Treadmarks: Distributed Shared Memory on Standard Workstations and Operating Systems


Presented by
2008-20864
Honggyu Kim
Treadmarks

• Motivation
  • No widely available DSM implementations
  • Performance problems

• Objectives
  • Determine efficiency of user-level DSM implementation
    • Reduce communication overhead
  • Commercially available workstations and OS
    • Standard Unix system on DECstation
Distributed Shared Memory

- **Software-based distributed shared memory (DSM)**
  - Provide an illusion of shared memory on a cluster
    - A private address space in each node, but a single globally shared address space on the cluster
  - Keep the main memories of the nodes coherent
    - Embedding a coherence protocol in the page fault handlers

- **High overheads of**
  - The protocol processing
    - Implemented in software
  - The large granularity (page) of coherence and communication
    - false sharing
TreadMarks Motivation (1)

- **Sequential Consistency**
  - Every write visible “immediately”
- **Problems**
  - number of messages
  - latency

![Diagram]

- $w(x)$
- $w(y)$
- $x$
- $y$
- $r(x)$
- $r(y)$
TreadMarks Motivation (2)

- **False sharing**
  - Pieces of the same page updated by different processors
  - Leads to “ping-pong” effect, increasing network traffic enormously
Eager Release Consistency (ERC) (Munin)

- Write access information is delivered to all the shared copies at the release point.
- Release blocks until acknowledgments have been received from all others.
Lazy Release Consistency (LRC)

- Write access information is delivered only to the next acquiring copy at the next acquire point.
- Fewer messages than ERC
- TreadMarks uses LRC model

Variables x and y are in the same page p

```
P0  P1  P2
r(x)  acq(L)  r(y)
w(x)  w(x)  rel(L)

r(y)
acq(L)
```

```
grant L + inv. p
```
ERC vs. LRC

ERC

P0
r(x)
acq(L)
w(x)
rel(L)
inv.(L)

P1
r(y)
acq(L)
w(x)
rel(L)
inv.(L)

P2
r(x)
acq(L)
w(x)
rel(L)
inv.(L)

LRC

P0
r(x)
acq(L)
w(x)
rel(L)
inv.(L)

P1
r(y)
acq(L)
w(x)
rel(L)
inv.(L)

P2
r(y)
acq(L)
w(x)
rel(L)
inv.(L)
Multiple Writer Protocol

- Mitigates the effect of false sharing by making each processor’s local copy of a shared page.

  - Variables $x$ and $y$ are in the same page $p$

- Their modifications are merged at the next synchronization operation:
  - at release points.
Variables $x$ and $y$ are in the same page $p$.

**P0**
- $r(x)$
- acq(L0)
- w(x)
- rel(L0)

**P1**
- $r(y)$
- acq(L1)
- w(y)
- rel(L1)

$\text{twin}$

$\text{diff}$

$x \ y + y = x \ y$

$r(x) \rightarrow y$

$y \rightarrow r(x)$

$x \rightarrow x \ y = x \ y + x$
Lazy Diff Creation

Variables x and y are in the same page p

\[ \text{P0} \]
- \text{acq}(L)
- \text{w(x)}

\[ \text{P1} \]
- \text{acq}(L)
- \text{r(y)}

- \text{grant L + inv. p}
- \text{make a twin}
- \text{create diff}
- \text{apply diff}

\[ \text{diff} \]
MWP vs. LDC

- **Multiple Writer Protocol**
  - A diff is created for each modified page and propagated to all other copies of the page at each release.

- **Lazy Diff Creation**
  - Allows diff creation to be postponed until the modifications are requested.
  - TreadMarks uses Lazy Diff Creation model.
Performance

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

• **Applications**
  - Water – molecular dynamics simulation
  - Jacobi – Successive Over-Relaxation
  - TSP – branch & bound algorithm to solve the traveling salesman problem
  - Quicksort – using bubblesort to sort subarray of less than 1K element
  - ILINK – genetic linkage analysis
Figure 3  Speedups Obtained on TreadMarks

Figure 4  Execution Statistics for an 8-Processor Run on TreadMarks
Execution Time Breakdown (1/2)

Figure 5  TreadMarks Execution Time Breakdown

Figure 6  Unix Overhead Breakdown
Execution Time Breakdown (2/2)

**Figure 7**  TreadMarks Overhead Breakdown

**Figure 8**  Execution Time for Water
Lazy vs. Eager Release Consistency (1/2)

**Figure 9** Comparison of Lazy and Eager Speedups

**Figure 10** Message Rate (messages/sec)
Lazy vs. Eager Release Consistency (2/2)

Figure 12  Diff Creation Rate (diffs/sec)
Conclusion

• Efficient user-level implementation

• Suggested novel techniques to reduce communication overhead
  • Lazy Release Consistency
  • Lazy Diff Creation

• Viable technique for parallel computation on clusters of workstations connected by suitable networking technology
  • Network speed has been improved