Memcached Design on High Performance RDMA Capable Interconnects

Jithin Jose, Hari Subramoni, Miao Luo, Minjia Zhang,
Jian Huang, Md. Wasi-ur-Rahman, Nusrat S. Islam,
Xiangyong Ouyang, Hao Wang, Sayantan Sur & D. K. Panda

Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University, USA

Outline

- Introduction
- Overview of Memcached
- Modern High Performance Interconnects
- Unified Communication Runtime (UCR)
- Memcached Design using UCR
- Performance Evaluation
- Conclusion & Future Work

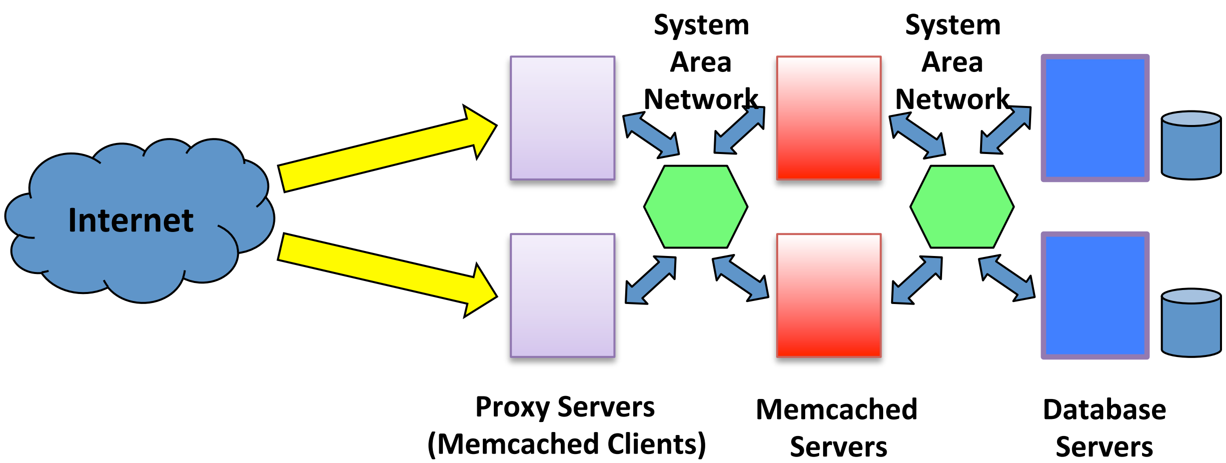
Introduction

- Tremendous increase in interest in interactive web-sites (social networking, e-commerce etc.)
- Dynamic data is stored in databases for future retrieval and analysis
- Database lookups are expensive
- Memcached – a distributed memory caching layer, implemented using traditional BSD sockets
- Socket interface provides portability, but entails additional processing and multiple message copies
- High-Performance Computing (HPC) has adopted advanced interconnects (e.g. InfiniBand, 10 Gigabit Ethernet/iWARP, RoCE)
  - Low latency, High Bandwidth, Low CPU overhead
- Many machines in Top500 list (http://www.top500.org)

Outline

- Introduction
- Overview of Memcached
- Modern High Performance Interconnects
- Unified Communication Runtime (UCR)
- Memcached Design using UCR
- Performance Evaluation
- Conclusion & Future Work

Memcached Overview

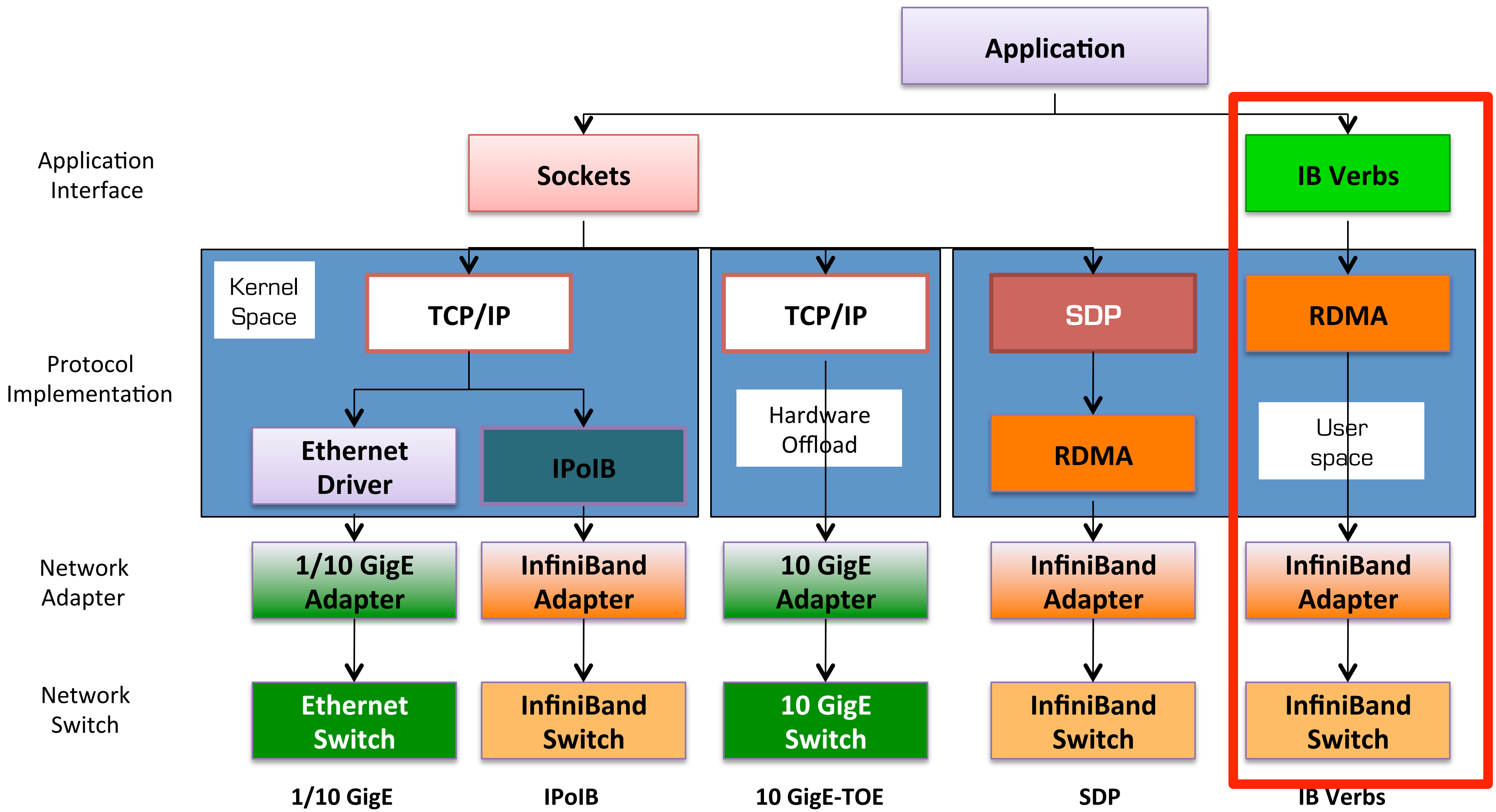


- Memcached provides a scalable distributed caching
- Spare memory in data-center servers can be aggregated to speedup lookups
- Basically a key-value distributed memory store
- Keys can be any character strings, typically MD5 sums or hashes
- Typically used to cache database queries, results of API calls or webpage rendering elements
- Scalable model, but typical usage very network intensive -Performance directly related to that of underlying networking technology

Outline

- Introduction
- Overview of Memcached
- Modern High Performance Interconnects
- Unified Communication Runtime (UCR)
- Memcached Design using UCR
- Performance Evaluation
- Conclusion & Future Work

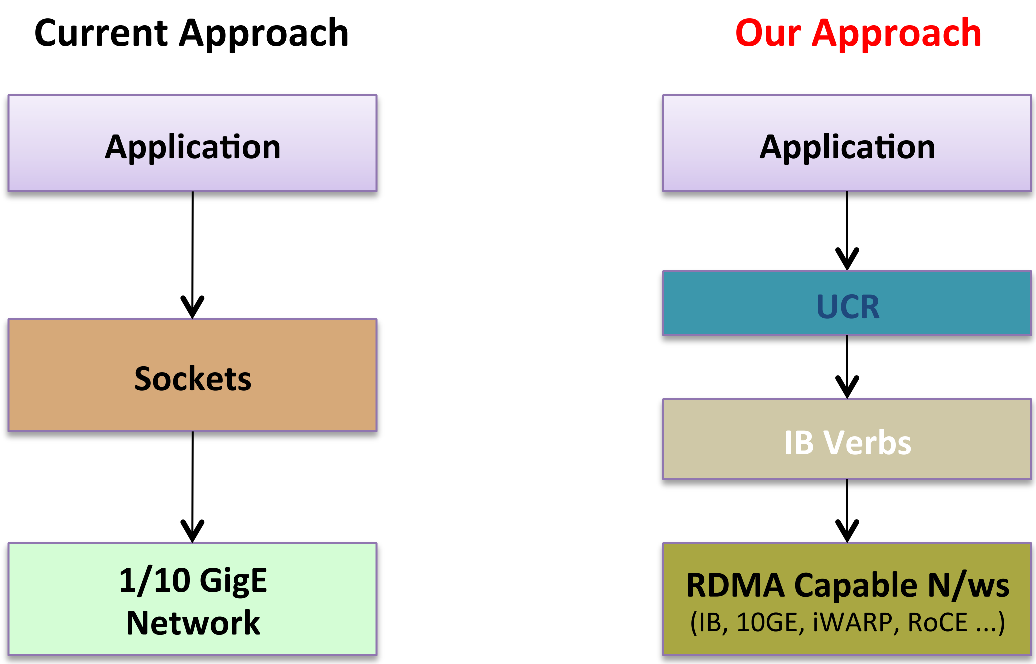
Modern High Performance Interconnects



Problem Statement

- High-performance RDMA capable interconnects have emerged in the scientific computation domain
- Applications using Memcached are still relying on sockets
- Performance of Memcached is critical to most of its deployments
- Can Memcached be re-designed from the ground up to utilize RDMA capable networks?

A New Approach using Unified Communication Runtime (UCR)



- Sockets not designed for high-performance
  - Stream semantics often mismatch for upper layers (Memcached, Hadoop)
  - Multiple copies can be involved

Outline

- Introduction & Motivation
- Overview of Memcached
- Modern High Performance Interconnects
- Unified Communication Runtime (UCR)
- Memcached Design using UCR
- Performance Evaluation
- Conclusion & Future Work

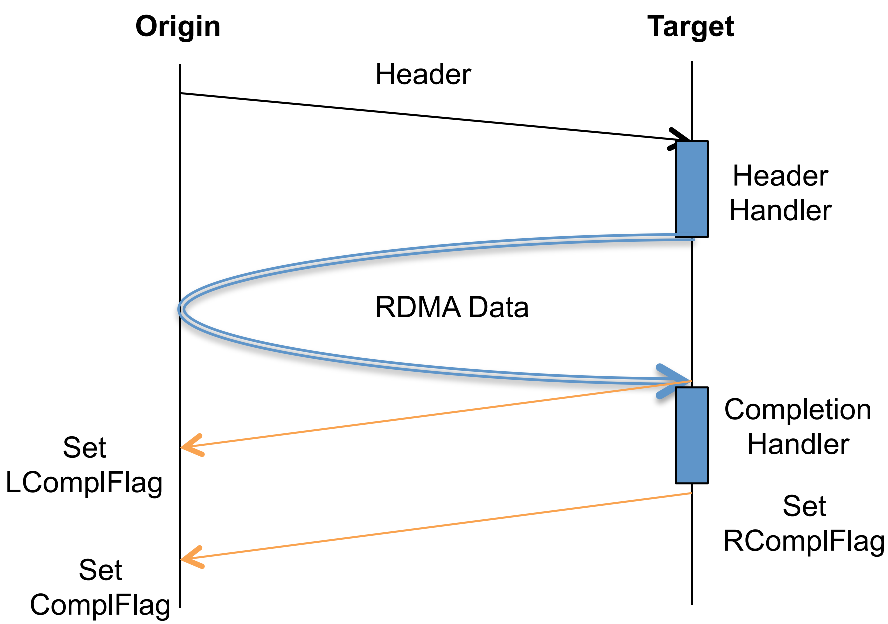
Unified Communication Runtime (UCR)

- Initially proposed to unify communication runtimes of different parallel programming models
  - J. Jose, M. Luo, S. Sur and D. K. Panda, Unifying UPC and MPI Runtimes: Experience with MVAPICH, (PGAS'10)
- Design of UCR evolved from MVAPICH/MVAPICH2 software stacks (http://mvapich.cse.ohio-state.edu/)
- UCR provides interfaces for Active Messages as well as one-sided put/get operations
- Enhanced APIs to support Cloud computing applications
- Several enhancements in UCR
  - end-point based design, revamped active-message API, fault tolerance and synchronization with timeouts.
- Communications based on endpoint, analogous to sockets

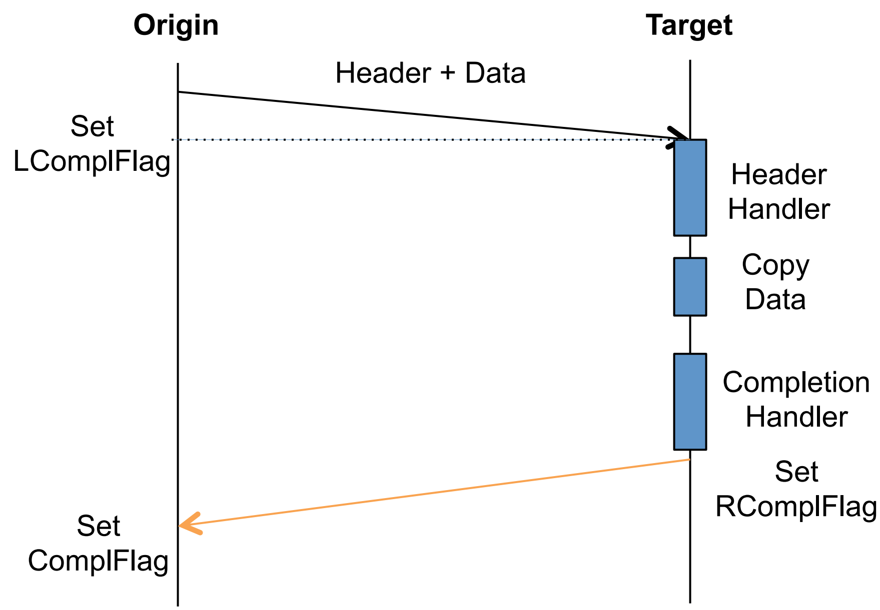
Active Messaging in UCR

- Active messages are proven to be very powerful in many environments
  - GASNet Project (UC Berkeley), MPI design using LAPI (IBM), etc.
- We introduce Active messages into the data-center domain
- An Active Message consists of two parts – header and data
- When the message arrives at the target, header handler is run
- Header handler identifies the destination buffer for the data
- Data is put into the destination buffer
- Completion handler is run afterwards (optional)
- Special flags to indicate local & remote completions (optional)

Active Messaging in UCR (contd.)



(General Active Message Functionality)

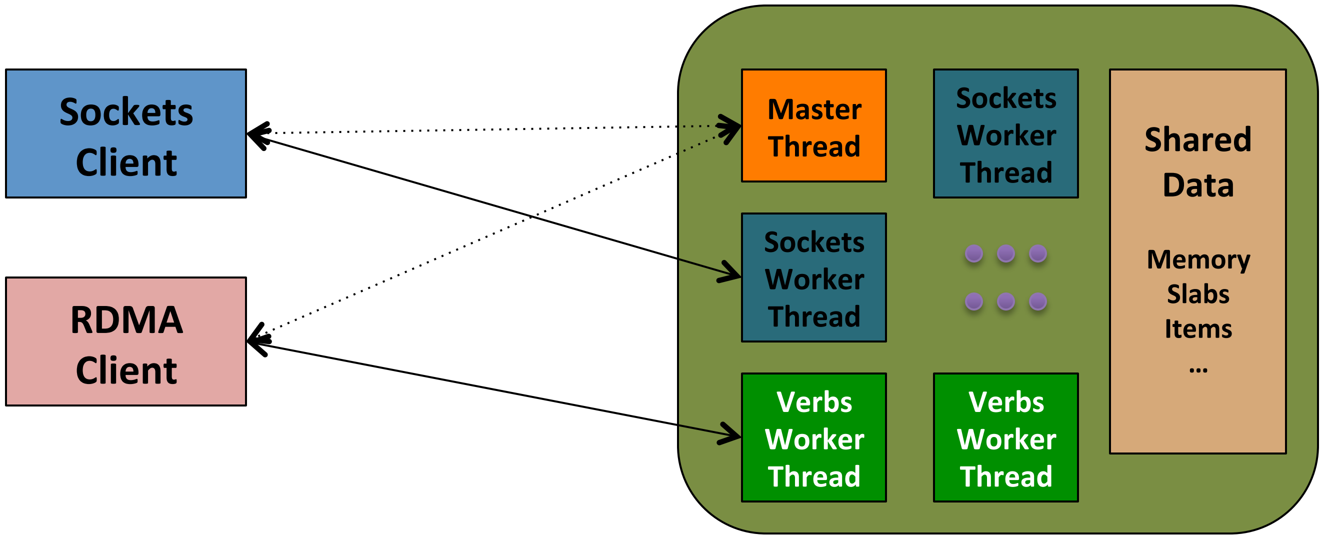


(Optimized Short Active Message Functionality)

Outline

- Introduction & Motivation
- Overview of Memcached
- Modern High Performance Interconnects
- Unified Communication Runtime (UCR)
- Memcached Design using UCR
- Performance Evaluation
- Conclusion & Future Work

Memcached Design using UCR



- Server and client perform a negotiation protocol
  - Master thread assigns clients to appropriate worker thread
- Once a client is assigned a verbs worker thread, it can communicate directly and is “bound” to that thread
- All other Memcached data structures are shared among RDMA and Sockets worker threads

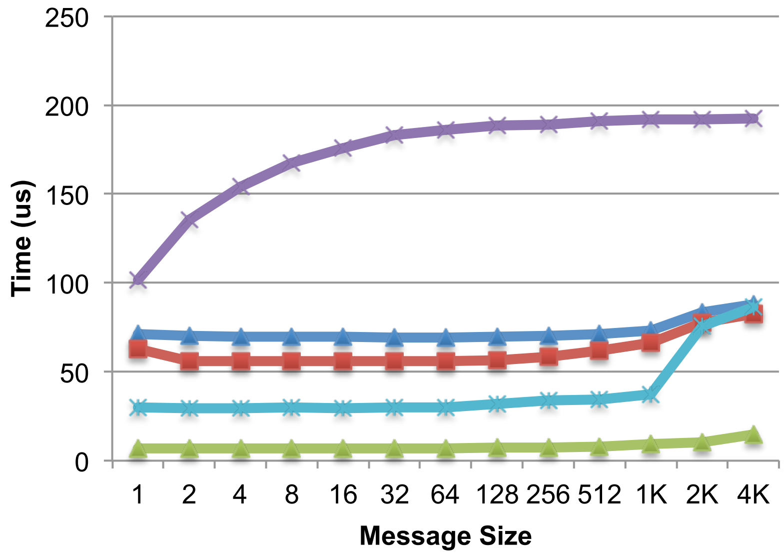
Outline

- Introduction & Motivation
- Overview of Memcached
- Modern High Performance Interconnects
- Unified Communication Runtime (UCR)
- Memcached Design using UCR
- Performance Evaluation
- Conclusion & Future Work

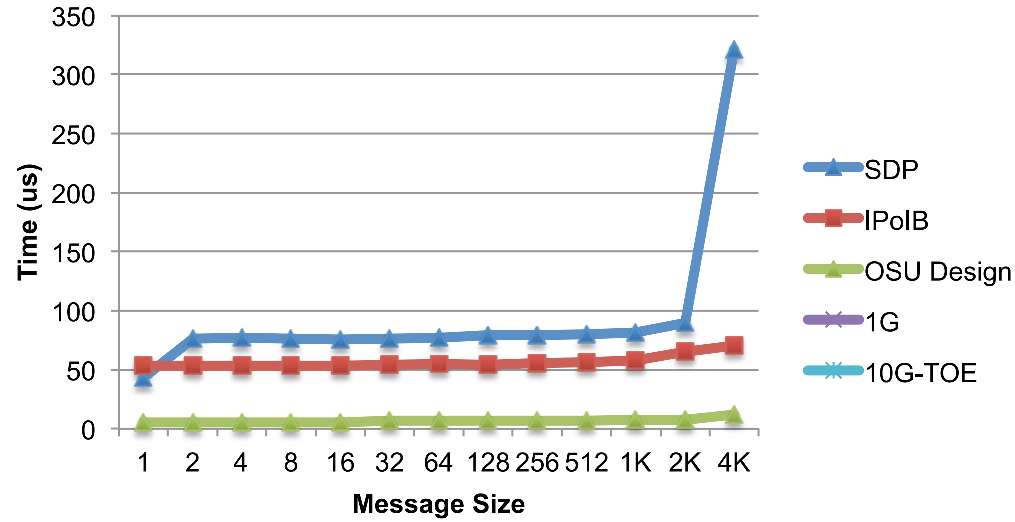
Experimental Setup

- Used Two Clusters
  - Intel Clovertown
    - Each node has 8 processor cores on 2 Intel Xeon 2.33 GHz Quad-core CPUs, 6 GB main memory, 250 GB hard disk
    - Network: 1GigE, IPoIB, 10GigE TOE and IB (DDR)
  - Intel Westmere
    - Each node has 8 processor cores on 2 Intel Xeon 2.67 GHz Quad-core CPUs, 12 GB main memory, 160 GB hard disk
    - Network: 1GigE, IPoIB, and IB (QDR)
- Memcached Software
  - Memcached Server: 1.4.5
  - Memcached Client: (libmemcached) 0.45

Memcached Get Latency (Small Message)



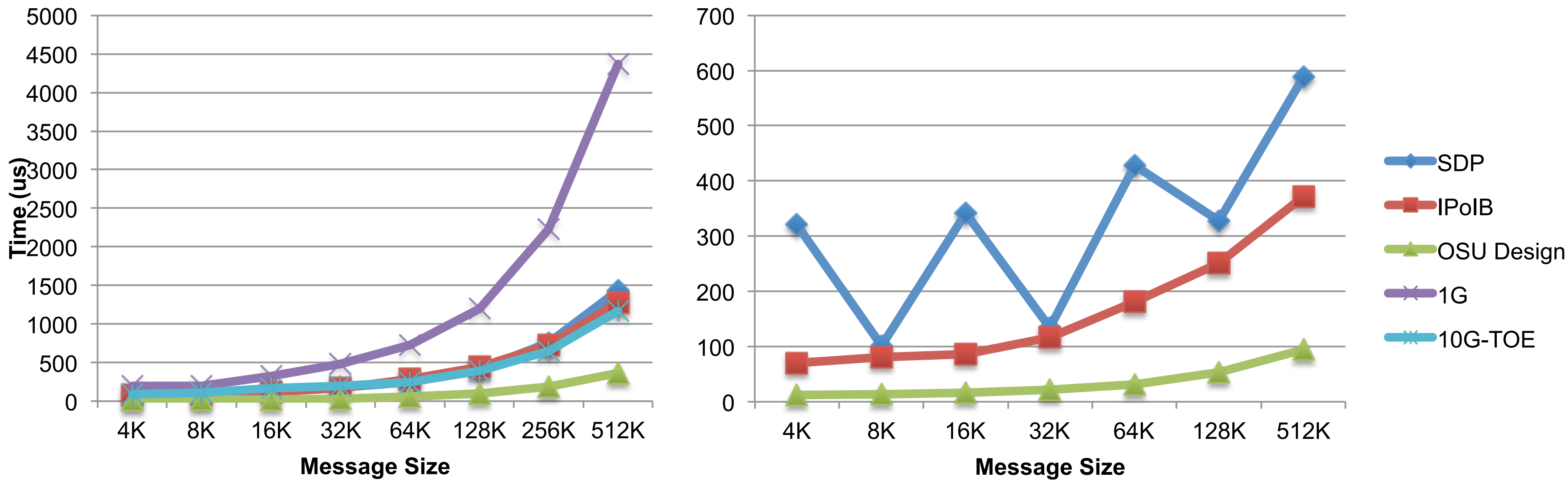
Intel Clovertown Cluster (IB: DDR)



Intel Westmere Cluster (IB: QDR)

- Memcached Get latency
  - 4 bytes – DDR: 6 us; QDR: 5 us
  - 4K bytes -- DDR: 20 us; QDR:12 us
- Almost factor of *four* improvement over 10GE (TOE) for 4KB on the DDR cluster
- Almost factor of *seven* improvement over IPoIB for 4KB on the QDR cluster

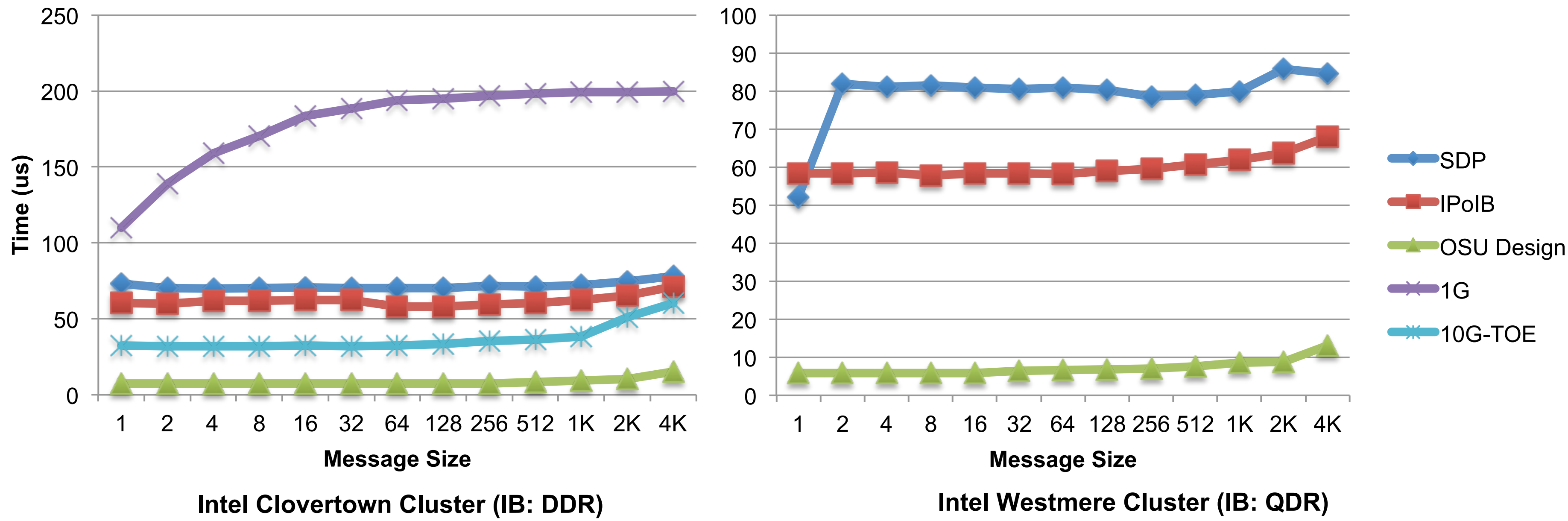
Memcached Get Latency (Large Message)



Intel Clovertown Cluster (IB: DDR) **Intel Westmere Cluster (IB: QDR)**

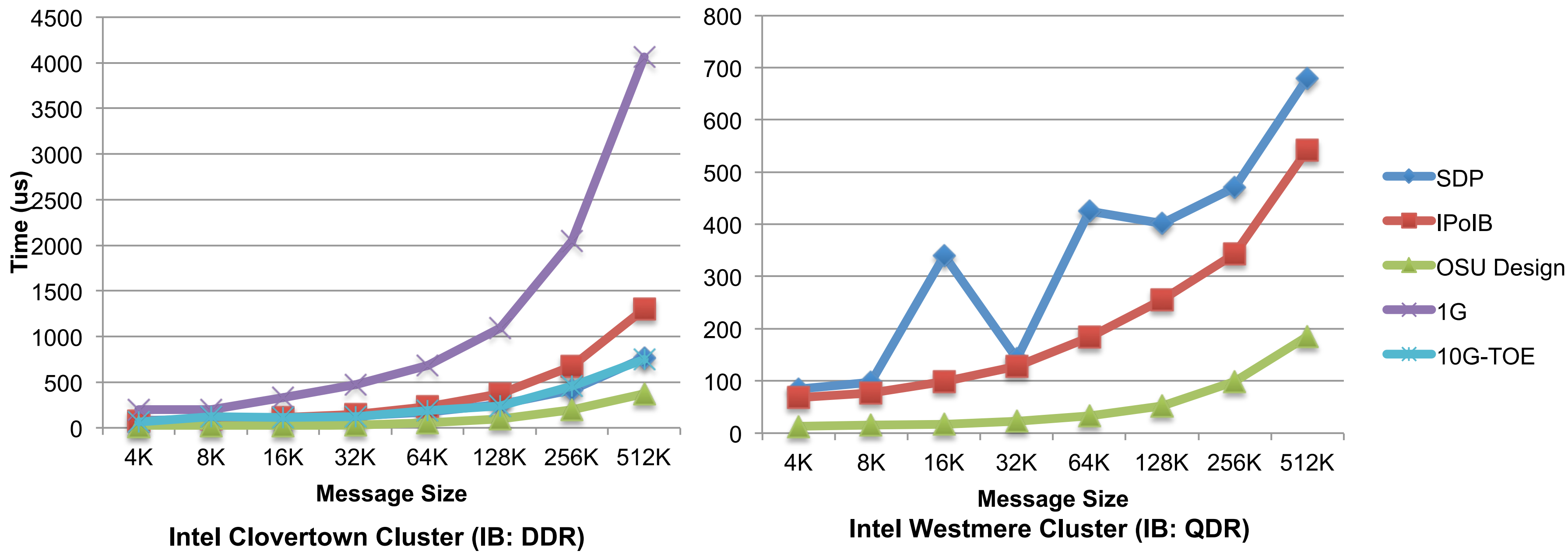
- Memcached Get latency
  - 8K bytes – DDR: 17 us; QDR: 13 us
  - 512K bytes -- DDR: 362 us; QDR: 94 us
- Almost factor of *three* improvement over 10GE(TOE) for 512KB on the DDR cluster
- Almost factor of *four* improvement over IPoIB for 512K bytes on the QDR cluster

Memcached Set Latency (Small Message)



- Memcached Set latency
  - 4 bytes – DDR: 7 us; QDR: 5 us
  - 4K bytes -- DDR: 15 us; QDR:13 us
- Almost factor of *four* improvement over 10GE (TOE) for 4KB on the DDR Cluster
- Almost factor of *six* improvement over IPoIB for 4KB on the QDR Cluster

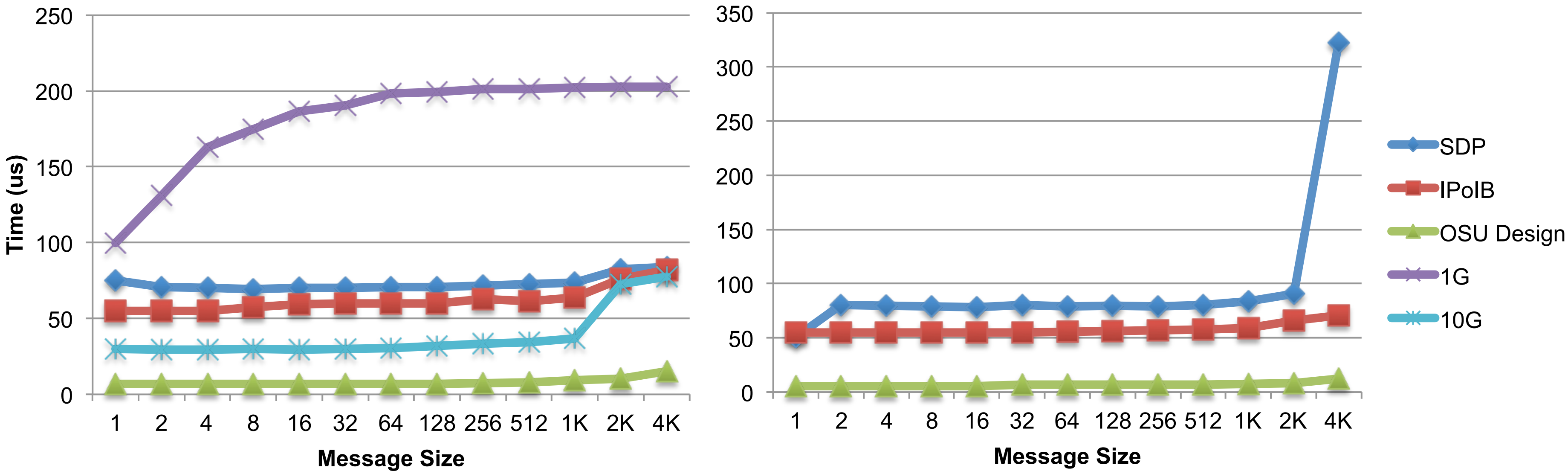
Memcached Set Latency (Large Message)



- Memcached Get latency
  - 8K bytes – DDR: 18 us; QDR: 15 us
  - 512K bytes -- DDR: 375 us; QDR:185 us
- Almost factor of *two* improvement over 10GE (TOE) for 512KB on the DDR cluster
- Almost factor of *three* improvement over IPoIB for 512KB on the QDR cluster

Memcached Latency

(10% Set, 90% Get)



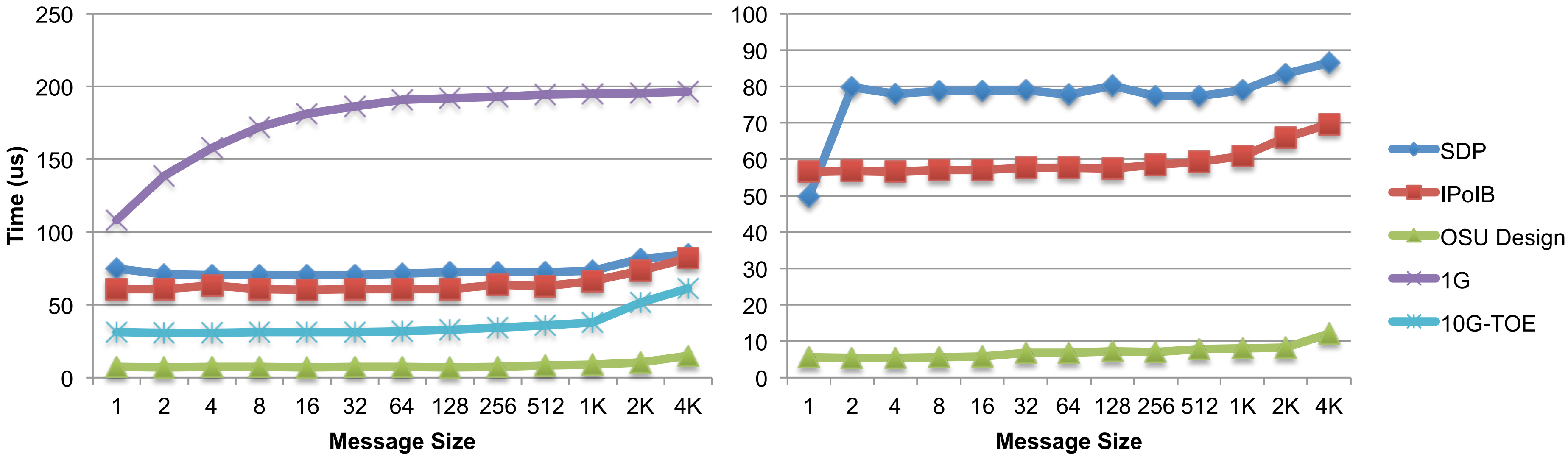
Intel Clovertown Cluster (IB: DDR)

Intel Westmere Cluster (IB: QDR)

- Memcached Get latency
  - 4 bytes – DDR: 7 us; QDR: 5 us
  - 4K bytes -- DDR: 15 us; QDR:12 us
- Almost factor of *four* improvement over 10GE (TOE) for 4K bytes on the DDR cluster

Memcached Latency

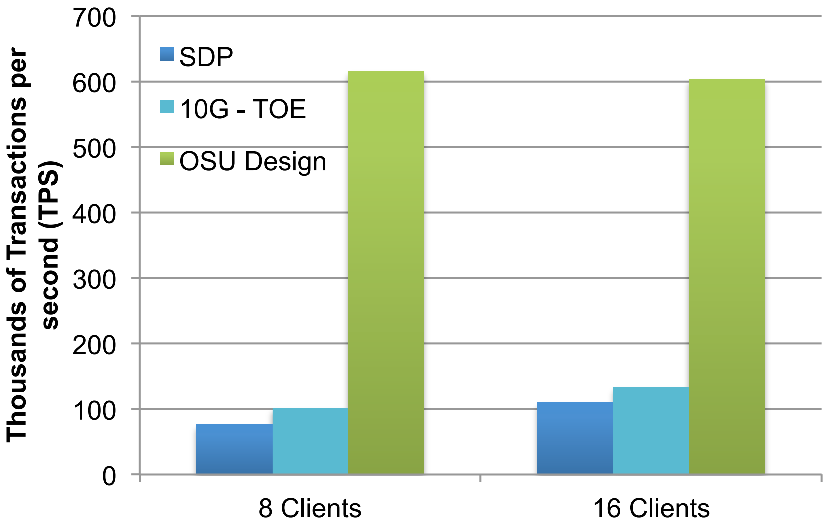
(50% Set, 50% Get)



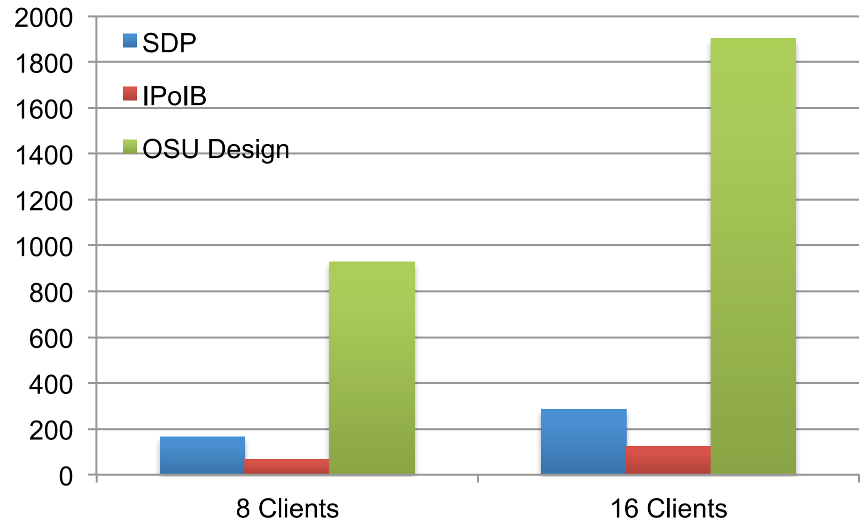
Intel Clovertown Cluster (IB: DDR) | Intel Westmere Cluster (IB: QDR)

- Memcached Get latency
  - 4 bytes – DDR: 7 us; QDR: 5 us
  - 4K bytes -- DDR: 15 us; QDR:12 us
- Almost factor of *four* improvement over 10GE (TOE) for 4K bytes on the DDR cluster

Memcached Get TPS (4byte)



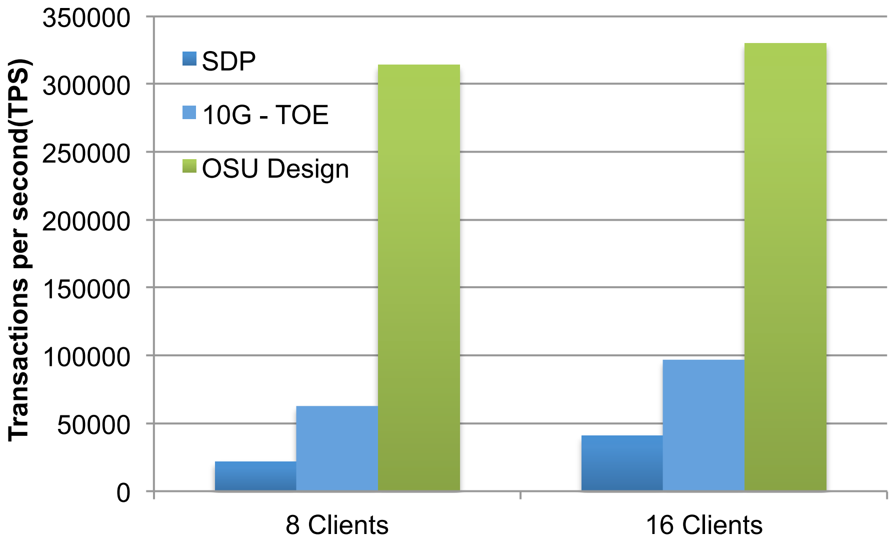
Intel Clovertown Cluster (IB: DDR)



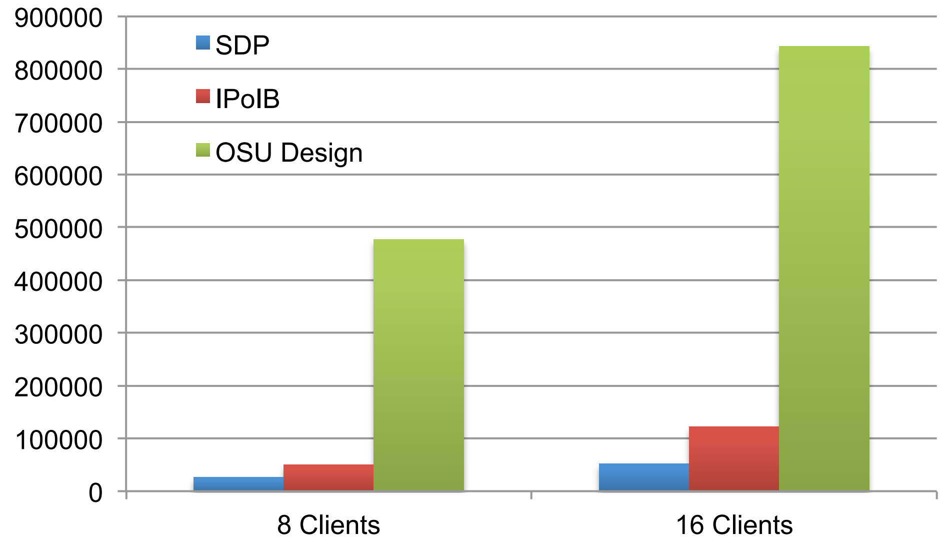
Intel Westmere Cluster (IB: QDR)

- Memcached Get transactions per second for 4 bytes
  - On IB DDR about 600K/s for 16 clients
  - On IB QDR 1.9M/s for 16 clients
- Almost factor of *six* improvement over 10GE (TOE)
- Significant improvement with native IB QDR compared to SDP and IPoIB

Memcached Get TPS (4KB)



Intel Clovertown Cluster (IB: DDR)



Intel Westmere Cluster (IB: QDR)

- Memcached Get transactions per second for 4K bytes
  - On IB DDR about 330K/s for 16 clients
  - On IB QDR 842K/s for 16 clients
- Almost factor of *four* improvement over 10GE (TOE)
- Significant improvement with native IB QDR

Outline

- Introduction & Motivation
- Overview of Memcached
- Modern High Performance Interconnects
- Unified Communication Runtime (UCR)
- Memcached Design using UCR
- Performance Evaluation
- Conclusion & Future Work

Conclusion & Future Work

- Described a novel design of Memcached for RDMA capable networks
- Provided a detailed performance comparison of our design compared to unmodified Memcached using sockets over RDMA and 10GE networks
- Observed significant performance improvement with the proposed design
    - Factor of four improvement in Memcached get latency (4K bytes)
    - Factor of six improvement in Memcached get transactions/s (4 bytes)
- We plan to improve UCR by taking into account the many features in OpenFabrics API , Unreliable Datagram transport and designing iWARP and RoCE versions of UCR, and thereby scaling Memcached
- We are working on enhancing the Hadoop/HBase designs for RDMA capable networks

Thank You!

{jose, subramon, luom, zhanjmin, huangjia, rahmanmd, islamn, ouyangx, wangh, surs,panda}@cse.ohio-state.edu

Network-Based Computing Laboratory

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

MVAPICH Web Page

http://mvapich.cse.ohio-state.edu/