TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems

Jeongho Kim, Youngseok Son
M.S. Student.
Real-Time Operating Systems Laboratory
Seoul National University
I. Introduction
II. Techniques
III. Implementation: TreadMarks
IV. Evaluation
The distributed shared memory (DSM) implements the shared memory model in distributed systems, which have no physical shared memory.

The shared memory model provides a virtual address space shared between all processors.

The high cost of communication in distributed systems is that DSM systems move data between processors.

I. Introduction

Distributed Shared Memory
I. Introduction

Problems of Existing DSM

- OS dependency
  - Kernel modifications

- Poor performance
  1. Communication overhead
  2. False sharing
Communication overhead

- If the communication occurs whenever the x variable is changed, it costs a lot of overhead.

Solution) The communication occurs, when the x variable is changed finally.
2. False sharing

- A situation in which two or more processes access different variables within a page.
  - If only one process is allowed to write to a page at a time, false sharing leads to unnecessary communication, called the “ping-pong” effect.
Objectives

- User-level implementation
- Good performance
  - Reduction of communication overhead
    - Lazy release consistency
    - Invalidate-based protocol
  - Solving false sharing
    - multiple writer protocol
Contents

I. Introduction
II. Techniques
III. Implementation: TreadMarks
IV. Evaluation
Related Works

1. Lazy release consistency
2. Multiple-writer protocol
1. Lazy Release Consistency

- Release Consistency (RC) is a memory consistency model that permits a node to delay making its changes of shared data visible to other nodes until certain synchronization accesses occur.

- Shared memory accesses
  - Ordinary access
  - Synchronization access
    - acquire access
    - release access

- Example

```
Process  acq(L₁)  w(x)₁  w(y)₁  rel(L₁)  t
L₁: lock related to x, y
```
1. Lazy Release Consistency

- Two types of RC
  1. Eager release consistency
  2. Lazy release consistency
1. Lazy Release Consistency

Eager release consistency (ERC) postpones sending the modifications to the next release.

- Understanding ERC

- Release operation does not complete (is not performed) until the acknowledgements from all the processes are received.
- $L_1$ is lock related to $x$, $y$ and $L_2$ is lock related to $z$
1. Lazy Release Consistency

① Eager release consistency (ERC) postpones sending the modifications to the next release.

- Understanding ERC

• Release operation does not complete (is not performed) until the acknowledgements from all the processes are received.
• \(L_1\) is lock related to \(x, y\) and \(L_2\) is lock related to \(z\)
Lazy release consistency (LRC) postpones sending of modifications until a remote processor actually needs them.

- Understanding ERC

It is guaranteed that the acquirer of the same lock sees the modification that precede the release in program order.

How did process B know whether variable x has to be updated?

1) Write notice
2) Timestamp
1. Lazy Release Consistency

② Lazy release consistency (LRC)

1) Write Notice
   - The releaser send write notice to all other processes
1. Lazy Release Consistency

2. Lazy release consistency (LRC)

2) Timestamp
   • Satisfying the happened-before relationship between all operations is enough to satisfy LRC
   • The ordering is applied to process intervals.
     – Interval is a segment of time in the execution of a single process.
     – New interval begins each time a process executes a synchronization operation.
1. Lazy Release Consistency

II. Related Works

② Lazy release consistency (LRC)

2) Timestamp
   • It is implemented as a vector timestamp

![Diagram of process interactions with timestamps](attachment:image.png)
2. Multiple-Writer Protocol

- Multiple-writer protocol is designed to solve false sharing
  - It is possible that several processes make modifications to different variables at the same page

- Two techniques
  - Twin
    - It is a copied page of original page
    - It will be compared with a changed page.
  - Diff
    - diff is a difference between twin and ‘copyset’
II. Related Works

2. Multiple-Writer Protocol

- Two techniques (cont’d)
  - **Twin**

```
write P  twin
          writable working copy
```

- **Diff**

```
release:  diff
```

- Note that twinning and diffing not only allows multiple independent writers but also significantly reduces the amount of data sent.
Summary

- Solving communication overhead
  - Lazy release consistency

- Solving false sharing
  - Multiple-writer protocol

- Implementation
  - Lazy release consistency + Multiple-writer protocol
  - Eager release consistency + Multiple-writer protocol
II. Related Works

Summary

- Eager release consistency + multiple-writer protocol
Lazy release consistency + multiple-writer protocol
Contents

I. Introduction
II. Techniques
III. Implementation: TreadMarks
IV. Evaluation
Implementation

1) Data structures
2) Interval and Diff creation
3) Access misses
4) Garbage collection
5) Etc
1) Data Structure (1/2)

- Lazy release consistency
  - Write notice
  - Timestamp
- Multiple-writer protocol
  - Diff
III. Implementation: TreadMarks

1) Data Structure (2/2)

If there is a write notice record in a page, happen before relationship should be considered through the vector time stamp.

- Page Array
  - Page State
  - Copyset
  - Write Notice Records
    - Write Notice Record 1
    - Write Notice Record 2
    - Write Notice Record 3

- Vector Timestamp
  - Diff value is managed by diff pool for efficient memory management

- Process Array
  - 1 2 3 4 ...

- Processor Array
  - 1 2 3 4 ...

- Diff Pool
2) Interval and Diff creation

- **Interval creation**
  - Logically
    - a new interval begins at each release and acquire
  - In practice
    - Interval creation can be postponed until we communicate with another process, avoiding overhead if a lock is reacquired by the same processor

- **Diff creation**
  - With lazy diff creation, these pages remain writable until a diff request or a write notice arrives for that page
    - a subsequent write results in a write notice for the next interval
3) Access Misses

- When access miss occurs
  - without write notice
    - Initially setup that processor 0 has the page
  - with write notice
    1. Get the diffs from the write notice with small timestamp
    2. Create an actual diff which is correction of all diff related to the page
    3. The twin is discarded and the result is copied to copyset

III. Implementation: TreadMarks
4) Etc.

- Lock & barrier
  - Statically assigned manager

- Garbage collection
  - It reclaim the space used by write notice records, interval records, and diffs
  - It is triggered when the free space drops below a threshold

- Unix Aspects
  - TreadMarks relies on Unix standard libraries
    - Remote process creation, interprocessor communication, and memory management.
Contents

I. Introduction
II. Techniques
III. Implementation: TreadMarks
IV. Evaluation
Experimental Environment

- **Experimental Environment**
  - 8 DECstation-5000/240
  - Connected to a 100-Mbps ATM LAN and a 10-Mbps Ethernet

- **Applications**
  - Water – molecular dynamics simulation, 343 molecules for 5 steps
  - Jacobi – Successive Over-Relaxation with a grid of 2000 by 1000 elements
  - TSP – branch & bound algorithm to solve the traveling salesman problem for 19 cities
  - Quicksort – sorts an array of 256K integers. Using bubblesort to sort subarray of less than 1K element
  - ILINK – genetic linkage analysis
Results

- **Speedup**

- **Message rate**
  - messages / sec
IV. Evaluation

Results

- Diff creation rate
  - diffs / rate

![Graph showing diff creation rates for different algorithms and execution strategies. The x-axis represents different algorithms (Water, Jacobi, TSP, Qsort, ILINK), and the y-axis represents the number of diffs per rate. The graph compares Lazy and Eager execution strategies.]
End of Doc.