JUMP-DP:
A Software DSM System with Low-Latency Communication Support

Benny Cheung, Cho-Li Wang, Kai Hwang
Department of Computer Science and Information Systems
The University of Hong Kong
Outline

• Introduction
• Our Objective
• The Migrating-Home Protocol
• Socket-DP
• Performance of JUMP-DP
• Observations
• Conclusion
Introduction

* Distributed Shared Memory (DSM)

\[ \begin{array}{cccc}
P_0 & P_1 & P_2 & \ldots & P_{n-1} \\
\text{MEM} & \text{MEM} & \text{MEM} & \ldots & \text{MEM} \\
\end{array} \]

Main Issue: To maintain memory consistency in different processors of the DSM system

Performance bottleneck: Communication in the Network
Previous Work

Unit of Memory
- Page-based
- Object-based

Consistency Model
- Sequential (SC)
- Eager Release (ERC)
- Lazy Release (LRC)
- Entry (EC)
- Scope (ScC)

Coherence Protocol
- Page Update
- Page Invalidate

O-O Language
- Emerald

Compiler
- Munin
- TreadMarks
- Midway
- JIAJIA

IVY

References:
- Blumrich94
- Carter91
- Keheler94
- Bershad93
- Iftode96
- Lamport79
- Zhou96
**Milestone DSM Systems**

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVY</td>
<td>1st Software DSM</td>
</tr>
<tr>
<td></td>
<td>Sequential Consistency (inefficient)</td>
</tr>
<tr>
<td>TreadMarks</td>
<td>Lazy Release Consistency (better)</td>
</tr>
<tr>
<td></td>
<td>Most Popular Software DSM</td>
</tr>
<tr>
<td>Midway</td>
<td>Entry Consistency</td>
</tr>
<tr>
<td></td>
<td>Very Efficient but hard to program</td>
</tr>
</tbody>
</table>

* Any efficient DSM with good programmability?
Our Objective

• To alleviate the network bottleneck.

• **JUMP-DP**: Two software solutions
  - **Migrating-Home Protocol** on ScC:
    * reducing the volume of data in the network e.g. relaxed memory model / protocol
  - **Socket-DP**:
    * improving the speed of communication by reducing the network protocol overhead.
Scope Consistency (ScC)

- A relaxed consistency model \([lftode96]\)
  - weaker than LRC
  - efficient, good programmability
  - “Scope” : all critical sections using same lock; opens at acquire, closes at release

Efficiency

Programmability
Scope Consistency (ScC)

- When a processor Q opens a scope previously closed by another processor P, P propagates the updates made within the same scope to Q.

In LRC, P propagates both the updates of x and y to Q while in ScC, P propagates the update of y only since only y is updated in the same scope as it is read by Q.
Migrating-Home Protocol (MHP)

- **Features of the Protocol:**
  - allows the home location of each page in DSM to **change** during program execution
  - the home of X is migrated from P to Q when Q requests the page from P, if the copy of X possessed by P is **totally** clean
  - Q’s updates need not propagate to other processors ➔ reduces network traffic
Important Data Structures

- **Migration Notice:**
  - short message to notify other processors in the cluster about the home change
  - broadcast nature: performance bottleneck?
  - concatenation of multiple migration notices

- **Diff:**
  - updates of a page by non-home processor
  - deals with false sharing
An Illustration of MHP

P0 gets the copy of X and is granted the new home of X
P2 gets the copy of X and the new home location

Migration notices

Diff
## 4 Different Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homeless (TreadMarks)</td>
<td>No fixed processor to store the most up-to-date copy of a page</td>
</tr>
<tr>
<td>Home-based (JIAJIA V1.1)</td>
<td>A fixed processor storing the most up-to-date copy of a page</td>
</tr>
<tr>
<td>Home Migration (JIAJIA V2.1)</td>
<td>The processor storing the most up-to-date copy of a page is changed at barrier synchronization</td>
</tr>
<tr>
<td>Migrating-Home (MHP) (JUMP)</td>
<td>The processor storing the most up-to-date copy of a page can be changed when serving a page fault</td>
</tr>
</tbody>
</table>
Comparing the 4 Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Comment on Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homeless</td>
<td>Serving a page fault may issue requests on multiple processors</td>
</tr>
<tr>
<td>Home-based</td>
<td>More efficient than homeless [Zhou96] but fixed home not well-adapted to access patterns</td>
</tr>
<tr>
<td>Home Migration</td>
<td>Try to adapt to DSM access patterns but home migration rule is too strict</td>
</tr>
<tr>
<td>Migrating-Home (MHP)</td>
<td>Adapt well to DSM access patterns while the home migration rule is more aggressive</td>
</tr>
</tbody>
</table>
Socket-DP

- A **low-latency** communication support
- Beneficial to DSM
  - transmission of short control messages
  - substantially reduces the startup cost

**Characteristics:**
- techniques to reduce protocol overhead
- features to enhance usability and user-friendliness
Socket-DP Design

- Directed-Point Model [Zhu2000]
Socket-DP Operation

1. Send Process (P0) calls sendto() to send a message.
2. The message is sent to the Token Buffer Pool.
3. The message is enqueued in the Outgoing FIFO queue.
4. The message is sent to the network.

5. Recv Process (P1) calls recvfrom() to receive a message.
6. A SIGIO signal is generated.
7. The message is received by the Token Buffer Pool.
8. The message is dequeued from the Incoming FIFO queue.
9. The message is delivered to the recv msg function.

Page Re-mapping is performed at the Kernel Level.
Reducing Protocol Overhead

- **Token Buffer Pool**: allows the Interrupt Handler to directly copy incoming messages to the dedicated buffer spaces through page re-mapping.

- **Light-weight Messaging Calls**: allows kernel level transmission routines to be triggered as light-weight messaging calls, reducing context switching overhead.
Enhancing Usability

• Supports **Asynchronous Send/Receive** with **signal handling**:
  - delivers a **SIGIO** to the receiving process

• **Message Assembly/Disassembly**:
  - to accommodate network requirements

• **A familiar user interface**:
  - use UNIX system calls `socket()`, `bind()`, `sendto()`, `recvfrom()` and `select()`
P2P Round-Trip Time

Improvement = \frac{(\text{BSD Socket Time} - \text{Socket-DP Time})}{\text{Socket-DP Time}}
Performance Evaluation

- Compare **JUMP-DP** with other systems:
  - **JIAJIA V1.1**: Home-based + BSD Sockets
  - **JUMP**: MHP + BSD Sockets
  - **JIAJIA V2.1**: Home Migration Protocol + BSD Sockets

- **Testing environment**:
  - 16 PIII 450MHz PCs, 128MB RAM each
  - Fast Ethernet + 100-based Switch
JUMP-DP Performance

(a) Matrix Multiplication (MM)

(b) Merge Sort (ME)

(c) Radix Sort (RX)

Legend
- JIAJIA V1.1
- JIAJIA V2.1
- JUMP
- JUMP-DP

Y-Axis: Time (sec)
JUMP-DP Performance

(d) LU Factorization (LU)

(e) Bucket Sort (BK)

(f) Red-Black SOR (SOR)

Legend
- JIAJIA V1.1
- JIAJIA V2.1
- JUMP
- JUMP-DP

Y-Axis: Time (sec)
## Observations

### Comparison | Observations
--- | ---
**JUMP over JIAJIA V1.1** | - Improvement in 5 out 6 programs  
- Maximum 3.16 times faster  
- MHP beats home-based protocol  
- JUMP favors larger programs

**JUMP-DP over JUMP** | - Socket-DP improves performance

**JUMP over JIAJIA V2.1** | - JUMP beats JIAJIA in 5 programs  
- JUMP’s MHP is more efficient
Conclusions & Future Work

- **Conclusions:**
  - MHP reduces network traffic
  - Socket-DP reduces communication latency
  - Improve DSM performance substantially

- **Future Work:**
  - Porting JUMP-DP to JESSICA 2 project ([http://www.srg.csis.hku.hk/jessica.htm](http://www.srg.csis.hku.hk/jessica.htm))
  - Further improvement of the MHP
HKU JESSICA 2 Project

ClusterProbe
(Java-based Cluster Monitoring Tool)

Socket-DP

DP-II (Gigabit Ethernet on HVS cluster)

Delta Execution
(thread migration)

JUMP-DP
(Software DSM)

Global Object Space

Single Thread Space