

Highly Available Shared Virtual Memory Clusters A Progress Report

*High Availability and Performance Computing Workshop
The 5th Symposium of the Los Alamos Computer Science Institute
Santa Fe, New Mexico
October 12, 2004*

*Angkul Kongmunvattana
Department of Computer Science and Engineering
University of Nevada, Reno*

Contact: angkul@unr.edu

Motivation

Why this work is worth pursuing ?

*Insatiable demand for computing power
Advanced in microprocessor and
networking technologies*



Large-scale cluster computing platform

Performance
and
Scalability

Reliability
and
Availability

Simplicity
and
Accessibility

Existing Support and Problems

What we have and what do we need ?

- Shared Virtual Memory (SVM) Systems
 - + Simplified parallel programming tasks
 - + Competitive performance on a small-scale cluster
- High probability of node failures on large clusters
 - Waste of computation time for long-running applications
 - Loss of customers as critical services interrupted or become unavailable

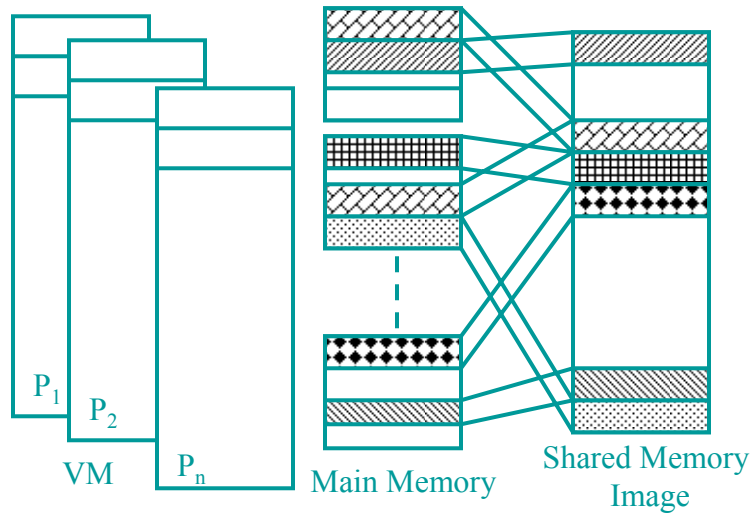
SVM

Its Components

- Inter-process communication (UDP)
- Reliable messages on UDP (Lightweight Protocol)
- Virtual memory mapping
- Memory coherence enforcement with LRC
- Queue-based lock in SW
- *Binomial spanning tree barrier in SW*

SVM

Shared Memory Address Space



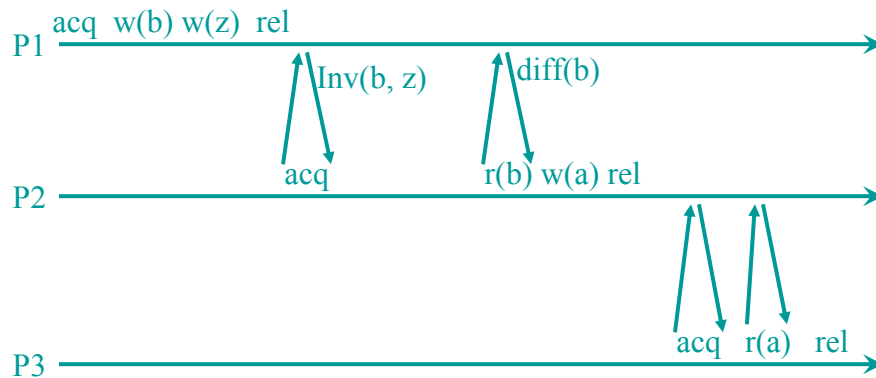
SVM

Its Performance

- Performance Factor
 - ◆ Number of messages exchanged
 - ◆ Amount of data transfer
- Performance Improvement Techniques
 - ◆ Lazy Release Consistency Model (Keleher ISCA'92)
 - ◆ Write-Invalidate Protocol
 - ◆ Diff Creation and Multiple-Writer Protocol (Carter et al)

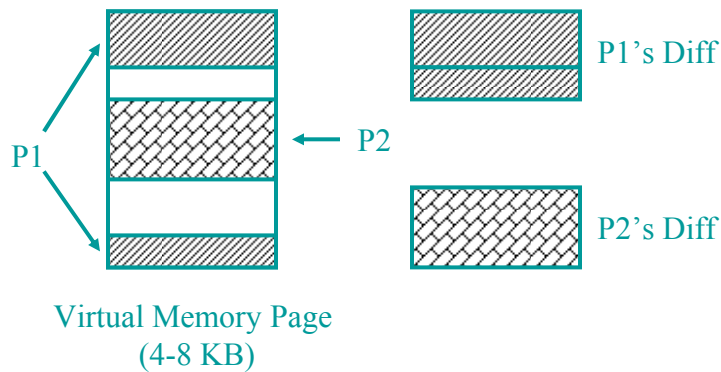
LRC and Write-Invalidate

When and how ?



Diff Creation and Multiple-Writer Protocol

Data Transfer Reduction



Fault-Tolerant Support in SVM

Previous Proposals

- Coordinated Checkpointing in SVM
 - + SC-based SVM
 - Estimation (Carter et al. COMPCON'93)
 - Simulation (Janakiraman and Tamir SRDS'94)
 - Real system (Cabillic et al. SRDS'95)
 - + LRC-based SVM
 - Partially implemented on real system (Costa et al. OSDI'96)

Fault-Tolerant Support in SVM

Previous Proposals (2)

- Independent checkpointing and message logging in SVM
 - + SC-based SVM
 - Theory (Richard III and Singhal SRDS'93)
 - + LRC-based SVM
 - Emulation (Suri, Janssens, and Fuchs FTCS'95)
 - Real system (Costa et al OSDI'96)
 - Simulation (Park and Yeom IPPS'98)

Coherence Centric Fault Tolerance

Our approach for fault-tolerant support in SVM

- + Correct recovery upon failure(s)
- + Transparent to users (no modification to applications)
- + Low overhead during a fault-free execution
- + Saving completed computation work
- + Shortening crash recovery time

Coherence-based Coordinated Checkpointing

Creation of Parallel Execution Snapshot

- Saving computation work
- Reducing services re-initiation time
- Simple implementation
- Low overhead

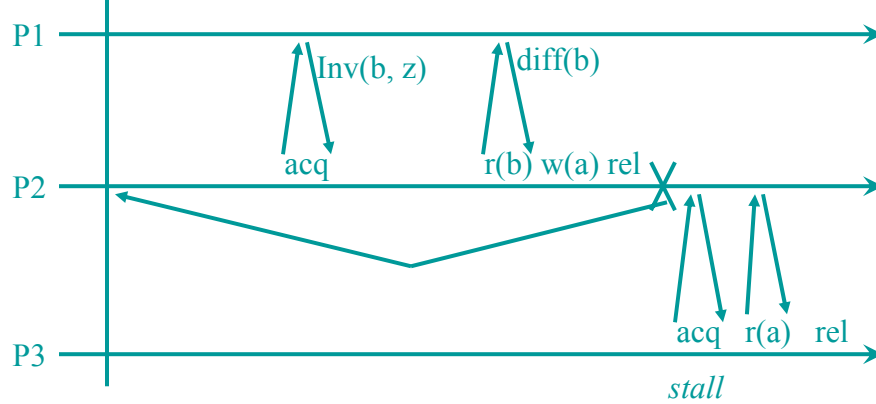
What do we need to keep ?

- Stack segment
- Text segment
- Shared data segment
- Local data segment (local variables, diffs, WI-notice, etc.)

Fault Recovery

Checkpoint Restoration

Checkpoint Creation

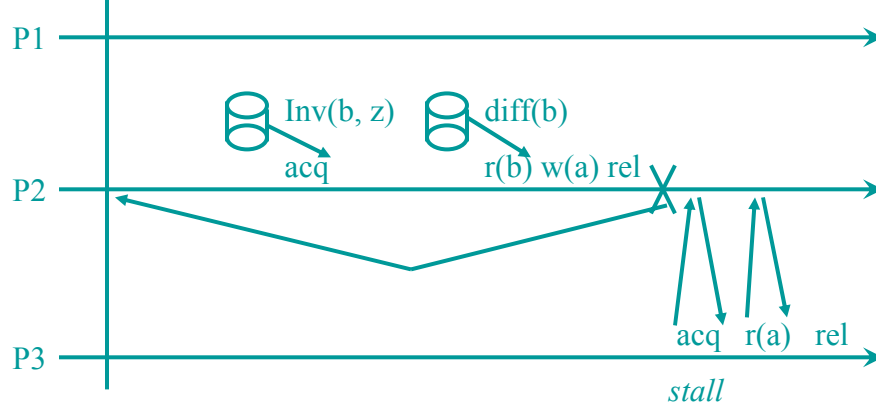


Integrated with Message Logging

Why ?

- Eliminate messages exchanged during recovery

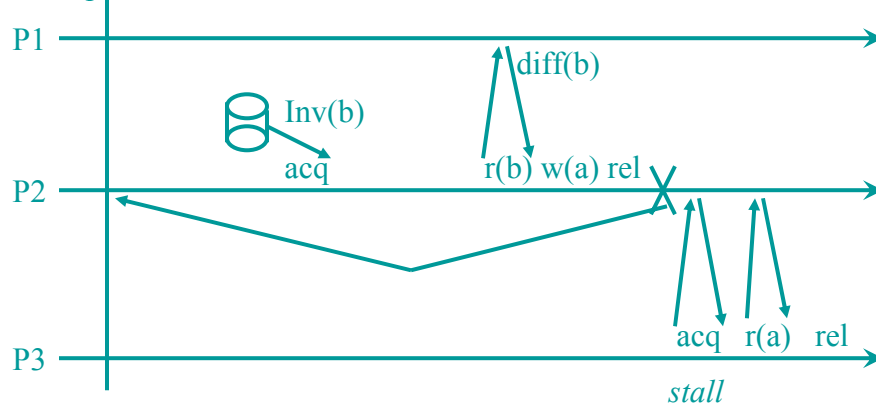
Checkpoint Creation



Coherence-Centric Logging Our Proposal - Part One

- Track the update requests within each interval

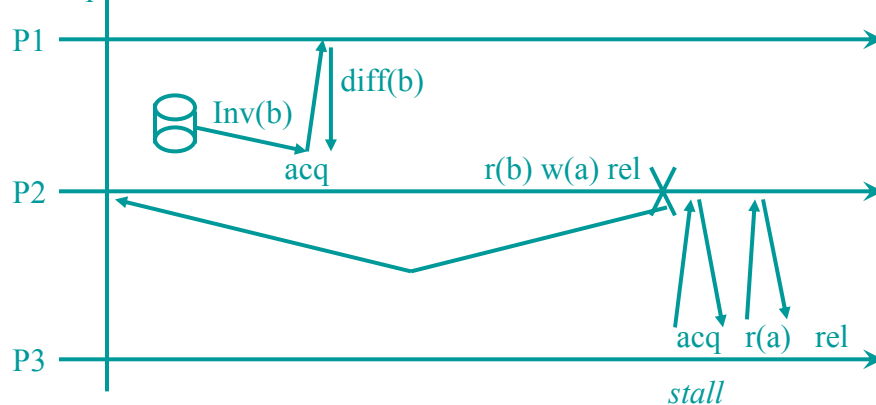
Checkpoint Creation



Run-ahead Crash Recovery Our Proposal - Part Two

- Prefetch Shared Data (Eliminate SEGV and Reduce Msg)

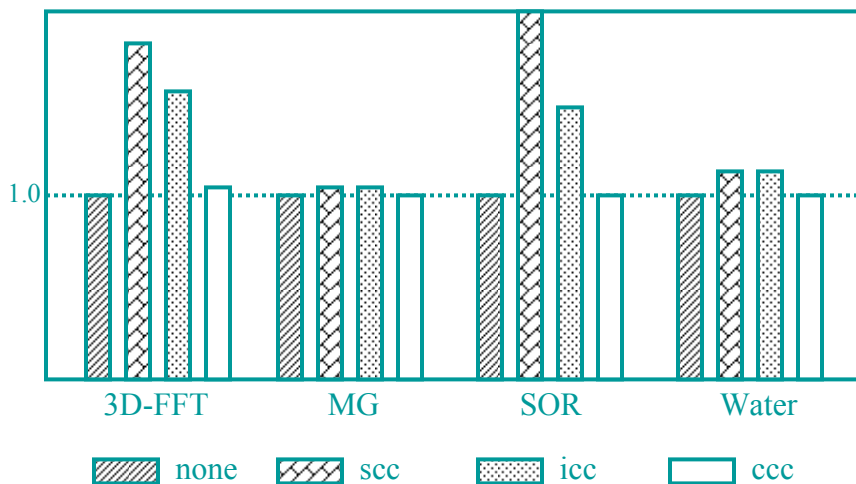
Checkpoint Creation



Coherence-Centric Fault Tolerance *Summary*

- Coherence-based Coordinated Checkpoint (CCC)
 - ◆ Re-create consistent state of execution
 - ◆ Saving computation work and reducing services re-initiation time
- Coherence-Centric Logging (CCL)
 - ◆ Eliminate synchronization messages
 - ◆ Allow data prefetch and multithreaded requests
- Run-ahead Crash Recovery (RCR)
 - ◆ Eliminate memory miss idle time (i.e., page faults and messages for memory update)

Checkpointing Overhead *Results*



Checkpointing Overhead

Detailed Results

Overhead per checkpoint for SOR

Checkpointing Technique	Checkpoint Components				Creation Time (Sec.)
	SH	Local	Stack	Total	
SCC	36	40.2	2.4	76.3	120.55
ICC	4.5	40.2	2.4	44.7	59.81
CCC	0	2	2.4	2.01	2.77

Message Logging Overhead

SOR

Logging Technique	Execution Time (Sec.)	Percentage Overhead	Log Size (KB)	# of Flushes
None	445.2	-	-	-
CCL	452.1	1.5	19	70
RSL	471.6	5.9	512	703

Crash Recovery Overhead

SOR

Logging Technique	Recovery Time (Sec.)	# of Sync. Messages	# of data Request	# of segv
None	445.2	9062	140	3219
PCR	410.5	0	140	0
SCR	664.7	0	140	3219

Conclusion and Future Work

What we achieved ? What's next ?

We make fault-tolerant support in SVM cluster affordable and attractive!

Ongoing Research

- Highly Available SVM on Large-Scale Clusters
 - ◆ Dynamic Resource Discovery
 - ◆ Automatic System Reconfiguration
 - ◆ Transparent Task Migration