Profiling with Rice HPCToolkit

Mark W. Krentel
Department of Computer Science
Rice University
krentel@rice.edu

SciDAC
Scientific Discovery through Advanced Computing

http://hpctoolkit.org

RICE

HPCToolkit Basic Features

- **Run application natively and every 100-200 times per second, interrupt program, unwind back to main(), record call stack, and combine these into a calling context tree (CCT).**
- **Combine sampling data with a static analysis of the program structure for loops, inline functions, etc.**
- **Present top-down, bottom-up and flat views of calling context tree (CCT) and time-sequence trace view. Metrics are displayed per source line in the context of their call path.**
- **Can sample on Wallclock (itimer), POSIX timers and Hardware Performance Counter Events (PAPI preset and native events): cycles, flops, cache misses, etc.**

HPCToolkit Advanced Features

- **Finely-tuned unwinder to handle multi-lingual, fully-optimized code, no frame pointers, broken return pointers, stack trolling, etc.**
- **Derived metrics -- compute flops per cycle, or flops per memory reads, etc. and attribute to lines in source code.**
- **Compute strong and weak scaling loss, for example:**

  strong: 8 * (time at 8K cores) - (time at 1K cores)

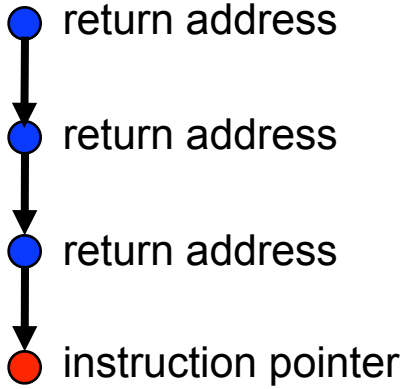
  weak: (time at 8K cores and 8x size) - (time at 1K cores)
- **Load imbalance -- display distribution and variance in metrics across processes and threads.**
- **Blame shifting -- when thread is idle or waiting on a lock, blame the working threads or holder of lock.**

Call Path Profiling

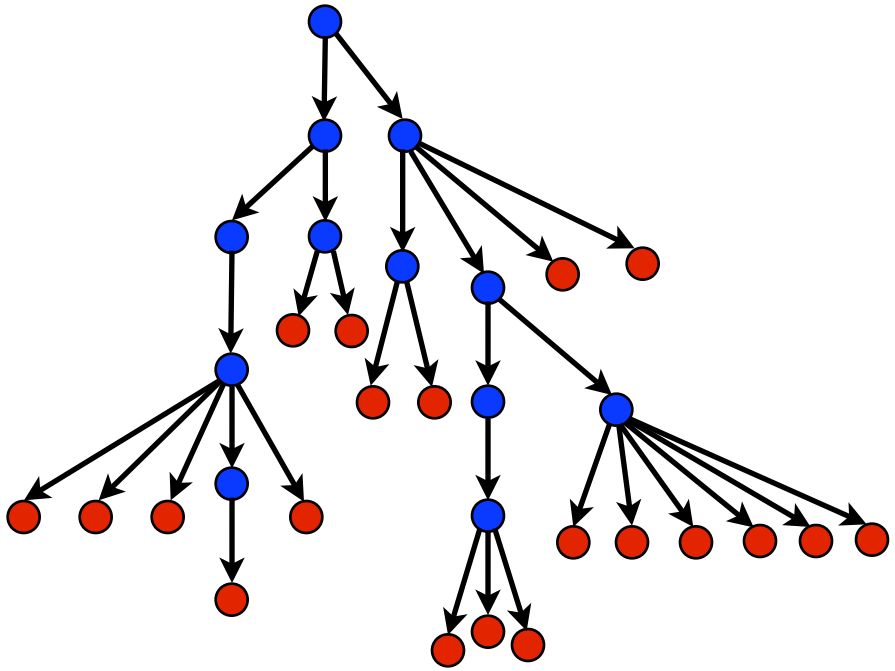
Measure and attribute costs in context

- **sample timer or hardware counter overflows**
- **gather calling context using stack unwinding**

Call path sample



Calling context tree



Overhead proportional to sampling frequency...
...not call frequency

Where to find HPCToolkit

- **Home site: user's manual, build instructions, links to source code, download viewers:**

  http://hpctoolkit.org/

- **On mira, vesta, cooley:**

  /projects/Tools/hpctoolkit/pkgs-mira/hpctoolkit/bin

  /projects/Tools/hpctoolkit/pkgs-vesta/hpctoolkit/bin

- **Source code now on GitHub:**

  http://github.com/hpctoolkit

  git clone https://github.com/hpctoolkit/hpctoolkit

  git clone https://github.com/hpctoolkit/hpctoolkit-externals

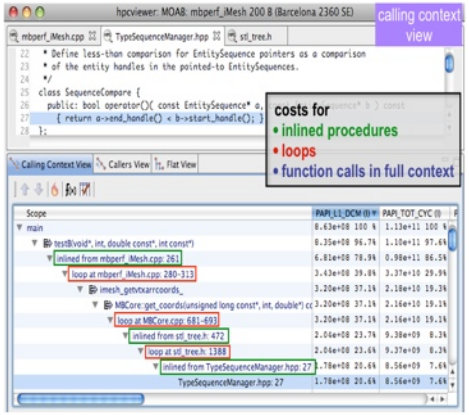
- **Send questions to:**

  hpctoolkit-forum at mailman.rice.edu

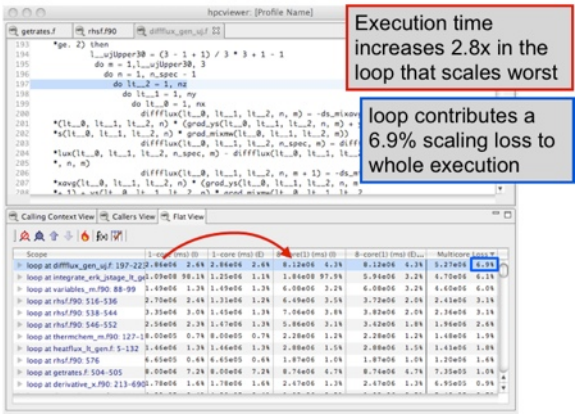
Using OpenMP Tools Library

- **Replace -fopenmp with libomp.a from llvm-openmp in hpclink. On vesta:**
  - **/projects/Tools/hpctoolkit/pkgs-vesta/llvm-openmp-gnu/lib**
- **Use hpclink from hpctoolkit-ompt.**
  - **/projects/Tools/hpctoolkit/pkgs-vesta/hpctoolkit-ompt**
- **Add event OMP_IDLE (no number) plus time-based event: WALLCLOCK or PAPI_TOT_CYC.**
- **Someday, this will be part of the OpenMP Tools standard, but not yet.**

HPCToolkit Capabilities at a Glance

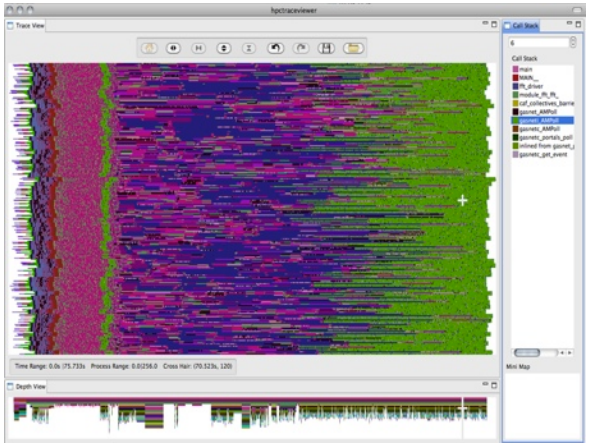


Attribute Costs to Code

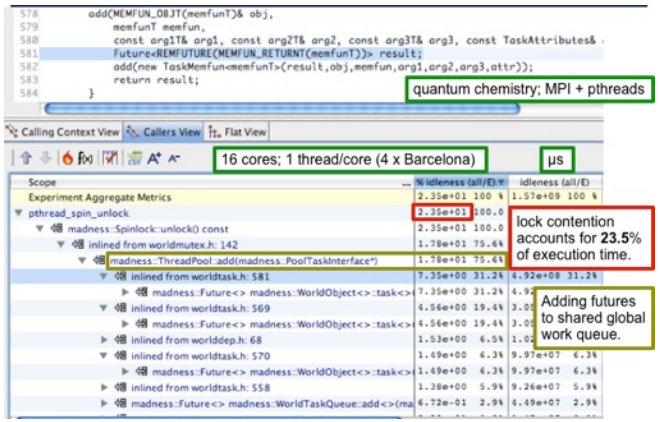


Pinpoint & Quantify Scaling Bottlenecks

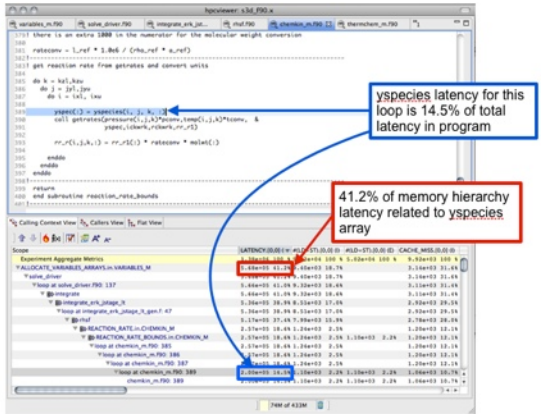
Assess Imbalance and Variability



Analyze Behavior over Time



Shift Blame from Symptoms to Causes



Associate Costs with Data

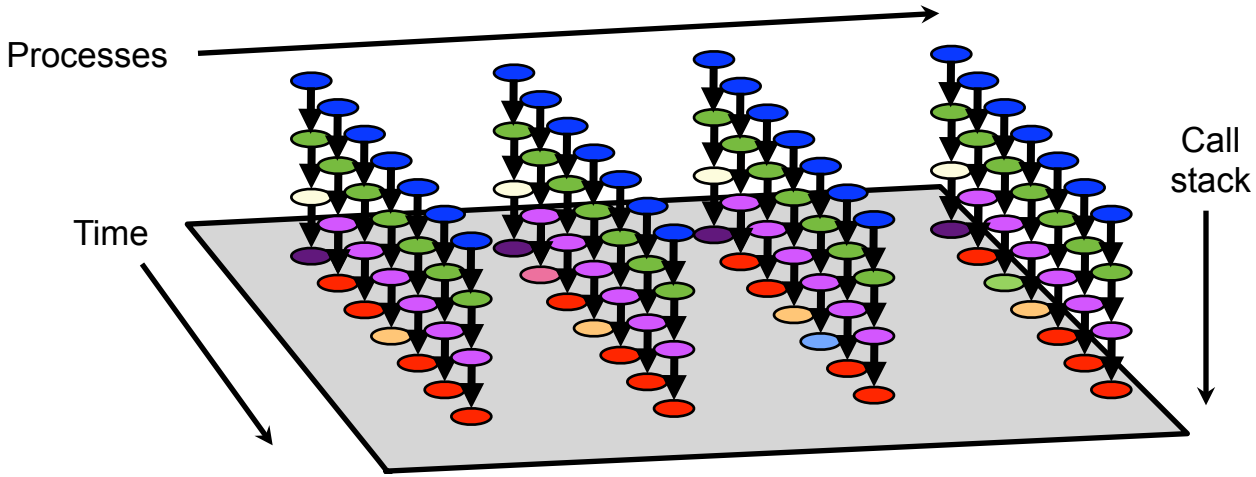
RICE

hpctoolkit.org

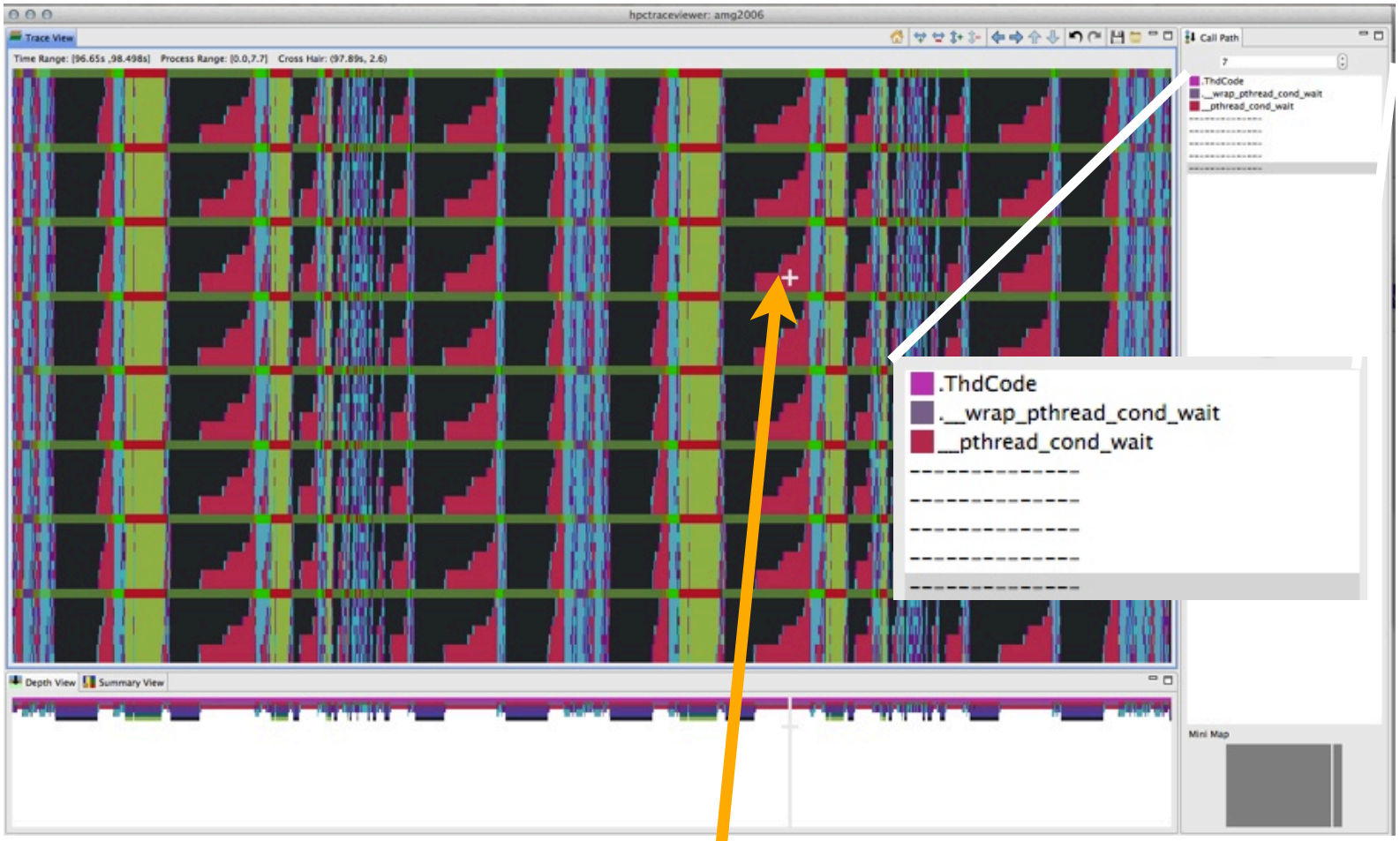
Understanding Temporal Behavior

- **Profiling compresses out the temporal dimension**
  - —temporal patterns, e.g. serialization, are invisible in profiles
- **What can we do? Trace call path samples**
  - —sketch:
    - N times per second, take a call path sample of each thread
    - organize the samples for each thread along a time line
    - view how the execution evolves left to right
    - what do we view?

      assign each procedure a color; view a depth slice of an execution

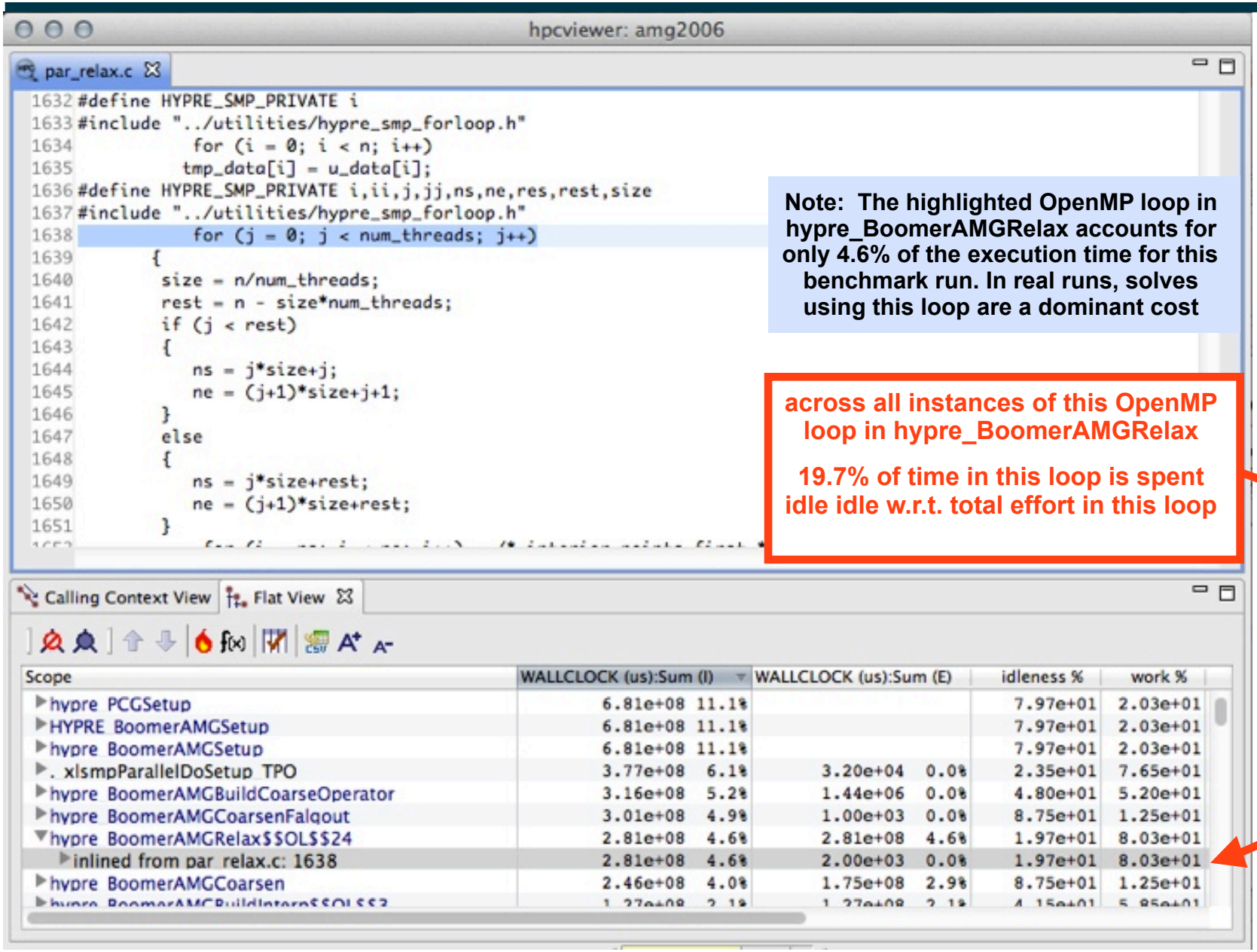


AMG2006: 8PE x 8 OMP Threads

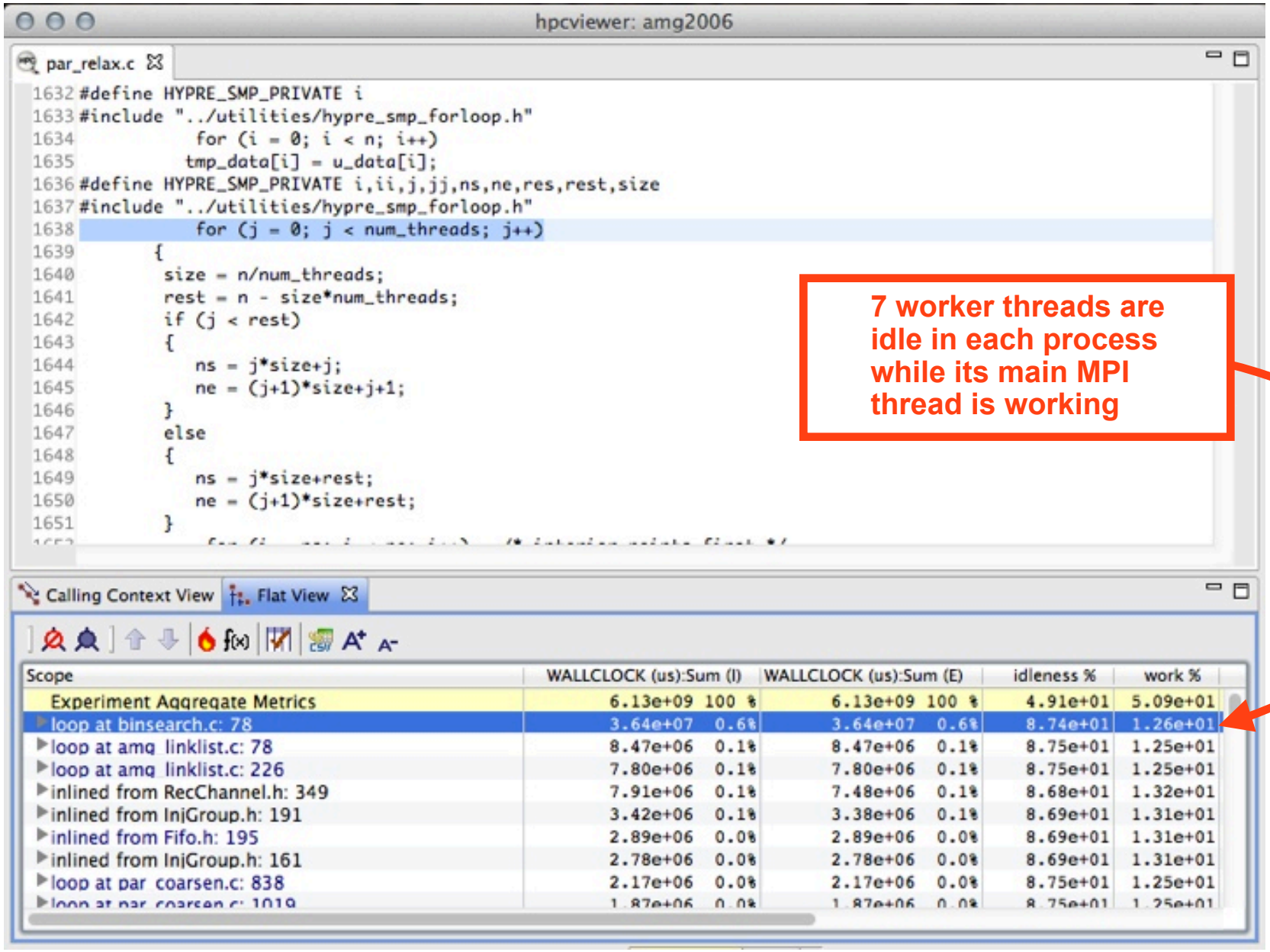


OpenMP loop in `hypre_BoomerAMGRelax` using static scheduling has load imbalance; threads idle for a significant fraction of their time

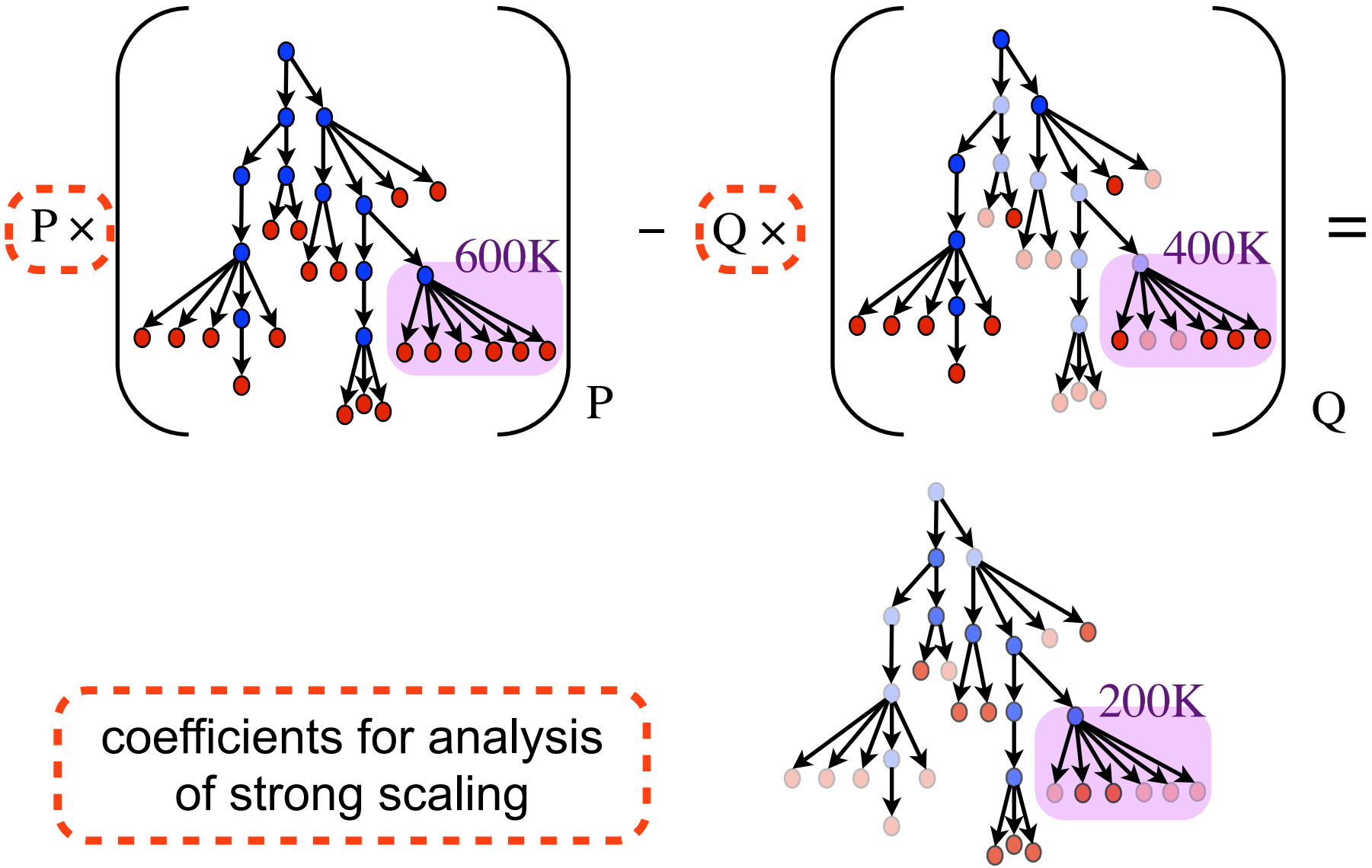
Code-centric view: `hypre_BoomerAMGRelax`



Serial Code in AMG2006 8 PE, 8 Threads



Pinpointing and Quantifying Scalability Bottlenecks



coefficients for analysis of strong scaling