DDSS: A Low-Overhead Distributed Data Sharing Substrate for Cluster-Based Data-Centers over Modern Interconnects

K. Vaidyanathan, **S. Narravula** and D. K. Panda

Network Based Computing Laboratory (NBCL)

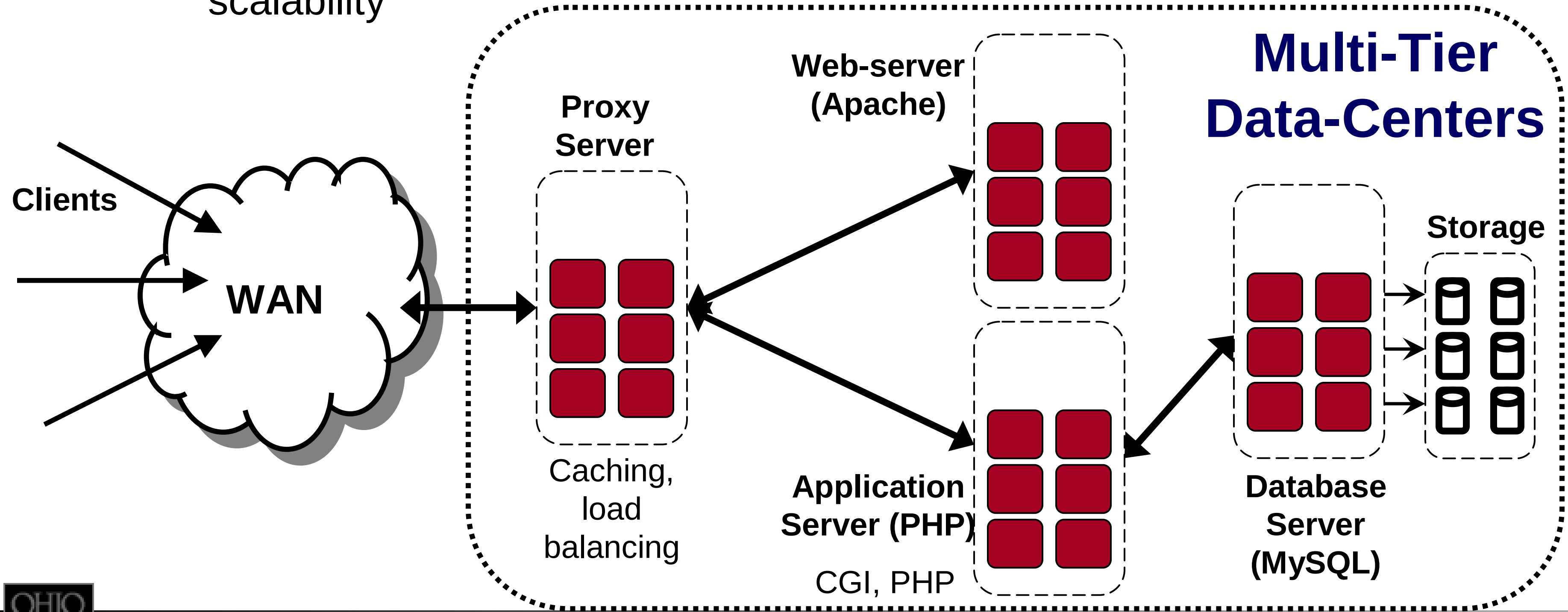
The Ohio State University

Presentation Outline

- **Introduction and Motivation**
- Proposed DDSS Framework
- Experimental Results
- Conclusions and Future Work

Introduction and Motivation

- Internet growth
  - Number of Users, Type of Service, Amount of data
  - E-Commerce, online-banking, stocks, airline reservations
- Data-centers enable such services
  - Process data and reply to queries
  - Need for services like caching, resource adaptation for performance, scalability

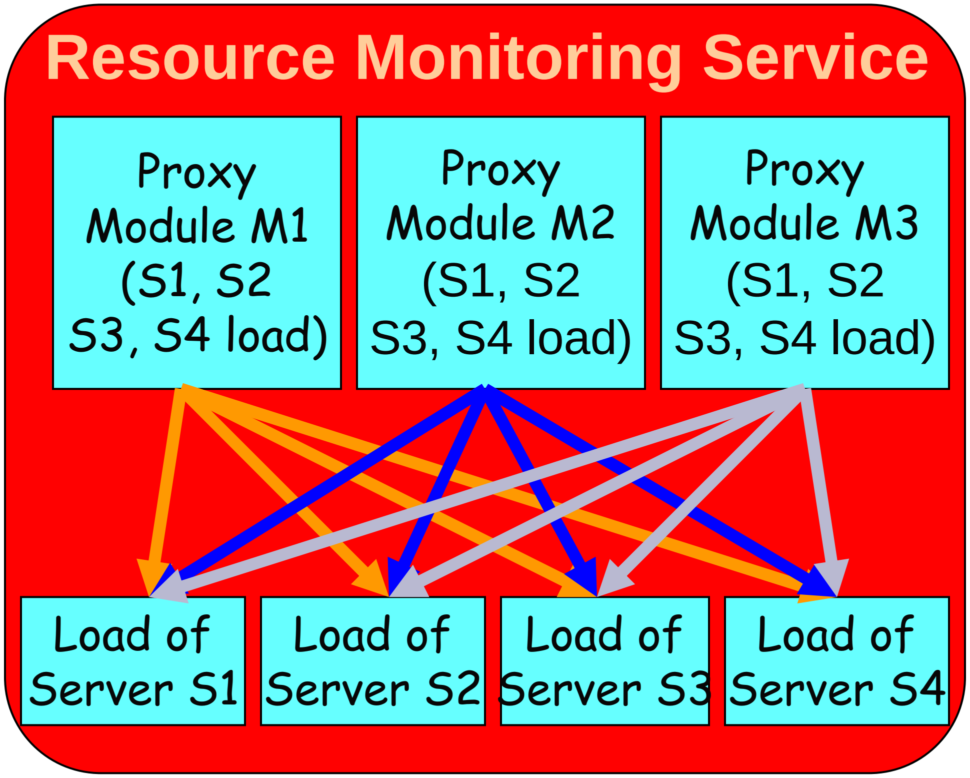


High-Performance Networks

- InfiniBand, 10 GigE
  - High Bandwidth
  - Low Latency
- Provides rich features
  - RDMA semantics, Atomic operations, Protocol offload
- OpenFabrics stack
  - Single interface for InfiniBand, iWARP/10 GigE, etc
- Targeted for Multi-Tier Data-Centers
- Can the data-center processes coordinate better?

Information-Sharing is common

- Applications typically employ their own
  - Data placement and management protocols
  - Synchronization protocols
- Data-Center services
  - Active Resource Adaptation
    - Maintain Server state information
    - Locking requirements
  - Caching
    - Coherency & Consistency requirements
  - Resource Monitoring (IBM Websphere)
    - Load information shared across several servers
  - Critical decisions based on shared information



Problems with Existing approaches

- Ad-hoc messaging protocols for exchanging data
- May have high overheads
- Performance may depend on the system load
- May not use the advanced features
- May not be scalable

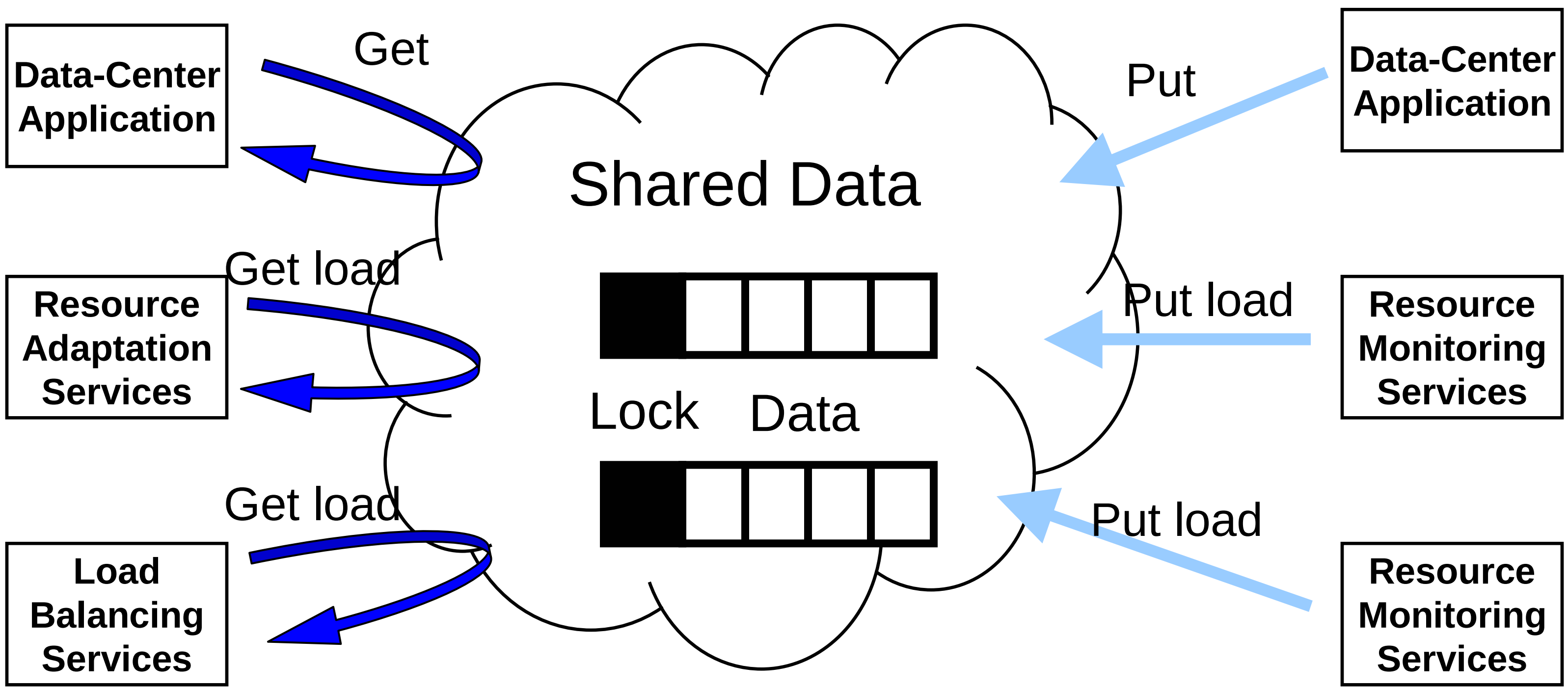
Objective

- Can we design a load resilient substrate (DDSS) for data-center applications and services utilizing advanced features such as RDMA, remote atomic operations?

Presentation Outline

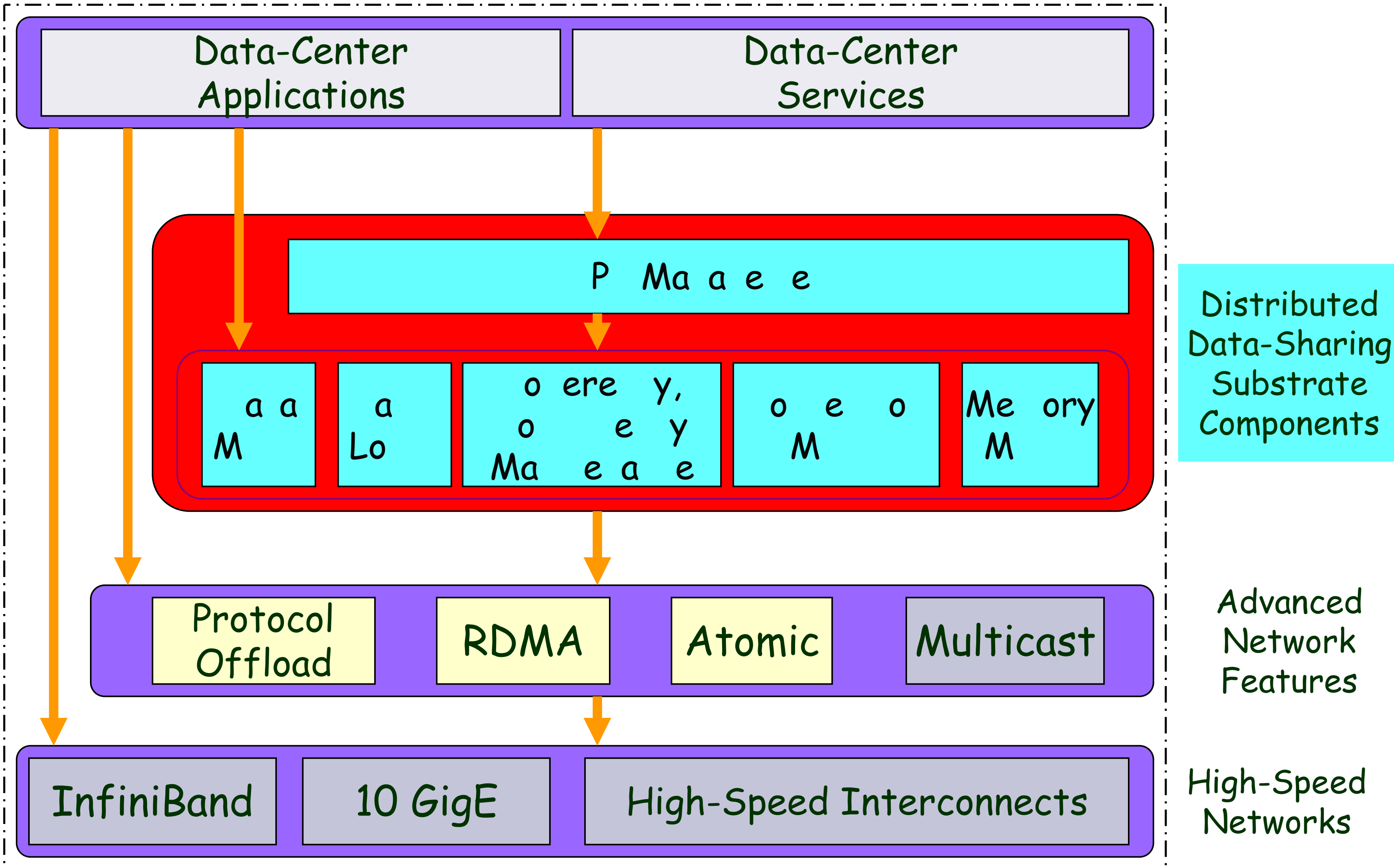
- Introduction and Motivation
- **Proposed DDSS Framework**
- Experimental Results
- Conclusions and Future Work

Distributed Data Sharing Mechanism



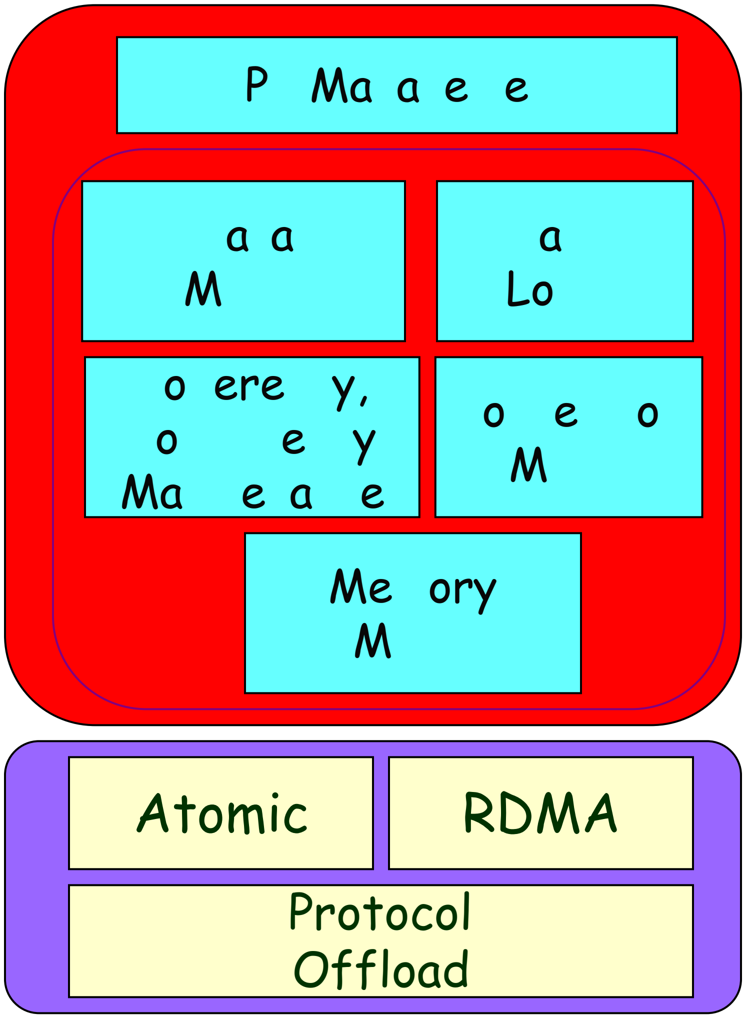
Provide an effective mechanism to share data across the data-center

Proposed DDSS Framework



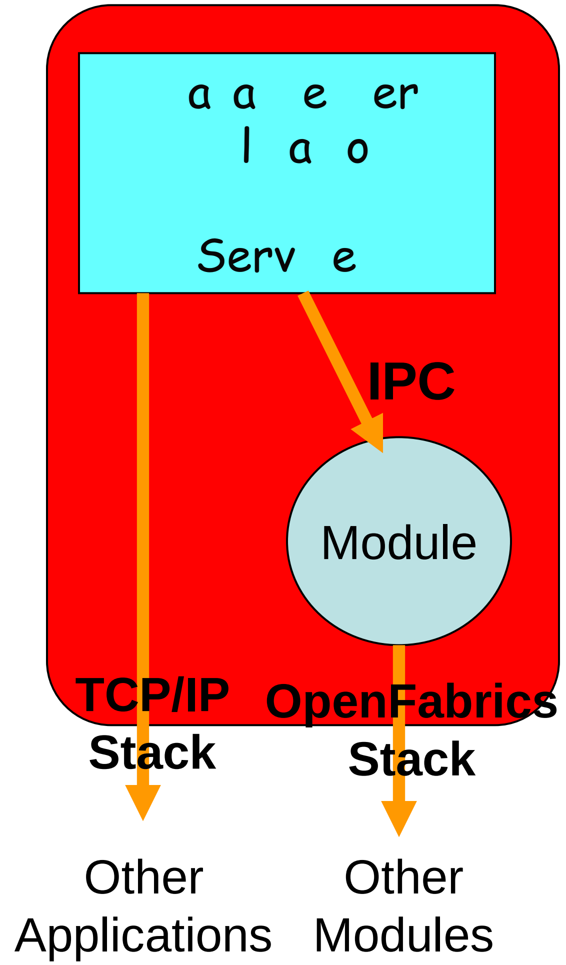
Proposed Framework Contd…

- Data Management
  - Local vs Remote, for load balancing
- Basic Locking
  - Through atomic operations (IBA)
- Coherency and Consistency Maintenance
  - Strict, Write/Read, Null, Delta, Version
  - Use of RDMA and atomic operations



Proposed Framework Contd…

- Connection Management
  - Takes care of connection-setup and teardown for nodes participating in DDSS
- Memory Management
  - Allocates a pool of memory for DDSS on each node
  - Manages allocation, release operations
- IPC Management
  - Access for multiple threads
  - Message Queues



DDSS Interface

DDSS Interface

- *allocate_ss(…)*
- *release_ss(…)*
- *get(…)*
- *put(…)*
- *acquire_lock_ss(…)*
- *release_lock_ss(…)*
- *…*

```
Key = allocate_ss(1024,
    NONCOHERENT_SS, 5000);
put(key, data, 10);
compute();
get(key,data, 10);
release_ss(key);
```

```
Key = allocate_ss(1024,
    WRITE_COHERENT_SS, 5000);
  acquire_lock_ss(key);
put(key, data, 10);
  release_lock_ss(key);
compute();
get(key,data, 10);
release_ss(key);
```

Presentation Outline

- Introduction and Motivation
- Proposed Framework
- Experimental Results
- Conclusions and Future Work

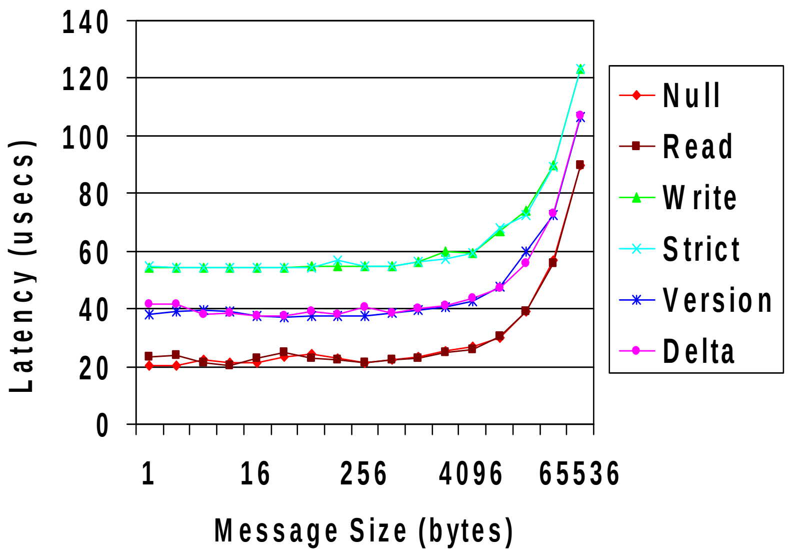
Experimental Testbed

- InfiniBand
  - Cluster with dual Intel Xeon 3.4 GHz, 1GB memory
  - MT25128 Mellanox HCA
- iWARP/GigE
  - Cluster with Intel dual Xeon 3.0 GHz, 512 MB memory
  - Ammasso 1100 Gigabit Ethernet NIC
- OpenFabrics stack
  - IB, Ammasso (iWARP)

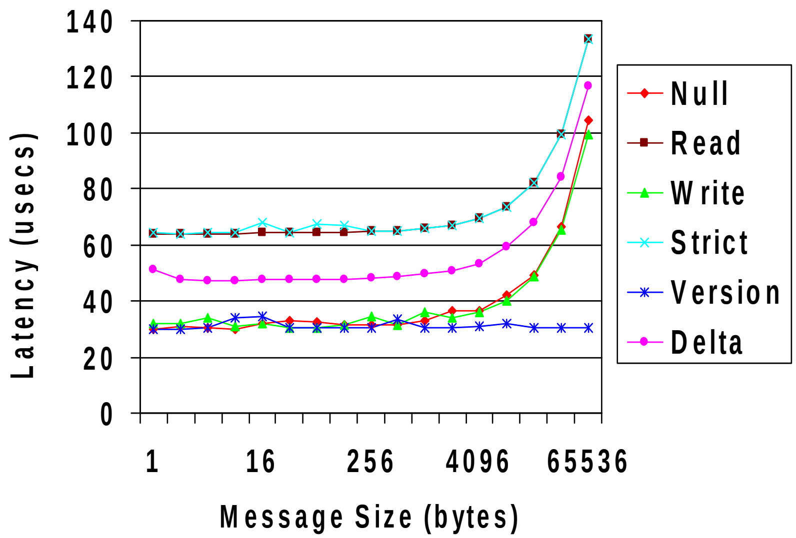
Experimental Results Outline

- Microbenchmarks
  – Performance of put() and get() operations
- Distributed Applications
  – Distributed STORM
  – Checkpointing Application
- Data-Center Services
  – Active Resource Adaptation
  – Active Caching

Microbenchmarks



put() performance



get() performance

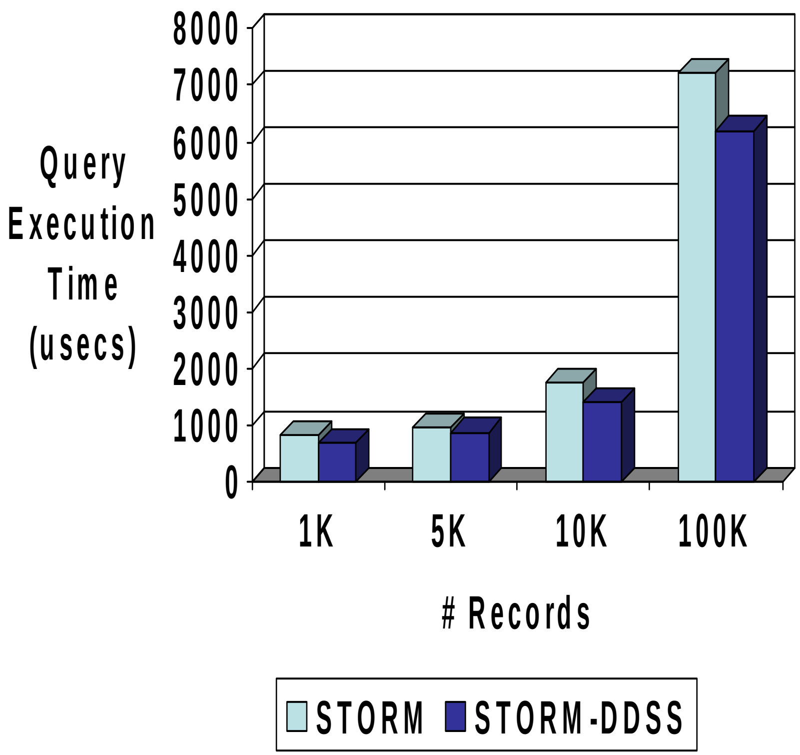
- performance of put() and get() operation for small messages is less than 65 usecs for all coherence models

Distributed STORM

- Select data of interest and transfer from storage to compute nodes
- Same dataset is processed by multiple STORM applications

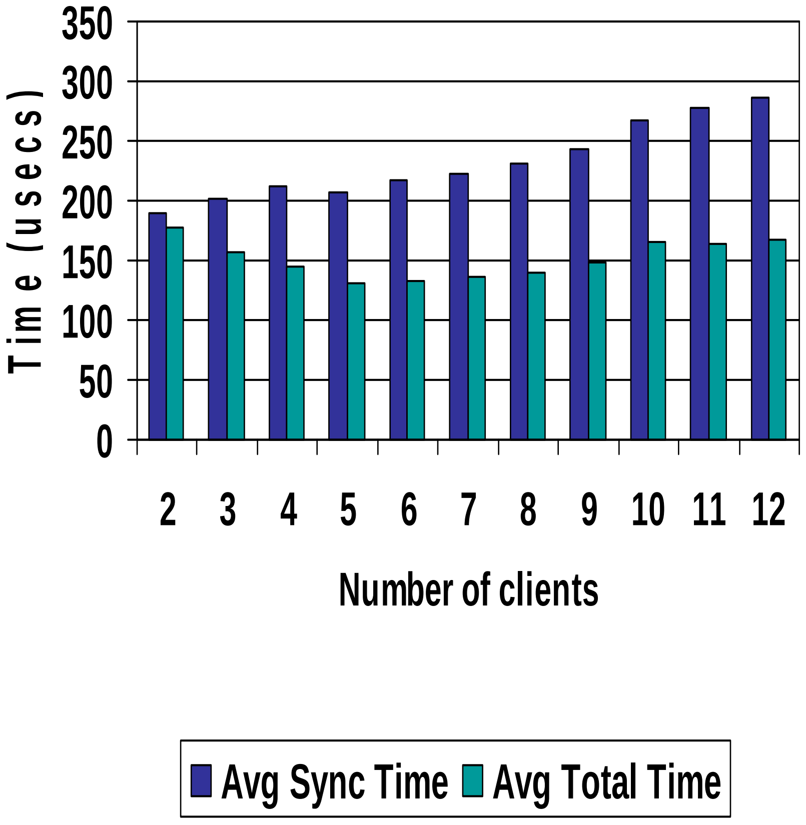
→ this shared dataset is placed in DDSS

- STORM using DDSS shows close to **19% improvement**



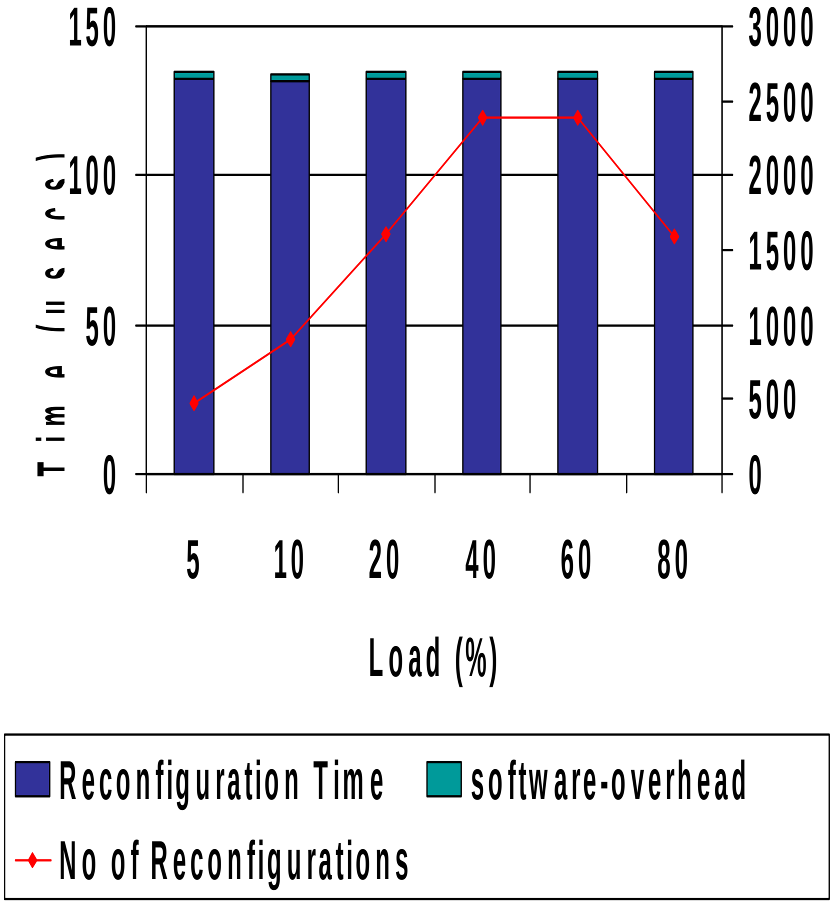
CR Coordination

- Checkpoint at random time
- Simulates restart from a consistent checkpoint
- Checkpoint uses DDSS for maintaining checkpoint information, locks, versions, etc
- Check-pointing applications using DDSS are highly scalable



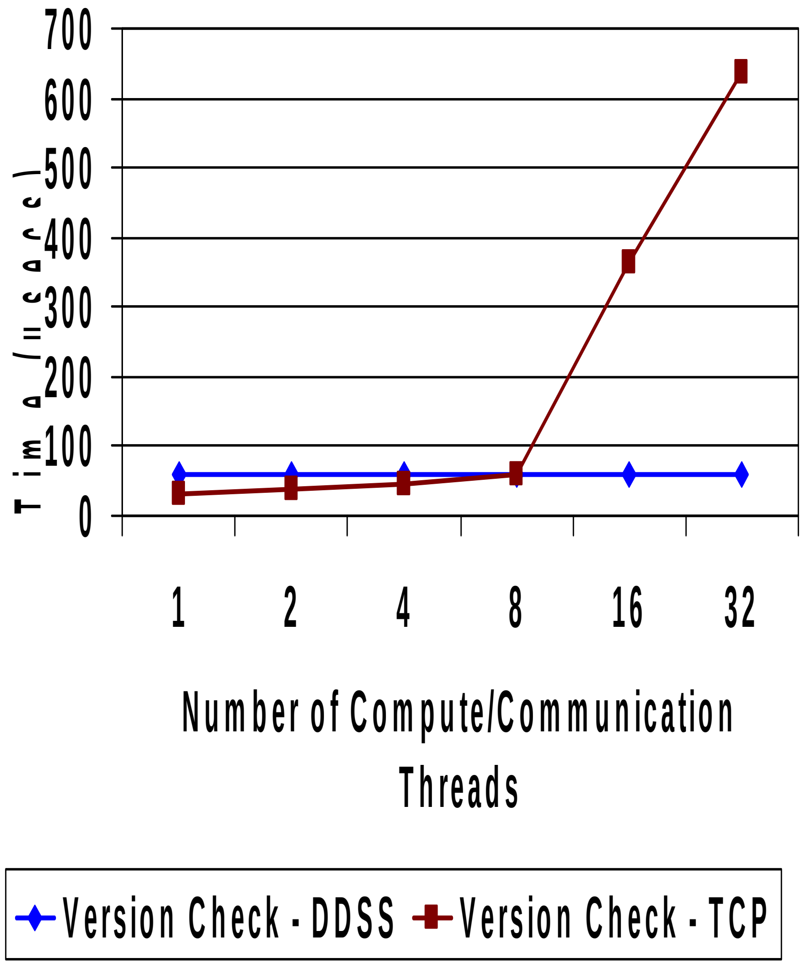
Active Resource Adaptation

- Monitors the load of different websites
- If a website is loaded, shift under-utilized servers to loaded websites
- Software Overhead of DDSS is **< 2%**



Active Caching

- Supports Strong Coherency for cached dynamic data
- Checks the back-end for current version using RDMA
- Active cache using DDSS is load-resilient



Conclusions & Future Work

- Proposed a distributed data sharing substrate
- Using DDSS, data-center applications and services, with very little modification, can get significant benefits in performance and scalability
- Implemented over OpenFabrics – applicable across InfiniBand, iWARP-capable adapters
- Future work on Fault-tolerance, support for large file sizes, advanced resource management schemes.

Acknowledgements

Our research is supported by the following organizations

- Current Funding support by

- Current Equipment support by

Web Pointers

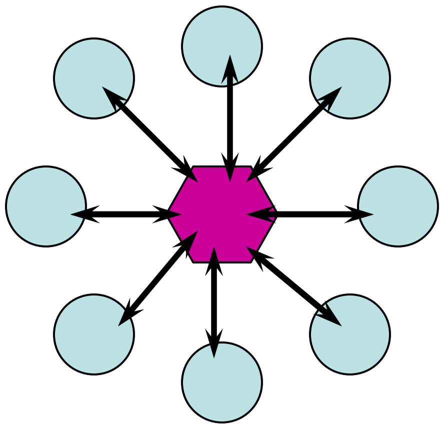
NBC-LAB

Group Homepage: http://nowlab.cse.ohio-state.edu

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

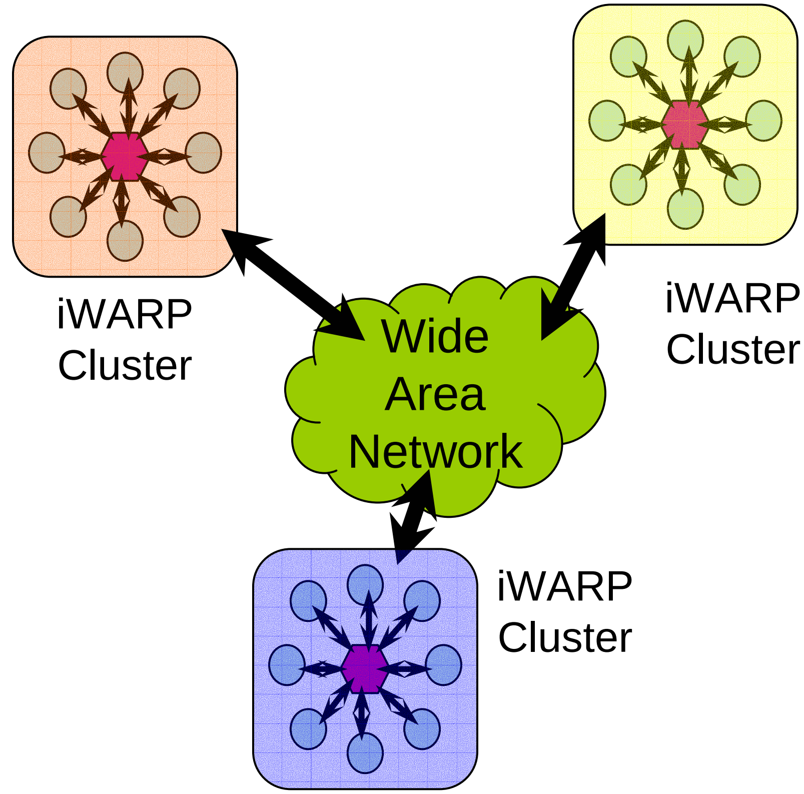
Backup Slides

High-Performance Networks in Data-Centers



Cluster-Based Data-Center Environment

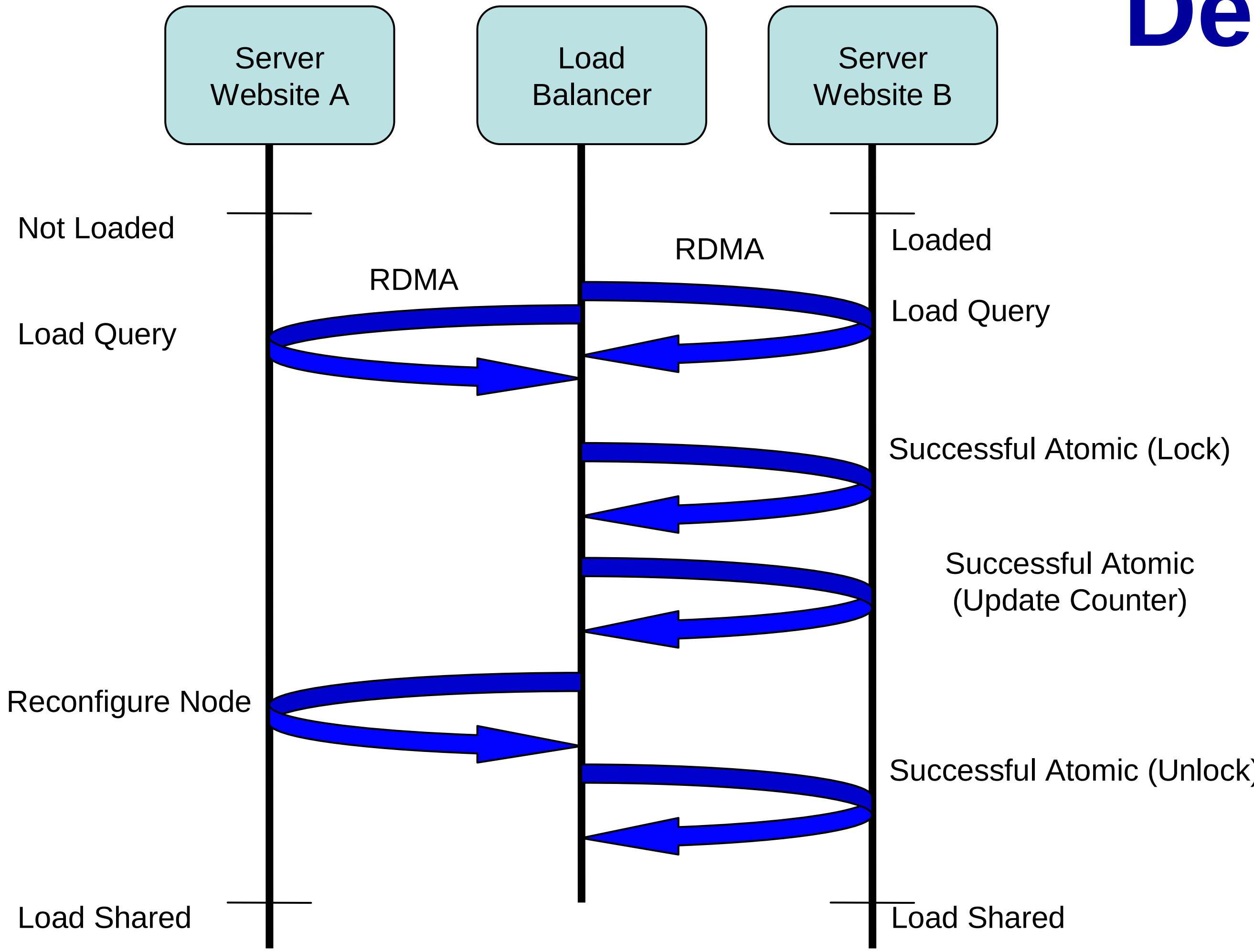
(InfiniBand, iWARP-capable Ammasso, 10 GigE)



Distributed Data-Center Environment

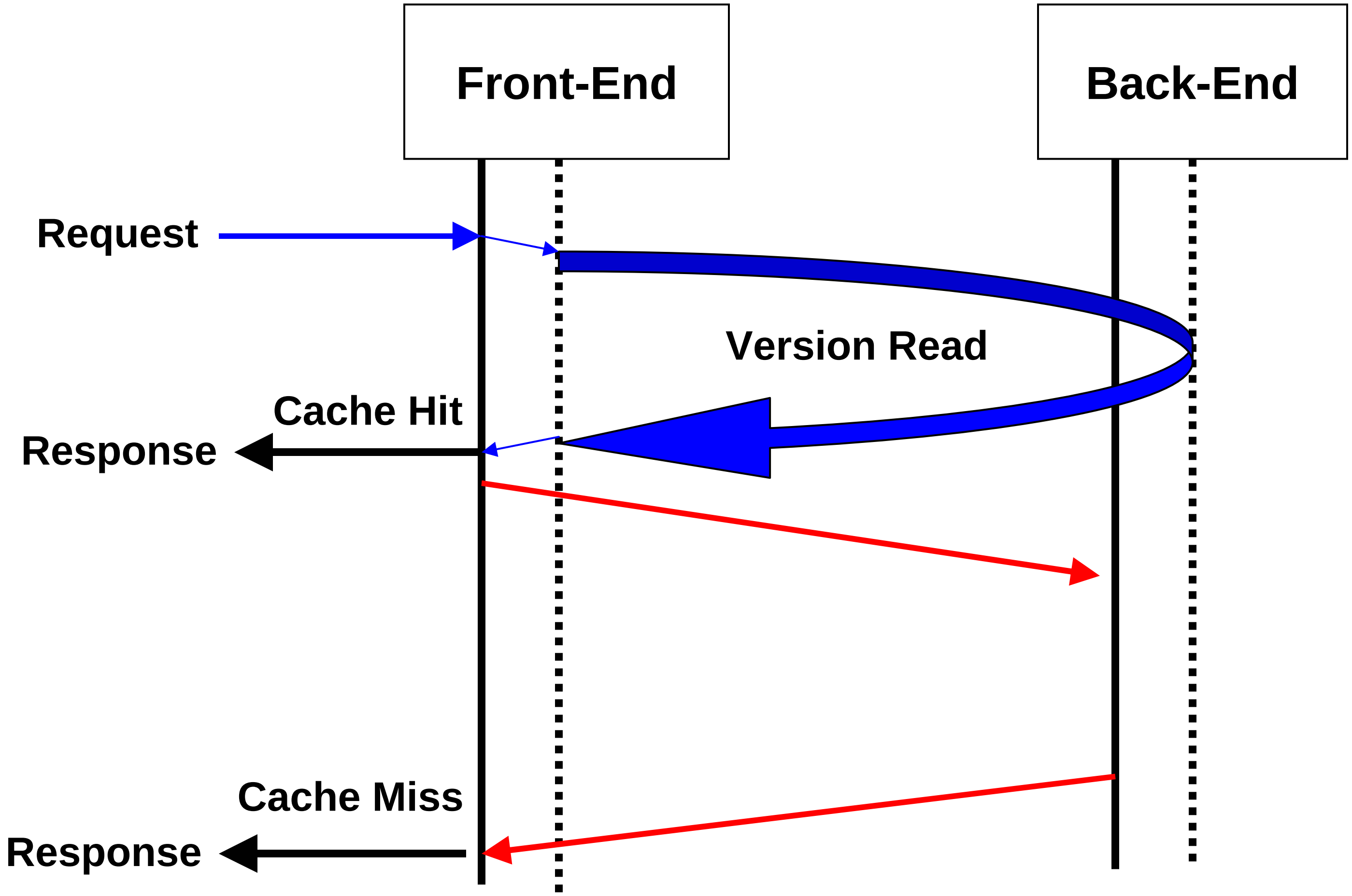
- InfiniBand, iWARP-capable adapters
  - Offer several features like RDMA, atomic operations (IB), iWARP (Ammasso, 10 GigE)

Active Resource Adaptation Design



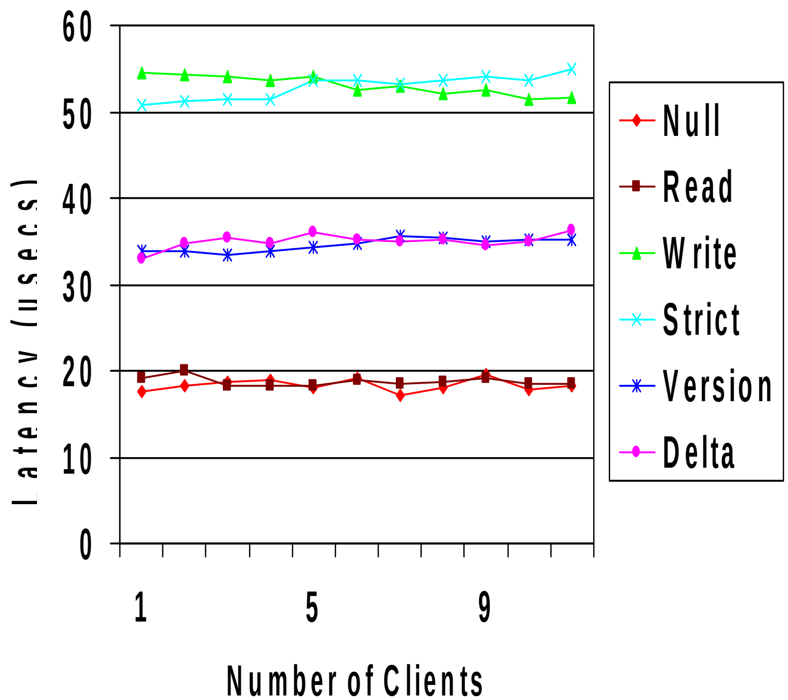
P. Balaji, K. Vaidyanathan, S. Narravula and D.K. Panda "Exploiting Remote Memory Operations to Design Efficient Reconfiguration for Shared Data-Centers over InfiniBand" presented at RAIT 2004

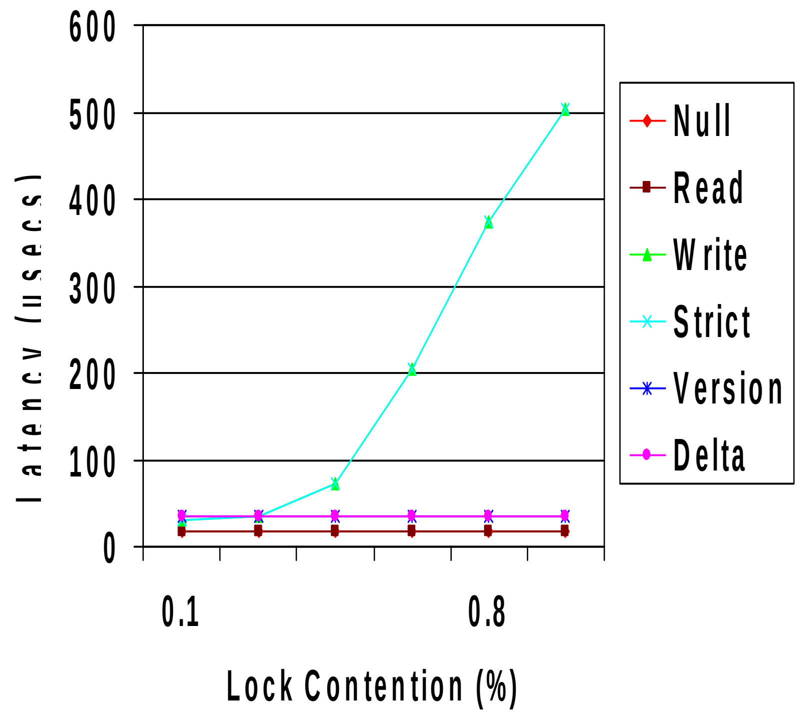
RDMA based Client Polling Design



S. Narravla, P. Balaji, K. Vaidyanathan, S. Krishnamoorthy, . u and D.K. Panda "Supporting Strong Coherency for Active Caches in Multi-Tier Data-Centers over InfiniBand" presented at SAN 2004

Microbenchmarks





- performance of put() and get() operation is less than 50 usecs