Hard Real-time Scheduling for Parallel Run-time Systems

Peter Dinda Xiaoyang Wang Jinghang Wang
Chris Beauchene Conor Hetland

Prescience Lab
Department of EECS
Northwestern University

pdinda.org
presciencelab.org

Paper in a Nutshell

- HPC node OS as an RTOS
  - Isolation in time-shared environment
  - Resource control with commensurate performance
  - Coordination via time instead of via synchronization
    - Barrier removal example
- Hard real-time threads in Nautilus kernel
  - Despite x64
  - ~10 us resolution (Xeon Phi KNL)
- Thread group scheduling and coordination
  - ~3 us synchronization for 255 threads (Phi)
- Publicly available codebase

Outline

- Motivation
  - Prior work on soft RT scheduling of distributed machines
  - Modern machines and interesting runtimes
- What is hard real-time?
  - Liu model
- Implementation in Nautilus
  - Threads
  - Groups
- Performance evaluation
  - Limits (mostly on KNL)
  - Fine-grain BSP benchmark
- Conclusions and future work

Experiences with Soft Real-time

- VSched soft RT scheduler extension for Linux
- Consolidation of interactive and batch VMs
- Time-sharing of distributed memory parallel applications on a cluster with performance isolation and control
  - Coordinated scheduling (i.e., gang scheduling based on time) so BSP applications achieve resource-commensurate performance

B. Lin, P. Dinda, *VSched: Mixing Batch and Interactive Virtual Machines Using Periodic Real-time Scheduling*, SC 2005

B. Lin, A. Sundararaj, P. Dinda, *Time-sharing Parallel Applications Through Performance-targeted Feedback-controlled Real-time Scheduling*, Cluster Computing, 11:3, 2008; ICAC 2007, patent application

Can This Apply Within a Node?

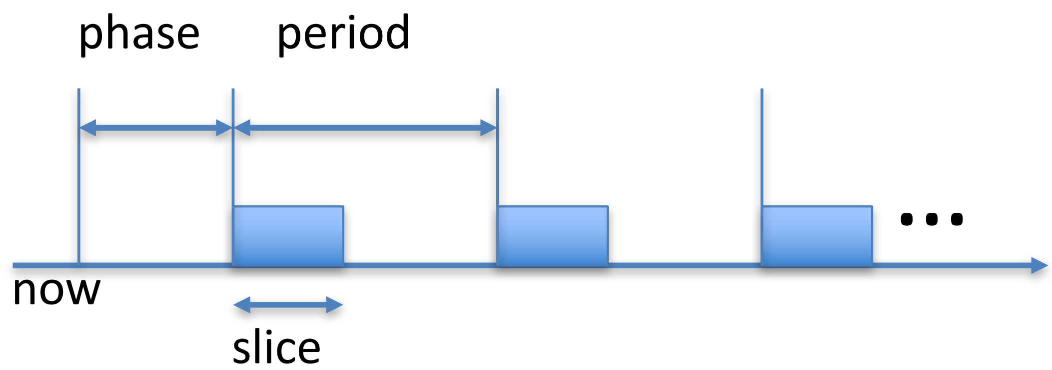
- Increasingly interesting target
  - Growing CPU count: Phi now at 256; NUMA, …
- OS noise concerns continue
- Much finer granularity scheduling and coordination needed
  - OpenMP loops and tasking
  - NESL VCODE model (abstract vector machine)
- New opportunity: substitute timing for synchronization
  - Example: potential barrier removal

What is Hard Real-time?

- Formal admission control process
  - Based on work and deadlines
  - Scheduler can say no
- Scheduler engine **guarantees** all deadlines
- Limitations
  - Scheduler overheads
  - Context switch overheads
- Our system: threads on a NUMA node

What is Hard Real-time?

- Aperiodic threads
  - Have priority
  - Always admitted
- Periodic threads
  - Phase, period, slice (deadline=period)
  - Selective admission (RMA tests)
- Sporadic threads
  - Phase, size, deadline, aperiodic priority
  - Selective admission (EDF tests)



J. W. S. Liu, *Real-time Systems*, Prentice Hall, 2000

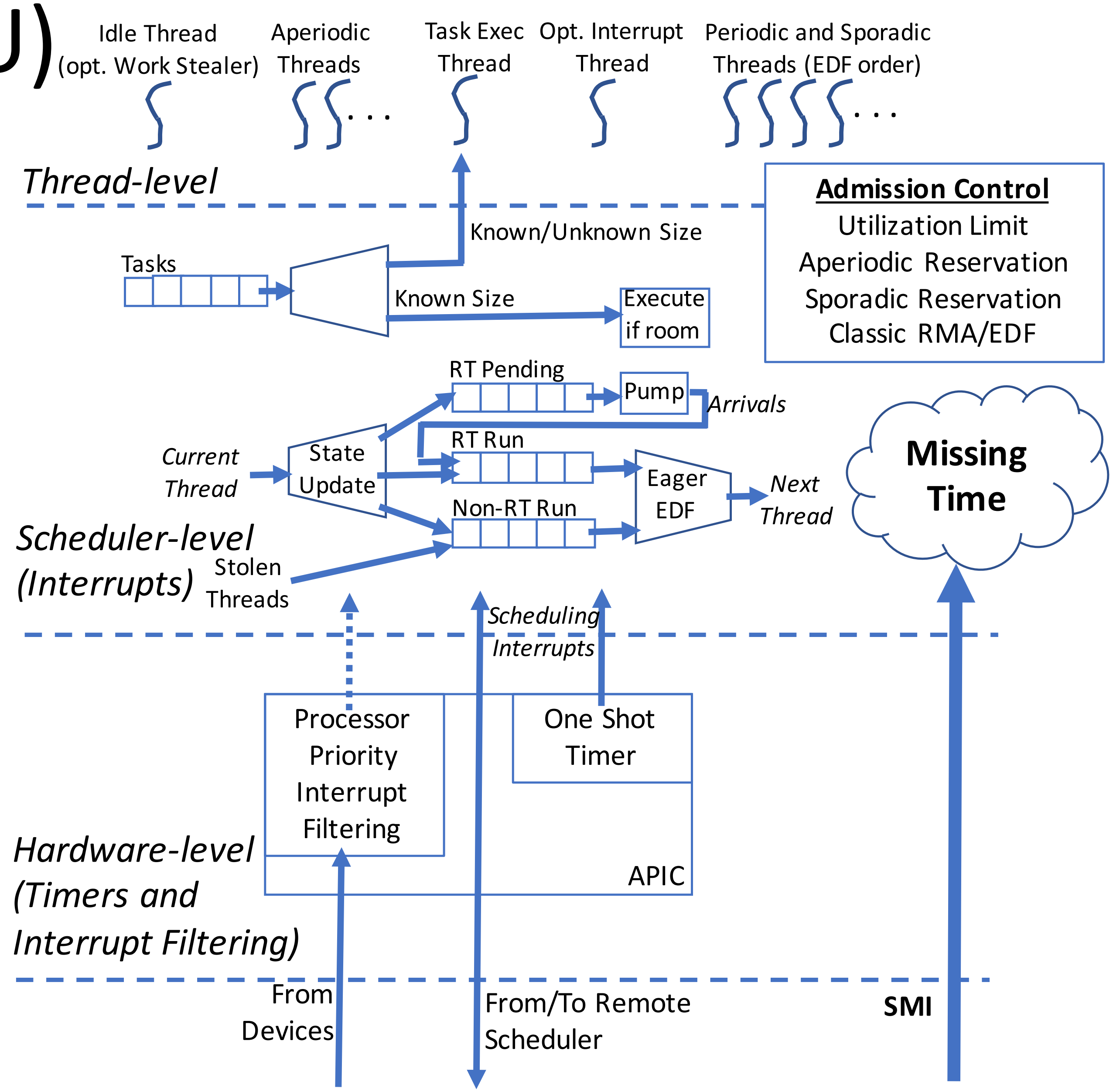
Nautilus as the Basis for an RTOS

- Nautilus: kernel framework for constructing hybrid run-times (HRTs) on x64
  - No userspace, simple address translation, single address space, streamlined primitives, NUMA, …
  - 15-40% speedup over Linux for Legion run-time
- Particularly salient for an RTOS:
  - No page faults, only capacity TLB misses
  - Deterministic path length in drivers and all core functionality
  - Steerable interrupts

K. Hale, P. Dinda, *A Case for Transforming Parallel Runtime Systems Into Operating System Kernels,* HPDC 2015
K. Hale, P. Dinda, *Enabling Hybrid Parallel Runtimes Through Kernel and Virtualization Support*, VEE 2016.
K. Hale, C. Hetland, P. Dinda, *Multiverse: Easy Conversion Of Runtime Systems Into OS Kernels Via Automatic Hybridization*, ICAC 2017

Local (per-CPU) Scheduler

- Complex, but core concept is Eager EDF
- Admission control in thread context
- Reservations for aperiodic, sporadic, missing time…
- Interrupt control



- ns-resolution time based on cycle counter (ARAT, ConstantTSC)
  - Scheduling interrupts: APIC timer (TSC deadline mode if available)

The Curse of Missing Time

- Unaccounted time within kernel itself
  - Scheduler overhead, context switch overhead, etc.

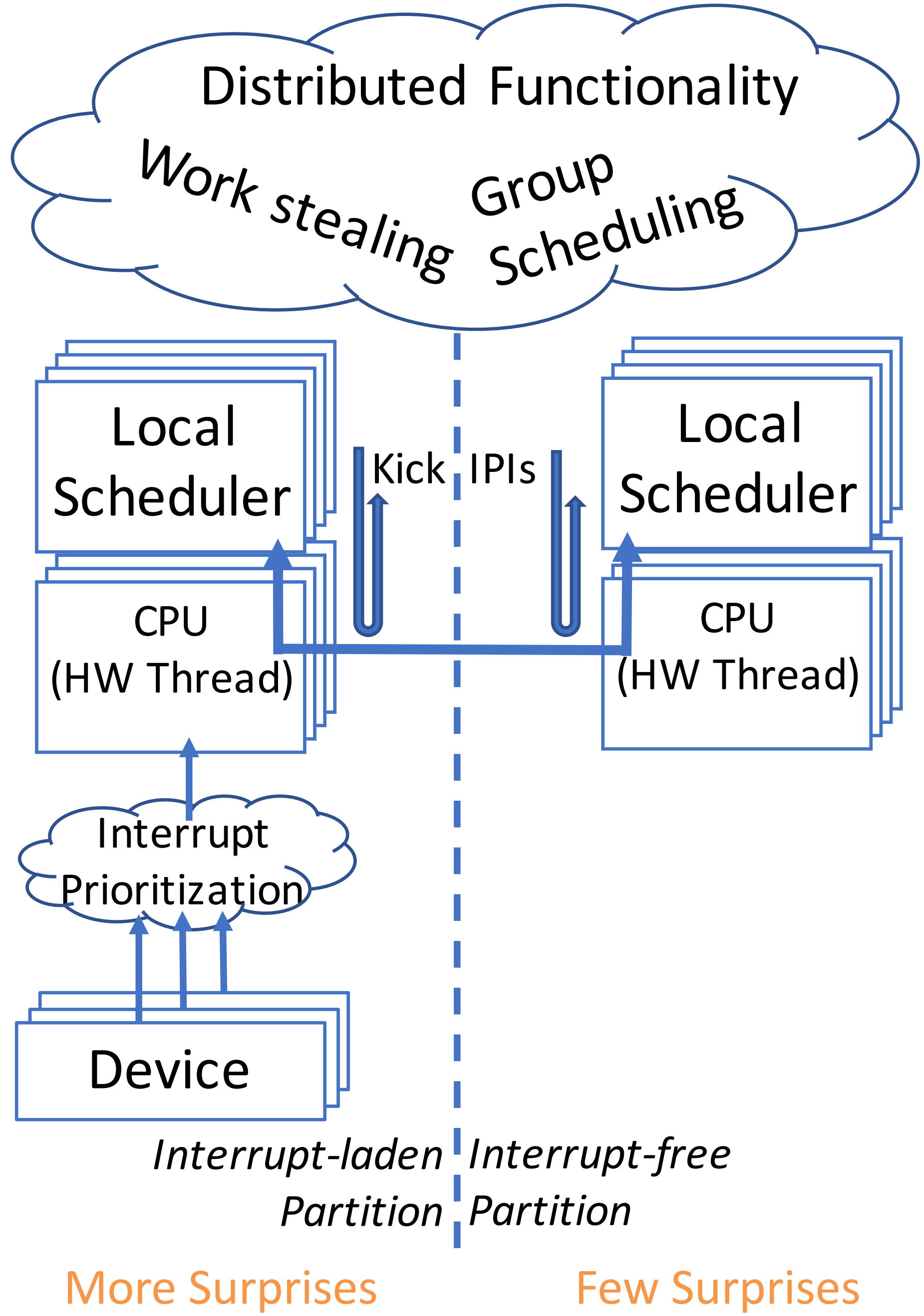
- Deliberate, nondeterministic, unaccounted time due to System Management Interrupts (SMIs)
  - Firmware-level interrupts
  - Higher privilege than kernel or even VMM
  - Cannot be turned off
  - Like an alien abduction from the scheduler's perspective
    - "My clock just jumped forward 10 us!"

- Our approach to both:
  (a) Reservations, (b) Eager Earliest Deadline First

B. Delgado, K. Karavanic, *Performance Implications of System Management Mode*, IISWC 2013

Global (per-node) Scheduler

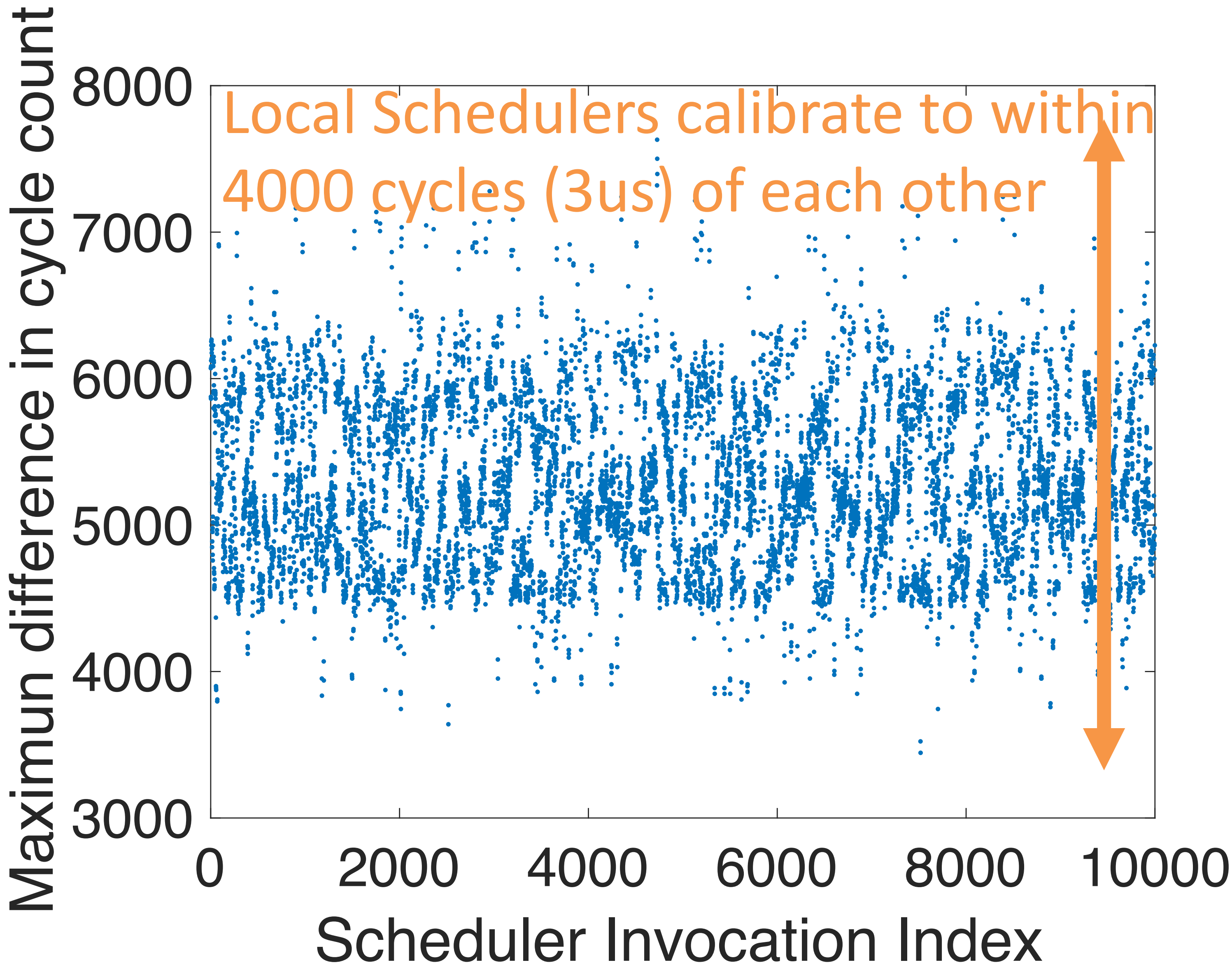
- Local scheduler coordination via
  - Time (mostly)
  - Interrupts (sparingly)
  - No global locking
- Interrupt steering and segregation
  - Interrupt-free CPUs see only scheduling-related interrupts
  - Interrupt-laden CPUs have careful interrupt control



Group Scheduling

- Local schedulers' clocks synchronized
  - Variance <1000 cycles (<1 us) over 256 CPUs on Phi
- Thread groups and group admission control
  - Main element is admission control done in parallel
  - All or nothing
- Phase correction to coordinate initial thread arrival on all involved local schedulers
- Same constraints on all local schedulers results in gang scheduling of the group of threads
  - Without explicit communication

Local Scheduler Synchronization on Phi



8 CPU example, 256 CPU similar

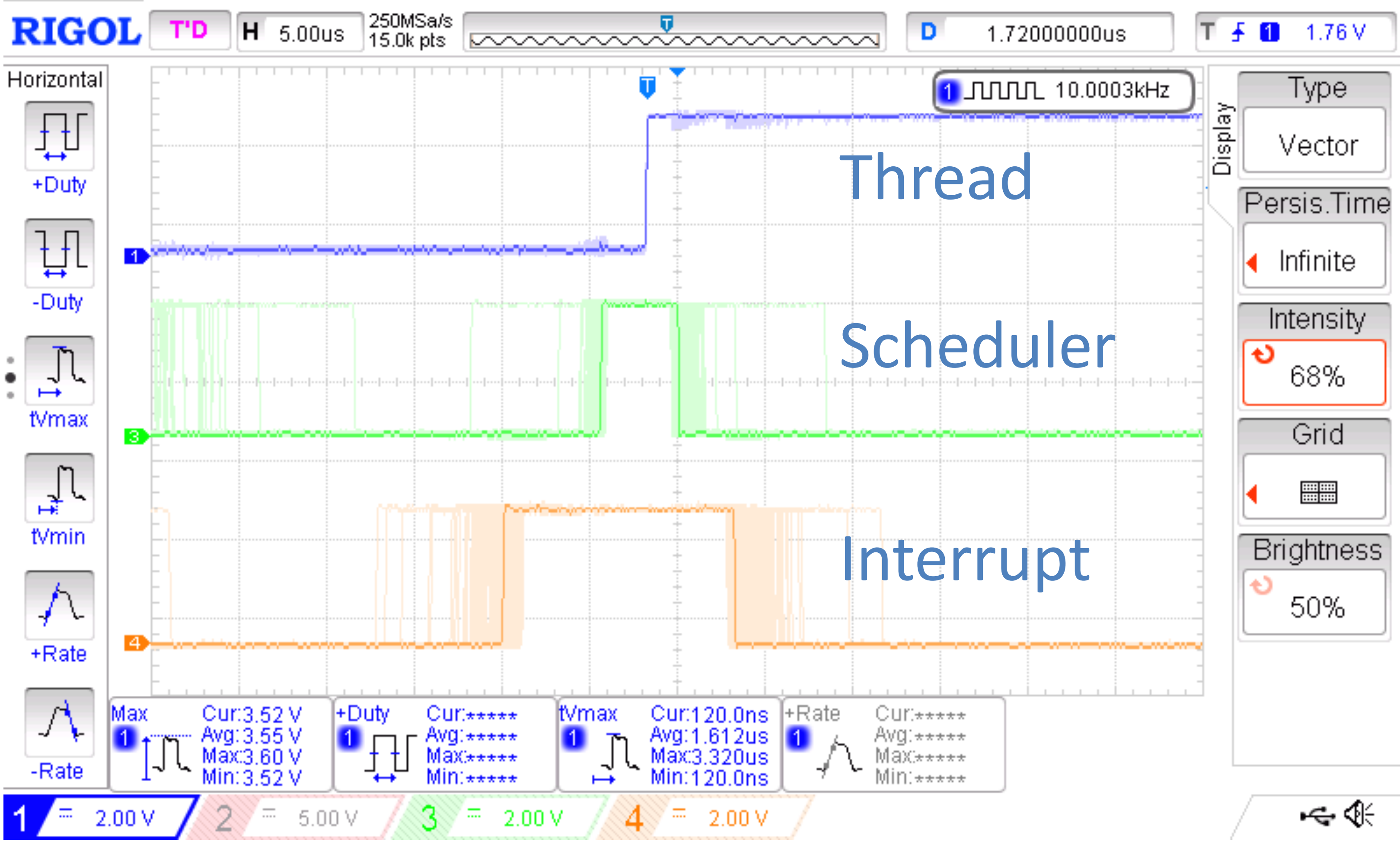
Code Measures

- Scheduler: ~5000 LoC (C)
  - Also includes work-stealing, thread pools, garbage collection support, and tasks
- Groups and Group Scheduling: ~1000 LoC (C)
- Other changes: ~2000 LoC (C+Assembly)
  - Low-level CPU-state maintenance / context switch
  - Additional thread states
  - Assorted

Test machines

- Phi
  - Supermicro 5038ki (”Colfax KNL Ninja”)
  - Intel Xeon Phi 7210 (“Knight’s Landing”)
    - 64 cores, 4 hardware threads per core
    - 1.3 GHz
    - 16 GB MCDRAM, 96 GB DRAM
    - All throttling/burst behavior disabled in BIOS
- R415
  - Dell R415
  - AMD 4122
    - 2 sockets, 8 cores/threads total,
    - 2.2 GHz
    - 16 GB DRAM
    - All throttling/bust behavior disabled in BIOS

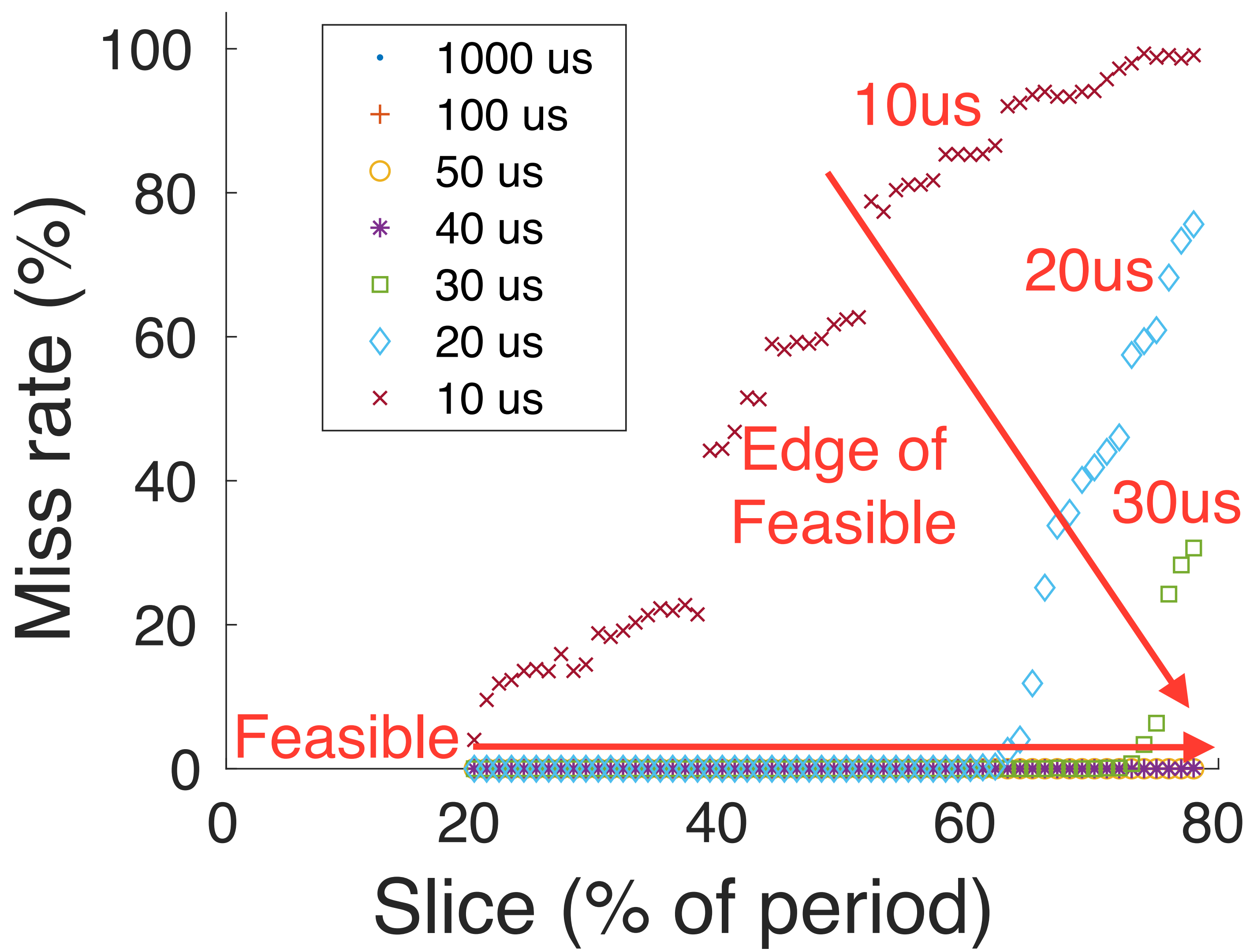
Validation Through External Monitoring



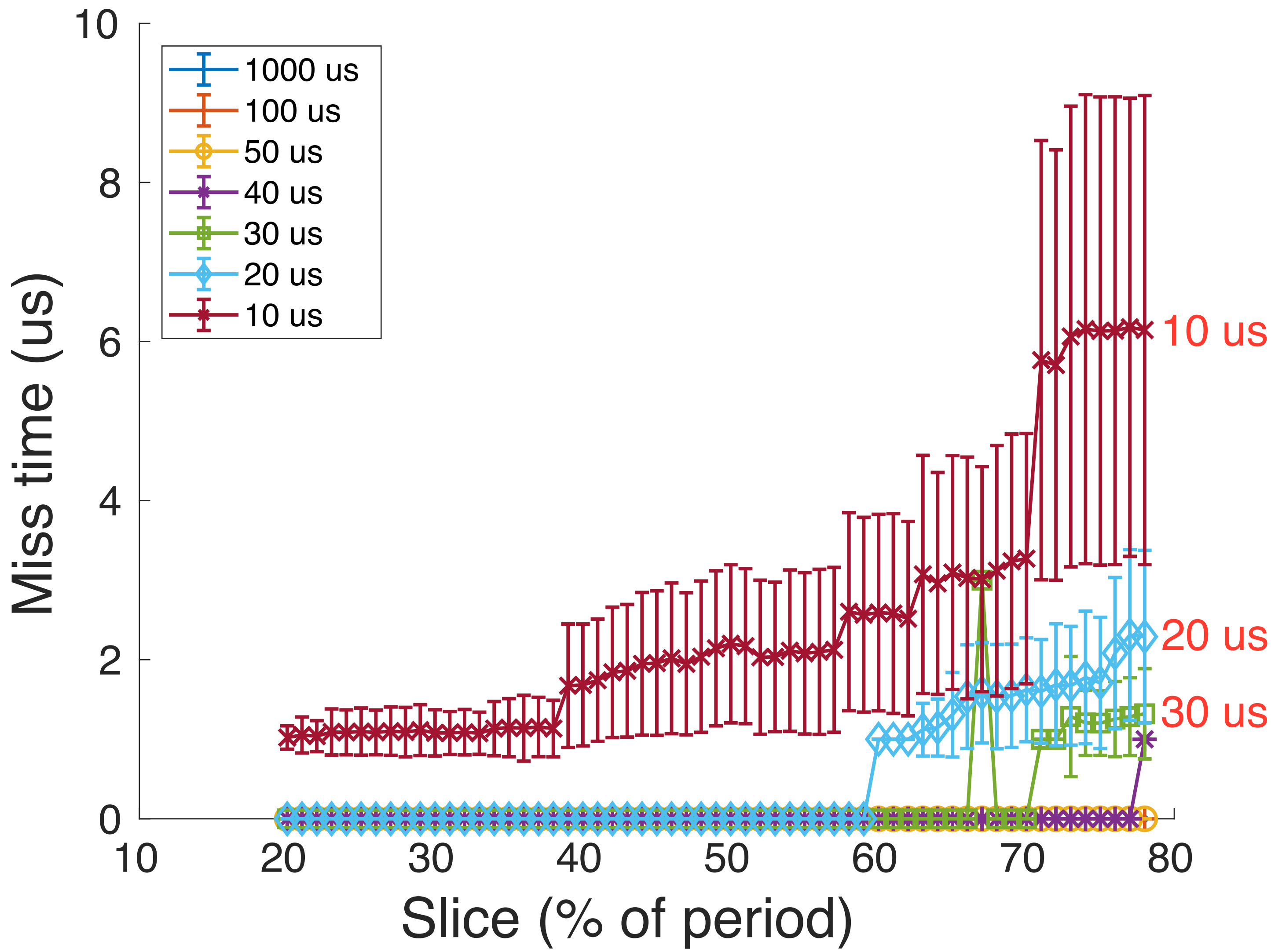
Phi with parallel port attached to oscilloscope

period=100us, slice=50us, phase=0

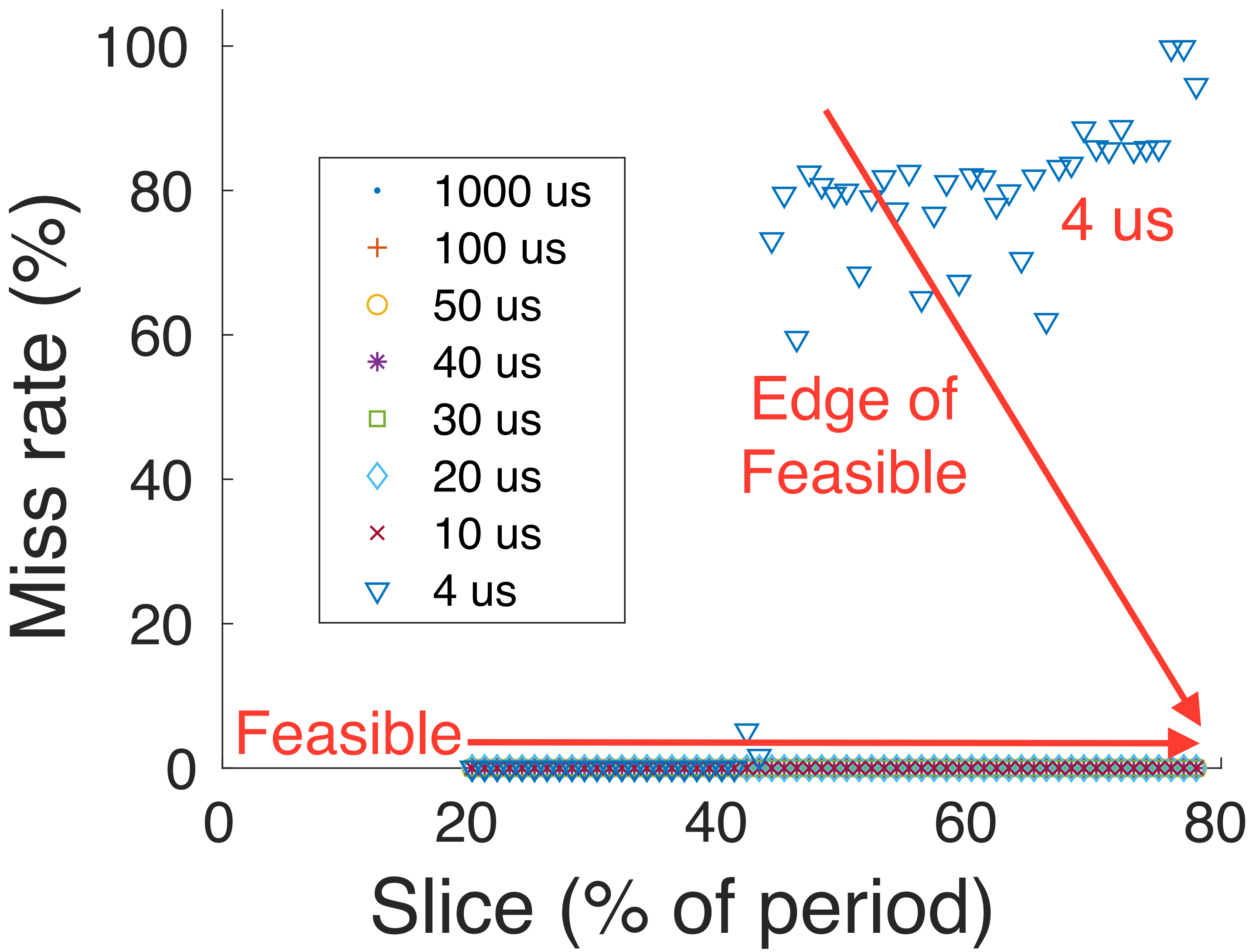
Limits on Phi: ~10us



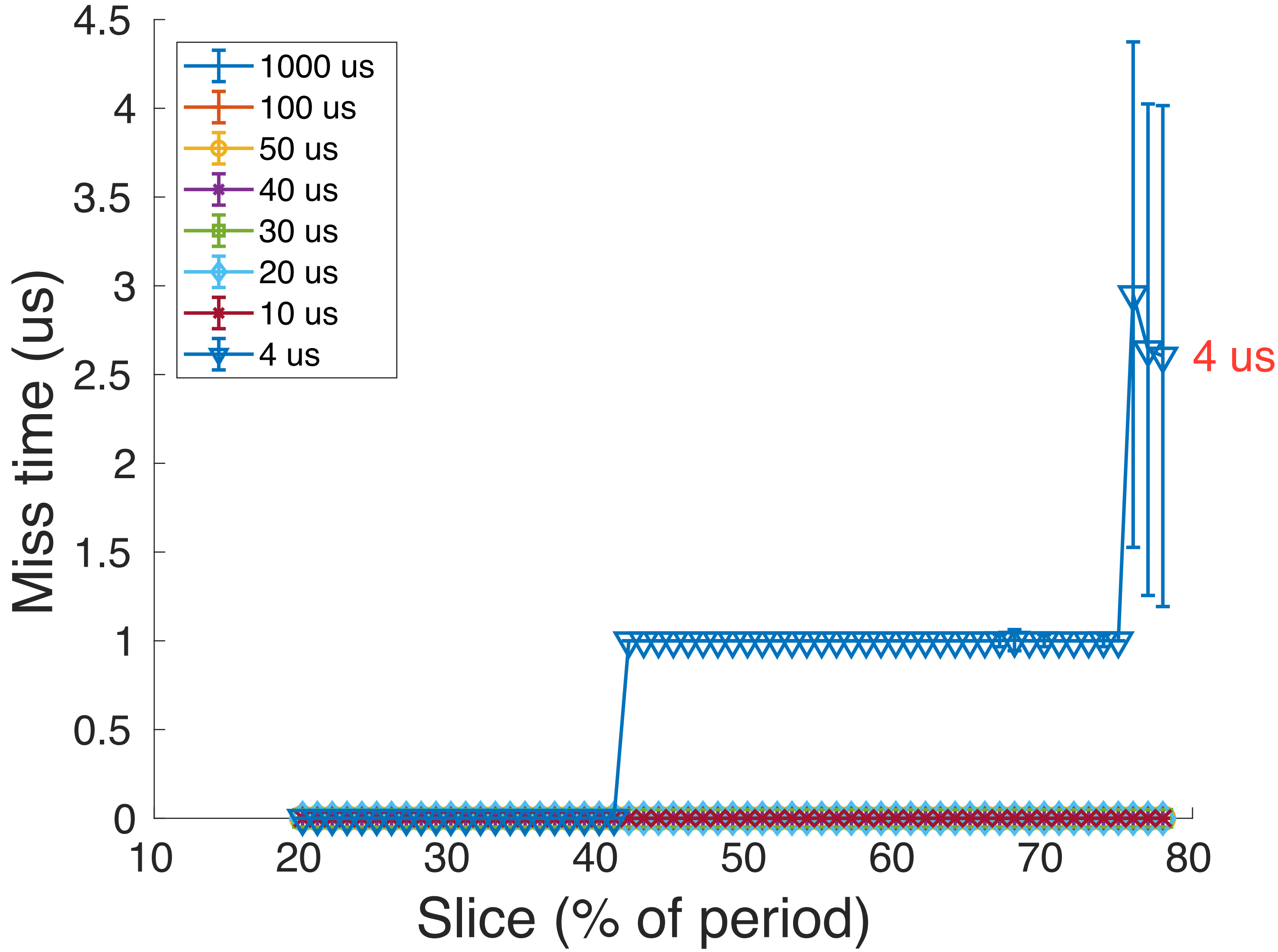
Controlled Miss Behavior on Phi



Limits on R415: ~4us



Controlled Miss Behavior on R415

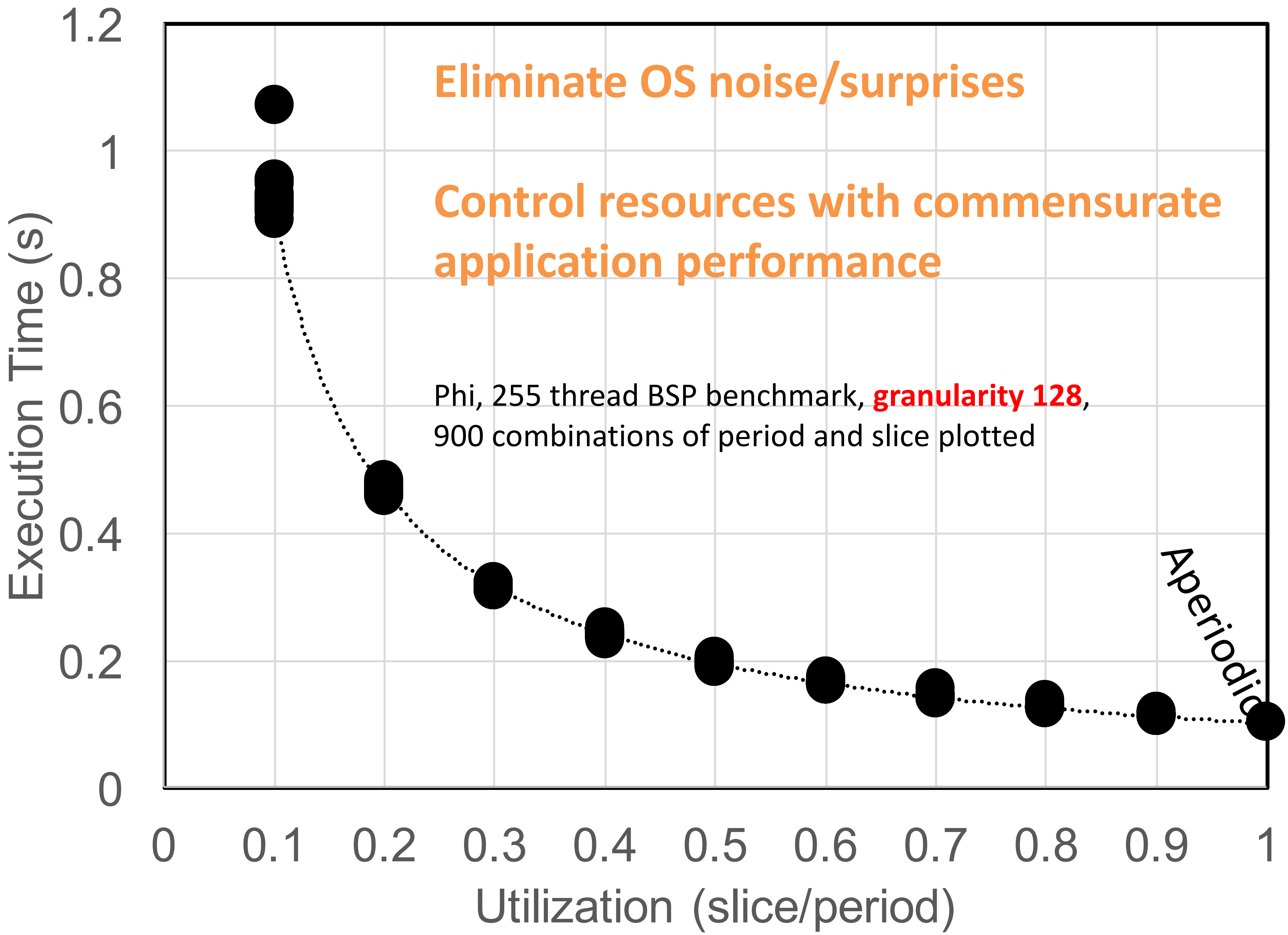


Fine-grain BSP Microbenchmark

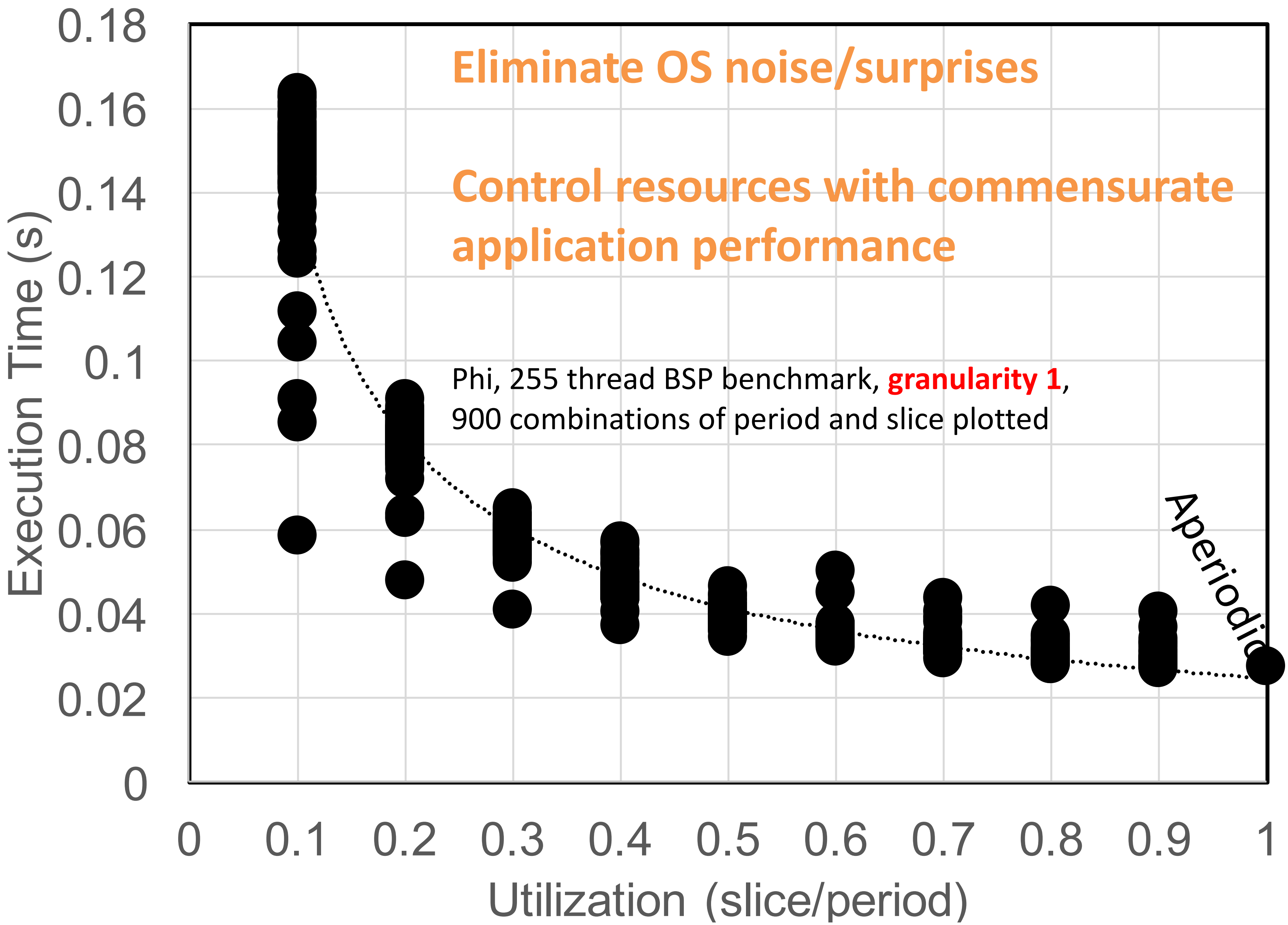
```
each thread in group:
  for (i=0;i<N;i++) {
    local_compute(granularity)
    optional_barrier();
    write_to_neighbor(granularity)
    optional_barrier();
  }
```

Barrier overhead grows with shrinking granularity
Barriers could be removed if the threads ran in lock-step

Resource Control With Commensurate Performance



Resource Control With Commensurate Performance



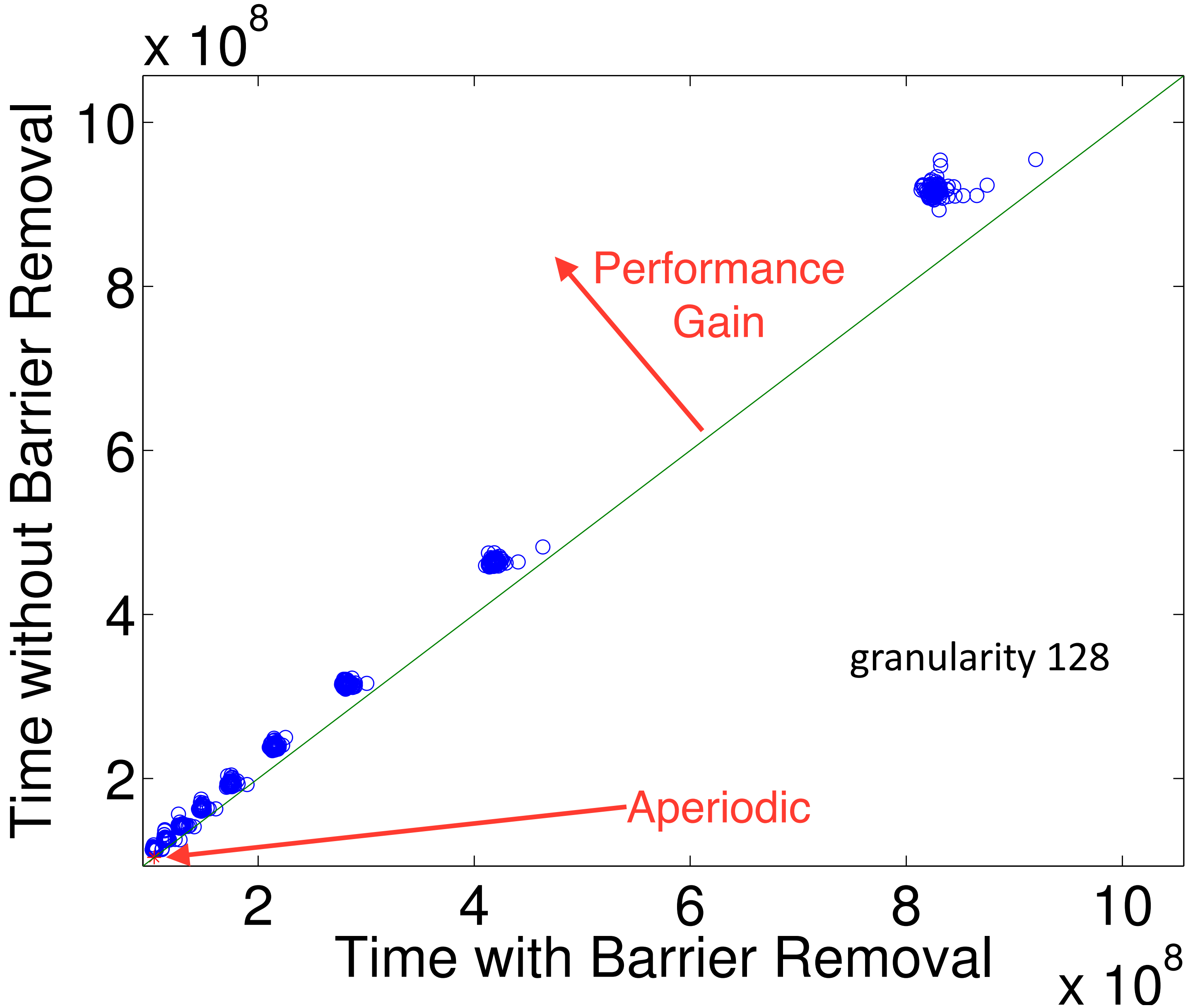
Fine-grain BSP Microbenchmark

```
each thread in group:
  for (i=0;i<N;i++) {
    local_compute(granularity)
    optional_barrier();
    write_to_neighbor(granularity)
    optional_barrier();
  }
```

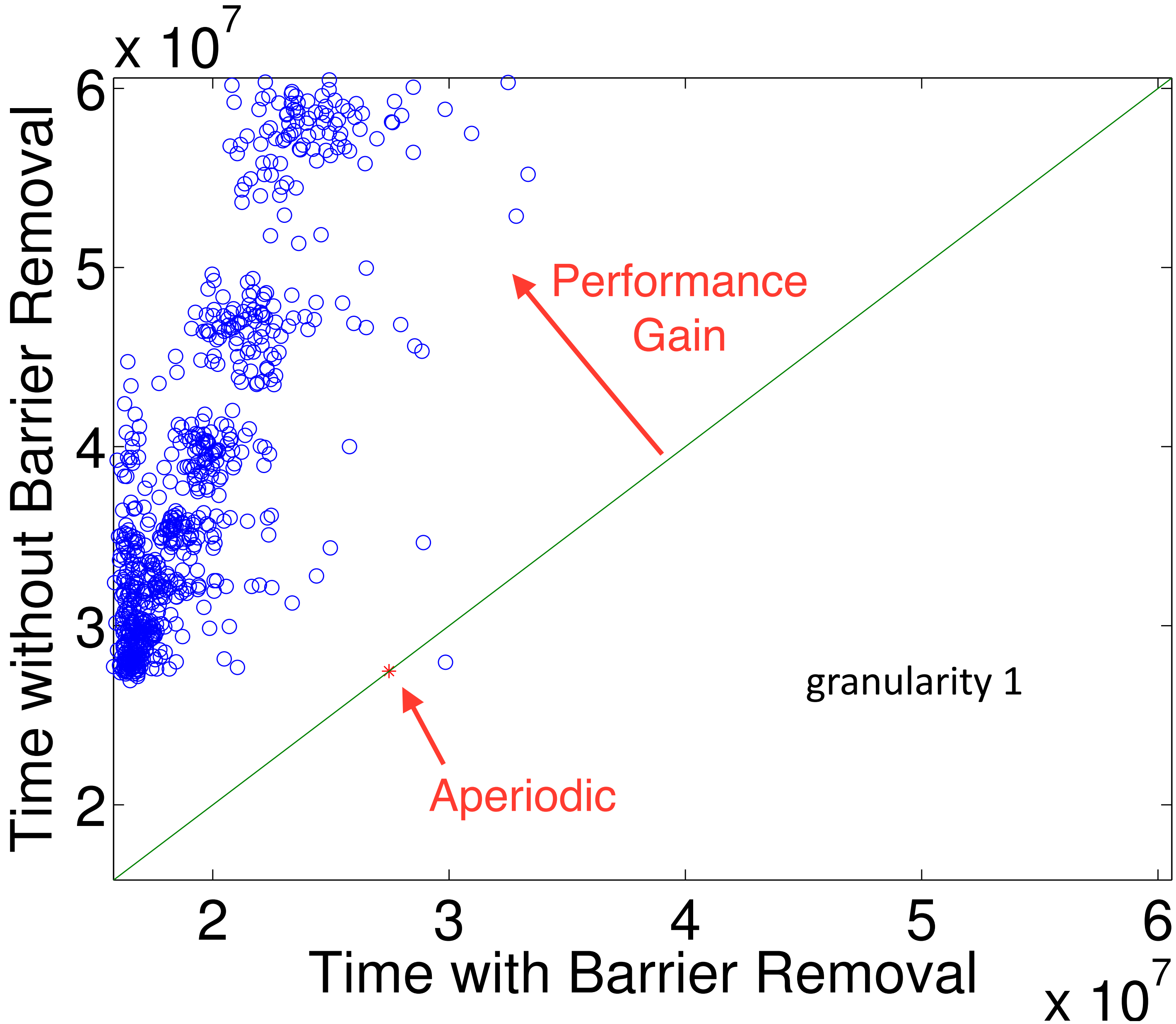
Barrier overhead grows with shrinking granularity

Barriers could be removed if the threads ran in lock-step

Barrier Removal



Barrier Removal



Related Work

- OS Noise
- Gang scheduling
- Vsched / Coordinated soft RT
  - As in motivation
- Mondragon
- RTVirt
- Tesselation
- Barrelfish
  - Also coordination via time

Ongoing/Future Work

- Further overhead reduction
  - Reduce granularity
- Real-time tasks
- Interrupt-free scheduling
  - Avoid interrupt overheads / Reduce granularity
  - Compiler-based injection of cooperative scheduling calls
- Real-time executive model
  - Scheduling implemented at compile-time as a superloop, as in safety-critical and/or smallest embedded systems
- Custom hardware for scheduling and synchronization
  - Intel HARP / FPGA

Paper in a Nutshell

- HPC node OS as an RTOS
  - Isolation in time-shared environment
  - Coordination via time instead of via synchronization
  - Barrier removal example
- Hard real-time threads in Nautilus kernel
  - Despite x64
  - ~10 us resolution (Xeon Phi KNL)
- Thread group scheduling and coordination
  - ~3 us synchronization for 255 threads (Phi)
- Publicly available codebase

For More Information

- Peter Dinda
  - pdinda@northwestern.edu
  - http://pdinda.org
- Codebase available
  - http://v3vee.org
- Prescience Lab
  - http://presciencelab.org
- Acknowledgements
  - NSF, DOE