Proactive Fault Tolerance for HPC using Xen Virtualization

Arun Babu Nagarajan, Frank Mueller

NC STATE UNIVERSITY

Christian Engelmann, Stephen L. Scott

Oak Ridge National Laboratory

Problem Statement

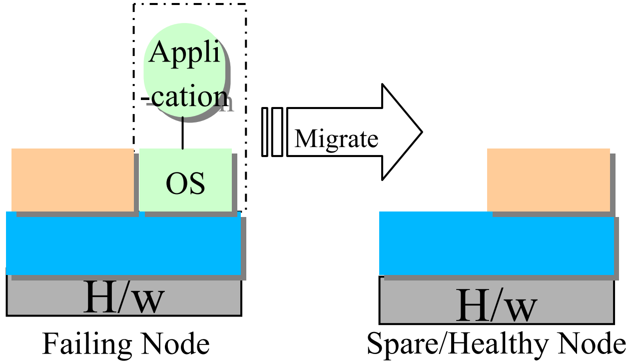
- Trend in HPC: high end systems with thousands of processors
  - Increased probability of a node failure: MTBF becomes shorter
  - CPU/memory/IO failures

System	# CPUs	MTBF/I, see [20]
ASCI Q	8192	6.5hrs
ASCI WHITE	8192	5/40 hrs
PSC Lemieux	3016	9.7hrs
Google	15000	20 reboots/day

- MPI widely used for scientific apps
  - Problem with MPI: no recovery from faults in the standard
- Currently FT exist but…
  - mostly reactive: process checkpoint/restart [3 DOE labs use this approach]
  - must restart entire job → inefficient if only one/few node(s) fail
  - overhead: re-execute some of prior work
  - issues: checkpoint at what frequency?
  - 100 hr job requires add'l 150 hrs for checkpointing on a petaflop machine (w/o failure) [*Philp, 2005*]

Our Solution

- Proactive FT
  - — anticipates node failure
  - — takes *preventive action* (instead of 'reacting' to a failure)
    - -migrate entire OS (to a healthy node)
    - -transparent to app (and to OS)



- ➢ avoids high overhead compared to reactive scheme
  - -overhead of our scheme: much smaller

➢ Complements reactive FT → less frequent checkpoints!

Design space

- 1. Mechanism to predict/anticipate node failures
  - OpenIPMI
  - lm_sensors (specific to x86 Linux)
- 2. Mechanism to identify best target node
  - Centralized approaches → don't scale / unreliable
  - Scalable distributed approach → based on Ganglia
- 3. Mechanism for preventive action: relocation of running app
  - Preserve apps state
  - Small overhead on app
  - Xen Virtualization w/ live migration [*Clark et al., NSDI'05*]
    - Open source

Mechanisms (1): Health Monitoring

Health Monitoring w/ OpenIPMI:

- Baseboard Mgmt Controller (BMC)
  - w/ sensors to monitor temperature, fan speed, voltage, etc.
- IPMI (Intelligent Platform Management Interface)
  - increasingly common in HPC
  - std. message-based interface to monitor H/W
  - raw messaging harder to use and debug
- OpenIPMI: open source, higher level abstraction from raw IPMI message-response system to communicate w/ BMC
  - read sensors portably/simple API

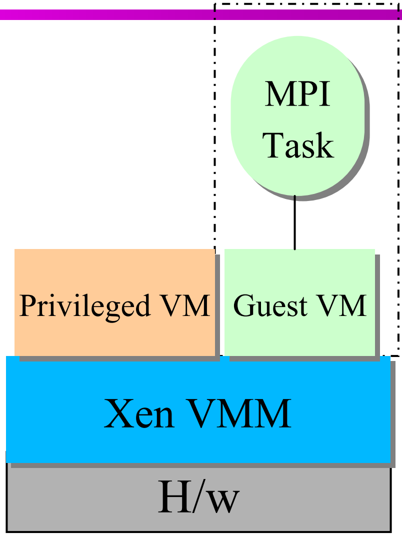
➢ OpenIPMI used to gather health information of nodes

Mechanisms (2): Distributed Monitoring

Distributed Monitoring with Ganglia:

- widely used, scalable distributed load monitoring tool
- All nodes in cluster run ganglia daemon
  - each node has a approximate view of entire cluster
- UDP to transfer messages
- Measures
  - CPU / memory / network utilization (by default)
  - identify least loaded node → migration target
- Ganglia protocol also extended to distribute IPMI sensor data

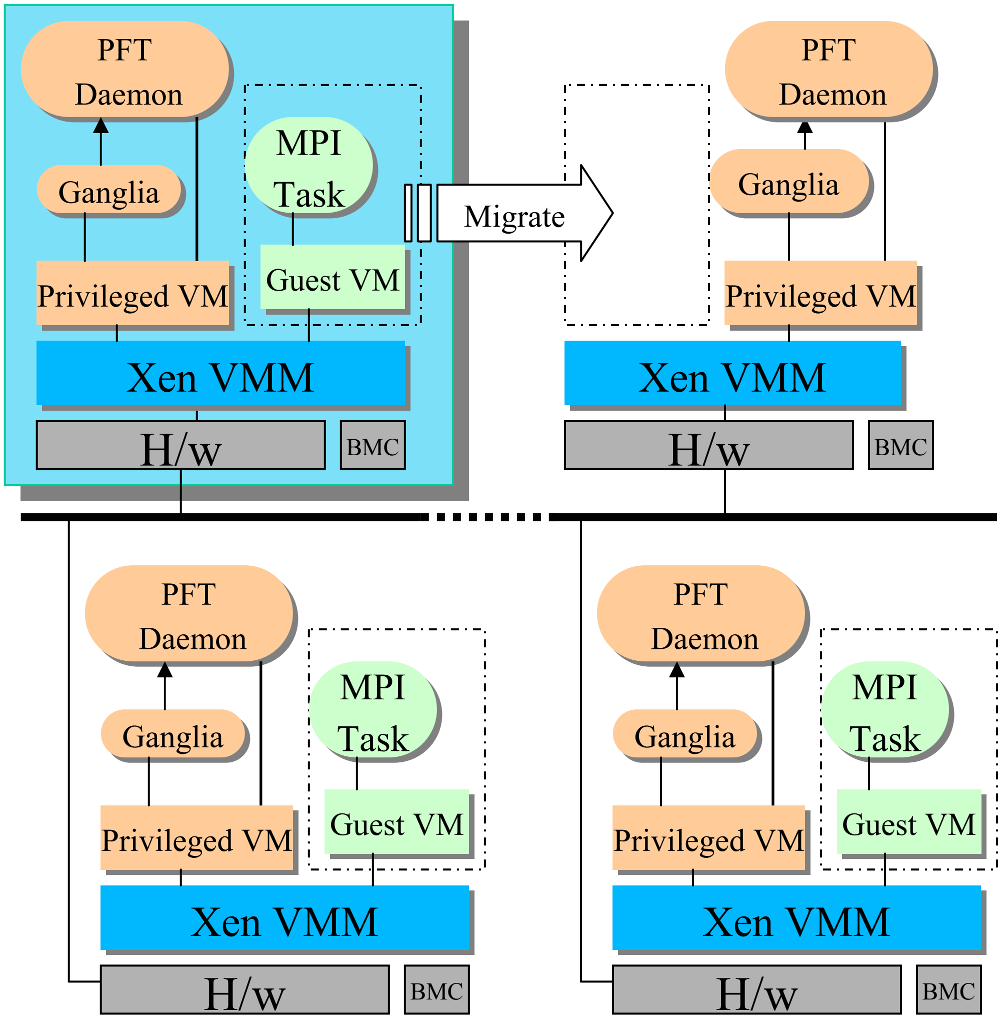
Mechanisms (3): Virtualization



Fault Tolerance w/ Xen:

- para-virtualized environment
  - OS modified
  - app unchanged
- Privileged VM & guest VM run on Xen hypervisor/VMM
- Guest VMs can live migrate to other hosts → little overhead
  - State of VM preserved
  - VM halted for insignificant period of time
  - Migration phases:
    - phase 1: send guest image → dst node, app running
    - phase 2: repeated diffs → dst node, app still running
    - phase 3: commit final diffs → dst node, OS/app frozen
    - phase 4: activate guest on dst, app running again

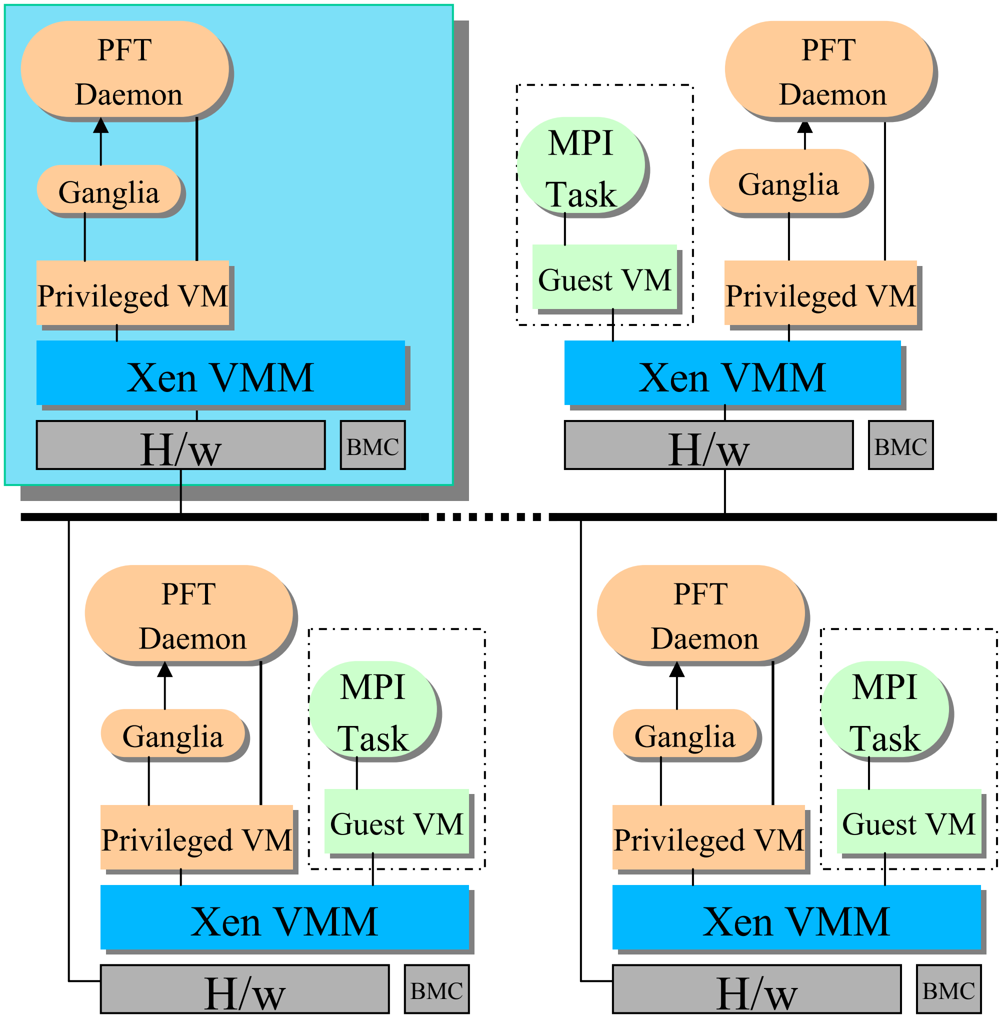
Overall set-up



BMC Baseboard Management Contoller

- Stand-by Xen host, no guest (spare node)
- Deteriorating health → migrate guest (w/ MPI app) to spare node

Overall set-up

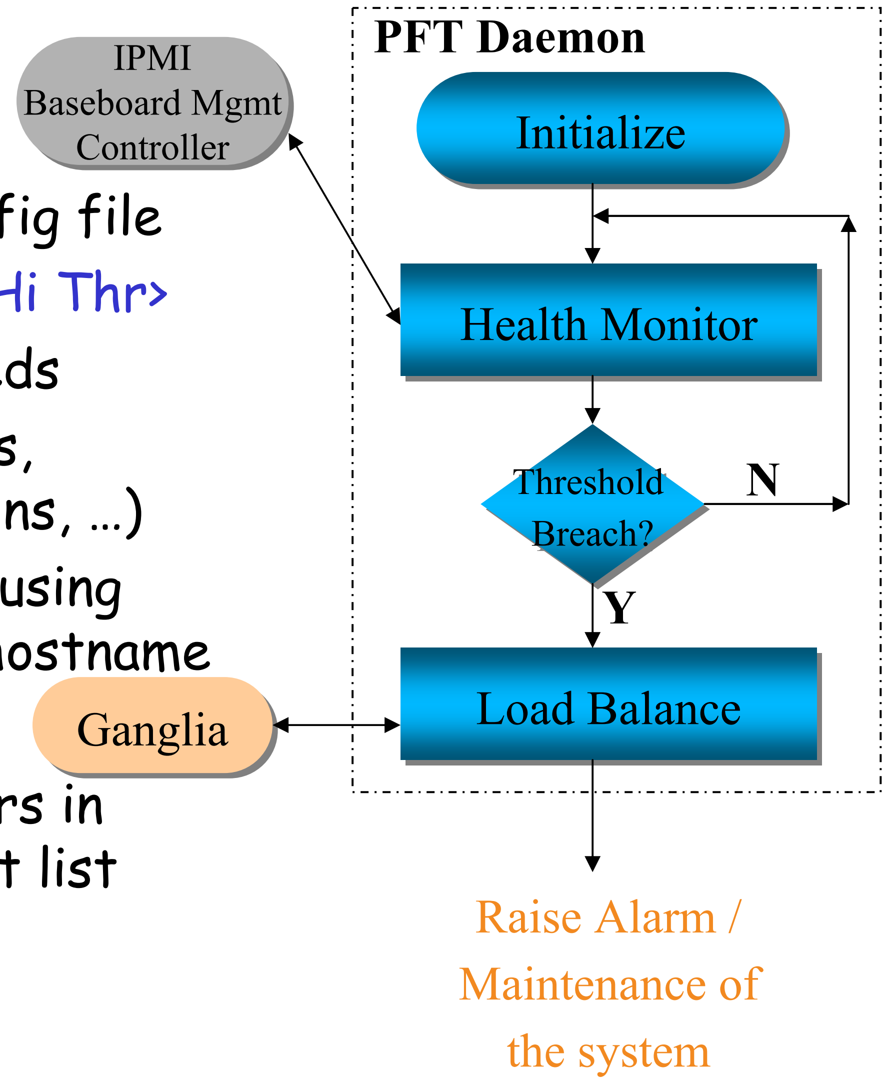


BMC Baseboard Management Contoller

- Stand-by Xen host, no guest (spare node)
- Deteriorating health → migrate guest (w/ MPI app) to spare node
- Destination host generates unsolicited ARP reply
  - indicates Guest VM IP has moved to new location
  - ARP tells peers to resend packets to new host

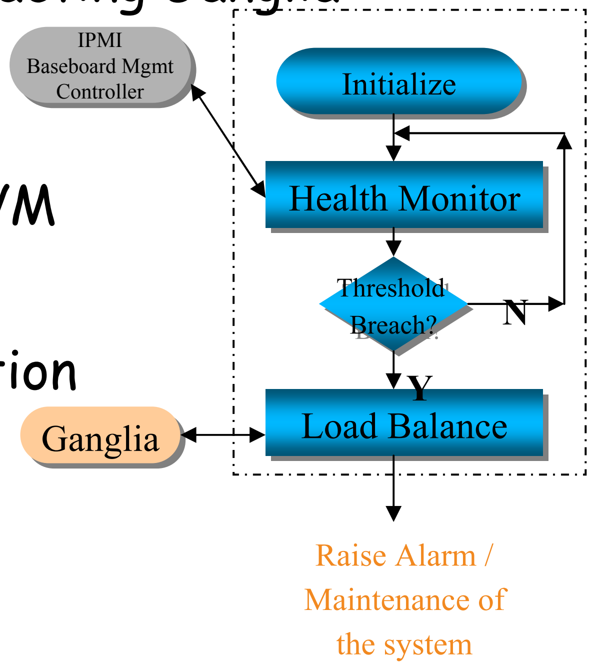
PFTd: Proactive Fault-Tolerance Daemon

- Runs on privileged VM (host)
- Initialize
  - Read safe threshold from config file
    - <Sensor name> <Low Thr> <Hi Thr>
    - CPU temperature, fan speeds
    - extensible (corrupt sectors, network, voltage fluctuations, …)
  - Init connection w/ IPMI BMC using authentication parameters & hostname
  - Obtains set of available sensors in system, validates it against out list



PFTd: Proactive Fault-Tolerance Daemon

- Health Monitoring
  - interacts w/ IPMI BMC (via OpenIPMI) to read sensors
  - Periodic sampling of data
  - threshold exceeded → control handed over to load balancing
- PFTd determines migration target by contacting Ganglia
  - — Load-based selection (lowest load)
  - — Load obtained by /proc file system
  - — Invokes Xen live migration for guest VM
- Xen user-land tools (at VM/host)
  - — command line interface for live migration
  - — PFTd initiates migration for guest VM



Experimental Framework

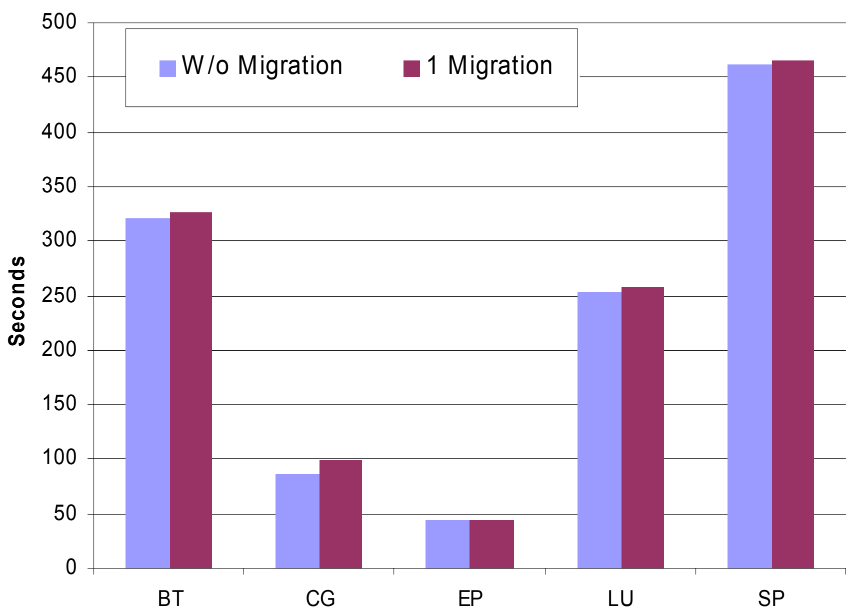
- Cluster of 16 nodes (dual core, dual Opteron 265, 1 Gbps Ether)
- Xen-3.0.2-3 VMM
- Privileged and guest VM run Linux kernel version 2.6.16
- Guest VM:
  - Same configuration as privileged VM
  - 1GB RAM
  - Booted on VMM w/ PXE netboot via NFS
  - Has access to NFS (same as privileged VM)
- Ganglia on Privileged VM (and also Guest VM) on all nodes

Experimental Framework

- NAS Parallel Benchmarks run on Guest VMs
- MPICH-2 w/ MPD ring on *n* GuestVMs (no job-pause required!)
- Experiment-aid process on privileged&guest domain:
  - monitors MPI task runs (on guest)
  - issues migration command (NFS used for synchronization)
- Measured:
  - wall clock time with and w/o migration
  - actual downtime + migration overhead (modified Xen migration)
    - with (a) live and (b) stop&copy migration
- benchmarks run 10 times, results report avg. (→ small std dev.)
- NPB V3.2.1: BT, CG, EP, LU and SP benchmarks
  - IS run is too short
  - FT, MG requires > 1GB for class C (guest VM RAM limit)

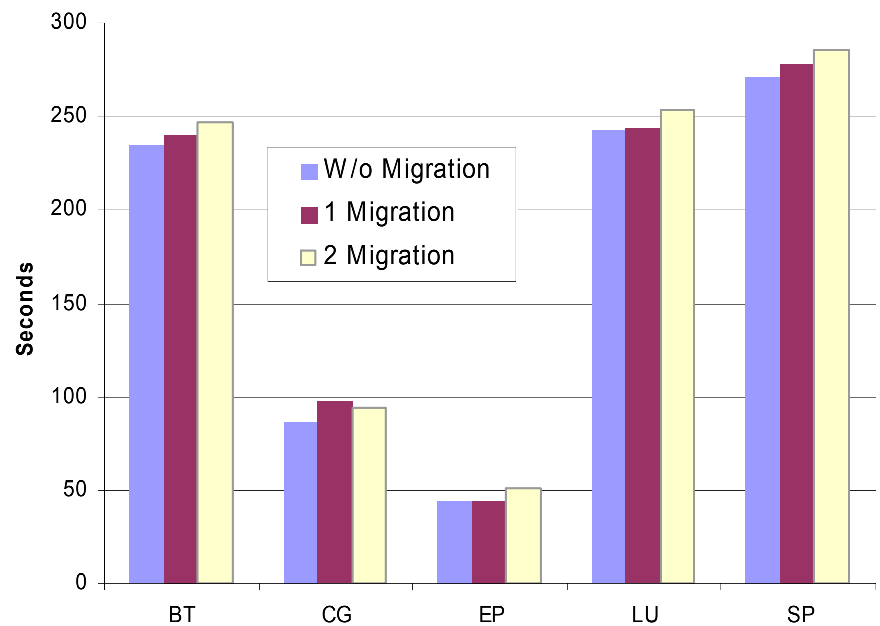
Results: Node Failures

1. Single node failure



NPB Class C / 16 nodes

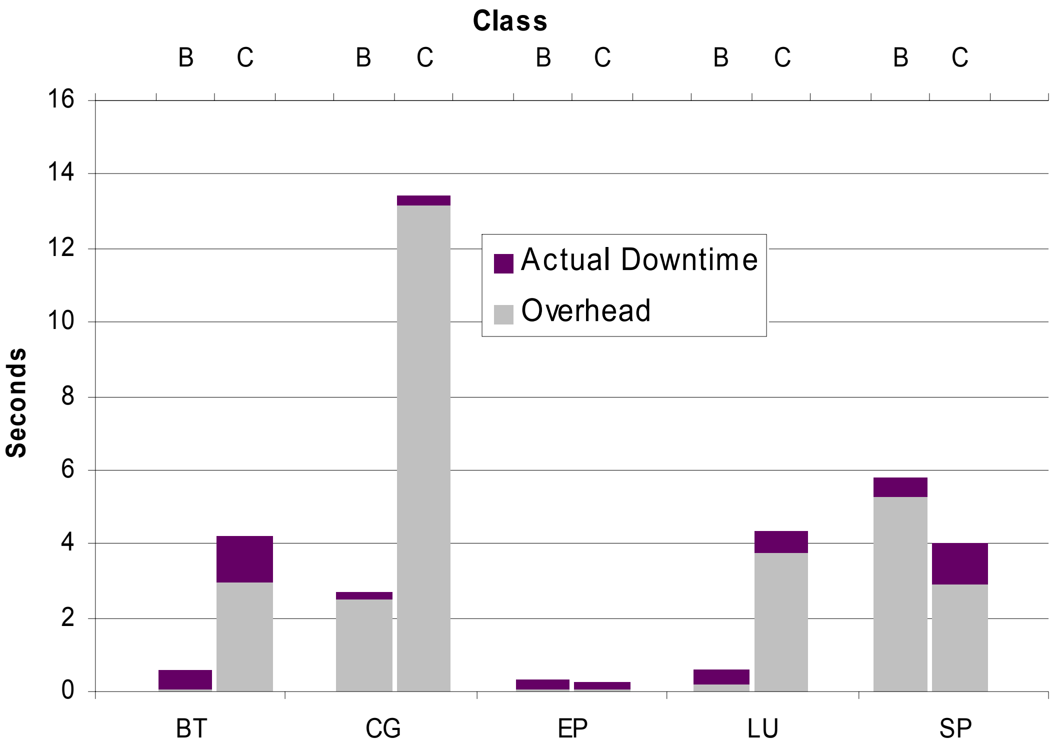
2. Double node failure



NPB Class B / 4 nodes

- Single node failure: 0.5-5% add'l cost over total wall clock time
- Double node failure: 2-8% add'l cost over total wall clock time

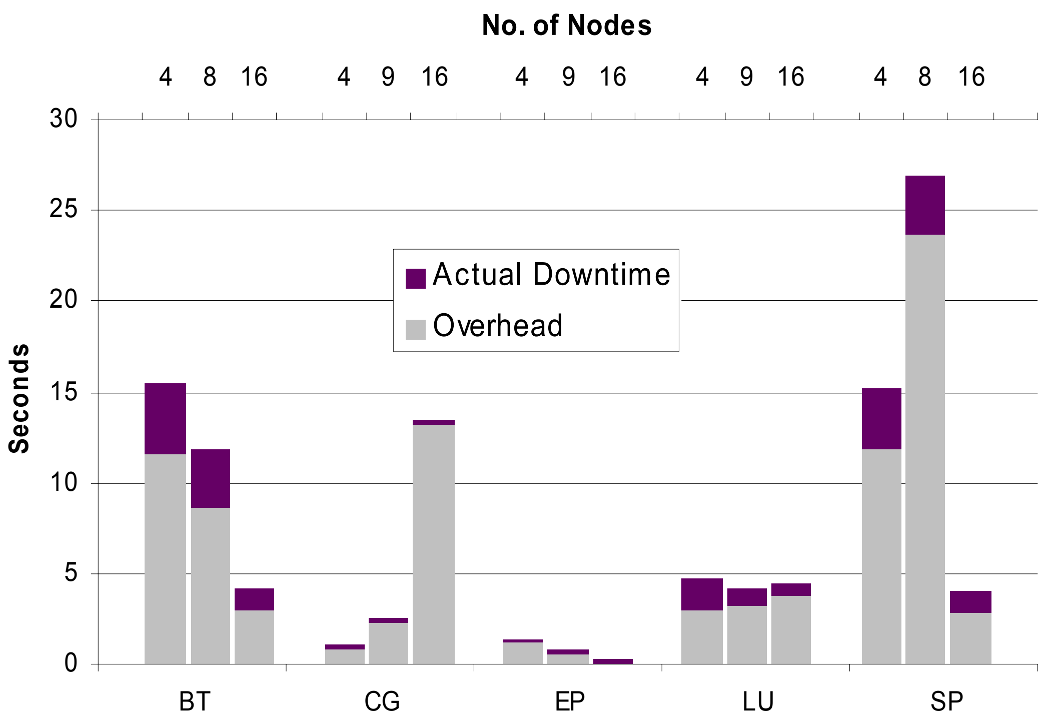
Results: Problem Scaling



NPB 16 nodes

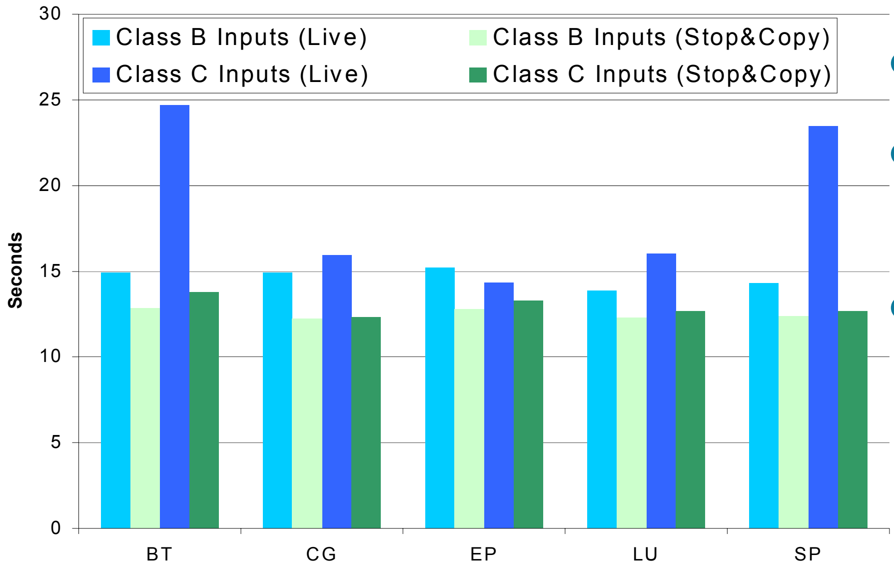
- Only overhead depicted
- Downtime: VM halted
- Overhead: migration delay (diff operation, etc.)
- Increasing problem size (B → C): overhead increases (expected)
- SP outlier: migration may have coincided w/ global sync. point → network contention (fixable)

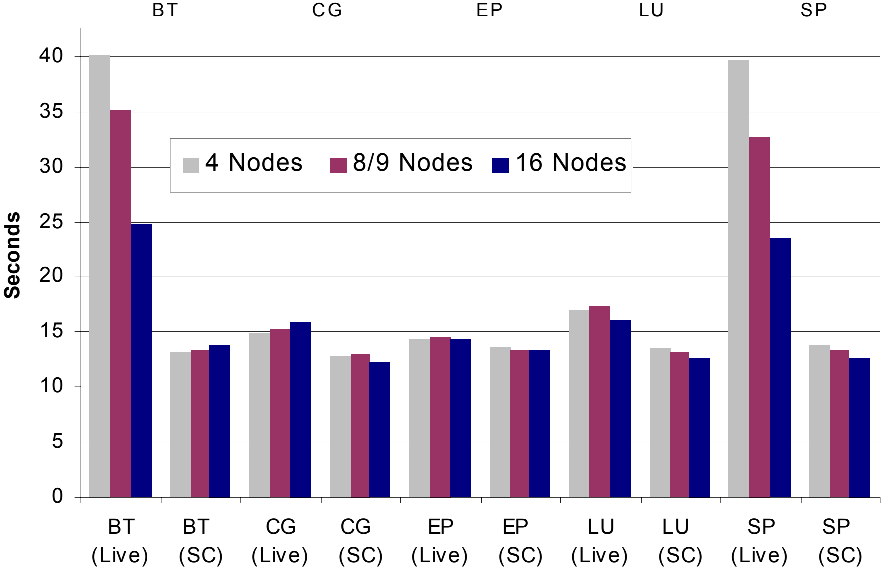
Results: Task Scaling



- expect decreased overhead for increasing # of nodes
  - see BT, EP, LU, SP
- CG: add'l msg overhead & smaller data sets/node
  - atypical
- Overall, indicates potential of our approach

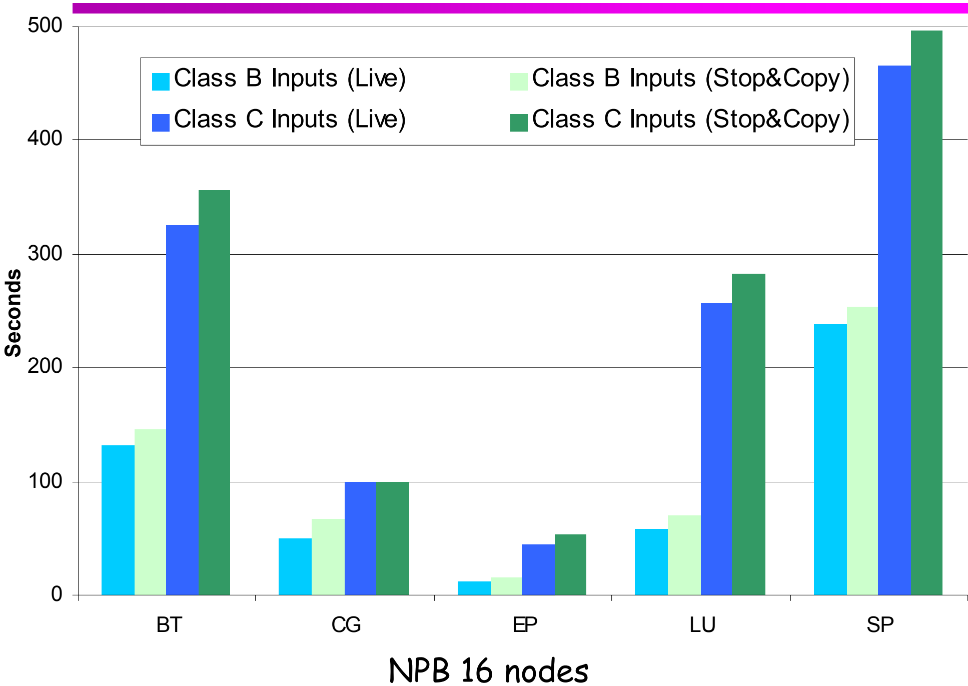
Results: Total Migration Duration





- Live vs. Stop&Copy
- min. 13secs: Xfer 1GB VM (w/o any active processes)
- Vary problem size: class B & C
  - Live: 14-24 secs (class B & C)
  - Stop&Copy: 13-14 secs
- Vary # nodes: 4, 8/9, 16
  - Live: Duration decreases / remains const. for > # nodes: 40–14 secs
  - Stop&copy: 13-14 secs

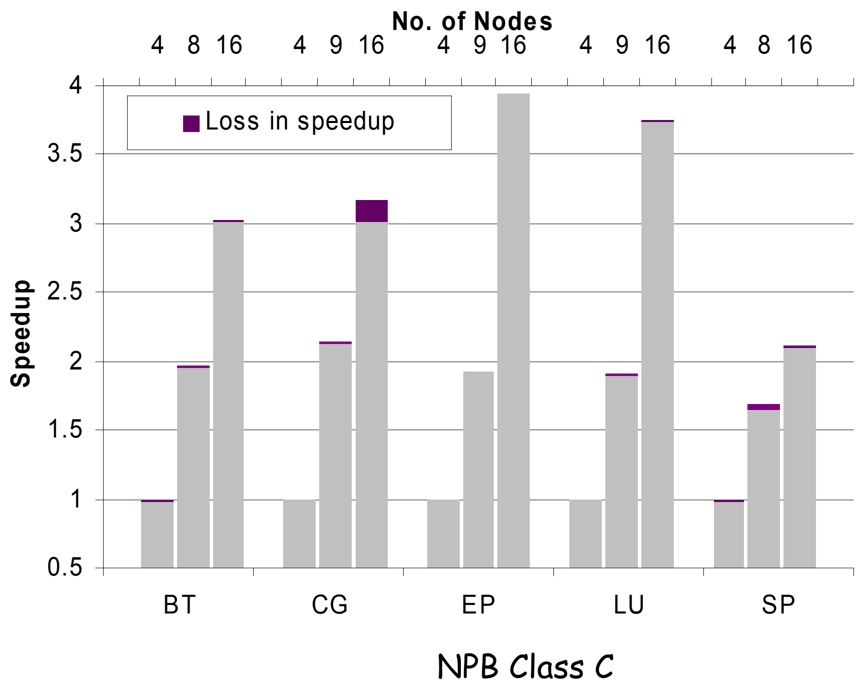
Results: Overall Execution Time



- Live migrations: takes longer but application is not stopped!
- Stop&copy: faster but app. stopped
- compare runtime for both modes
- Overall: T(Live) < T(Stop&copy)

- Migration duration important metric: should be minimized
  - How much advance warning? health degrades → actual failure
    - little to no prior work in this area
  - Our solution could benefit from learning techniques
    - identifying false warnings, feedback-based learning

Results: Task Scaling vs. Total Exec. Time



- Speedup of benchmarks not affected (up to 16 nodes)
- Wanted: large-scale cluster to run customized Xen

Related Work

- FT – Reactive approaches more common
  - Automatic
    - Checkpoint/restart (e.g., BLCR) [*Sankaran et al., LACSI '03*], [*G.Stellner, IPPS ' 96*]
    - Log based (Log msg + temporal ordering) [*G. Bosilica, SC'02*]
  - Non-automatic
    - Explicit invocation of checkpoint routines [*Aulwes et al., IPDPS'04*], [*Fagg/Dongarra,Ero PVM/MPI'00*]
- Virtualization in HPC: little/no overhead [Huang et al., ICS '06]
- VMM-bypass for I/O → MPI w/ virtualization competitive [*Liu et al., USENIX'06*]
- Optimize network virtualization [*Menon et al., USENIX'06*]
- Job pause under LAM/MPI+BLCR [C.Wang, IPDPS '06, our Group]

Conclusion

- Novel, proactive FT scheme w/ virtualization
  - Provides transparent & automatic FT for arbitrary MPI apps
  - Less overhead than reactive
  - still, complements reactive → lower checkpoint frequency

- Need studies on potential to detect health deterioration
- Currently pursuing further opportunities to reduce overhead…

Backup Slides

- How much time before failure?
  - The upper threshold is the memory limit (1GB for a vm). So a 1 minute warning suffices, which is possible in case of disk and fan failures..