Fault-Tolerance Support in MVAPICH2

MVAPICH2 User Group (MUG) Meeting

by

Raghunath Rajachandrasekar

The Ohio State University

E-mail: rajachan@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~rajachan

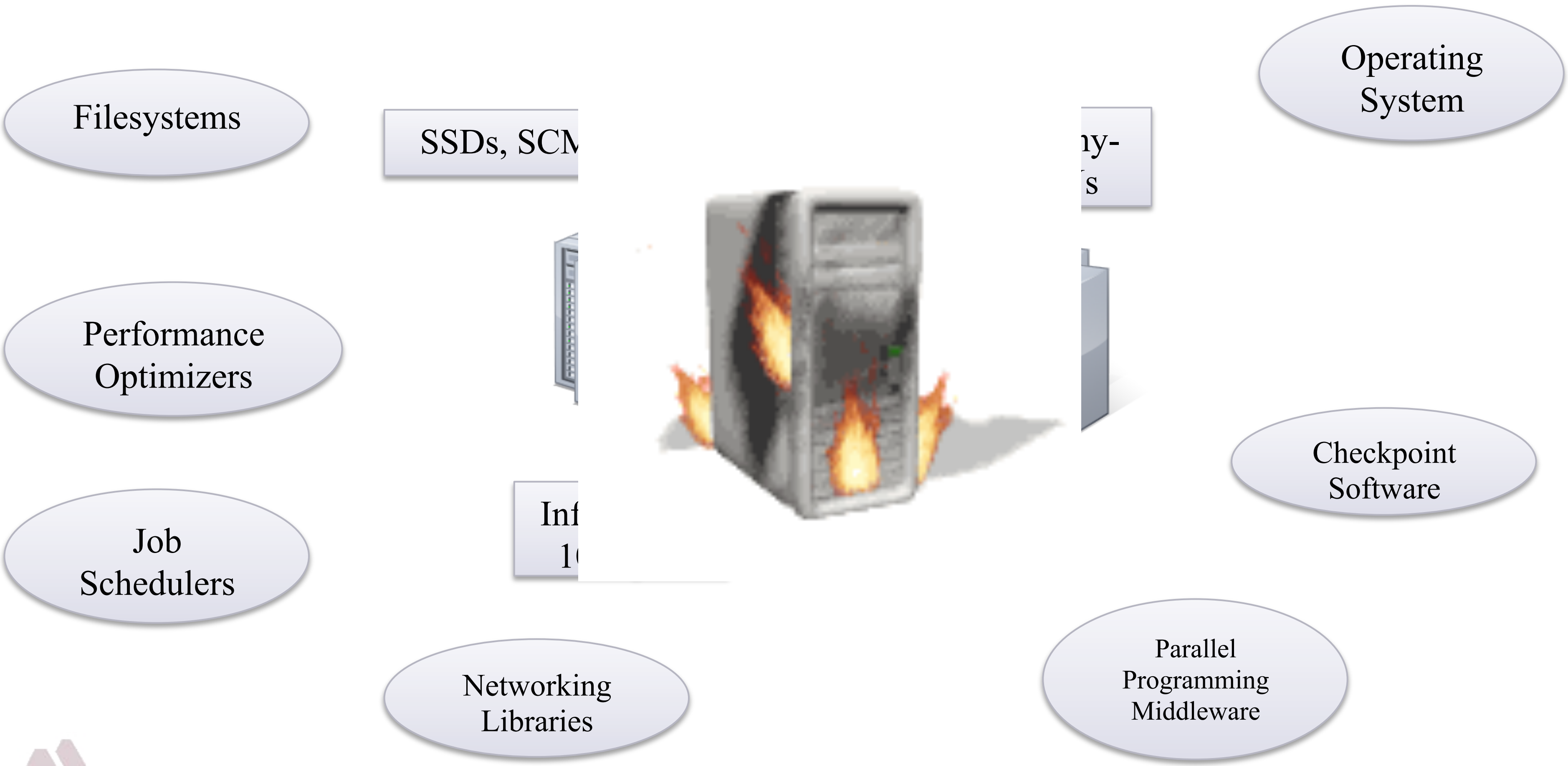
Agenda

- Introduction
- Checkpoint-Restart Schemes
- Process-Migration Schemes
- Automatic Path Migration
- Fault-Tolerance standardization effort in the MPI Forum
- Future directions

Why is Fault-Tolerance critical?

Tremendous increase in the HPC system sizes

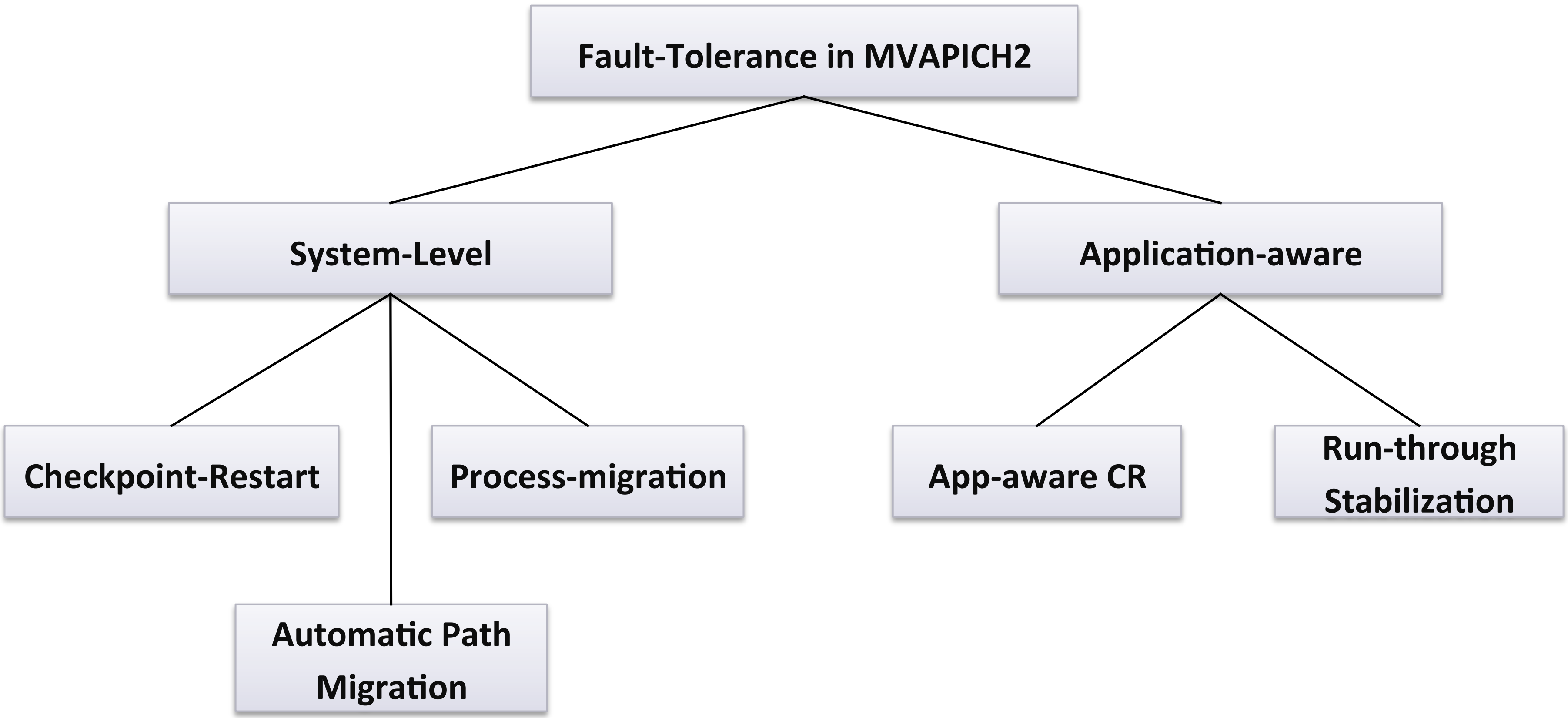
Explosive growth in the HPC software



Why is Fault-Tolerance critical?

- It is imperative to design resilient systems!
- Many of the s/w libraries and h/w architectures do tolerate failures – **but they act in isolation**
- System components should be able to correlate information from different sources to make informed decisions
- MVAPICH team's R&D driven by the need for:
  - Performance
  - Scalability
  - Productivity
  - **Fault-Tolerance**

Fault-Tolerance in MVAPICH2



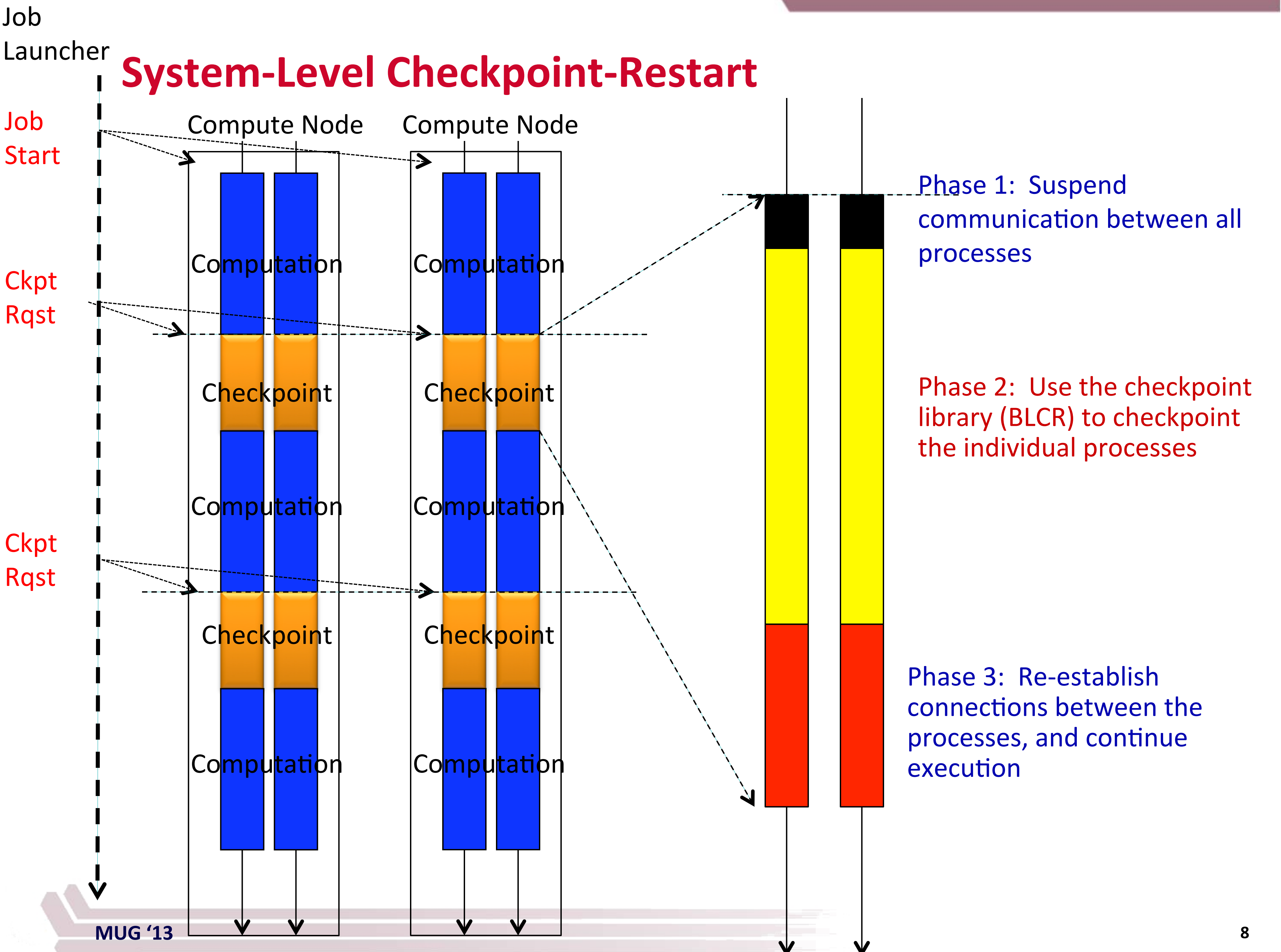
Fault-Tolerance in MVAPICH2

Feature	MVAPICH2 version	Release Year
BLCR-based system-level MPI application Checkpointing	0.9.8	2006
FTB-enabled Checkpoint-Restart	1.4	2008
FUSE-assisted Write-Aggregation Scheme	1.6	2010
Basic File-Copy based Process Migration	1.6	2010
Pipelined Process Migration using RDMA	1.7	2011
Checkpoint-Restart support for the Nemesis-IB channel	1.8	2012
Scalable Multi-level Checkpointing using SCR	1.9	2013
More features under development	2.x	2013

Agenda

- Introduction
- Checkpoint-Restart Schemes
  - System-Level Checkpoint Restart
  - Using the CR Feature
  - Multicore-Aware Checkpoint I/O Aggregation
  - Multi-Level Checkpointing with ScalableCR (SCR)
  - Quality-of-Service Aware Checkpoint-Restart
- Process-Migration Schemes
- Automatic Path Migration
- Future directions

System-Level Checkpoint-Restart



Using the Checkpoint-Restart Feature

- Requires Berkeley Lab Checkpoint-Restart (BLCR) library
- Build with CR support: --**enable-ckpt** (or) –with-blcr=$PATH_TO_BLCR_INSTALLATION
- Launching the job:

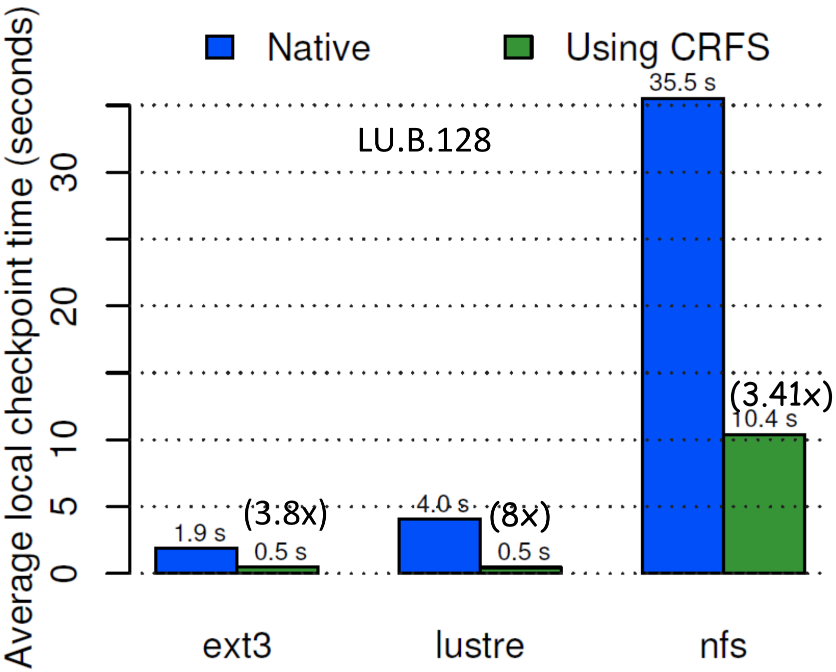
```
$mpirun_rsh -np 2 -hostfile ./hfile
        MV2_CKPT_FILE = /pfs/ckpt/app1
        MV2_CKPT_MAX_SAVE_CKPTS = 3
        MV2_CKPT_NO_SYNC = 0 ./a.out
```

- Triggering a checkpoint:
  - `$ cr_checkpoint -p <PID of mpirun_rsh>`
  - Run $MV2_INSTALL_DIR/bin/**mv2_checkpoint** and select the job to checkpoint
  - Call **MVAPICH2_Sync_Checkpoint()** from within the application
  - Set **MV2_CKPT_INTERVAL = 30** for automated checkpointing
- Restarting from a checkpoint:
  - `$cr_restart /pfs/ckpt/context.<pid>`

Ref: Section 6.15.1 of the MVAPICH2-2.0a User-guide

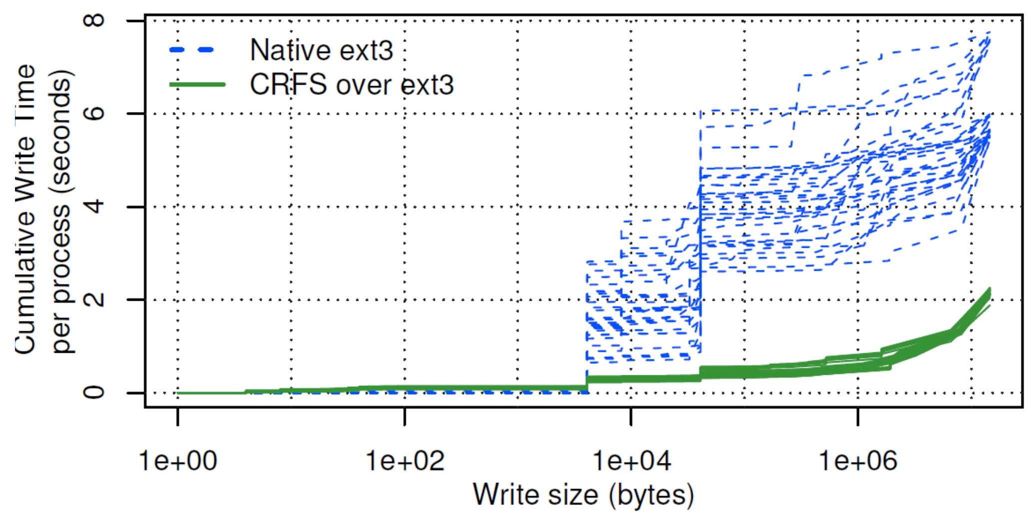
Multicore-aware Checkpoint I/O Aggregation

- Requires FUSE version 2.8+ , better performance for kernels newer than version 2.6.26
- Enable **–enable-ckpt-aggregation** or –with-fuse=<path_to_fuse_installation>
- Toggle at runtime using **MV2_CKPT_USE_AGGREGATION** variable
- Ensure that FUSE kernel module is loaded



(128 MPI processes on 16 nodes, 8 processes/node)

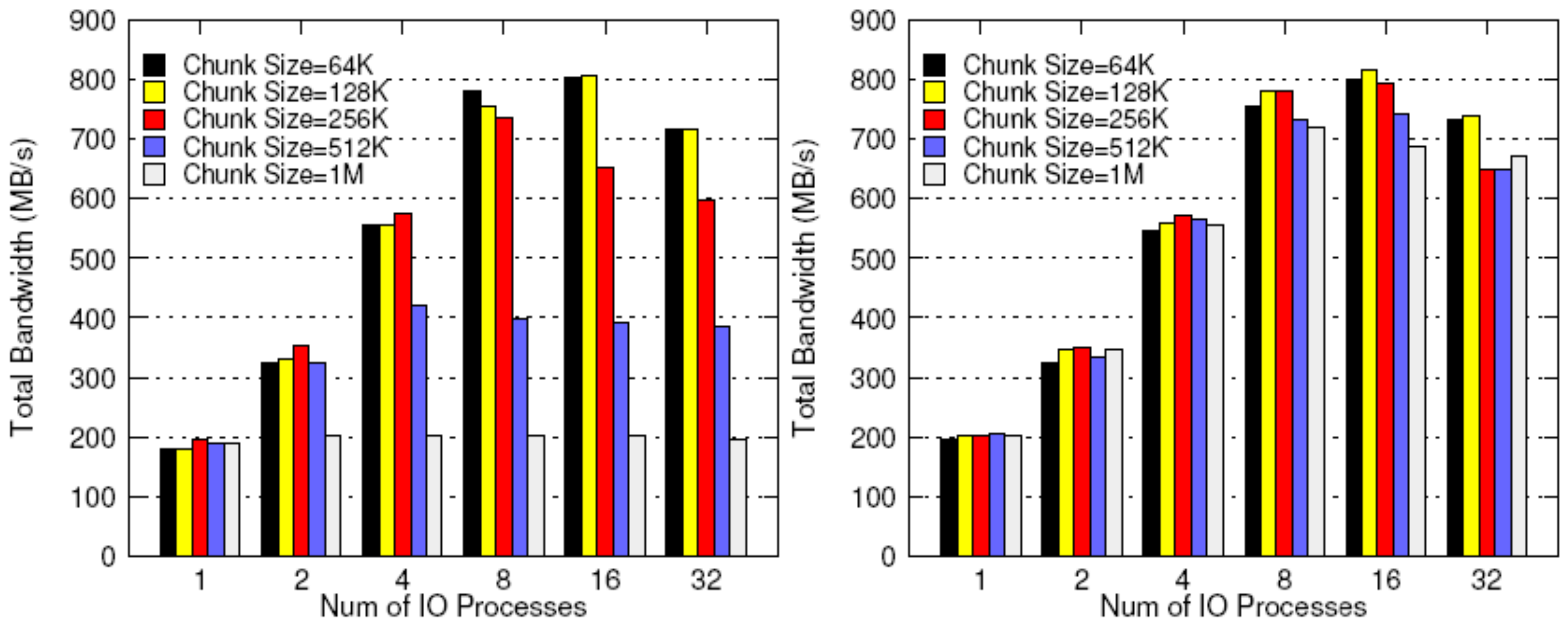
Cumulative Write time (LU.C.64)
(64 MPI processes on 8 nodes, 8 processes/node)



X. Ouyang, R. Rajachandrasekar, X. Besseron, H. Wang, J. Huang and D. K. Panda, CRFS: A Lightweight User-Level Filesystem for Generic Checkpoint/Restart, Int'l Conference on Parallel Processing (ICPP '11), Sept. 2011.

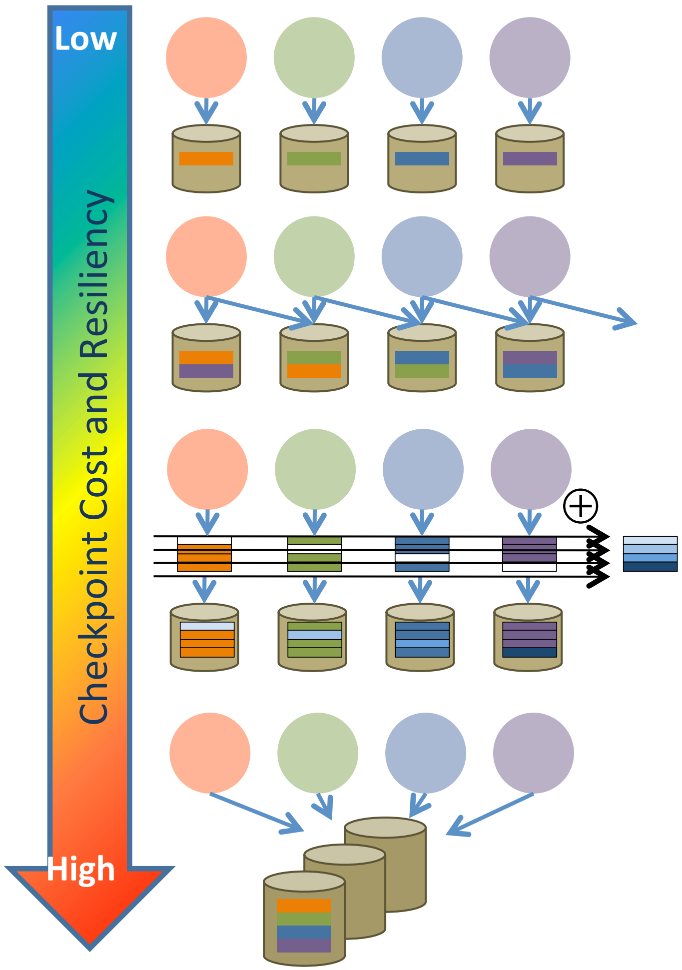
Multicore-aware Checkpoint I/O Aggregation

- Tunables
  - MV2_CKPT_AGGREGATION_BUFPOOL_SIZE (size of buffer pool used to aggregate I/O)
  - MV2_CKPT_AGGREGATION_CHUNK_SIZE (chunks in which coalesced data is written to disk)



(a) Buffer Pool=1MB (b) Buffer Pool=8MB

Multi-Level Checkpointing with ScalableCR (SCR)

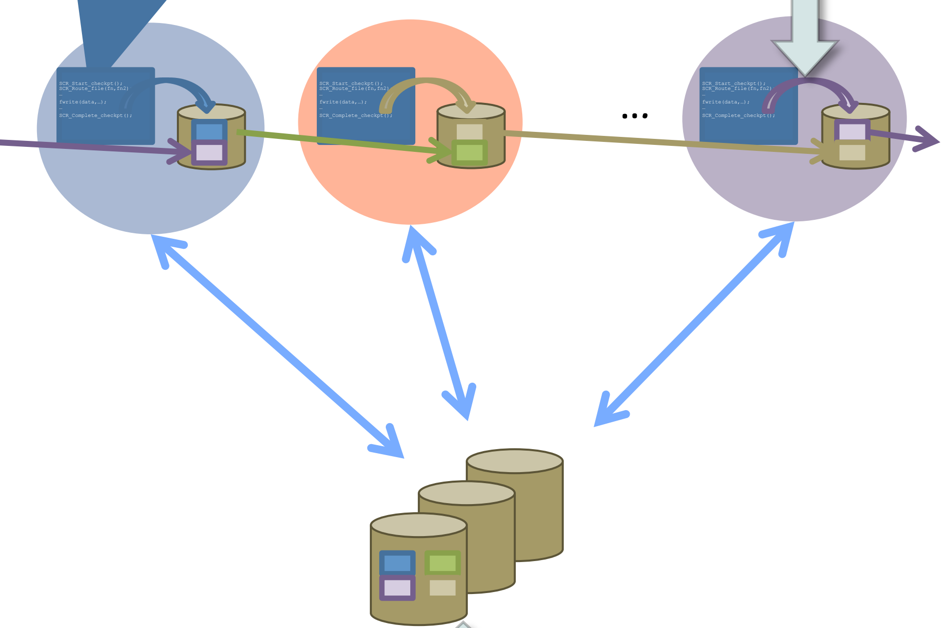


- LLNL's Scalable Checkpoint/Restart library
- Can be used for application guided and application transparent checkpointing
- Effective utilization of storage hierarchy
  - **Local:** Store checkpoint data on node's local storage, e.g. local disk, ramdisk
  - **Partner:** Write to local storage and on a partner node
  - **XOR:** Write file to local storage and small sets of nodes collectively compute and store parity redundancy data (RAID-5)
  - **Stable Storage:** Write to parallel file system

Application-guided Multi-Level Checkpointing

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

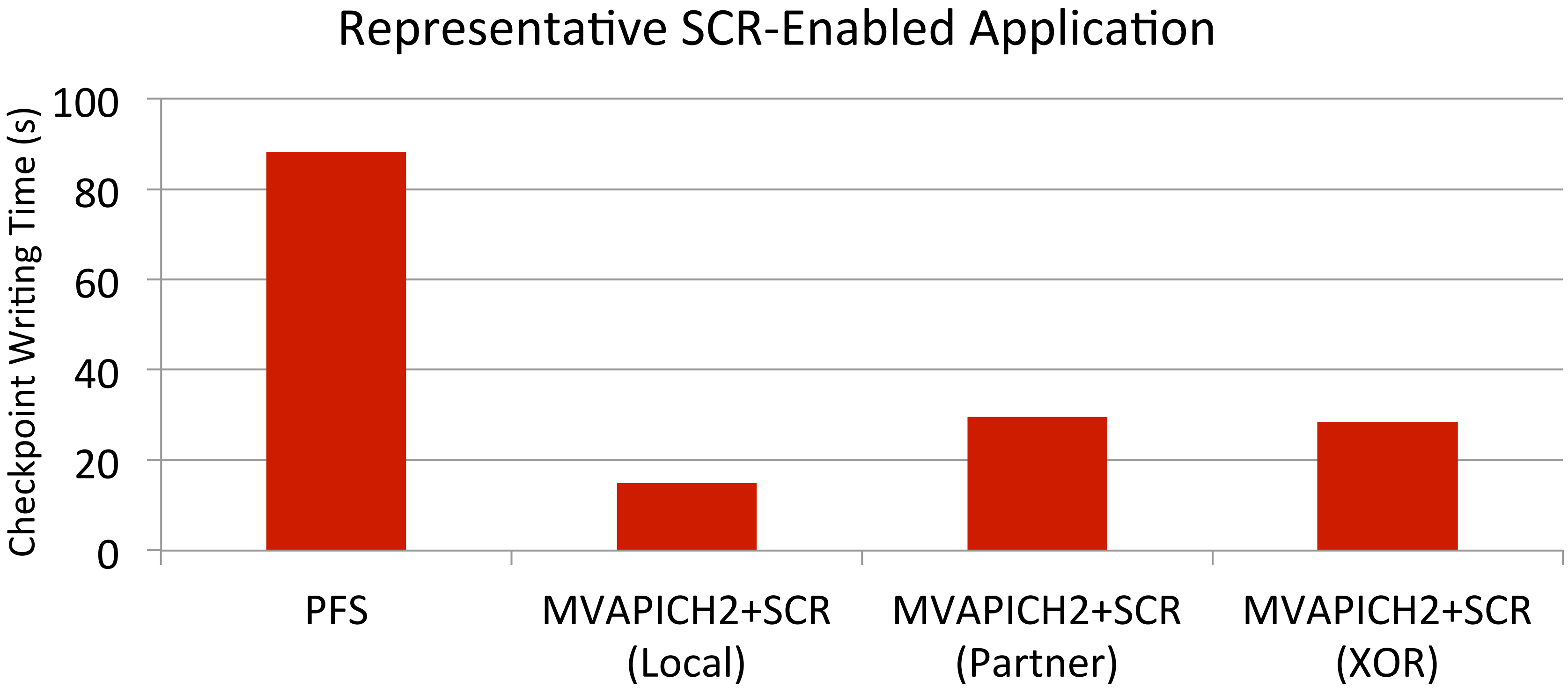
• First write checkpoints to node-local storage
• When checkpoint is complete, apply redundancy schemes



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

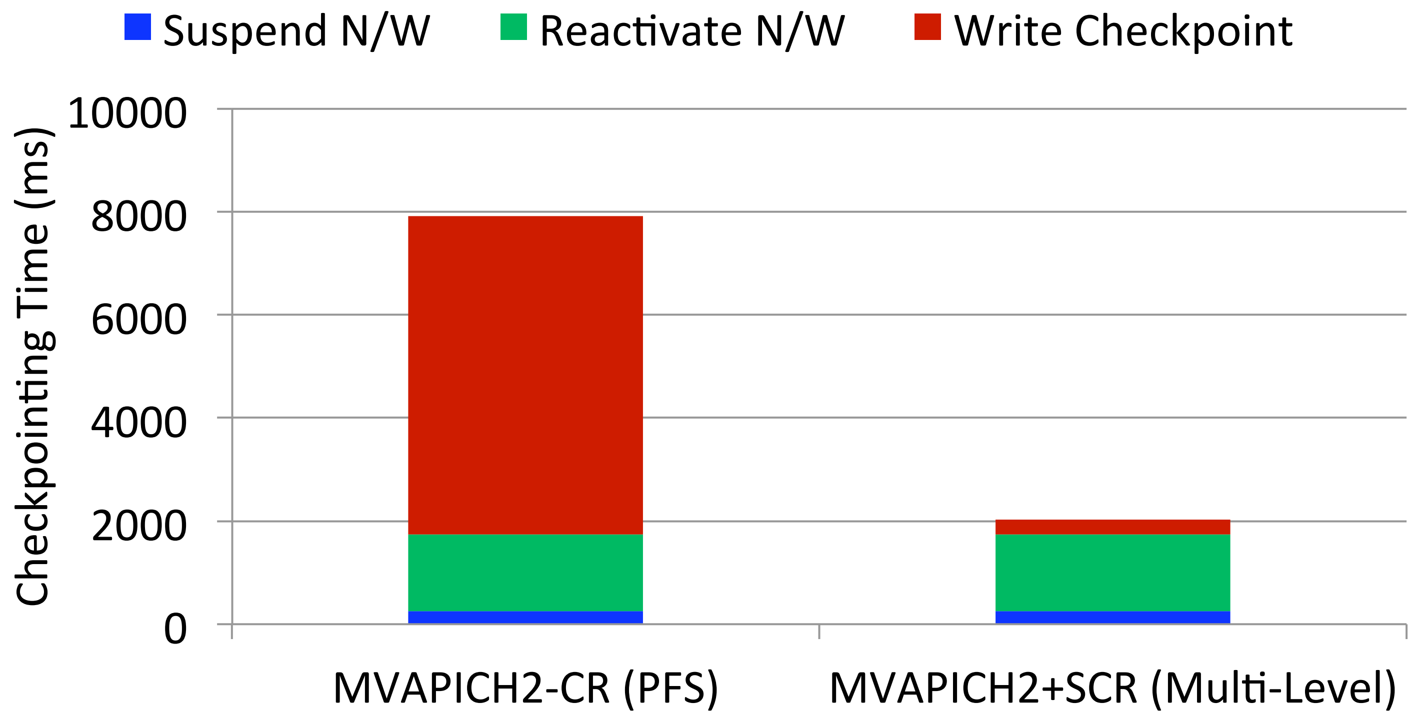
```
void checkpoint() {
  SCR_Start_checkpoint();

  int rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  char file[256];
  sprintf(file, "rank_%d.ckpt", rank);

  char scr_file[SCR_MAX_FILENAME];
  SCR_Route_file(file, scr_file);
  FILE* fs = fopen(scr_file, "w");
  if (fs != NULL) {
    fwrite(state, ..., fs);
    fclose(fs);
  }

  SCR_Complete_checkpoint(1);
  return;
}
```

Application-guided Multi-Level Checkpointing



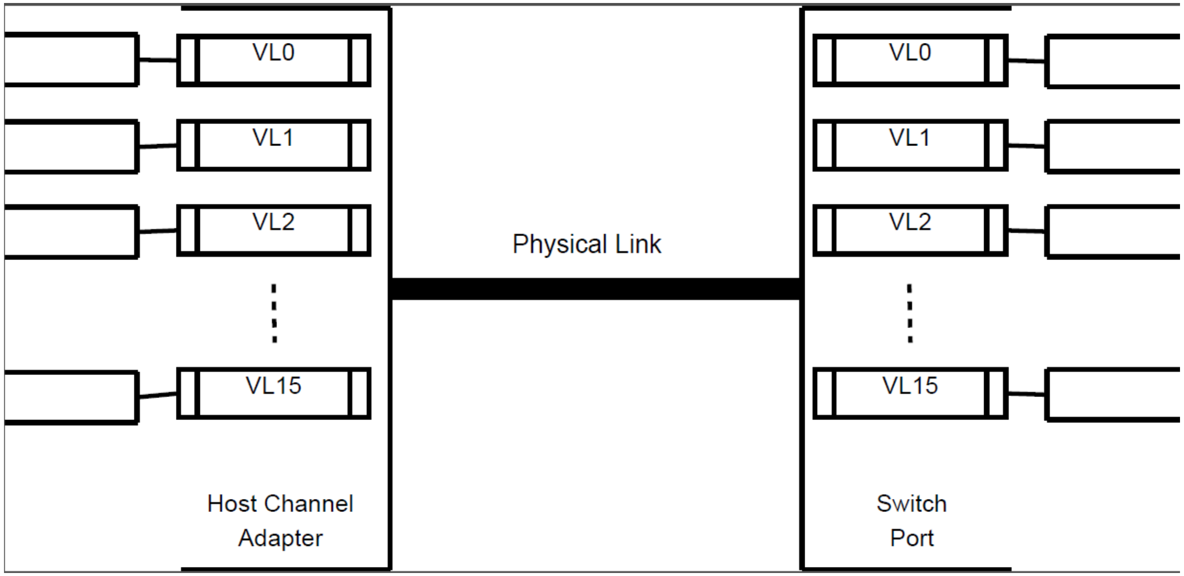
- Checkpoint writing phase times of representative SCR-enabled MPI application
- **512** MPI processes (8 procs/node)
- Approx. **51 GB** checkpoints

Transparent Multi-Level Checkpointing



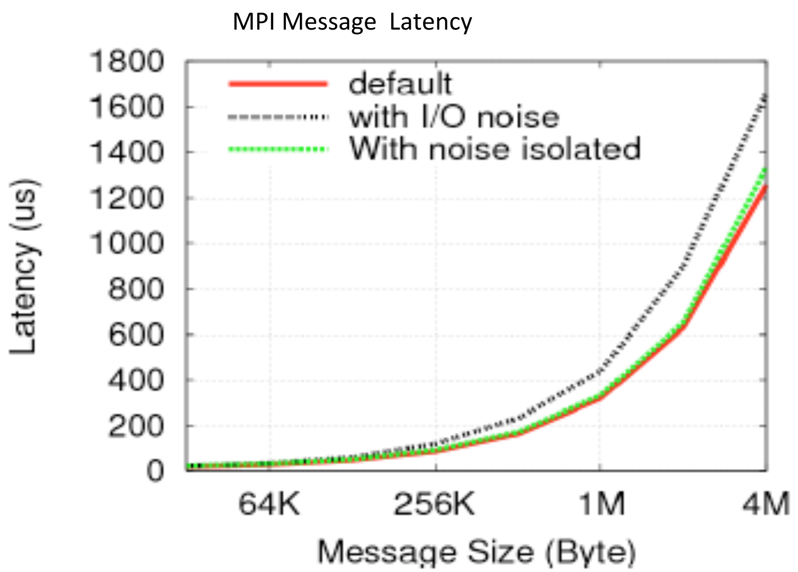
- **ENZO Cosmology application** – Radiation Transport workload
- Using MVAPICH2's CR protocol instead of the application's in-built CR mechanism
- **512** MPI processes (8 procs/node)
- Approx. **12.8 GB** checkpoints

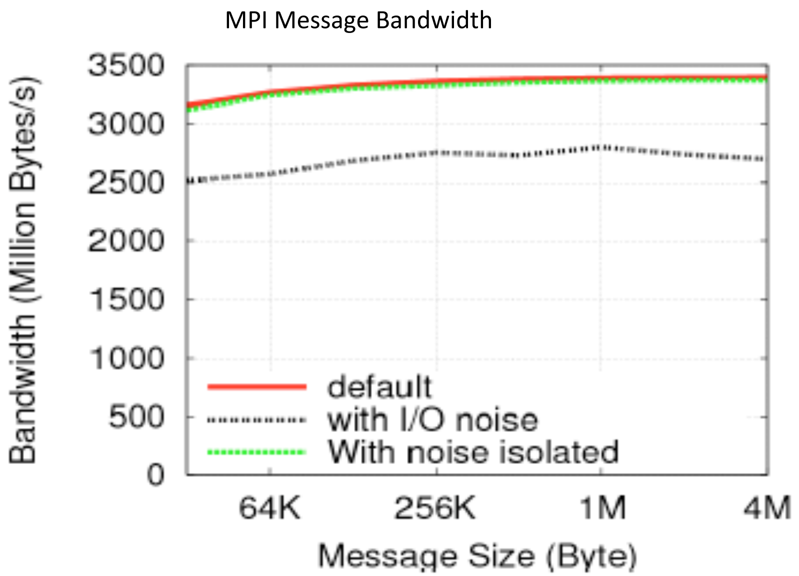
Quality-of-Service Aware Checkpoint-Restart

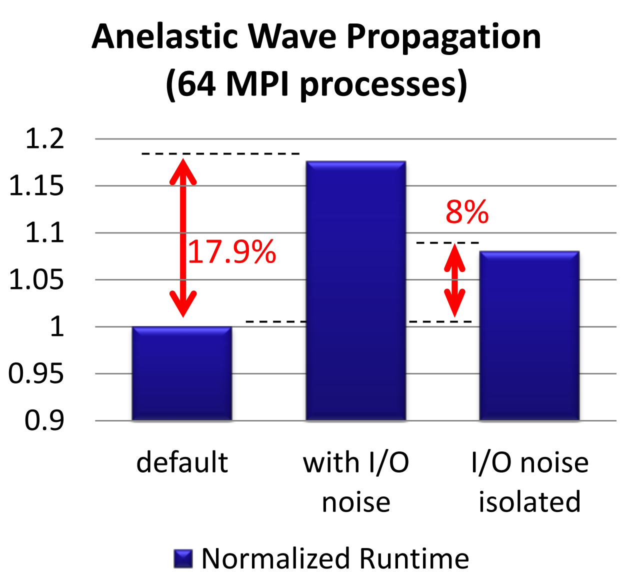


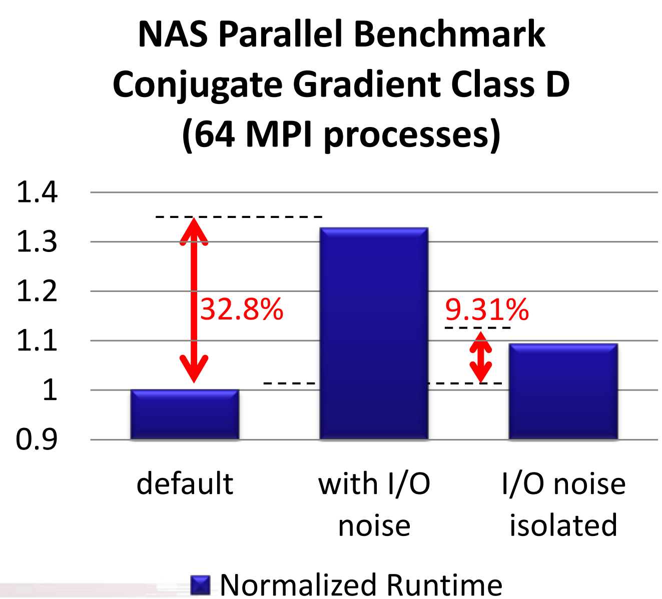
- QoS to increase or limit priority of different data-flows
- Multiple virtual 'lanes' share the same physical link
- Exclusive buffering and flow-control for each virtual lane
- Abstractions to configure priority
  - Service Level (SL) at switch-level
  - Traffic Class (TClass) at the router-level
- **opensm.conf:** SL-VL mapping and VL arbitration

Quality-of-Service Aware Checkpoint-Restart





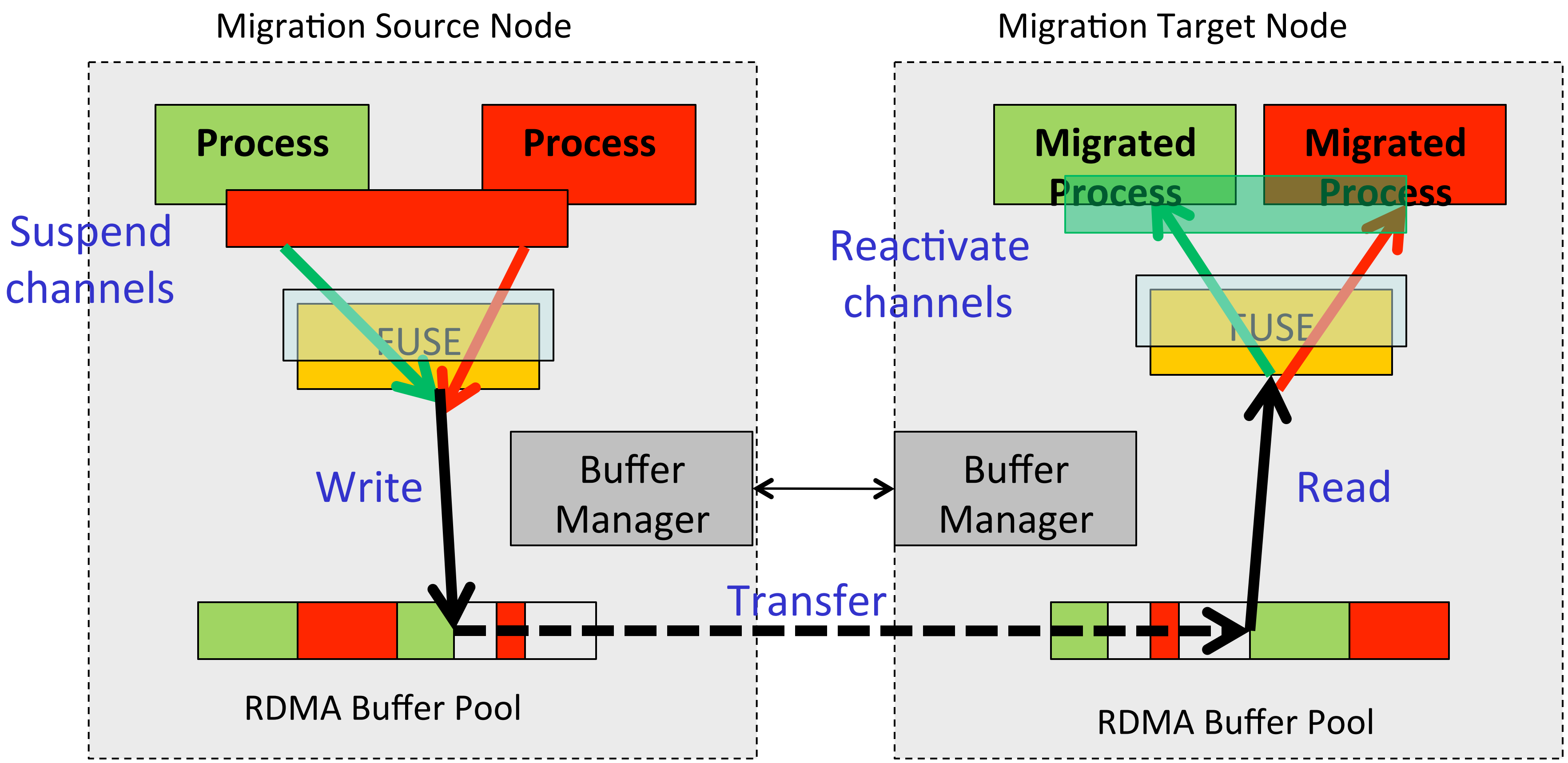




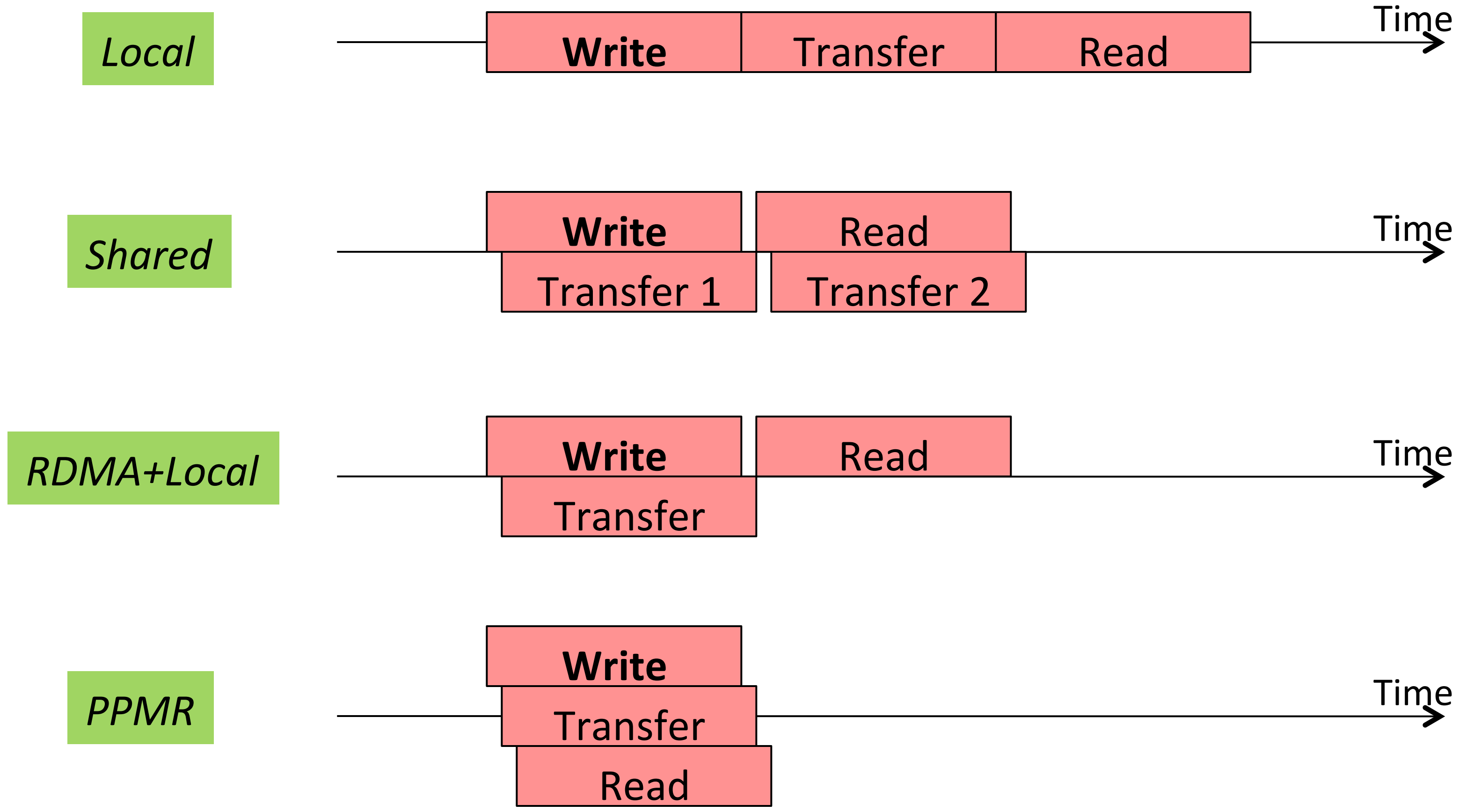
Agenda

- Introduction
- Checkpoint-Restart Schemes
- Process-Migration Schemes
  - RDMA-Based Process Migration
  - Using the Process-Migration Feature
  - Low-Overhead Failure Prediction with FTB-IPMI
- Automatic Path Migration
- Future directions

RDMA-based Pipelined Process-Migration



RDMA-based Pipelined Process-Migration



Using the Process-Migration Feature

- Requires BLCR, Fault-Tolerance Backplane (FTB), and FUSE (RDMA-based)
- Build with Migration support: **--enable-ckpt-migration**
- Setup FTB and launch the job:

```
$mpirun_rsh -np 4 -hostfile ./hosts -sparehosts ./spares ./a.out
```

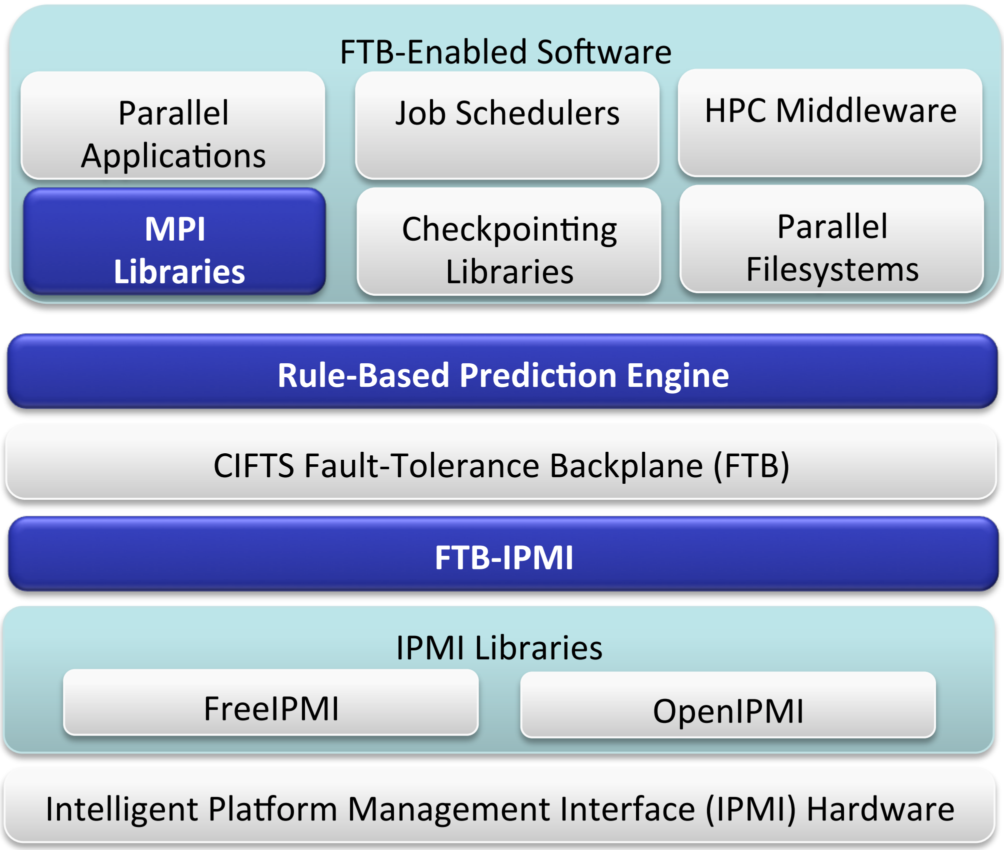
- Triggering a migration:
  - Send **SIGUSR2** signal to '**mpispawn** 'on source/failing node
  - `$MV2_INSTALL_PATH/bin/mv2_trigger <hostname_of_source_node>`
  - Automatically triggered by **FTB-IPMI** available at

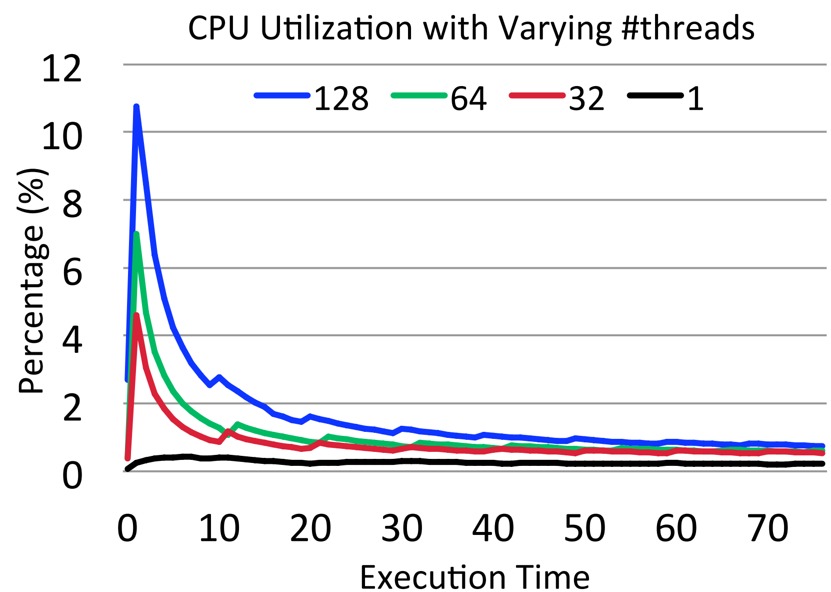
    http://nowlab.cse.ohio-state.edu/projects/ftb-ib/#FTB-IPMI

Ref: Section 6.15.3 of the MVAPICH2-2.0a User-guide

Low-Overhead Failure Prediction with IPMI

- Real-time failure prediction needed for proactive fault-tolerance mechanisms like process migration
- System-wide failure information coordination necessary to make informed decisions
- FTB-IPMI – provides low-overhead distributed fault-monitoring and failure event propagation

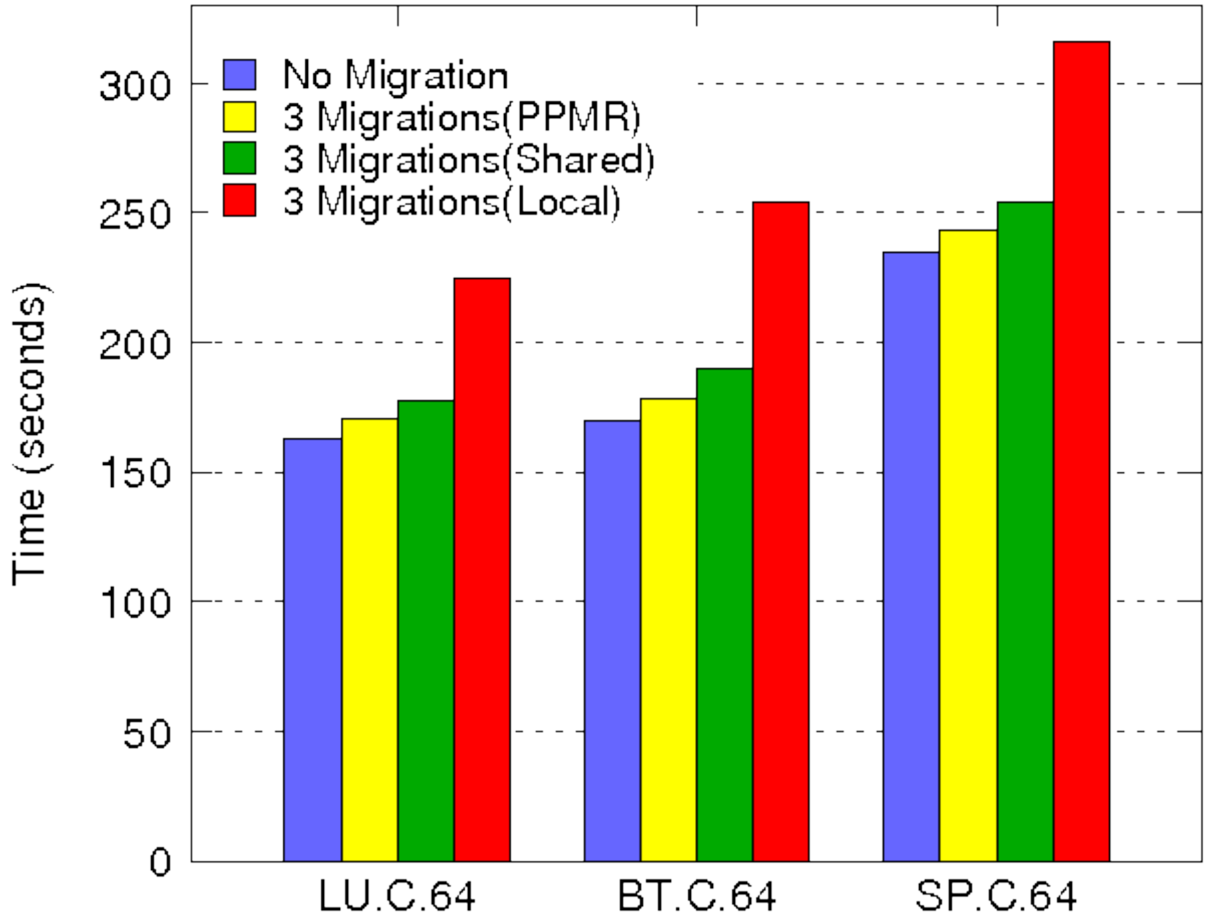




- Iteration delay – 10secs; 128-node tasklist
- Avg CPU utilization – 0.35%
- Single iteration of sensor sweep – 0.75 seconds

R. Rajachandrasekar, X. Besseron and D. K. Panda, Monitoring and Predicting Hardware Failures in HPC Clusters with FTB-IPMI, Int'l Workshop on System Management Techniques, Processes, and Services ; in conjunction with IPDPS '12, May 2012

Performance of Pipelined Process-Migration



X. Ouyang, R. Rajachandrasekar, X. Besseron, D. K. Panda, High Performance Pipelined Process Migration with RDMA, CCGrid 2011

Network-Level FT with Automatic Path Migration (APM)

- Allows recovery from network faults in the presence of multiple paths
- Enabled by the LID-Mask Count (LMC) mechanism
- Run with APM support:
  - `$ mpirun_rsh -np 2 host1 host2 MV2_USE_APM=1 ./a.out`
- Test APM in the absence of actual network faults:
  - `$ mpirun_rsh -np 2 host1 host2 MV2_USE_APM=1 MV2_USE_APM_TEST=1 ./a.out`
  - Periodically migrates between primary and alternate paths

Ref: Section 6.15.5 of the MVAPICH2-2.0a User-guide

Fault-Tolerance standardization effort in the MPI Forum

- FT working-group working on a proposal
- Earlier proposals did not make it to MPI 3.0
  - https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/FaultToleranceWikiPage
- Current Proposal for MPI 3.1/4.0 (ULFM)
  - Process failures
    - Explicitly handle fail-stop failures
  - Silent (memory) errors & Byzantine errors are outside of the scope
  - Failure detectors are very specific to the system they run on
    - Some systems may have hardware support for monitoring
    - All systems can fall back to arbitrary/configurable timeouts if necessary
  - Minimal set of tools for MPI FT
    - Failure Notification
    - Failure Propagation
    - Failure Recovery
    - Fault Tolerant Consensus

Run-Through Stabilization

- Proposal made to the MPI Forum's FT working group pre-3.0
- Communication failures not treated as fatal errors
- Return error code on process failure to user-set handler
- Outstanding send/recv/wild-card recv (with MPI_ANY_SOURCE) posted to failed communicator returns error code
- Supported in the Nemesis-IB channel (**--with-device=ch3:nemesis:ib**)
- Run with mpiexec.hydra
  - Set **MV2_RUN_THROUGH_STABILIZATION = 1**
  - Add **--disable-auto-cleanup** flag
- Query list of failed processes from application:
  - `MPI_Comm_get_attr(MPI_COMM_WORLD, MPICH_ATTR_FAILED_PROCESSES, &failed_procs, &flag);`

https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/276

Future Directions

- In-memory checkpointing with SCR
- Support for more resource managers (Slurm, Torque, etc)
- Incremental checkpointing
- Checkpoint compression
- Parity-based process-snapshot migration using SCR
- N-N vs N-1 checkpointing schemes

Web Pointers

NOWLAB Web Page

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

MVAPICH Web Page

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

MVAPICH