Reducing Diff Overhead in Software DSM Systems using RDMA Operations in InfiniBand

Ranjit Noronha and Dhabaleswar K. Panda
Department of Computer Science and Engineering
The Ohio State University
Outline

- *Introduction and Motivation*
  - Software DSM
  - Modern computer networks

- Design and Implementation
  - Diff creation and Issues
  - Protocol Examples
  - Design Challenges

- Experiments
  - Application characteristics
  - Results

- Conclusions and Future Work
Software DSM

- Software DSM (SDSM)
  - HLRC/VIA (Rutgers), TreadMarks (Rice)

- Depends on user and software layer

- Depends on communication protocols provided by the system such as TCP, UDP, etc.

- Degraded performance because of false sharing and high overhead of communication

- Has scaling problems
HLRC

- HLRC/VIA (Rutgers)
  - Home Based Lazy Release Consistency Model
  - Page Based DSM System
- Internal basic operations
  - Page
  - Diff
  - Lock
HLRC Programming Example

- Initial value of $X = 0$
- $B$ is home node for page $P$ containing $X$

A

- Acquire_Lock (L1)
- $X = X + 1$
- Release_Lock(L1)

B

- Acquire_Lock (L1)
- $X = X \times 2$
- Release_Lock(L1)

Read page $P$ (containing $X$) from $B$

Send diffs for $P$ to $B$
Modern Interconnects

- InfiniBand, Myrinet, Quadrics
- Low Latency (InfiniBand 4.8 µs)
- High Bandwidth (InfiniBand 4X upto 10 Gbps)
- Programmable NIC
- User Level Protocols (VAPI, GM, Elan-4)
- Can deliver performance close to that of the underlying hardware
- RDMA Write/Read, Atomic Operations, Service Levels, Multicast
SDSM and Modern Networks

- DSM applications are communication intensive
- Latency critical (request messages)
- Bandwidth intensive (response messages)
- InfiniBand is a high-bandwidth/low-latency network
- Can InfiniBand be exploited to deliver better performance?
HLRC and InfiniBand

Software DSM Requirements

Barrier/Write notice

Locks

Page fetching

Signal handler

Diff's

Multicast

Atomic operations

RDMA Write

RDMA Read

Service Level

IBA Features

Outline

- Introduction and Motivation
  - Software DSM
  - Modern computer networks

- Design and Implementation
  - Diff creation and Issues
  - Protocol Examples
  - Design Challenges

- Experiments
  - Application characteristics
  - Results

- Conclusions and Future Work
Diffing in HLRC

- Each page assigned a home node
- Execution divided into intervals
- Updates sent at the end of intervals in the form of diffs to the home node
- Create a run-length encoding of the dirty page and its clean copy
- Diffs sent to home node
Diff Issues

- Diffs can be fairly large
- Send diffs together
  - improve bandwidth
- Breakdown diffs
  - Earlier update
- Which approach is best?
Terminology

- Default diff protocol
  - Called ORIG
  - Sends an individual diff, then waits for an ACK

- Protocol with packing/pipelining
  - Called PIPE
  - Can “pack” several diffs together
  - Multiple outstanding unacked diffs
ORIG protocol

- Node A arrives at a synchronization point
- Must send all diffs to home node
- For each page modified
  - Compare to twin and create a run-length encoding
  - Send to home node B
  - Home node applies diff and sends ACK to node A
  - Source continues with computing the next diff
ORIG protocol example
PIPE protocol

- Node A arrives at a synchronization point
  - Starts computing diffs by comparing dirty pages to their clean copies
  - Creates and copies run-length encoding into a buffer X of predefined length
  - Continues to create diffs and copy them into buffer X until it is full
  - Sends this to the home node B
  - If there are additional buffers available, use them without waiting for an ACK from B
- Node B applies the diffs and sends an ACK back for the corresponding buffer
PIPE protocol example

DIFF (P1,P2)

DIFF (P3,P4)

Combined ACK (P1,P2)
Design Challenges

- Network Primitives
- Pipeline Depth
- Packed Diff Size
Network Primitives

- RDMA
  - No need to post a receive descriptor
  - Can write directly into destination processes memory

- Send/Receive
  - Need a posted descriptor

- RDMA v/s Send/Receive
  - RDMA shows better performance over InfiniBand
Pipeline Depth

- Number of packed diffs that may be sent before waiting for an ACK
- Longer pipeline
  - network lightly loaded
- Shorter Pipeline
  - Network heavily loaded
- Practically depth=2 best
  - Can achieve sufficient overlap
Packed Diff Size

- Larger
  - Updates delayed
  - Better bandwidth utilization

- Smaller
  - Updates earlier
  - Lower bandwidth utilization
Outline

- Introduction and Motivation
  - Software DSM
  - Modern computer networks
- Design and Implementation
  - Diff creation and Issues
  - Protocol Examples
  - Design Challenges
- Experiments
  - Application characteristics
  - Results
- Conclusions and Future Work
Experimental Setup

- HLRC/ VIA (Rutgers) modified to work with VAPI
- InfiniScale MTS2400 24 port switch
- Mellanox InfiniHost MT23108 DualPort 4X HCA’s
- 16 node cluster
- 8 SuperMicro SUPER P4DL6
  - Dual Pentium Xeon 2.4 GHz
  - 512 MB memory
  - 133 MHz PCI-X bus
- 8 SuperMicro SUPER X5DL8-GG
  - 1 GB memory
  - 133 MHz PCI-X bus
- Linux 2.4.22 kernel
Applications

- 4 applications were evaluated (SPLASH-2)
  - Barnes
  - Integer Sort (IS)
  - Non-contiguous LU decomposition (LU)
  - Non-contiguous Ocean simulation (Ocean)
- Different communication patterns
- Communication intensive
Application Characteristics

- **Barnes**
  - N-body simulation using the hierarchical Barnes-HUT method
  - Sharing pattern irregular and true

- **IS**
  - Bucket Sort
  - Global array contains buckets

- **LU**
  - Factors a dense matrix into the product of a lower and upper triangular matrix
  - Exploits blocking for temporal locality on individual sub-matrices

- **Ocean**
  - Simulates large scale ocean movements based on eddy and boundary currents
  - Uses locks for synchronization
• LU and Ocean send a large number of diffs
• IS sends diffs of the order of 4K
• Each interval marks a synchronization point like a lock or barrier

• Ocean has the highest number of intervals
• Barnes and IS send a large number of diffs every interval
Diff Characteristics Summary

- **Barnes**
  - Few Small diffs
  - Large diff burst size

- **IS**
  - Send few large diffs
  - Large diff burst size

- **LU and Ocean**
  - Large number of diffs
  - Small diffs
  - Smaller diff burst size
Average diff traffic per node generated by the application

IS has the largest diff traffic
• LU has considerable traffic in the network
• 35% reduction in execution time for LU
Remaining Bottlenecks

- Barrier wait time 45% for Barnes
Outline

- Introduction and Motivation
  - Software DSM
  - Modern computer networks
- Design and Implementation
  - Diff creation and Issues
  - Protocol Examples
  - Design Challenges
- Experiments
  - Application characteristics
  - Results
- Conclusions and Future Work
Conclusions

- Explored reducing diff overhead
- Diff packing and pipelining implemented
- Different applications evaluated
- Reduction in execution time upto 35%
- All applications benefited
Future Work

- Read diffs using RDMA Read
  - Node computes the diffs
  - Diff is stored
  - Other nodes read it on demand

- Investigate wait times
  - Efficient barrier
  - Effect of sequential phases
Web Pointers

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

E-mail: {noronha, panda}@cse.ohio-state.edu
Backup Slides
## Application Characteristics

<table>
<thead>
<tr>
<th>Application</th>
<th>Barnes</th>
<th>IS</th>
<th>LU</th>
<th>Ocean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Diff Traffic (MegaBytes)</td>
<td>1.83</td>
<td>29.55</td>
<td>10.9</td>
<td>9.16</td>
</tr>
<tr>
<td>Average number of Diffs</td>
<td>6060</td>
<td>7680</td>
<td>15114</td>
<td>14327.56</td>
</tr>
<tr>
<td>Average Diff Size (bytes)</td>
<td>317</td>
<td>4034</td>
<td>756.21</td>
<td>670.38</td>
</tr>
<tr>
<td>Average Number of Intervals</td>
<td>13</td>
<td>17</td>
<td>129</td>
<td>937</td>
</tr>
<tr>
<td>Average number of Diffs per interval</td>
<td>466.15</td>
<td>451.76</td>
<td>117.16</td>
<td>15.29</td>
</tr>
<tr>
<td>Average traffic including Diff Traffic (MegaBytes)</td>
<td>48</td>
<td>94.24</td>
<td>964.62</td>
<td>157.77</td>
</tr>
</tbody>
</table>