Implementing TreadMarks on GM over Myrinet: Challenges, Design Experience and Performance Evaluation

Ranjit Noronha
And
Dhabaleswar K. Panda
Department of Computer and Info. Science
The Ohio State University
E-mail: {noronha,panda}@cis.ohio-state.edu
Presentation Overview

• Introduction
• Motivation
• Overview of TreadMarks
• Overview of Myrinet and GM
• Design Challenges
• Performance Evaluation
• Conclusions
Introduction—Distributed Shared Memory (DSM)

- Abstraction of shared memory on physically distributed machines
- Expand the notion of virtual memory to different nodes
- 2 types
  - Software DSM; eg provides the shared memory abstraction on a network of workstations like TreadMarks (Rice), HLRC (Rutgers)
  - Hardware DSM; eg use cache consistency protocols to support shared memory between physically separate remote memories like SGI origin and Sequent NUMA-Q
Introduction - Software DSM

- Software DSM
  - Consistency model; lazy release consistency
  - Execution divided into intervals
  - Allows multiple writers to write to the same page by dividing it into smaller portions and creating diff's when required by a reader
  - Pages in the interval made consistent at synchronization points like a lock acquire or a barrier
- Software DSM Issues
  - 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
Motivation

- **Modern Interconnects**
  - Low Latency (InfiniBand and Myrinet < 10 us)
  - High Bandwidth (InfiniBand 10GBps, Myrinet 2 GBps)
- **User Level Protocols (ULP)**
  - Can deliver performance close to that of the underlying hardware
- **Software DSM over ULP**
- **How does Software DSM perform with efficient communications layers?**
- **Can Software DSM outperform/out Scale Hardware DSM?**
TreadMarks

- Developed at Rice University
  - Overview paper:

- Runs in user space (no modification to the kernel)
- Implements lazy release consistency protocol (LRC)
- User level memory management techniques
- Communication protocol-UDP
TreadMarks – Communication Model

Node 1
- Send Request
- Receive Response

Node 2
- Interrupt SIGIO
  - Request Handler
- Send Response
TreadMarks - Communication Primitives

Treadmarks Routines

Send Request/Response
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless

Recv Request
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless
- Automatic allocation of temporary buf

Recv Request/Response
- Contiguous/Noncontiguous
- From any user buffer
- Receiving from any node of a group

Send, sendto(contig)
Sendmsg(non-contig)

Recv
Recv, recvfrom(contig)
Recvmmsg(non-contig)

Recv_any
Select

SIGIO handler

UDP/TCP
Myrinet and GM

- **Myrinet**
  - Low latency, high bandwidth network
  - Full duplex links; 2+2 gigabits per second
  - Programmable Myrinet NIC; 200 MHz processor and up to 4 MB SRAM

- **GM**
  - User level protocol
  - Reliable, connectionless data delivery
  - Transmits to and from pinned, memory
  - No asynchronous notification
  - No scatter, gather operations
TreadMarks over GM-Challenges

- No asynchronous notification
  - Polling thread
  - Timer based implementation
  - Modify GM to generate an interrupt

- Buffer allocation
  - Buffer allocation automatic in UDP
  - GM buffers have to be allocated before the message arrives
  - TreadMarks disables interrupts
TreadMarks on GM: Challenges

- GM-Memory registration
- GM-Message length $l$ has to correspond to size $s = \log_2(l+2)$
- TreadMarks uses two ports between every process-GM allows for a maximum eight ports
TreadMarks over GM: Proposed Substrate

- Treadmarks Routines
  - Send Request/Response
    - Contiguous/Noncontiguous
    - From any user buffer
    - Connection oriented/Connectionless
  - Recv Request
    - Contiguous/Noncontiguous
    - From any user buffer
    - Connection oriented/Connectionless
    - Automatic allocation of temporary buffer

- Substrate
  - Connect
  - Synchronous Send
  - Synchronous Receive
  - Asynchronous Send
  - Asynchronous Receive
  - Buffer Management
  - Preposting Receive Buffers
  - Schemes for Handling Async. Messages

- GM
  - RDMA Write
  - Send
  - Recv
  - Interrupts

SIGIO handler
TreadMarks over GM: Proposed Substrate

- **Connection Management**
  - A single synchronous and asynchronous port per process
  - Allows for selectively generating an interrupt
  - More scalable
TreadMarks over GM: Proposed Substrate

Treadmarks Routines

Send Request/Response
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless

Recv Response
- Contiguous/Noncontiguous
- From any user buffer
- Receiving from any node of a group

Recv Request
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless
- Automatic allocation of temporary buffer

SIGIO handler

Connect

Synchronous Send

Synchronous Receive

Asynchronous Send

Asynchronous Receive

Preposting Receive Buffers

Schemes for Handling Async. Messages

RDMA Write

Send

Recv

Interrupts

Connection Management

Buffer Management

GM
TreadMarks over GM: Proposed Substrate

• Buffer Management
  - Send and receive buffers in registered memory
  - Messages copied between TreadMarks and GM buffers
  - Allows for message pipelining
  - Other solutions, pass a pointer to a buffer
    • Complicated—requires modifications to TreadMarks
TreadMarks over GM: Proposed Substrate

Treadmarks Routines

Send Request/Response
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless

Recv Request
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless
- Automatic allocation of temporary buffer

SigIO handler

Connect

Synchronous Send

Synchronous Receive

Asynchronous Send

Asynchronous Receive

Preposting Receive Buffers

Schemes for Handling Async. Messages

GM

RDMA Write

Send

Recv

Interrupts

Connection Management

Buffer Management
TreadMarks over GM: Proposed Substrate

- Pre-posting receive buffers
- Asynchronous requests
  - (n-1) outstanding requests possible for n processes
  - Post (n-1) buffers for sizes 4 (8 bytes) to 15 (32K)
  - Requires 64K*(n-1) per process
- Synchronous requests
  - Single buffer for sizes 4 to 15
  - 64K per process
- Total requirement is 64K*(n-1)+64K
- For 256 nodes 16MB required
- Rendezvous protocol
TreadMarks—Communication primitives and GM

Treadmarks Routines

Send Request/Response
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless

Recv Response
- Contiguous/Noncontiguous
- From any user buffer
- Receiving from any node of a group

Recv Request
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless
- Automatic allocation of temporary buf

SIGIO handler

Connection Management

Connect

RDMA Write

Send

Preposting Receive Buffers

Asynchronous Send

Async Request

Asynchronous Receive

Schemes for Handling Async. Messages

Buffer Management

Send

GM

Recev

Interrupts
TreadMarks over GM: Proposed Substrate

- Schemes for handling asynchronous requests
  - On receiving an asynchronous request from a particular node, don’t reply until a buffer has been pre-posted
TreadMarks-Communication primitives and GM

Treadmarks Routines

Send Request/Response
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless

Recv Request
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless
- Automatic allocation of temporary buffer

Synchronous Send

Synchronous Receive

Asynchronous Send

Asynchronous Receive

Preposting Receive Buffers

Schemes for Handling Async. Messages

Send Request/Response
- Contiguous/Noncontiguous
- From any user buffer
- Connection oriented/Connectionless

Recv Response
- Contiguous/Noncontiguous
- From any user buffer
- Receiving from any node of a group

Connection Management

Buffer Management

GM

RDMA Write

Send

Recv

Interrupts

SIGIO handler
TreadMarks over GM: Proposed Substrate

- Asynchronous Notification
  - Interrupt
    - Requires modification to GM Machine Control Program
    - Best performance
  - Polling Thread
  - Timer
Performance Evaluation

• Our implementation (FAST/GM) compared with original implementation (UDP/GM)
• Test bed
  - 16 machines with Quad 700 MHz Pentium III, 1Gb main memory connected by a 2.1 Gbps Myrinet network running GM 1.5.2.1. Myrinet NIC is a LanAI 9 with 4MB memory and a 134MHz CPU
• Evaluation carried out using
  - Microbenchmarks; measure latency of basic operations like page, diff, barrier and lock
  - Applications (Sor, Jacobi, Tsp and 3Dfft)
    • Effect of increase in system size on scale measured
    • Effect of increase in application size on scale measured
Performance Evaluation - Microbenchmarks

- Order of magnitude decrease in time to fetch a page, diff and lock
Performance Evaluation - System Size

- 3Dfft, Tsp, for UDP/GM execution time increases, but decreases for FAST/GM
- For Sor execution time much lower in the case of FAST/GM
• Wait time are significantly reduced
•Recv and send times reduced
•Signal handler time reduced thanks to tighter integration with the communication layer
Performance Evaluation - Scaling with Application Size

- Significant decrease in execution time for 3Dfft, Sor and tsp
Conclusions

• Designed and developed a new framework where software DSM systems like TreadMarks can exploit low latency, high bandwidth networks like Myrinet
• Performance evaluated in terms of
  - Microbenchmarks
    • Cost of basic software DSM operations significantly reduced by order of magnitude
  - System Size
    • Speedup upto a maximum of 6.3 for FAST/GM
  - Application Size
    • Execution time improved by a maximum factor of 5.5 for FAST/GM over UDP/GM
Future Work

- Scaling to a large number of nodes
  - NIC based implementations-barrier, caching
  - Communication optimizations
    - Diff processing constitutes a significant overhead
      - Possible to eliminate diff processing
  - Ported HLRC (Rutgers) to InfiniBand
    - Barrier takes a significant percentage of execution
    - Reduce overhead through multicast
- New protocols and challenges
  - Eager protocols would have less overhead on a network like InfiniBand
Additional Information

- More information about this paper and other work can be found at:

  NBC Home Page
  http://nowlab.cis.ohio-state.edu
  Network Based Computing Group
  The Ohio State University

- By e-mail
  Prof. D.K. Panda - panda@cis.ohio-state.edu
  Ranjit Noronha - noronha@cis.ohio-state.edu