Critically Missing Pieces on Accelerators: A Performance Tools Perspective

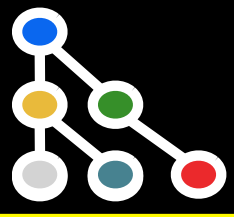
Milind Chabbi, Karthik Murthy,
Mike Fagan, and John Mellor-Crummey
Rice University

SC 2013
Denver, CO
November 20, 2013

What Is Missing in GPUs?

- Source-level performance measurement/attribution
  - Kernel is the finest granularity of performance monitoring
- Sampling support
  - Sampling is essential for scalable performance monitoring on large parallel systems
- Instantaneous notification
  - Idle to/from working transitions
- O/S and/or driver activity information
- Standard performance tools interface across multiple accelerator programming models
  - CUDA, OpenCL, OpenACC, HSA, OpenMP

HPCToolkit

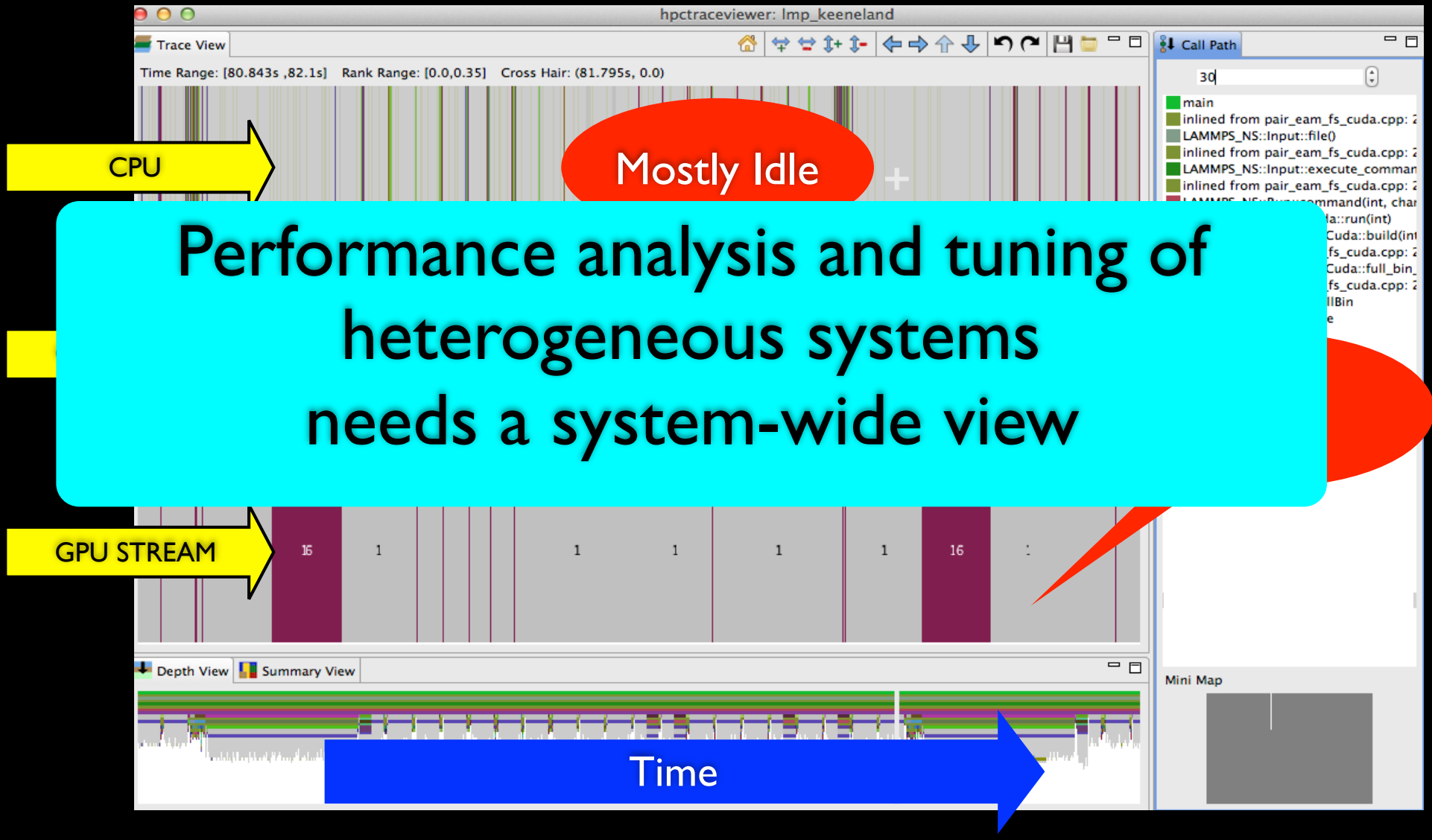


Performance tool for large parallel systems

- Supports multilingual, fully-optimized, statically or dynamically linked applications (no source modification)
- Measures performance using asynchronous sampling of timers and hardware performance counters
  - ✦ Low overhead (under 5%) for both profiling and tracing
- Attributes performance to full call paths
  - ✦ Pthreads, OpenMP, MPI, and any combination
- Decentralized → Scales to 1000s of nodes
- Provides GUIs for code- and time- centric analysis

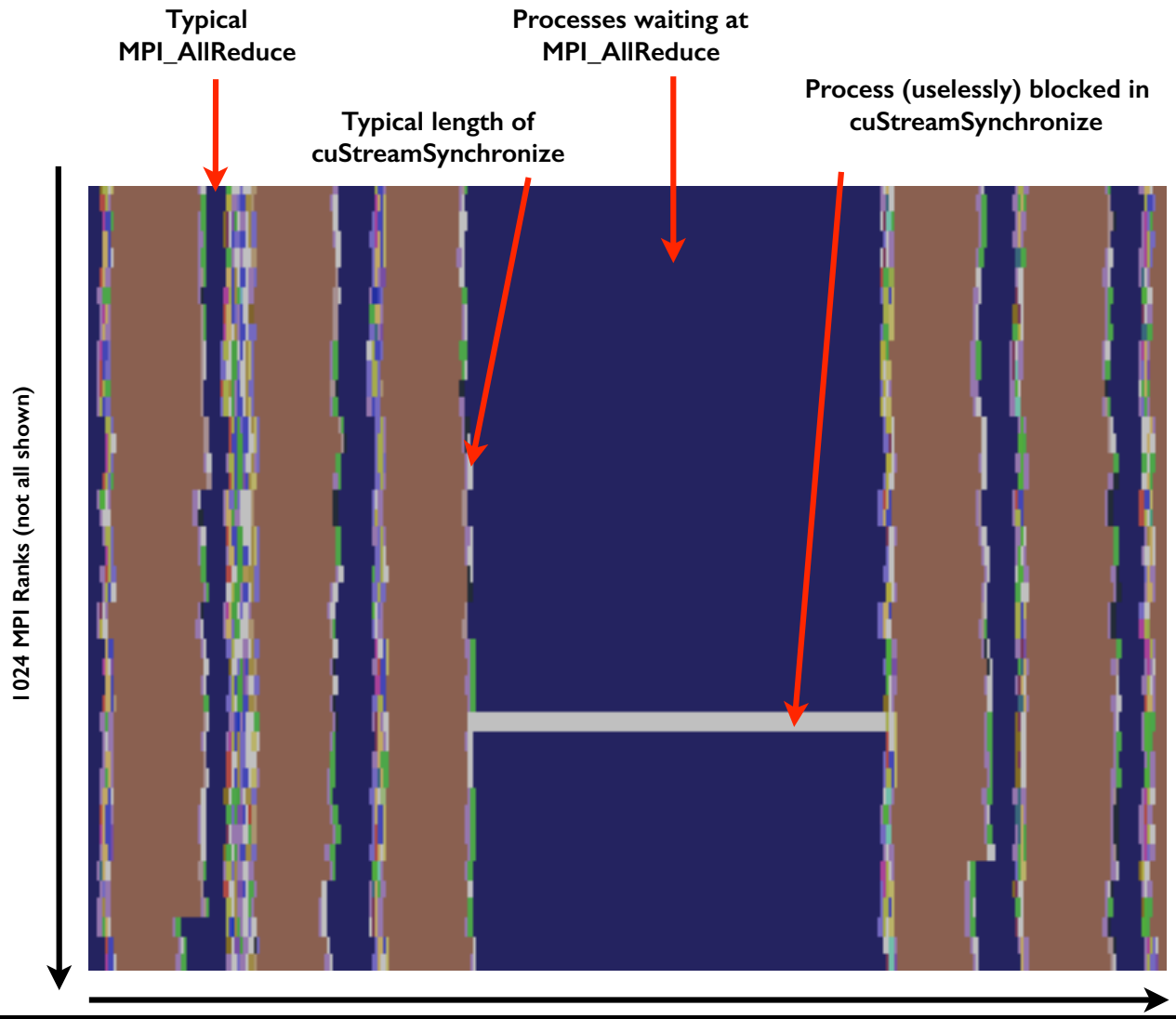
Motivation for Idleness Analysis

LAMMPS-CUDA on Keeneland



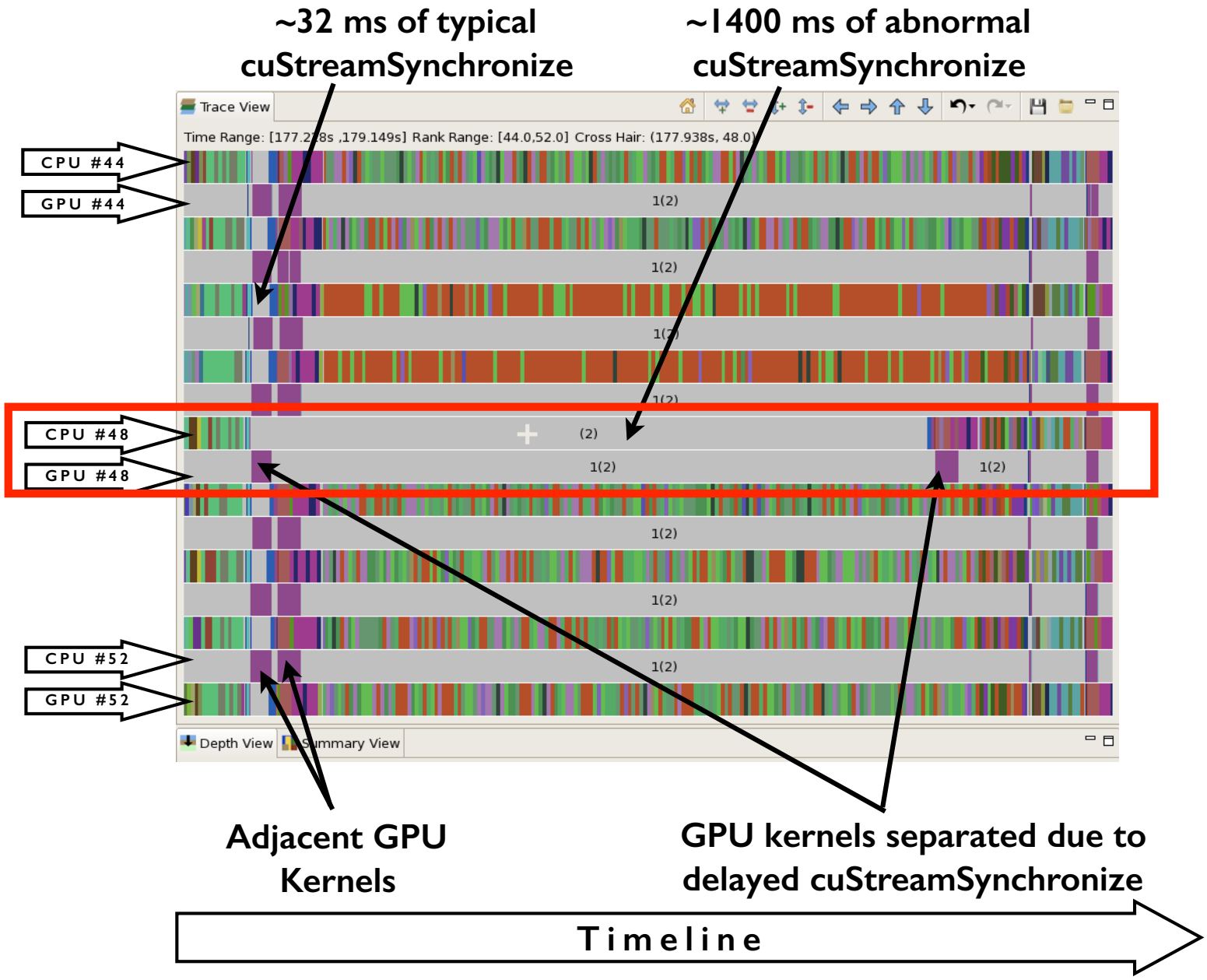
Operating System Blocking on Titan

LAMMPS



One blocked process can sabotage the collective communication

Operating System Blocking on Titan



Pinpoint and Quantify Idleness

- Idleness offers a promising avenue for tuning
  - ✦ Offloading entire computation to GPUs wastes CPU
  - ✦ Performing entire computation on CPUs wastes GPU
- Quantify resource idleness
- Attribute idleness to its causes
  - ✦ If GPU is idle → blame CPU code for not offloading (enough) work
  - ✦ If CPU is waiting for results from GPU → blame GPU kernel(s) involved

a.k.a. "CPU-GPU Blame Shifting"

Limitations Using CUPTI

CUPTI: CUDA Profiling Tools Interface

- Counters are not kernel-specific
- Must serialize kernels or work with throughput metrics
- Kernel is the finest granularity of GPU counters
- Activity API
  - ✦ Can be queried only at synchronization points
  - ✦ Does not record H/W counters
  - ✦ Need to collect timestamped traces to correlate CPU and GPU activities (problematic to use with HPCToolkit)

Implementation Challenges for Blame Shifting in CUDA

- Problem: GPUs neither provide idleness notifications nor account CPU idleness
  - ✦ Did not want to collect traces (CPU/GPU) for profiling
- Solution: Bracketed GPU kernels with cudaEvents
  - ✦ Query events during CPU samples to infer idleness or attribute blame
- Limitations
  - ✦ Can't query events from signal handlers when the CPU is inside CUDA runtime
    - ★ Nvidia does not foresee meeting this need
  - ✦ Events serialize concurrent kernels until GK110
- Implementation: CUDA function overriding to avoid host serialization

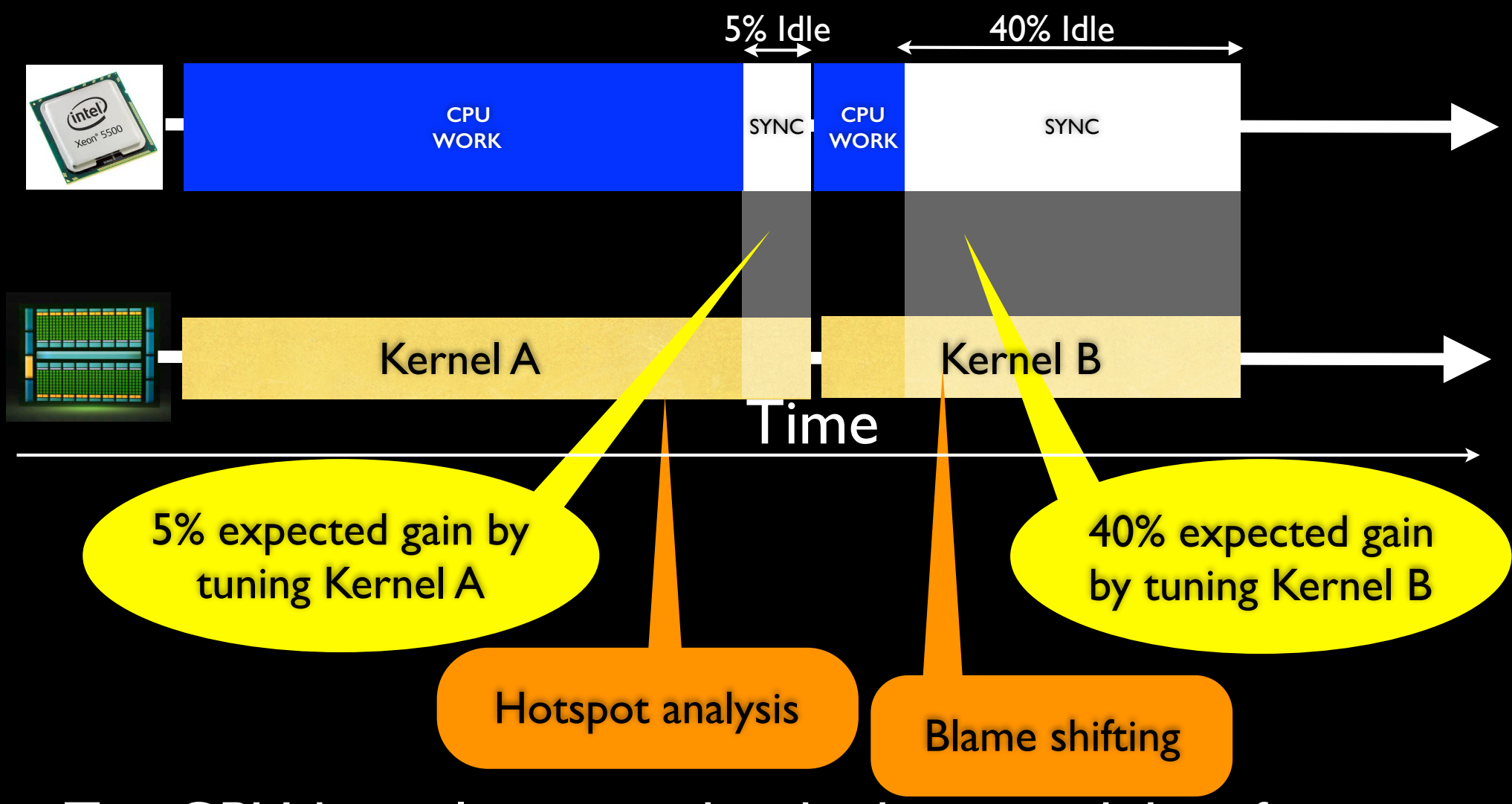
Conclusions

- Need sampling support for performance monitoring on accelerators
- Need instantaneous work/idle notification
- Need source-level attribution mechanism within kernels
- Need better performance diagnostic capability from inside the O/S and drivers
- Need unified performance tools interface across accelerator programming models

hpctoolkit.org

Backup slides

Quantifying Tuning Opportunities with CPU-GPU Blame Shifting



Top GPU-kernel may not be the best candidate for tuning

Project Goals

- Provide performance analysis tools for emerging heterogeneous supercomputers
  - GPU kernels alone → whole application performance analysis
- Provide performance improvement "insights"
  - Guide developer toward tuning opportunities
  - Relate data back to source code and calling context
  - Attribute blame to CPU and GPU code that cause idleness
- Quantitative assessment w/o trace collection
- Introduce minimal execution perturbation
  - Collect compact measurement data at scale