35th Salishan Meeting: Random Access Session

HPX-5 Runtime System Overhead Times

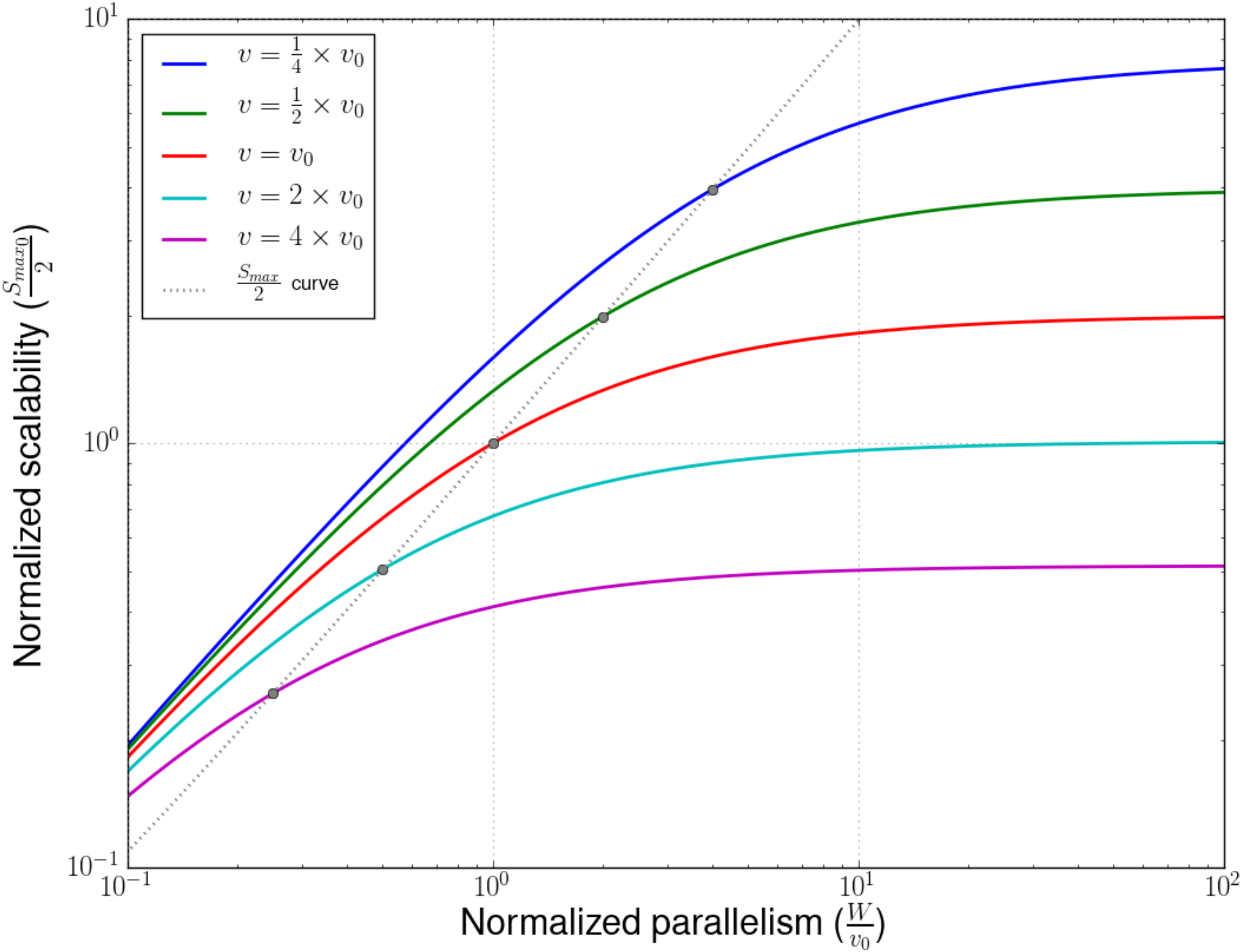
Thomas Sterling

Chief Scientist, CREST
Professor, School of Informatics and Computing
Indiana University

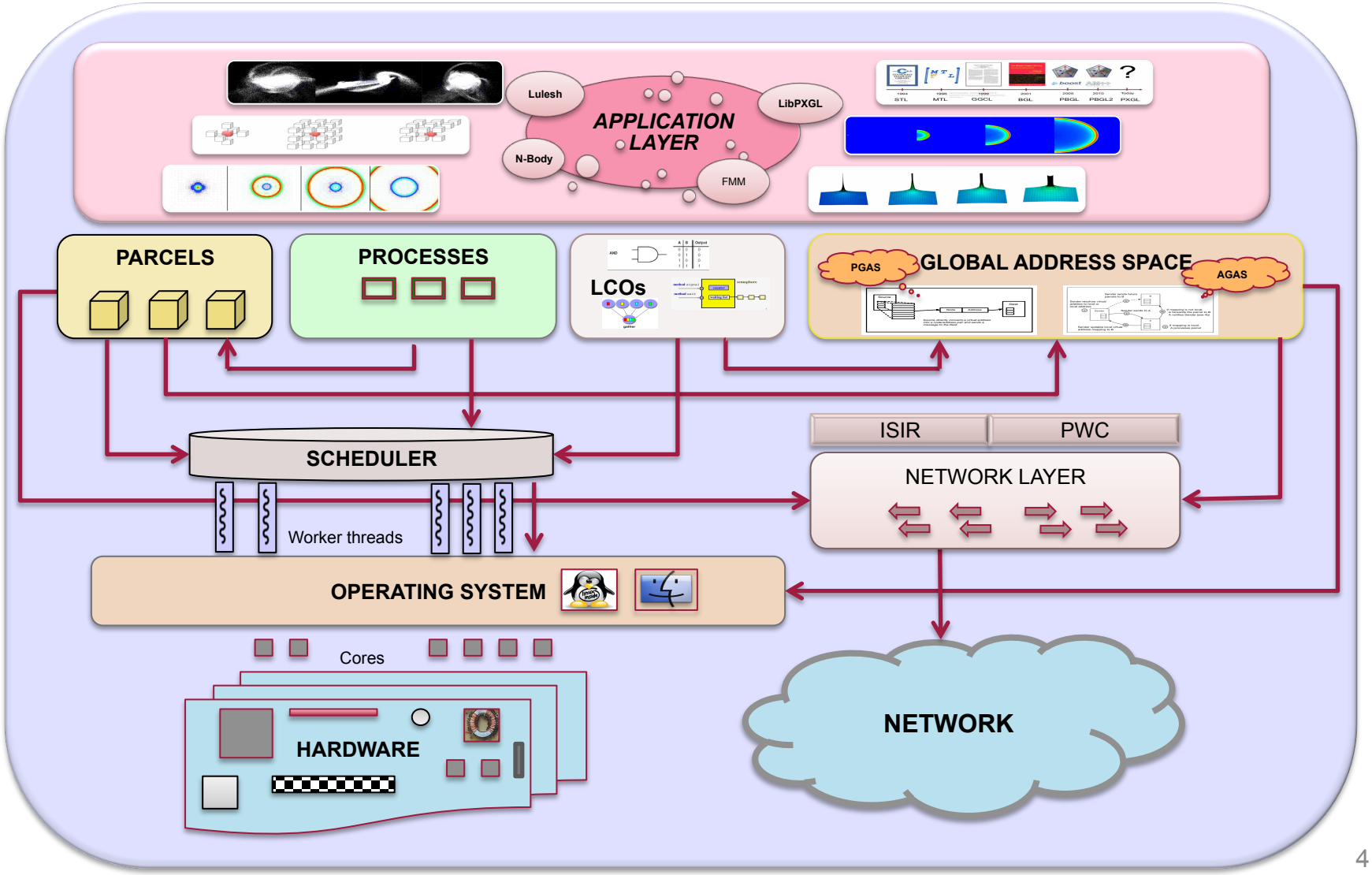
April 27, 2016

Introduction

- The “Runtime Hypothesis” is that the time to solution of a fixed workload can be reduced by adding more work – a paradox.
- Driven by promise of exploitation of compute time information to guide application execution to achieve efficiency and scalability improvement.
- An example: HPX-5 runtime based on the experimental ParalleX execution model to address SLOWER performance model parameters (e.g., overhead).
- What are the costs of runtime mechanisms? They impose bounds on parallelism granularity and in other ways get in the way.
- Presented here: quantitative

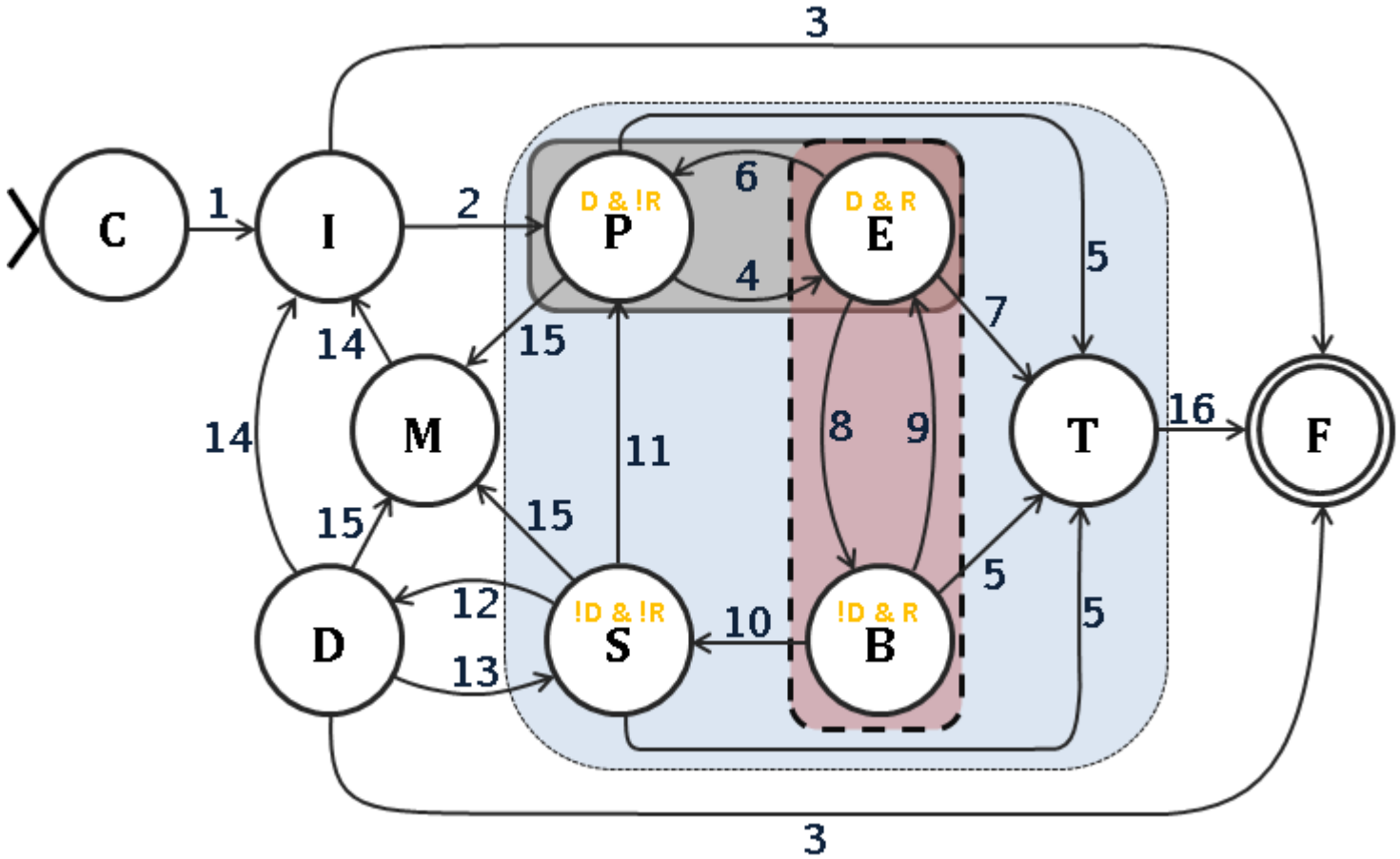

$v = \frac{1}{4} \times v_0$
$v = \frac{1}{2} \times v_0$
$v = v_0$
$v = 2 \times v_0$
$v = 4 \times v_0$
$\frac{S_{max}}{2}$ curve
Normalized scalability ($\frac{S_{max_0}}{2}$)
Normalized parallelism ($\frac{W}{v_0}$)
10^{1}
10^{0}
10^{-1}
10^{-1}
10^{0}
10^{1}
10^{2}

HPX-5 Runtime Software Architecture



Courtesy of Jayashree Candadai, IU

ParalleX Computation Complex



-- Runtime Aware -- Logically Active -- Physically Active

Time Required to Check if Memory Address is Local or Remote in HPX5

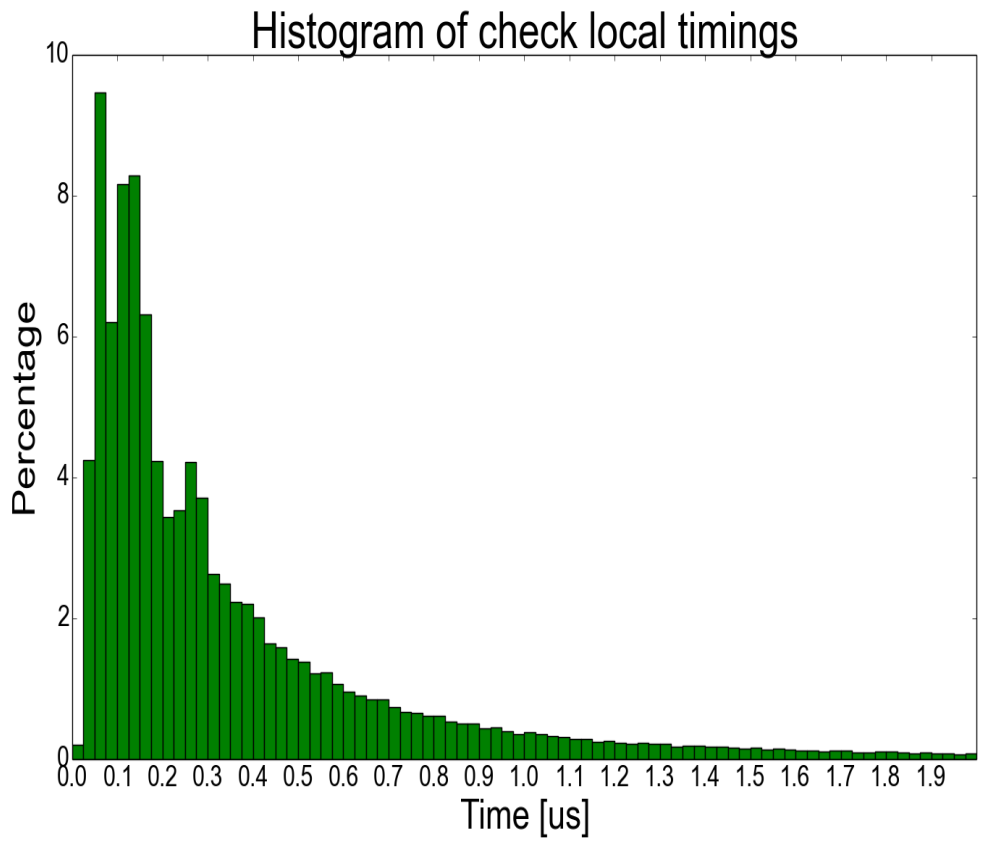
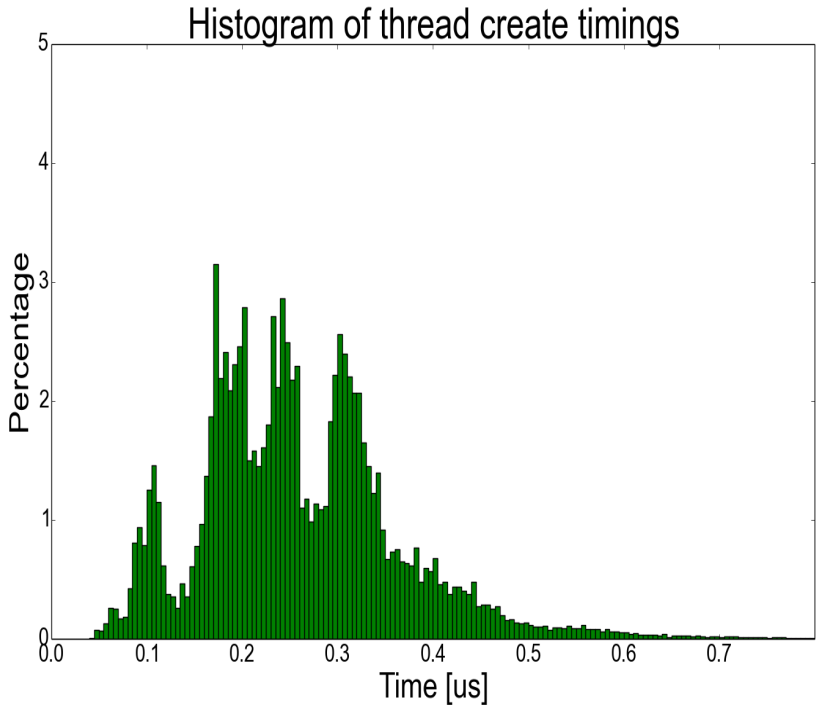


Chart courtesy of Daniel Kogler, IU

Time Required to Create a New Lightweight Thread in HPX5



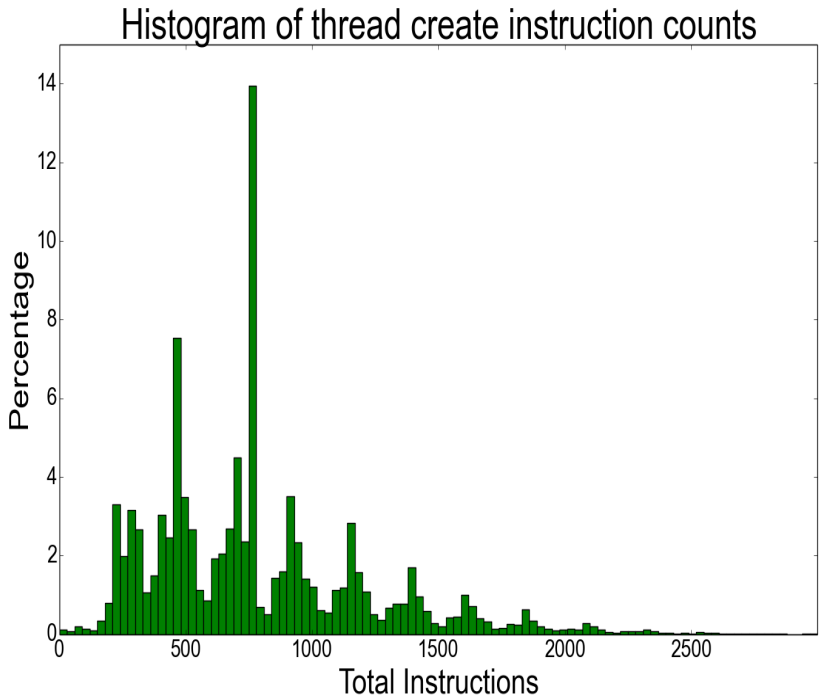


Chart courtesy of Daniel Kogler, IU

Time Required to Perform a Context Switch Between Lightweight Threads in HPX5

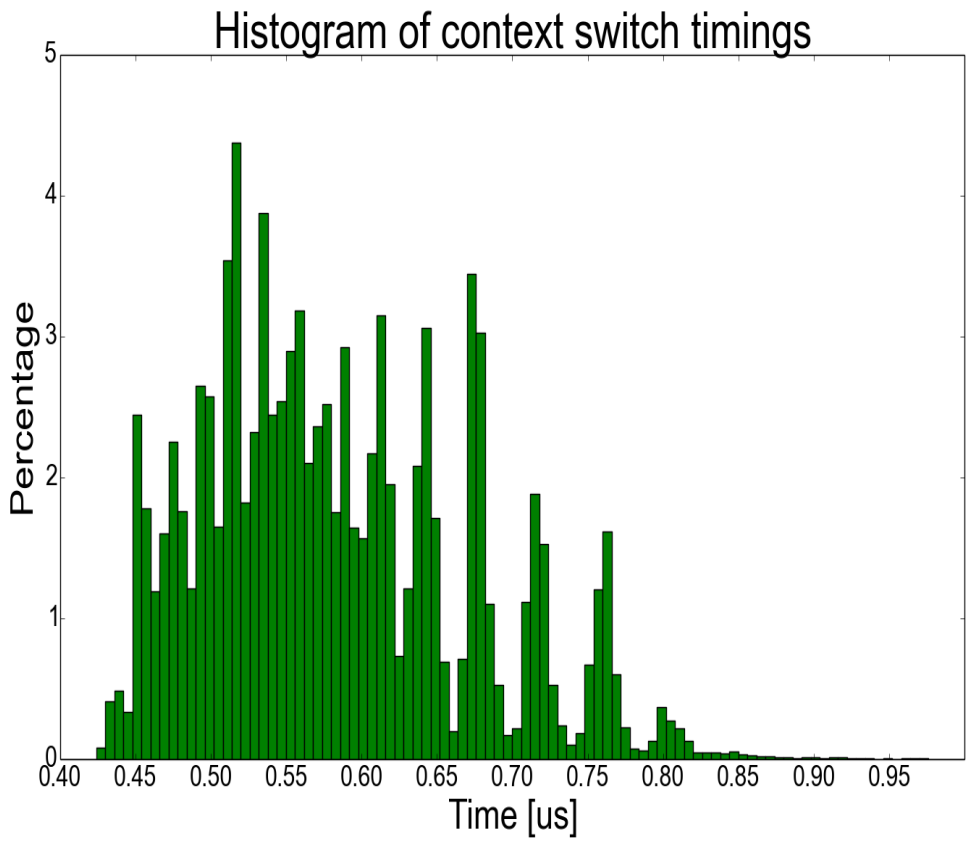
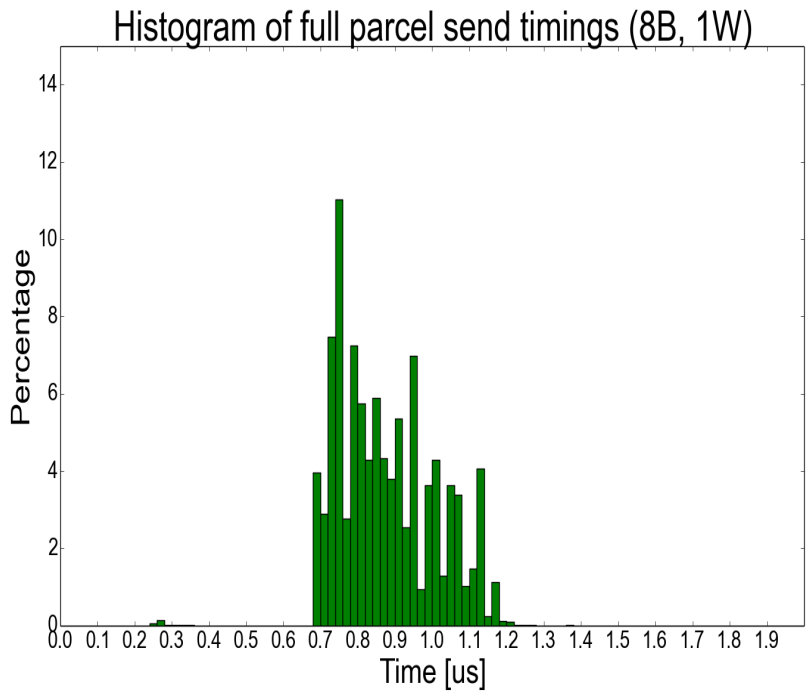


Chart courtesy of Daniel Kogler, IU

Time Required to Acquire, Prepare, and Put a Parcel into the Send Queue (8 Byte payload, no contention, jemalloc allocator)



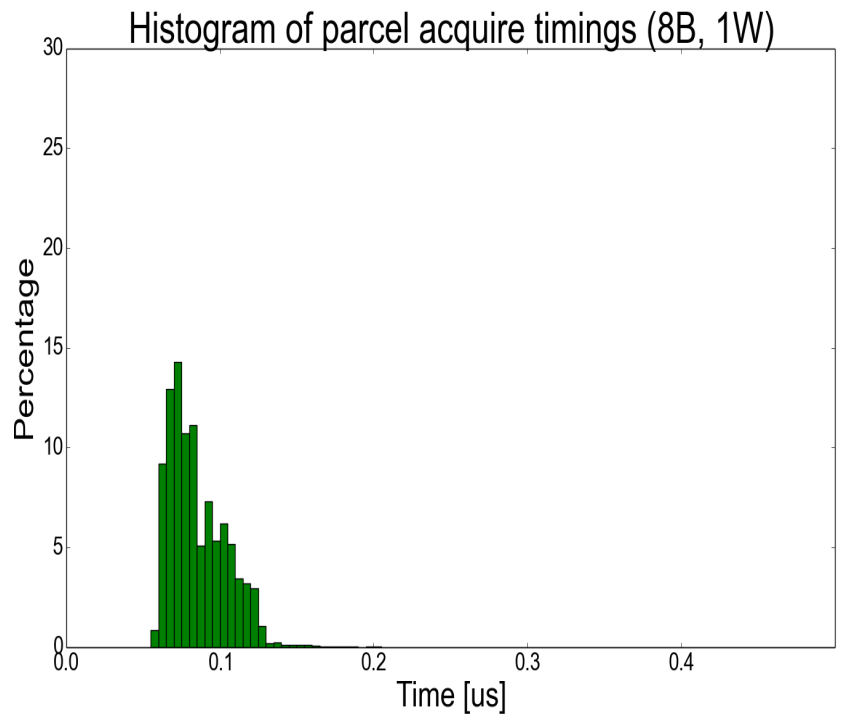


Chart courtesy of Daniel Kogler, IU

Concluding Observations

- All mechanisms shown implemented in software using conventional hardware support (e.g., RDMA)
- Key mechanisms achieved in less than 1 microsecond with some less than 100 nanoseconds.
- Measured costs distributed by as much as a factor of 2.
- Sensitivities observed to cache miss rates.
- Sensitivities less dependent on TLB misses.
- Actual impact is dependent on incident rate determined by real-world application workload.
- Lessons learned may suggest architecture advances for improved runtime overhead, efficiency, and scalability.