Development of a Burst Buffer System for Data-Intensive Applications

Teng Wang†, Sarp Oral‡, **Michael Pritchard**†, Kevin Vasko†, Weikuan Yu†

Auburn University†

Oak Ridge National Laboratory‡

Outline

- Background
- Motivation
- Design
- Evaluation
- Future Work

Background

- Computational power of HPC systems has grown explosively

Jaguar (2009)
2.3 Petaflops/s

Titan (2012)
27 Petaflops/s

> 10x

Background

- HPC growth allows more sophisticated scientific applications
  - Climate modeling
  - Seismic hazard analysis
  - Protein folding
  - Fusion simulation

AUBURN UNIVERSITY

Motivation

- To offset potential loss due to system failure, long-running applications utilize periodic **checkpointing**
  - Per-checkpoint I/O demand increases as supercomputers grow in size
  - MTBF is expected to reach 3-26 minutes for Exascale systems
- We can characterize execution of applications incorporating checkpointing as cycles of computation and I/O

Motivation

- Improvements to I/O bandwidth fail to achieve parity with the rapid growth of computing power
  - Upgrades to the Spider filesystem yielded only a 4x improvement in aggregate bandwidth (240 GB/s to 1 TB/s)
- As a result, checkpointing applications are becoming **increasingly I/O bound**

Motivation

Can we shorten the periods of I/O?

Utilize burst buffers

- Large buffering space provisioned by high-performance storage devices (e.g. DRAM, SSDs)
- Rapidly offload checkpoint data, allowing the next period of computation to begin sooner

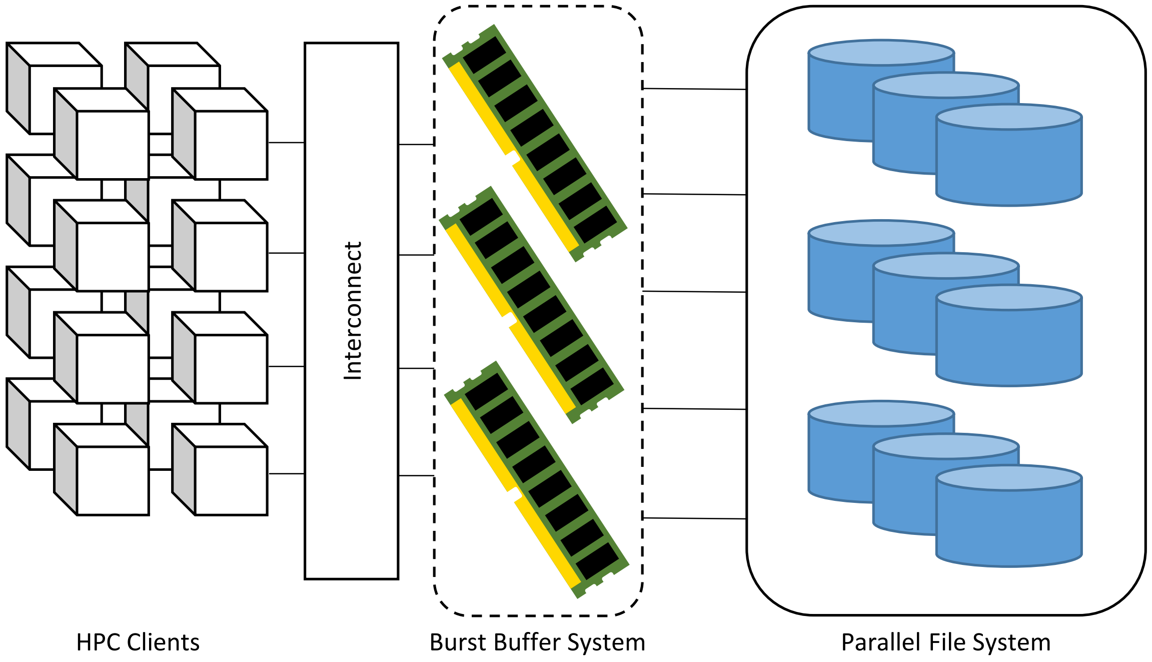
Design Goals

- Rapid recovery after application restart
- Coordinated, balanced data flushing
- Fault Tolerance

AUBURN UNIVERSITY

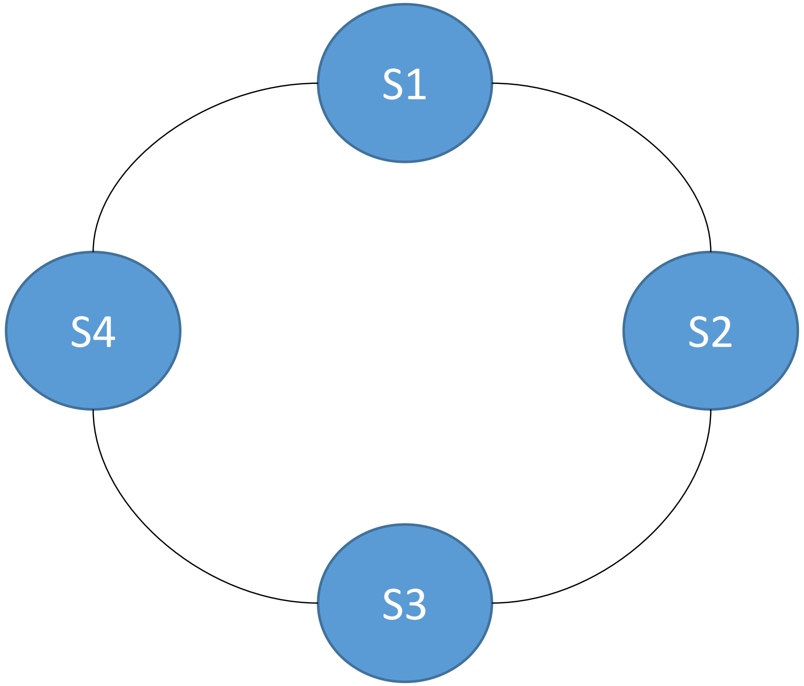
Design

- Burst buffer system lies between processing elements and back-end persistent storage



Design

- Three software components:
  - Clients
    - Reside on each compute node
  - Servers
    - Located on each burst buffer node
    - Logically linked via a ring topology
    - Each server maintains a list of its neighbors
  - Manager
    - Located on one burst buffer node
    - A new manager can be elected in the event of a failure



AUBURN UNIVERSITY

Rapid Client Recovery

- Typically, clients access checkpoint data from PFS following failure and restart
  - Accessing PFS is expensive
- By retaining recent checkpoints in burst buffers, we can improve client recovery times

AUBURN UNIVERSITY

Data Flushing

- Gains realized through a burst buffer system can be severely limited if data flushing is not handled properly
  - Unbalanced workloads can result in spillover to secondary storage devices (e.g. SSD)
  - Uncoordinated flushing can generate significant PFS lock contention

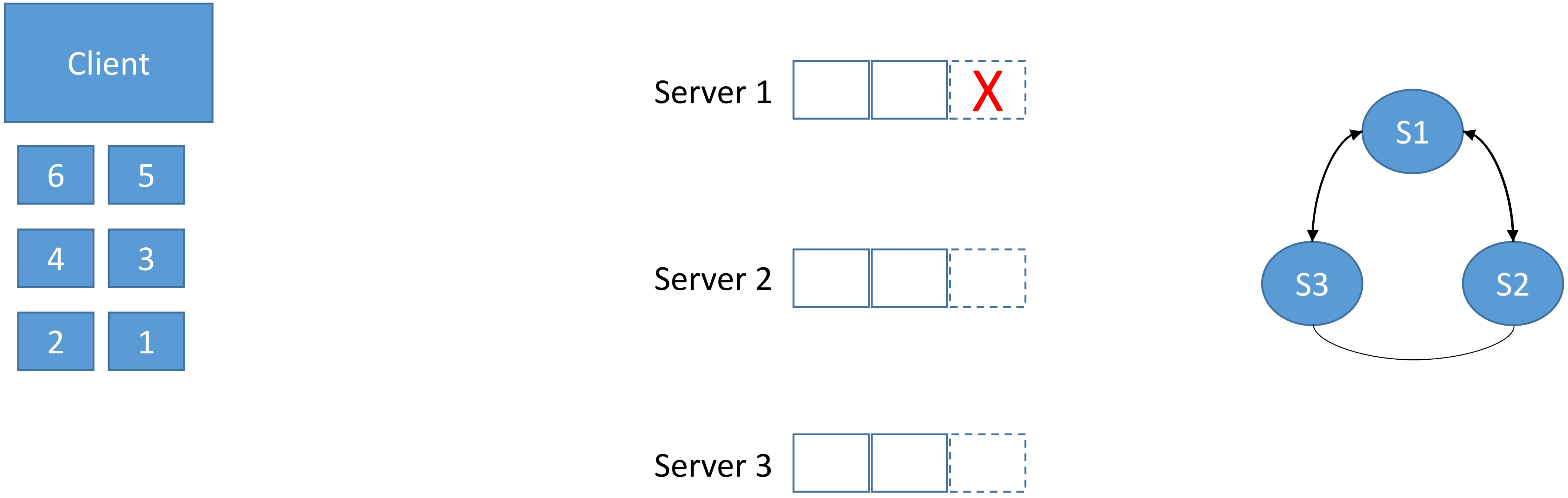
AUBURN UNIVERSITY

Load Balancing

- Avoiding spillover to secondary storage is important for maximizing I/O
- To avoid spillover, a server with insufficient memory sends messages along the ring to identify the least-full server.
- The client is the informed of the new destination for the data

AUBURN
UNIVERSITY

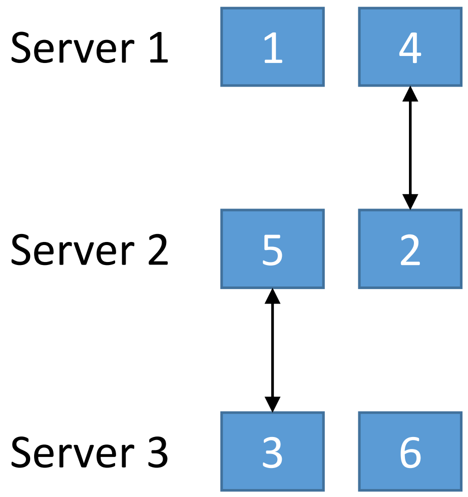
Load Balancing Example



Coordinated Data Flushing

- Write requests aren't guaranteed to be of a size divisible by the PFS stripe size
- Each burst buffer server may have numerous noncontiguous data segments
- After all data has been received by the servers, the manager coordinates an optimum inter-server shuffling

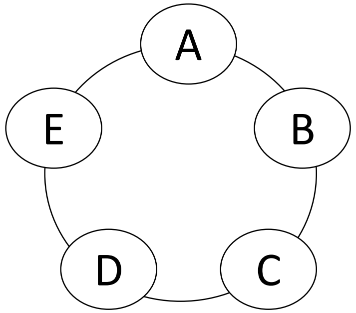
Data Shuffling



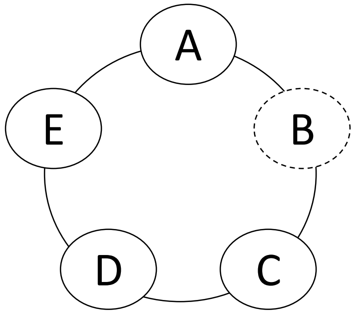
Fault Tolerance

- Burst buffer servers are themselves not immune to failures
  - Fault tolerance is facilitated using the ring topology
- Servers periodically synchronize with each other and the manager to account for failures and new joins
  - Synchronization handles neighbor list updates and data replication

Server Failure



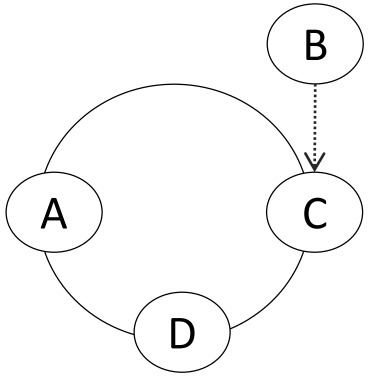
ID	S1	S2	P
A	B	C	E
B	C	D	A
C	D	E	B
D	E	A	C
E	A	B	D



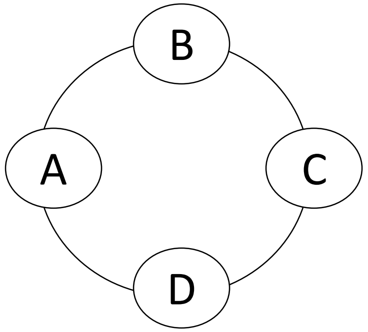
ID	S1	S2	P
A	C	D	E
C	D	E	A
D	E	A	C
E	A	C	D

AUBURN UNIVERSITY

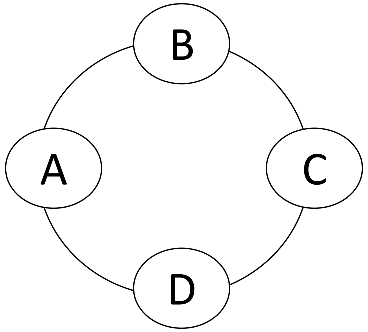
Server Join



ID	S1	S2	P
A	C	-	D
B	C	-	-
C	D	-	A
D	A	-	C

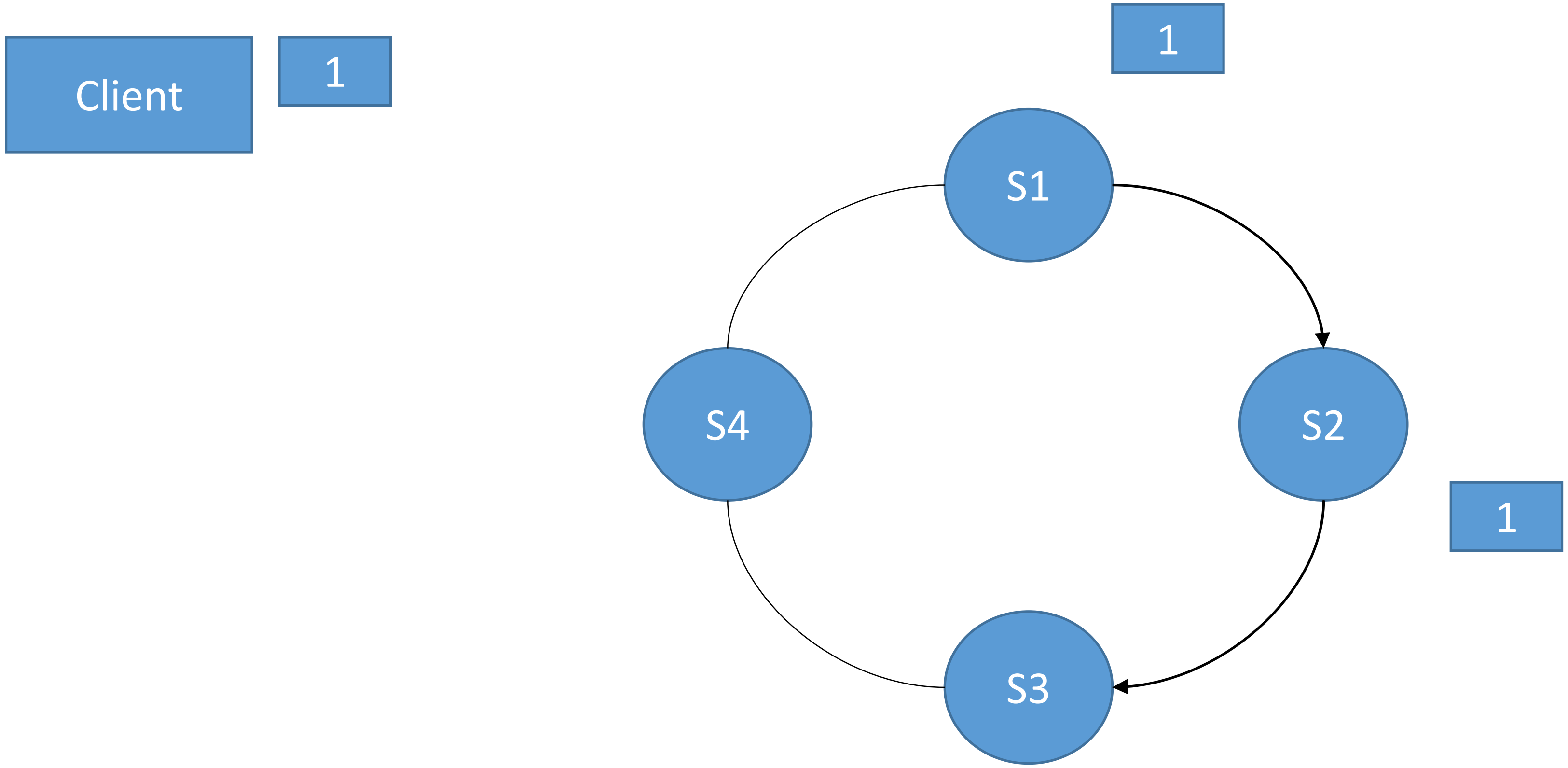


ID	S1	S2	P
A	B	-	D
B	C	-	A
C	D	-	B
D	A	-	C



ID	S1	S2	P
A	B	C	D
B	C	D	A
C	D	A	B
D	A	B	C

Data Replication

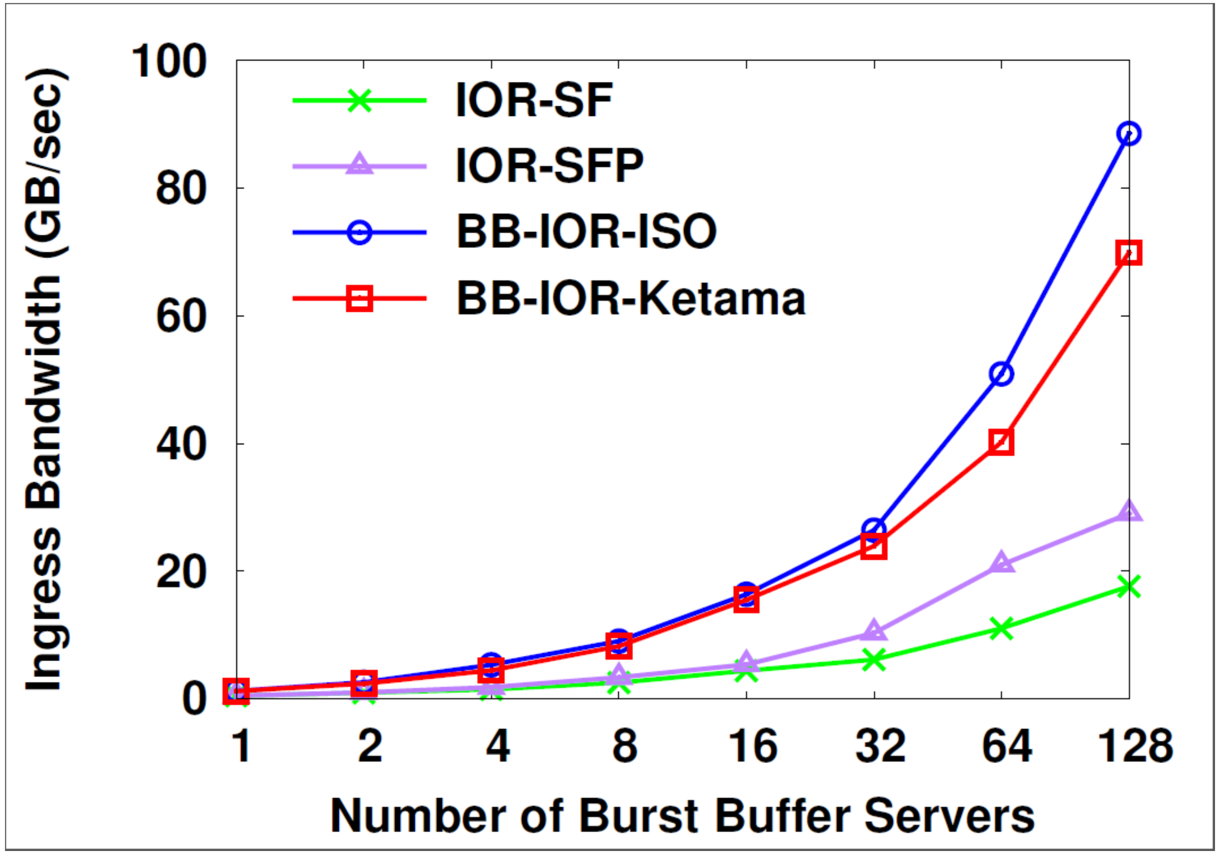


Evaluation

- Initial evaluations conducted using the Titan supercomputer
  - 128 compute nodes (burst buffer clients)
  - 1 – 128 nodes allocated as burst buffer servers
  - Spider II file system

Ingress Bandwidth

- BB-IOR-Ketama
  - Balanced distribution of client workloads to all servers
- BB-IOR-ISO
  - Client writes all of its data to a particular server
  - 2.8x improvement over IOR-SFP
  - 1.7x improvement over IOR-SF



AUBURN UNIVERSITY

Future Work

- Leveraging burst buffer to improve performance of read-bound applications
  - BLAST+
  - Argus
- Creating an application-agnostic client API to improve portability

AUBURN UNIVERSITY

Acknowledgements

AUBURN
UNIVERSITY

OAK RIDGE
National Laboratory

AUBURN
UNIVERSITY