Performance Portability for Burst Buffers with the Scalable Checkpoint/Restart Library (SCR)

DOE COE Performance Portability Meeting

Kathryn Mohror and Michael Pozulp

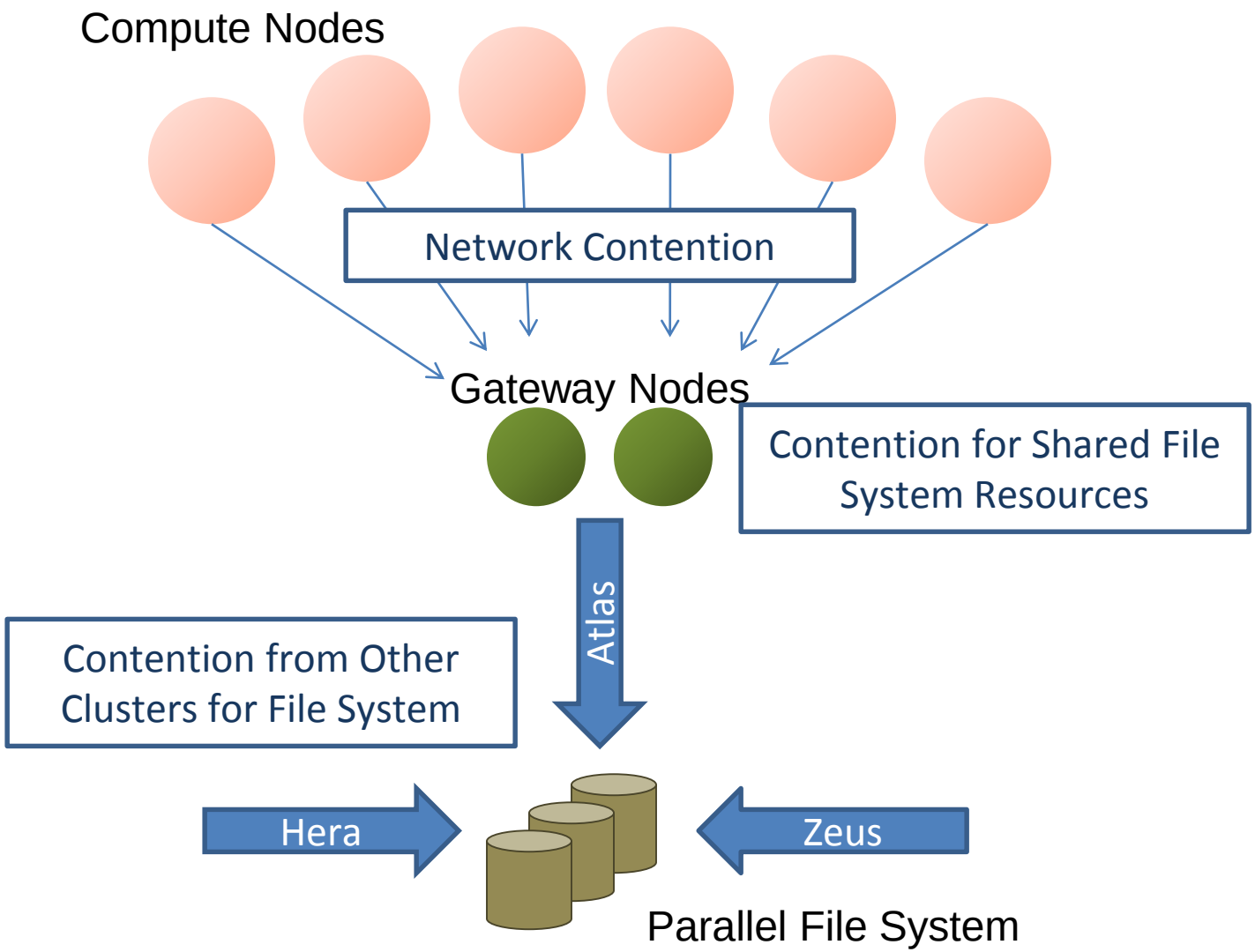
April 21, 2016

LLNL-PRES-689447

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

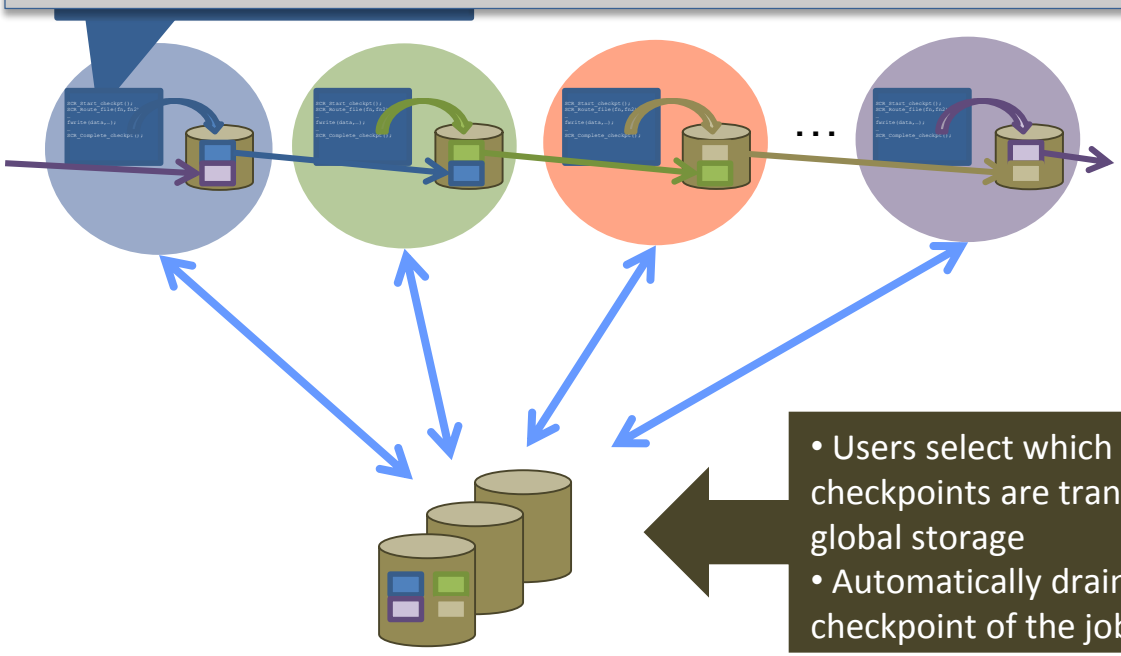
Lawrence Livermore National Laboratory

Writing checkpoints to the parallel file system is very expensive



The Scalable Checkpoint/Restart Library (SCR) is a production-ready implementation of multilevel checkpointing

- Multilevel checkpointing
  - Use different levels of storage for checkpoints, e.g. node local storage
- Treat checkpoints as “special” files
  - By nature temporary, cache most on node local storage, throw them away
  - Typically failures affect a single node at a time

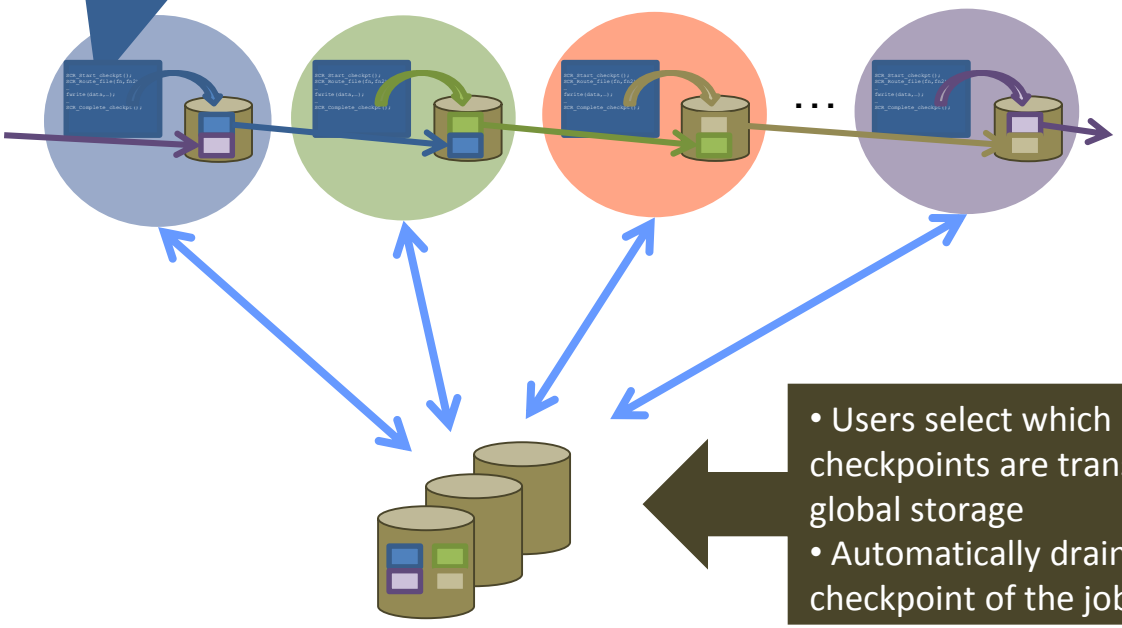


- First write checkpoints to node-local storage
- When checkpoint is complete, apply redundancy schemes

- Users select which checkpoints are transferred to global storage
- Automatically drain last checkpoint of the job

The Scalable Checkpoint/Restart Library (SCR) is a production-ready implementation of multilevel checkpointing

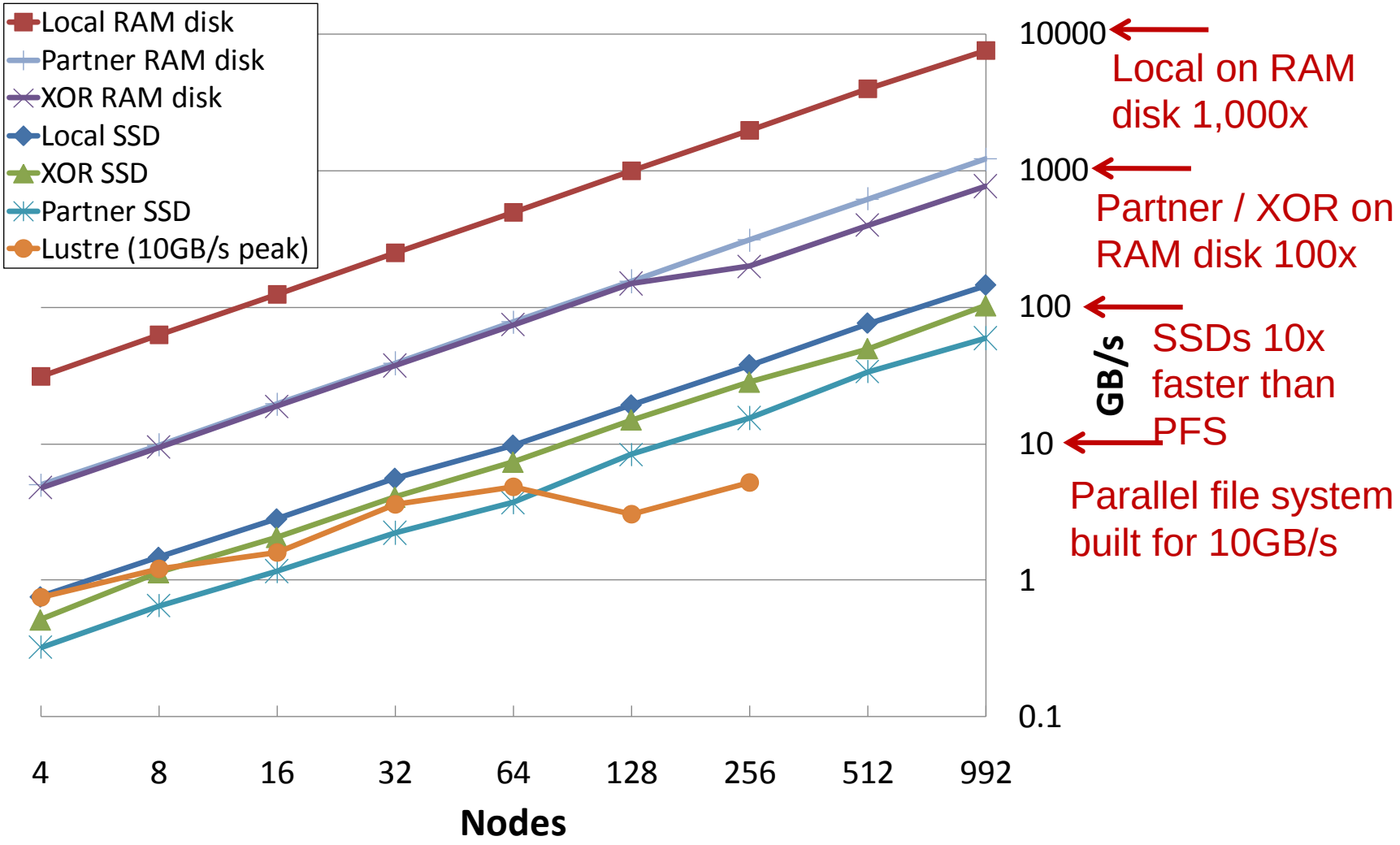
```
SCR_Start_checkpt();
SCR_Route_file(fn,fn2);
…
fwrite(data,…);
…
SCR_Complete_checkpt();
```



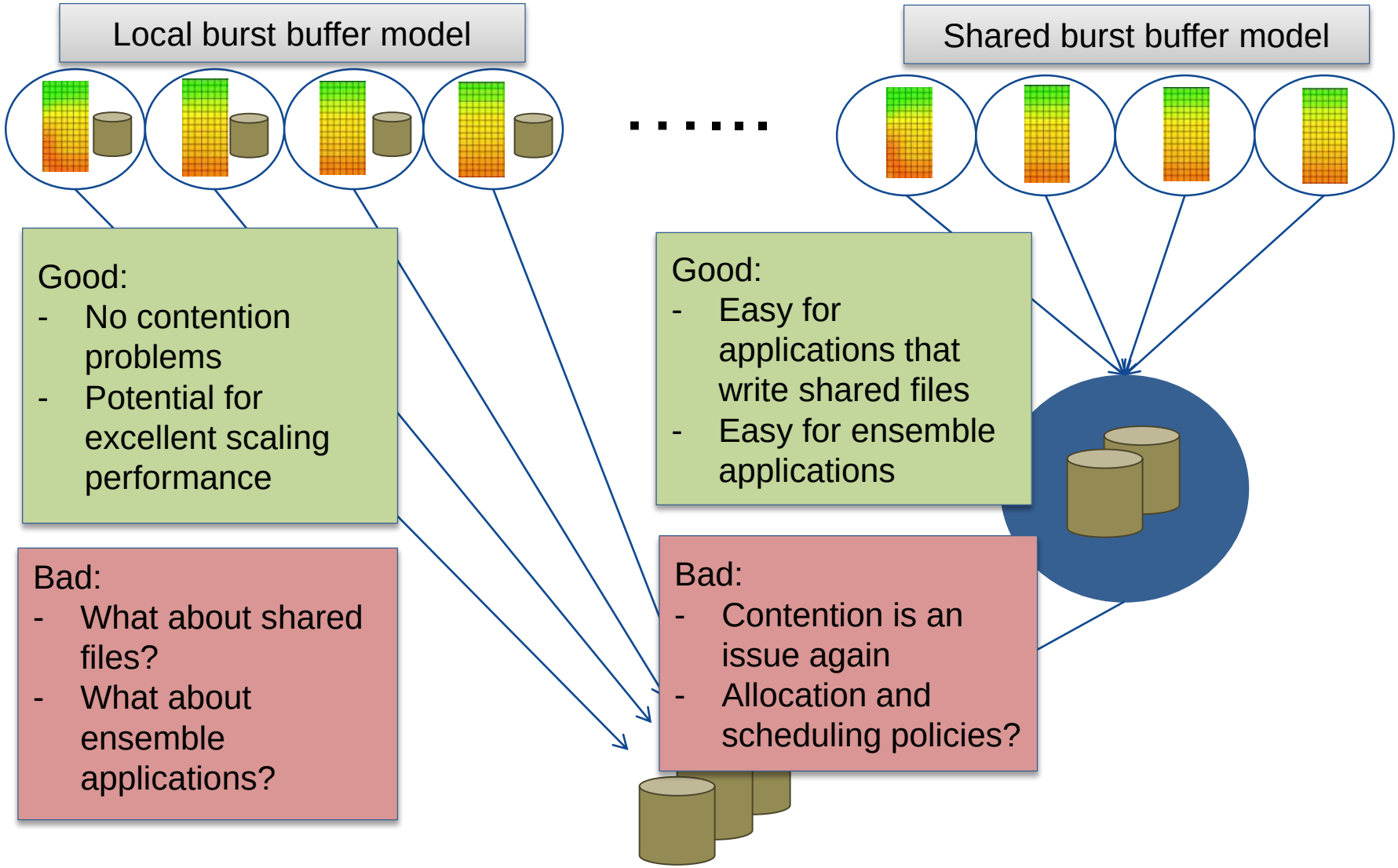
- First write checkpoints to node-local storage
- When checkpoint is complete, apply redundancy schemes

- Users select which checkpoints are transferred to global storage
- Automatically drain last checkpoint of the job

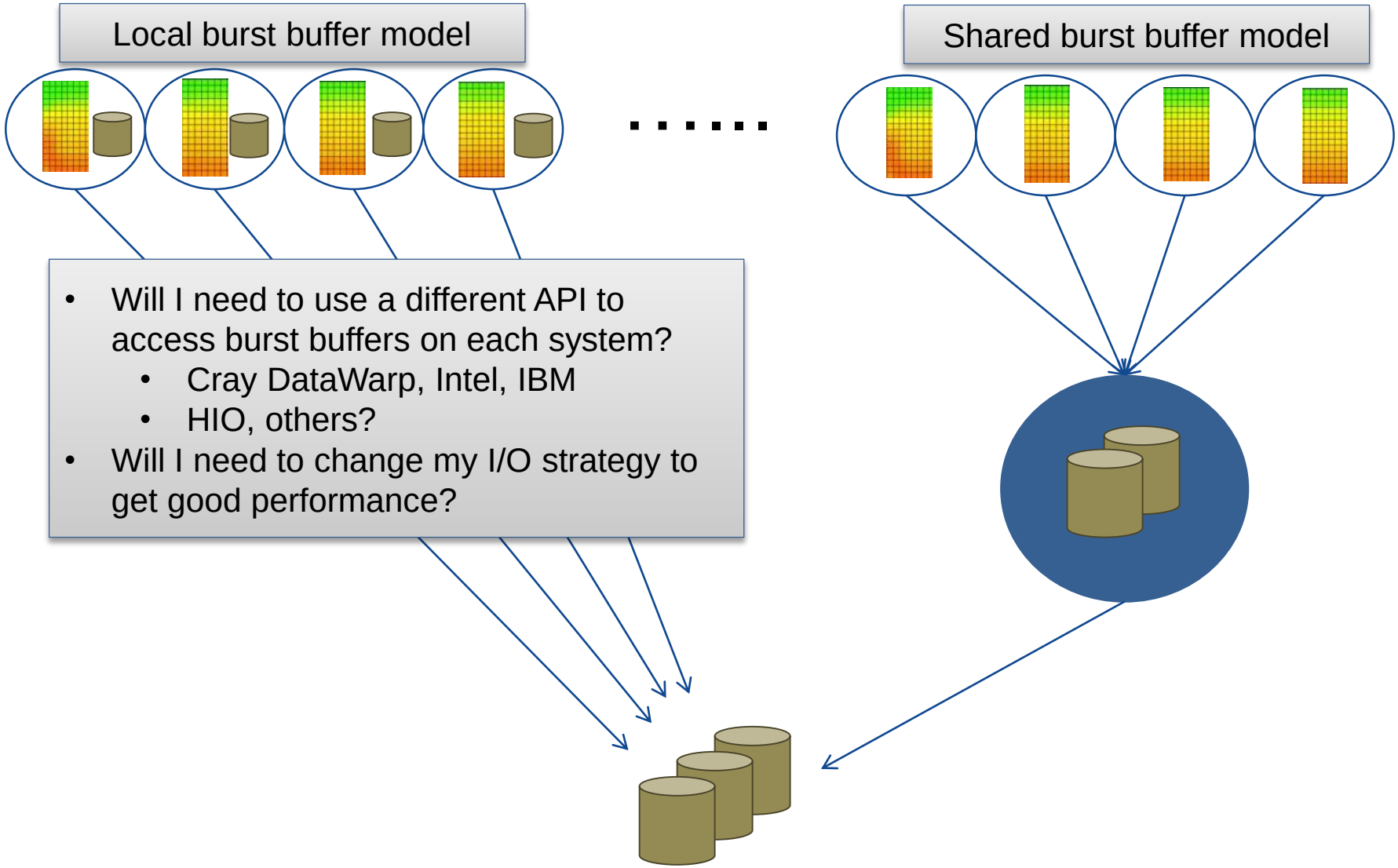
Aggregate checkpoint bandwidth to node-local storage scales linearly on Coastal



Burst buffers are a new layer in the storage hierarchy to buffer the bursty I/O operations of HPC applications



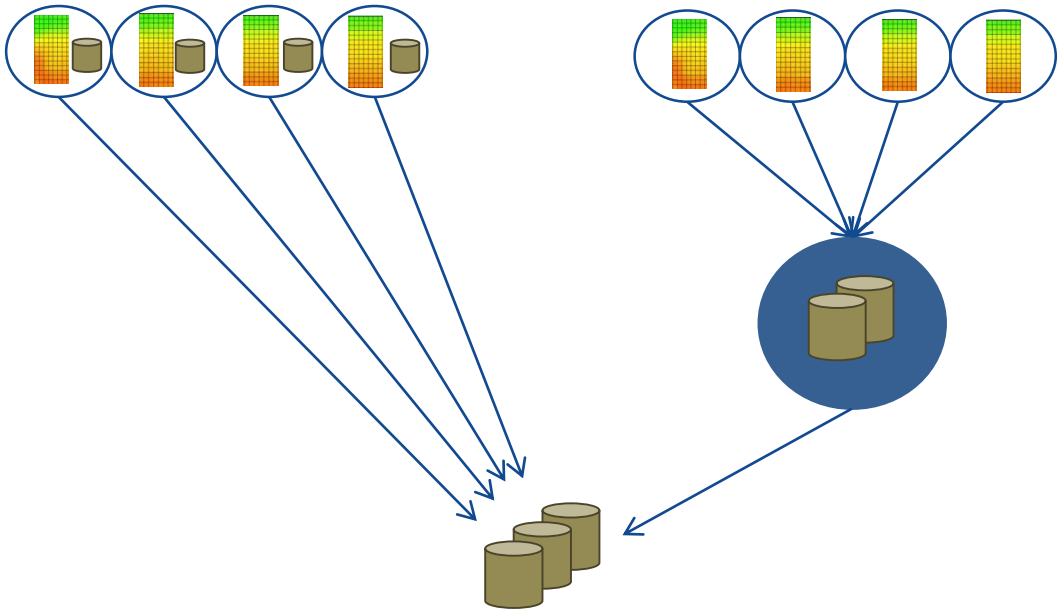
How can applications achieve portable I/O performance across burst buffer implementations?



- Will I need to use a different API to access burst buffers on each system?
  - Cray DataWarp, Intel, IBM
  - HIO, others?
- Will I need to change my I/O strategy to get good performance?

SCR can abstract away the differences in hierarchical storage systems

- Goal is for applications to integrate SCR, and never have to worry about the underlying storage system
  - Integrate SCR versus altering application for each new system
  - Once you integrate SCR, SCR will do the Right Thing™ with your checkpoints



Using SCR for Multilevel Checkpointing in Mercury's Trinity Open Science Project

DOE COE Performance Portability Meeting

LLNL-PRES-689447

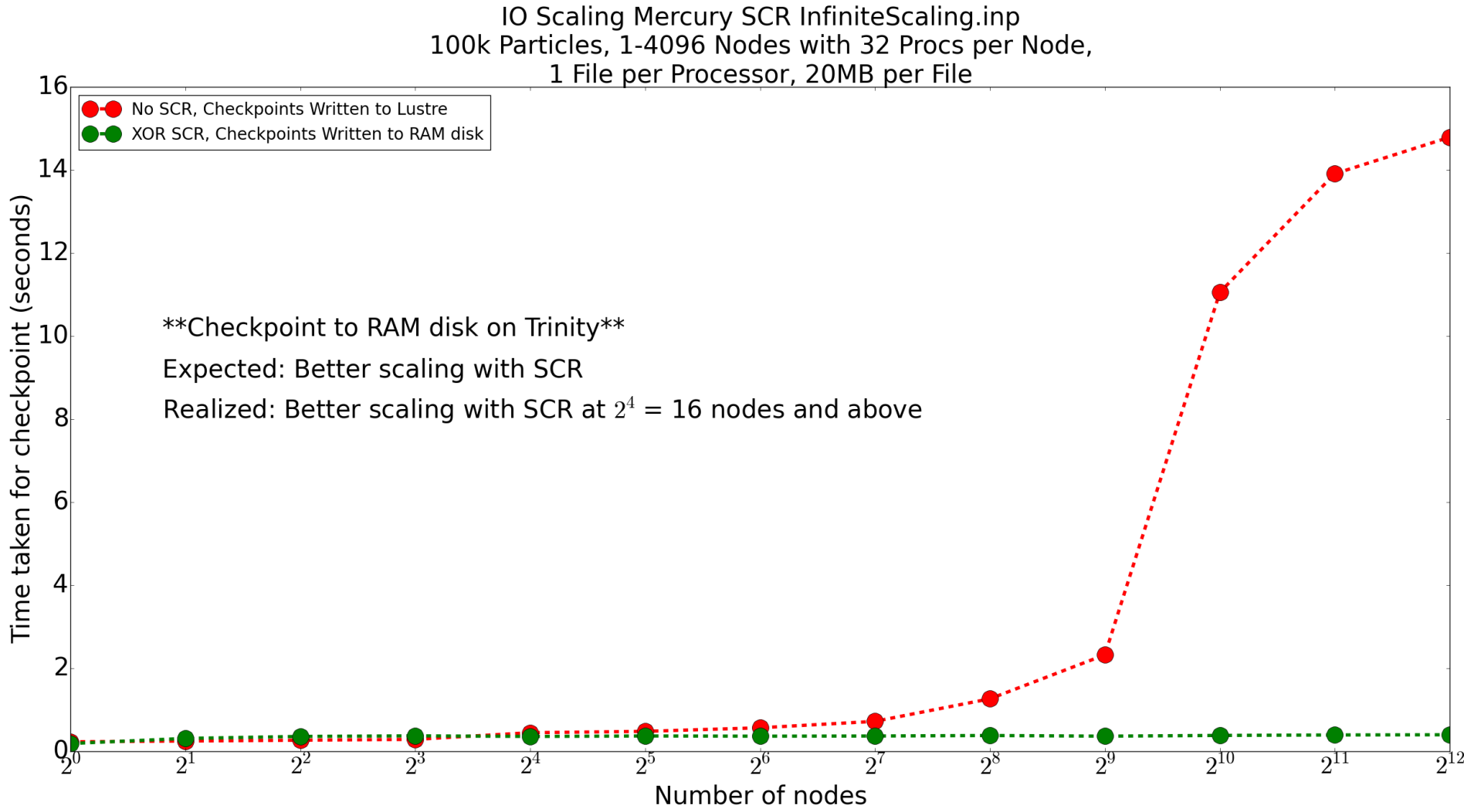
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

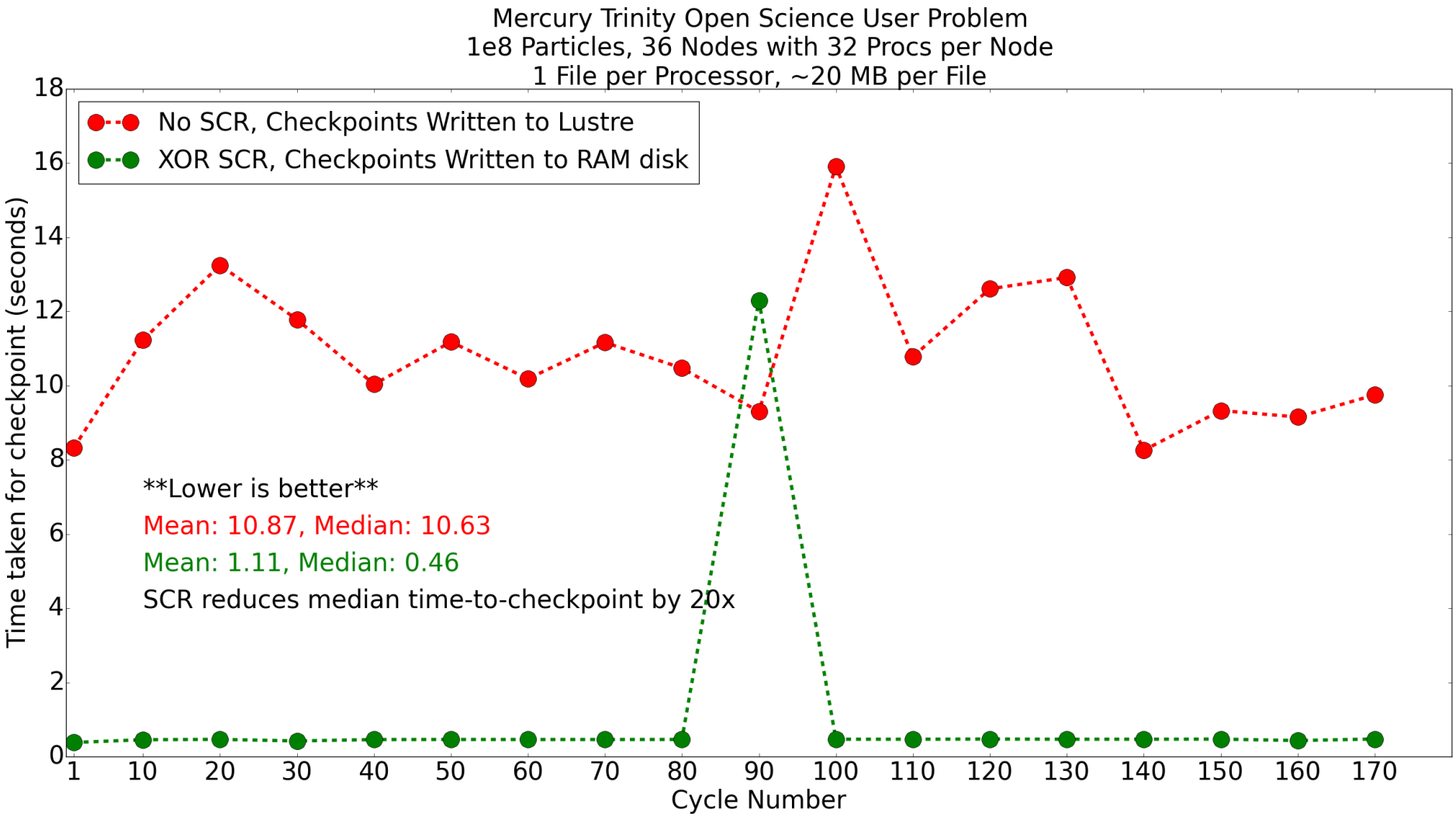
Mercury
Monte Carlo Particle Transport

- Mercury + SCR selected for Trinity Open Science
- Weak scaling problem (I/O per processor is constant)
  - Checkpoint once to RAM disk vs. Lustre
  - Scale from 1 to 4096 nodes in powers of 2
  - 30X maximum speedup at 4096 nodes
- User problem
  - Checkpoint every 10 cycles to RAM disk vs. Lustre
  - Run on 36 nodes
  - 20X reduction in median time-to-checkpoint

Mercury Checkpoint Scaling Study



Mercury User Problem



Conclusion for Mercury + SCR

- Observed worthwhile speedups for checkpointing with SCR
- Interested in replicating scaling study on Trinity's Burst Buffers
- A note on how we integrated SCR into Mercury
  - Used SCR SC'14 and SC'15 Tutorial Slides, SCR User's Guide
  - Checkpoint was as simple as substituting a file system path using SCR_Route_File() API call
  - Restart was harder, but not complicated
    - Cannot assume that all ranks can read every file that was written during a checkpoint

Questions?

- Contact the SCR team
  - http://computation.llnl.gov/projects/scalable-checkpoint-restart-for-mpi
    - Project description
    - User's guide
    - Links to software
  - scalablecr-discuss@lists.sourceforge.net

Lawrence Livermore
National Laboratory