CRFS: A Lightweight User-Level Filesystem for Generic Checkpoint/Restart

Xiangyong Ouyang, Raghunath Rajachandrasekar,
Xavier Besseron, Hao Wang, Jian Huang,
Dhabaleswar K. Panda

Department of Computer Science & Engineering
The Ohio State University

Outline

- Introduction and Motivation
- Checkpoint Profiling and Analysis
- CRFS: A lightweight user-level Filesystem
- Performance Evaluation
- Conclusions and Future Work

Introduction

- High Performance Computing (HPC) keeps growing in terms of scale and complexity
  - Mean-time-between-failures (MTBF) is getting smaller
  - Fault-Tolerance is becoming imperative
  - Checkpoint/Restart is becoming increasingly important
- Checkpoint/Restart (C/R): the most widely adopted Fault-tolerance approach
  - Phase 1: build a global consistent state (suspend communications)
  - Phase 2: create a snapshot of every process, save it to a shared parallel filesystem
  - Phase 3: Resume communications and execution

Problems with Basic C/R

- Checkpoint/Restart mechanisms incur intensive I/O overhead
  - × Sheer amount of data
  - × Simultaneous IO streams leads to severe contentions
  - × Large variations of individual process completion time
- A lot of studies to tackle the I/O bottleneck
  - – Performed inside specific MPI stack or checkpoint library or applications
  - × **Not portable**
  - × **Constrained to certain MPI stacks**

Problem Statements

- What are the primary causes of intensive I/O overhead for Checkpoint / Restart?
- How to design a portable and generic solution with optimizations to improve C/R performance?
- Can such a portable solution benefit a wide range of MPI stacks?
- What will be the performance benefits?

Outline

- Introduction and Motivation
- Checkpoint Profiling and Analysis
- CRFS: A lightweight user-level Filesystem
- Performance Evaluation
- Conclusions and Future Work

MVAPICH/MVAPICH2 Software

- MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1) and MVAPICH2 (MPI-2)
  - Used by more than 1,650 organizations worldwide (in 63 countries)
  - Empowering many TOP500 clusters (7^{th}, 17^{th} …)
  - Available with software stacks of many IB, 10GE/iWARP and RoCE, and server vendors including Open Fabrics Enterprise Distribution (OFED)
  - Available with Redhat and SuSE Distributions
  - http://mvapich.cse.ohio-state.edu/
- Has supported Checkpoint/Restart and Process Migration for the last several years
  - Already used by many organizations

Checkpoint Writing Profiling (1)

Checkpoint Writing information [1]

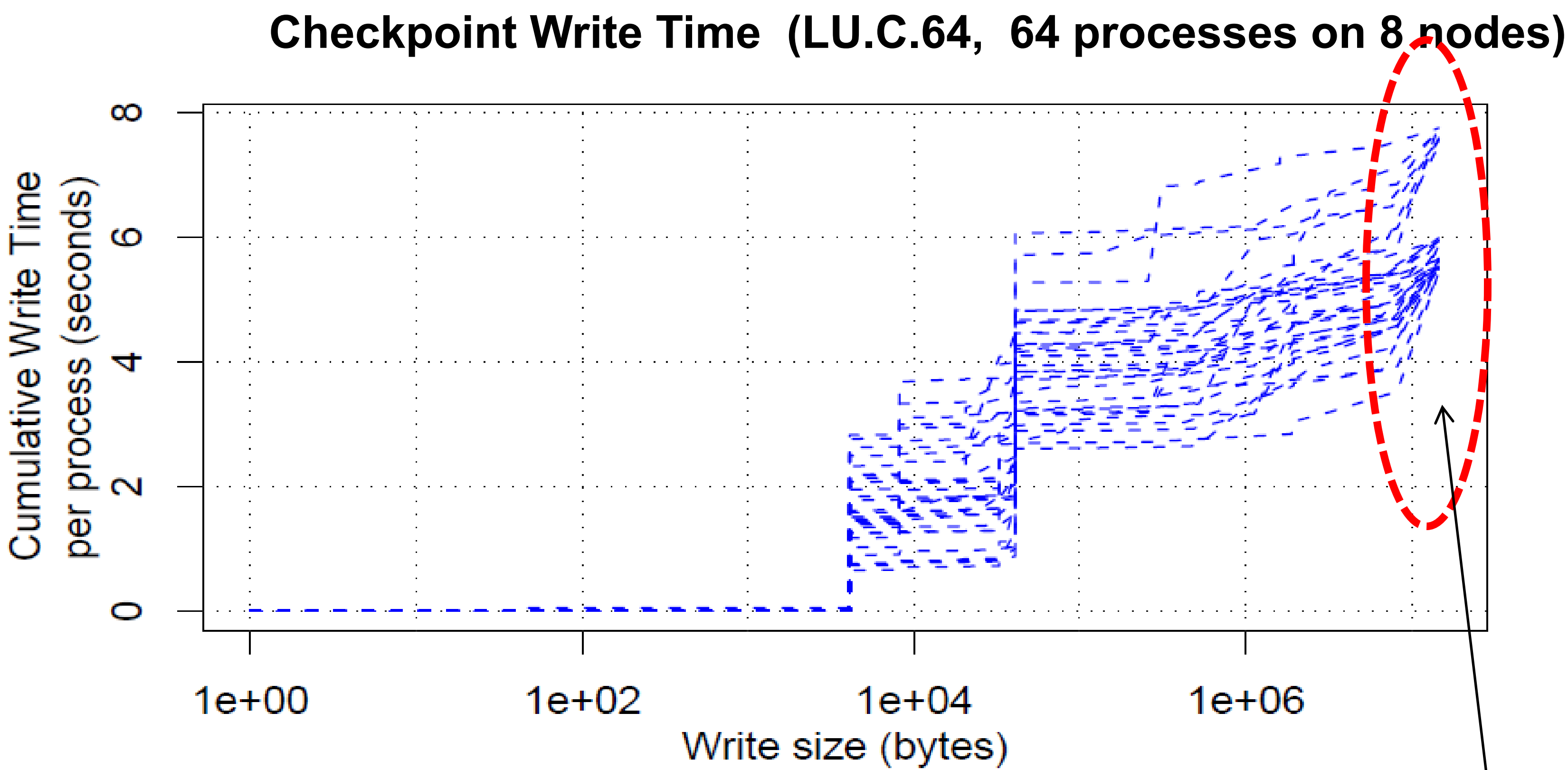
Write Size	% of Writes	% of Data	% of Time
0-64	50.86	0.04	0.17
64-256	0.61	0.00	0.00
256-1K	0.25	0.01	0.00
1K-4K	9.46	1.53	0.01
4K-16K	36.49	11.36	44.66
16K-64K	0.74	0.77	6.55
64K-256K	0.49	3.79	11.80
256K-512K	0.25	3.58	1.75
512K-1M	0.61	17.72	14.72
> 1M	0.25	61.21	20.35

Lots of small/median writes ➔ Inefficient IO

- **NAS Parallel Benchmark - LU.C.64**
- **Compute nodes: dual Quad-core Xeon,**
- **MVAPICH2-1.6 with Checkpoint/Restart support**
- **Checkpoint to ext3**
- **Checkpoint size: 1,472 MB, VFS write calls per node: 7800**

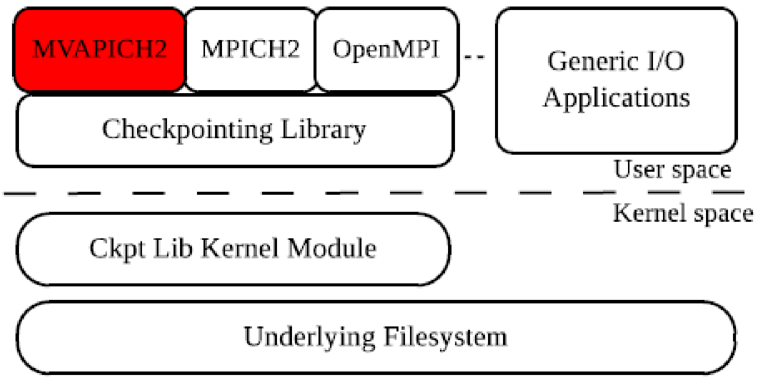
[1] X. Ouyang, K. Gopalakrishnan, D. K. Panda, "Accelerating Checkpoint Operation by Node-Leve Write Aggregation on Multicore Systems", ICPP 2009

Checkpoint Writing Profiling (2)



- contentions caused by concurrent writes ➔ wide variation of completion time.
- Faster process has to wait for slower counterparts
- Slow down checkpoint as a while

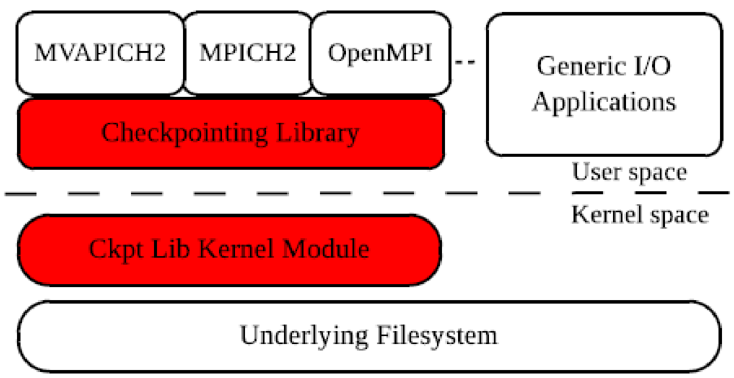
Optimize IO at Different Layers



Optimizations in specific MPI stacks

- × **Only benefit certain MPI implementations**

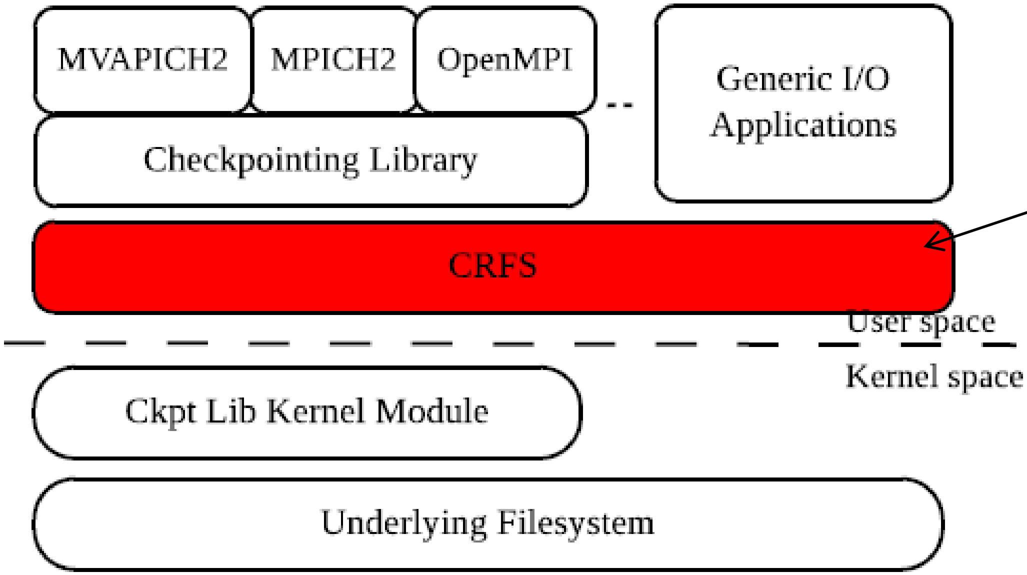
➔ How to get both performance and portability at the same time?



Optimizations in checkpoint library

- × **Require changes in kernel modules, not portable**

Our Approach



CRFS: a generic Checkpoint/Restart Filesystem

CRFS: a user-level filesystem optimized for checkpoint I/O

√ **User-level design: portable**

√ **Optimizations inside filesystem: transparently benefit a wide range of MPI stacks and applications**

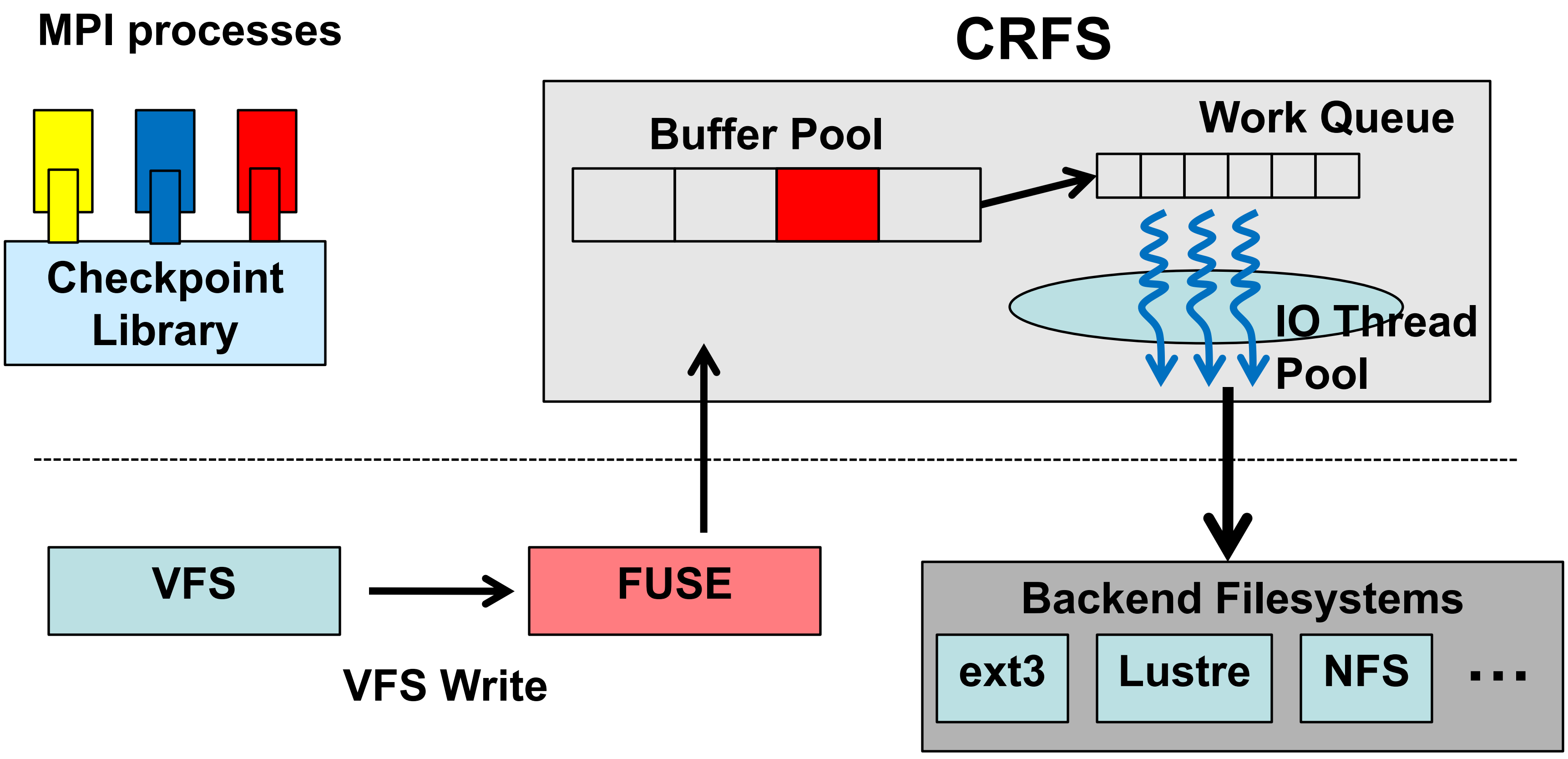
Outline

- Introduction and Motivation
- Checkpoint Profiling and Analysis
- CRFS: A lightweight user-level Filesystem
- Performance Evaluation
- Conclusions and Future Work

CRFS Design Strategies

- Based on FUSE: user-level filesystem
- Intercepts VFS write system calls
  - Aggregates many writes into bigger (fewer) chunks (better IO efficiency)
- Internal IO thread pool: asynchronously write bigger data chunks to back-end filesystem
  - Reduce IO contentions
  - ext3, NFS, Luster etc.

CRFS Design



Design choices
- Buffer pool size
- IO thread pool

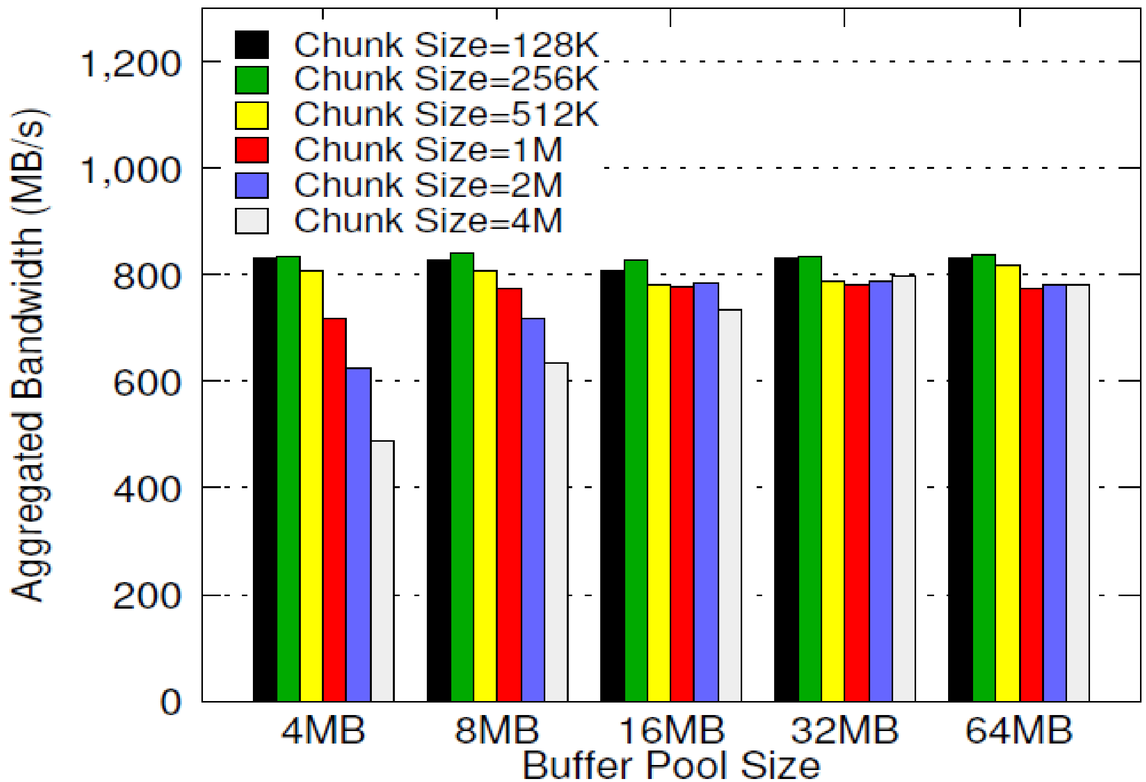
Outline

- Introduction and Motivation
- Checkpoint Profiling and Analysis
- CRFS: A lightweight user-level Filesystem
- Performance Evaluation
- Conclusions and Future Work

Experimental Setup

- Environment
  - Dual-socket Quad core Xeon , InfiniBand DDR, Linux 2.6.30, FUSE-2.8.5
  - NAS parallel Benchmark (NPB) 3.3, LU with class B/C/D input
  - MVAPICH2-1.6rc3, OpenMPI 1.5.1, MPICH2 1.3.2p1
    - With Checkpoint/Restart support
  - No modifications to any MPI stacks required
  - Backend Filesystem:
    - Ext3, NFSv3, Lustre 1.8.3 (3 OSS, 1 MDS, o2ib transport)
- **Experiments**
  - **Single node RAW IO bandwidth**
    - **To select proper design parameters**
  - **Checkpoint time with 3 MPI stacks**
    - **Evaluate CRFS performance**
  - **CRFS scalability with varied level of IO multiplexing**
  - **CRFS capability to reduce variation in checkpoint completion time**

CRFS Raw Write Bandwidth



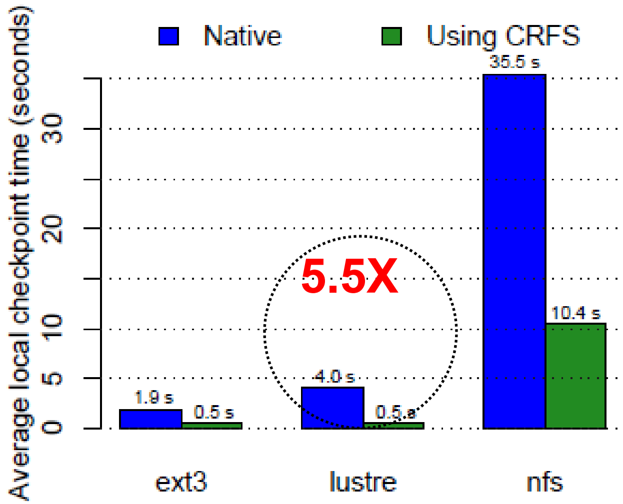
8 processes writing concurrently on a node.

- ✓**16 MB buffer pool can generate good throughput**
- ✓**128 KB chunk size performs well**
- ✓**4 IO threads yield the best performance in most cases**

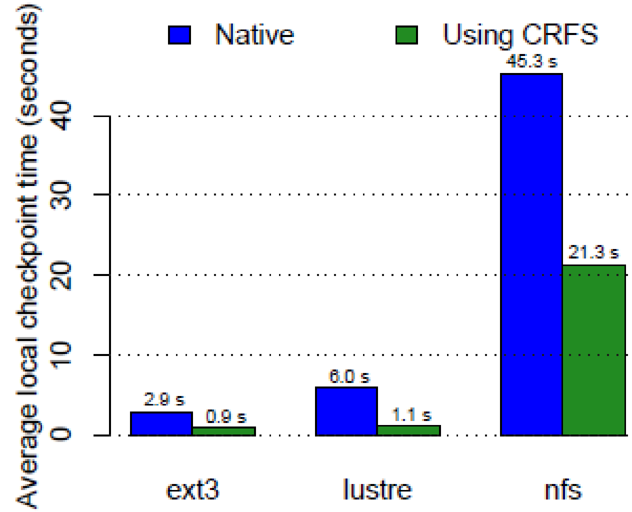
Checkpoint Sizes

Benchmark	MPI Library	Total Checkpoint Size (MB)	Process Image Size (MB)
LU.B.128	MVAPICH2-IB	903.2	7.1
	OpenMPI-IB	909.1	7.1
	MPICH2-TCP	497.8	3.9
LU.C.128	MVAPICH2-IB	1,928.7	15.1
	OpenMPI-IB	1,751.7	13.7
	MPICH2-TCP	1,359.6	10.7
LU.D.128	MVAPICH2-IB	13,653.9	106.7
	OpenMPI-IB	13,864.9	108.3
	MPICH2-TCP	13,261.2	103.6

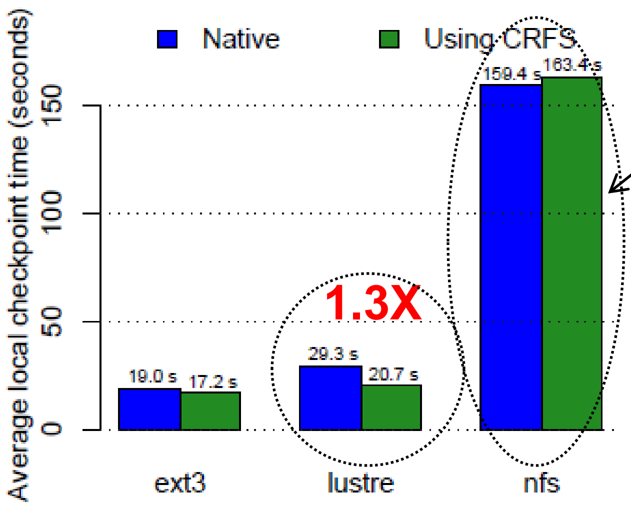
Checkpoint Time (MVAPICH2)



(a) LU.B.128



(b) LU.C.128

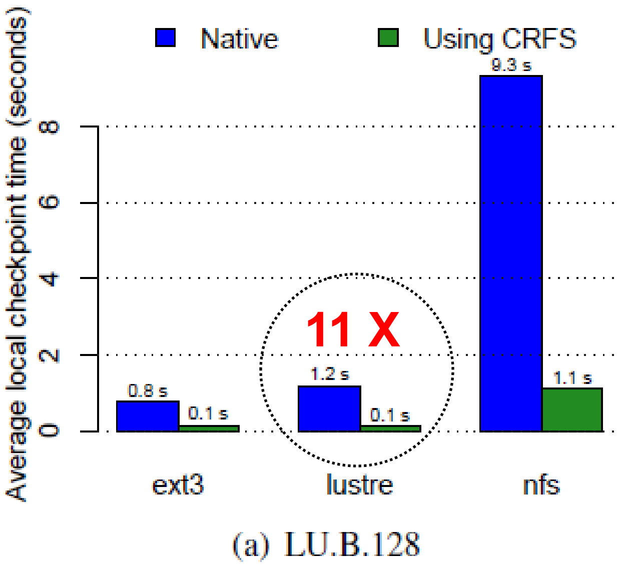


(c) LU.D.128

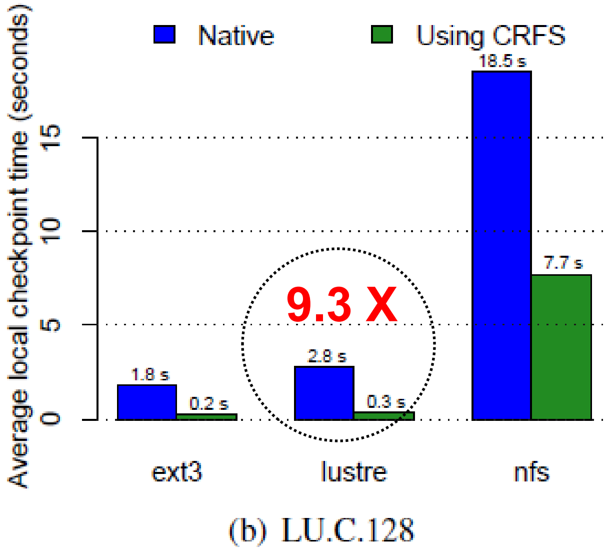
- Single NFS server cannot Handle heavy IO
- FUSE overhead manifested

- **Lustre: CRFS is 5.5X / 4.5X / 1.3X faster than native with class B/C/D inputs**
- **Ext3: CRFS is 2.8X / 2.2X / 1.1X faster than native with class B/C/D inputs**

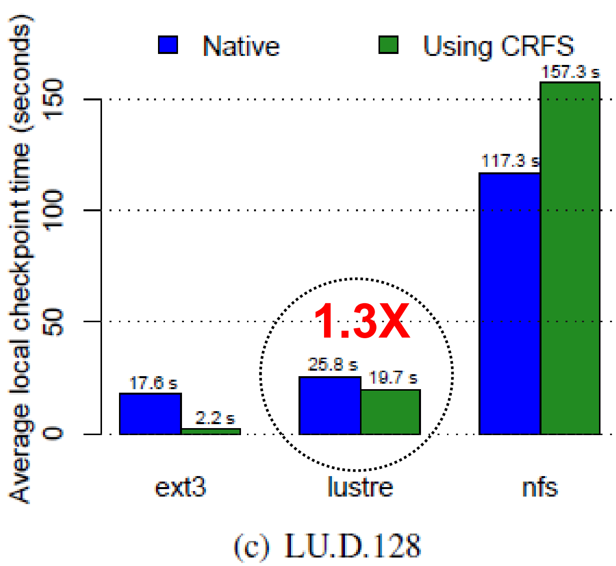
Checkpoint Time (MPICH2)



(a) LU.B.128



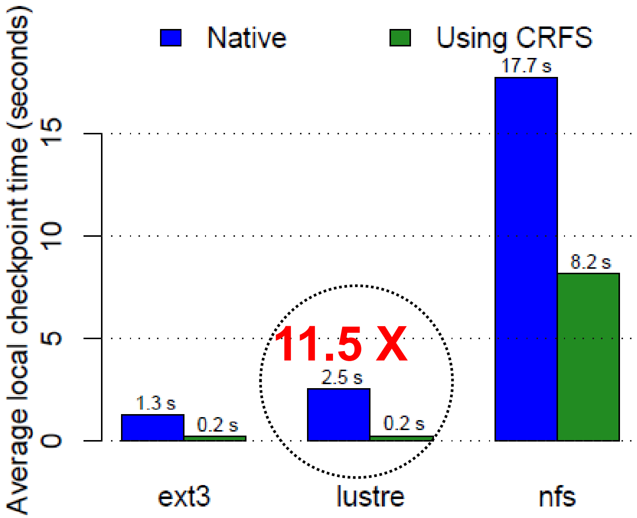
(b) LU.C.128



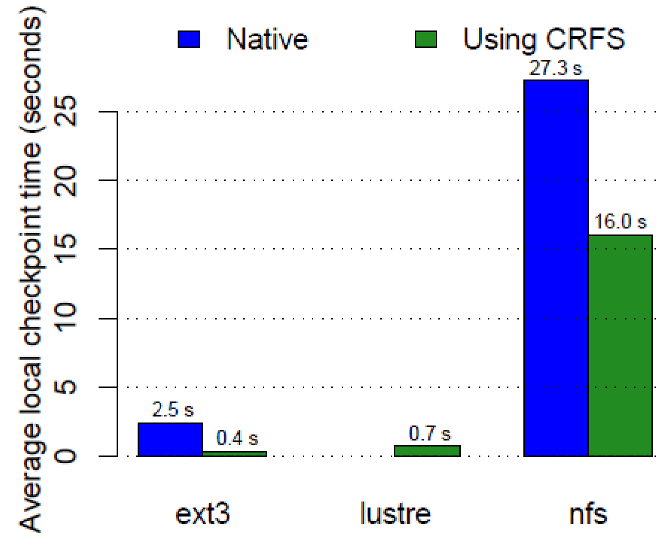
(c) LU.D.128

- **Lustre: CRFS is 11X / 9.3X / 1.3X faster than native with class B/C/D inputs**
- **Ext3: CRFS is 7X / 8X / 6.9X faster than native with class B/C/D inputs**

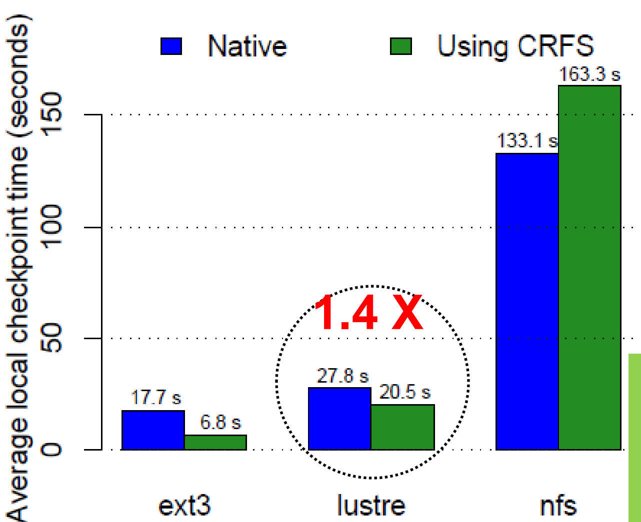
Checkpoint Time (OpenMPI)



(a) LU.B.128



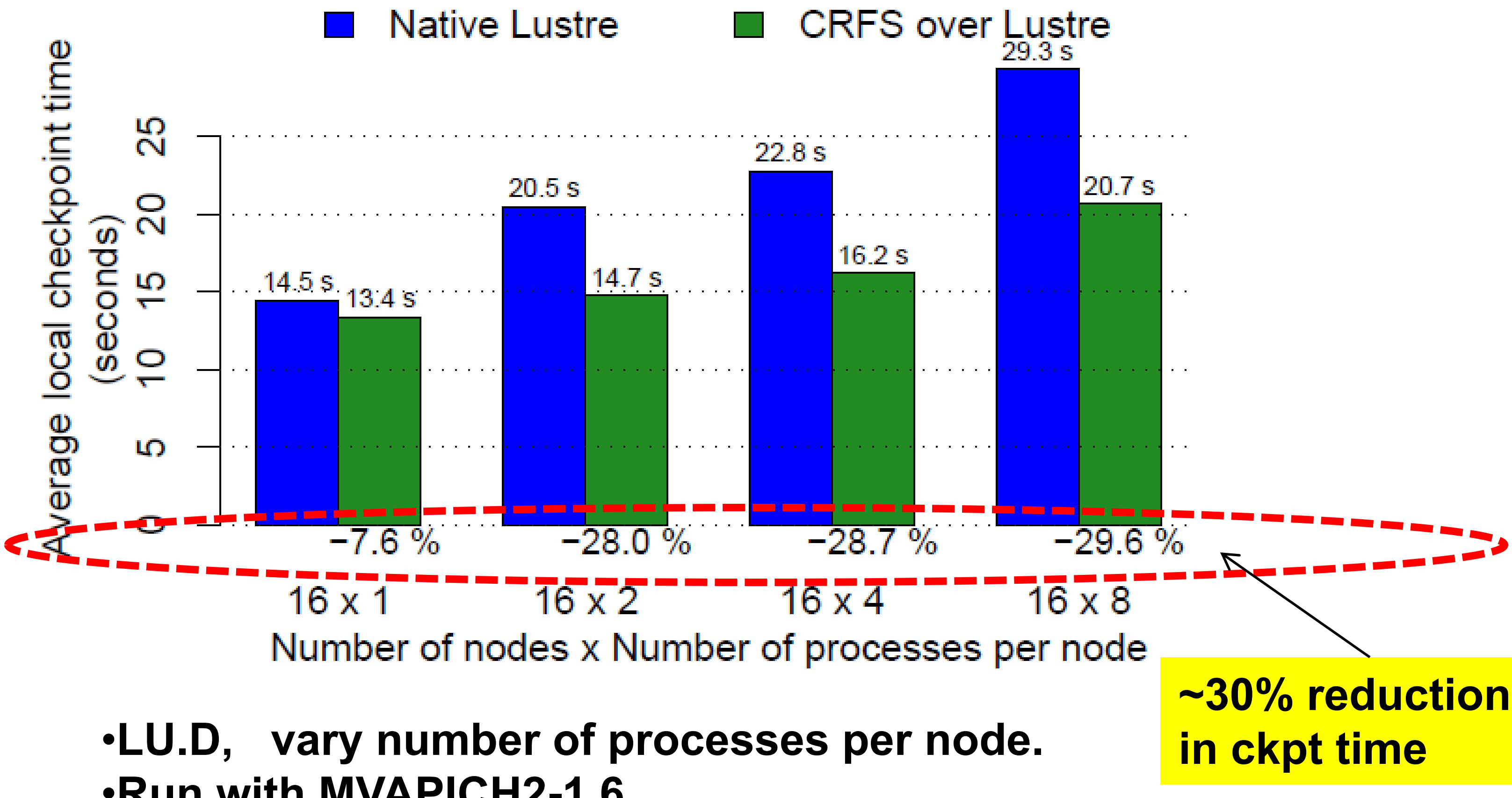
(b) LU.C.128



(c) LU.D.128

- **Lustre: CRFS is 11.5X / 1.4X faster than native with class B/D inputs**
- **Ext3: CRFS is 5.5X / 5.2X / 1.6X faster than native with class B/C/D inputs**

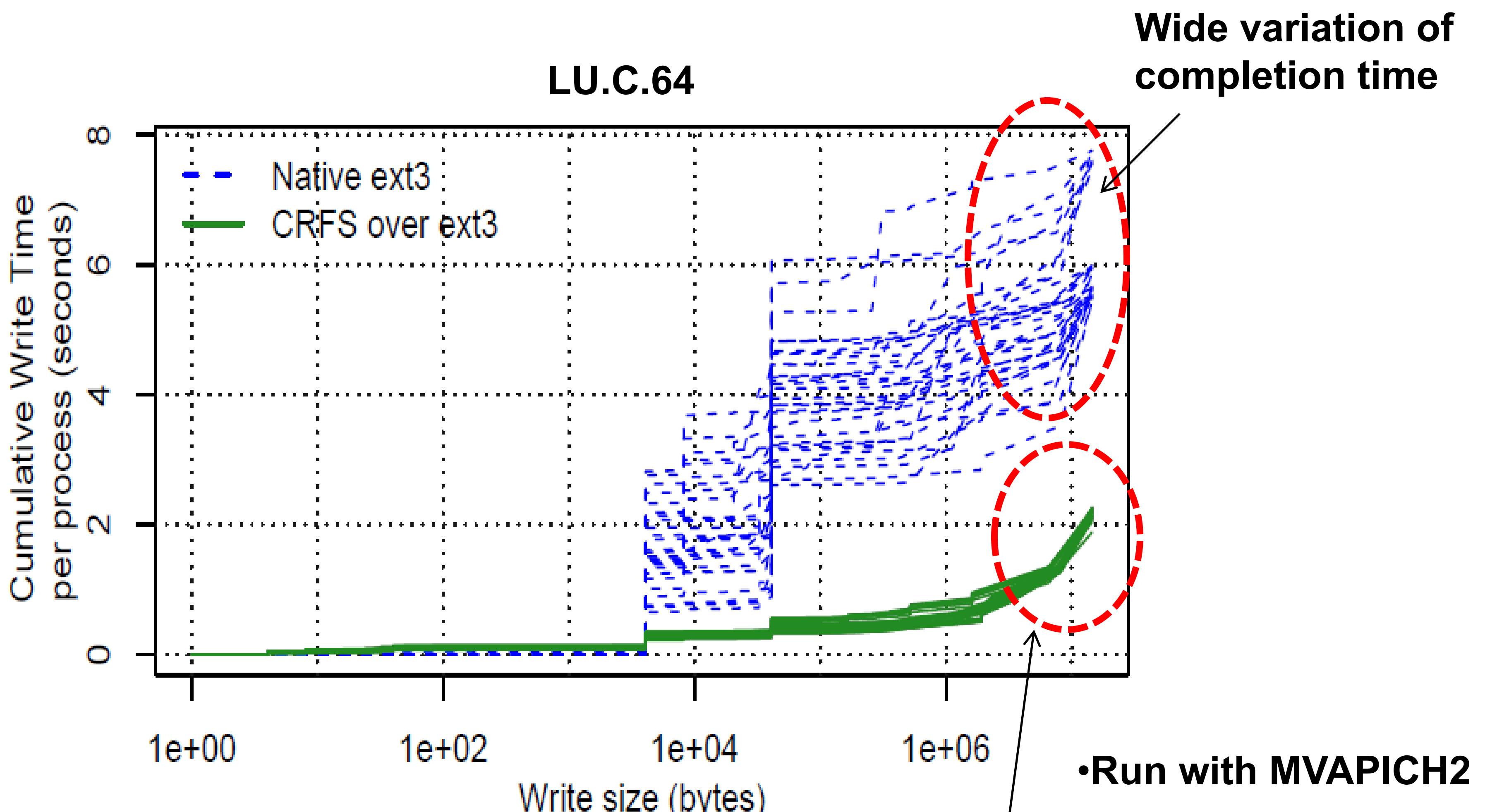
CRFS: Multiplexing Scalability



- LU.D, vary number of processes per node.
- Run with MVAPICH2-1.6
- Checkpoint to Lustre w/o CRFS

✓CRFS effectively reduces node-level IO multiplexing contention
✓Diminish checkpoint writing overhead

Checkpoint Completion Time



•**Run with MVAPICH2**

✓**CRFS diminishes IO contentions**

✓**Reduce the completion waiting ➔ faster resumption of execution**

Outline

- Introduction and Motivation
- Checkpoint Profiling and Analysis
- CRFS: A lightweight user-level Filesystem
- Performance Evaluation
- Conclusions and Future Work

Conclusions

- Checkpoint Writing incurs intensive IO overhead
  - Sheer amount of data, contentions from concurrent writes, wide variation of write completion
- Existing optimizations are not portable, not generic
- CRFS: a user-level filesystem
  - ✓ ***Portable***: a user-level filesystem, work with any MPI stacks without any modifications
  - ➢ ***High Performance***: write aggregation, reduced contention, asynchronous bulk IO
  - ✓ ***Generic***: Optimizations inside filesystem, can work with any MPI stacks / IO intensive applications

Future Work

- How to optimize inter-node concurrent IO using CRFS
- How to extend CRFS to benefit other generic IO intensive applications

Thank you!

MVAPICH

http://mvapich.cse.ohio-state.edu

{ouyangx, rajachan, besseron, wangh, huangjia, panda}@cse.ohio-state.edu

Network-Based Computing Laboratory

Related Work

- PLFS [1] (Parall Log Filesystem)
  - deal with N-1 sceanrio, cannot handle MPI system level checkpoint (N-N)
- Optimizations inside MPI library:
  - [2]: write aggregation at MPI and BLCR library
    - require modifications in kernel module, not portable
  - [3]: node-level buffering of data
    - specific to only one MPI stack

[1] J. Bent, G. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate, "PLFS: a checkpoint filesystem for parallel applications," in Proc. of SC '09, 2009.

[2] X. Ouyang, K. Gopalakrishnan, and D. K. Panda, "Accelerating Checkpoint Operation by Node-Level Write Aggregation on Multicore Systems," ICPP 2009.

[3] J. Hursey, J. Squyres, T. Mattox, and A. Lumsdaine, "The design and implementation of checkpoint/restart process fault tolerance for open mpi," in 12th IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems, 2007.