Checkpointing Tips

Scott Atchley

OLCF User Call

March 29, 2023

ORNL is managed by UT-Battelle LLC for the US Department of Energy

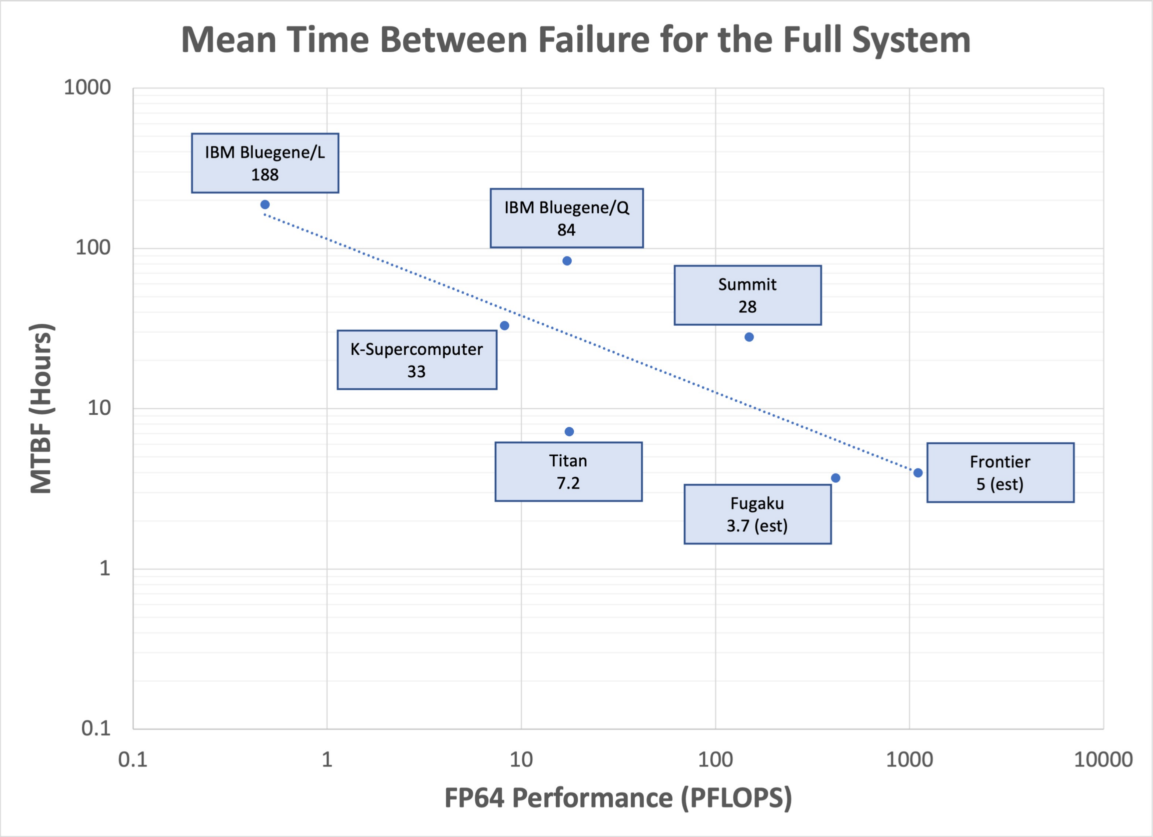
Agenda

- Mean Time Between Failure (MBTF)
- Understanding Scaling Impact on MTBF
- Mitigations

Mean Time Between Failure (MTBF)

Mean Time Between Failure (MTBF)

- MTBF is average time between node failures
  - MTBF is given for the entire system
- As systems grow in scale and complexity, the MTBF has continued to decline
  - If a part has a MTBF of 1M hours and if the system has 1M of these parts, then the MTBF for the system is 1 hour
  - Frontier has over 60 million parts
    - Some parts have 10s-100s of sub-parts
    - E.g., 1 GPU has two GPU chips, 8 HBM stacks, 10s of power converters, etc.
- MTBF was identified as one of the four key challenges for reaching exascale

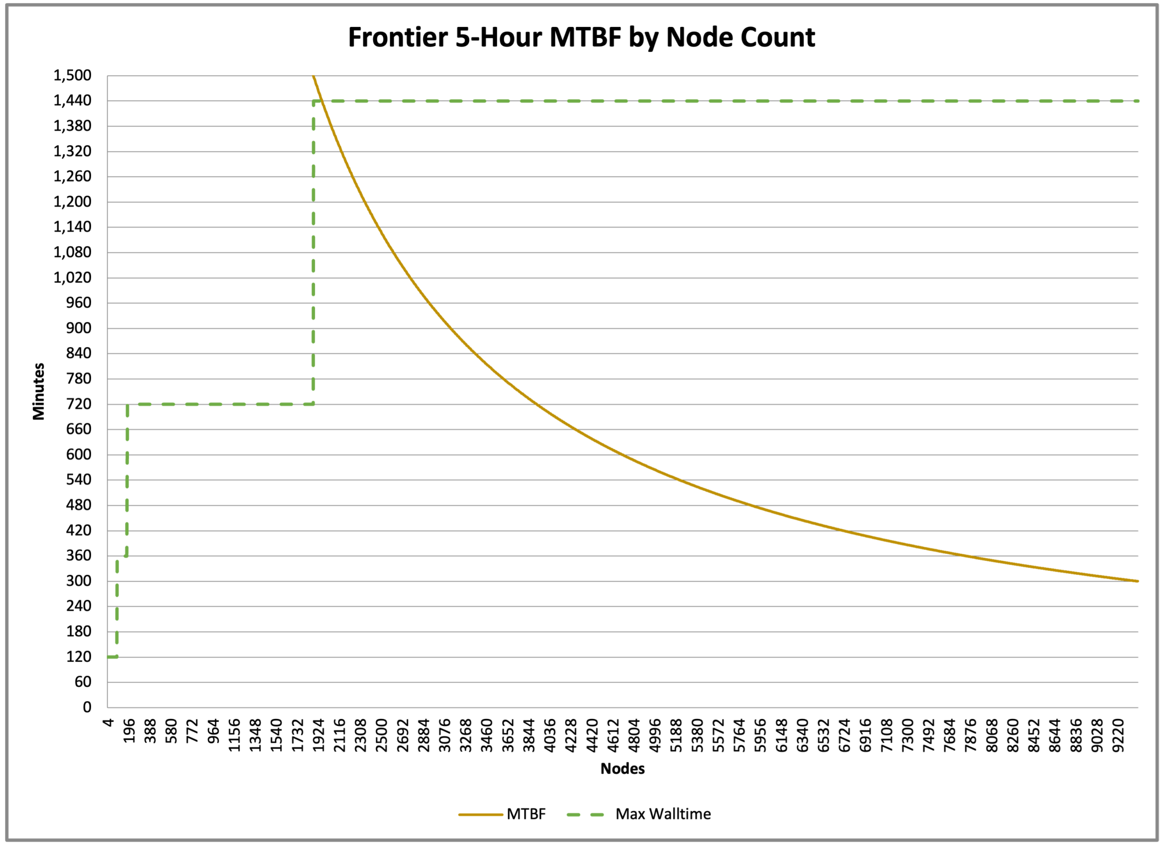


Slide courtesy of Samsung USA

Understanding Scaling Impact on MTBF

Understanding Scaling Impact on MTBF

- MTBF is for the full system
  - Varies day-to-day
  - Varies by workload
- MTBF for a subset of nodes scales linearly
  - If using 50% of the system, the MTBF is 2x
  - If using 20%, then it is 5x higher
- Should improve over time
  - Still early in the *bathtub curve*
  - Replacing components when they fail, and when we can get replacements
    - Supply chain issues are still present
- **Leadership jobs will likely run into node failure**



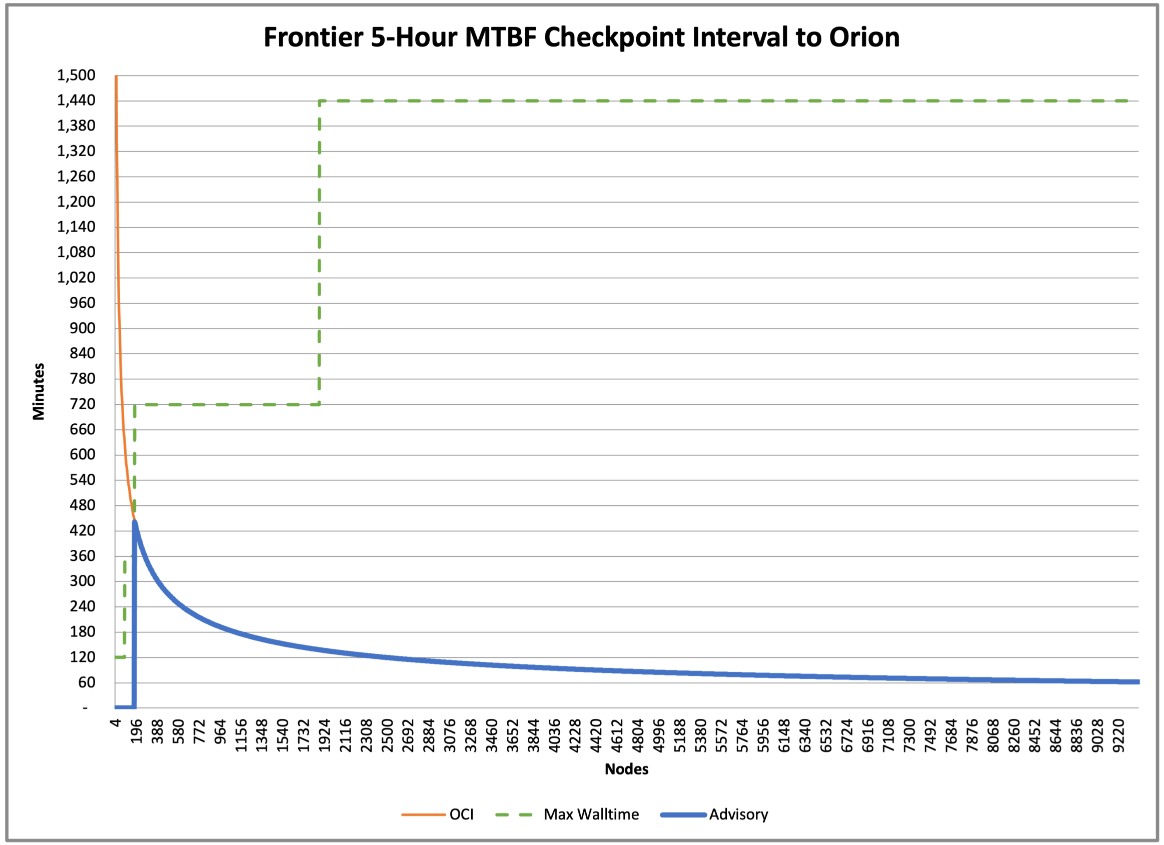
Mitigations

Mitigations

- Checkpoint/Restart
  - Checkpoint Frequency
  - Accelerating I/O
  - Defensive Checkpointing
  - Managing Defensive Checkpoints Using SCR
- Handling node failure
  - Continuing After Failure

Checkpoint Frequency

- Daly’s optimal checkpoint frequency
  - Considers compute time, checkpoint time, rework time, and restart time
  - Checkpointing too ***frequently*** increases total solve time when there is no failure
  - Checkpointing too ***infrequently*** increases rework time when encountering failure
  - Faster checkpoints argues for more frequent checkpoints
    - Formula does *not* account for storage capacity
- Assumptions
  - 5-hour MTBF
  - ~2 TB/s to Orion (Lustre)



For full system: ~1 hour
For 20% of the system: ~2 hours

Accelerating I/O (1/2)

- Lustre
  - File per process instead of single shared file
  - If using single, shared file, then stripe wide
- Node-Local SSDs
  - Faster writes, but not available if the node crashes
    - Common fault domain
  - OLCF is developing SPECTRAL to redirect writes to the SSD to allow the application to resume work and then copies the data to Lustre in the background.
    - Not for checkpointing, but OLCF is also developing HVAC to cache reads on the SSD to accelerate AI/ML/DL workloads.

Accelerating I/O (2/2)

- ADIOS
  - Manages I/O
    - Can switch to File Per Process
    - Also provides ability to perform in situ analysis
  - Apps running on Summit have seen 33% speedups
  - Apps running on 2,048 Frontier nodes are getting up to 5 TB/s

ADIOS documentation: https://adios2.readthedocs.io/en/latest/index.html
ADIOS Examples: https://adios2-examples.readthedocs.io/en/latest/

ADIOS source code: https://github.com/ornladios/ADIOS2

Tutorials:
ECP 2021: https://www.youtube.com/watch?v=GvuZLSYqmNs https://users.nccs.gov/~pnorbert/ADIOS_tutorial_ECP_AHM_Apr2021.pdf
ECP 2023 recent updates: https://users.nccs.gov/~pnorbert/ADIOS_tutorial_ECP_2023Feb.pdf

Online help:
ADIOS2 GitHub Issues: https://github.com/ornladios/ADIOS2/issues

Defensive Checkpointing

- So far, assuming that a checkpoint is a usable/intended output
- A defensive checkpoint is an output that you would typically not want and only is meant for restart
  - Would not normally be included in the output analysis
- Is only valuable until the next checkpoint (usable or defensive) is written (and moved into Lustre)
- Need to be cleaned up (deleted) at some point

Managing Defensive Checkpoints Using SCR

- Livermore created Scalable Checkpoint/Restart (SCR) library
- Manages checkpoints for applications
- Takes advantage of *close* storage including node-local SSDs
- Can manage usable and defensive checkpoints
  - E.g., move every Nth checkpoint from node-local to Lustre
- Can decouple checkpoints from the node's fault domain
- Not ready yet on Frontier/Crusher

Continuing After Failure

- By default, if a jobstep (i.e., srun) fails, Slurm will kill the job
  - Back to the queue
- To try to continue, do:
  - Allocate an extra node (or nodes)
  - In a loop,
    - run a jobstep with --no-kill (srun --no-kill …)
    - Check the return code of the jobstep
      - If success, exit the loop
      - If failure, re-launch by pointing at the most recent checkpoint
  - This works for small jobs, but might not work for leadership jobs
    - We are working with SchedMD to enable this for large jobs

Conclusion

- Frontier's MTBF is less than previous OLCF systems
  - Less than the longest queue time of 12 hours for leadership jobs
- Will require users to evaluate their applications' output frequency
- Add defensive checkpoints as needed
- Consider using tools to accelerate I/O and/or to manage checkpoints
- Integrate restart into your job script to avoid going to the back of the queue (when fixed)

Questions?