BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Topology Matters

Jeremy Enos jenos@illinois.edu

System Management & Development Lead

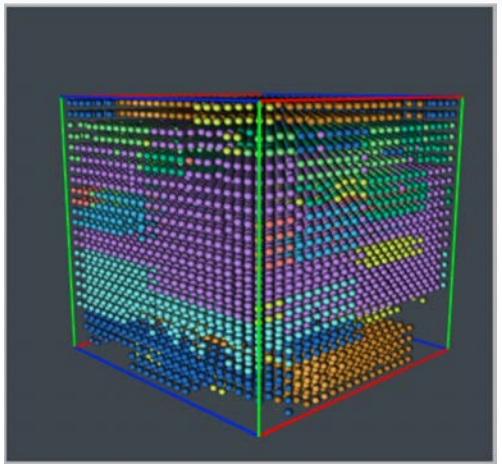

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

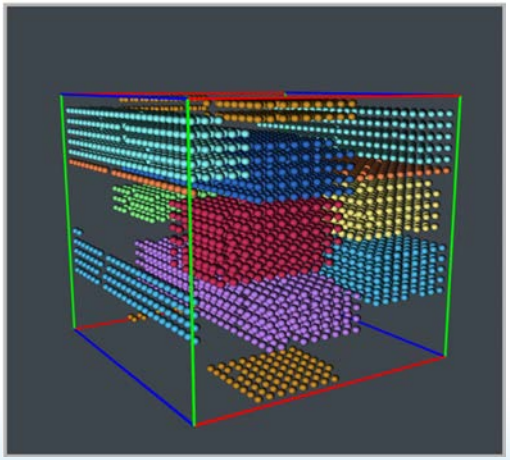
TAS Enhancements Due 2016

- Serial job optimizations
  - Processing
  - Transparency
- Maintenance optimizations
- Internal and External Fragmentation Exploitation
- TAS bypass option
- Max job count, utilization, and iteration time improvement deliverables
- Just 1 bug related to TAS code after initial ramp-up period!

TAS: A Brief Review

- Medium to large, communication intensive applications competed for network resource due to torus placement, resulting in longer and unpredictable runtimes
- Scheduler modified to place jobs into route-friendly prisms to reduce inter-job contention

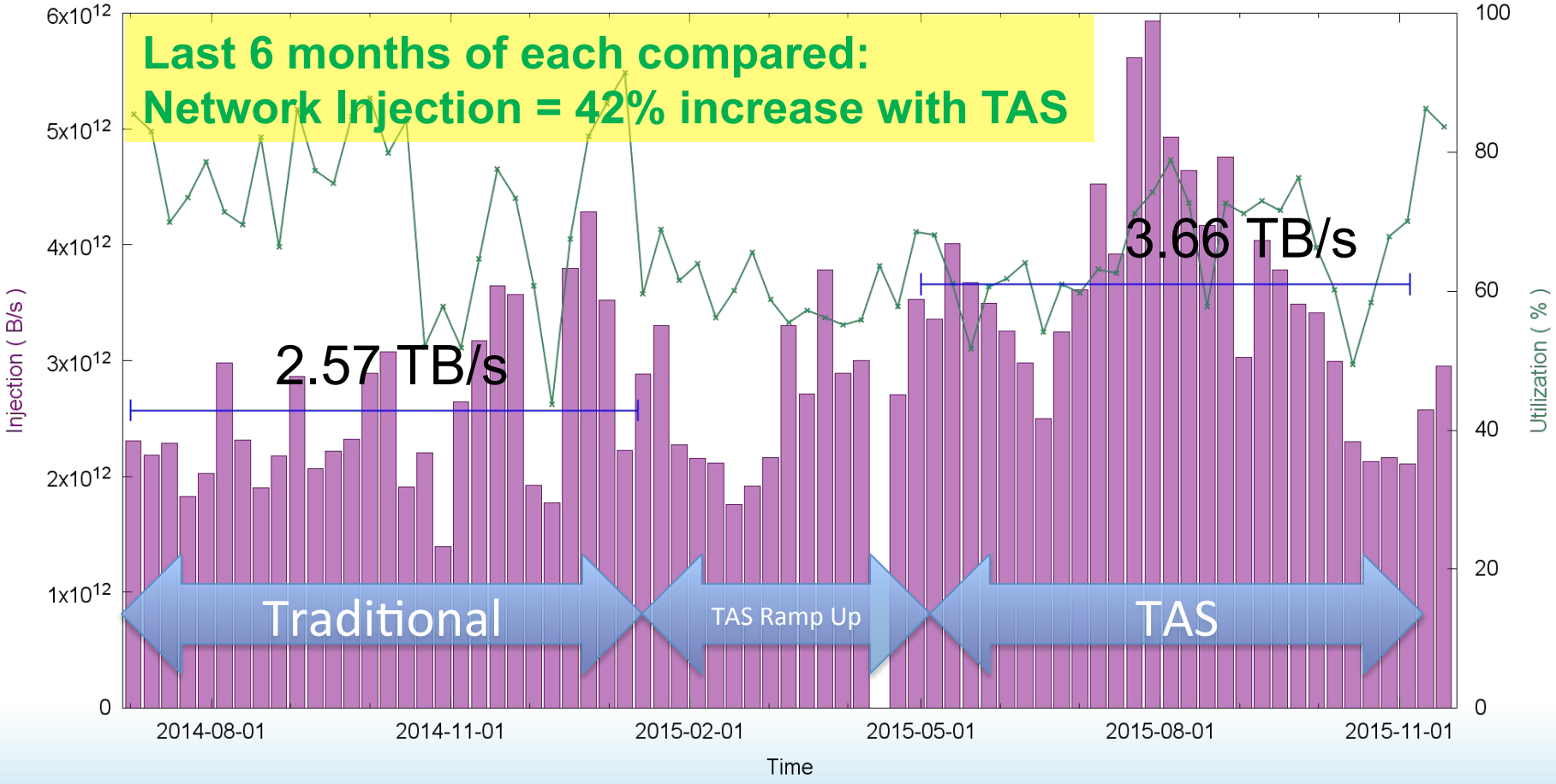




TAS Effects

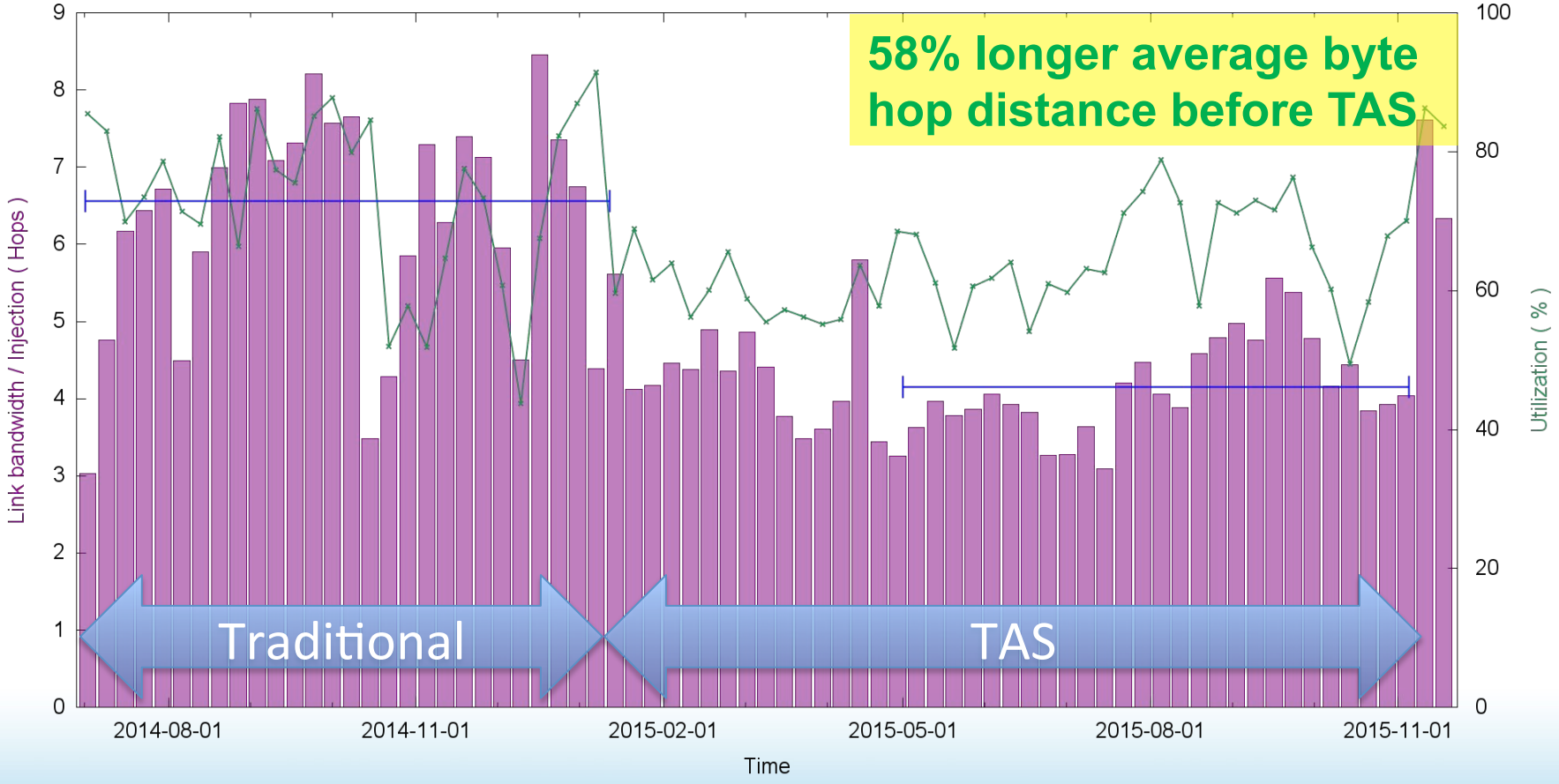
- Faster and more consistent runtimes for large communication intensive jobs
- Faster turnaround time for small jobs
- Longer turnaround time for medium jobs (1k nodes) – we're working on this
- Gemini failure (rare) impact isolated to fewer jobs
- Reduced system utilization
- More work done overall

System-wide aggregate network injection rate average by week



Improved average byte*hop count

Represents network burden per byte transmitted (lower is better)

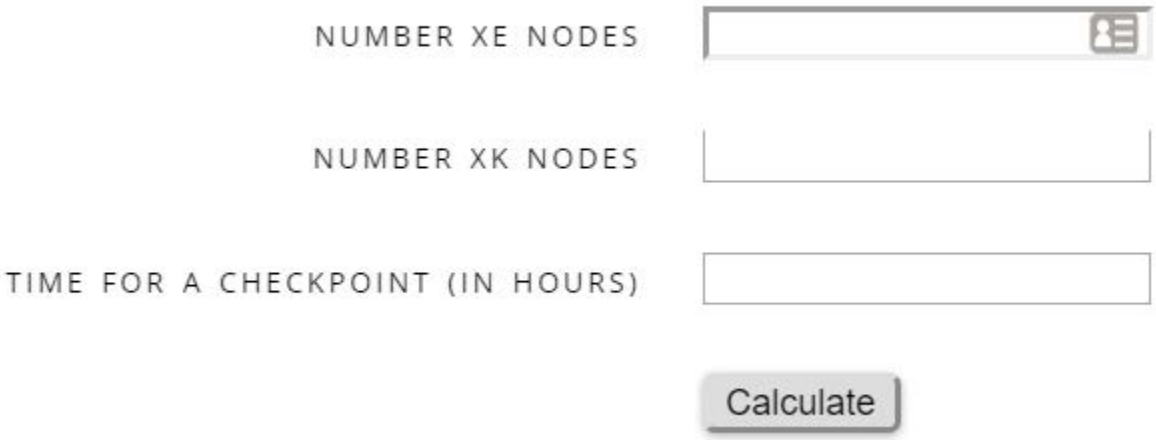


Get the most out of the scheduler

- Use backfill opportunities (showbf)
  - >90% jobs backfill, ~48% by node hour
- Use flexible walltime specification; checkpoint
- Don't hyper-exaggerate requested walltime
- Charge factor discount incentives for the above
- Job to job interference improbable – additional options can guarantee it if needed (except for I/O traffic)

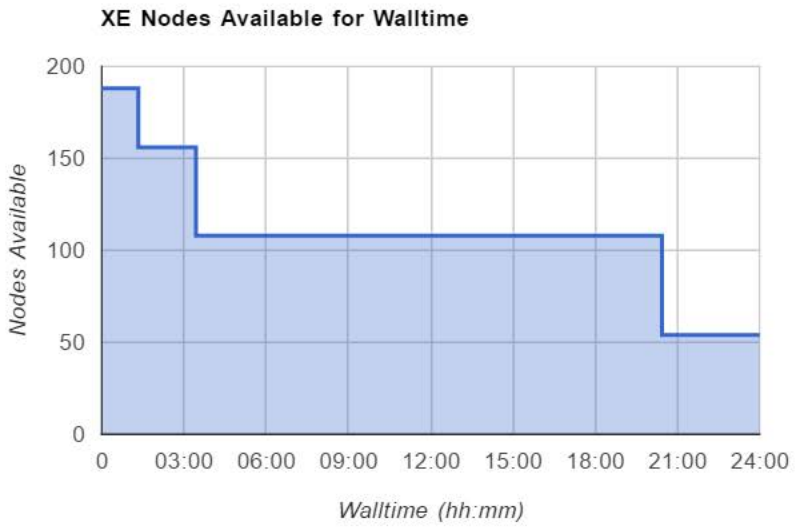
Portal: Optimal checkpoint interval

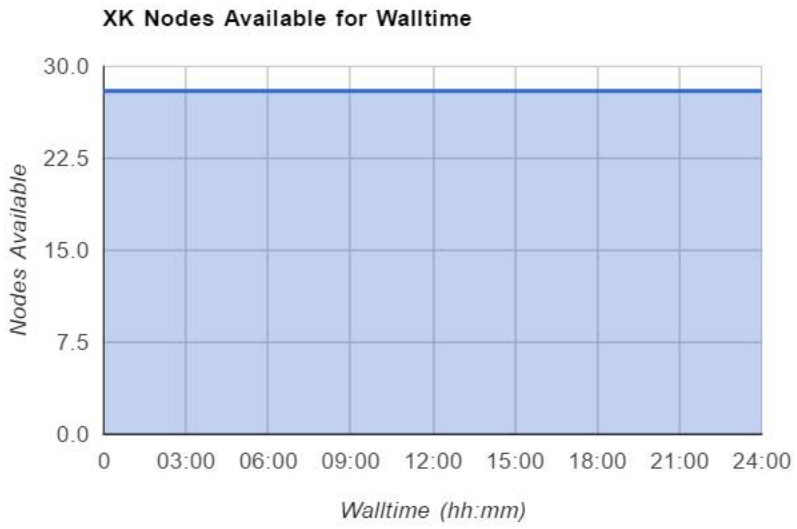
https://bluewaters.ncsa.illinois.edu/storage#checkpoint



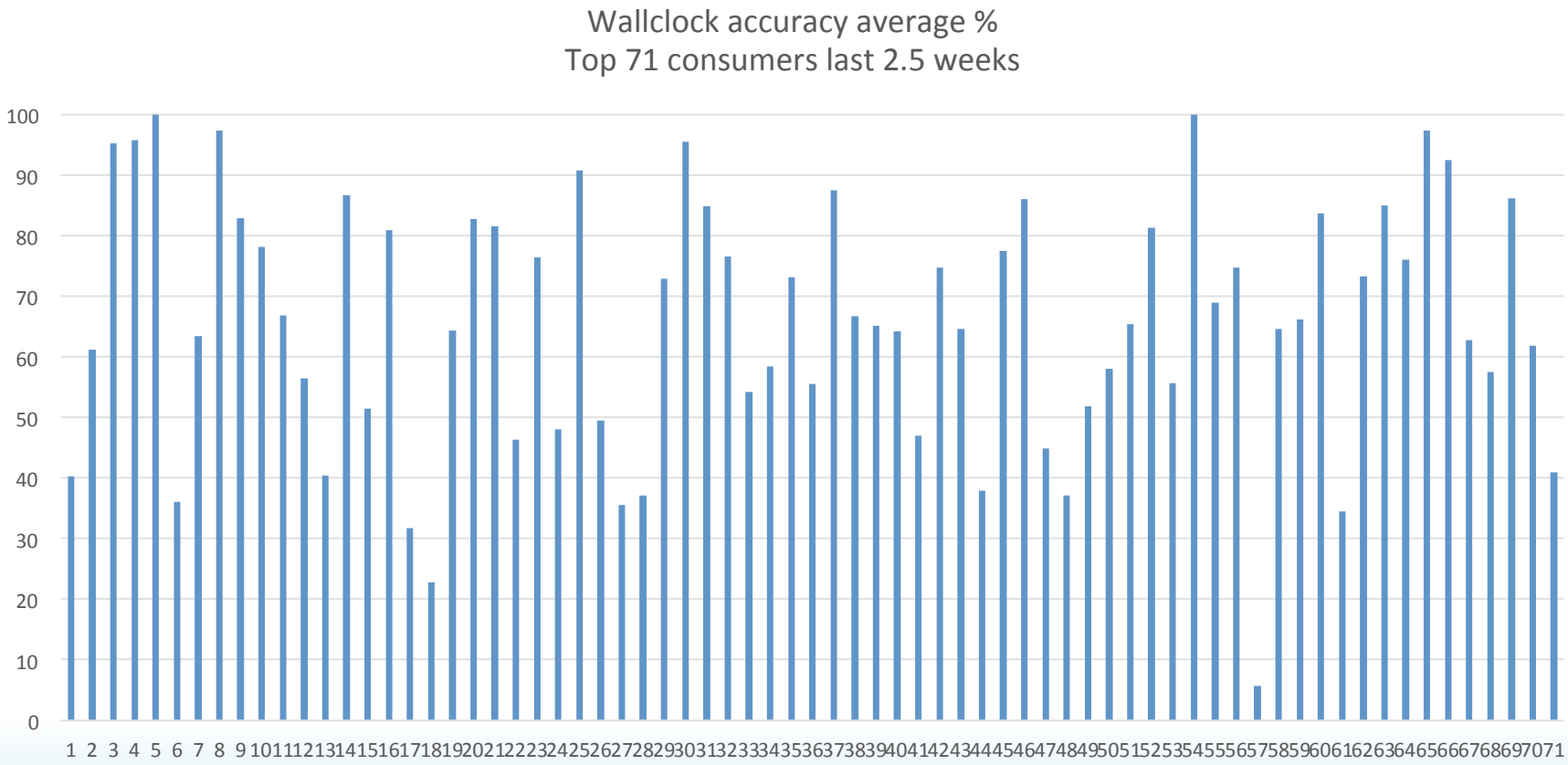
Portal: Backfill representation

https://bluewaters.ncsa.illinois.edu/machine-status





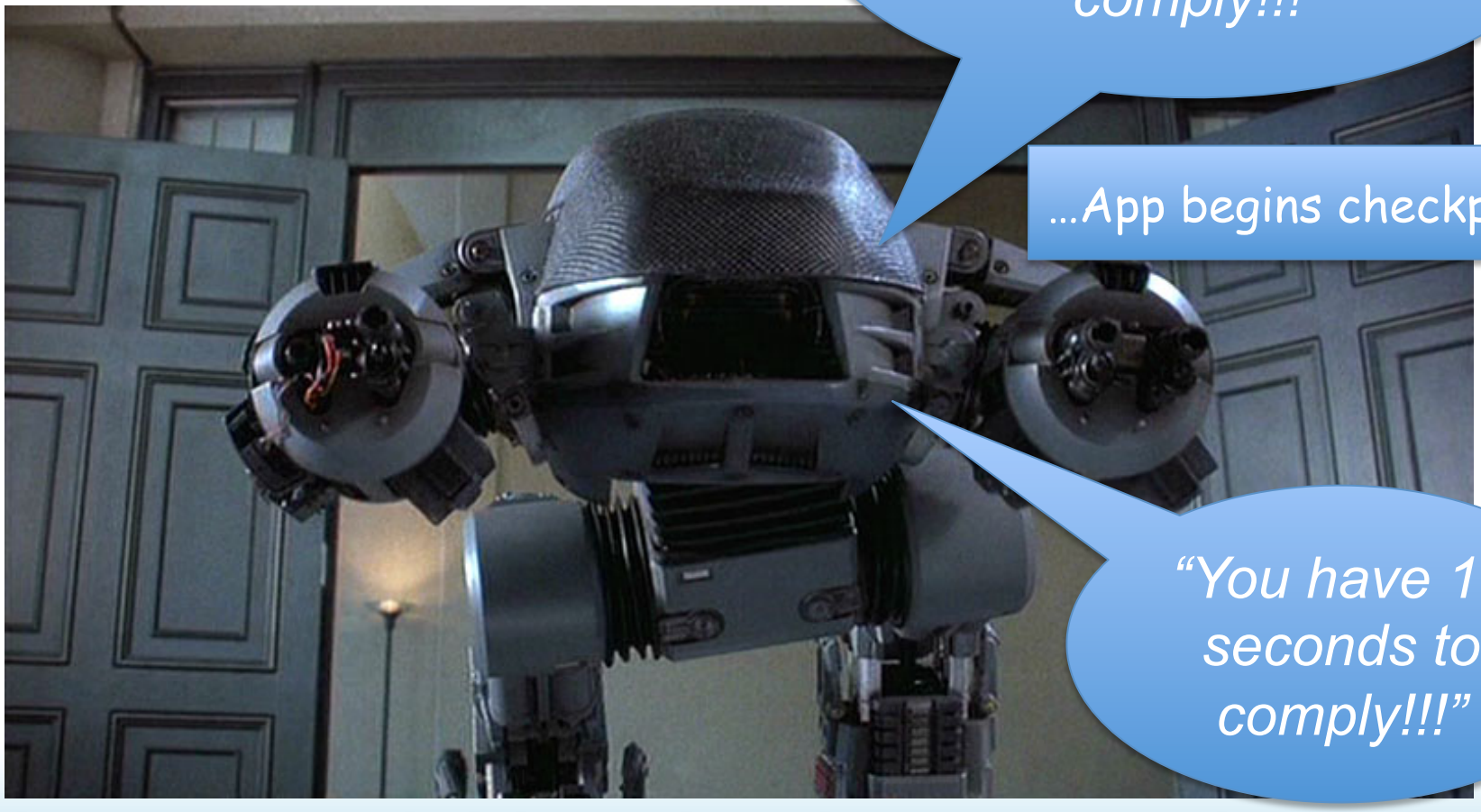
Wall clock accuracy



Top priority issues

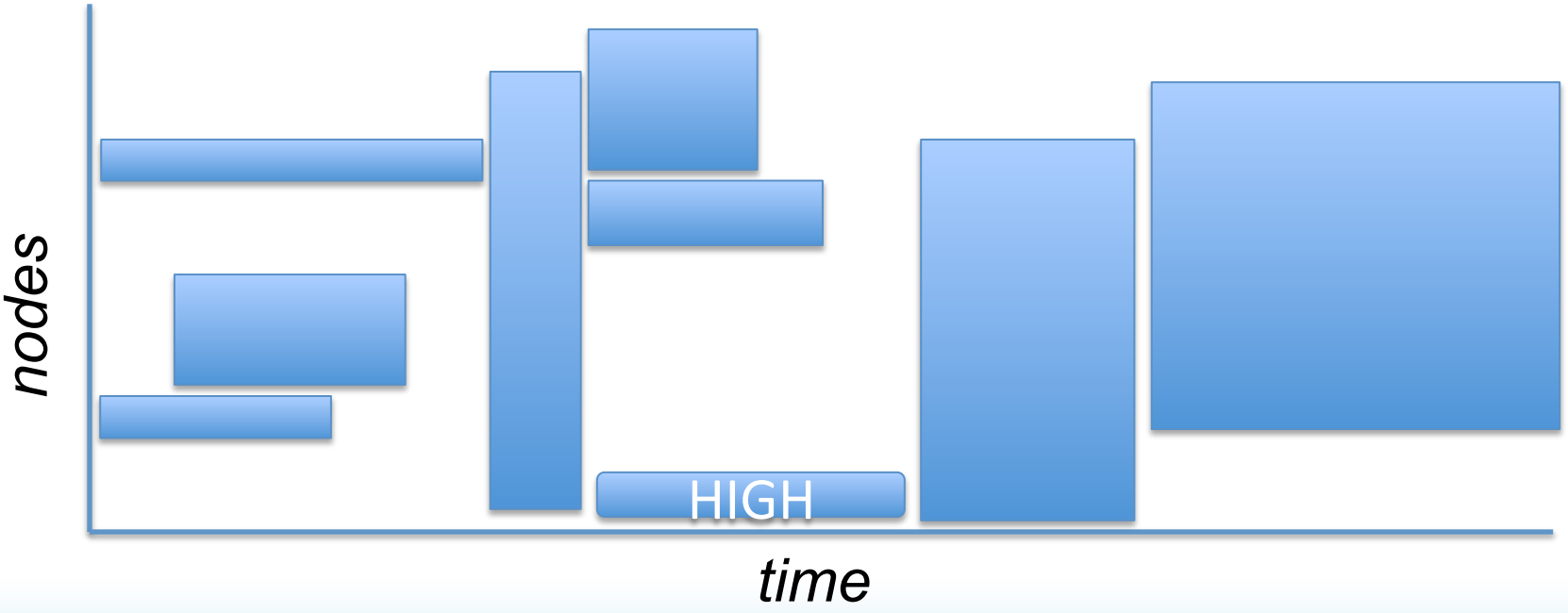
- Job dependency reliability (fixed)
- Iteration time
  - Interactive job response
  - Reservation depth (to reach medium size jobs)
- Efficient job dependency trains
- Graceful preemption
  - Checkpoint warning signals (kill signal not delayed properly)

ED 209 Checkpoint Warning System

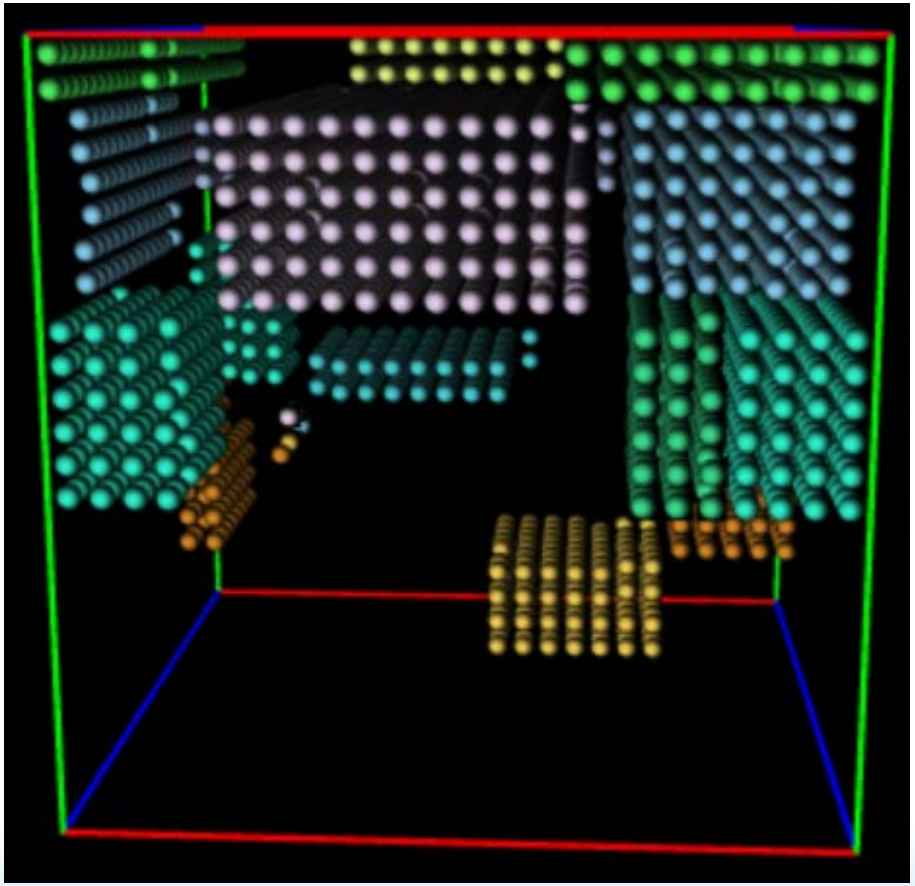


Challenging Job Mixes

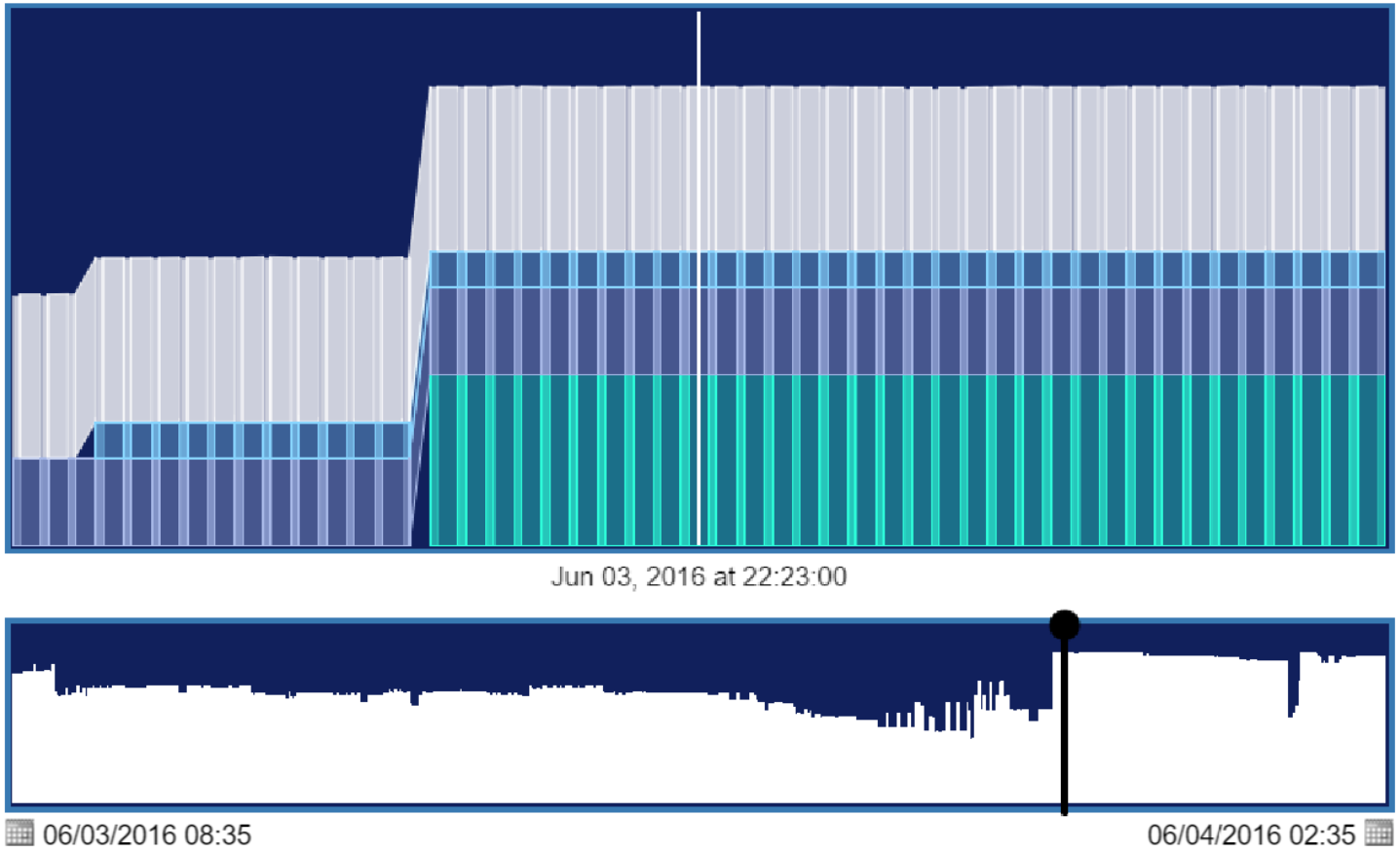
- Wide & Short (shouldn't see >1 in queue at once)
- Long & Narrow (fenced)



Location aspect increases challenge



Example drain cost



Serial Jobs and Bundling

- Serial workloads see no TAS benefit
- Small fill is good provided:
  - < 3-4k jobs. Algorithms have *node_count* x *job_count* functions
  - Task length is short, else fragmentation over time
- Bundled fill is good provided:
  - Task length is relatively long (to be worth drain penalty)
  - Task length is relatively consistent (intra-job efficiency)
- Enhancements are coming, recommendation will change
- Exceptions exist on case by case basis, ask for help

Serial Jobs and Bundling

- Task subscheduler solutions exist on Blue Waters
  - E.g. Swift
- If large job and flexible task count, try to maximize prism efficiency
  - Use “checkjob” to see “internal fragmentation %”
  - Blue Waters staff can help provide recommendations for job sizes

Questions

Outline todo

- Top bugs/issues
  - Iteration time/speed - resdepth
  - Checkpoint signal for preempted jobs
- Backfill plots on portal
- Commflags
- Wide and short multiple – interactive instead?
- Long and narrow
- Walltime overspecification
  - 48 hrs now
- Serial jobs
  - Bundling, unbundling
  - Prism fill, intfrag, checkjob
- Checkpointing signal warning
- Discounts
- Priority reservations – 93% jobs backfill, 48% node hours backfill

TAS: Acceptance

- SPP and other codes used for acceptance test recommendation
- showed a 3% net gain in throughput (performance gain vs utilization reduction) and a 13.4% runtime CV reduction
- December 2014 NSF Panel Recommendation:
  "… the implementation of TAS may impact expansion time for all jobs and system utilization. The panel recommends that the team re-evaluate these metrics after gaining experience with TAS in operations and adjust as necessary."

TAS Transition Plan (December 2014)

- Ramp up period as teams learn advanced options
- Tune knobs for placement conservatism, policy incentives
- Continued analysis of utilization and science throughput
- Evaluate alternative measurements of science throughput comparison

2015 Timeline

- Jan 15 TAS enabled
- Q1
  - Bug fixes (utilization impacted)
  - Configuration correction for HSN characteristics
- Q2
  - Policy tuning to increase placement aggressiveness
  - Configuration tuning to workload sizes
  - XE/XK physical move (to benefit utilization)
  - Worked with teams to tune job submission for improved backfill eligibility

2015 Timeline

- Q3
  - Introduce discount incentives for utilization-friendly job submission parameters
- Q4
  - Prototype DP FLOP rate monitoring enabled
  - New comparison period with traditional scheduler

A word about "utilization"

- System utilization traditionally refers to "node occupancy percentage"
- Peaking node occupancy does not necessarily peak system output, which is also affected by:
  - Network utilization
  - Filesystem utilization
  - FLOPs utilization
- Using "node occupancy" term going forward

Observations:

- TAS is successful in improving performance and consistency of communication intensive jobs
- TAS is successful in eliminating job to job interference due to network contention
- Average node occupancy decreased in part due to additional placement constraints
- System aggregate bytes delivered *increased* in spite of occupancy reduction
- Expansion factor decreased overall
- Partner complaints about slowness diminished

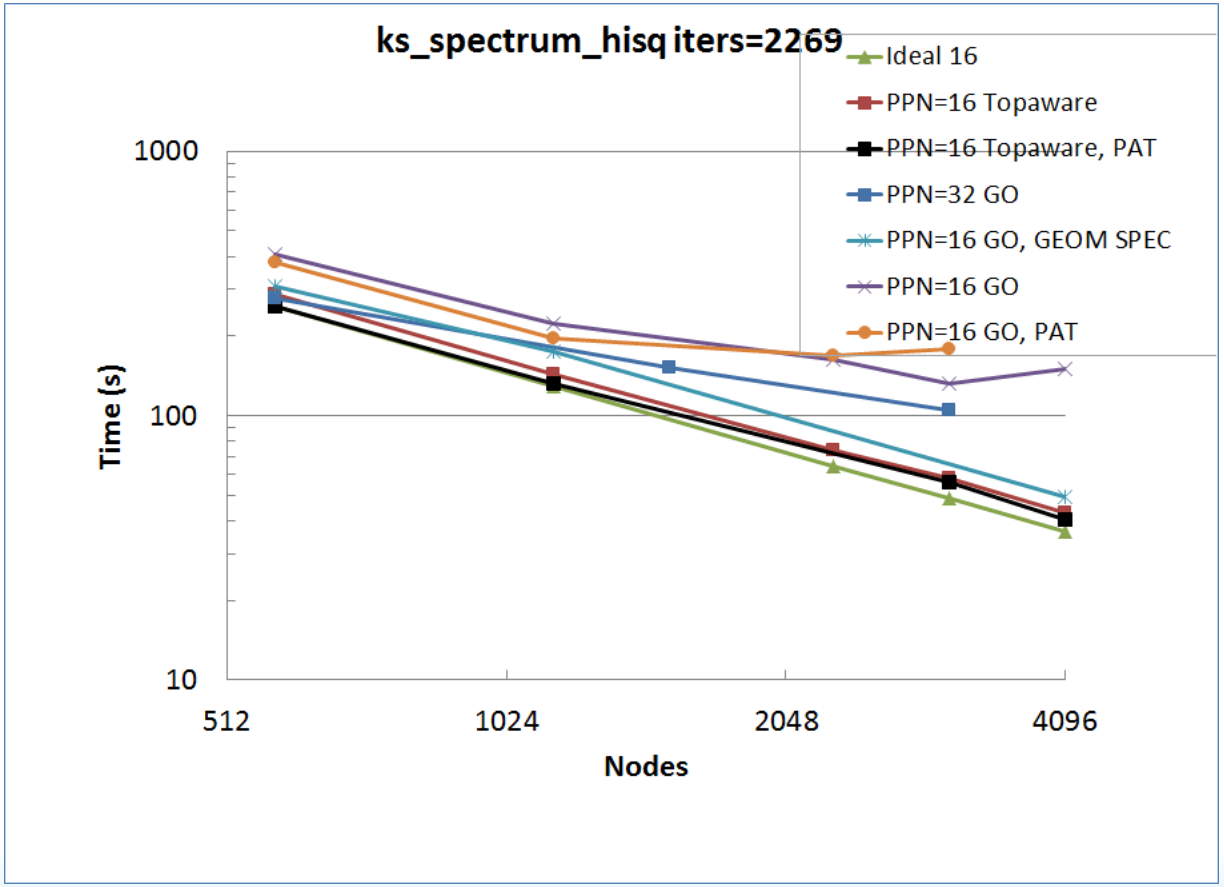
What scientists report on performance

- "My application configurations run 1.5 - 2x faster with topology aware placement" –*P.K. Yeung*
- "NAMD runs up to 25% faster under the topology aware scheduler. Blue Waters is where we go to benchmark our code because it is so consistent." –*Jim Phillips*

Scaling effect example (MILC)

1.45x speedup at 576 nodes

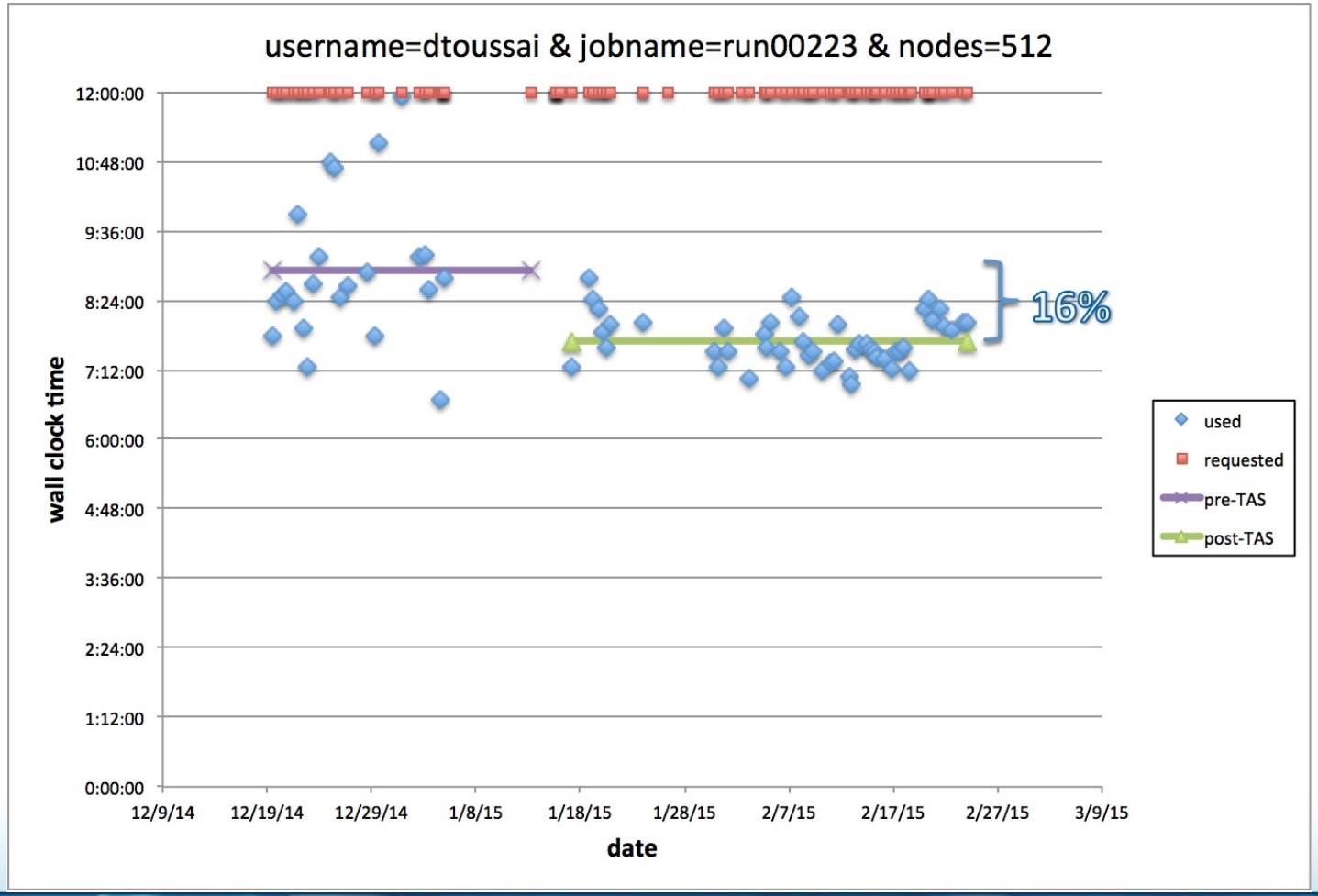
Near linear scaling only possible with TAS placement



Evaluating Science Throughput

- Performance increase needs to be greater than node occupancy reduction impact
- Partner feedback on speedup less than hoped
- Identify similar/same application runs to compare
  - UserID+JobName+JobSize+lowTASvariance
- Compare runtimes, consistency, performance metrics
- Analysis is complex! Traversed 43 TB, 5 trillion data points, 19 billion log events
- Re-enable traditional scheduler for recent comparison

Example:



Throughput Evaluation Approaches

- Within “like” job sets, network injection and FLOPs can be representative of performance
- System wide aggregate measures should be considered over broad time spans if they can vary substantially by application mix
- Per project rate comparisons
- Long periods and recent periods compared

Similar Application Set Comparisons

Dates Compared	From: Traditional ~6 months (July 1 2014 - Jan 13) To: TAS ~10 months (Jan 15– Nov 5)
Representation	92 comparable job sets 16 projects 29 distinct partners 228 MNH of allocation
App Runtime	TAS improved by 16%
App Runtime Consistency (CV)	TAS improved CV by 63%
Network Injection by app	TAS improved by 19% weighted average by node*hrs run

System-wide Aggregate Comparisons

Dates Compared	From: Traditional ~6 months (July 1 2014 - Jan 13) To: TAS ~6 months (May 1– Nov 5)
System-wide Network Injection	TAS increased by 42% 60% of projects showed increased rate 36% projected increase for weighted average by allocation size
Expansion Factor (Job size bin quartiles by node*hr contribution)	4k+ jobs increased by 202% 864-4k jobs increased by 167% 128-863 jobs increased by 3% 1-127 jobs decreased by 55% Overall average (by job count): Decrease of 50%
Gross node occupancy	12% less (9% raw difference)
Break even speedup required	15%

2 week Experiment: Application Comparison

Dates Compared	From: Traditional (Nov 10 – Nov 23) To: TAS (Sep 4 – Oct 11)
Representation	68 comparable job sets 13 projects 19 distinct partners 267 MNH of allocation
App Runtime	TAS improved by 12%
App Runtime Consistency (CV)	TAS improved by 7%
Network Injection by app	3% increase weighted average by node*hrs run
DP FLOPs	0% increase weighted average by node*hrs run

2 week Experiment: System-wide Comparison

Dates Compared	From: Traditional (Nov 10 – Nov 23) To: TAS (Sep 4 – Oct 11)
System-wide Network Injection	41% of projects showed increased rate 27% projected increase for weighted average by allocation size
DP FLOPs	35% of projects showed increased rate 26% increase for weighted average by allocation size
Expansion Factor * Measured 10/23-11/5 to maximize adjacency	4k+ jobs increased by 425% 864-4k jobs increased by 229% 128-863 jobs increased by 40% 1-127 jobs decreased by 17% Overall Average (by job count): Decrease of 11%
Gross node occupancy	Decreased by 20% (14% raw difference)

Switch to Traditional was silent, but noticed

12/3/15, UserX: "During the last few weeks I am getting significantly lower benchmarks for my simulations on NAMD even though I am using the same protocol, same number of nodes (70 nodes), and basically same simulations. Is there any problem with NAMD or the entire computing nodes? I usually run with 70 nodes and now I am getting 4-7 nanoseconds/day. But before my average performance for the same number of nodes were 8-12 nanoseconds/day. Something around 50% reduction in my benchmarks."

12/1/15, UserY: "We can run without topaware, but it does cost a lot. For the largest jobs I am doing, the ones names "fpi.XXX.XXX", using topaware gives and improvement of nearly a factor of two when using a (perhaps somewhat awkward) 1152 node partition. For our next biggest set of tasks, the jobs names "run000569...", we get a still very nice 30% speedup. So, as you can imagine, I'm very eager to get back to using topaware for these jobs."

Improvement Opportunities

- Processes and Policy
  - Preemption and flexible wallclock time (done)
  - Incentives to use backfill and submit with accurate walltime (done)
  - Explore innovations to use remaining idle nodes
  - Analyze SP/DP FLOPs, vector instructions, filesystem I/O
- Scheduler algorithm
  - Small job handling improvement expected
  - May permit deeper reservation calculation to improve large and medium job turnaround time

Summary

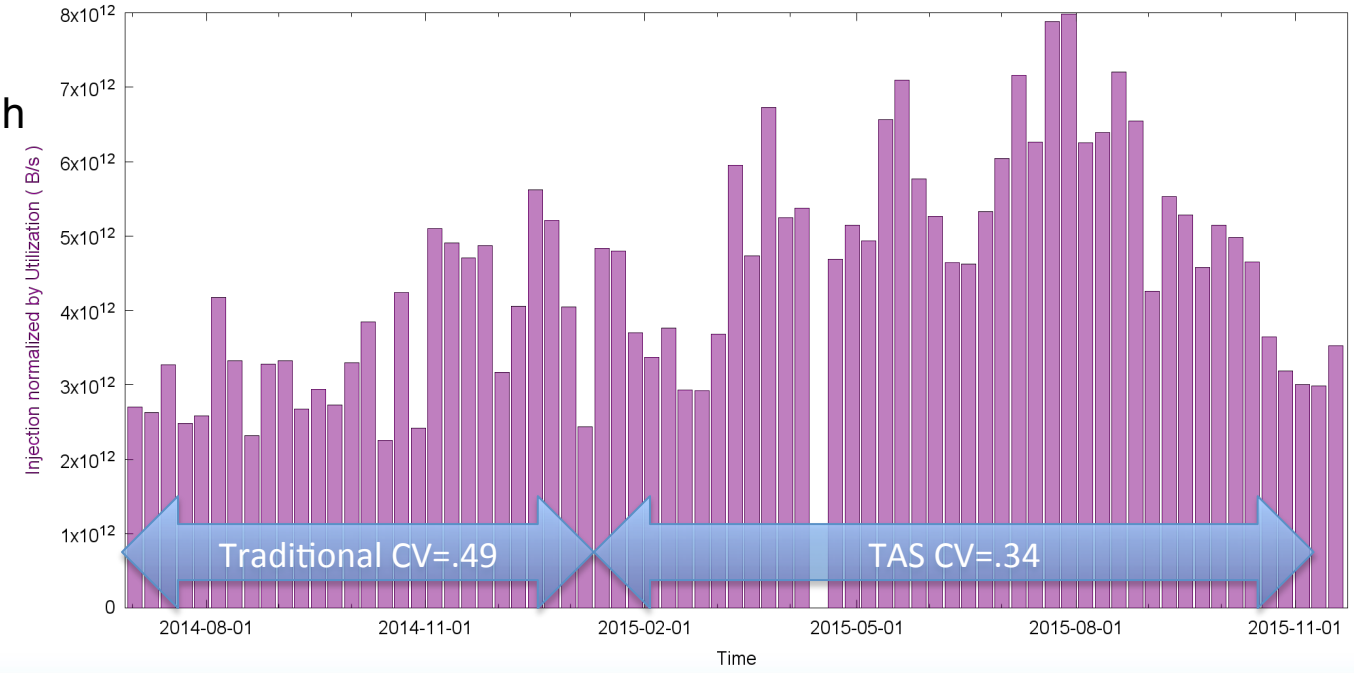
- System is delivering more science with TAS with more consistent job runtimes, with an acceptable tradeoff in turnaround time
  - Expansion factors have decreased by 39%, but increased for large jobs
- Measured Utilization impact given appropriate backlog :
  - DP FLOPs neutral for limited comparisons, but +26% projected
  - Network +42%
  - Node occupancy: -12% (excluding untuned and bug period of Q1)
- Innovative efforts to improve node occupancy will continue
  - With certain job mixes, TAS has demonstrated very high node occupancy
- For science delivery comparisons of machines or schedulers, node occupancy alone is not a sufficient metric
- We plan continued use of TAS to maximize science output and "*effective system utilization*"

Backup Slides

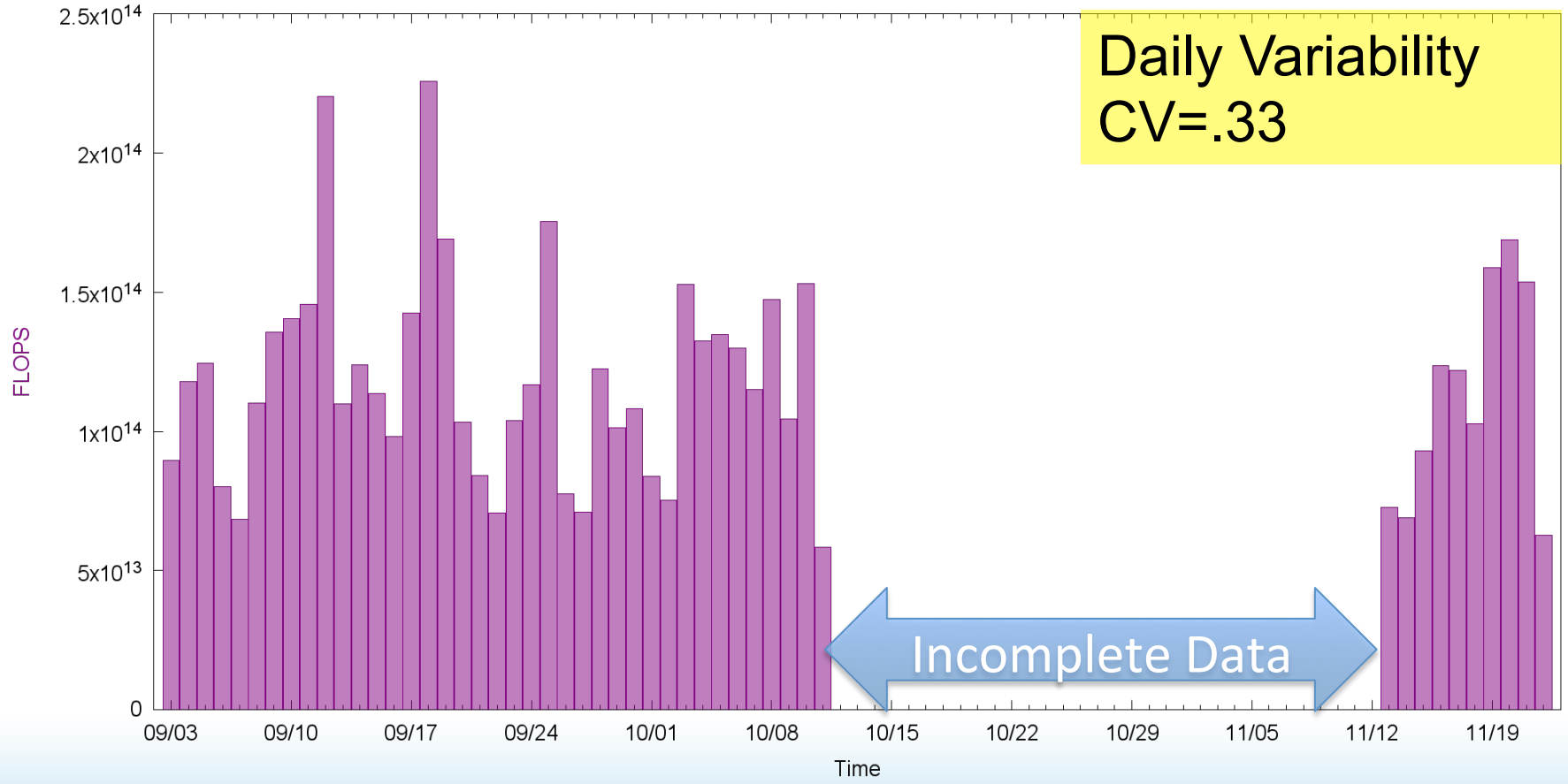
System-wide Network Injection Normalized for *Nodes in Use*

Weekly variability due to application mix, not node occupancy

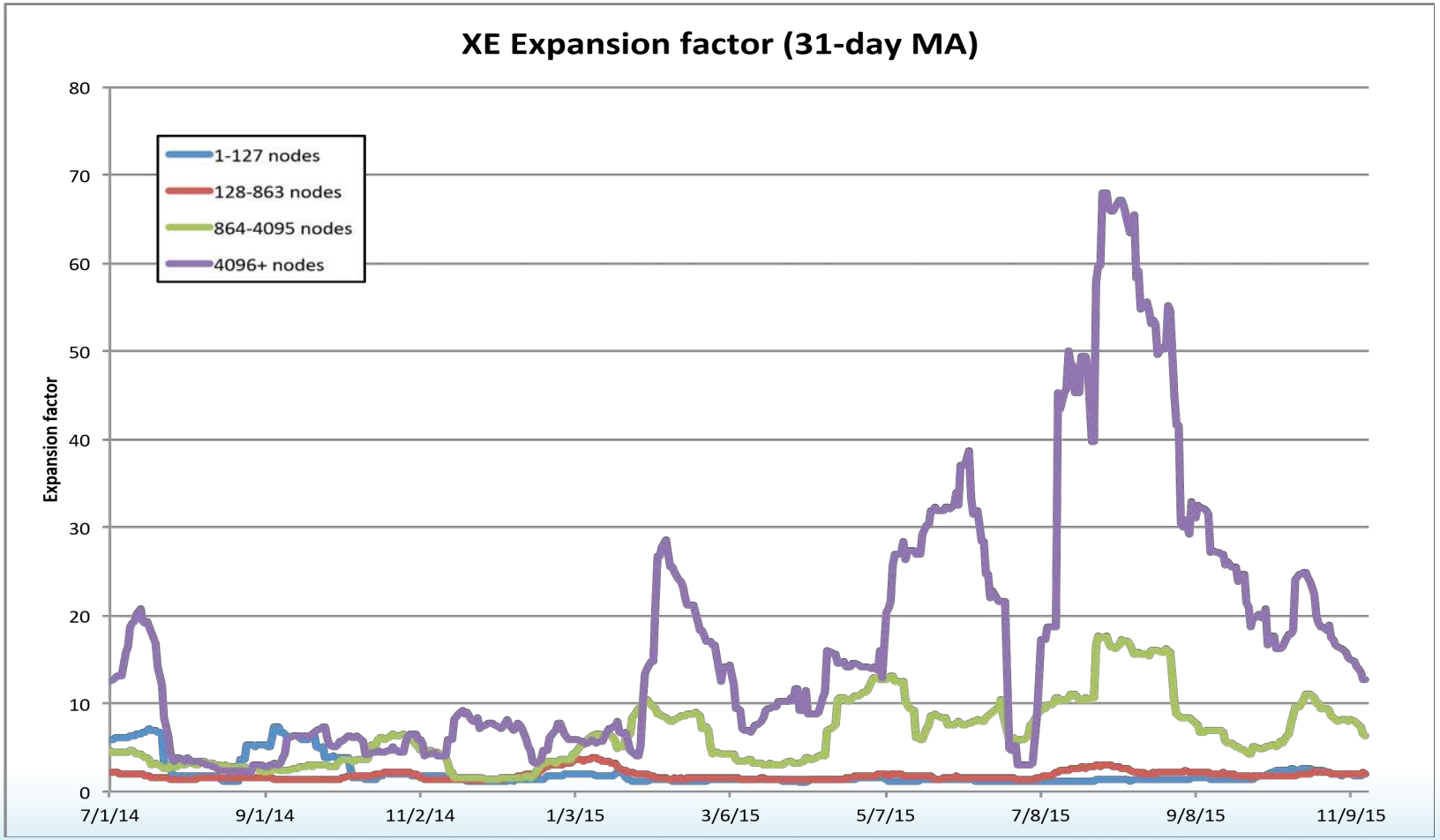
Since variance high due to application mix, longer time periods are necessary to compare system-wide network injection metric



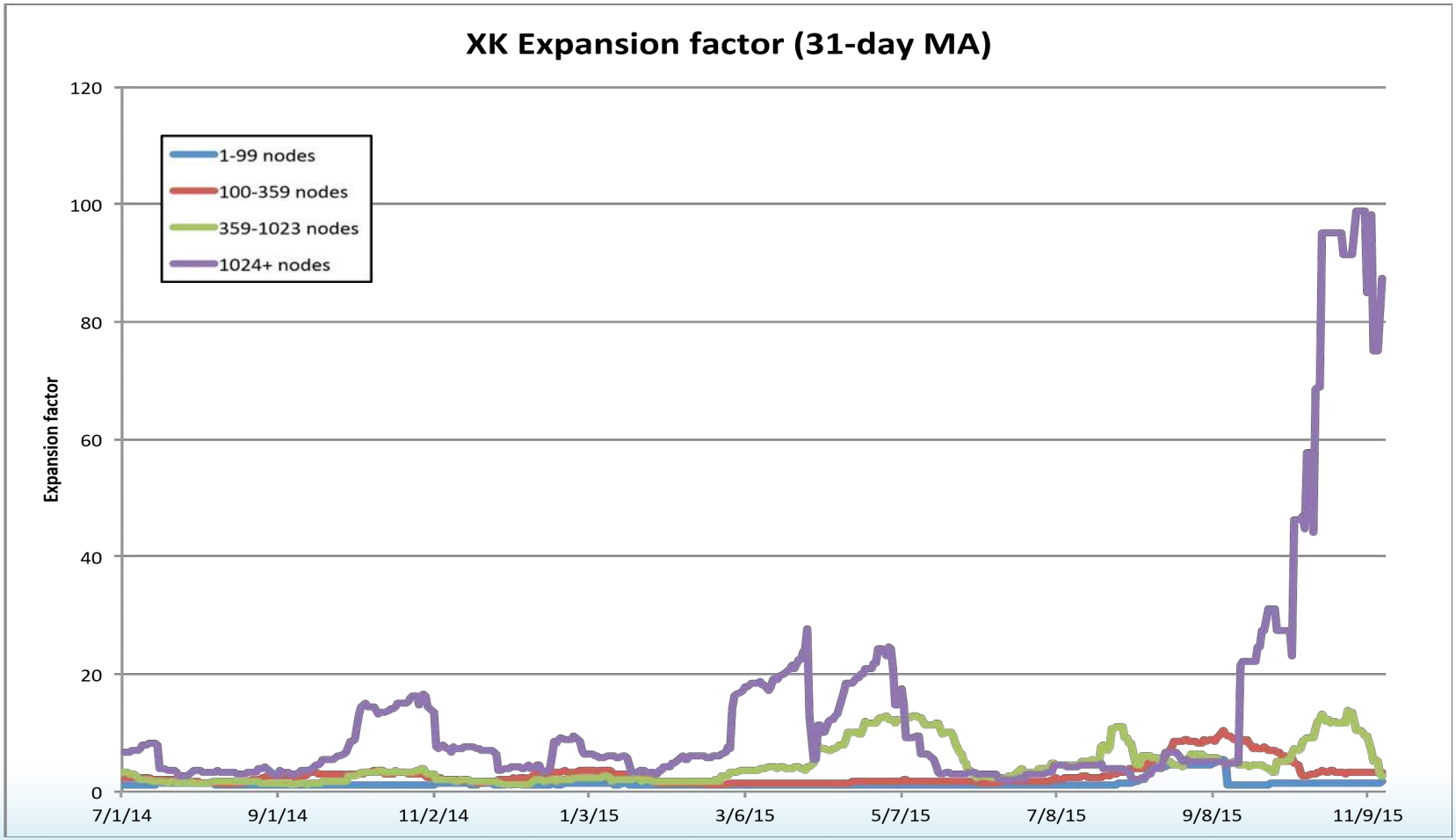
System-wide FLOPs per *Node in Use*



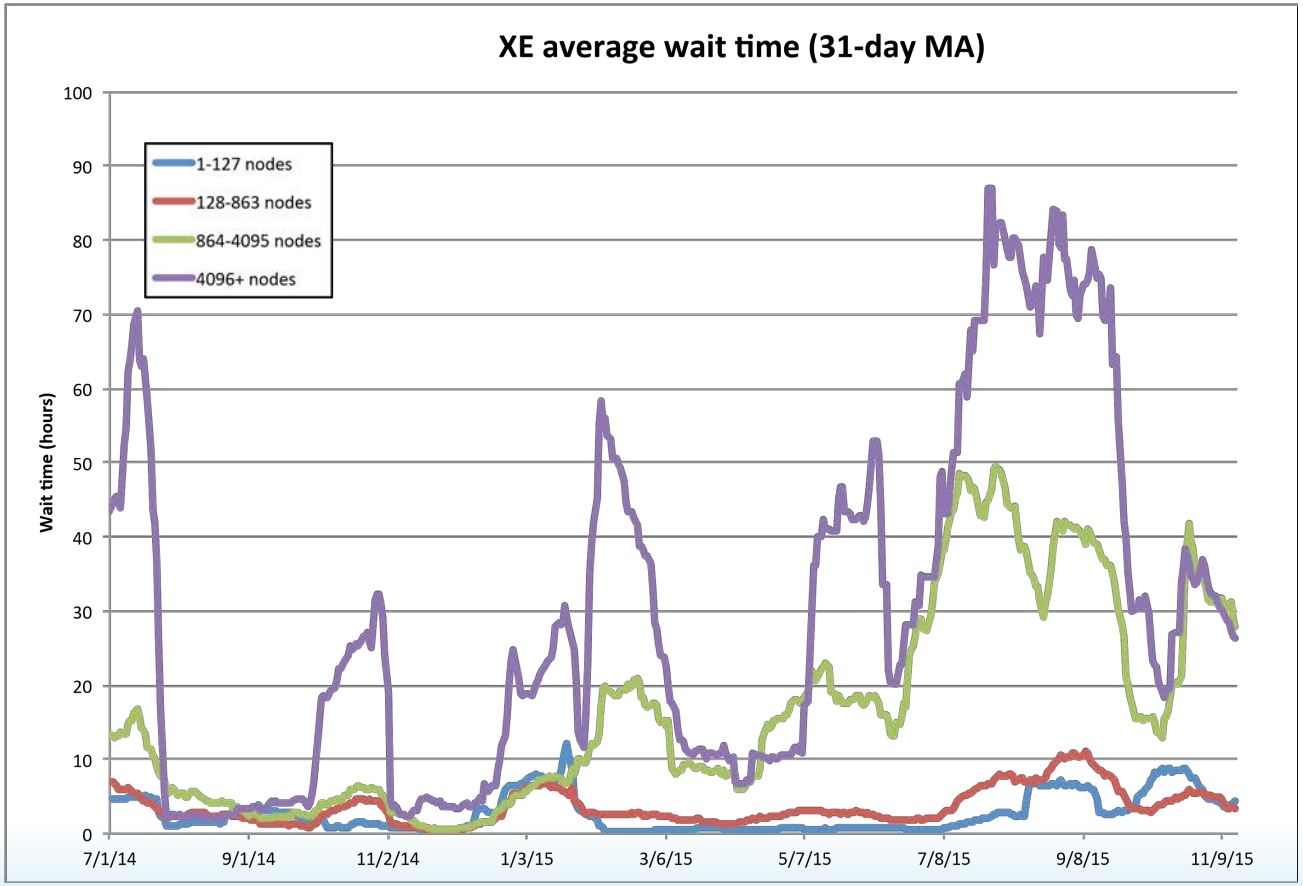
Expansion factor: XE



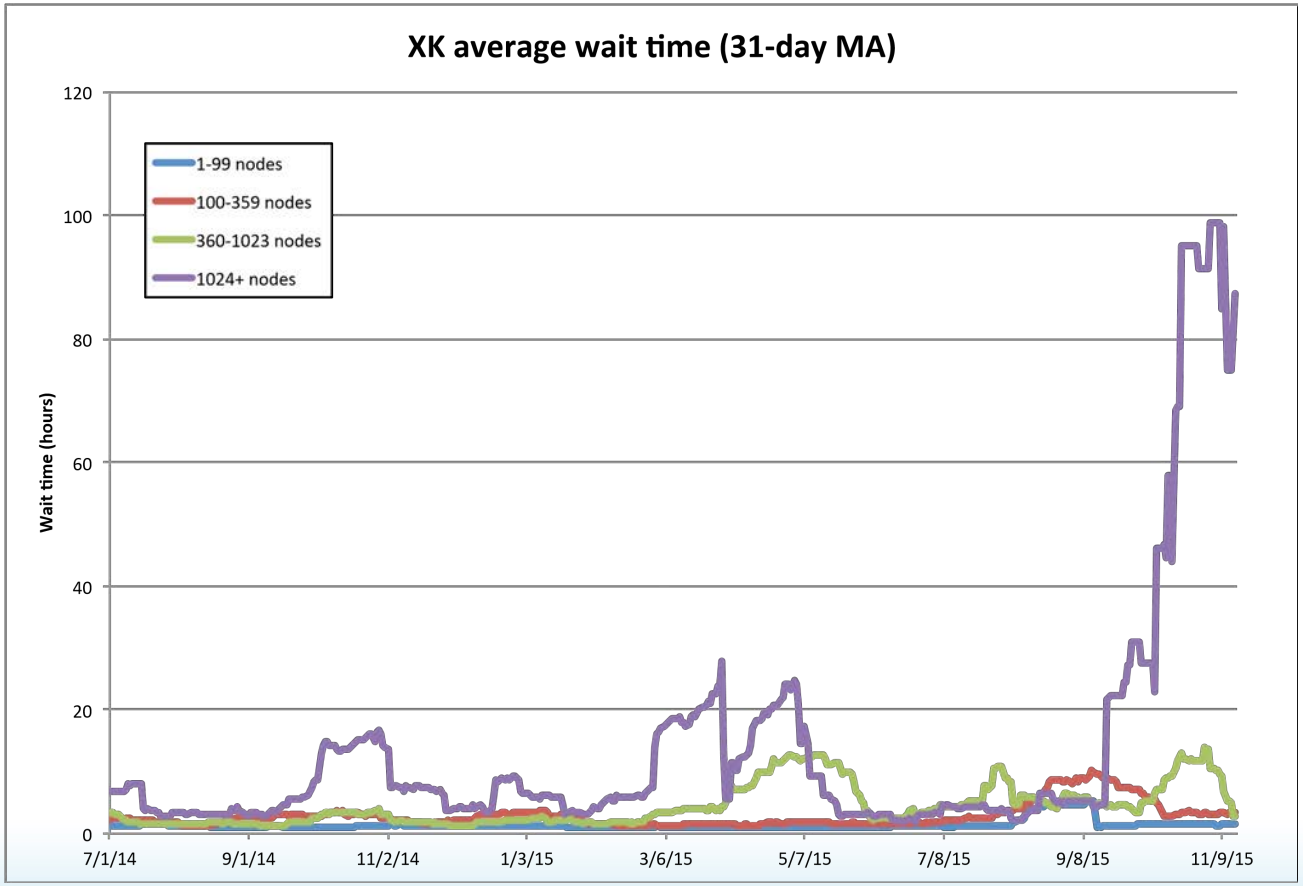
Expansion factor: XK



Average XE turnaround time



Average XK turnaround time



Equations

Throughput increase % = $100*(1-Tt/T0*U0/Ut)$

Breakeven speedup: $T0/Tt$ must be at least $U0/Ut$

Weighted injection by allocation:

$$\sum projM\uparrow projN\,I\downarrow tas/I\downarrow trad*proj\_alloc/total\_alloc$$

Expansion Factor Calculation

$$Expansion\_factor = \frac{wait\_time + requested\_time}{requested\_time}$$

- Very short jobs have *requested_time* inflated to 30 minutes.
- Jobs subjected to fair-share penalties are omitted
- Jobs submitted before but starting after a system outage are omitted

Runtime Speedup Calculation

- Find all combinations of user, job name and job size for one period, including count, average runtime and standard deviation of that runtime. We filter for completed jobs that are over 30 minutes and under 23 hours in order to eliminate stat skewing short jobs as well as 'run-to-termination' 24-hr jobs.
- Next we find and calculate the same stats for the matching sets in the other period.
- Third, we count the comparable job sets and calculate the total node hours of the matches. We also break this down into sets that showed a performance improvement with TAS and those that showed a slow-down, and calculate the node-hours saved and/or lost.

System-wide FLOPs