Process-Level Fault Tolerance for Job Healing in HPC Environments

Chao Wang, Frank Mueller

Department of Computer Science

North Carolina State University

Stephen L. Scott, Christian Engelmann

Systems Research Team (SRT)

Computer Science Group

Oak Ridge National Laboratory

HPC Resiliency Summit: Workshop on Resiliency for Petascale HPC
Los Alamos Computer Science Symposium
October 15, 2008, Santa Fe, New Mexico

Outline

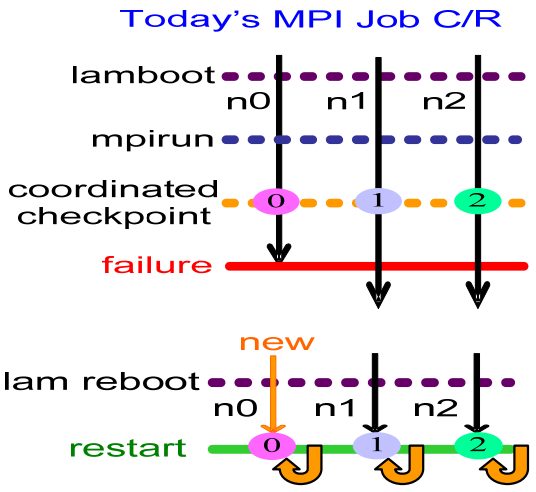
- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Problem Statement

- Trends in HPC: high end systems with > 100,000 processors
  - MTBF/I becomes shorter
- MPI widely accepted in scientific computing
  - But no fault recovery method in MPI standard
- Transparent C/R:
  - Coordinated: LAM/MPI w/ BLCR *[LACSI '03]*
  - *Uncoordinated,* Log based: MPICH-V *[SC 2002]*
- Non-transparent C/R: Explicit invocation of checkpoint routines
  - LA-MPI *[IPDPS 2004]* / FT-MPI *[EuroPVM-MPI 2000]*
- Frequently deployed C/R helps but…
  - 60% overhead on C/R *[I.Philp HPCRI'05]*
    - 100 hrs job -> 251 hrs
  - Must restart all job tasks
    - Inefficient if only one (few) node(s) fails
    - Staging overhead
  - Requeuing penalty



Our Solution

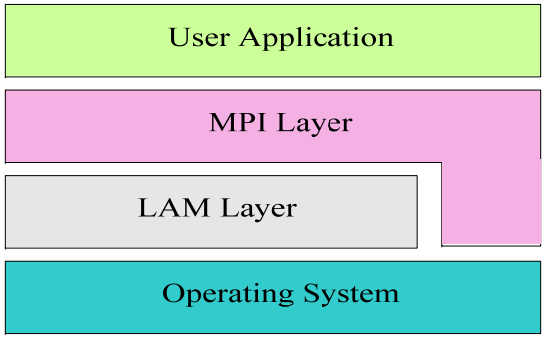
- Job-pause service – reactive migration
  - Integrate group communication to detect node failure
  - Processes on live nodes remain active (roll back to last checkpoint)
  - Only processes on failed nodes dynamically replaced by spares (resumed from the last checkpoint)
- Proactive live migration with failure prediction
  - Processes on live nodes remain active (no rollback, no checkpoint)
  - Only processes on "unhealthy" nodes are lively migrated to spares
- Hybrid full/incremental checkpoint/restart
  - Only the subset of modified pages checkpointed (incremental)
  - Incremental checkpoints complement full checkpoints
  - Fast restart (any page only written once)

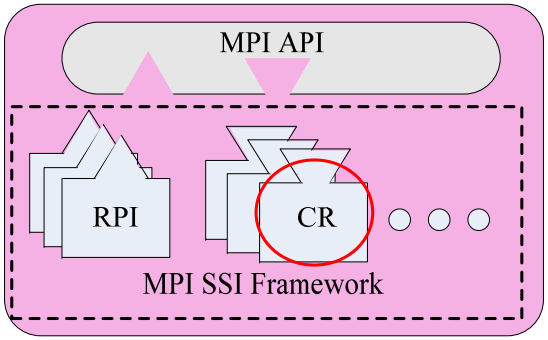
Outline

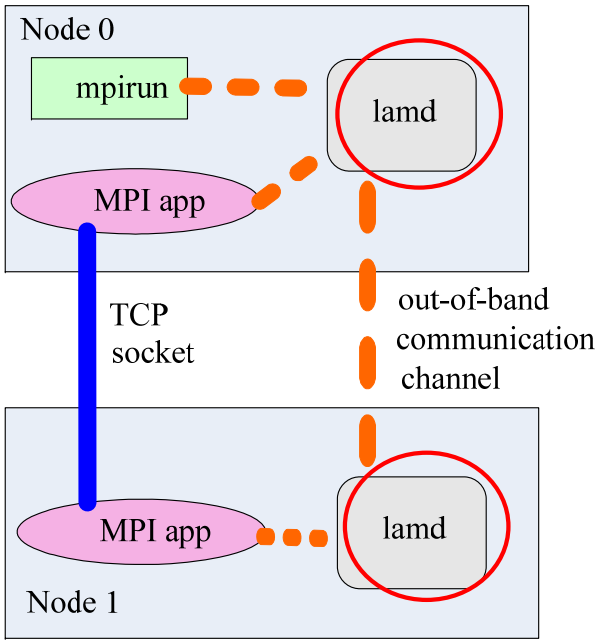
- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

LAM-MPI Overview

- Modular, component-based architecture
  - 2 major layers
  - Daemon-based RTE: lamd
  - "Plug in" C/R to MPI SSI framework:
  - Coordinated C/R & support BLCR







- Example: A two-way MPI job on two nodes

RTE: Run-time Environment
SSI: System Services Interface
RPI: Request Progression Interface
MPI: Message Passing Interface
LAM: Local Area Multi-computer

BLCR Overview

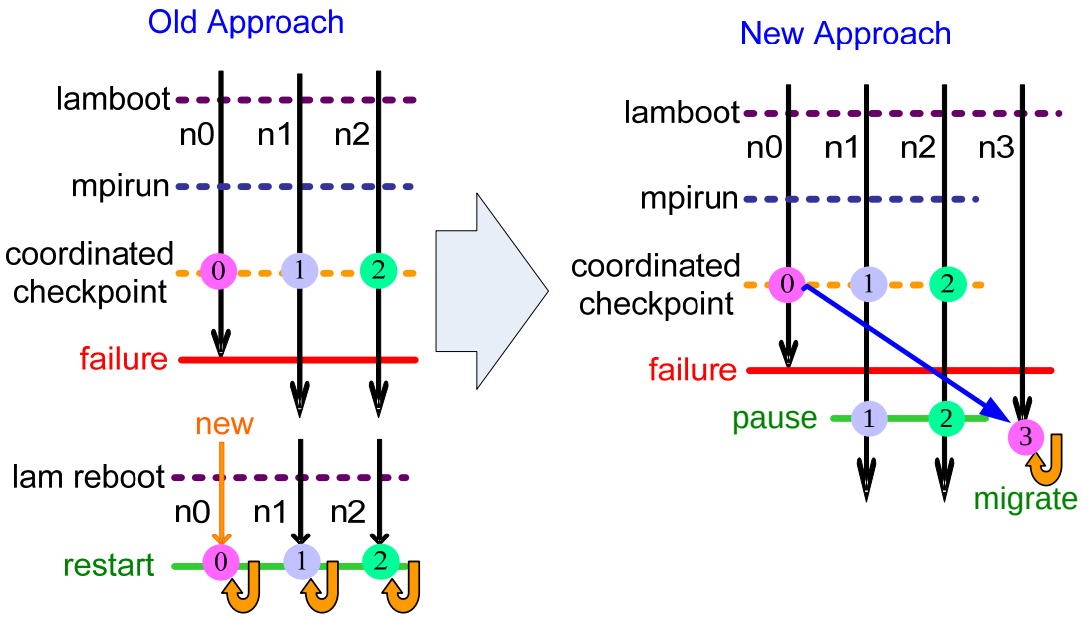
- Kernel-based C/R: Can save/restore almost all resources
- Implementation: Linux kernel module, allows upgrades & bug fixes w/o reboot
- Process-level C/R facility: single MPI application process
- Provides hooks used for distributed C/R: LAM-MPI jobs

Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Phase 1: a Job-Pause Service

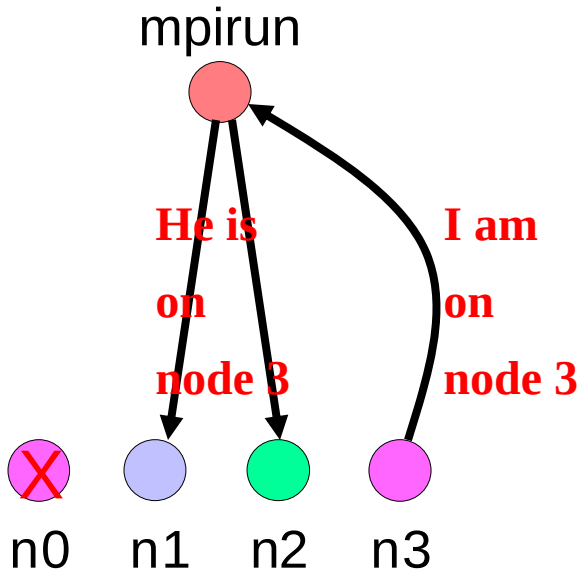
- Integrate group communication
  - Add/delete nodes
  - Detect node failures automatically
- Processes on live nodes remain active (roll back to last checkpoint)
- Only processes on failed nodes dynamically replaced by spares
  - resumed from the last checkpoint



- Hence:
  - no restart of entire job
  - no staging overhead
  - no job requeue penalty
  - no Lam RTE reboot

Process Migration – LAM/MPI

- Change addressing information of migrated process
  - in process itself
  - in all other processes
- Use node id (not IP) for addressing information
- Update addressing information at run time
  1. Migrated process tells coordinator (mpirun) about its new location
  2. Coordinator broadcasts new location
  3. All processes update their process list
- No change to BLCR for Process Migration

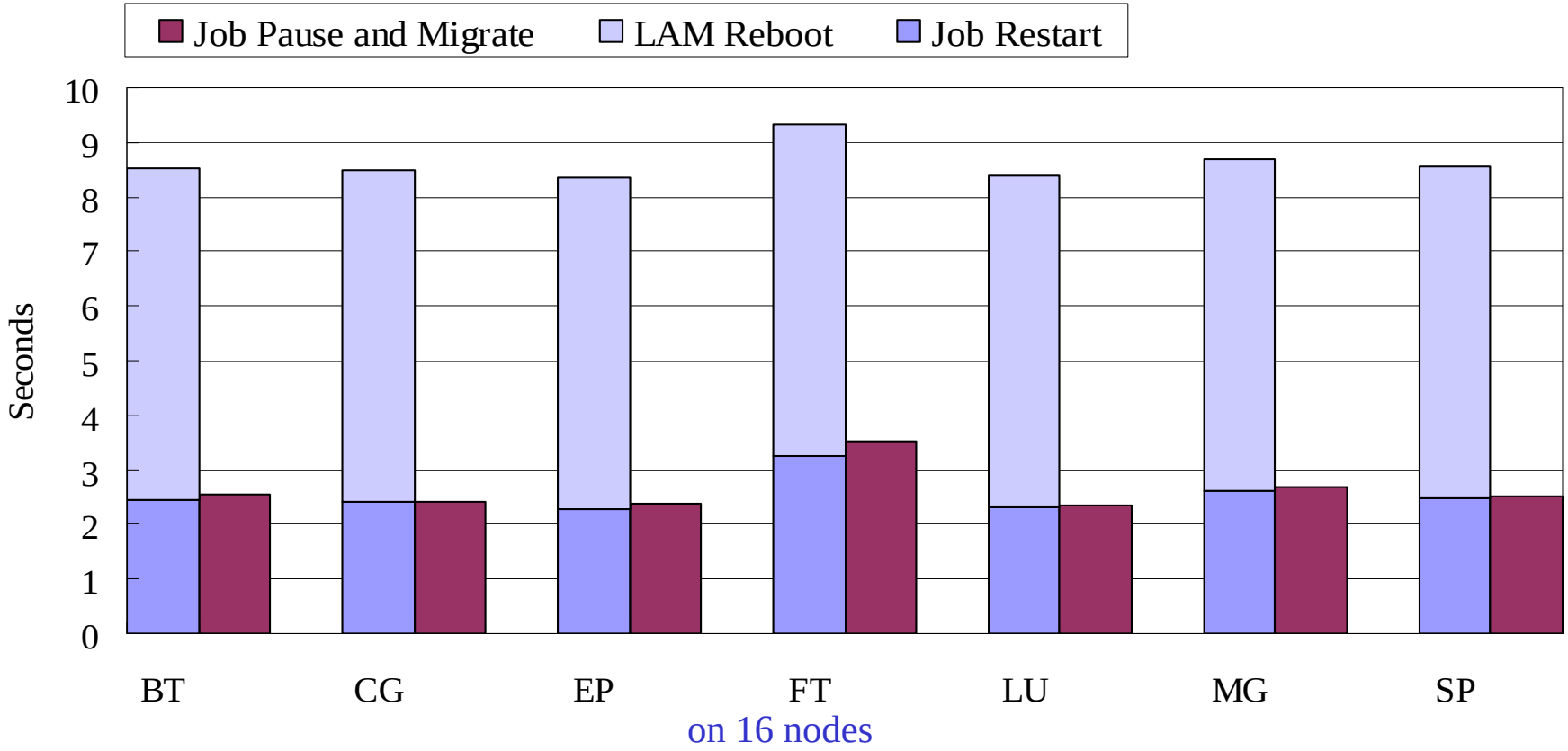


Experimental Framework

- Experiments conducted on
  - Opt cluster: 16 nodes, 2 core, dual Opteron 265, 1 Gbps Ether
  - Fedora Core 5 Linux x86_64
  - Lam/MPI + BLCR w/ our extensions

- Benchmarks
  - NAS V3.2.1 (MPI version)
    - run 5 times, results report avg.
    - Class C (large problem size) used
    - BT, CG, EP, FT, LU, MG and SP benchmarks
    - IS run is too short

Job Migration Overhead



69.6% < job restart + lam reboot

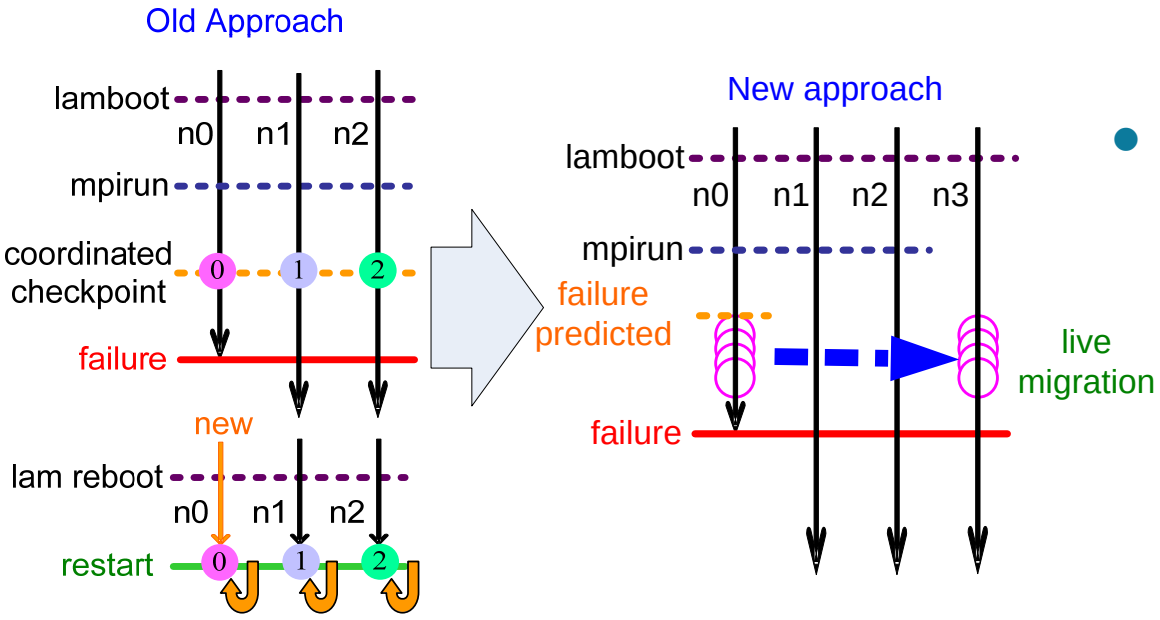
- NO LAM Reboot
- No requeue penalty
- Transparent continuation of exec
  - Less staging overhead

Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Phase 2: Proactive Live Migration

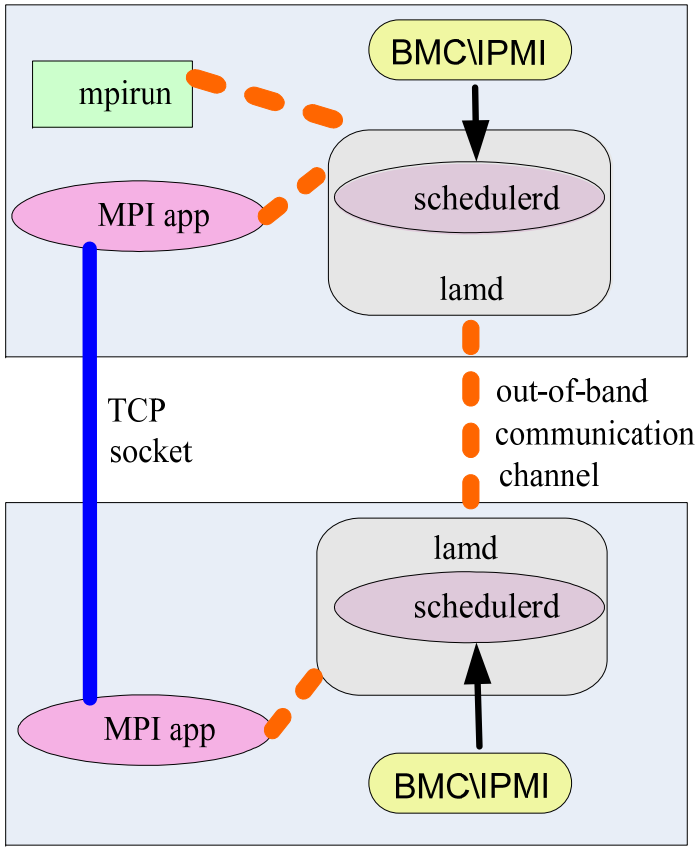
- High failure prediction accuracy with a prior warning window:
  - >70% reported [Gu et. Al, ICPP'08] [R.Sahoo et.al KDD '03]
  - Active research field
  - Premise for live migration
- Processes on live nodes remain active
- Only processes on "unhealthy" nodes are lively migrated to spares



- Hence, avoid:
  - High overhead on C/R
  - Restart of entire job
    - Staging overhead
  - Job requeue penalty
  - Lam RTE reboot

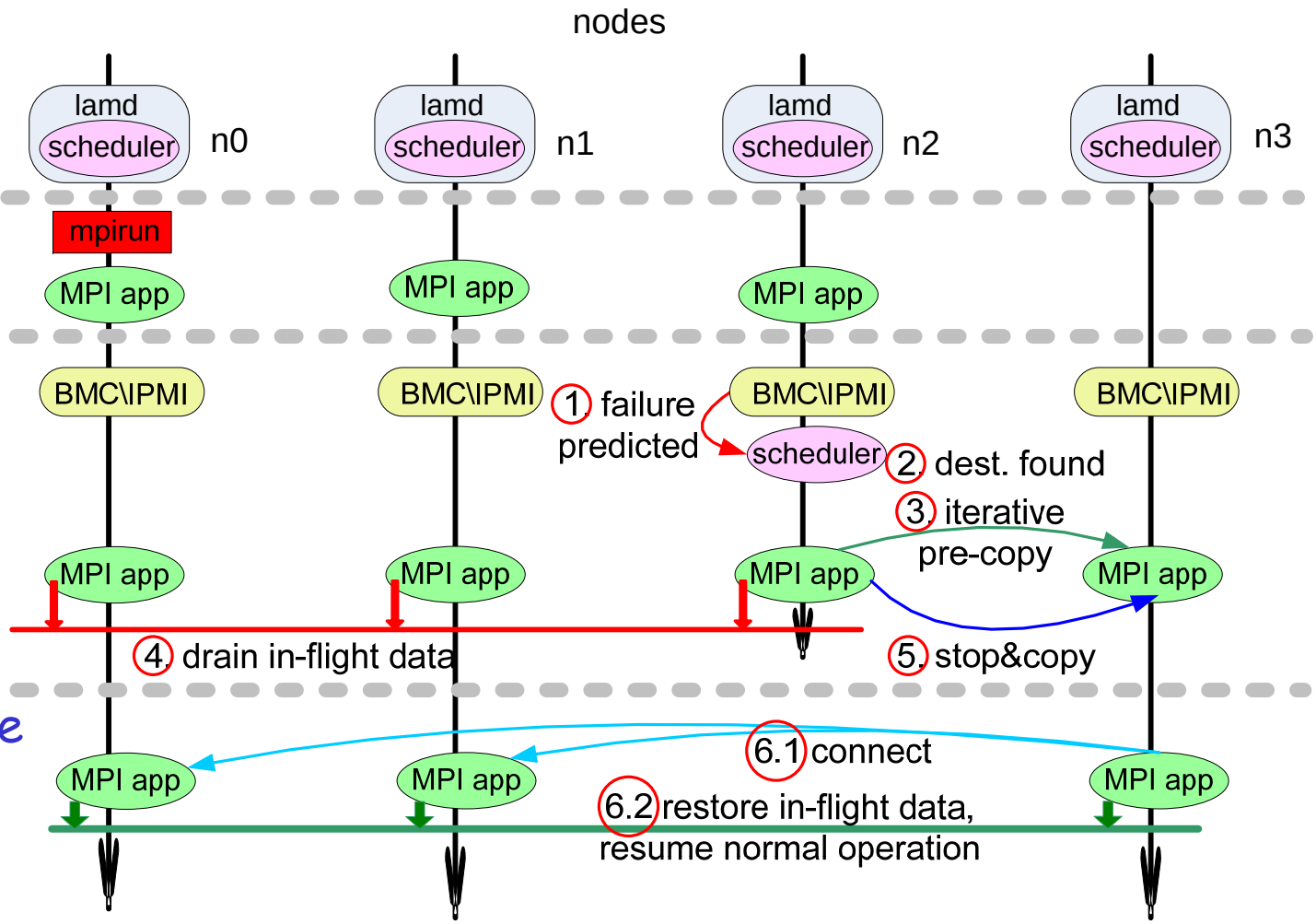
Our Design & Implementation – LAM/MPI

- Per-node health monitoring mechanism
  - Baseboard management controller (BMC)
  - Intelligent platform management interface (IPMI)
- NEW: Decentralized scheduler
  - Integrated into lamd
  - Notified by BMC/IPMI
  - Migration destination determination
  - Trigger migration

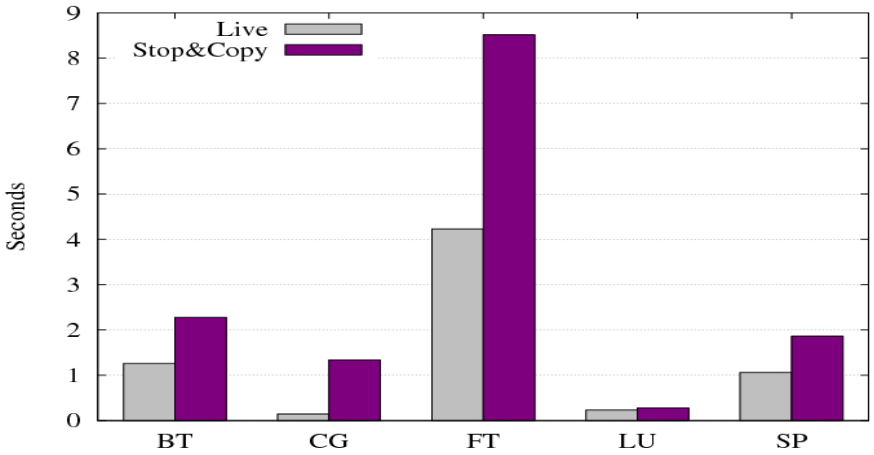


Live Migration Mechanism – LAM/MPI & BLCR

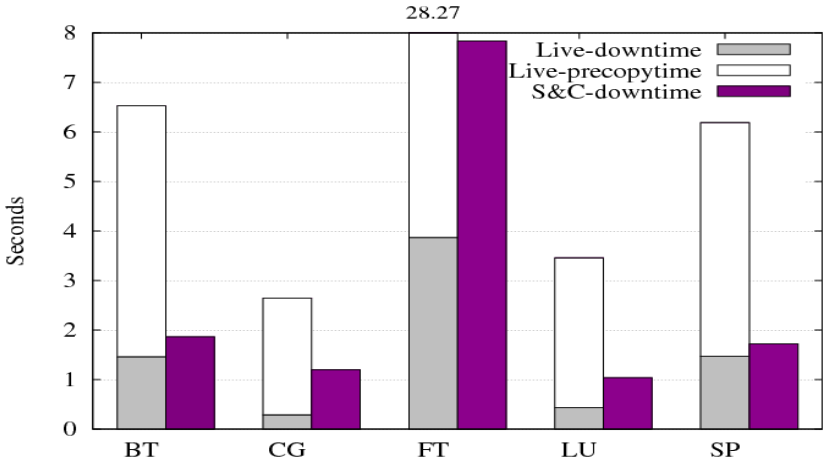
- MPI RTE setup
- MPI Job running
- Live migration
- Job exec. resume
- Step 3 can be omitted: live migration vs. stop&copy



Migration Overhead and Duration



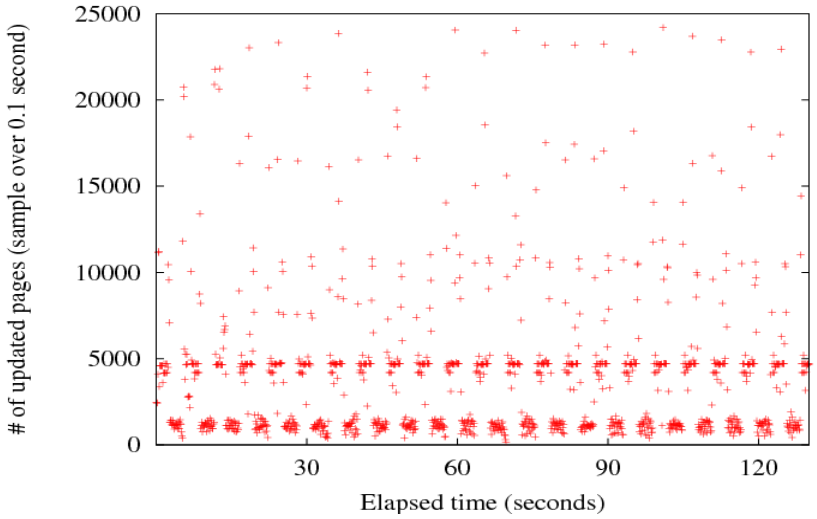
Migration Overhead



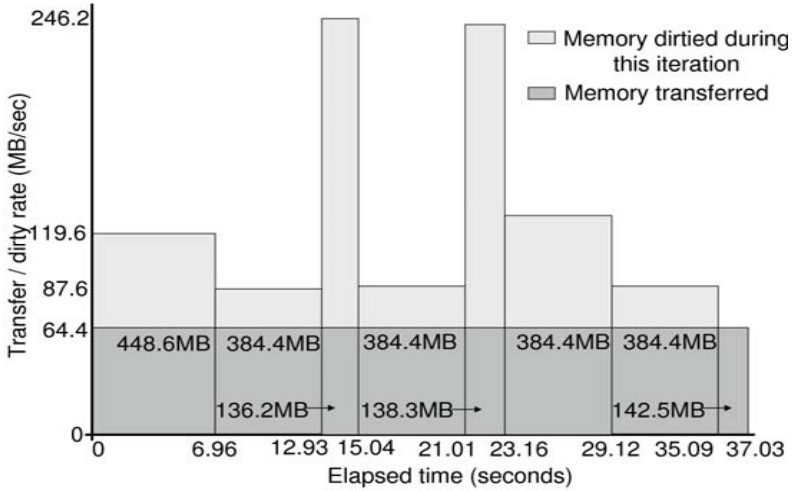
Migration Duration

- NPB Class C on 16 Nodes
- Overhead: difference of job run time w/ and w/o migration
- Live: 0.08-2.98% overhead Stop&copy: 0.09-6%
- Penalty of shorter downtime of live migration: prolonged precopy
  - No significant impact to job run time, long prior warning window required

Page Access Pattern & Iterative Migration



Page access pattern of FT



Iterative live migration of FT

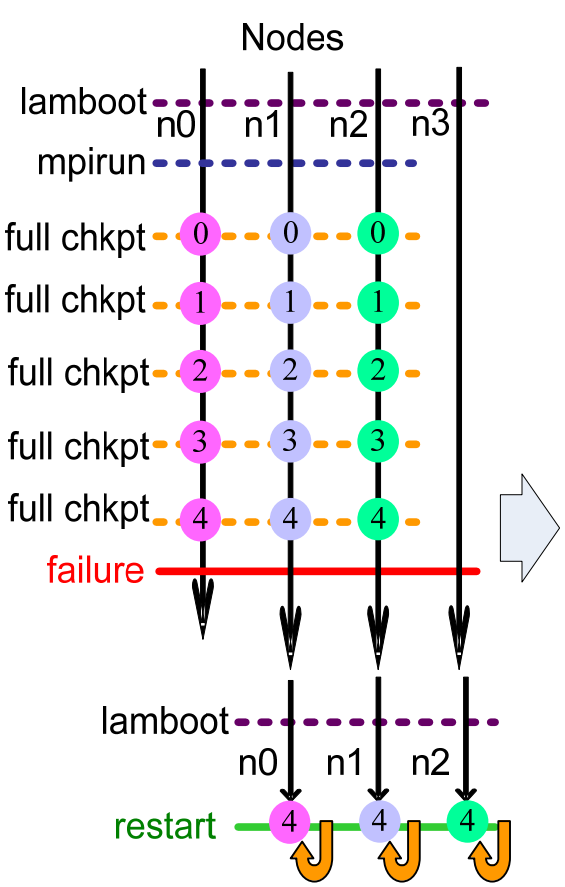
- Page write patterns are in accord with aggregate amount of transferred memory
- FT: 138/384MB -> 1200/4600 pages/.1

Outline

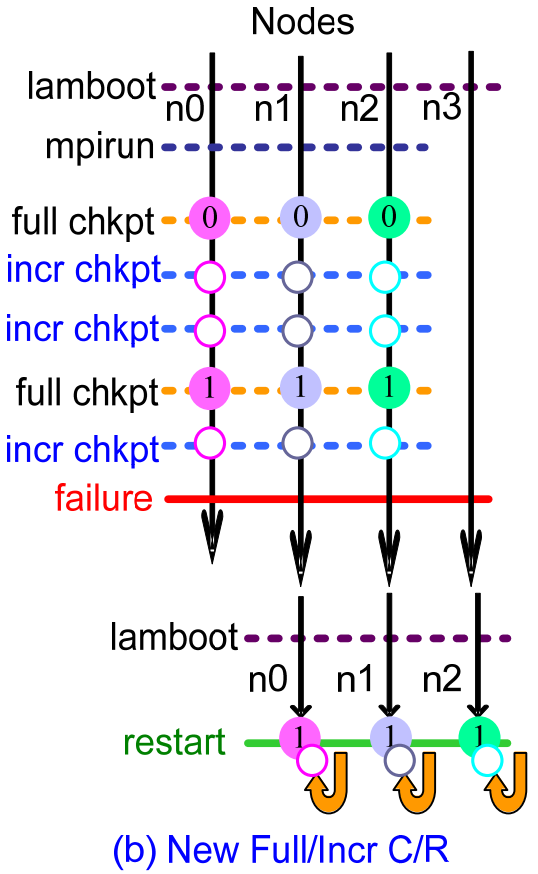
- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Phase 3: Current Work – Hybrid Full/Incr C/R

- Incremental checkpoint
  - Dirty pages saved only
- TICK [SC05] etc.:
  for single process,
  not for MPI tasks
- Hybrid full/incr. Chkpt
- Fast restart
- Hence:
  - Reduced I/O
    bandwidth requirement
  - Less storage space
  - Lower rate of full
    checkpoint



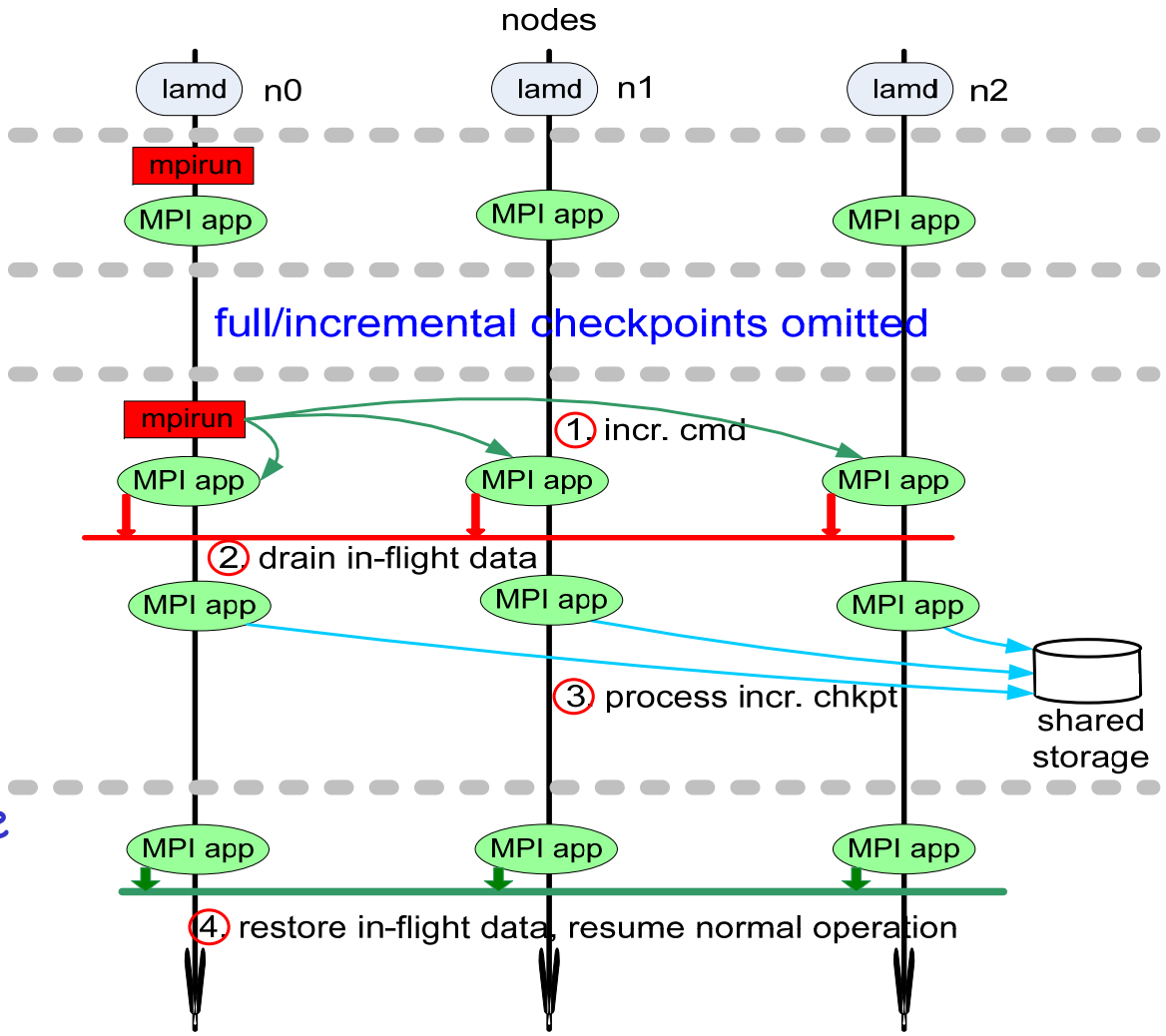
(a) Old Full C/R



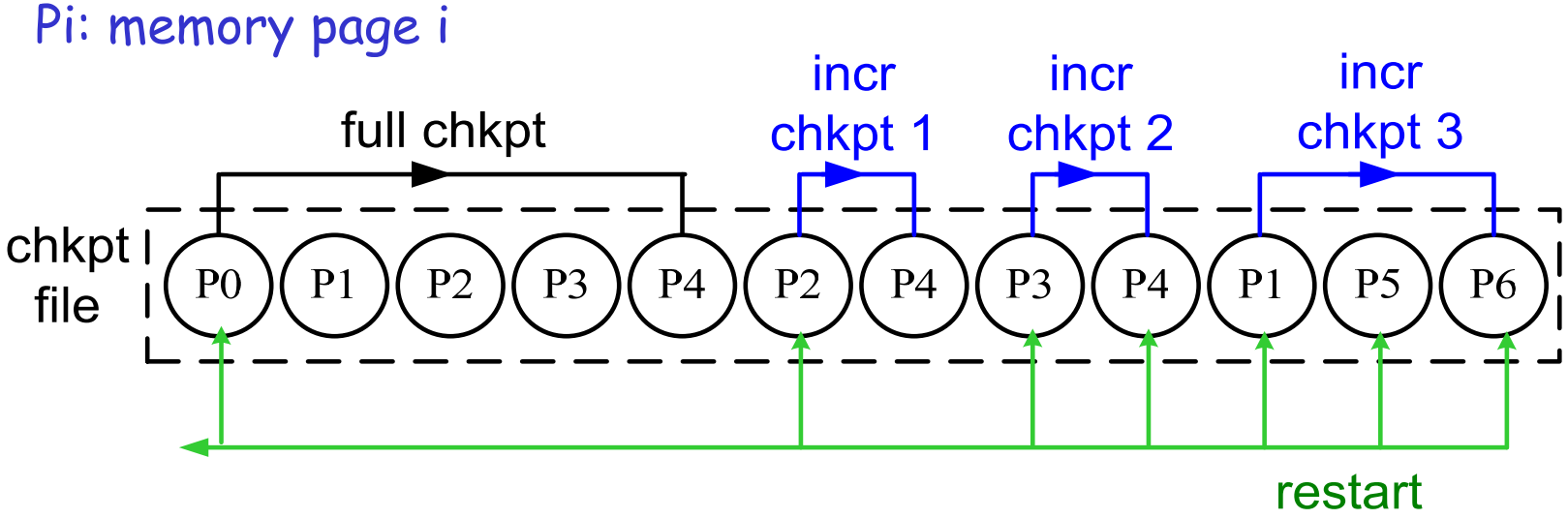
(b) New Full/Incr C/R

Incremental Checkpoint – LAM/MPI

- MPI RTE setup
- MPI Job running
- Incr. Chkpt
- Job exec. resume



Fast Restart



- Recovery scans all checkpoints in reverse sequence
  1. Allows the recovery of the last stored version of a page
  2. Any page only needs to be written once

- Overhead is equivalent to that of restoring from a single, full checkpoint

Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Related Work

- Transparent C/R
  - LAM/MPI w/ BLCR *[S.Sankaran et.al LACSI '03]*
    - Process Migration: scan & update checkpoint files [Cao et. Al, ICPADS, 05]
    → still requires restart of entire job
  - Log based (Log msg + temporal ordering): MPICH-V *[SC 2002]*
- Non-transparent C/R: Explicit invocation of checkpoint routines
  - LA-MPI *[IPDPS 2004]* / FT-MPI *[EuroPVM-MPI 2000]*
- Failure prediction: Predictive management *[Gu et. Al, ICPP08] [Gu et. Al, ICDCS08] [Sahoo et. Al, KDD03]*
- Fault model: Evaluation of FT policies *[Tikotekar et. Al, Cluster07]*
- Process migration: MPI-Mitten *[CCGrid06]*
- Incremental checkpoint: for single process, not for MPI tasks
  - TICK *[SC05] etc.*

Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Conclusion

- Design generic for any MPI implementation / process C/R
- Implemented over LAM-MPI w/ BLCR

Job-Pause

- Completely transparent
- Low overhead: 69.6% < job restart + lam reboot
  - No job requeue overhead/ Less staging cost/ No LAM Reboot

Live migration

- Cut the number of chkpts in half when 70% faults handled proactively
- Low overhead: Live: 0.08-2.98% Stop&copy: 0.09-6%
  - No job requeue overhead/ Less staging cost/ No LAM Reboot

Incr. chkpt

- Lower rate of full checkpoint
- Restart equal to that from one single full chkpt

Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Future Work

- Incremental checkpoint: (target a conference paper)
  - Finish debugging + evaluate w/ NPB
  - Create/assess applications with varying memory pressure to measure the tradeoff between full & incremental chkpts
- Live migration: (journal extensions)
  - Heuristic algorithm for tradeoff between live & frozen migrations
  - Back migration upon node recovery
- Hybrid full/incr. chkpt: (journal extensions)
  - Optimal/heuristic algorithm for full/incr. chkpt placement
  - Replace full chkpt with live chkpt
- A reliable FT framework
  - Combine the job pause, live migration and incremental checkpoint
- System pre-deployment
  - Pre-deploy process images on spare nodes

Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Phase 1: a Job-Pause Service
- Phase 2: Proactive Live Migration
- Phase 3: Hybrid Full/Incremental Checkpoint/Restart
- Related work
- Conclusion
- Future Work
- Publications

Publications

- "MOLAR: Adaptive Runtime Support for High-end Computing Operating and Runtime Systems" by C. Engelmannn, S. Scott, D. Bernholdt, N. Gottumukkala, C. Leangsuksun, J. Varma, C. Wang, F. Mueller, A. Shet and P. Sadayappan, ACM SIGOPS OSR 2006
- "Scalable, Fault-Tolerant Membership for MPI Tasks on HPC Systems" by J. Varma, C. Wang, F. Mueller, C. Engelmannn, and S. Scott, ICS 2006
- "A Job Pause Service under LAM/MPI+BLCR for Transparent Fault Tolerance" by C. Wang, F. Mueller, C. Engelmannn, and S. Scott, IPDPS 2007
- "Proactive Process-Level Live Migration in HPC Environments" by C. Wang, F. Mueller, C. Engelmannn, and S. Scott, Supercomputing 2008
- "Process-Level Fault Tolerance for Job Healing in HPC Environments" Supercomputing 2008 Doctoral Research Showcase: 2007, C. Wang, ACM/IEEE HPC Fellows

Recovery/Replication of Job Input Data in Supercomputers:

- "Optimizing Center Performance through Coordinated Data Staging, Scheduling and Recovery" by Z. Zhang, C. Wang, S. Vazhkudai, X. Ma, G. Pike, J. Cobb and F. Mueller, Supercomputing 2007
- "On-the-fly Recovery of Job Input Data in Supercomputers" by C. Wang, Z. Zhang, S. Vazhkudai, X. Ma and F. Mueller, ICPP 2008
- "Improving the Availability of Supercomputer Job Input Data Using Temporal Replication" by C. Wang, Z. Zhang , X. Ma, S. Vazhkudai and F. Mueller, ready to submit…

Process-Level Fault Tolerance for Job Healing in HPC Environments

Chao Wang, Frank Mueller

Department of Computer Science

North Carolina State University

Stephen L. Scott, Christian Engelmann

Systems Research Team (SRT)

Computer Science Group

Oak Ridge National Laboratory

HPC Resiliency Summit: Workshop on Resiliency for Petascale HPC
Los Alamos Computer Science Symposium
October 15, 2008, Santa Fe, New Mexico