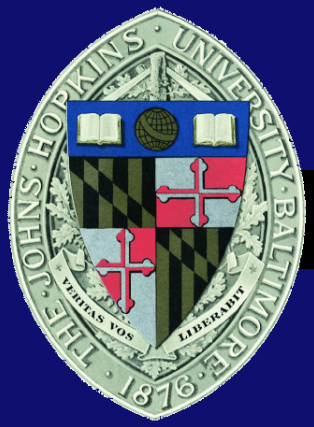
Lecture 17.3
I/O Performance

EN 600.320/420

Instructor: Randal Burns

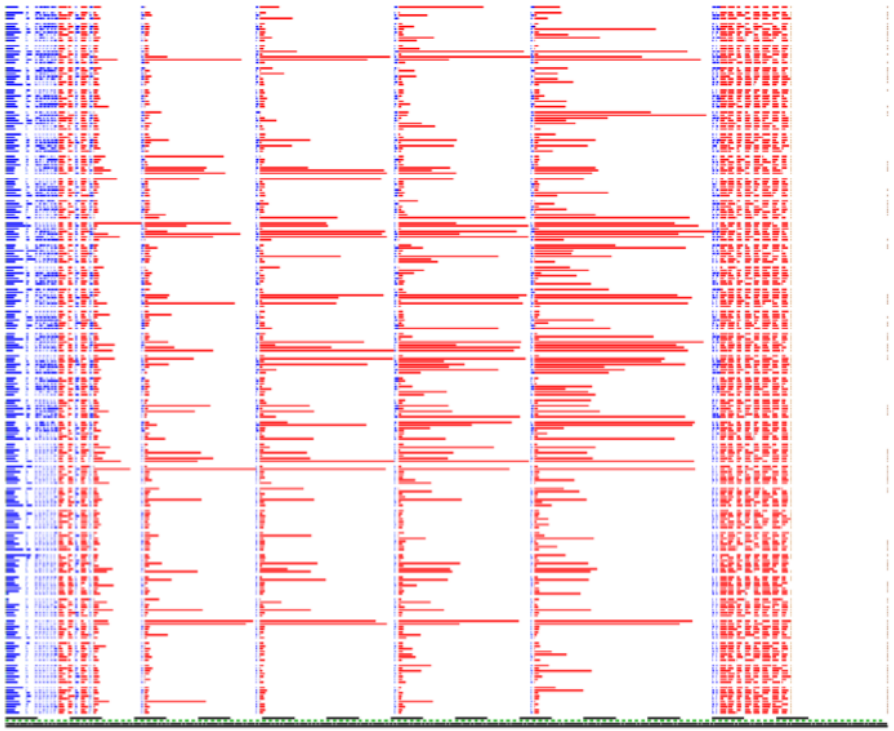
12 April 2017



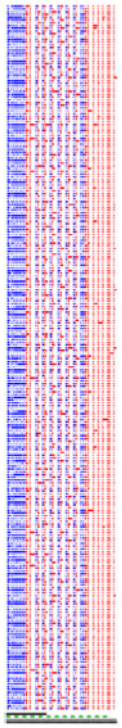
Department of Computer Science, *Johns Hopkins University*

Fixing I/O Performance

- Compare same I/O benchmark on two platforms
  - 256 nodes of Franklin and Jaguar

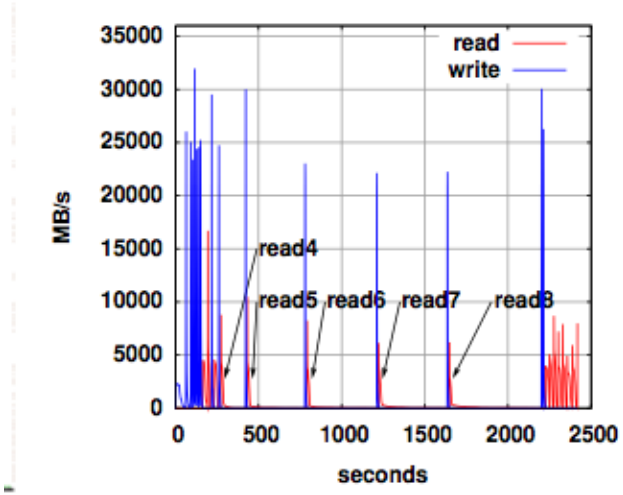


(a) Franklin trace

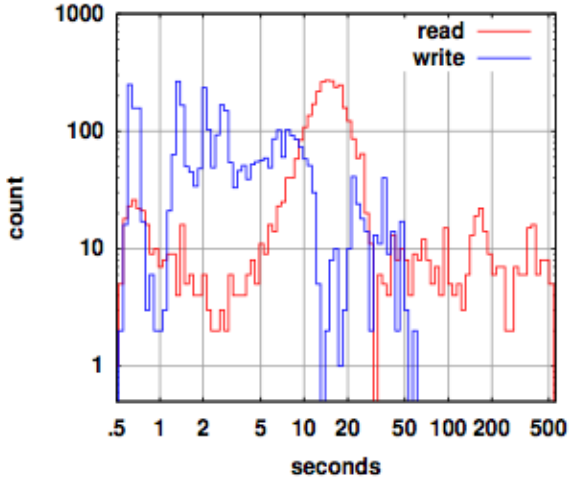


(d) Jaguar trace

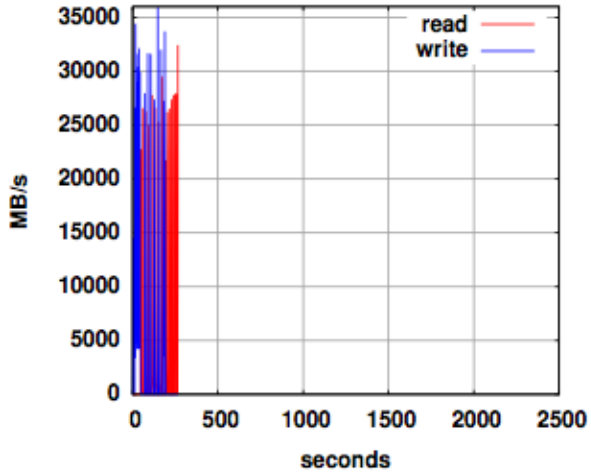
Problem = Long Read Delays



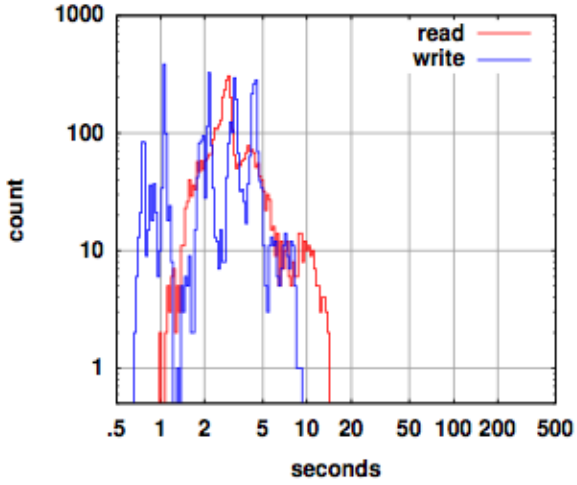
(b) Franklin aggregate I/O rate



(c) Franklin histogram



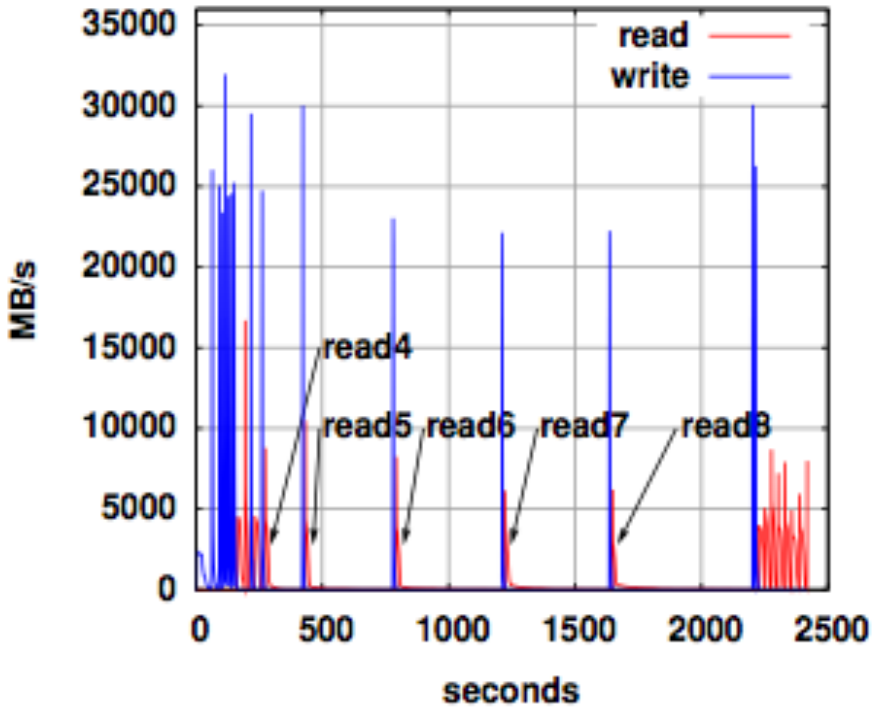
(e) Jaguar aggregate I/O rate



(f) Jaguar histogram

Problem Analysis

- Not all reads are slow
  - Just 4-8
- What special property do they have?
  - None: the reads are the same as earlier and later reads
- So, maybe something about ordering



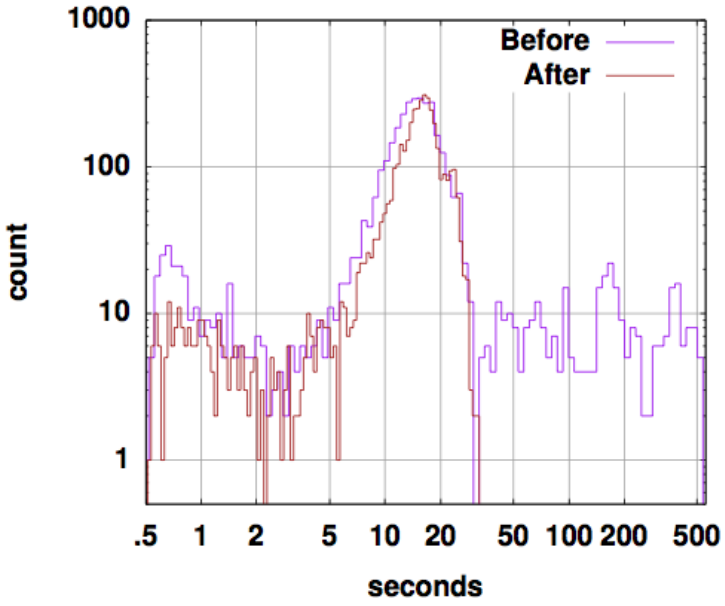
(b) Franklin aggregate I/O rate

Problem Analysis

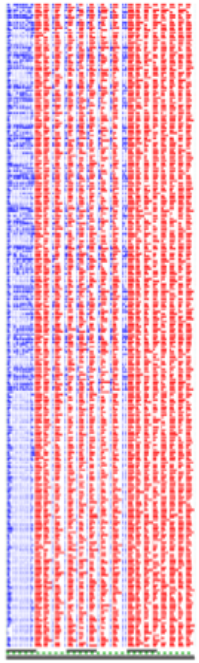
- After third read, system detects strided read pattern and performs read-ahead
  - Requires client side buffering of data
- Other uses of memory (client writes) consumed buffer space, preventing the read-ahead from working
- Lustre file system executed a fall-back code path
  - Perform small reads when no buffer space is available
  - Small reads are very inefficient

Problem Resolution

- Patch the file system
  - Turn off read-ahead in this case
- Problem solved (4x improvement)



(b) Read performance before and after middleware update



(c) Franklin trace after update

Another Code (Resolution Process)

- Reduce the number of tasks (10K -> 80) and have each task do smaller I/Os
  - Variability reduction from more small I/Os
  - Reduce resource use and contention (fewer actors)
- Align the request size to file system parameters
  - Increase transfer rate
- Defer and aggregate metadata writes
  - Avoid lots of small updates
  - < 3KB into single 1MB

Thought on MPI Performance

- Visualization tools work and matter
  - Examples of 5x to 10x differences
- I/O is a huge component of performance
  - This is only trending up
  - Memory capacity and processor speed makes more data
  - Scale requires more frequent checkpoints
- HPC is a complex and fragile ecosystem
  - Many parameters and implementation subtleties
- Order statistics rule
  - Only as fast as the slowest member
  - This gets more problematic as we use more nodes
  - HW errors and SW misconfiguations on one node can ruin a cluster. Must diagnose!