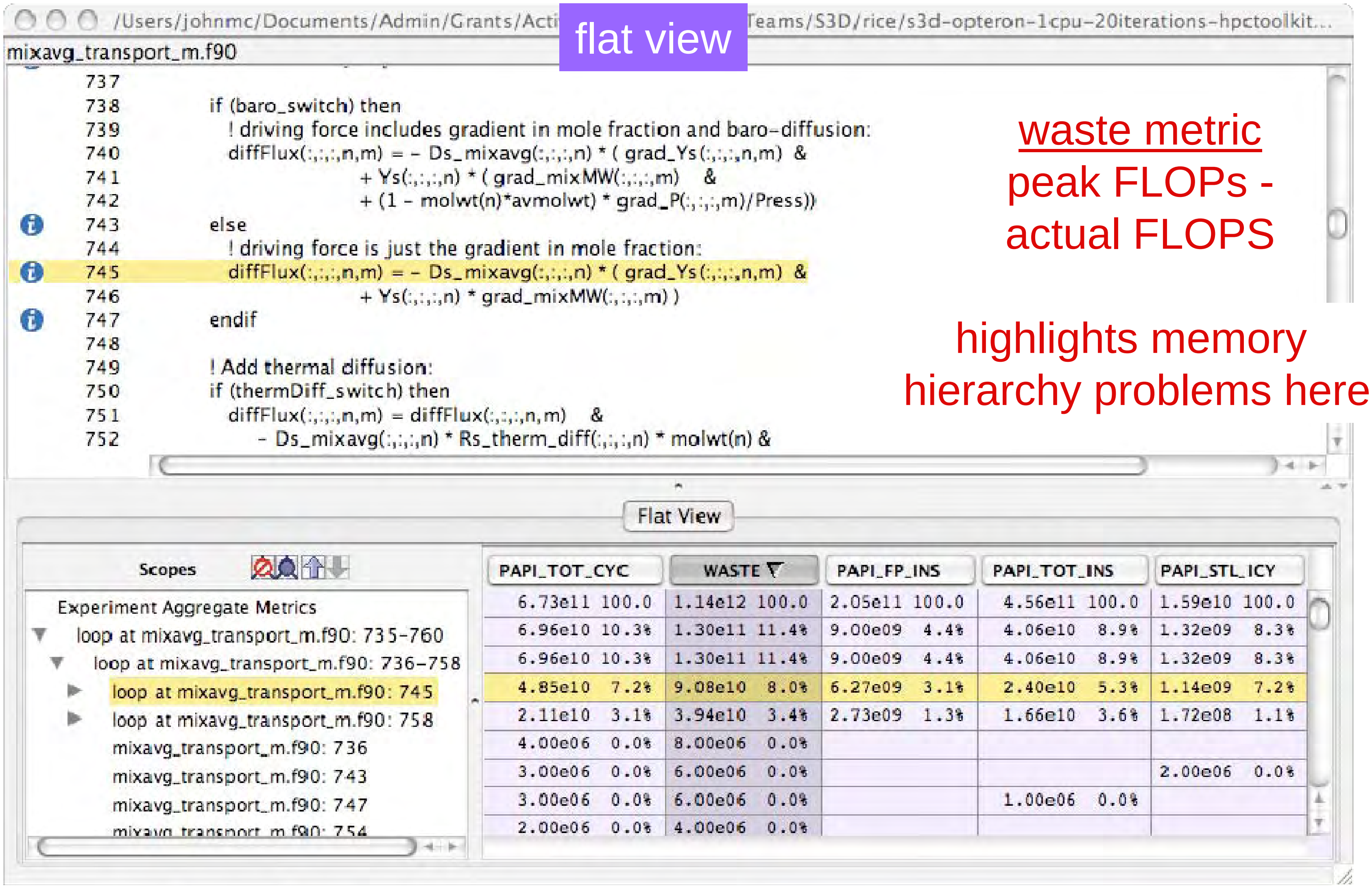
S3D Solver for Turbulent, Reacting Flows



highlights costs for an implicit loop that copies non-contiguous 4D slice of 5D data to contiguous storage

http://hipersoft.cs.rice.edu/hpctoolkit/

S3D Solver for Turbulent, Reacting Flows



http://hipersoft.cs.rice.edu/hpctoolkit/

Status

- Research prototype available only on Cray XD1
  - —being refined for broader use
- Porting to Catamount and CNL for Cray XT3 & XT4
  - —support for statically-linked binaries
- Adding support for HW counter call path profiling
- Adding support for comparative analysis
  - —viewer currently analyzes node programs
  - —enhance to analyze processes
    - within executions
    - across executions

 http://hipersoft.cs.rice.edu/hpctoolkit/

Acknowledgments

- HPCToolkit development is currently being supported by the Department of Energy's Office of Science as part of the SciDAC
  - Performance Engineering Research Institute (PERI) under Cooperative Agreement No. DE-FC02-06ER25762
  - Center for Scalable Application Development Software (CScADS) under Cooperative Agreement No. DE-FC02-07ER25800
  - Experiments performed on Cray XD1 funded by the National Science Foundation under Grant CNS-0421109, and a partnership between Rice University, Advanced Micro Devices, and Cray