Optimized Distributed Data Sharing Substrate in Multi-Core Commodity Clusters: A Comprehensive Study with Applications

K. Vaidyanathan, P. Lai, S. Narravula and D. K. Panda

Network Based Computing Laboratory (NBCL)
The Ohio State University
Presentation Outline

• Introduction and Motivation
• Distributed Data Sharing Substrate
• Proposed Design Optimizations
• Experimental Results
• Conclusions and Future Work
Introduction and Motivation

- Interactive data-driven applications
  - Stock trading, airline tickets, medical imaging, online auction, online banking, web streaming, ...
  - Ability to interact, synthesize and visualize data
- Datacenters enable such capabilities
  - Processes data and reply to client queries
  - Common and increasing in size (IBM, Amazon, Google)
- Datacenters unable to meet increasing client demands
Datacenter Architecture

- Applications host web content online
- Services improve performance and scalability
- State sharing is common in applications and services
  - Communicate and synchronize (intra-node, intra-tie r and inter-tier)
State Sharing in Datacenters

Intra-Node State Sharing

Tier 0
- Proxy Server
- Resource monitoring
- Load balancing
- Resource adaptation
- Resource adaptation

Tier 1
- Application Server
- Apache
- Caching
- STORM A
- STORM B
- Servlets
- App
- Apache
- Res Mgmt

Inter-Tier State Sharing

Resource adaptation
Load balancing
Tier 1 load
Tier 1 load
Tier 1 load
Tier 1 load
Tier 1 load

Resource monitoring
Resource monitoring
Resource monitoring
Resource monitoring
Resource monitoring
Resource monitoring

Network-based Computing Laboratory
State Sharing in Datacenters...

- Several applications employ their own
  - data management protocols
  - maintain versions of stored data
  - synchronization primitives

- Datacenter Services frequently exchange
  - System load, system state, locks
  - Cached data

Issues

- Ad-hoc messaging protocols for exchanging data/resource
- Same data/resource at multiple places (e.g., load information, data)
- Protocols used are typically TCP/IP, IPC mechanisms, memory copies, etc
- Performance may depend on the back-end load
- Scalability issues
High-Performance Networks

- InfiniBand, 10 Gigabit Ethernet

- High-Performance
  - Low latency (< 1 usecs) and high bandwidth (> 32 Gbps with QDR adapters)

- Novel features
  - One-sided RDMA and atomics, multicast, QoS

- OpenFabrics alliance (http://www.openfabrics.org/)
  - Common stack for several networks including iWARP (LAN/WAN)
Datacenter Research at OSU

Existing Datacenter Components

- Active Resource Adaptation
- Dynamic Content Caching
- Soft Shared State
- Lock Manager
- Global Memory Aggregator

Advanced Resource Adaptation Services

- Active Resource Adaptation
- Dynamic Content Caching

Advanced Communication Protocols and Subsystems

- Sockets Direct Protocol
- RDMA
- Atomics
- Multicast

High-Speed Networks

- High-Performance Networks (InfiniBand, iWARP 10GigE)

Datacenter Homepage: http://nowlab.cse.ohio-state.edu/projects/data-centers/
Distributed Data Sharing Substrate

Load Info
System State
Meta-data
Data

Datacenter Application
Get
Get
Get
Put
Put
Put
Put

Datacenter Application

Datacenter Application

Datacenter Services

Datacenter Services
Multicore Architectures

- **Increased cores per-chip**
  - More parallelism available
- **Intel, AMD**
  - Dual-core, quad-core
  - 80-core systems are currently built
- **Significant benefits for datacenters**
  - Applications are multi-threaded in nature
  - Design Optimizations in state sharing mechanisms
  - Opportunities for dedicating one or more cores
Objective

• Can we enhance the distributed data sharing substrate using the features of multicore architectures by dedicating one or more of the cores?

• How do these enhancements help in improving the overall performance with datacenter applications and services?
Presentation Outline

• Introduction and Motivation

• Distributed Data Sharing Substrate

• Proposed Design Optimizations

• Experimental Results

• Conclusions and Future Work
Distributed Data Sharing Substrate

• Use of a common service thread to get access to the shared state
• Applications get shared state information using the service thread
• Several design optimizations in communicating with the service thread
  – Message Queues (MQ-DDSS)
  – Memory mapped queues for request (RMQ-DDSS)
  – Memory mapped queues for request and response (RCQ-DDSS)
Message Queue-based DDSS (MQ-DDSS)
Message Queue-based DDSS

- **Kernel involvement**
  - IPC Send and Receive operations
  - Communication Progress

- **Limitations**
  - Several context-switches
  - Interrupt overheads
Presentation Outline

• Introduction and Motivation

• Distributed Data Sharing Substrate

• Proposed Design Optimizations

• Experimental Results

• Conclusions and Future Work
Request/Message Queue-based DDSS (RMQ-DDSS)

Application Threads
Request Queue
Service Thread
Completion Queue
NIC
Kernel Message Queues
IPC_Recv
IPC_Send
Kernel Involvement
User Space
Kernel Space
Request/Completion Queue-based DDSS (RCQ-DDSS)
RMQ-DDSS and RCQ-DDSS Schemes

- **RMQ-DDSS scheme**
  + Lesser number of interrupts and context-switches compared to MQ-DDSS
  + Improvement in response time as request is sent via memory mapped queues
    - May occupy significant CPU

- **RCQ-DDSS scheme**
  + Avoids kernel involvement
  + Significant improvement in response time as request and response are sent via memory mapped queues
    - May occupy more CPU as compared to RMQ-DDSS - apps & service thread need to poll on the completion queue
Presentation Outline

- Introduction and Motivation
- Distributed Data Sharing Substrate
- Proposed Design Optimizations
- Experimental Results
- Conclusions and Future Work
Experimental Testbed

- InfiniBand experiments
  - 560-core cluster consisting of 70 compute nodes with dual 2.33 GHz Intel Xeon quad-core processors
  - Mellanox MT25208 dual port HCA

- 10-Gigabit experiments
  - Intel dual quad-core Xeon 3.0 GHz, 512 MB memory
  - Chelsio T3B 10 GigE PCI-Express adapters

- OpenFabrics stack
  - OFED 1.2

- Experimental outline
  - Microbenchmarks (performance and scalability)
  - Application performance (R-Trees, B-Trees, STORM, checkpointing)
  - Dedicating cores for datacenter services (resource monitoring)
Basic Performance of DDSS

- RCQ-DDSS scales with increasing client threads
- RCQ-DDSS performs better than RMQ-DDSS and MQ-DDSS
• Hybrid approach is required for scalability with large number of threads
• DDSS scales when keys are distributed
Performance with R-Trees, B-Trees, STORM

- MQ-SS shows significant improvement compared to traditional implementations but RCQ-SS shows marginal improvements compared to MQ-SS
Data Sharing Performance in Applications

- RCQ-DDSS shows significant improvement as compared to RMQ-DDSS and MQ-DDSS
Performance with checkpointing

- Hybrid approach is required for scalability with large number of threads
Performance with Dedicated Cores

- Dedicating a core for resource monitoring can avoid up to 50% degradation in client response time
Conclusions & Future Work

• Proposed multicore optimizations for distributed data sharing substrate
• Evaluations with several applications shows significant improvement
• Showed the benefits of dedicating cores for services in datacenters
• Future work on dedicating other datacenter services, datacenter-specific operations
Web Pointers

Datacenter Homepage: http://nowlab.cse.ohio-state.edu/projects/data-centers/

Emails: {vaidyana, laipi, narravul, panda}@cse.ohio-state.edu