panorama360

On the Use of Burst Buffers for Accelerating Data-Intensive Scientific Workflows

Rafael Ferreira da Silva, Scott Callaghan, Ewa Deelman

12th Workflows in Support of Large-Scale Science (WORKS) – SuperComputing'17
November 13, 2017

Funded by the US Department of Energy under Grant #DE-SC0012636

U.S. DEPARTMENT OF ENERGY | Office of Science

USC Viterbi
School of Engineering
Information Sciences Institute

OUTLINE

Introduction

Next-generation Supercomputers
Data-intensive Workflows
In-transit / In-situ

Burst Buffers

Overview
Node-local / Remote-shared
NERSC BB

Model and Design

BB Reservations
Execution Directory
I/O Read Operations

Workflow Application

CyberShake
Workflow Implementation
I/O Performance: Darshan

Experimental Results

Overall Write/Read Operations
I/O Performance per Process
Cumulative CPU Time
Rupture Files

Conclusion

Summary of Findings
Future Directions

A Brief Introduction and Contextualization

Traditionally,

Workflows have used the file system to communicate data between tasks

To cope with increasing application demands on I/O operations, solutions targeting ***in situ*** and/or ***in transit*** processing have become mainstream approaches to attenuate I/O performance bottlenecks.

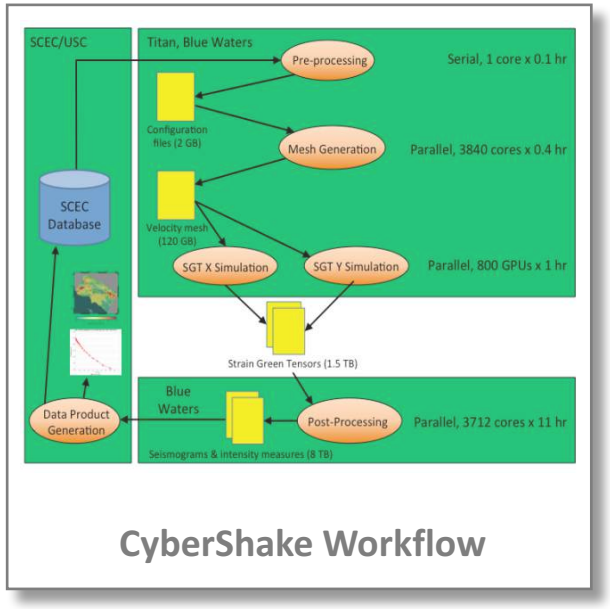
Next-generation of Exascale Supercomputers

Increased processing capabilities to over **10^{18} Flop/s**

Memory and **disk** capacity will also be significantly increased
Power consumption management

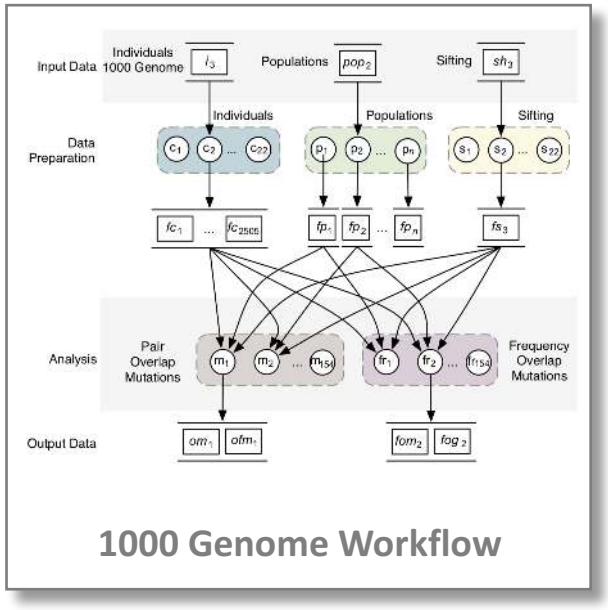
I/O performance of the parallel file system (PFS) is not expected to improve much

Data-Intensive Scientific Workflows



CyberShake Workflow

*Consumers/produces over **700GB** of data*



1000 Genome Workflow

*Consumers/produces over **4.4TB** of data, and requires over **24TB** of memory across all tasks*

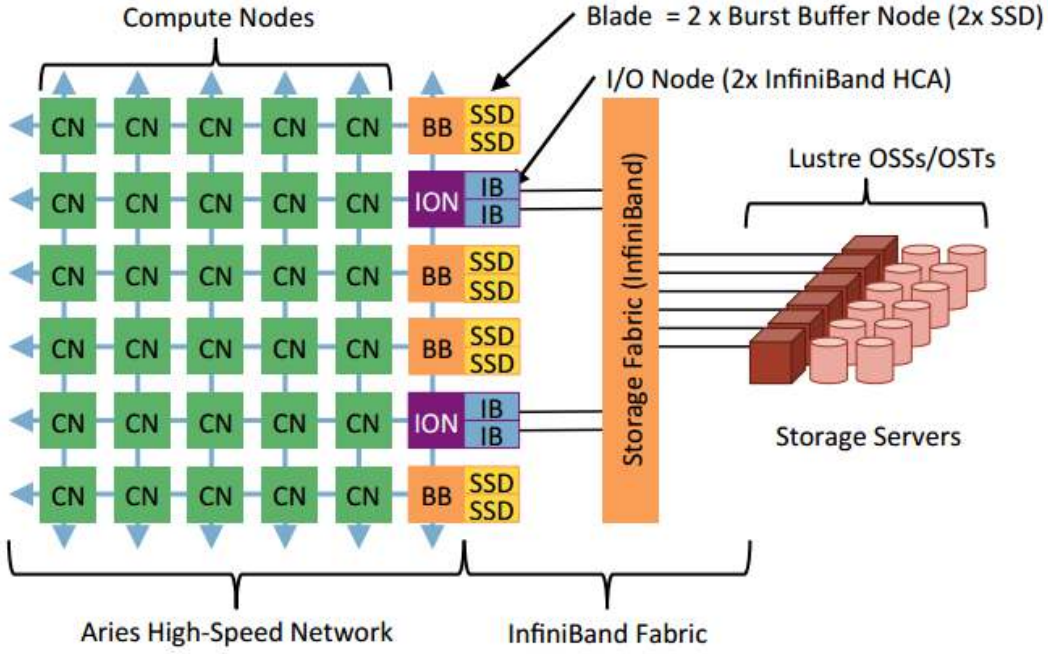
*While **in situ** is well adapted for computations that conform with the **data distribution** imposed by simulations, **in transit** processing targets applications where **intensive data transfers** are required*

Improving I/O Performance with Burst Buffers

In Transit Processing

Burst buffers have emerged as a **non-volatile storage** solution that is positioned between the processors' memory and the PFS, buffering the large volume of data produced by the application at an **higher rate than the PFS**, while seamlessly draining the data to the PFS asynchronously.

> *A burst buffer consists of the combination of rapidly accessed persistent memory with its own processing power (e.g., DRAM), and a block of symmetric multi-processor compute accessible through high-bandwidth links (e.g., PCI Express)*



Placement of the burst buffer nodes within the **Cori** system (**NERSC**)

Node-local vs Remote-shared

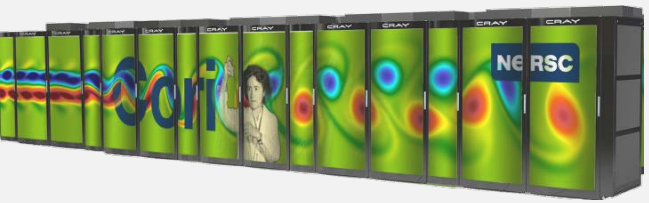
First Burst Buffers Use at Scale

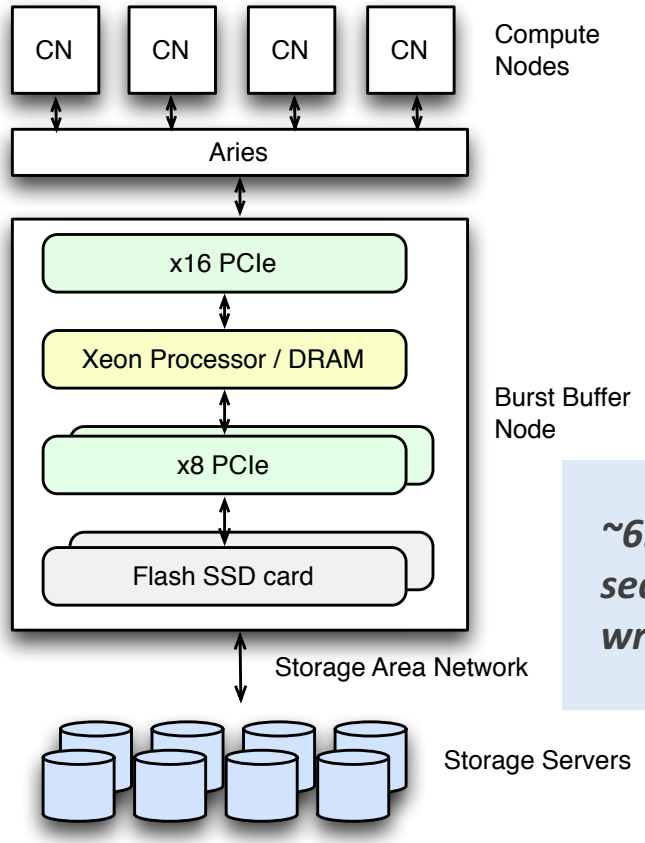
Cori System (NERSC)

Cori is a petascale HPC system and #6 on the June 2017 Top500 list

NERSC BB is based on Cray DataWarp
(Cray's implementation of the BB concept)

*Each BB node contains a **Xeon** processor, **64 GB of DDR3** memory, and **two 3.2 TB NAND** ash SSD modules attached over two **PCIe gen3 x8** interfaces, which is attached to a Cray Aries network interconnect over a PCIe gen3 x16 interface*





~6.5 GB/sec of sequential read and write bandwidth

Architectural overview of a burst-buffer node on Cori at **NERSC**

Model and Design: Enabling BB in Workflow Systems

Automated BB Reservations

BB reservations operations (either persistent or scratch) consist in the **creation** and **release**, as well as **stage in** and **stage out** operations

Transient reservations: needs to implement stage in/out operations at the beginning/end of each job execution

I/O Read Operations

Read operations from the BB should be transparent to the applications

*Approaches: point the execution directory to the BB reservation, or create **symbolic links** to data endpoints into the BB*

Execution Directory

Automated **mapping** between the workflow **execution directory** and the BB reservation

No changes to the application code are necessary, and the application job directly writes its output to the BB reservation

Workflow Application: CyberShake Workflow

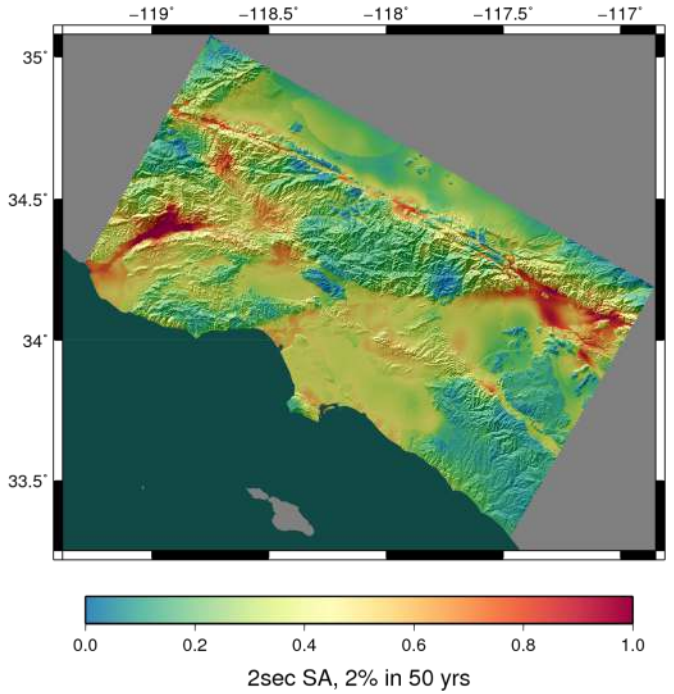
CyberShake Workflow

CyberShake is a high-performance computing software that uses 3D waveform modeling to calculate PSHA estimates for populated areas of California

*Constructs and Populates a 3D mesh of **~1.2 billion elements** with seismic velocity data to compute Strain Green Tensors (SGTs)*

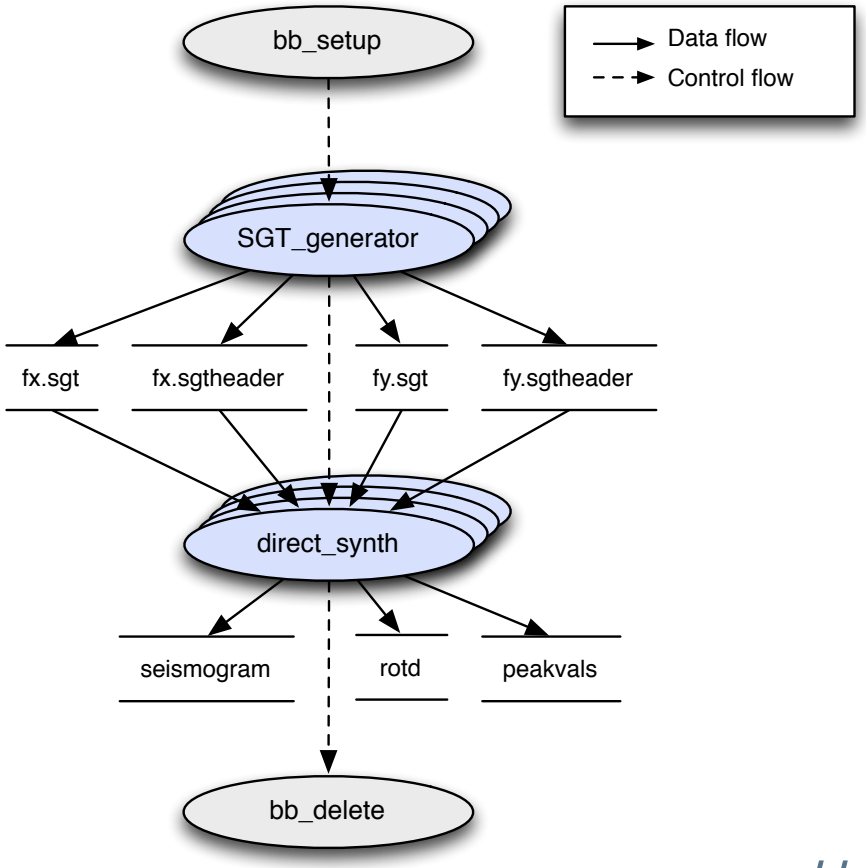
Post-processing: SGTs are convolved with slip time histories for each of about **500,000 different earthquakes** to generate synthetic seismograms for each event

We focus on the two CyberShake job types which together account for **97% of the compute time**: the wave propagation code **AWP-ODC-SGT**, and the post-processing code **DirectSynth**



CyberShake hazard map for Southern California, showing the spectral accelerations at a 2-second period exceeded with a probability of 2% in 50 years

Burst Buffers: Workflow Implementation



A general representation of the CyberShake test workflow

Pegasus WMS

Workflow is composed of two **tightly-coupled parallel** jobs (SGT_generator; and direct_synth), and two system jobs (**bb_setup** and **bb_delete**)

Generates/consumes about **550GB** of data

https://github.com/rafaelfsilva/bb-workflow

```
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -C haswell
#SBATCH -t 00:05:00
#BB create_persistent name=csbb capacity=700GB access=striped type=scratch
```

bb_setup job

Goes through the regular queuing processing

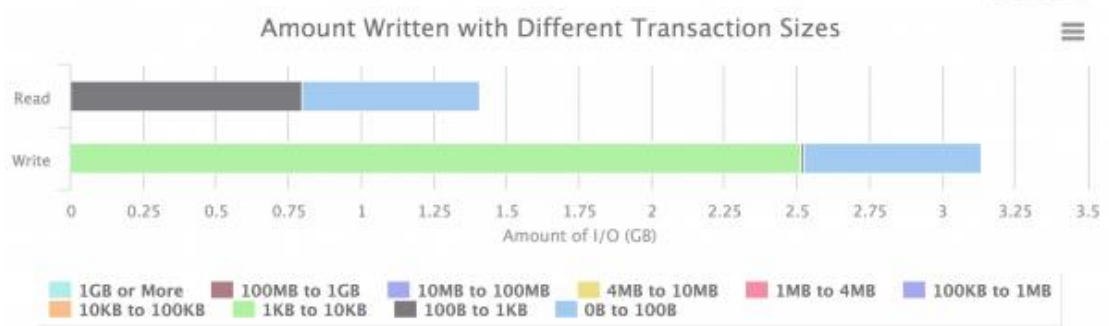
```
#SBATCH -p regular
#SBATCH -N 64
#SBATCH -C haswell
#SBATCH -t 05:00:00
#DW persistentdw name=csbb
```

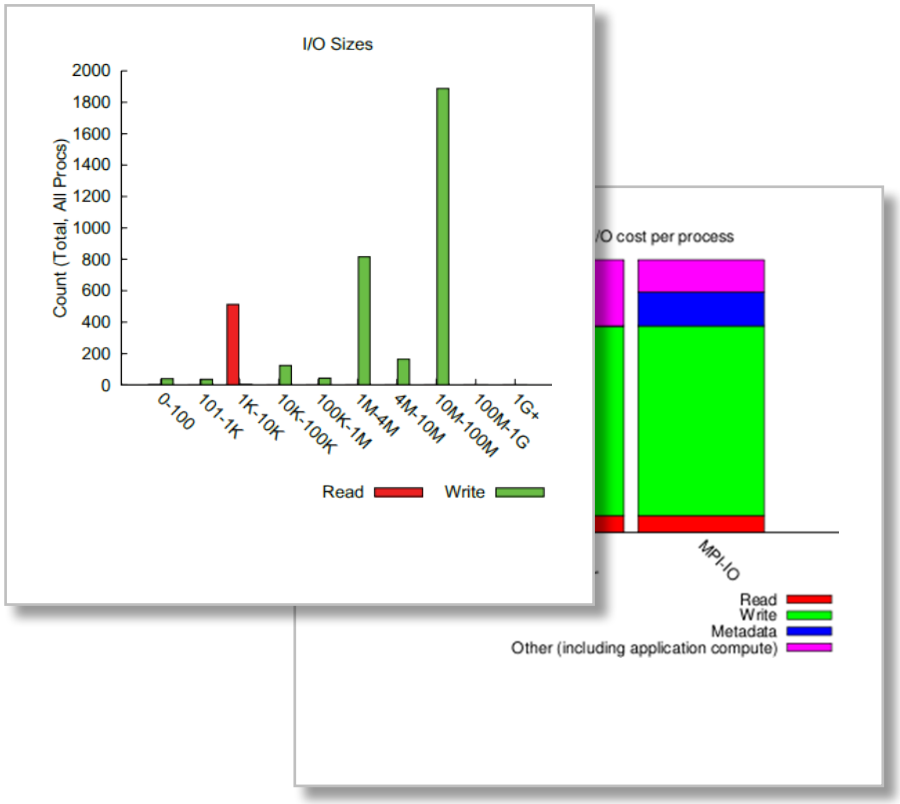
Collecting I/O Performance Data with Darshan

Darshan: HPC I/O Characterization Tool

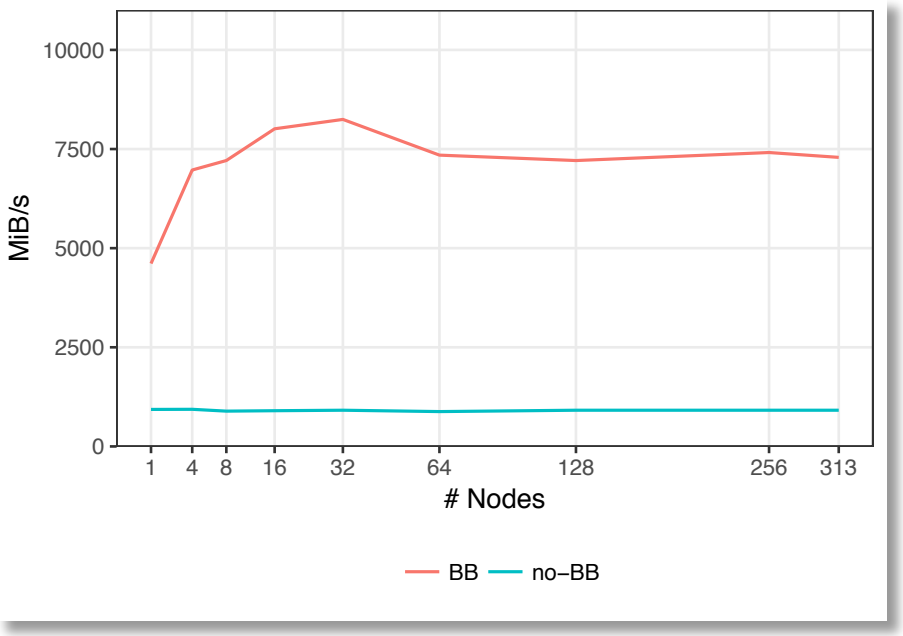
HPC lightweight **I/O profiling** tool that captures an accurate picture of I/O behavior (including **POSIX IO**, **MPI-IO**, and HDF5 IO) in MPI applications

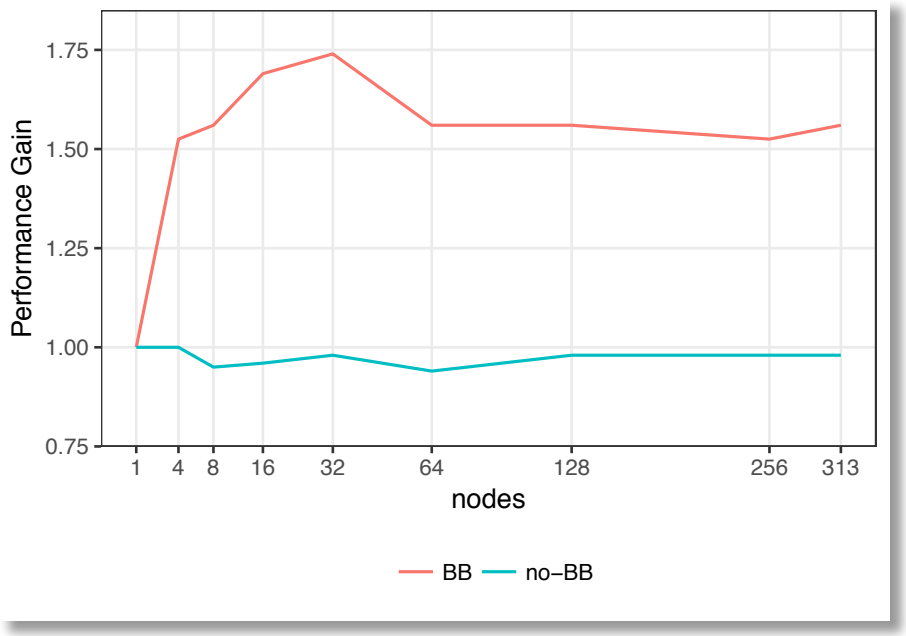
Darshan is part of the default software stack on Cori





Experimental Results: Overall Write Operations

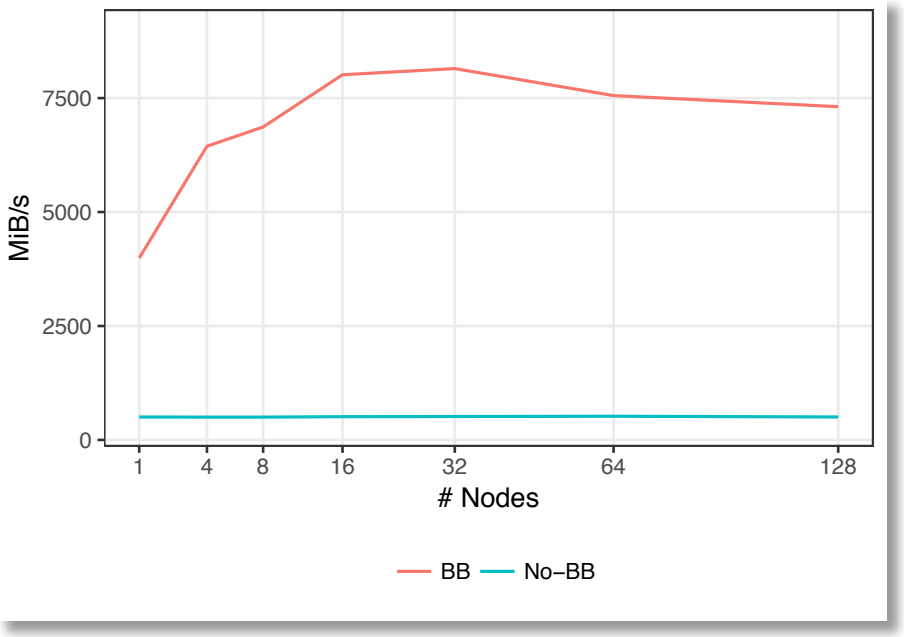


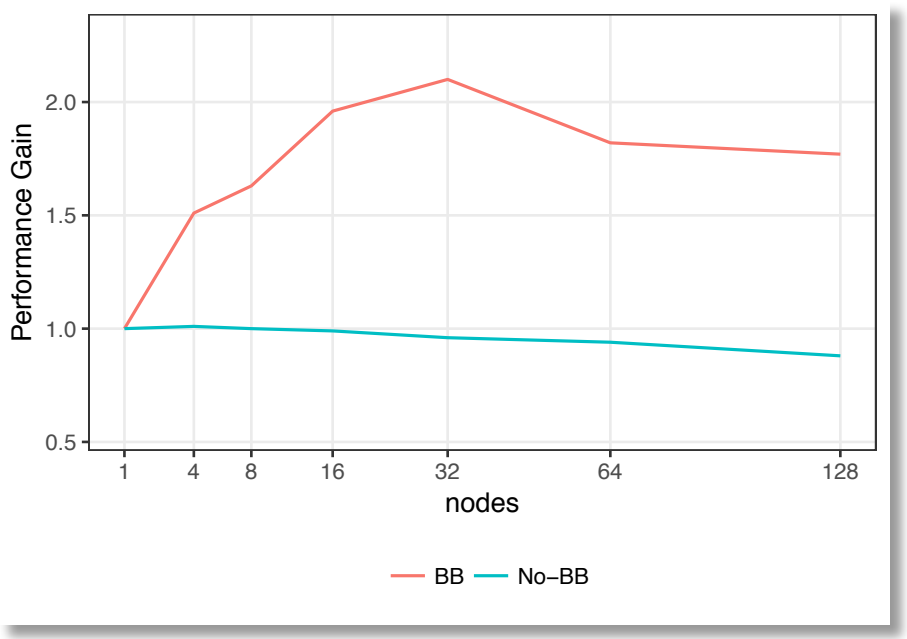


Average I/O performance estimate for write operations at the MPI-IO layer (left), and average I/O write performance gain (right) for the SGT_generator job

- Overall, write operations to the PFS (**No-BB**) have nearly **constant** I/O performance
- No-BB: **~900 MiB/s** regardless of the number of nodes used
- Base values obtained for the **BB executions** (1 node, 32 cores) are over **4,600 MiB/s**, and peak values scale up to **~8, 200 MiB/s** for 32 nodes (1,024 cores)
- Slight **drop** in the I/O performance (#nodes ≥ 64)
  large number of concurrent *write operations*

Experimental Results: Overall Read Operations



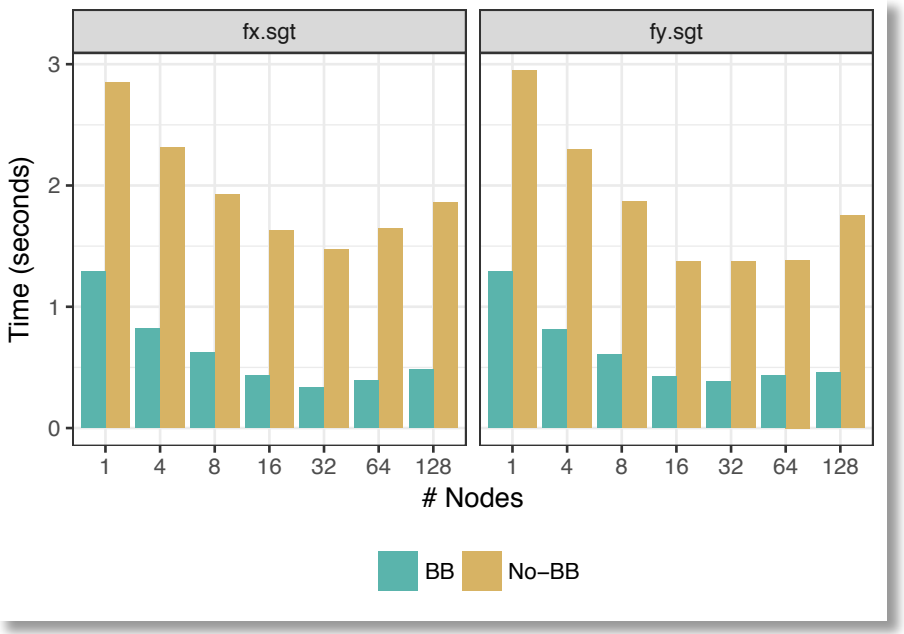


I/O performance estimate for read operations at the MPI-IO layer (left), and average I/O write performance gain (right) for the direct_synth job

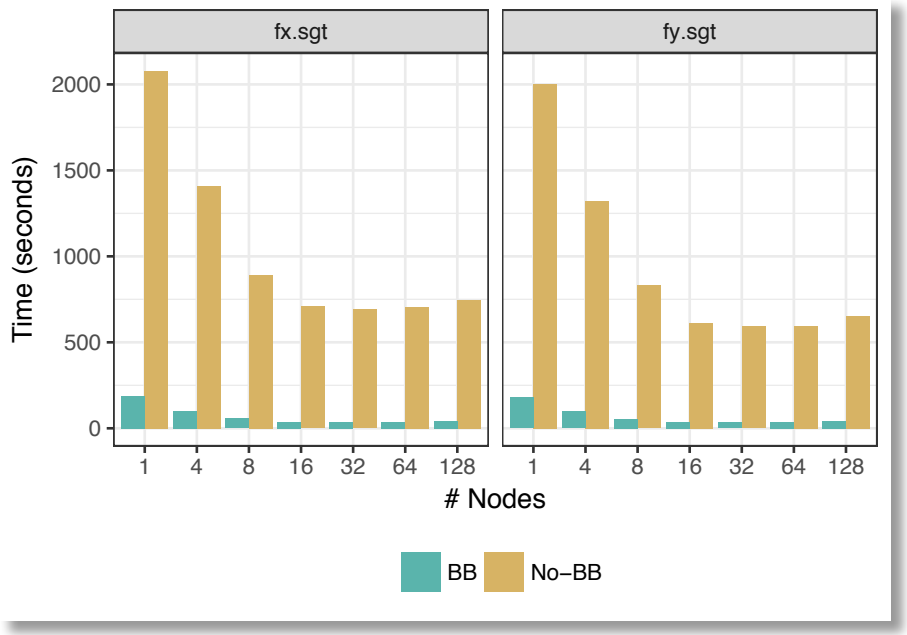
- I/O read operations from the **PFS** yield similar performance regardless of the number of nodes used: **~500 MiB/s**
- **BB**: single-node performance of **4,000 MiB/s**, peak values up to about **8,000 MiB/s**

- Small drop in the performance for runs using 64 nodes or above – may indicate an **I/O bottleneck** when draining the data to/from the underlying parallel file system

Experimental Results: I/O Performance per Process



POSIX module data: Average time consumed in I/O read operations per process for the direct_synth job

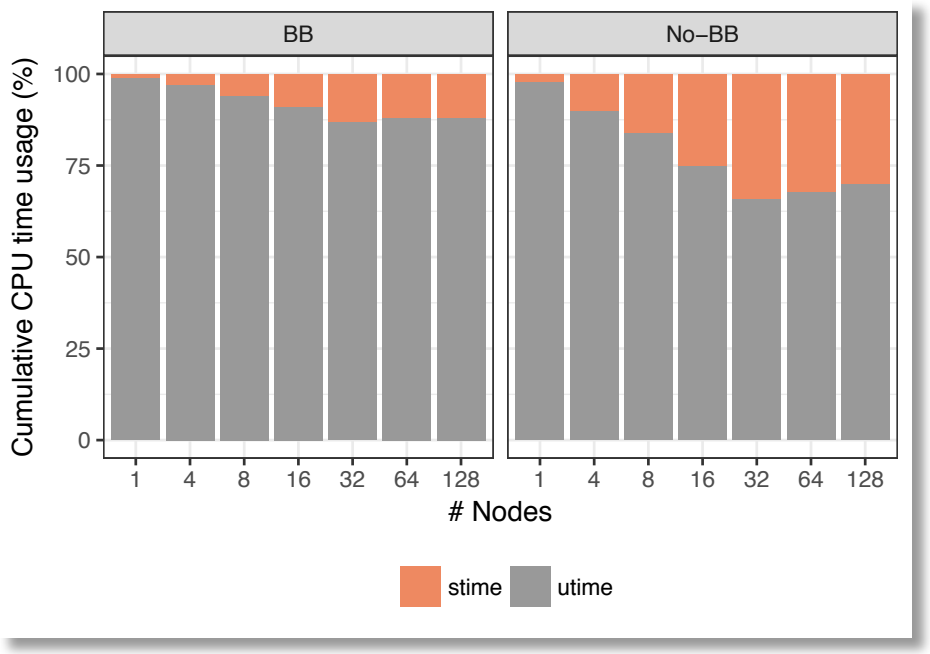


MPI-IO module data: Average time consumed in I/O read operations per process for the direct_synth job

- **POSIX operations** (left) represent buffering and synchronization operations with the system
- POSIX values are negligible when compared to the job's total runtime (~8h for 64 nodes)

- MPI-IO: **BB accelerates** I/O read operations **up to 10 times** in average
- for larger configurations (≥ 32 node), the average time is nearly the same as when running with 16 nodes for the No-BB

Experimental Results: Cumulative CPU Time



Ratio between the cumulative time spent in the user (utime) and kernel (stime) spaces for the direct_synth job, for different numbers of nodes

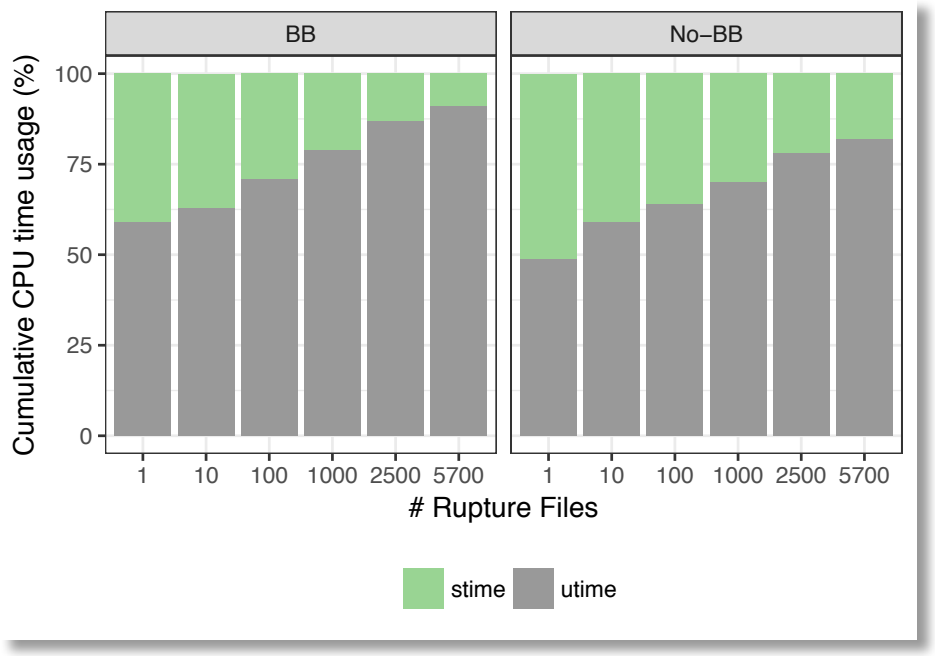
- Averaged values (for up to thousands of cores) may **mask slower processes**

  *In some cases, e.g. 64 nodes, slowest time consumed in I/O read operations can **slowdown** the application **up to 12 times** the averaged value*

- Ratio between the time spent in the user (*utime*) and kernel (*stime*) spaces – handling I/O-related interruptions, etc.

- Performance at **64 nodes** is **similar** to **32 nodes**, suggesting **gains in** application **parallel efficiency** would **outweigh a slight I/O performance hit** at 64 nodes and lead to decreased overall runtime

Experimental Results: Rupture Files



Ratio between the cumulative time spent in the user (utime) and kernel (stime) spaces for the direct_synth job for different numbers of rupture files (workflow runs with **64 nodes**)

- A typical execution of the CyberShake workflow for a selected site in our experiment processes about **5,700 rupture files**
- The processing of rupture files drive most of the CPU (*user space*) activities for the *direct_synth* job
- The use of a **BB attenuates** (about **15%**) the I/O processing time of the workflow jobs, for both read and write operations

Conclusion and Future Work

Major Findings

- **I/O write** performance was **improved** by a **factor of 9**, and **I/O read** performance by a **factor of 15**
- Performance **decreased slightly** at node counts above **64** (potential I/O ceiling)
- I/O performance must be **balanced** with parallel efficiency when using burst buffers with highly parallel applications
- **I/O contention** may **limit** the broad **applicability** of burst buffers for all workflow applications (e.g., in situ processing)

What's Next?

- Solutions such as I/O-aware scheduling or in situ processing may also not fulfill all application requirements

  We intend to investigate the use of ***combined in situ and in transit analysis***

- Development of a production solution for the Pegasus workflow management system

ON THE USE OF BURST BUFFERS FOR ACCELERATING DATA-INTENSIVE SCIENTIFIC WORKFLOWS

Thank You

Questions?

Rafael Ferreira da Silva, Ph.D.

Research Assistant Professor, Computer Science Department
Computer Scientist, USC Information Sciences Institute

rafsilva@isi.edu – http://rafaelsilva.com

U.S. DEPARTMENT OF ENERGY | Office of Science

Funded by the US Department of Energy under Grant #DE-SC0012636

USC Viterbi School of Engineering
Information Sciences Institute

panorama360