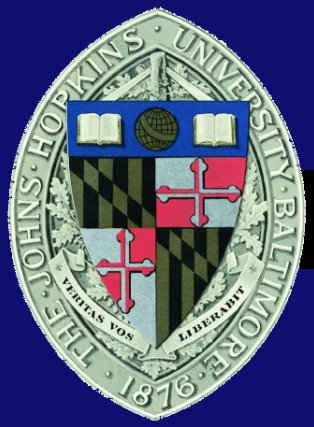
Lecture 17.2
I/O Performance and Checkpoints

EN 600.320/420

Instructor: Randal Burns

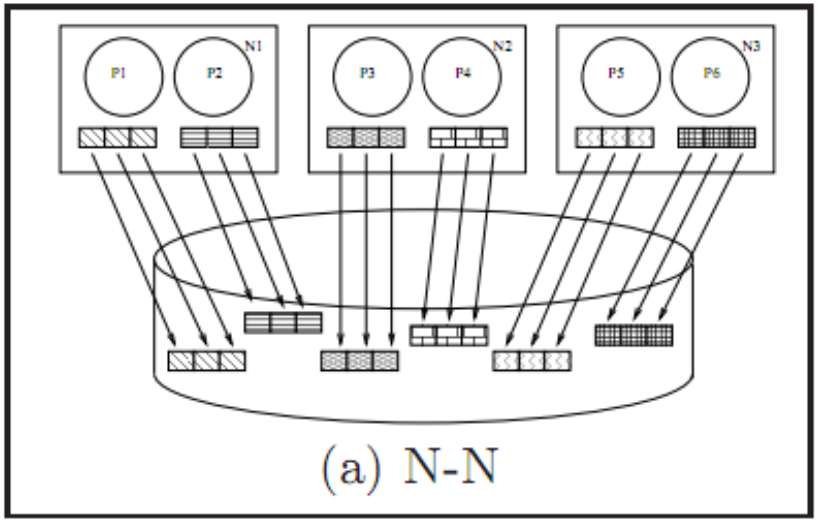
12 April 2017

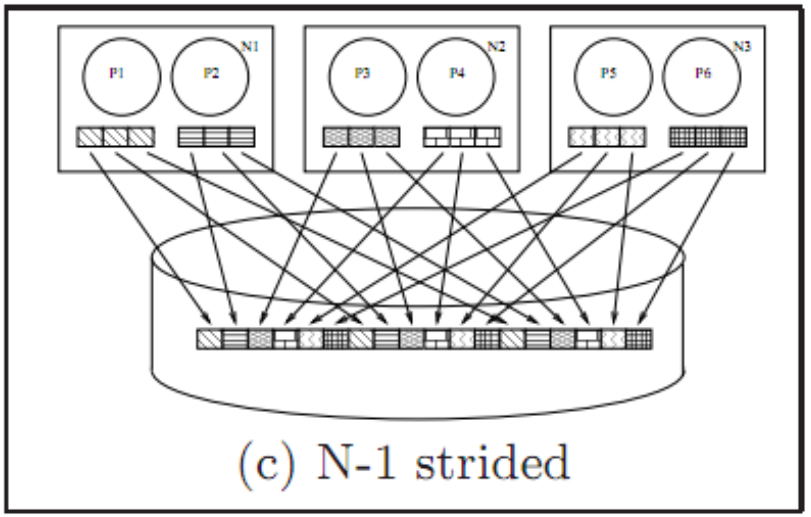


Department of Computer Science, *Johns Hopkins University*

Where does the I/O Come From?

- Checkpointing!
  - And, writing output from simulation (which is checkpointing)
- Checkpoint workload
  - Every node writes local state to a shared file system
  - Using POSIX calls (FS parallelized) or MPI I/O





J. Bent et al. PLFS: A Checkpoint File Systems for Parallel Applications. SC, 2009.

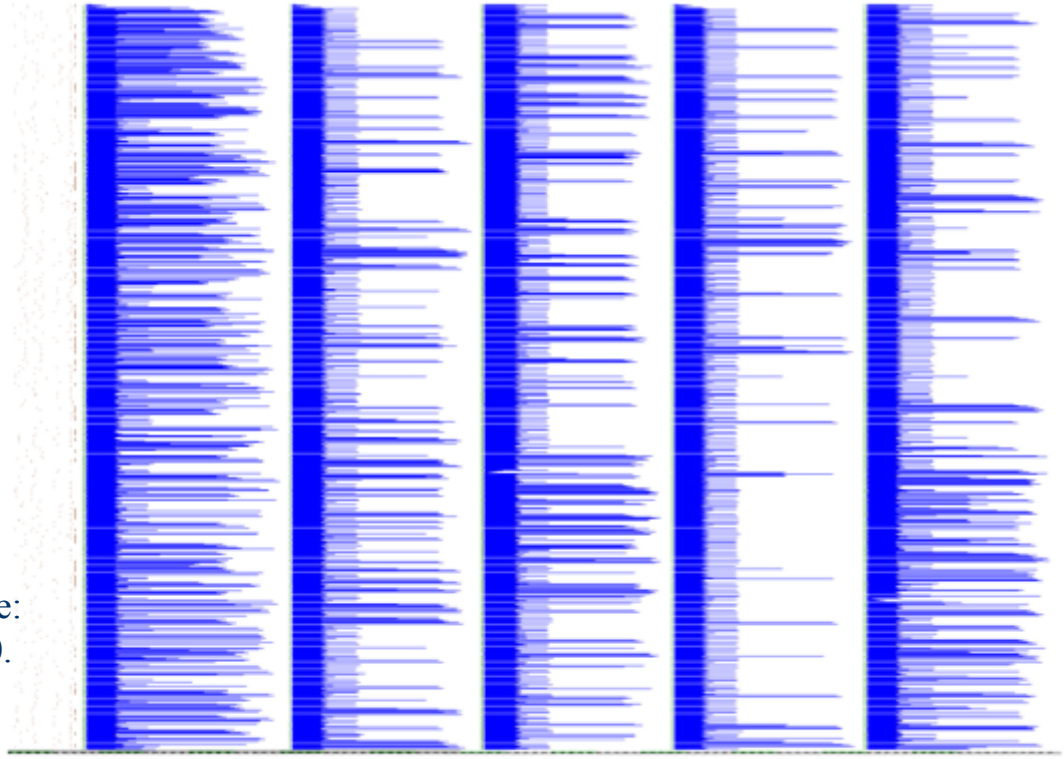
Why Checkpointing

- At scale failures occur inevitably
  - MPI synchronous model means that a failure breaks the code
  - Lose all work since start (or restart)
- Each checkpoint provides a restart point
  - Limits exposure, loss of work to last checkpoint
- Store enough state to restart computation
  - Automatic: store contents of memory and program counters
    - Brute force, large data, inefficient, but easy
  - Application specific: keep data structures and metadata representing current progress. Hand coded by developer.
    - Smaller, faster, preferred, but tedious.

A Checkpoint Workload

- IOR benchmark
  - Each node transfers 512 MB
- Barriers

- *How much parallelism?*
- *What effects?*

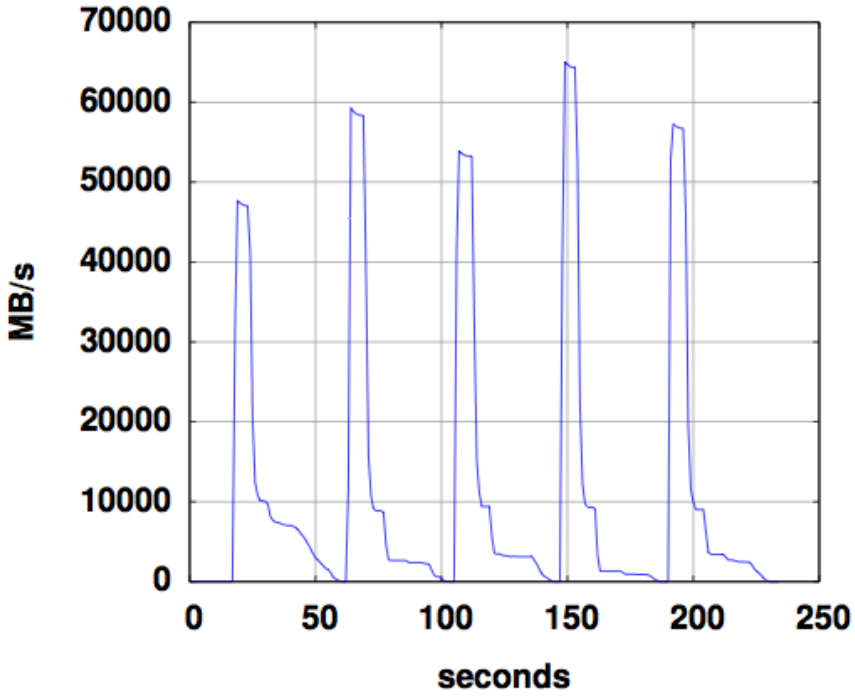


(a) I/O trace diagram

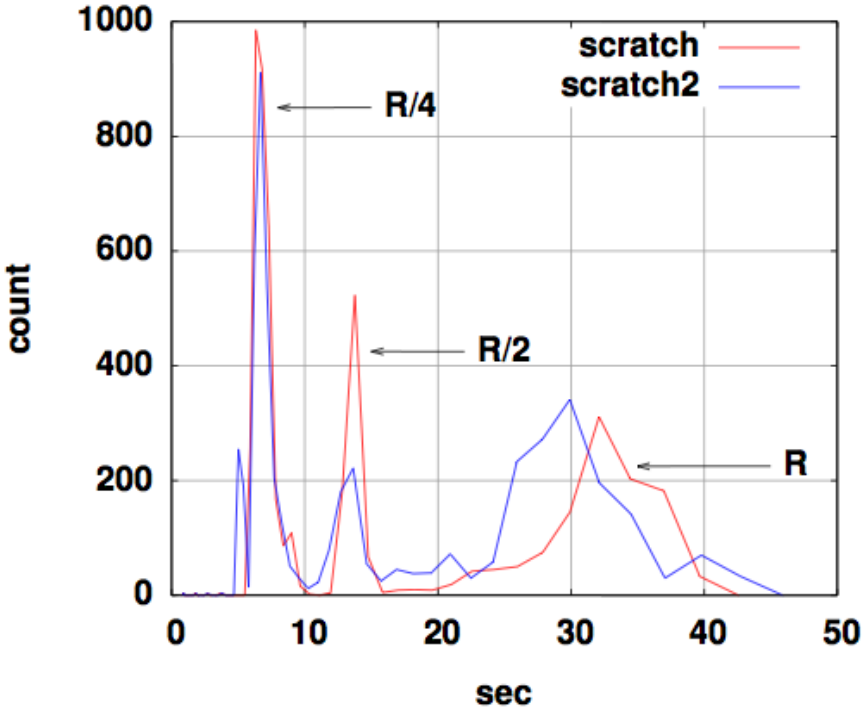
M. Uselton et al. Parallel I/O Performance: From Events to Ensembles. IPDPS, 2010.

I/O Rates and PDF

- *What features do you observe?*



(b) Aggregate I/O rate

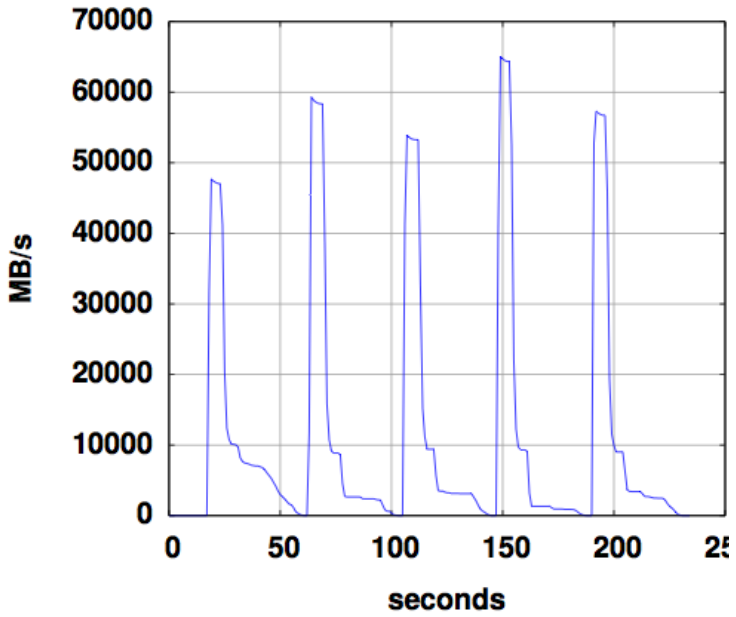


(c) I/O time distribution

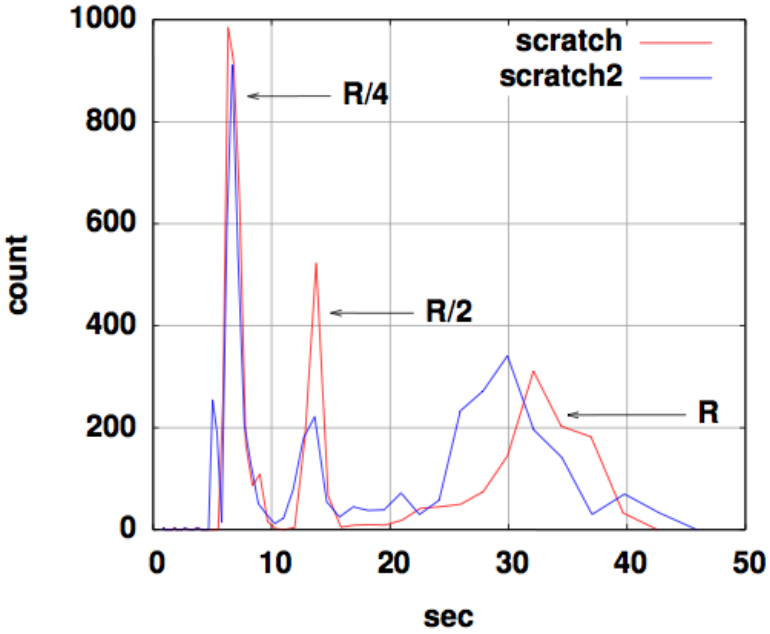
M. Uselton et al. Parallel I/O Performance: From Events to Ensembles. IPDPS, 2010.

I/O Rates and PDF

- *What features do you observe?*
  - Lagging processes = not realizing peak I/O performance
  - Harmonics in I/O distribution = unfair resource sharing



(b) Aggregate I/O rate



(c) I/O time distribution

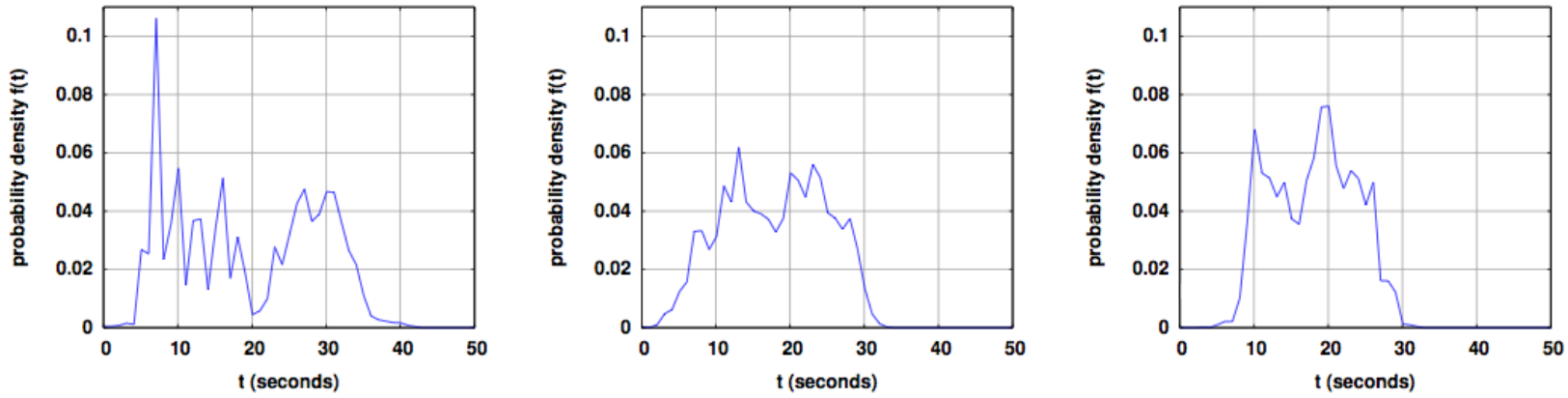
M. Uselton et al. Parallel I/O Performance: From Events to Ensembles. IPDPS, 2010.

Statistical Observations

- Order statistics
  - Fancy way of saying, the longest operation dominates overall performance
  - This should be intuitive based on IPM graph
- Law of large numbers
  - I don't think that they make this analysis cogent
  - It's right, but Gaussian distribution is not what matters
  - A better, intuitive conclusion is
- (RB) smaller files are better
  - The worst case slow down on a smaller transfer takes less time than on a large transfer
  - Assuming there is some minimum bandwidth

Smaller Files Improve Performance

- Non-intuitive
  - Smaller operations seems like more overhead
  - But, a property of statistical analysis
- Smaller better as long as fixed costs are amortized
  - Obviously, 1 byte is too small



(a) two 256 MB transfers (b) four 128 MB buffers (c) eight 64 MB transfers

Figure 2: IOR 512 MB transfer using 1024 processors where: a) 512 MB written via two 256MB `write()` calls. b) Four calls (128MB). c) Eight calls (64MB). Note that the distributions become progressively narrower and more Gaussian.