Scaling Insights to Exascale: An Integration of Simulation and Modeling

Daniel Ernst, PhD

Senior Principal Engineer

Cray Advanced Technology

dje@cray.com

@ernstdj

linkedin.com/in/danernst/

Performance at Scale is Cray's Business

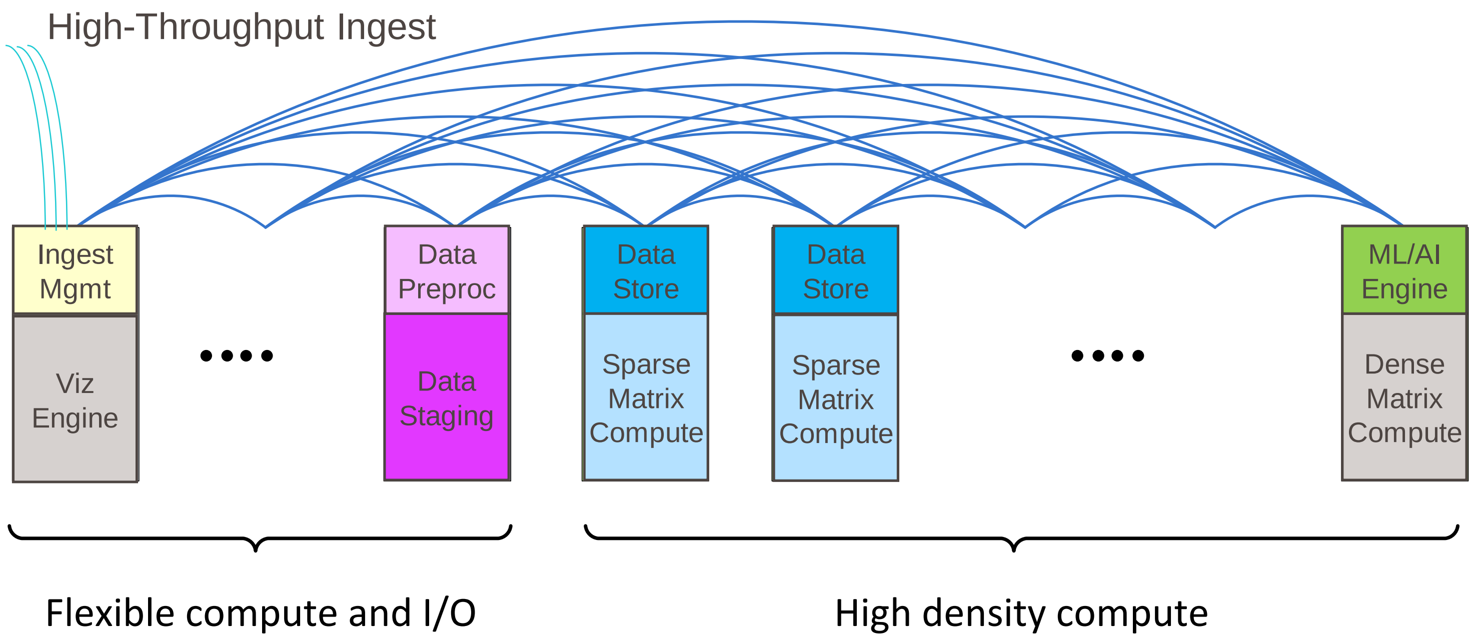
Why Model Exascale?

- Because system design has been in progress for over a year, so we need to:
  - Evaluate available compute and memory options
  - Understand application scalability
  - Balance complex behaviors against cost trade-offs
  - Expose requirements on software and applications

Enabling Exascale Science Workflows

- Rising use of diverse workflows for science
  - Increased use of AI/Analytics
- Compute and data hardware components are also diversifying
- Cray systems must enable both diversities to exist together

- Leverage full HW capabilities:
  - Increase utilization
  - Reduce data movement
  - Simplify workflows

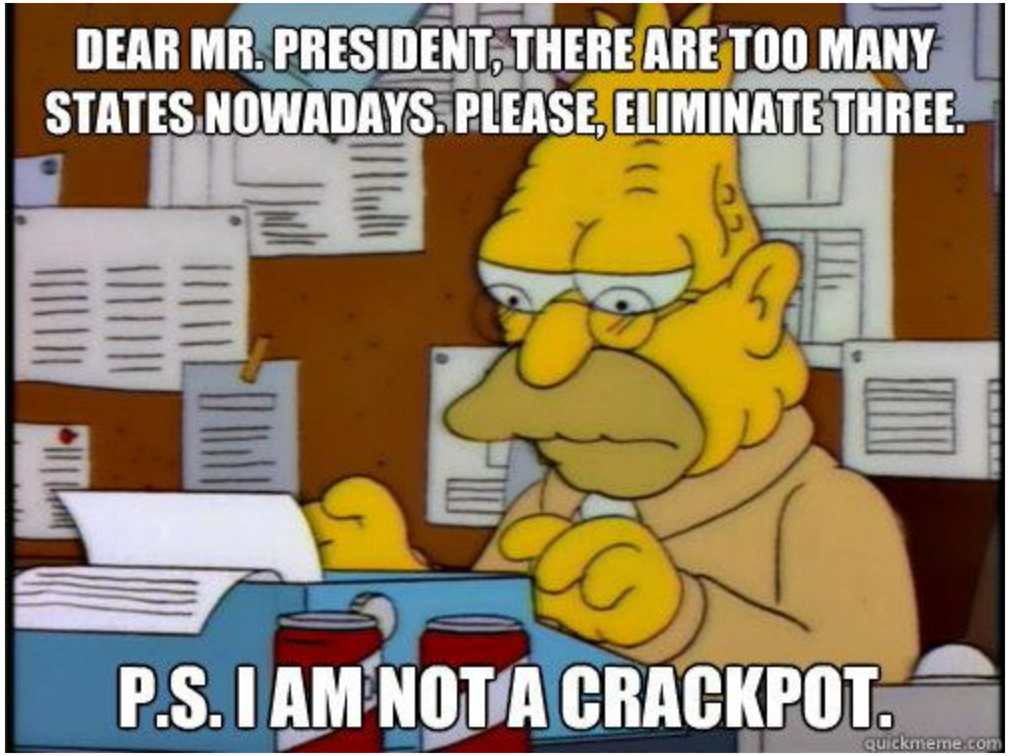


Problem Statement

- Systems of this scale have an *immense* state space

- Components:
  - Multiple heterogeneous compute elements
  - Memory / Storage
  - Networks
  - Interactions between them!

- Applications!



Understanding Applications

- Mini-apps are a nuanced communication device
  - They are NOT benchmarks
  - But you can use them to estimate performance
  - They express not just a point instance, but often an entire range of uses
  - This range of uses is bounded largely by scientific properties

- The number of people who can comprehend this end-to-end is *extremely small*.

- It took us 2+ years of modeling, refining, estimating, and just plain *handling* of FF2 mini-apps to understand a moderate range of how their use could map to hardware
  - Even then the pool is limited to the ones we had the most success with

Divide and Conquer

- Cracking this state space is infeasible with a unified simulation infrastructure

- Alternate approach:
  - Simulate/Model things that matter in each component level
  - Integrate context from other components where it's important

- Examples:
  - Model networks with traffic patterns based on application/node interactions
  - Model node hardware using application parameters mimicking at-scale usage
  - Model system app performance given sensitivities gleaned from node-scale

Tackling Node-Level Insights with Sage

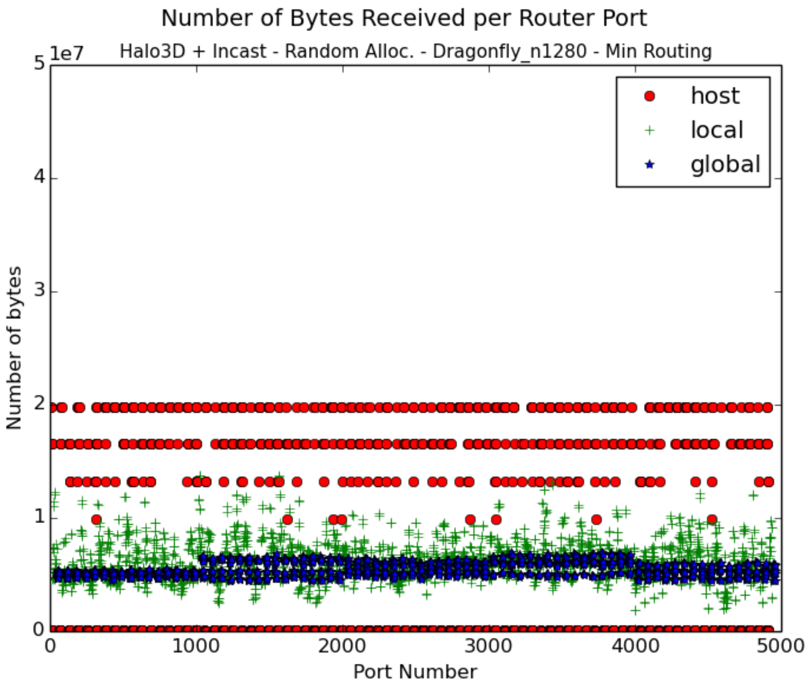
- Sage is an application characterization toolset that is built to expose the sensitivities of applications to different architectures
  - Focus is on modeling future “what if” architectural designs, including ISA extensions, memory systems, NICs, thread synchronization enhancements, etc.

- Sage models performance of *full* *node* hybrid parallel apps
  - (MPI + OpenMP + Vector)
  - At least 256 threads on a node (SoC)
  - Simulate and model real apps at full footprints at a rate that can complete in a reasonable time (hours, not days)
    - Characterize real at-scale app performance for given target architecture

- Validated using kernels with known performance attributes to verify accuracy against hardware or calculated performance

Network Insights for Scaling Workloads

- Cray adopted SST as our simulation platform part way through our DesignForward program, with a focus on studying mixed workloads:
  - The impact of one application on another
  - The impact of I/O traffic on applications
  - The impact of how jobs are distributed over nodes
- Infrastructure has been extended with near-cycle accurate models of Slingshot Rosetta switch as design progressed
- SST chosen because it is performant *at scale*
  - Systems of 65+ groups have been simulated



Application Expertise: Cray Performance Team

- Evaluating and predicting performance
  - Why are current codes/hardware performing at the level they are today?
  - How fast will a future application/hardware combination run?
  - Combining data with experience

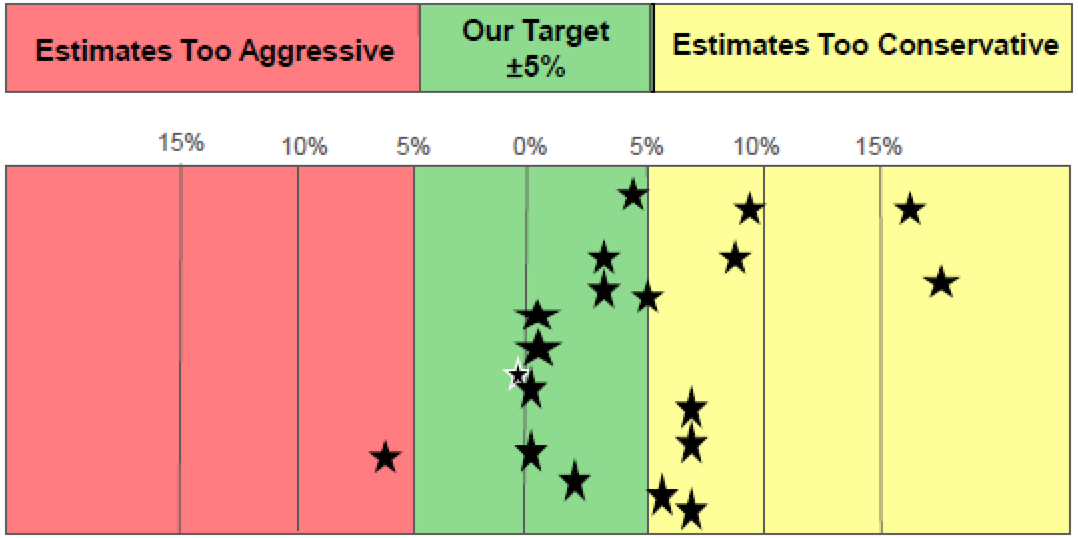
- Make codes run faster
  - Changes to the program, but also to software / hardware

- Calculate hundreds of performance estimates in a given year
  - Don’t get to choose the programs we evaluate
  - Need tools to keep them focused on where human analysis really counts

- Look for characteristics that will be the primary driving factors in future performance

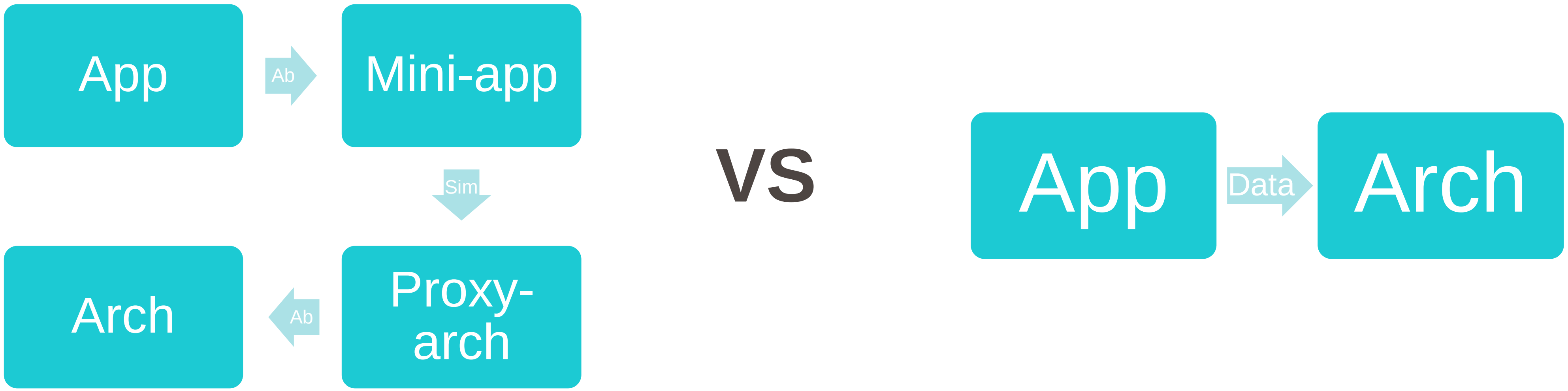
Projecting System-Level Performance

- Multi-level modeling methodology
- Component-wise calculation
  - Understanding how each component impacts each part of an application
- Analytical model combining these into bounded-error projections
- Being wrong costs us money
  - Cliff to the left
  - Slope to the right



Estimation Accuracy:
Estimates Compared to Initial Results on Delivered Hardware

Experimental Science Has a Place



- Critical design elements can often use less nuanced inputs
- If you don't have to provide depth and nuance, data is better than proxy

- Sometimes it pays to cut out the "middle men"

Memory – How Much Do We Need?

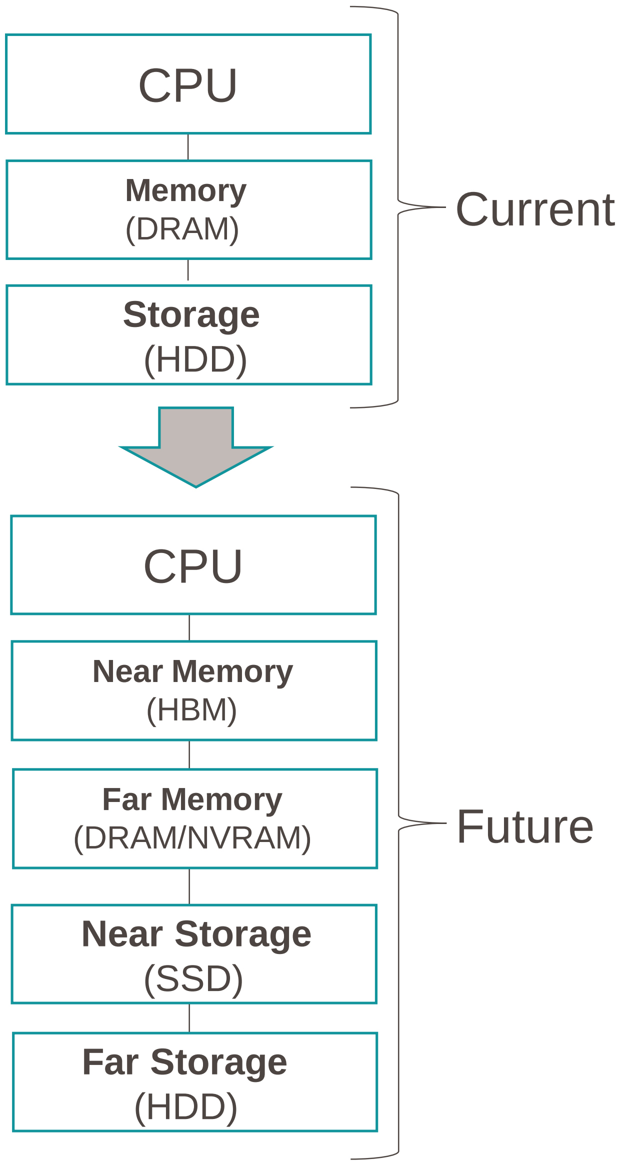
GiB/core	%
[0-0.5)	65.4%
[0.5-1.0)	86.8%
[1.0-2.0)	96.2%
[2.0-4.0)	99.9%
[4.0-8.0]	100.0%

A. Turner, and S. McIntosh-Smith. "A survey of application memory usage on a national supercomputer: an analysis of memory requirements on ARCHER." In PMBS, IEEE/ACM SuperComputing 2017.

What Insights Have We Gained?

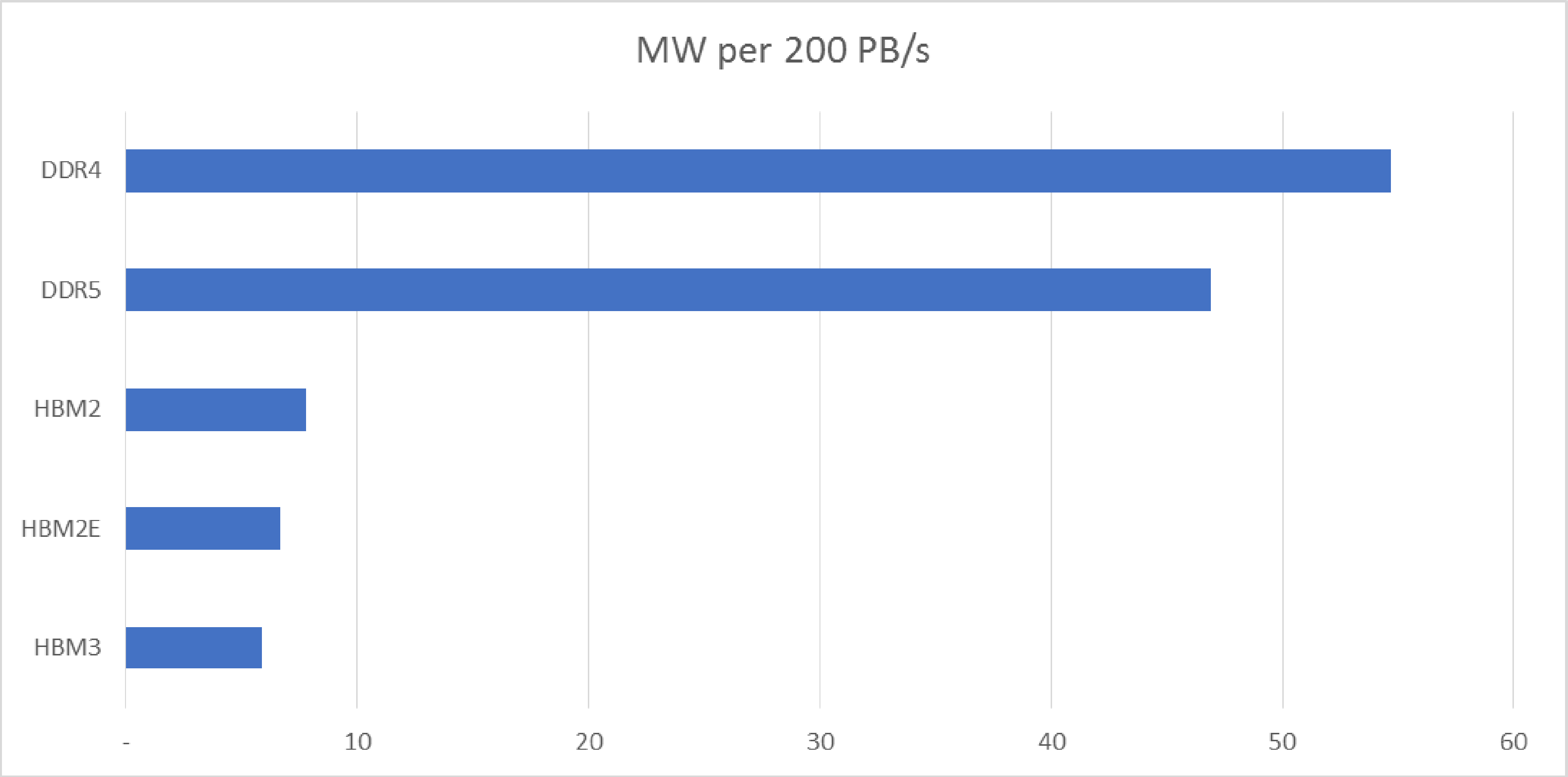
System Memory: Tiering Feasibility

- Just like compute, memory systems are becoming more heterogeneous
- What does the system design space look like for different memory solutions?
  - How should they be combined?
  - How should they be managed?
  - How would applications use them?
- There is a big difference between “can we do it?” and “should we do it?”



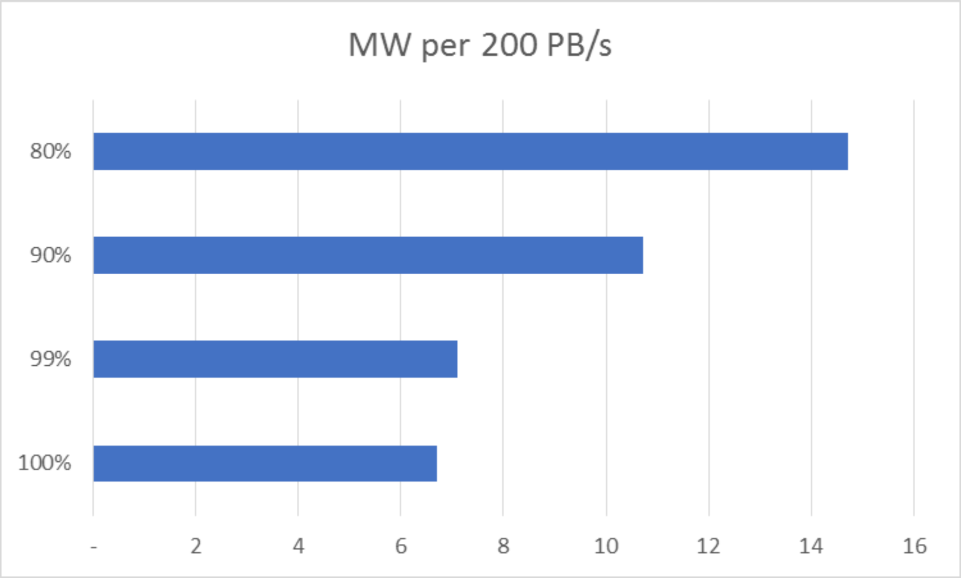
System Design Space

- Power is an important design characteristic

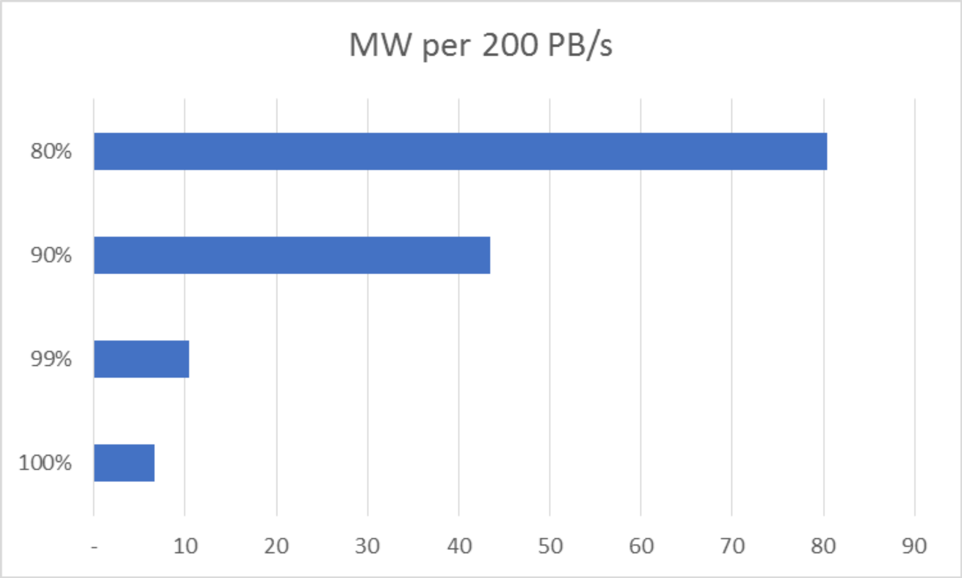


Note: Media power *only*

Tiers: Bandwidth Allocation Boundaries



HBM with DDR5



HBM with DCPMM

- Y-axis is % of target bandwidth provided by HBM
- Note: Media power *only*

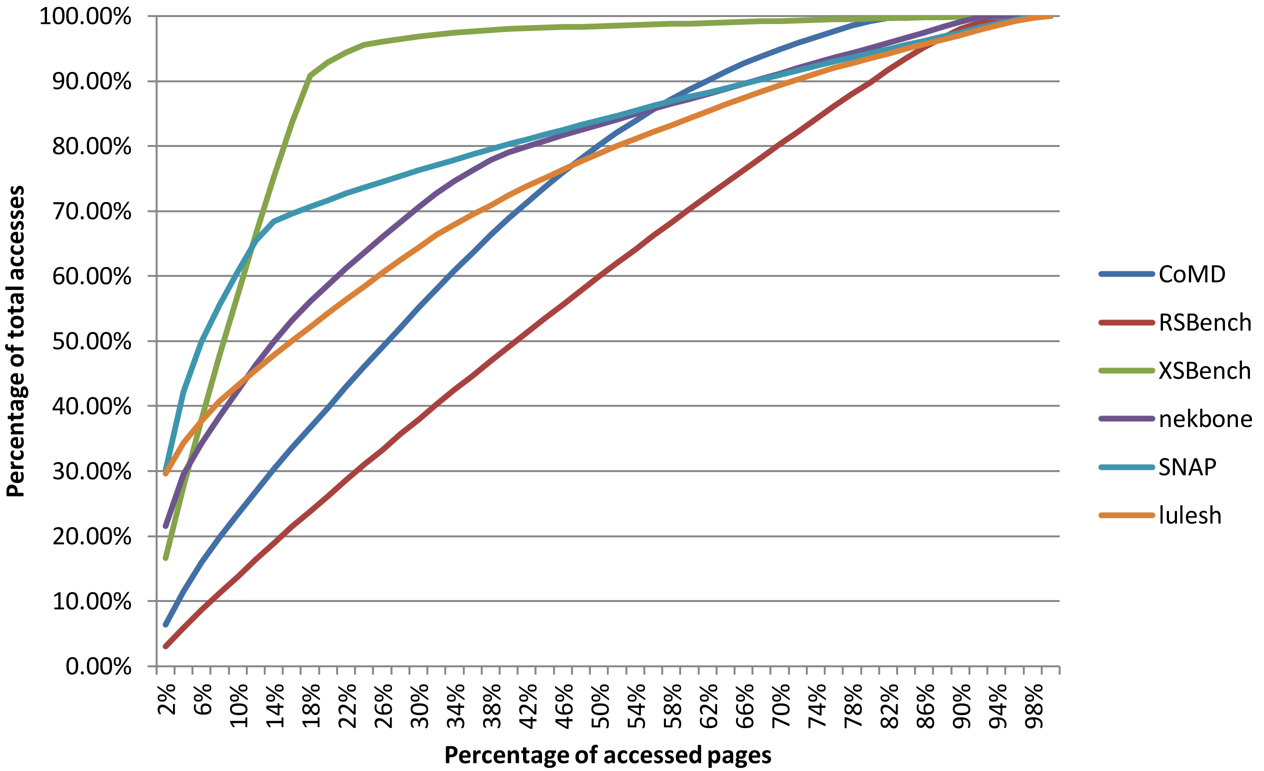
Applications Guidance For Future Systems

- HBM is a required technology for both performance and power efficiency

- HBM:DDR:PCM bandwidths likely to have **100:10:1** ratio *at best*
  - Likely better in the short term, but configurations will eventually be power (and cost) constrained

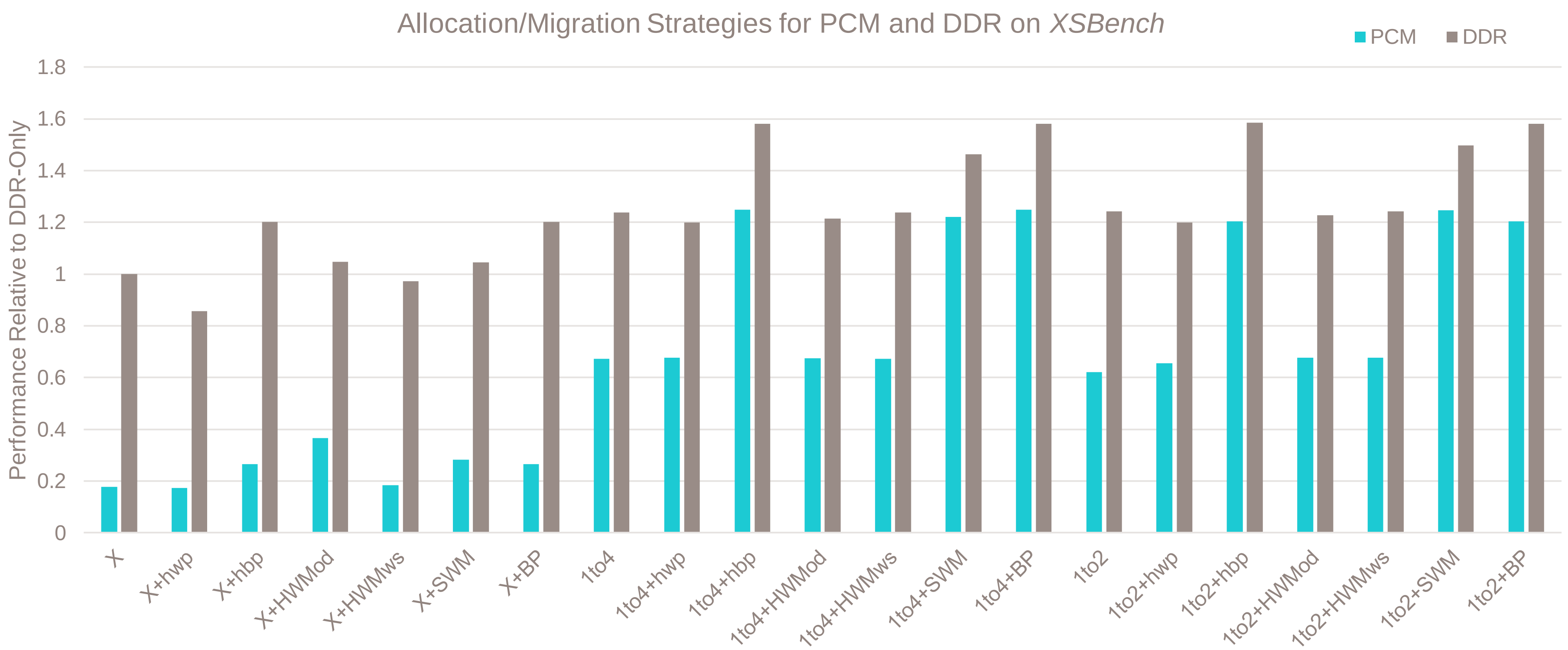
- However, tiering configurations are likely to vary by customer
  - Use cases / workflows will have big impact on these decisions
  - These guidelines provide *an* answer, but not *the* answer

Mini-Apps Tiering Readiness (from FF2)



	Footprint per Rank	Top 2% pages	> 70% accesses	> 80% accesses	> 90% accesses
CoMD	85.0 Mb	6.4% accesses	42% pages	50% pages	62% pages
RSBench	29.9 Mb	3.0% accesses	60% pages	70% pages	82% pages
XSBench	1021.9 Mb	16.6% accesses	14% pages	16% pages	18% pages
nekbone	232.7 Mb	21.5% accesses	30% pages	44% pages	68% pages
SNAP	1609.2 Mb	30.1% accesses	18% pages	40% pages	68% pages
lulesh	4171.8 Mb	29.6% accesses	38% pages	54% pages	72% pages

Putting it All Together



- This analysis is largely ***impossible*** without Sage, as it depends on combo of:
  - Cores (ORB, prefetch)
  - NoC (congestion, runtime) and Memory arch (mem parallelism)
  - Software layout

Summary - Tiering

- Tiering within applications: yuck
  - Bandwidth ratios quickly approaching infeasibility
  - Heavy impacts on users, tools, and management (Lang’s Law)

- Tiering within workflows: yes
  - Data pre-staging, Distributed Checkpoint/Restart, Data exchange for Multiphysics
    - All use cases that have typically fallen to storage
  - Also has impacts on users, tools, management
    - Currently seem far less invasive

How Does Your HPC Network Behave at Scale?

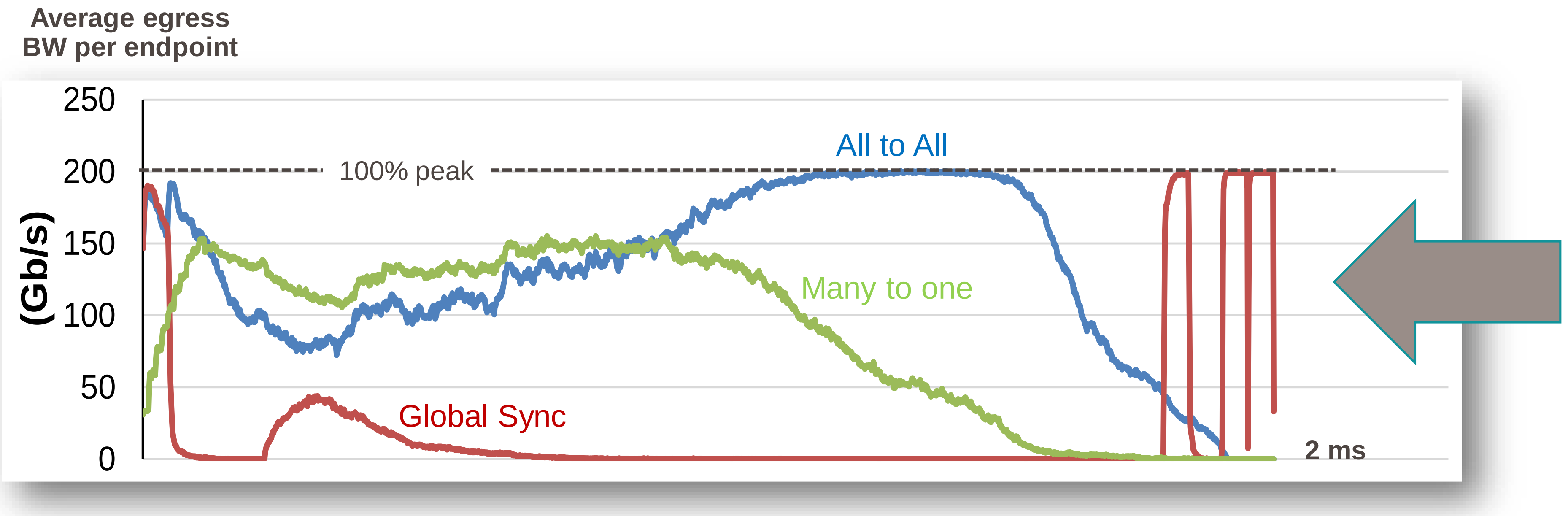
Your commute at 4:00 AM

MPI Ping-Pong Latency

Your commute at rush hour

Halo Exchange Under Load

Performance Under Load



Job Interference in today's networks

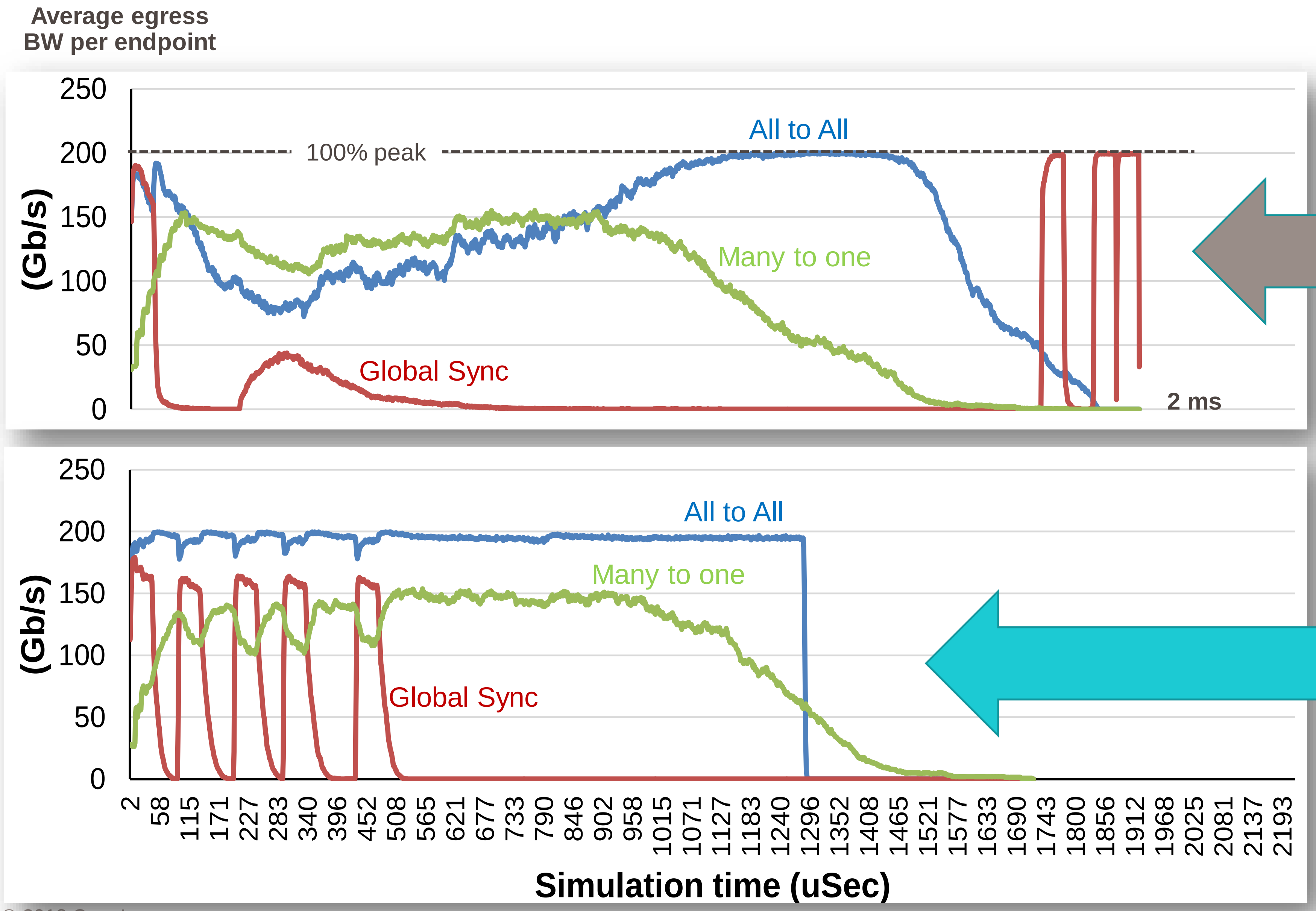
Congesting (green) traffic hurts well-behaved (blue) traffic, and *really* hurts latency-sensitive, synchronized (red) traffic.

Slingshot Congestion Management

CONGESTION MANAGEMENT

- Hardware automatically tracks *all* outstanding packets
  - Knows what is flowing between every pair of endpoints
- Quickly identifies and controls causes of congestion
  - Pushes back on sources… *just enough*
  - Frees up buffer space for everyone else
  - Other traffic not affected
  - Avoids HOL blocking *across entire fabric*
  - Fundamentally different than traditional ECN-based congestion control
- Fast and stable across wide variety of traffic patterns
  - Suitable for *dynamic HPC traffic*
- Performance isolation between apps on same QoS class
  - Applications much less vulnerable to other traffic on the network
  - Predictable runtimes
  - Lower mean *and tail* latency – a big benefit in apps with global synchronization

Congestion Management Provides Performance Isolation

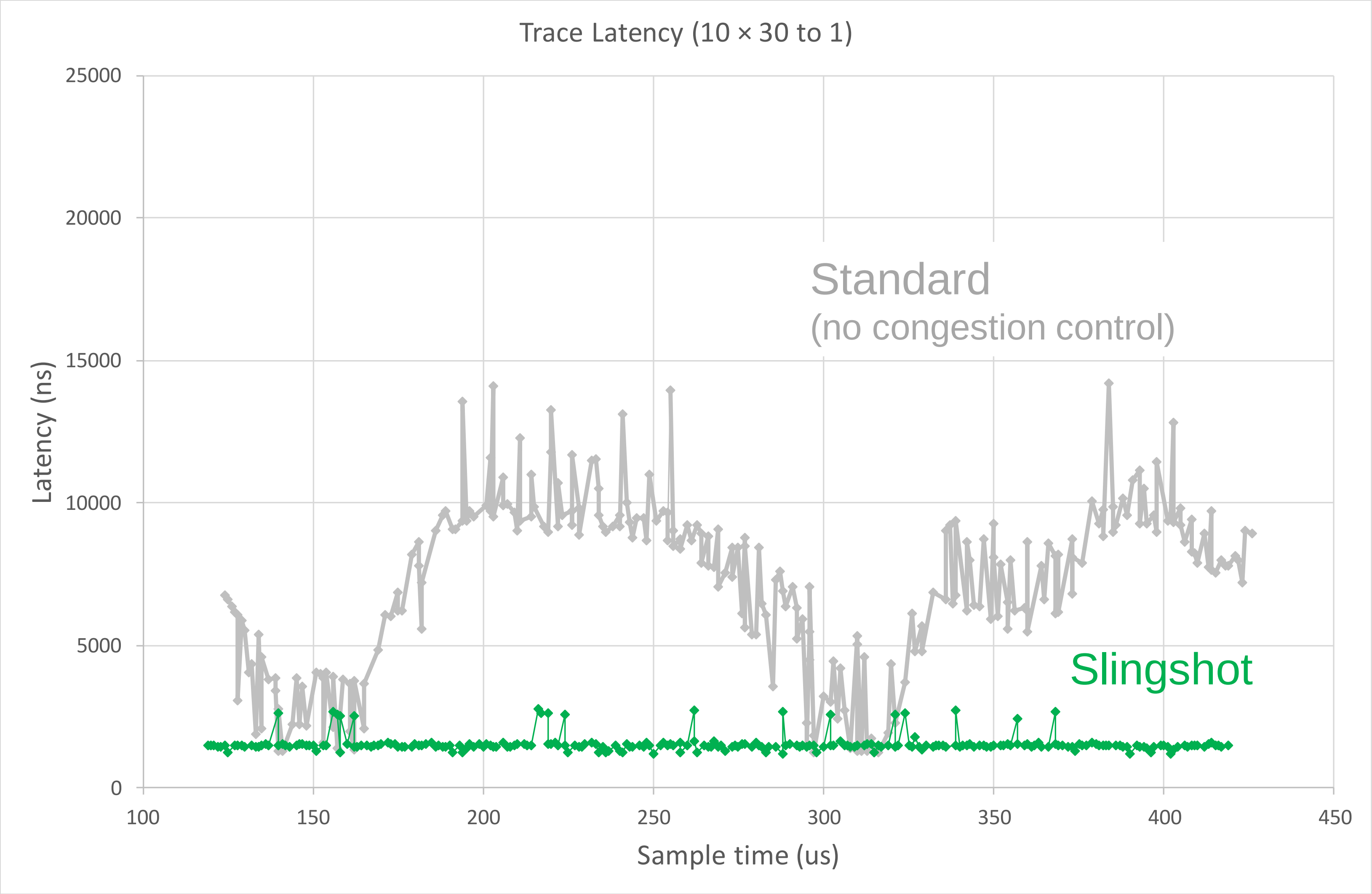


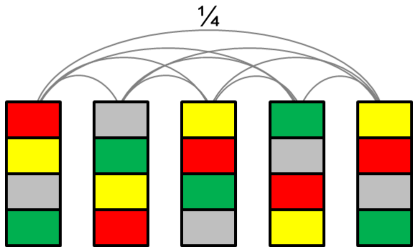
Job Interference in today's networks

Congesting (green) traffic hurts well-behaved (blue) traffic, and *really* hurts latency-sensitive, synchronized (red) traffic.

With Slingshot Advanced Congestion Management

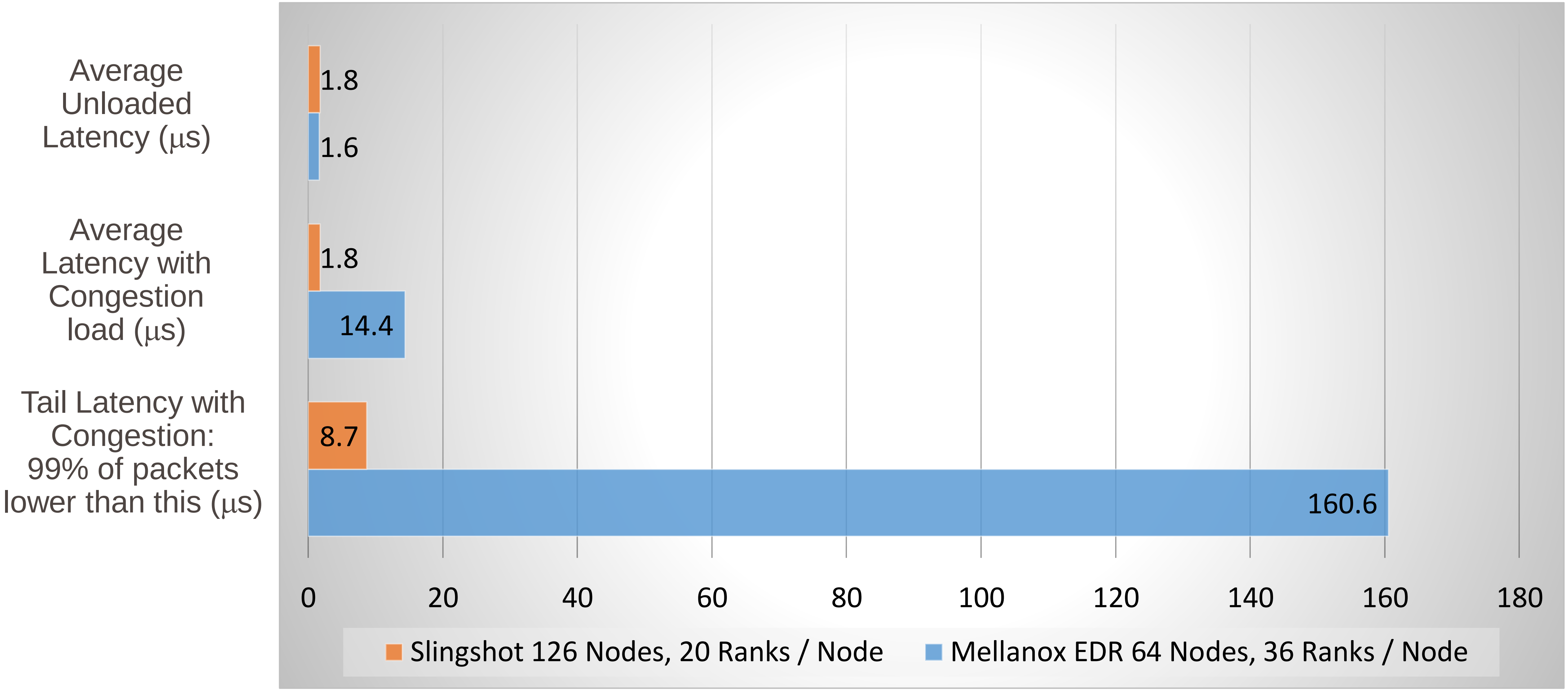
Low Packet Latency with Tight Distributions





Mix of background applications running, some of which are causing congestion.

GPCNeT: Random Ring Latency Congestion Test



Combinational Modeling at System Scale

- Ingredients:
  - Architectural parameters deemed relevant to applications
  - Network parameters gleaned from at-scale simulation of relevant patterns
  - Application sensitivities to a range of parameters for given input sets
- This last part is – for now – mostly a human element
  - Knowing what is and isn't relevant to a given application/architecture combination isn't easily automated

Code	Baseline	1/2 BW	1.5x Cores	1.5x Cores 1/2 BW
AMG	159.6	84.3	159.6	84.3
BDAS	664.9	642.2	977.5	931.4
HACC	50.0	50.0	75.0	75.0
Kripke	288.6	148.3	288.6	148.3
LAMMPS	268.5	257.3	268.5	257.3
MLDL	352.9	259.6	445.5	306.5
Nekbone	83.6	42.1	83.6	42.1
PENNANT	68.9	53.7	68.9	53.7
QMCPACK	26.9	21.2	33.9	25.2
Quicksilver	16.9	13.3	21.7	16.1
GEOMEAN	114.3	83.8	132.8	95.5

Summary

- Cray's performance modeling expertise is *in production* targeting Exascale
  - Ongoing use in refining estimates and architectures
  - Continues to contribute to development of Slingshot network
- Long-term investment in modeling node and network technologies had to be combined with developing application expertise in order to be successful
  - *aka* this requires deep partnership with customers, which Cray (and DOE!) has invested in heavily

THANK YOU

QUESTIONS?

dje@cray.com

@ernstdj

linkedin.com/in/danernst/