Capturing Computer Performance Variability in CESM Experiments

Patrick Worley
Oak Ridge National Laboratory

17th Annual CESM Workshop
June 18-21, 2012
Breckenridge, CO

Acknowledgements

- This work was sponsored by the Climate and Environmental Sciences Division of the Office of Biological and Environmental Research and by the Office of Advanced Scientific Computing Research, both in the Office of Science, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.
- This work used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725 and of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.
- These slides have been authored by a contractor of the U.S. Government under contract No. DE-AC52-07NA27344. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

And thanks to Marcia Branstetter and Kate Evans at Oak Ridge National Laboratory for allowing me to collect data from their production runs.

Performance Variability, Aspects of

1. Significant differences in execution rate between similar jobs on the same platform when using the same resource requests (e.g. processor count) and in the same computing environment (system software versions, etc.)
   a. Due to the two jobs being run on processor subsets with different 'topologies', affecting communication performance?
   b. Due to different sets of concurrently running jobs competing for shared resources (interconnect bandwidth? I/O?), and in different ways (see (a))?
   c. Due to one of the the jobs being allocated a resource that is running suboptimally ('slow compute node')

Many possible sources, some not easily identified by user, so difficult to diagnose.

Performance Variability, Aspects of

2. Significant differences in execution rate during the execution of a single job (not related to changes in the job's execution characteristics)
   a. Due to changes in competition for shared resources as other jobs come and go during the job?

   This is an aspect of (1), but can be more difficult to diagnose than when two jobs demonstrate static differences in execution rate.

3. Significant differences in execution time or execution rate between similar jobs on the same platform when using the same resource requests (e.g. processor count) but over a period of timer during which things have changed: CESM version, compiler version, communication library version, etc.

   Some change is expected, but do not want a degradation in performance to pass unnoticed. This may reflect a performance bug, and require regression to earlier versions of the code or of the software stack.

Performance Variability, Implications of

1. Jobs exceed requested time, and are aborted
   a. Some waste of allocation, depending on frequency of checkpoints
   b. Some waste of person time, as the failure is identified and required actions taken
   c. Failed jobs are resubmitted to queue, and suffer typical queue delay, slowing project productivity.
2. Increased checkpoint frequency, to decrease loss in failed jobs, consumes allocation (unproductively) in all jobs and is itself a performance variability hotspot (I/O)
3. Decreased job simulation time for a given wallclock request, to decrease failure rate due to performance variability, requires submitting more jobs to achieve same total simulation duration and thus spending more time in the queue waiting to be scheduled. This also slows project productivity.

Performance Variability, Implications of

4. Performance variability can mask application performance issues that should be eliminated, and can be mistaken for (fictitious) issues that consume software engineering time trying to correct.
5. Performance benchmarks are not reliable indicators of production run performance, and wallclock requirements:
   - Expensive to capture performance “envelope” (statistically significant sampling and retesting to capture code and system changes)
   - Expensive to use production-like benchmarks (typical runtime, variety of different code versions, variety of different configurations)

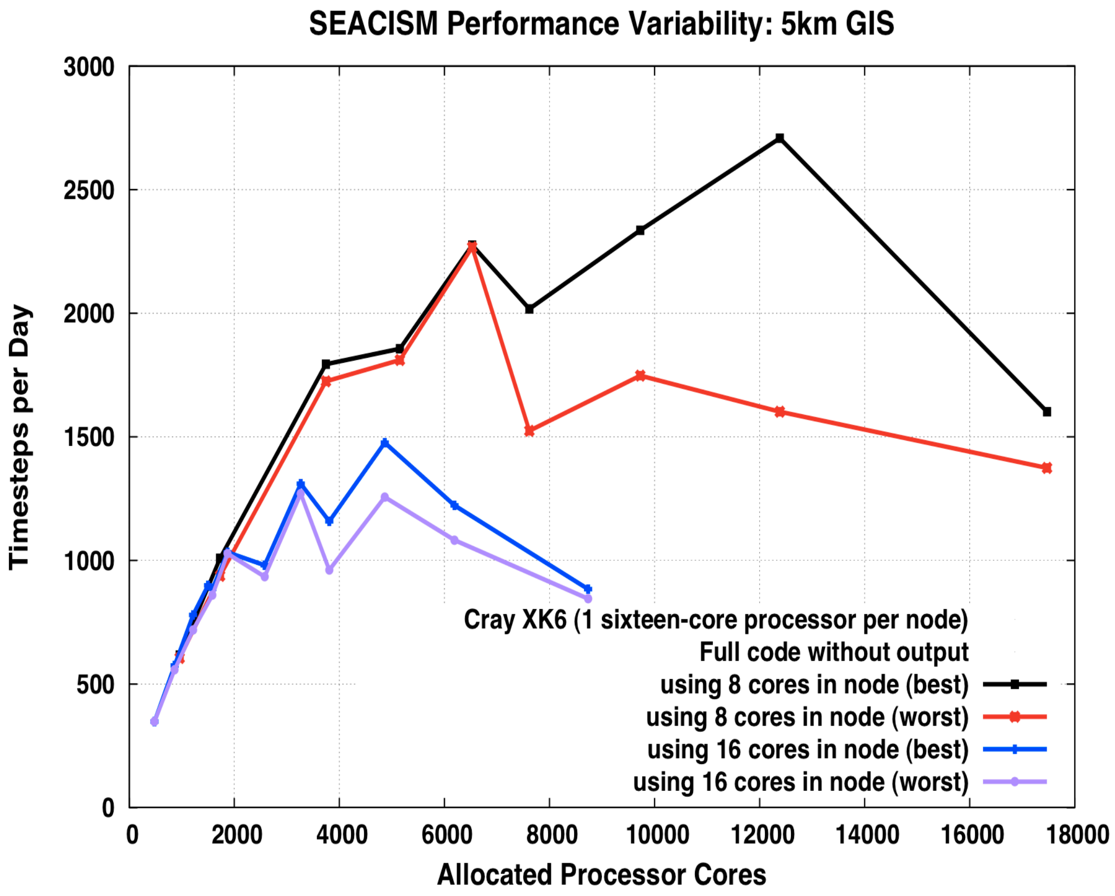
therefore

a. Users do not have dependable data for estimating wallclock requirements for individual jobs.

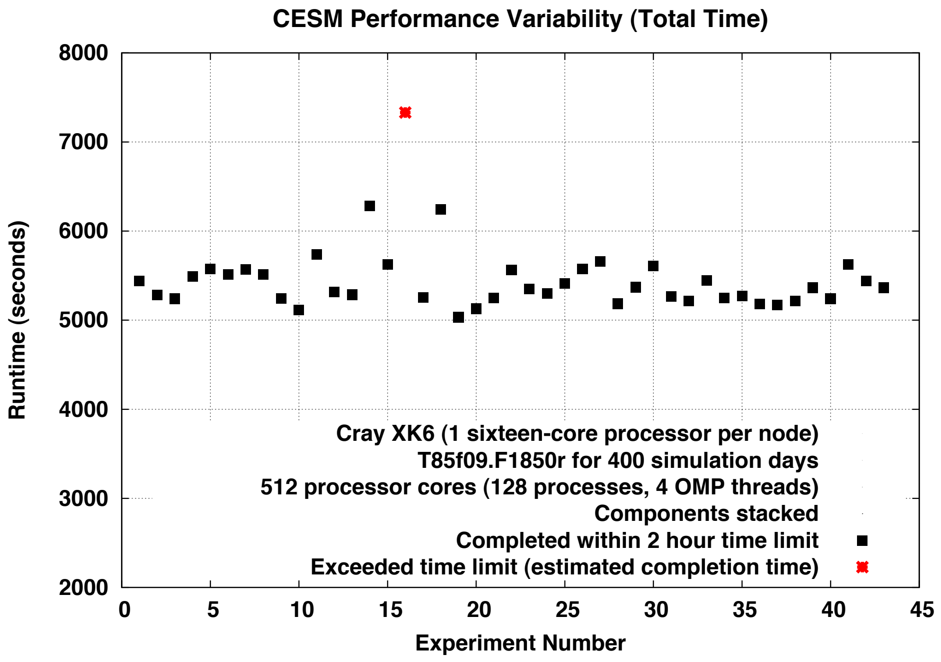
b. Project PIs do not have reliable estimates for project-wide allocation requirements.

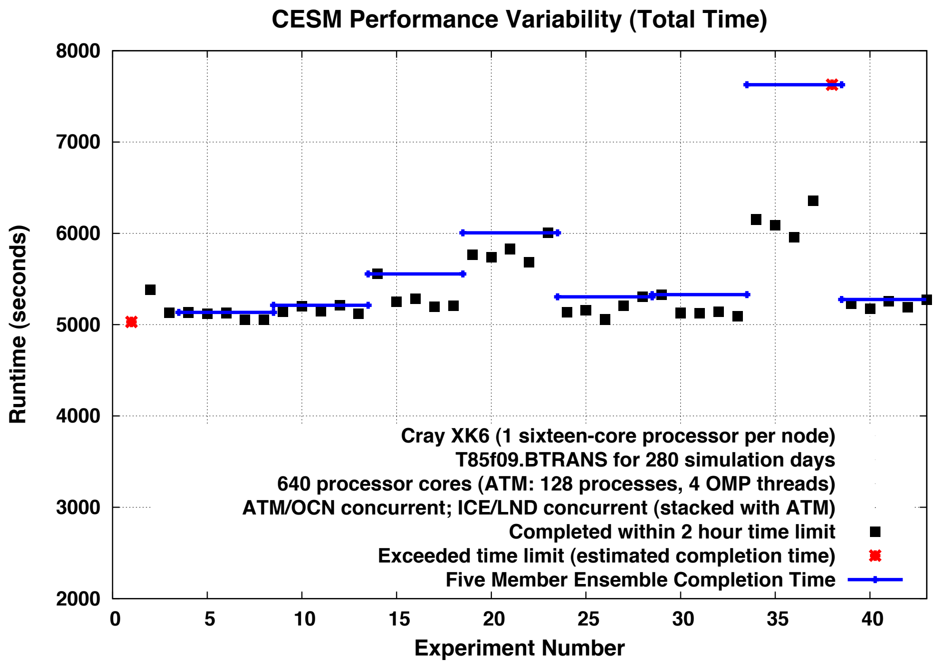
Example: CISM (5km Greenland Ice Sheet)

Even when comparing between just two or three runs each for a sequence of processor core counts, the distributed solution of the linear system at the core of the Newton_Krylov solver for the ice sheet velocities exhibits significant performance variability for large processor counts, affecting total model performance. Just recently quantified (data collected June 4-9, 2012) – will now start identifying sources and mitigation options.



Examples: CESM (T85f09: F1850r and BTRANS)

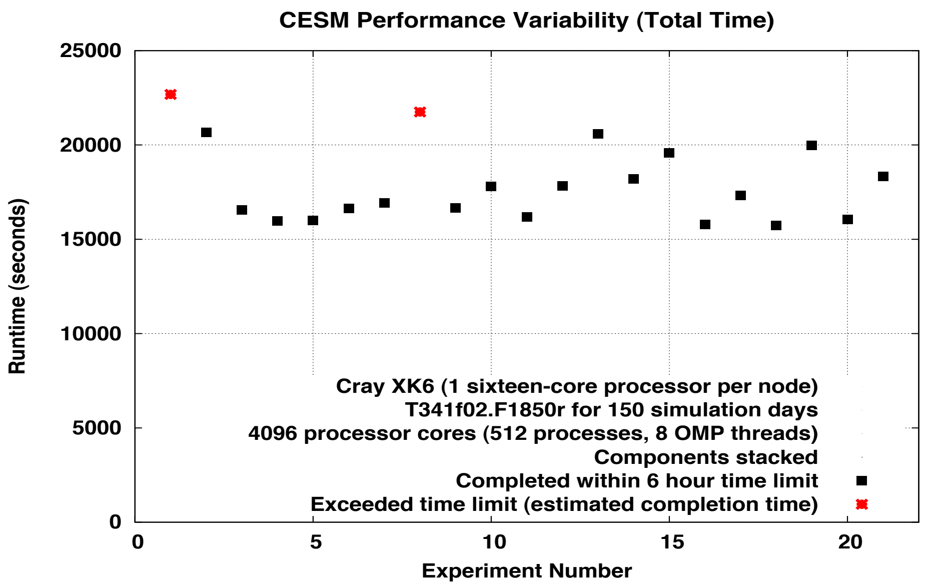


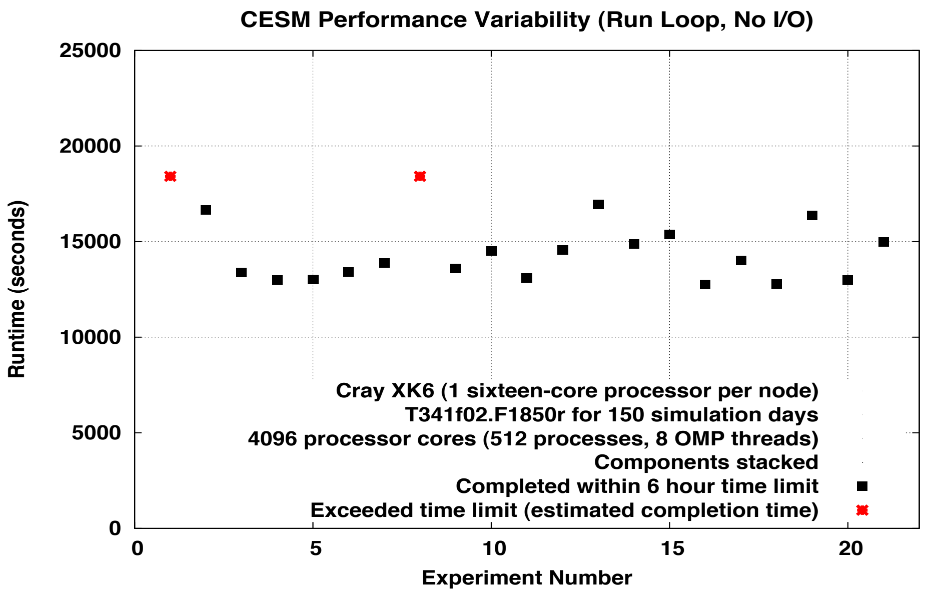


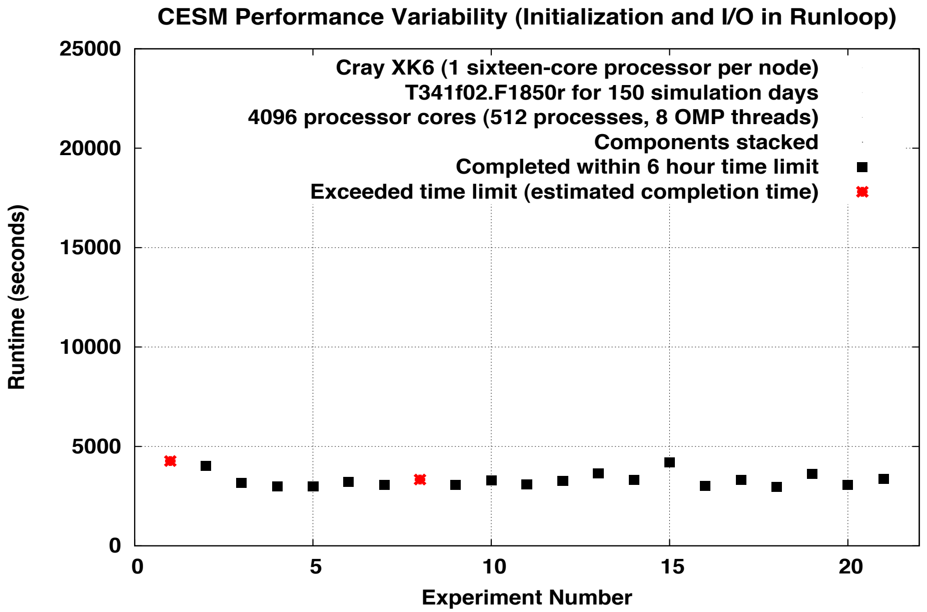
- 43 jobs, each computing 400 simulation days. Data collected between May 15 and May 22, 2012
- One exceeded 2 hour limit, sometime between simulation days 325 and 400.
- Slowest successful job took 1 hour, 45 minutes; fastest took 1 hour, 25 minutes.

- 43 jobs, including 8 five-element ensembles (each submitted as a single job), each computing 280 simulation days. Data collected between May 23 and June 6, 2012
- Two jobs exceeded 2 hour limit, both between simulation days 250 and 280.
- Slowest successful ensemble took 1 hour, 40 minutes; fastest took 1 hour, 26 minutes.

Example: CESM (T341f02.F1850r)

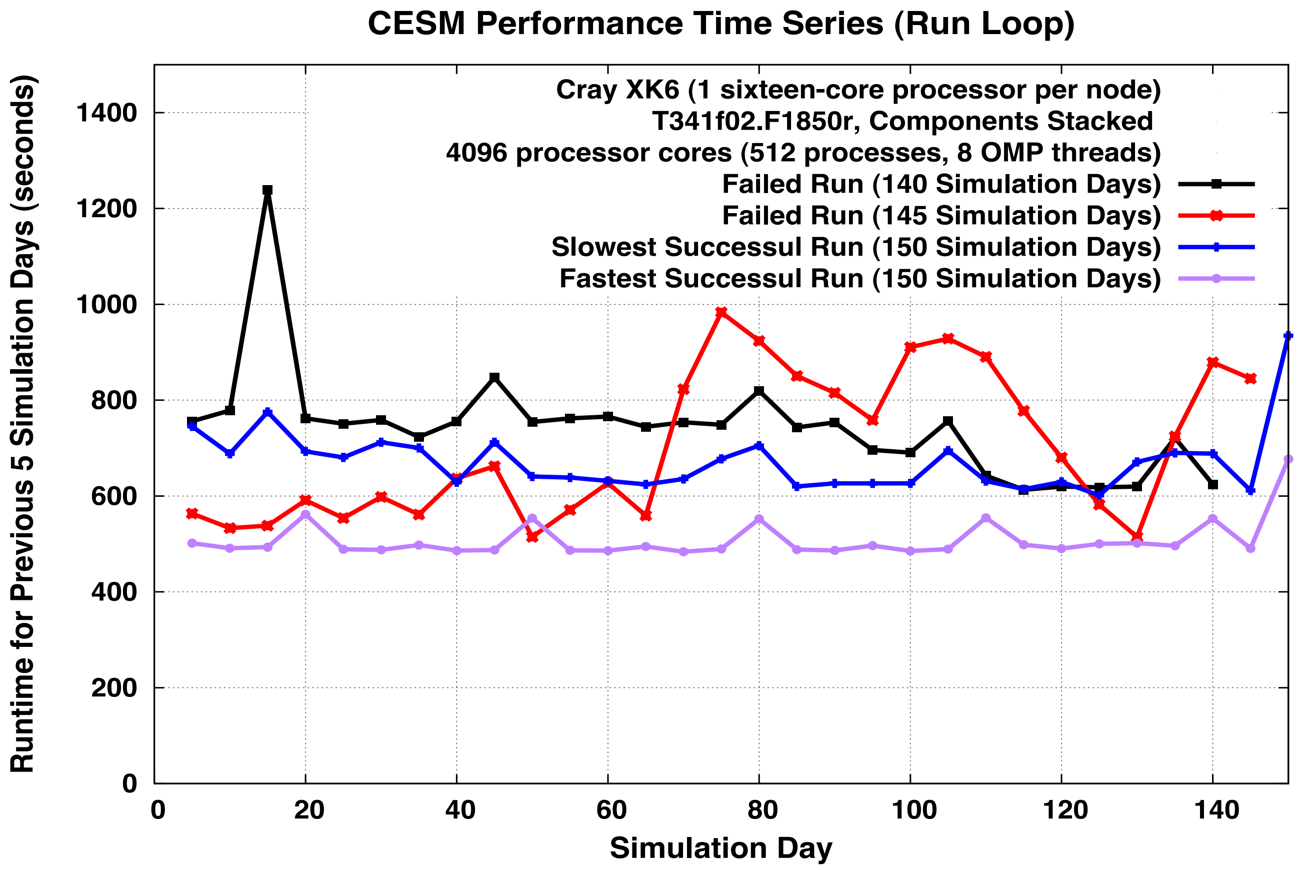






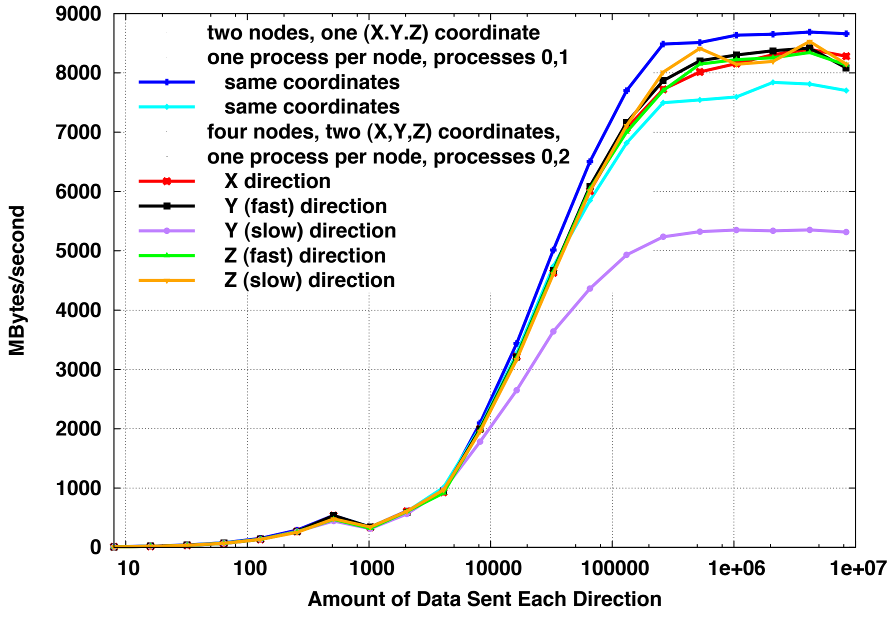
- 21 jobs, each computing 150 simulation days. Data collected between May 15 and June 8, 2012.
- Two exceeded 6 hour limit, one between simulation days 140 and 145 days, and one between 145 and 150.
- Based on benchmark runs, expected to be able to complete 180 days in 6 hours.
- Note that both I/O and non-I/O demonstrated performance variability

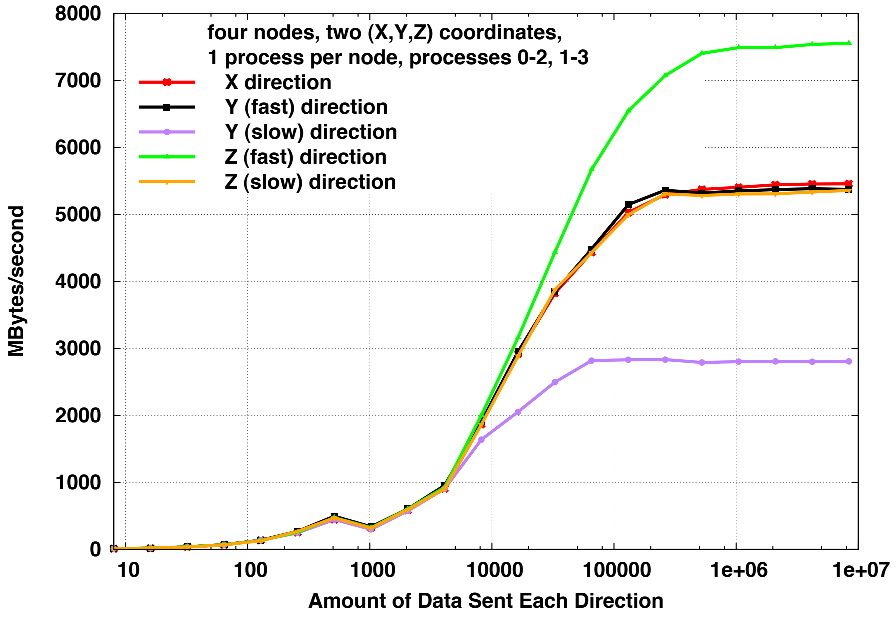
Example: CESM (T341f02.F1850r)



- Comparing performance between two jobs that exceeded the 6 hour wallclock limit and the fastest and slowest successful runs (completed in 4.5 and 5.75 hours, respectively) . The failed experiments exhibit high internal performance variability. The successful runs have primarily different “base” (or static) performance levels. This was just happenstance - neither of these are necessary characteristics of “failed” or “successful” runs.

Aside: Gemini Interconnect Asymmetries





- 3D torus interconnect topology
- Two compute nodes per (X,Y,Z) coordinate, connected via a single Gemini switch
- Messages between nodes differing by one in either X, Y, or Z coordinates go through two Gemini switches
- For communication between nodes that are neighbors in the Y direction, performance differs depending on whether the smaller Y-coordinate is even (“faster”) or is odd (“slower”).
- For communication between nodes that are neighbors in the Z direction, every eighth link is “slower”.

Example: CESM (T341f02.F1850r)

- Process assignments for fastest successful job
  - All processes assigned to “complete” node pairs
  - 4 X-coordinates: 21,22,23,24 (128 each)
  - 2 Y-coordinates: 2, 3 (256 each, so no ‘slow’ Y links)
  - 16 Z-coordinate: 8-23 (32 each)

  So contiguous 4x2x16 allocation (no ‘holes’) with no “slow” Y links. Did include 1 “slow” Z link.

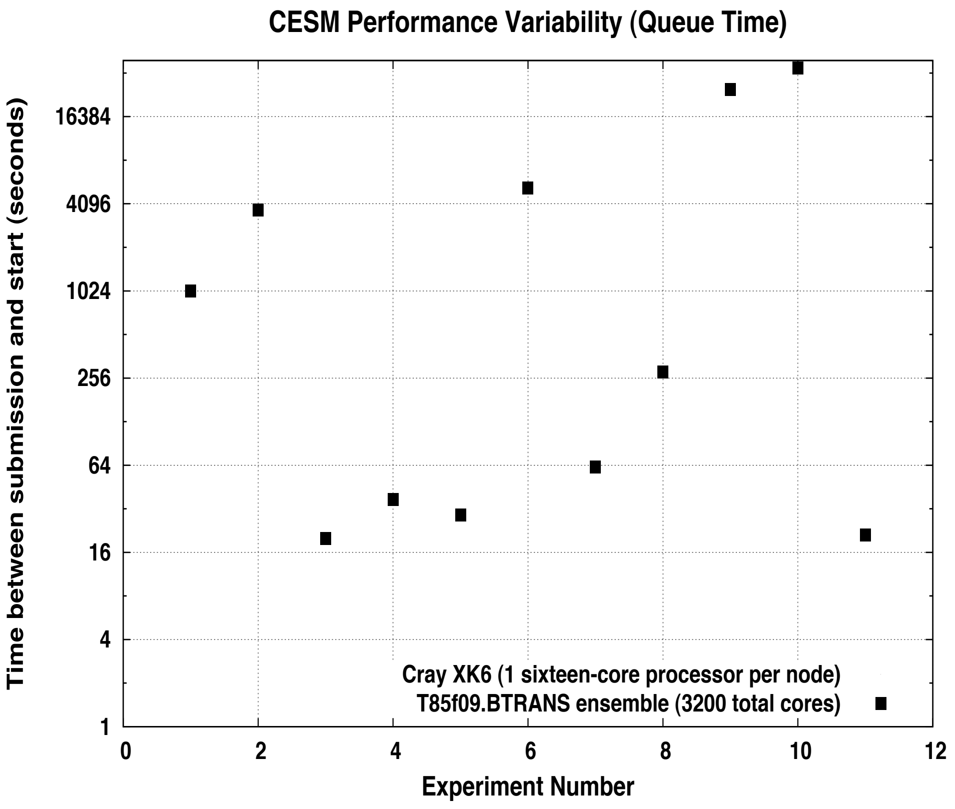
Example: CESM (T341f02.F1850r)

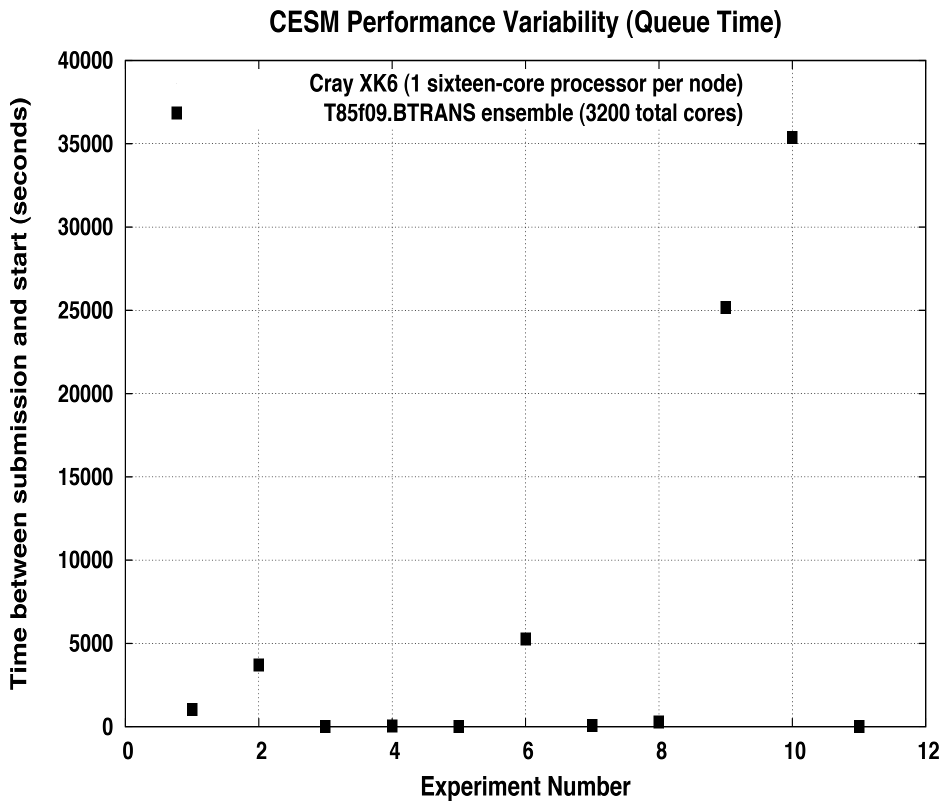
- Process assignment for slowest successful job
  - 272 processes assigned to 136 “incomplete” node pairs (53%)
  - processes from 9 other running jobs assigned to the “other” nodes in 130 of these node pairs, with 5 other nodes allocated but with no running jobs yet, when the T341f02.F1850r jobs started. (Details on these jobs are available, as well as changes during the execution of the T341f02.F1850r job.)
  - 8 X-coordinates: 17-24 (varying between 14 and 142 each)
  - 10 Y-coordinates: 0-5, 8-9, 14-15 (varying between 16 and 92 each)
  - 24 Z-coordinates: 0-23 (varying between 6 and 44 each)

  So widely scattered nodes, with many holes in the vertex cover, and perhaps including communication over a “slow” Y link.

  Potential for interconnect contention from jobs running on other nodes in incomplete node pairs, and from other jobs sharing other interconnect links, based on noncompact node allocation (but nothing proven by these data).

Example: CESM (T85f09 BTRANS)





- Time spent in queue for an ‘enhanced priority’ project – still includes delays as long as 9 hours. Same data plotted on left (linear-log) and on right (linear-linear). “Normal priority” job submissions can stay in queue for multiple days.

Performance Variability, Summary

1. Performance variability has been observed for many years, and is likely to continue to be an issue for CESM target platforms in the future.
2. Large and frequent performance variability is **costly**.
3. First steps are identification, quantification, and diagnosis
    a. Without this, cannot identify mitigation strategies, nor convince those who might be able to address the issues directly that there is in fact a problem worth addressing.
    b. Also need to capture full costs, including time spent compiling, time spent in queue waiting to be scheduled, time postprocessing, in order to understand the true impacts, and to understand what are the true performance bottlenecks in the project workflow.
4. Preliminary work on augmentation to existing performance data capture logic that can be used to document and diagnose performance variability is promising, but needs to be included in released CESM versions and used in production runs

CESM Instrumentation Proposal (General / New Defaults)

1. Capture additional timing data in the model
   - already supported in model, but not on by default
2. Capture runtime global statistics data and individual process data for representative processes for each component (e.g. component root)
   - currently collect individual process data for all processes and not collecting runtime global statistics
   - by using runtime global statistics, do not need data for all processes in order to generate summarization
3. Capture performance data periodically during the run
   - already supported in model, but not on by default
   - frequency currently based on number of days, steps, simulation date, etc., but should be a function of elapsed wallclock time
4. Job-specific timing directory name, so that preserved when job aborts (not overwritten by next run)
   - add option to delete if not needed after timing summary generated

CESM Instrumentation Proposal
(System and Project Specific – Support for in Model)

1. Add time job queued and scheduled to CaseStatus
2. Capture system interconnect topology
3. Capture what physical processors job is running on, for whole job and for each component
4. Capture what other jobs are running, and where, just before job starts, and right after it ends
5. Run background job to query what other jobs are running, to complement periodic timing data capture
6. Copy job performance data to project archive when complete
7. Track performance variability, generating reports automatically to
   - identify anomalies
   - identify trends
   - calculate true critical path and costs for project as a whole

Next Steps

1. Refine logic and introduce into currently instrumented and as yet uninstrumented production jobs, including 0.1 degree ocean and running on much larger process counts.
2. Refine and further develop postprocessing tools.
3. Improve performance data archiving (perhaps by introducing a database)
4. Convince CSEG that something like this capability is worth including in a release.
5. Develop user-level scheduling procedures that can mitigate performance variability:
    a. Run time evaluation and abort if potentially bad node allocation?
    b. Overallocation and use more efficient subset?
6. Convince Centers to do use less aggressive scheduling algorithms, or let users place restrictions on characteristics of allocations willing to accept?