On the Role of Burst Buffers in Leadership-Class Storage Systems

Ning Liu, Jason Cope, Philip Carns, Christopher Carothers, Robert Ross, Gary Grider, Adam Crume, Carlos Maltzahn

Contact: liun2@cs.rpi.edu, chrisc@cs.rpi.edu, rross@mcs.anl.gov

Rensselaer

Argonne NATIONAL LABORATORY

What have we done?

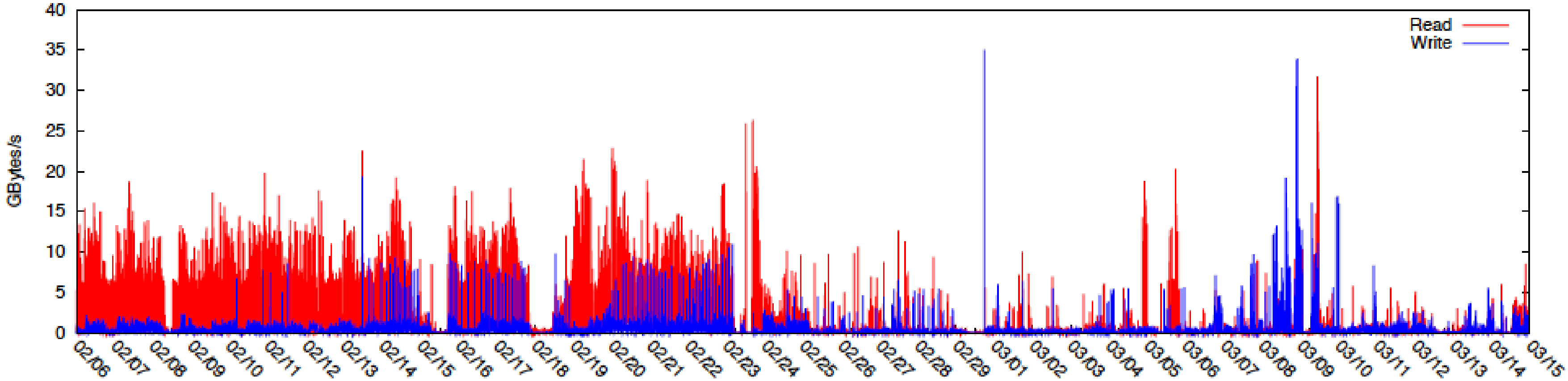
- CODES: Enabling Co-Design of Multi-Layer Exascale Storage Architectures (RPI and ANL)
- Goal: apply simulation to the understanding and design of complex HPC storage system.
- We modeled a leadership class storage system[1].
  - Argonne's Blue Gene/P
  - Simulate 128K way parallelism
  - Includes a file system model: PVFS
- Burst Buffer model and study is based on BG/P model.

[1] Modeling a Leadership-scale Storage System, N Liu, C Carothers, J Cope, P Carns, R Ross, A Crume, C Maltzahn 9th International Conference on Parallel Processing and Applied Mathematics 2011 (PPAM 2011)

Why do we need burst buffer?

- Modern HPC storage systems are designed to absorb peak I/O burst resulting in bandwidth waste.
- This storage system designs cannot keep pace with the growing data demands as supercomputers evolve to the era of exascale.

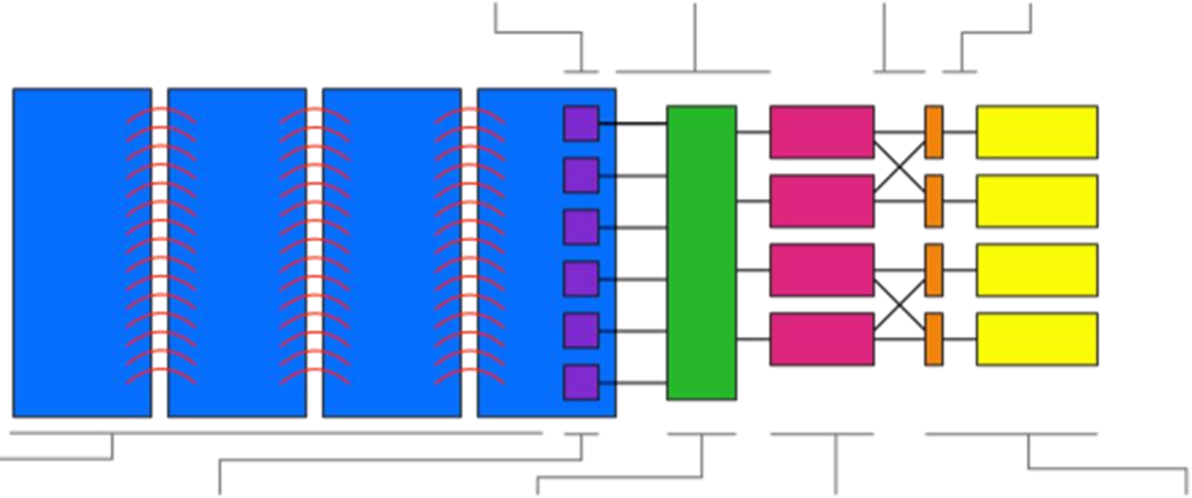
e.g. statistics from ALCF Intrepid Blue Gene/P system shows that 98.8% of the time the I/O system is at 33% bandwidth capacity or less, and 69.6% of the time the system is at 5% bandwidth capacity or less.



Aggregate I/O throughput over one-minute intervals on ALCF BG/P [2]

[2] Understanding and improving computational science storage access through continuous characterization Philip Carns, Kevin Harms, William Allcock, Charles Bacon, Samuel Lang, Robert Latham, Robert Rossl

Simulated System: IBM Blue Gene/P



BG/P Tree
6.8 Gbit/sec

Ethernet
10 Gbit/sec

InfiniBand
16 Gbit/sec

Serial ATA
3.0 Gbit/sec

Compute nodes

40,960 Quad core PowerPC 450 nodes with 2 Gbytes of RAM each

64 compute nodes per gateway node

Gateway nodes

640 Quad core PowerPC 450 nodes with 2 Gbytes of RAM each

16 gateway nodes per rack

Commodity network

900+ port 10 Gigabit Ethernet Myricom switch complex

Storage nodes

123 two dual core Opteron servers with 8 Gbytes of RAM each

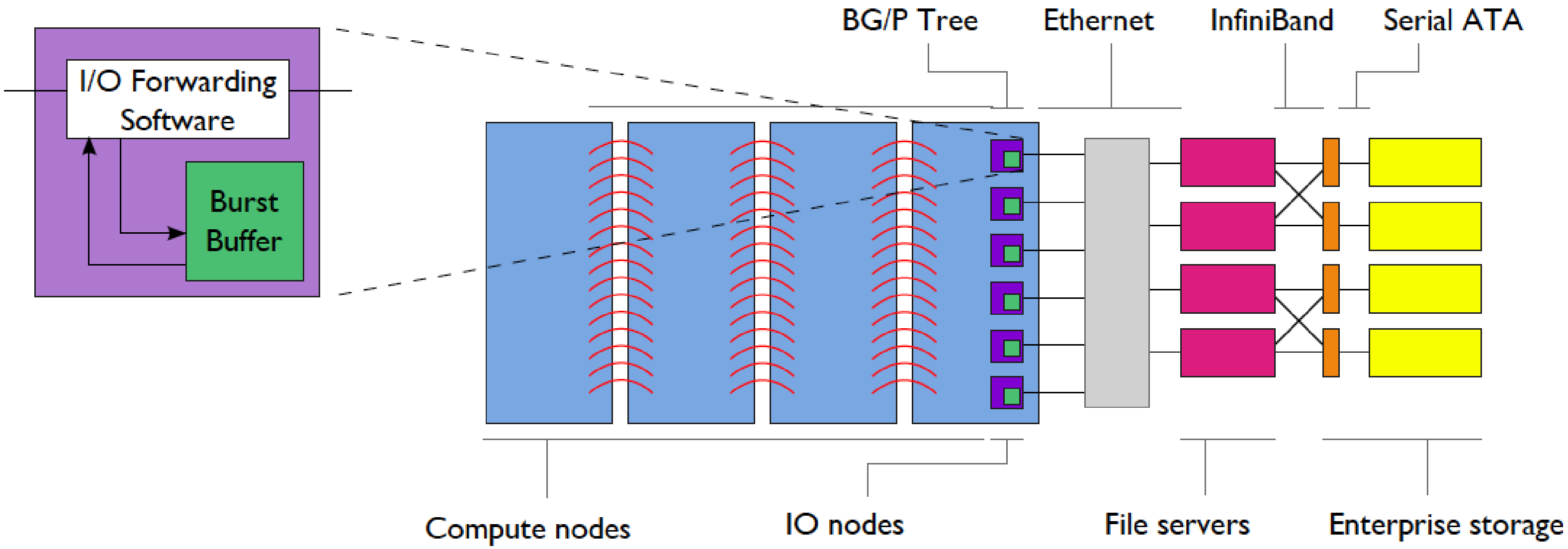
4-5 gateway nodes per storage node

Enterprise storage

DataDirect S2A9900 controller pairs with 480 1 Tbyte drives and 8 InfiniBand ports per pair

8 storage nodes per DDN

Incorporating Burst Buffers on Edge of System



- We propose augmenting existing I/O node designs with a tier of solid-state disk (SSD) burst buffers.

Tools: Discrete Event Simulation

- Discrete event simulation: computer model for a system where changes in the state of the system occur at *discrete* points in simulation time.

- Fundamental concepts:
  system state (state variables)
  state transitions (events)

- A DES computation can be viewed as a sequence of event computations, with each event computation is assigned a (simulation time) time stamp

- Each event computation can
  modify state variables
  schedule new events

Why PDES?

Why Parallel Discrete-Event Simulation (PDES)?

- Large-scale systems are difficult to understand
- Analytical models are often constrained

Parallel DES simulation offers:

- Dramatically shrink model's execution-time
- Prediction of future "what-if" systems performance
- Potential for real-time decision support
  - Minutes instead of days
  - Analysis can be done right away
- Example models: national air space (NAS), ISP backbone(s), distributed content caches, next generation supercomputer systems.

ROSS: Parallel Discrete Event Simulator

- Developed in ANSI C,

API is simple and lean.

- Using Jefferson's Time Warp

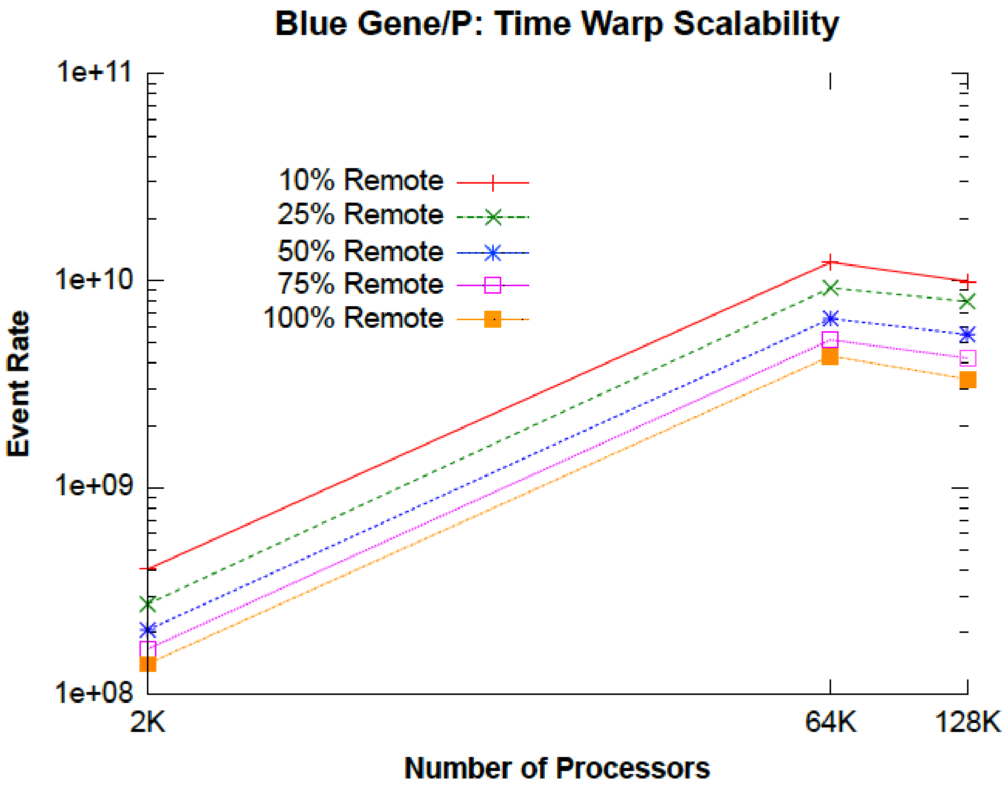
event scheduling mechanism.

- Reverse computation.

- Global virtual time algorithm

exploits IBM Blue Gene's fast

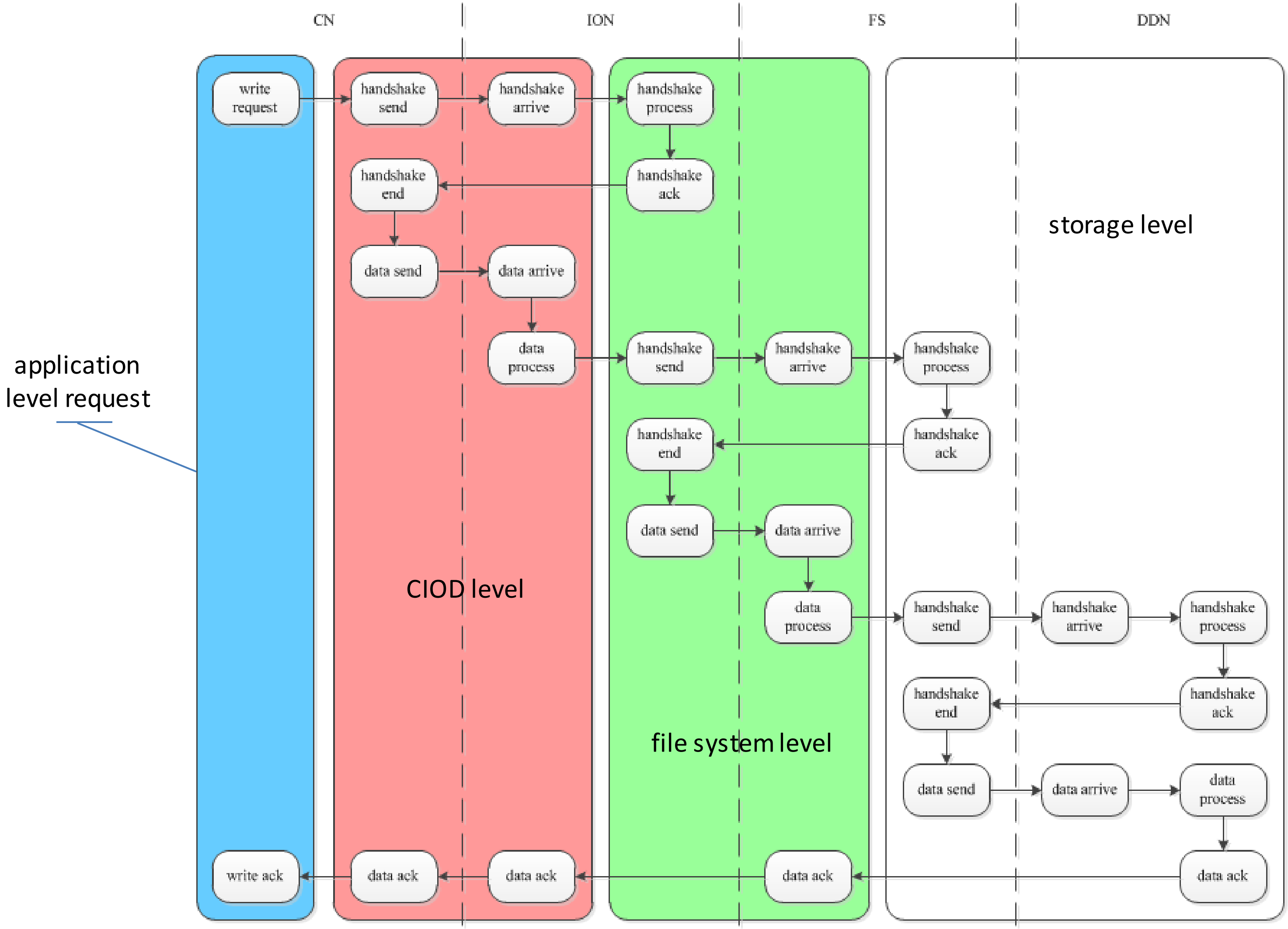
barrier and collective networks.



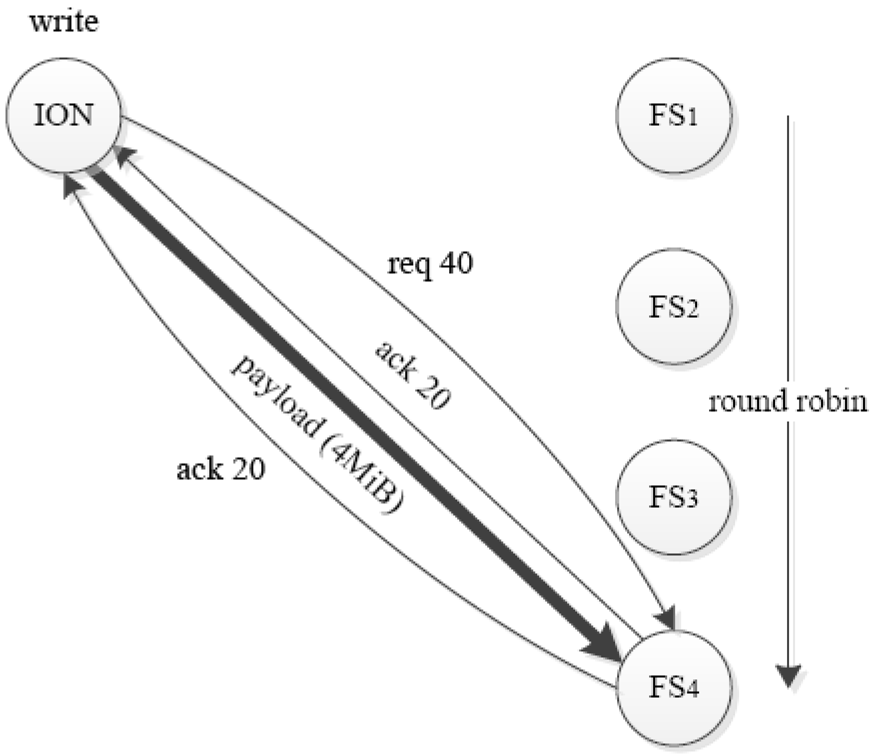
- ROSS main page:

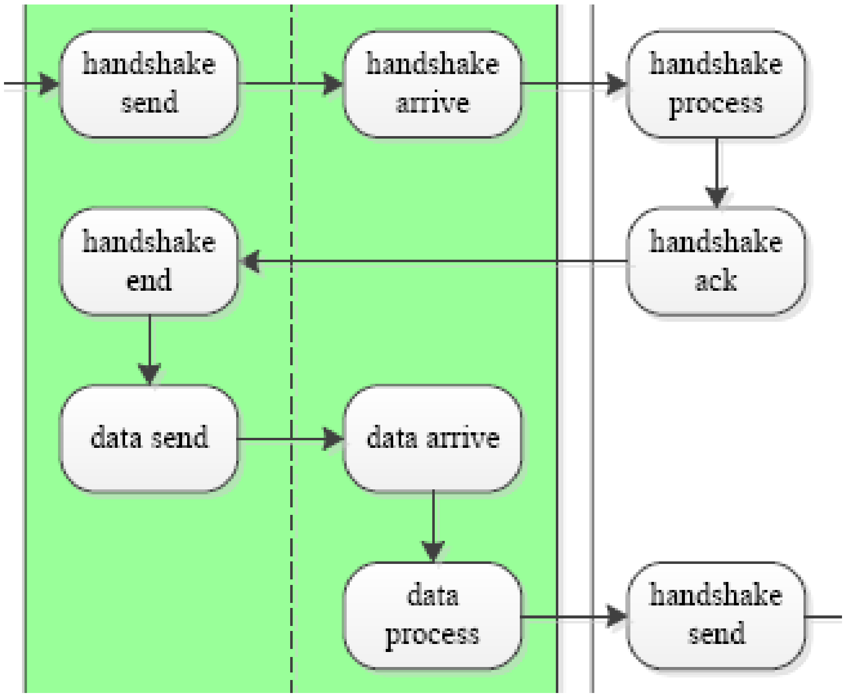
http://odin.cs.rpi.edu/ross/index.php/Main_Page

Write Request Event-Driven Model

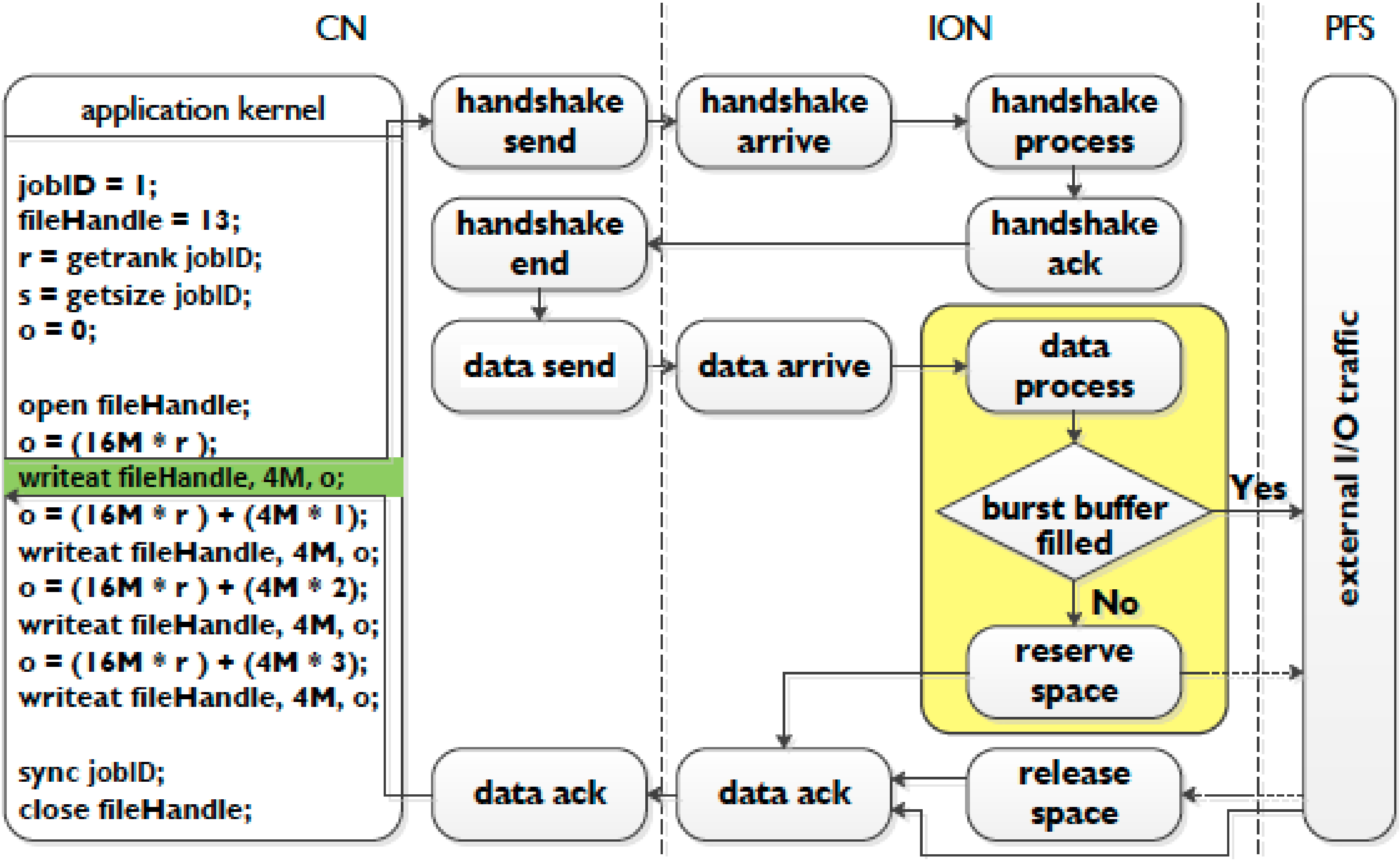


Snapshot of the PVFS Write Model





Burst Buffer PDES Model



CN

ION

PFS

application kernel

```
jobID = 1;
fileHandle = 13;
r = getrank jobID;
s = getsize jobID;
o = 0;

open fileHandle;
o = (16M * r );
writeat fileHandle, 4M, o;
o = (16M * r ) + (4M * 1);
writeat fileHandle, 4M, o;
o = (16M * r ) + (4M * 2);
writeat fileHandle, 4M, o;
o = (16M * r ) + (4M * 3);
writeat fileHandle, 4M, o;

sync jobID;
close fileHandle;
```

handshake send

handshake arrive

handshake process

handshake end

handshake ack

data send

data arrive

data process

burst buffer filled

Yes

No

reserve space

external I/O traffic

data ack

data ack

release space

Study of Bursty Applications

- Argonne's Blue Gene/P system host many scientific applications.
- We quantify the I/O behavior by analyzing one month of production I/O activity from December 2011 using Darshan.
- Darshan captures application-level access pattern information with per process and per file granularity. (lightweight I/O characterization tool)

TABLE I: Top four write-intensive jobs on Intrepid, December 2011

Project	**Procs**	**Nodes**	**Total Written**	**Run Time (hours)**	**Avg. Size and Subsequent Idle Time for Write Bursts>1 GiB**				
					Count	Size	Size/Node	Size/ION	Idle Time (sec)
PlasmaPhysics	131,072	32,768	67.0 TiB	10.4	1	33.5 TiB	1.0 GiB	67.0 GiB	7554
					1	33.5 TiB	1.0 GiB	67.0 GiB	end of job
Turbulence1	131,072	32,768	8.9 TiB	11.5	5	128.2 GiB	4.0 MiB	256.4 MiB	70
					1	128.2 GiB	4.0 MiB	256.4 MiB	end of job
					421	19.6 GiB	627.2 KiB	39.2 MiB	70
AstroPhysics	32,768	8,096	8.8 TiB	17.7	1	550.9 GiB	68.9 MiB	4.3 GiB	end of job
					8	423.4 GiB	52.9 MiB	3.3 GiB	240
					37	131.5 GiB	16.4 MiB	1.0 GiB	322
					140	1.6 GiB	204.8 KiB	12.8 MiB	318
Turbulence2	4,096	4,096	5.1 TiB	11.6	21	235.8 GiB	59.0 MiB	3.7 GiB	1.2
					1	235.8 GiB	59.0 MiB	3.7 GiB	end of job

Burst Buffer Model Parameters

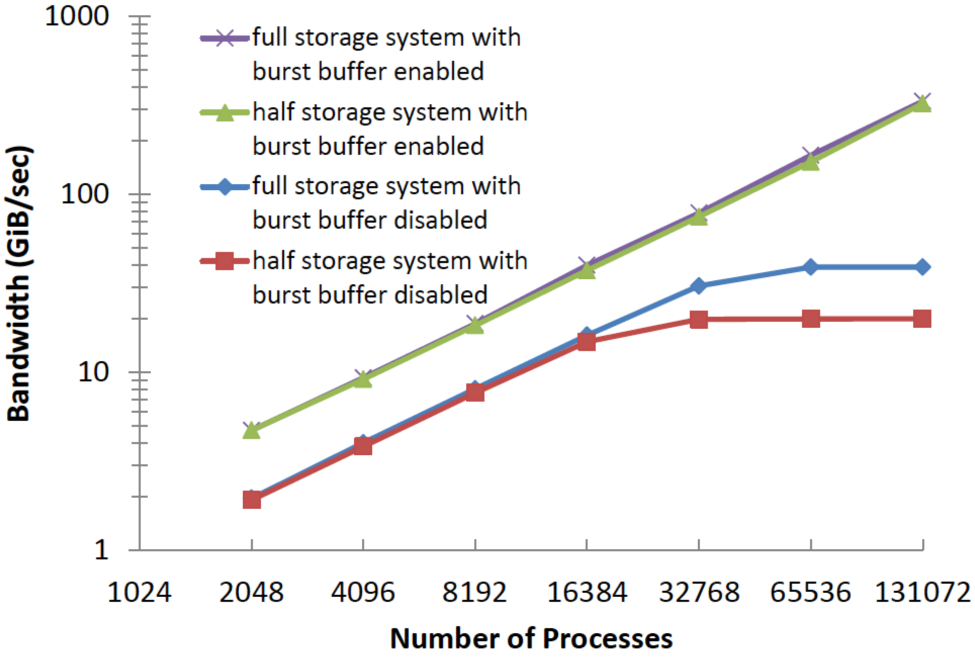
TABLE II: Summary of relevant SSD device parameters and technology available as of January 2012

Vendor	Size (TiB)	NAND	Bandwidth (GiB/s)		Latency (µs)	
			Write	Read	Write	Read
FusionIO	0.40	SLC	1.30	1.40	15	47
FusionIO	1.20	MLC	1.20	1.30	15	68
Intel	0.25	MLC	0.32	0.50	80	65
Virident	0.30	SLC	1.10	1.40	16	47
Virident	1.40	MLC	0.60	1.30	19	62

Burst buffer latency model: $T = \frac{L_{data}}{B_{BB}} + T_{BB}$

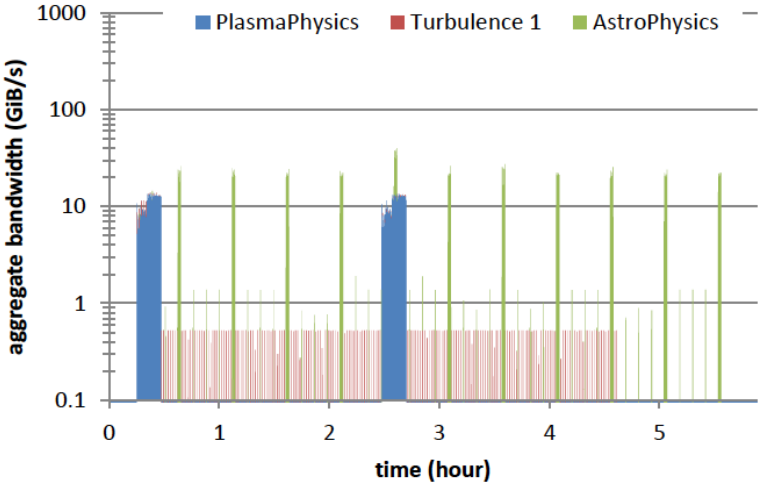
Single I/O Workload Case

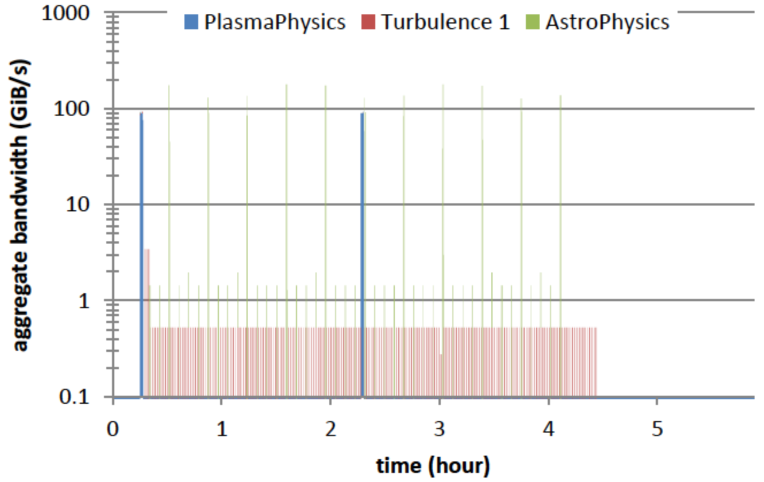
- Use IOR benchmark
(consecutive 4MiB write
 requests)
- Capture write time, excludes open/close time
- Full storage system uses 128 file servers
- Half storage system uses 64 file servers



Simulated performance of IOR for various storage system and burst buffer configurations

Mixed Workload: Application View



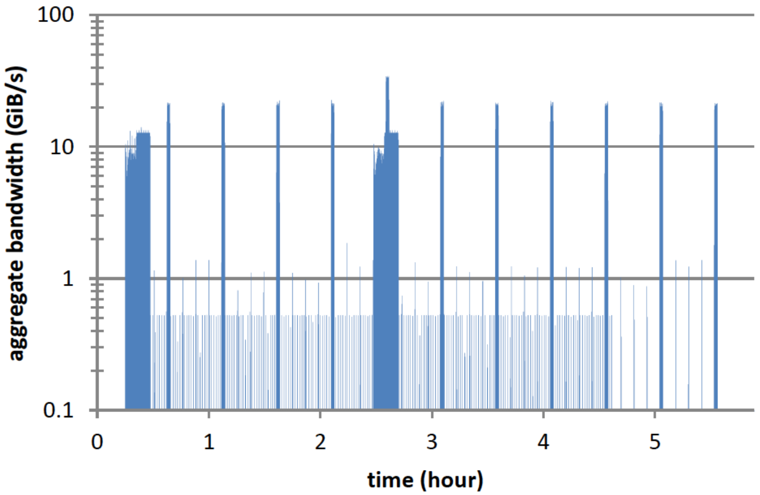


PlasmaPhysics: 2 large (256 MiB) write with 2-hour interval. (32K processes)
AstroPhysics: 3 small write followed by 1 large write, repeat 11 times. (64K processes)
Turbulence1: 220 small write. (32K processes)

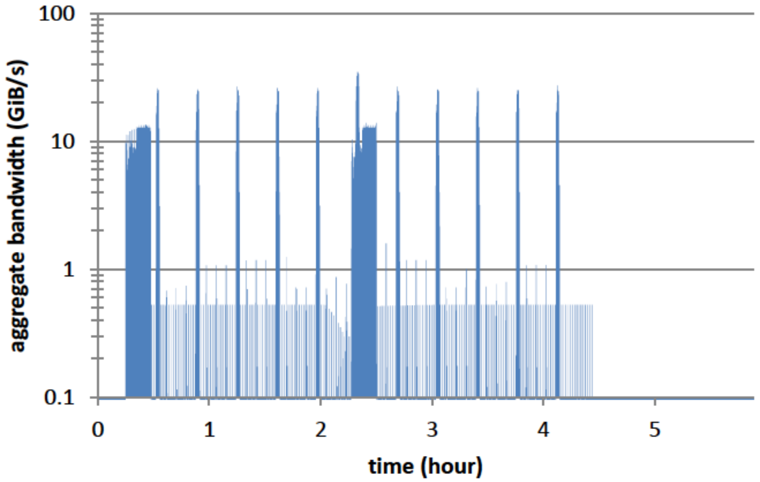
Full storage system with burst buffer disabled.
Jobs execution time = 5.5 hours

Full storage system with burst buffer enabled.
Jobs execution time = 4.4 hours

Mixed Workload: Server View

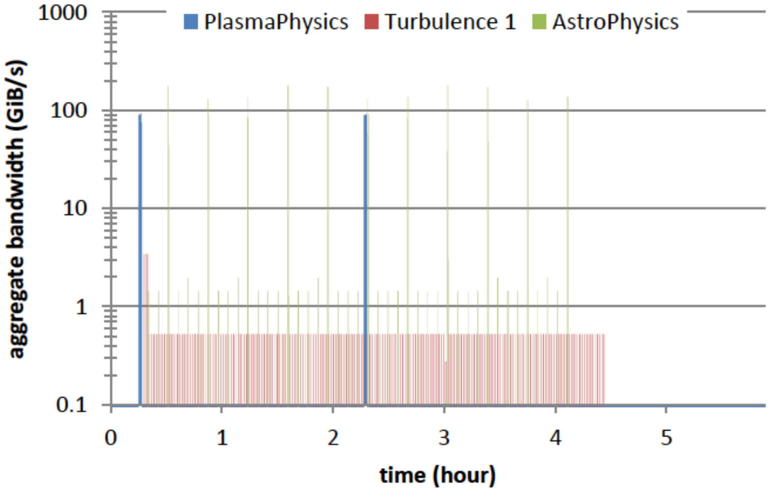


Full storage system with burst buffer disabled.



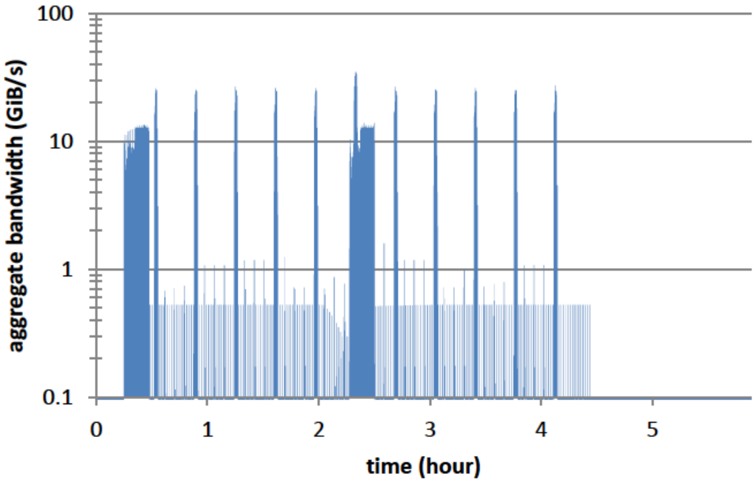
Full storage system with burst buffer enabled.

Application View vs. Server View



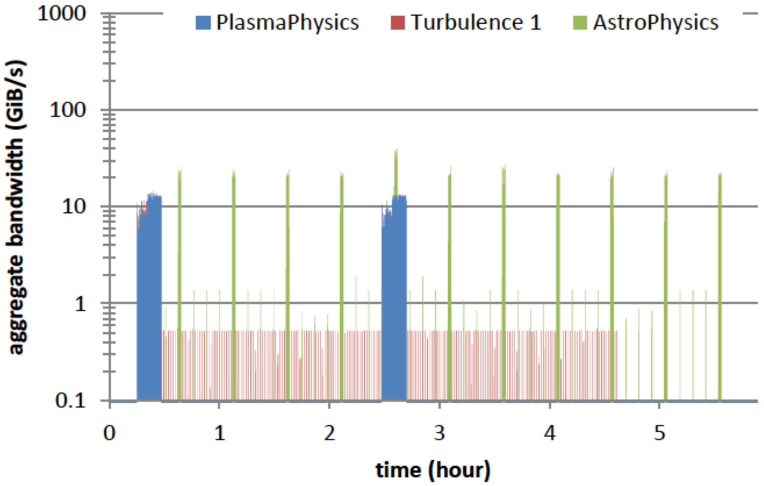
Full storage system with burst buffer enabled. (Application View)

Jobs execution time = 4.4 hours



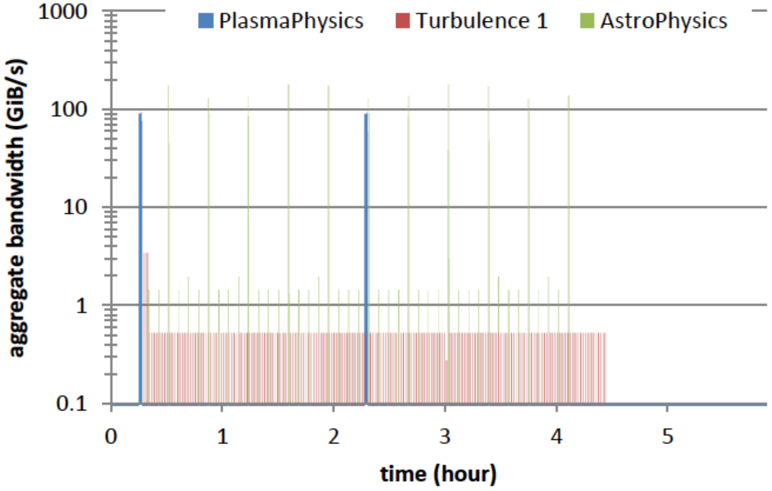
Full storage system with burst buffer enabled. (Server View)

Full Storage vs. Half Storage: Application View



Full storage system with burst buffer disabled.

Jobs execution time = 5.5 hours



Half storage system with burst buffer enabled.

Jobs execution time = 4.4 hours

Conclusions

- Burst buffers have been proposed as a way to meet peak I/O rates with lower performance external storage.
- We've tried to better quantify the benefits of this approach in this work.
- Bursts from applications today are of modest size, as measured on our system, allowing use of limited size buffers.
- In the context of the BG/P architecture, buffers integrated into I/O nodes could provide a measurable improvement in application time to solution while simultaneously enabling the deployment of a less capable external I/O system.

Future Work

- Investigate the use of burst buffer (in-system storage) on different tiers of the storage system
- Enable the use of simulator framework to future system architecture
- Understand complex system components and facilitate the design through simulation

Acknowledgements

- Special thanks to Dr. Jason Cope;
- Special thanks to my advisor Prof. Christopher Carothers, thesis advisor Dr. Robert Ross;
- Thanks to Philip Carns, Kevin Harms, Gary Grider, Carlos Maltzahn, and Adam Crume.

- This work was supported in part by the Office of Advanced Scientific Computer Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357 and partially by Contract DE-SC0005428 and the LANL/UCSC Institute for Scalable Scientific Data Management (ISSDM).

- Thanks everyone here today!
- Questions?