Treadmarks: Distributed Shared Memory on Standard Workstations and Operating Systems

P. Keleher, A. Cox, S. Dwarkadas, and W. Zwaenepoel
The Winter Usenix Conference 1994

 발표 : 이동은 (2006–21240)
Introduction

• DSM (distributed shared memory)
  – For most programmers, easier to use than a message passing paradigm
    • With a global address space, the programmer can focus on algorithmic development rather than on managing partitioned data sets and communicating values

• TreadMarks
  – User level implementation w/o modification to the operating system kernel
  – Runs on commonly available Unix systems

![Distributed Shared Memory Diagram](image.png)
TreadMarks Design 1

- **Release consistency**
  - Relaxed memory consistency model that permits a processor to delay making its changes to shared data visible to other processors until *release* operation occurs
    - Reduces # of messages
    - Reduces latency

![Diagram showing release consistency](image)

Sequential consistency

Release consistency
TreadMarks Design 2

- **Lazy release consistency**
  - The propagation of modifications is postponed until the time of acquire
  - At this time, the acquiring processor determines which modifications it needs to see
  - Fewer messages

- **Cf) Eager release consistency**
  - Write access information is delivered to all the shared copies at the release point
  - Release blocks until acknowledgments have been received from all others

---

Diagram showing the difference between eager and lazy release consistency.
TreadMarks Design 3

- **Multiple-Writer protocols**
  - **False sharing problem for early DSM systems (Single-writer protocols)**
    - A write to any variable of a page causes the entire page to become invalid on all other processors that cache the page.
    - A subsequent access on any of these processors incurs an access miss and causes the modified copy to be brought in over the network.
  - **Multiple-Writer protocol**
    - Two or more processors can simultaneously modify their local copy of a shared page.
    - Their modifications are merged at the next synchronization operation, thereby reducing the effect of false sharing.
TreadMarks Design 4

- Lazy diff creation
  - Allows diff creation to be postponed until the modifications are requested
  - Decrease in the number of diffs created

If there is a request on this page or a write notice arrives:

- Create twin

run-length encoding
TreadMarks Operation Example

1: (A) Acquire → two writes → Release
2: (B) Acquire send CVT (Current Vector Timestamp) Lock request
3: (A) reply write notices (it means B’s page is invalid) Lock grant
4: (B) access the page (access miss!), delta(diffs) request (Lazy diff creation)
5: (A) send diffs
Implementation

- **PageArray**: one entry for each shared page
  - Current state: no access, read-only access, or read-write access
  - Approximate copyset: specifying the set of processors that are believed to currently cache this page
- **ProcArray**: one entry for each processor
- **Interval Records**: contains the vector time stamp for that interval
  - Interval: a new interval begins at each release and acquire
- **Write Notice Records**
- **Diff Pool**

*Figure 2* Overview of TreadMarks Data Structures
• Interval Creation
  – Logically, a new interval begins at each release and acquire
  – In practice, interval creation can be postponed until a processor communicate w/ another processor
    • Avoiding overhead if a lock is reacquired by the same processor
• Locks
  – All locks have a statically assigned manager
  – Lock management is assigned in a round-robin fashion among the processors
  – The manager records which processor has most recently requested the lock
  – All lock requests are directed to the manager
• Barriers
  – Barriers have a centralized manager
• Garbage collection
  – Necessary to reclaim the space used by write notice records, interval records, and diffs
  – During garbage collection, each processor validates its copy of every page that it has modified (similar to barrier operation)
    • All other pages, all interval records, all write notice records and all diffs are discarded
  – Triggered when the amount of free space for consistency information drops below a threshold
• Unix aspects
  – Relies on Unix and its standard libraries
  – Interprocess communication (socket)
    • SIGIO signal, select system call
  – Consistency protocol
    • mprotect system call to control access to shared pages
    • Any attempt to perform a restricted access on a shared page generates a SIGSEGV signal
Performance

• Experimental Environment
  – 8 DECstation-5000/240 running Ultrix V4.3
  – Connected to a 100-Mbps ATM LAN and a 10-Mbps Ethernet

• Applications
  – Water — molecular dynamics simulation
  – Jacobi — Successive Over-Relaxation (SOR)
  – TSP — branch & bound algorithm to solve the traveling salesman problem
  – Quicksort — using a bubblesort to sort subarrays of less than 1K element
  – ILINK — genetic linkage analysis
Experimental Results

**Figure 3**  Speedups Obtained on TreadMarks

<table>
<thead>
<tr>
<th>Input</th>
<th>Water</th>
<th>Jacobi</th>
<th>TSP</th>
<th>Quicksort</th>
<th>ILINK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (secs)</td>
<td>15.0</td>
<td>32.0</td>
<td>43.8</td>
<td>13.1</td>
<td>1113</td>
</tr>
<tr>
<td>Barriers/sec</td>
<td>2.5</td>
<td>6.3</td>
<td>0</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>Locks/sec</td>
<td>582.4</td>
<td>0</td>
<td>16.1</td>
<td>53.9</td>
<td>0</td>
</tr>
<tr>
<td>.Msgs/sec</td>
<td>2238</td>
<td>334</td>
<td>404</td>
<td>703</td>
<td>456</td>
</tr>
<tr>
<td>Kbytes/sec</td>
<td>798</td>
<td>415</td>
<td>121</td>
<td>788</td>
<td>164</td>
</tr>
</tbody>
</table>

**Figure 4**  Execution Statistics for an 8-Processor Run on TreadMarks
Execution Time Breakdown

Figure 5  TreadMarks Execution Time Breakdown

Figure 6  Unix Overhead Breakdown

Figure 7  TreadMarks Overhead Breakdown
Breakdown Summary

- The greatest potential to improve overall performance is reducing *the software communication overhead*
  - Lower-overhead user-level communications interfaces or a kernel-level implementation would improve performance
This is due to the fact that TSP is not a properly labeled program, which means read accesses are not synchronized.