Hybrid Checkpointing for MPI Jobs in HPC Environments

Chao Wang, Frank Mueller
North Carolina State University

Christian Engelmann, Stephen L. Scott
Oak Ridge National Laboratory

ICPADS’10 Dec. 9/10 Shanghai, China
Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Our Design and Implementation
- Experimental Framework
- Performance Evaluation
- Related Work
- Conclusion
Problem Statement

- **Trends in HPC:** MTBF/I becomes shorter, Failure a norm!
  - High end systems with > 100,000 processing cores
  - MTBF/I: 6.5-40 hours
  - Peta-scale systems: MTBF 1.25 hours

<table>
<thead>
<tr>
<th>System</th>
<th># Cores</th>
<th>MTBF/I</th>
<th>Outage source</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCI Q</td>
<td>8,192</td>
<td>6.5 hrs</td>
<td>Storage, CPU</td>
</tr>
<tr>
<td>ASCI White</td>
<td>8,192</td>
<td>40 hrs</td>
<td>Storage, CPU</td>
</tr>
<tr>
<td>PSC Lemieux</td>
<td>3,016</td>
<td>6.5 hrs</td>
<td></td>
</tr>
<tr>
<td>Google</td>
<td>15,000</td>
<td>20 reboots/days</td>
<td>Storage, memory</td>
</tr>
<tr>
<td>Jaguar</td>
<td>23,416</td>
<td>37.5 hrs</td>
<td>Storage, memory</td>
</tr>
</tbody>
</table>

- **MPI** widely accepted in scientific computing, Frequently deployed C/R helps but...
  - 60% overhead on C/R: 100 hrs job -> 251 hrs
  - C/R efficiency: 55-85%
  - Coordinated C/R: all job tasks checkpointed
    - Inefficient if only a subset of process image changes b/w checkpoints
    - Extremely high I/O bandwidth demand
Our Solution – Hybrid Checkpointing

- Hybrid full/incremental checkpoint over LAM/MPI+BLCR
- Incremental checkpoint
  Dirty pages saved only
- Fast restart
- Hence:
  - Reduced I/O bandwidth requirement
  - Less storage space
  - Lower rate of full checkpoint
  - Less overhead of C/R

(a) Old Full C/R
(b) New Full/Incr C/R
Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Our Design and Implementation
- Experimental Framework
- Performance Evaluation
- Related Work
- Conclusion
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

**Diagram:**
- User Application
- MPI Layer
- LAM Layer
- Operating System

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

**Example:** A two-way MPI job on two nodes
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
- Our Design and Implementation
- Experimental Framework
- Performance Evaluation
- Related Work
- Conclusion
Scheduler & Incremental Chkpt @ LAM/MPI

- A decentralized scheduler: issues Full/Incr. chkpt commands

- **MPI RTE setup**

- **MPI Job running**

- **Incr. Chkpt**

- **Job exec. resume**
Incremental Checkpoint @ BLCR

Call-back kernel thread: coordinates user command process and app. process

(In kernel: dashed lines/boxes)

1. app registers threaded callback
   → spawns callback thread
2. thread blocks in kernel
3. incr_chkpt utility calls ioctl(), unblocks callback thread
4. All threads complete callbacks & enter kernel
5. Only save dirty pages
6. Run regular application code from restored state
Checkpoint Files & 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 \(\approx\) that of restoring from a single, full checkpoint
Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Our Design and Implementation
- Experimental Framework
- Performance Evaluation
- Related Work
- Conclusion
Experimental Framework

- **Experiments conducted on**
  - Opt cluster: 18 nodes, 2 cores, dual Opteron 265, 1 Gbps Ether
  - Fedora Core 5 Linux x86_64 w/ our dirty bit patch
  - Lam/MPI + BLCR w/ our hybrid full/incremental C/R extensions

- **Benchmarks**
  - NPB V3.3 (MPI version)
  - mpiBLAST (parallel implementation of NCBI BLAST)
Full Chkpt Overhead vs. Execution Time

- One full chkpt overhead vs. base execution time < 1% (MG except)
- MG: large checkpoint files, but short overall exec time

NPB-D&mpiBLAST Full Checkpoint Overhead
Incr. chkpt overhead less significant, thus:

hybrid Full/Incr. chkpt reduces chkpt overhead compared to full chkpt throughout
**Checkpoint File Size (=> Chkpt Overhead)**

- Full/Incr. chkpt overhead proportional to chkpt file size
- Full chkpt overhead nearly same at any time of job exec.
- Incr. chkpt overhead nearly same at any interval
- Incr. chkpt overhead lower than full chkpt overhead, except EP
Restart Overhead

NPB-C-4/8/9/16 restart overhead

NPB-D-16 restart overhead

mpiBLAST restart overhead

- Restart time: Full+3Incr. is 68% (1.17 secs) larger restart from Full, but chkpt file size of Full+3Incr. 185% larger than that of Full
- Chkpt time of 3Incr is 16.64 secs shorter that for 3Full
Benefit of Hybrid C/R Mechanism

- **Overall savings:**
  \[ S_n = n \times (O_f - O_i) - (R_{f+n_i} - R_f) \]
  
  - **Sn:** saving w/ n incr. chkpts b/w two full chkpts
  - **Of:** full chkpt overhead
  - **Oi:** incr. chkpt overhead
  - **Rf+ni:** restart overhead from full+n incr. chkpts
  - **Rf:** restart overhead from one full chkpt

- incr. chkpt overhead ↓ -> chkpt frequency ↑ -> job work lost ↓
- Restart cost (Rf+ni - Rf) is low, compared to (Of - Oi)
- All benchmarks benefit from hybrid Full/Incr. C/R mechanism
- Naksinehaboon et al. provide a model/formula for optimal n
  - n = 9 with our results → more savings
Outline

- Problem vs. Our Solution
- Overview of LAM/MPI and BLCR
- Our Design and Implementation
- Experimental Framework
- Performance Evaluation
- Related Work
- Conclusion
Related Work

- **Checkpoint/Restart**
  - Coordinated: LAM/MPI w/ BLCR [S. Sankaran et al. LACSI ’03]
  - Uncoordinated: MPICH-V [SC 2002]: Log based
  - Both checkpoint entire process image → high overhead

- **Incremental checkpoint:**
  - for single process, not for MPI tasks:
    - TICK [SC05]
    - Pickpt [ACM Symposium on Applied computing 05], etc.
  - Language specific solutions:
    - Charm++ [Chakravorty et. Al, HiPC06], etc.

- **Checkpoint Interval Model:**
  - Young [26]: model for fixed chkpt interval; Daly [27]: improve it
  - Liu et al. [IPDPS08]: model for optimal full C/R stategy
  - Naksinehaboon et al. [CCGrid08]: model/formula used here
Conclusion

- Novel hybrid C/R mechanism over LAM-MPI + BLCR
  - Decentralized scheduler
  - Lower rates for full chkpt
  - Dirty bit mechanism to track and save modified pages
  - Reduced I/O bandwidth & storage requirement
  - Fast restart from Full+nIncr. Checkpoints
    - any page only written once
- Better performance of hybrid C/R mechanism over original full C/R
  - Savings by 3Full → 3Incr.: 15.47 seconds
    ( = 16.64 savings on chkpt - 1.17 cost on restart)
  - 1:9 b/w Full&Incr. checkpoints → optimal balance
- On-going work: OpenMPI extensions + BLCR release for incr. Chkpts
Questions?

Thank you!

This work was supported in part by:

- **NSF Grants**: CCR-0237570, CNS-0410203, CCF-0429653
- **DOE GRANT**: DE-FG02-08ER25837
- **Office of Advanced Scientific Computing Research**
- **DOE Contract**: DE-AC05-00OR22725

Project websites:

**NCSU**: http://moss.csc.ncsu.edu/~mueller/