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

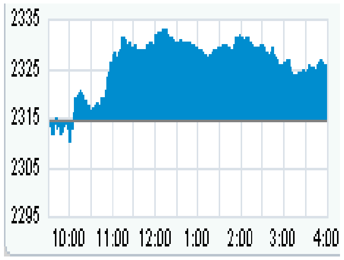
Network Based Computing Laboratory (NBCL)

The Ohio State University

Presentation Outline

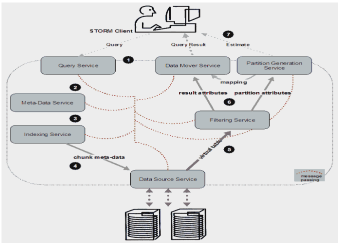
- **Introduction and Motivation**
- Distributed Data Sharing Substrate
- Proposed Design Optimizations
- Experimental Results
- Conclusions and Future Work

Introduction and Motivation



Stock markets

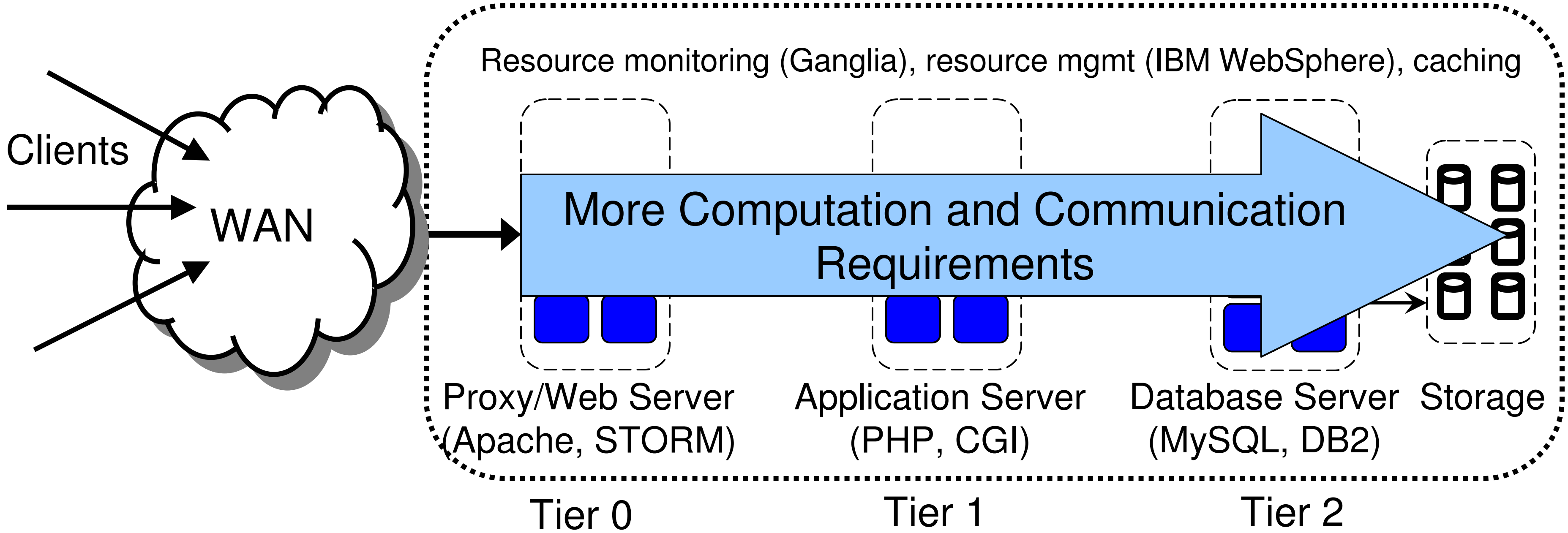
Airline industries


Medical imaging

Online auction

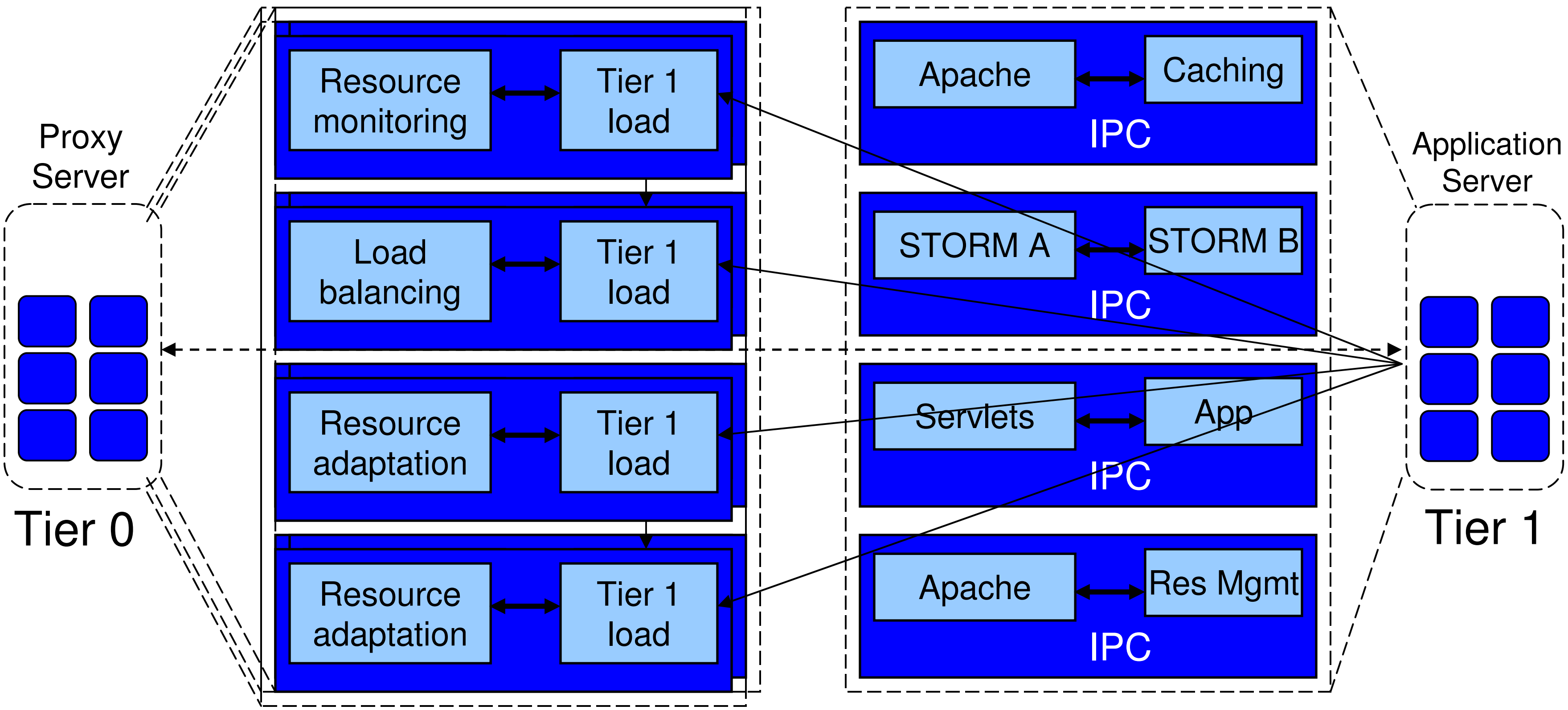
- Interactive data-driven applications
  - Stock trading, airline tickets, medical imaging, online auction, online banking, web streaming, …
  - Ability to interact, synthesize and visualize data
- Datacenters enable such capabilities
  - Processes data and reply to client queries
  - Common and increasing 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-tier and inter-tier)

State Sharing in Datacenters



Inter-Tier State Sharing
Intra-Node State Sharing

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

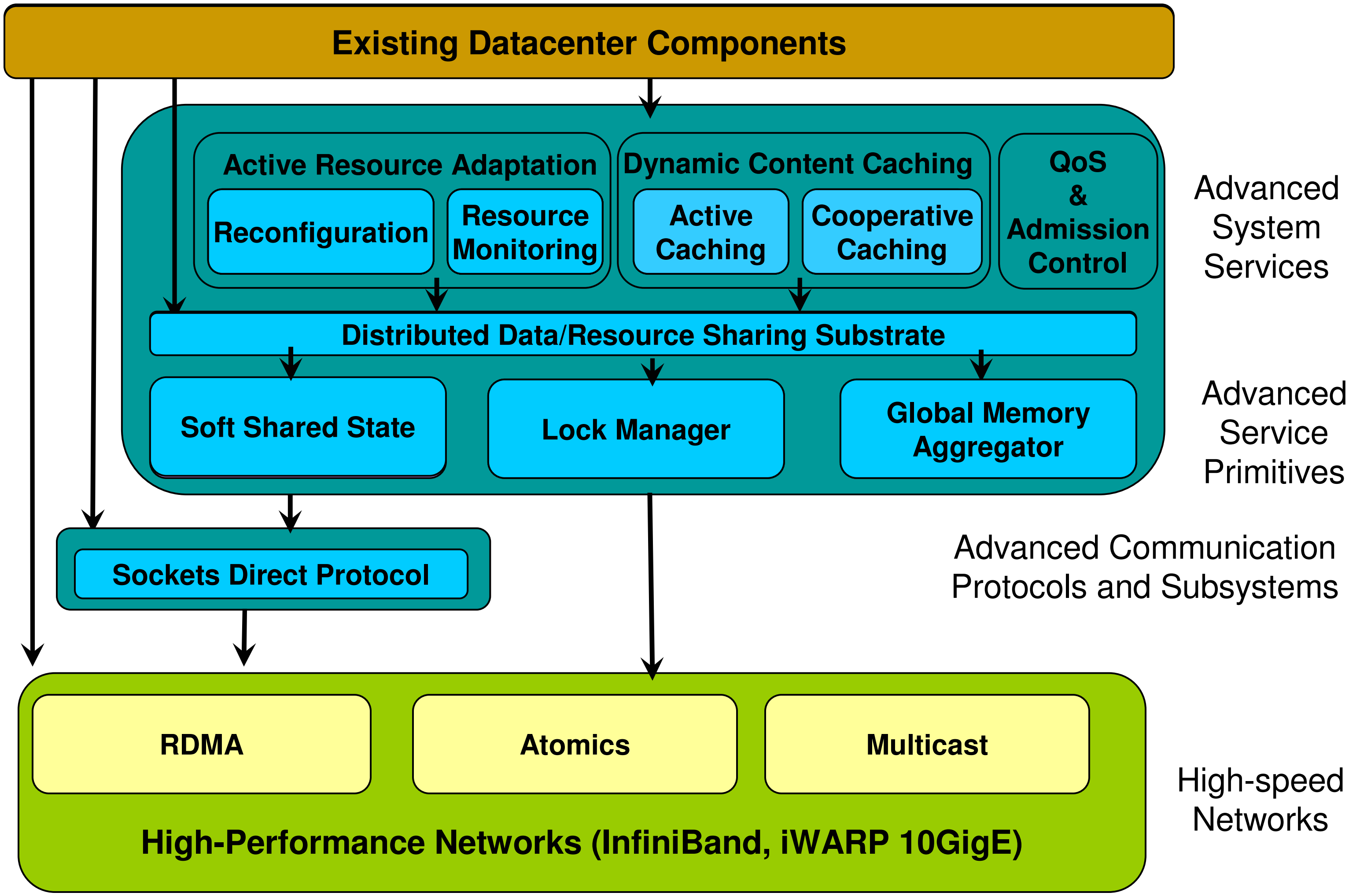
Issues

- 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

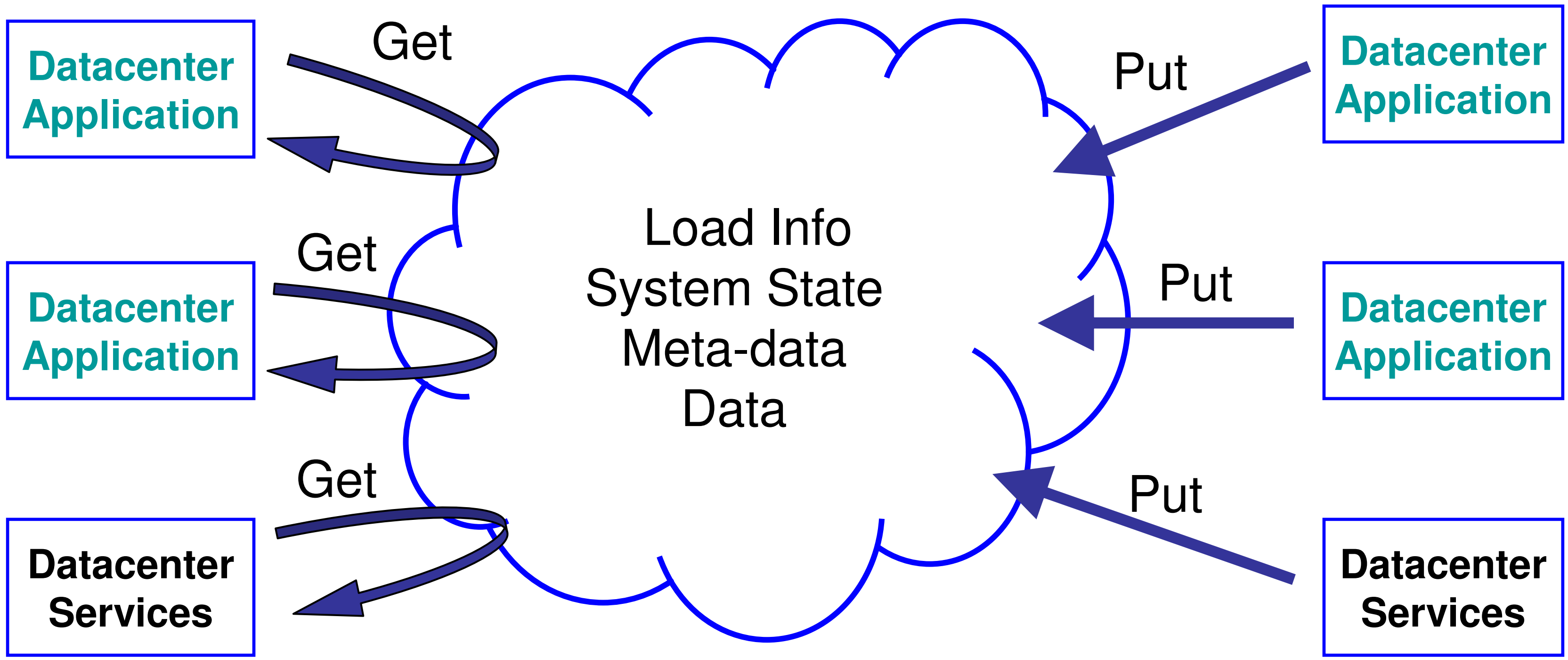
- InfiniBand, 10 Gigabit Ethernet
- High-Performance
  - Low latency (< 1 usecs) and high bandwidth (> 32 Gbps with QDR adapters)
- Novel features
  - One-sided RDMA and atomics, multicast, QoS
- OpenFabrics alliance (http://www.openfabrics.org/)
  - Common stack for several networks including iWARP (LAN/WAN)

Datacenter Research at OSU



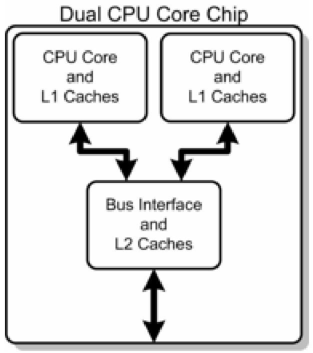
Datacenter Homepage: http://nowlab.cse.ohio-state.edu/projects/data-centers/

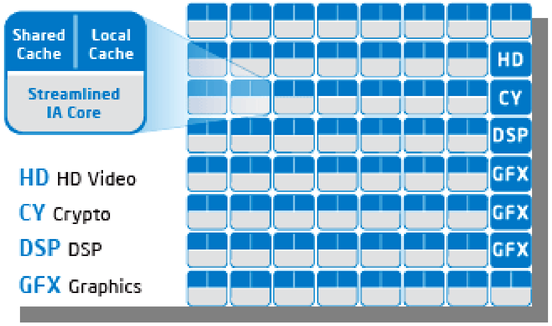
Distributed Data Sharing Substrate



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





Future multicore systems

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?

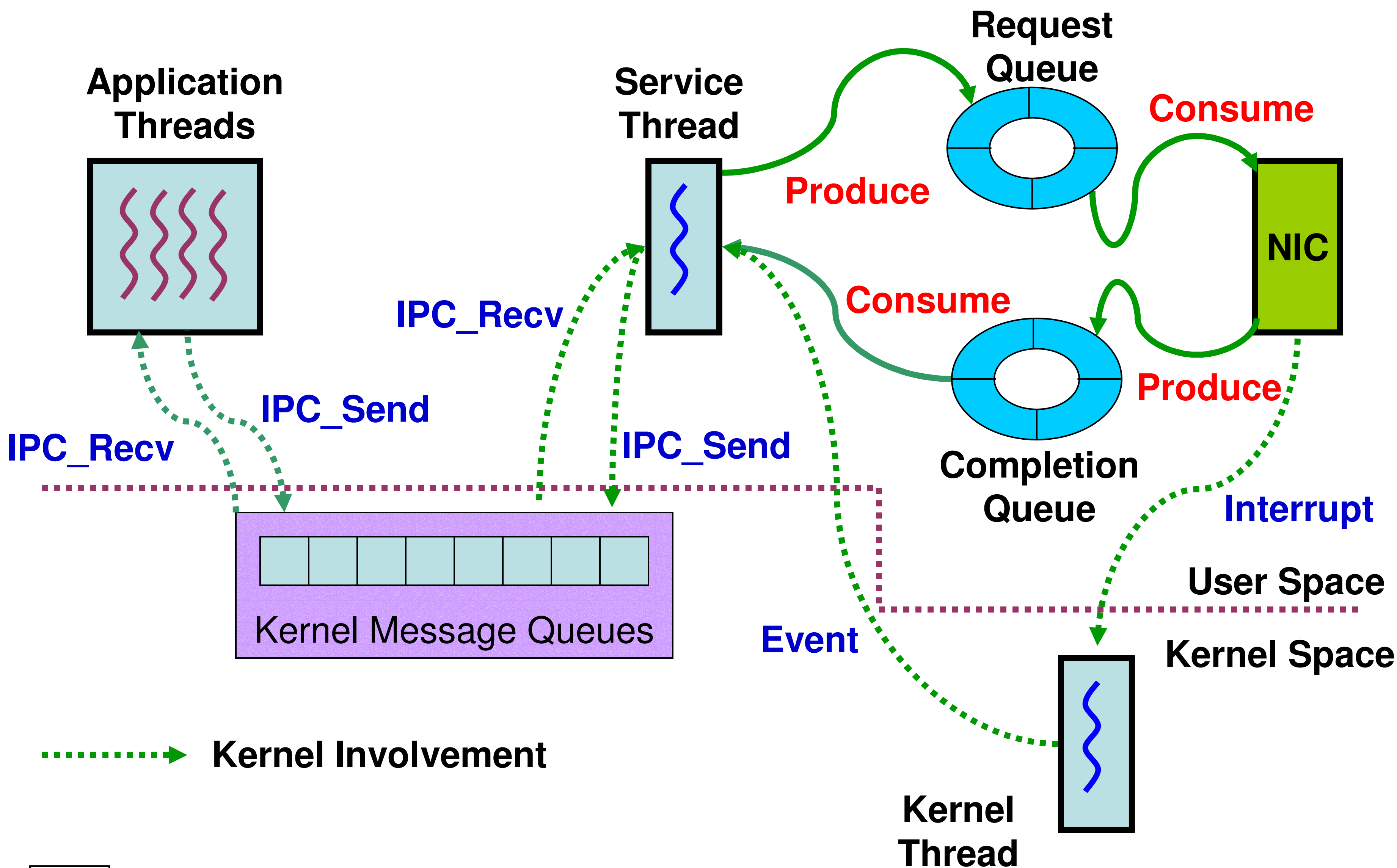
Presentation Outline

- 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)



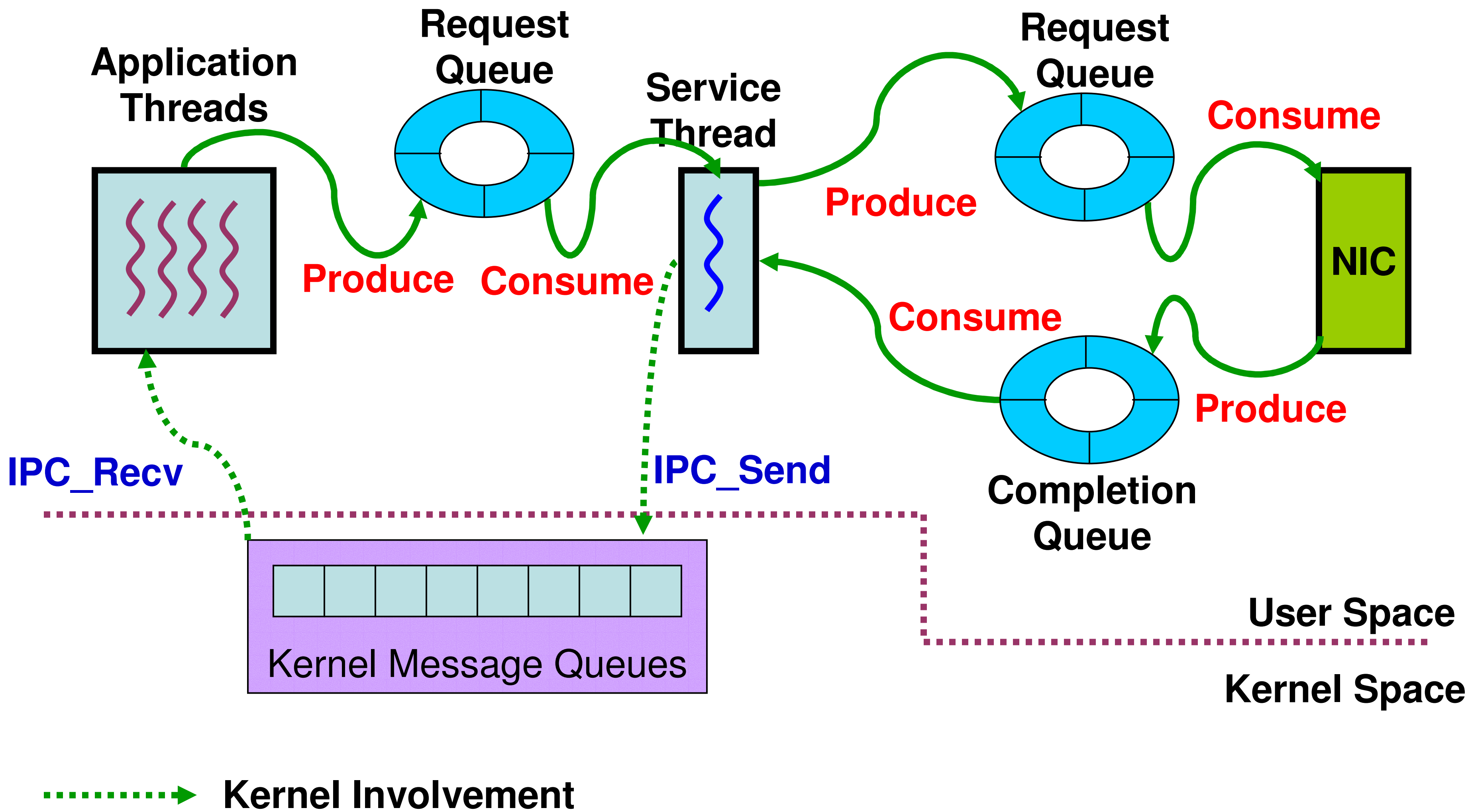
Message Queue-based DDSS

- Kernel involvement
  - IPC Send and Receive operations
  - Communication Progress
- Limitations
  - Several context-switches
  - Interrupt overheads

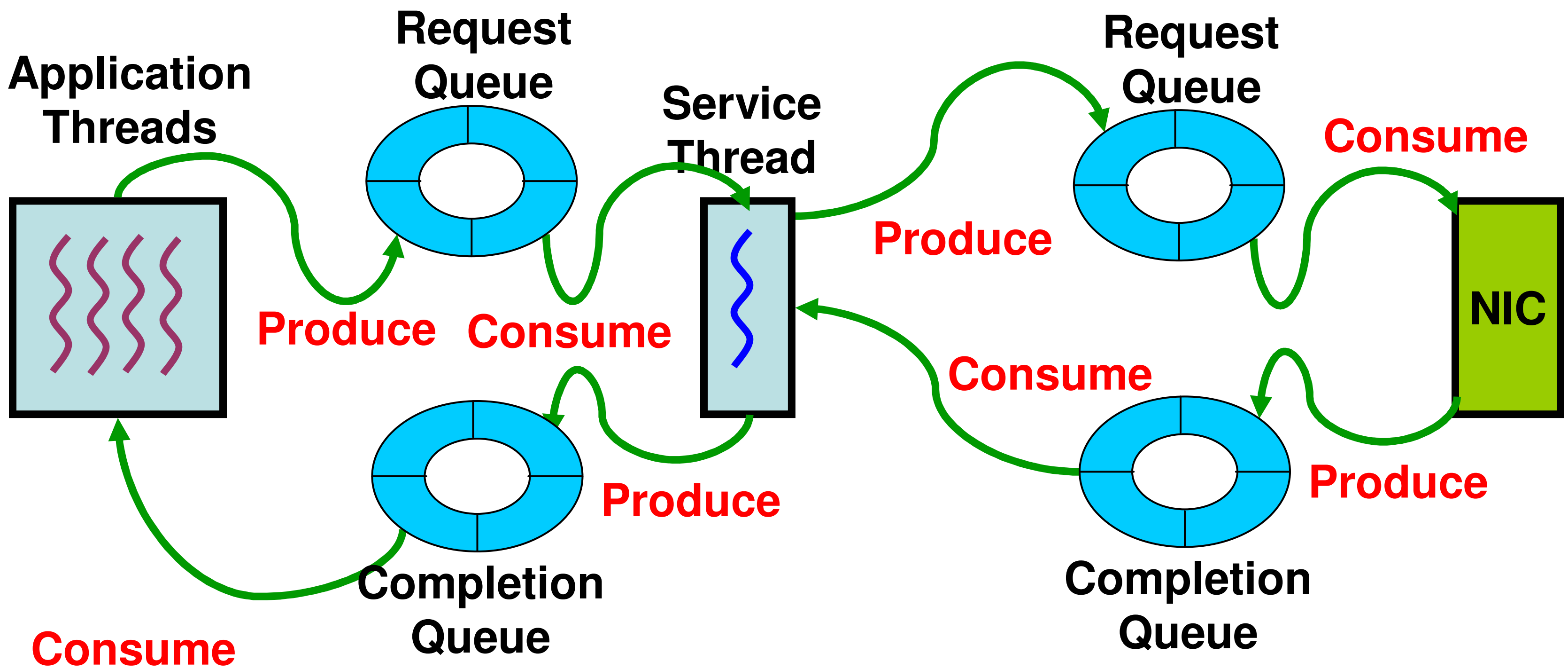
Presentation Outline

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

Request/Message Queue-based DDSS (RMQ-DDSS)



Request/Completion Queue-based DDSS (RCQ-DDSS)



User Space

Kernel Space

No Kernel Involvement

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

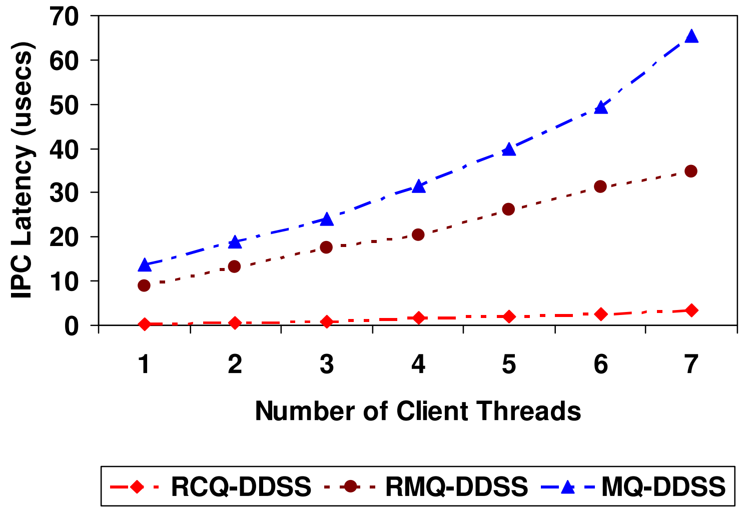
Presentation Outline

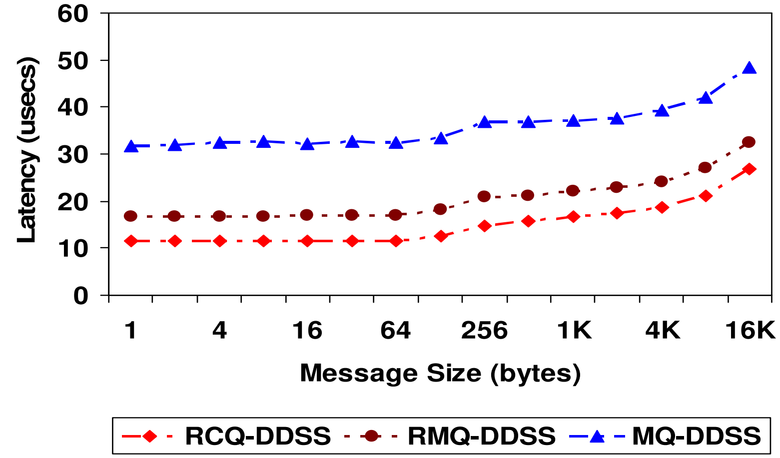
- Introduction and Motivation
- Distributed Data Sharing Substrate
- Proposed Design Optimizations
- **Experimental Results**
- Conclusions and Future Work

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)

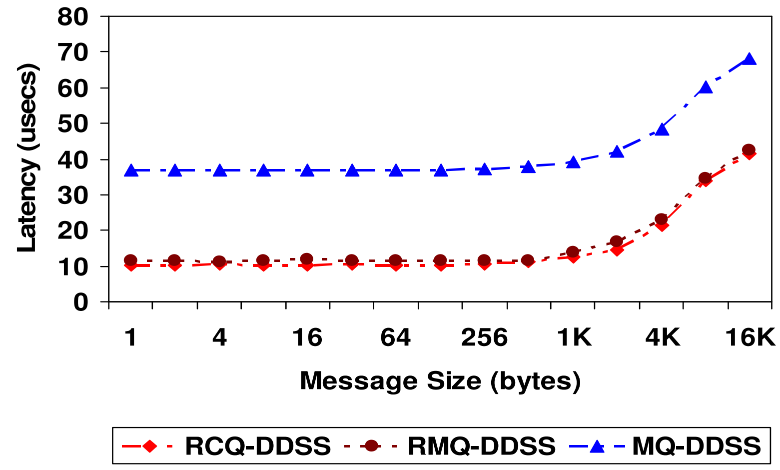
Basic Performance of DDSS





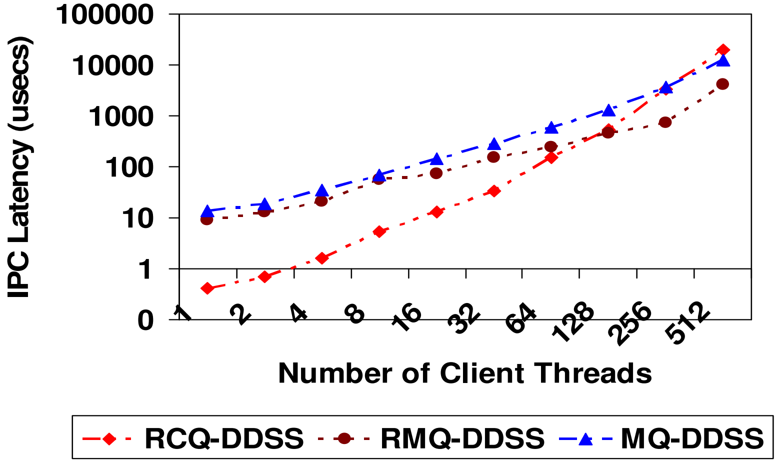
InfiniBand

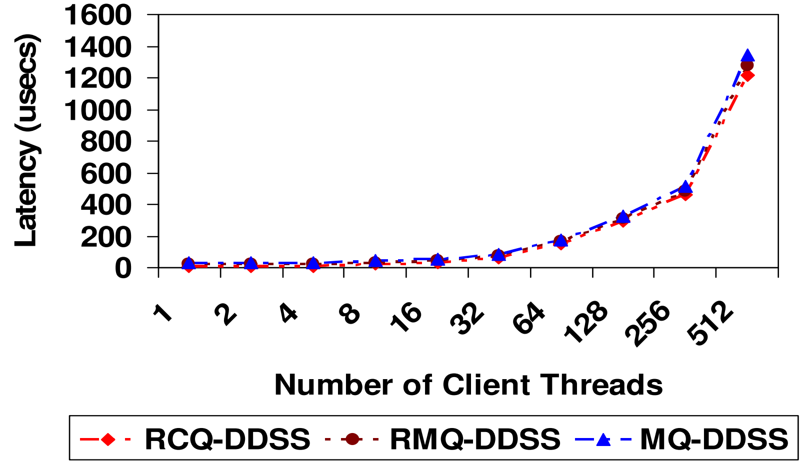
- RCQ-DDSS scales with increasing client threads
- RCQ-DDSS performs better than RMQ-DDSS and MQ-DDSS



10-Gigabit Ethernet

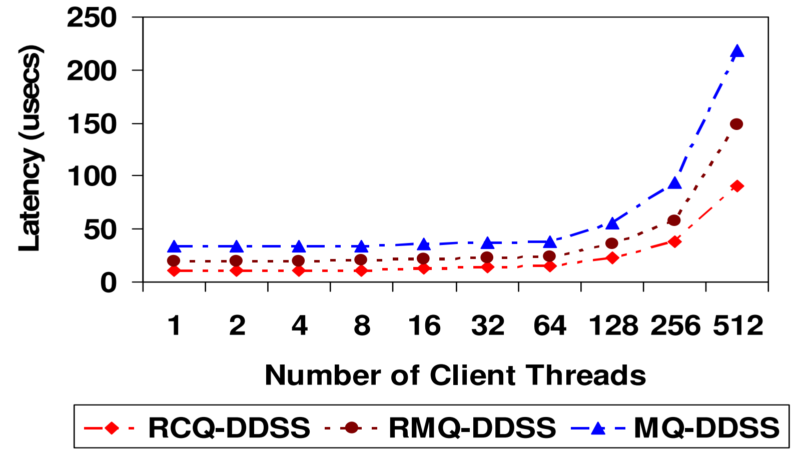
DDSS Scalability





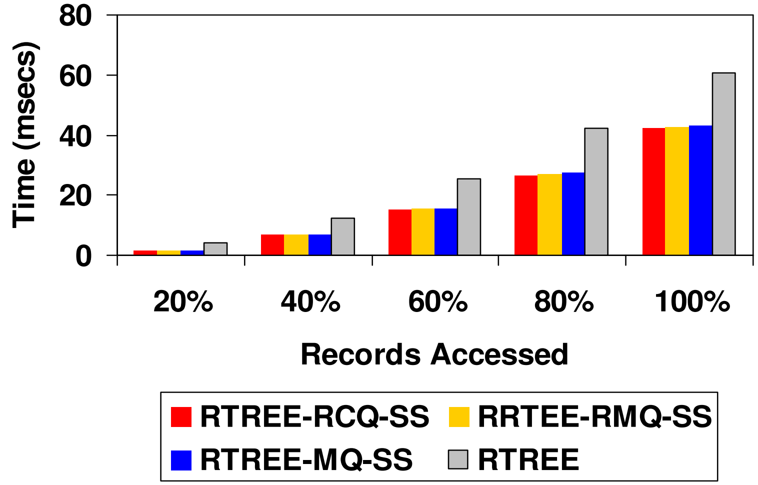
Keys are on a single node

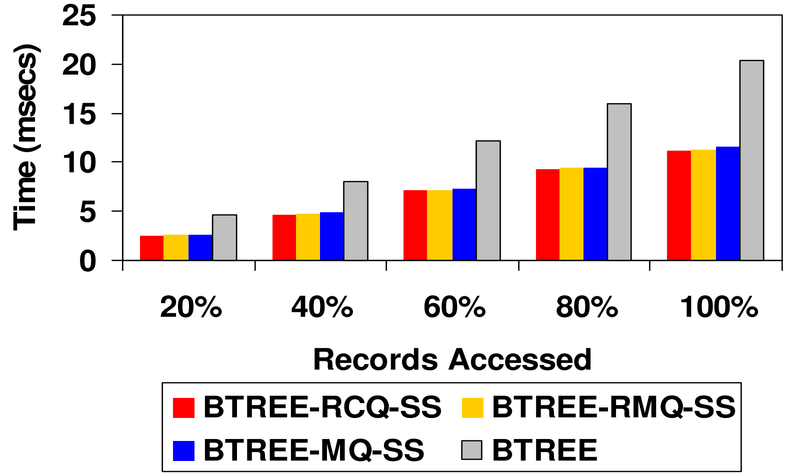
- Hybrid approach is required for scalability with large number of threads
- DDSS scales when keys are distributed



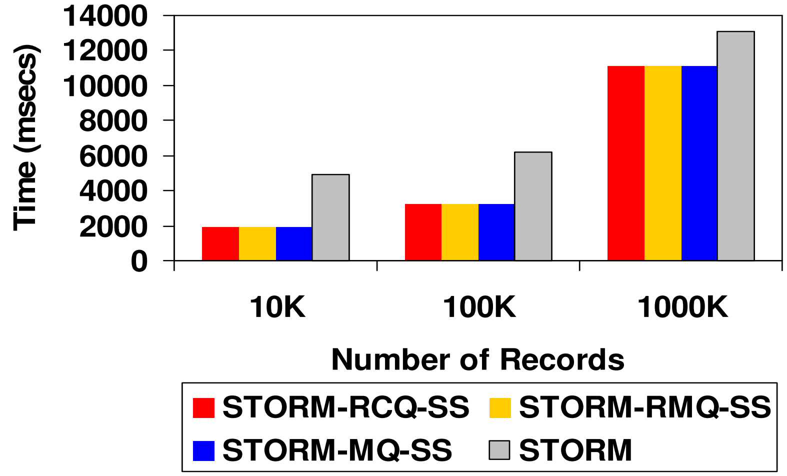
Keys are distributed

Performance with R-Trees, B-Trees, STORM

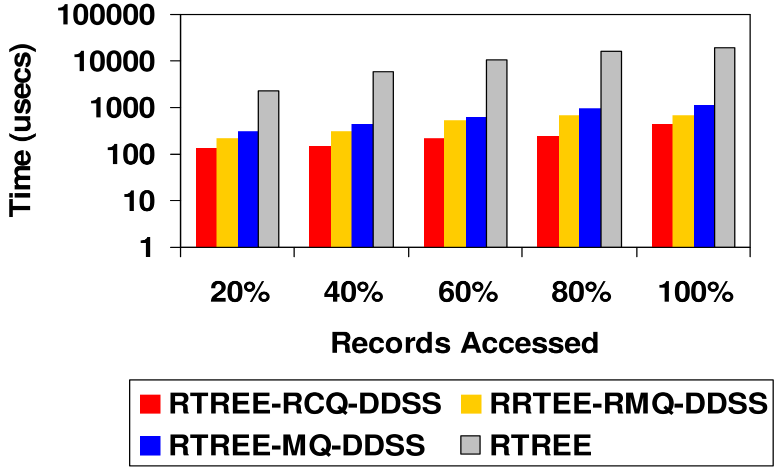


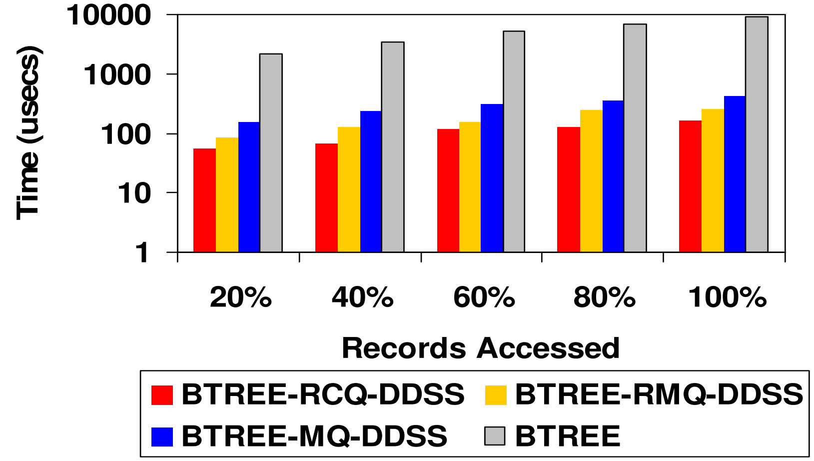


- MQ-SS shows significant improvement compared to traditional implementations but RCQ-SS shows marginal improvements compared to MQ-SS

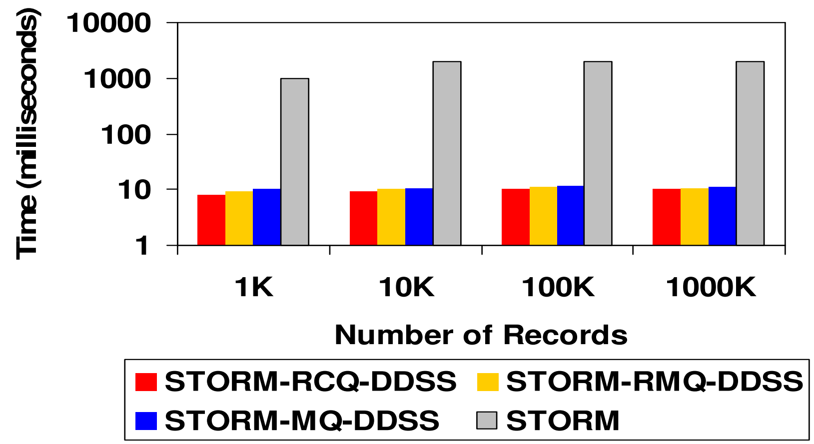


Data Sharing Performance in Applications

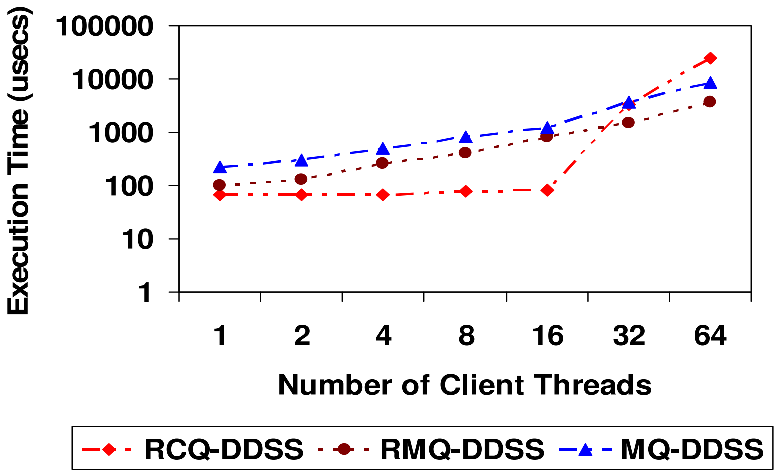




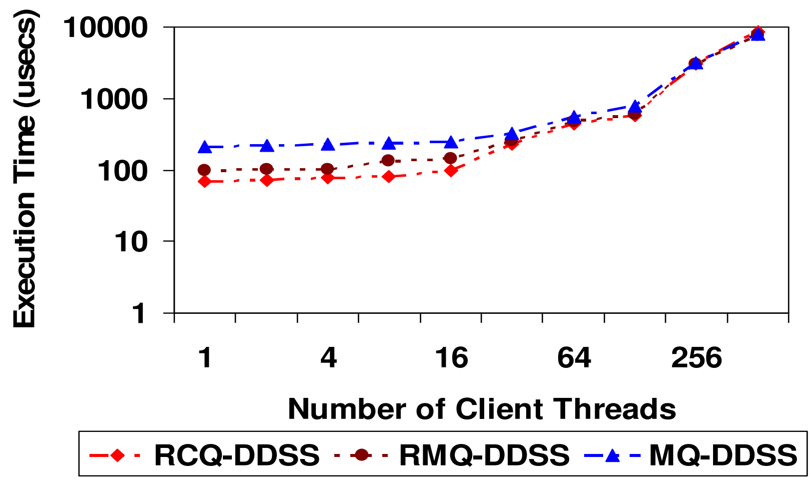
- RCQ-DDSS shows significant improvement as compared to RMQ-DDSS and MQ-DDSS



Performance with checkpointing

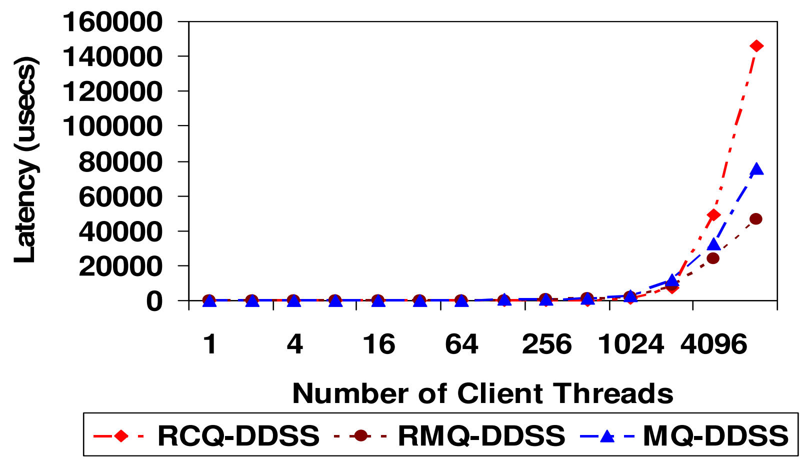


Clients on single node (non-distributed)



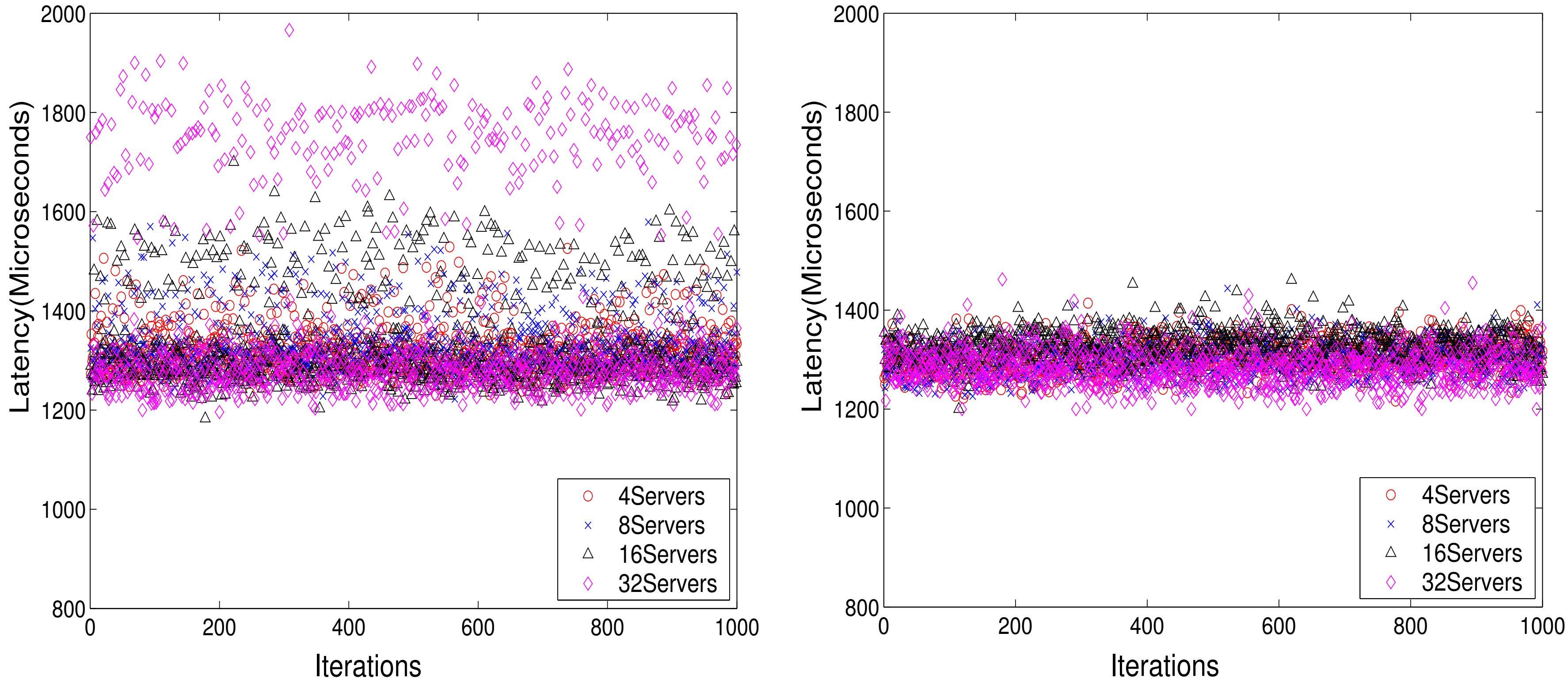
Clients on diff node (non-distributed)

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



Clients on diff node (non-distributed)

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

NBC-LAB

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

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