Design and performance evaluation of NUMA-aware RDMA-based end-to-end data transfer systems

Yufei Ren, Tan Li, Dantong Yu, Shudong Jin, Thomas G. Robertazzi

Presented by Zach Yannes

Introduction

- Need to transfer large amounts of data long distances (end-to-end high performance data transfer)
- i.e. inter-data center transfer
- Goal: design a network to overcome three common bottlenecks of large-haul end-to-end transfer systems
  - Achieve 100 Gbps data transfer throughput

Bottleneck I

- **Problem**: Processing bottlenecks of individual hosts
- **Old solution**: Multi-core hosts to provide ultra high-speed data transfers
  - Uniform memory access (UMA)
  - All processors share memory uniformly
  - Access time independent of where memory retrieved from
  - Best used for applications shared by multiple users
  - However, as number of CPU sockets and cores increases, latency across all CPU cores decreases

Bottleneck I, Cont'd

- **New solution**: Replace external memory controller hub with a core memory controller on the CPU die
  - Separate memory banks for each processor
  - Non-uniform memory access (NUMA)
  - CPU-to-bank latencies no longer independent (exploits temporal locality)
  - Reduces volume and power consumption
  - Tuning an application for local memory improves performance

Bottleneck II

- **Problem**: Applications do not utilize full network speed
- **Solution**: Employ advanced networking techniques and protocols
  - Remote direct memory access (RDMA)
    - Network adapters transfer large memory blocks; eliminates data copies in protocol stacks
    - Improves performance of high-speed networks
    - Low latency and high bandwidth
  - RDMA over Converged Ethernet (RoCE)
    - RDMA extension for joining long-distance data centers (thousands of miles)

Bottleneck III

- **Problem**: Low bandwidth magnetic disks or flash SSDs in backend storage system
  - Host's processing speed > memory access time
  - Lowers throughput
- **Solution**: Build storage network with multiple storage components
  - Bandwidth equivalent to host's processing speed
  - Requires iSCSI extension for RDMA (iSER)
    - Enables RDMA networks use of SCSI commands and objects

Experiment

- Hosts: Two IBM X3640 M4
- Connected by three pairs of 40 Gbps RoCE connections
  - Each RoCE adapter installed in eight-lane PCI Express 3.0
- Bi-directional network
- Possible 240 Gbps max bandwidth of system
- Measured memory bandwidth and TCP/IP stack performance before and after tuning for NUMA locality

Experiment, Cont'd

1) Measuring maximum memory bandwidth of hosts
    - Compiled STREAM (Memory bandwidth benchmark)
    - OpenMP option for multi-threaded testing
    - Peak memory bandwidth for *Triad test* for two NUMA nodes is 400 Gbps
        - Socket-based network applications require two data copies per operation
        - Max TCP/IP bandwidth is 200 Gbps

Experiment, Cont'd

2) Measure max bi-directional end-to-end bandwidth

- Test TCP/IP stack performance via *iperf*
- Only want to test accesses that require more than one memory read, increase sender's buffer
  - Cannot store entire buffer in cache, removes cache effect from test
- Average aggregate bandwidth is 83.5 Gbps
- 35% of CPU usage from kernel and user space memory copy routines (i.e. *copy_user_generic_string*)

Experiment Observations

- Experiment repeated after tuning *iperf* for NUMA locality
- Average aggregate bandwidth increased to 91.8 Gbps
  - 10% higher than default Linux scheduler
- Two observations of end-to-end network data transfer:
  - TCP/IP protocol stack has large processing overhead
  - NUMA has greater hardware costs for same latency
    - Requires additional CPU cores to handle synchronization

End-to-End Data Transfer System Design

- Back-End Storage Area Network Design
  - Use iSER protocol to communicate between "initiator" (client) and "target" (server)
    - Initiator sends I/O requests to server who transfers the data
    - Initiator *read* = RDMA *write* from target
    - Initiator *write* = RDMA *read* from target

End-to-End Data Transfer System Design

- Back-End Storage Area Network Design, Cont'd
  - Integrate NUMA into target
  - Requires locations of PCI devices
  - Two methods:
    1. *numactl* – Binds target process to logical NUMA node
       - Explicit, static NUMA policy
    2. *libnuma* – Integrate into target implementation
       - Too complicated
       - Scheduling algorithm for each I/O request

End-to-End Data Transfer System Design

- Back-End Storage Area Network Design, Cont'd
  - File system = Linux *tmpfs*
  - Map NUMA node memory to specific location of memory file using *mpol* and *remount*
  - Each node handles local I/O requests for a mapped target process
    - Each I/O request (from initiator) handled by a separate link
    - Low latency → best throughput

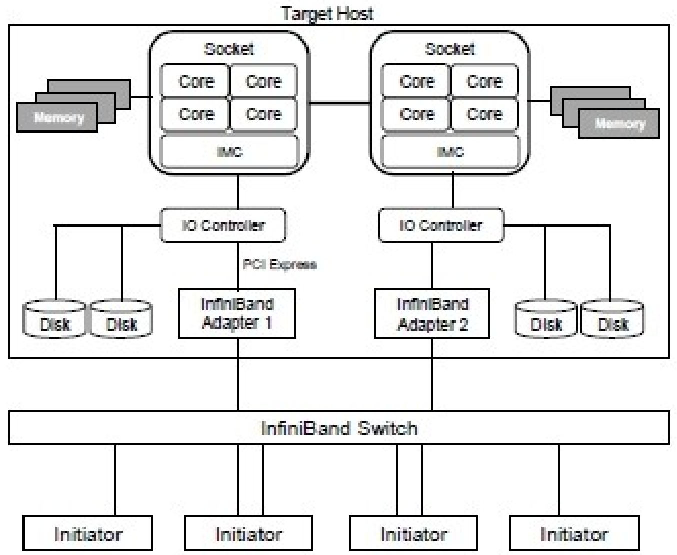


Figure 2: iSER tuning in NUMA architecture with multiple adapters.

End-to-End Data Transfer System Design

- RDMA Application Protocol
  - Data loading
  - Data transmission
  - Data offloading
  - Throughput and latency depend on type of data storage

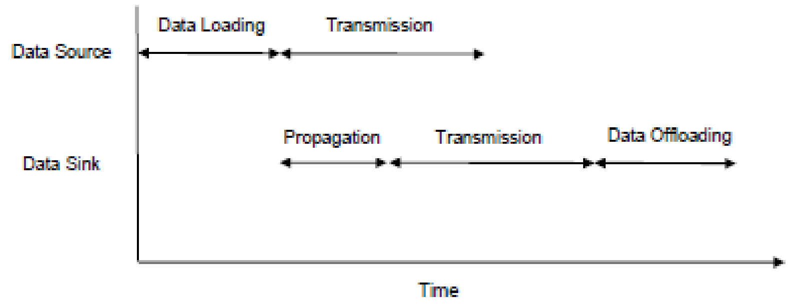


Figure 3: Data block transfer delay breakdown.

End-to-End Data Transfer System Design

- RDMA Application Protocol, Cont'd
  - Uses RFTP, RDMA-based file transfer protocol
  - Supports pipelining and parallel operations

Experiment Configuration

- Back-end
  - Two Mellanox InfiniBand adapters
  - Each with FDR, 56 Gbps
  - Connected to Mellanox FDR InfiniBand switch
  - Maximum load/offload bandwidth: 112 Gbps
- Front-end: Three pairs of QDR 40 Gbps RoCE network cards connect RFTP client and server
  - Maximum aggregate bandwidth: 120 Gbps

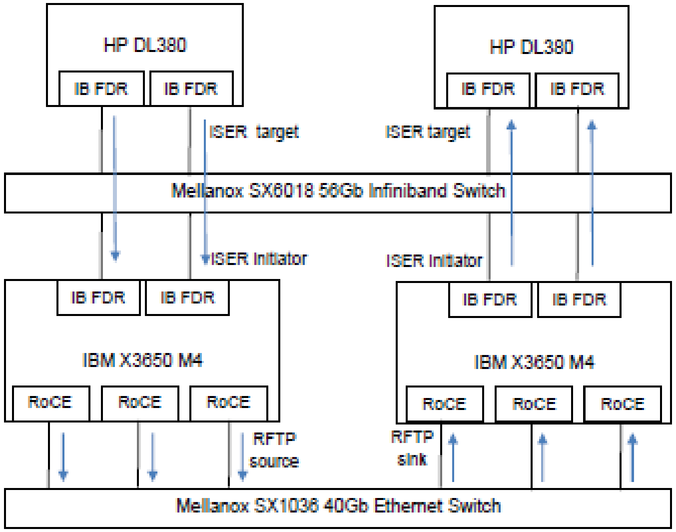


Figure 5: RDMA-based end-to-end system connectivity in LAN.

Experiment Configuration, Cont'd

- Wide area network (WAN)
  - Provided by DOE's Advanced Networking Initiative (ANI)
  - 40 Gbps RoCE wide-area network
  - 4000-mile link in loopback network
  - WANs connected by 100 Gbps router

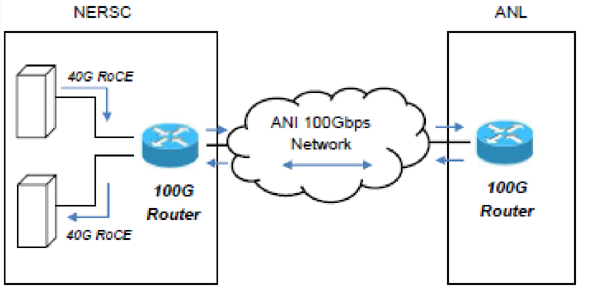


Figure 6: The DOE's ANI 40 Gbps RoCE WAN between NERSC and ANL. This 4000-mile link is a loopback network from NERSC to ANL and then back to NERSC. The RTT of the link is about 95 milliseconds.

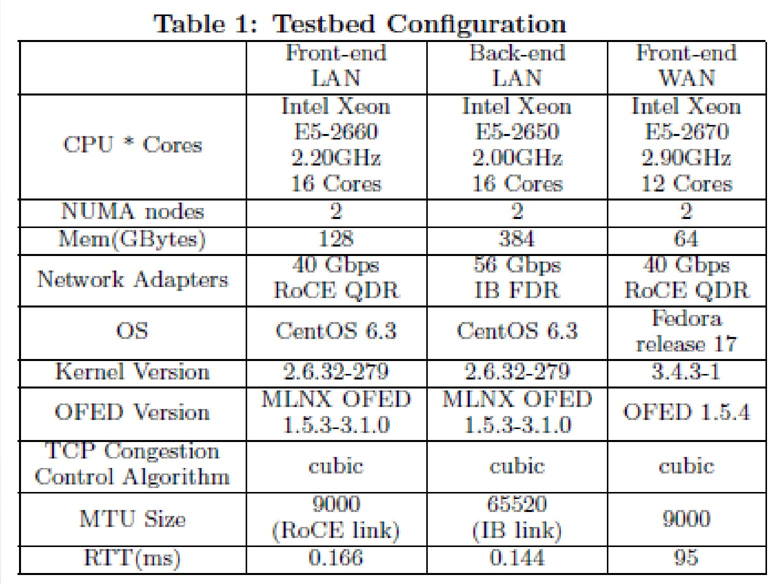


Table 1: Testbed Configuration

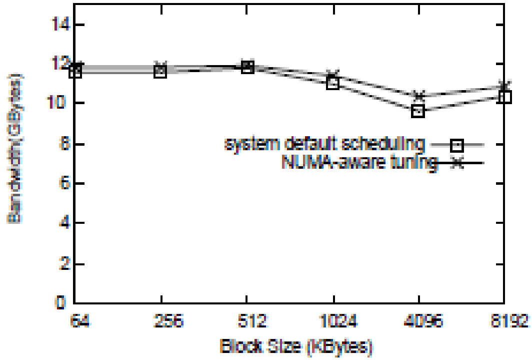
	Front-end LAN	Back-end LAN	Front-end WAN
CPU * Cores	Intel Xeon E5-2660 2.20GHz 16 Cores	Intel Xeon E5-2650 2.00GHz 16 Cores	Intel Xeon E5-2670 2.90GHz 12 Cores
NUMA nodes	2	2	2
Mem(GBytes)	128	384	64
Network Adapters	40 Gbps RoCE QDR	56 Gbps IB FDR	40 Gbps RoCE QDR
OS	CentOS 6.3	CentOS 6.3	Fedora release 17
Kernel Version	2.6.32-279	2.6.32-279	3.4.3-1
OFED Version	MLNX OFED 1.5.3-3.1.0	MLNX OFED 1.5.3-3.1.0	OFED 1.5.4
TCP Congestion Control Algorithm	cubic	cubic	cubic
MTU Size	9000 (RoCE link)	65520 (IB link)	9000
RTT(ms)	0.166	0.144	95

Experiment Scenarios

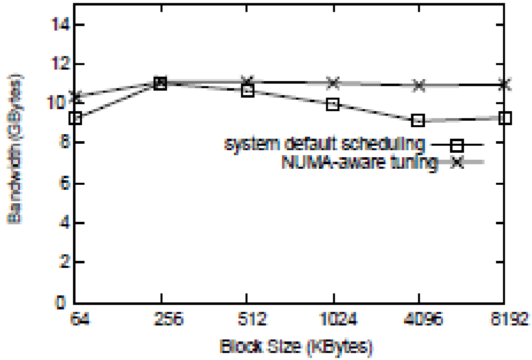
- Evaluated under three scenarios:
  1) Back-end system performance with NUMA-aware tuning
  2) Application performance in end-to-end LAN
  3) Network performance over a 40 Gbps RoCE long distance path in wide-area networks

Experiment 1

1) Back-end system performance with NUMA-aware tuning
   - Performance gains plateau after a number of threads (threshold=4)
   - Too many I/O threads increases contention
   - Benchmark: Flexible I/O tester (fio)
   - Read bandwidth: 7.8% increase from NUMA binding
   - Write bandwidth: Up to 19% increase for >4MB block sizes



(a) Read bandwidth

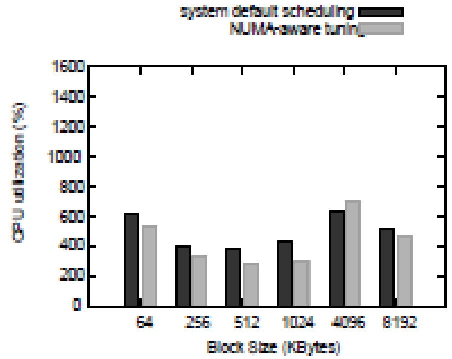


(b) Write bandwidth

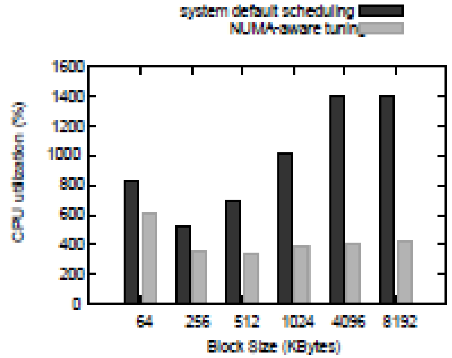
Figure 7: iSER bandwidth comparison between default scheduling and NUMA-tuning.

Experiment 1

1) Back-end system performance with NUMA-aware tuning
    - Read CPU utilization
        - insignificant decrease
    - Write CPU utilization
        - NUMA-aware tuning utilizes CPU up to three times less than default Linux scheduling



(a) Read CPU utilization



(b) Write CPU utilization

Figure 8: iSER CPU utilization comparison between default scheduling and NUMA-tuning.

Experiment 1

1) Back-end system performance with NUMA-aware tuning
   - Read operation performance does not improve
     - Already has little overhead
     - On *tmpfs,* regardless of NUMA-aware tuning, the data copies are not set to “modified”, only “cached” or “shared”
     - On *tmpfs, a* write invalidates all data copies in other NUMA nodes without NUMA tuning, or only invalidates data copies on local NUMA node when tuned
   - Read requests have 7.5% higher bandwidth than write requests
     - Hypothesized to result from RDMA write implementation
     - RDMA write writes data directly to initiator's memory for transfer

Experiment 2

2) Application performance in end-to-end LAN

- Issue: How to adapt application to real-world scenarios?
- Solution: Application interacts with file system through POSIX interfaces
  - More portable, simple
  - Comparable throughput differences via different protocols
    - iSER protocol
    - Linux universal ext4 FS
    - XFS over exported block devices ← selected FS

Experiment 2

2) Application performance in end-to-end LAN

- Evaluated end-to-end performance between RFTP and GridFTP
- Bound processes to a specified NUMA node (*numactl*)
- RFTP has 96% effective bandwidth
- GridFTP has 30% effective bandwidth (max is 94.8 Gbps)
    - Overhead from kernel-user data copy and interrupt handling
    - Single-threaded, waits on I/O request
    - Requires higher CPU consumption to offset I/O delays
    - Front-end send/recv hosts suffer cache effect

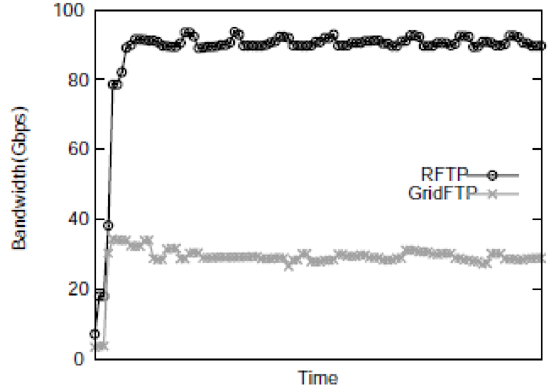


Figure 9: Throughput of end-to-end data transfer over 25 minutes.

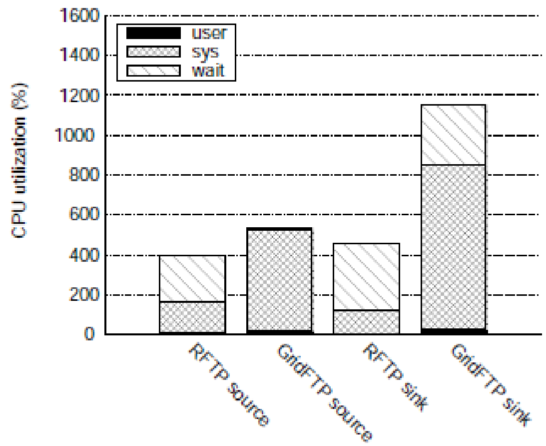


Figure 10: CPU utilization breakdown for RFTP and GridFTP.

Experiment 2

2) Application performance in end-to-end LAN (Bi-directional)

- Evaluated bi-directional end-to-end performance between RFTP and GridFTP
- Same configuration, but each end sends simultaneous messages
- Full bi-directional bandwidth not achieved
  - RFTP = 83% improvement from unidirectional
  - GridFTP = 33% improvement from unidirectional
- resource contention
  - Intense parallel I/O requests (back-end hosts)
  - Memory copies
  - Higher protocol processing overhead (front-end hosts)

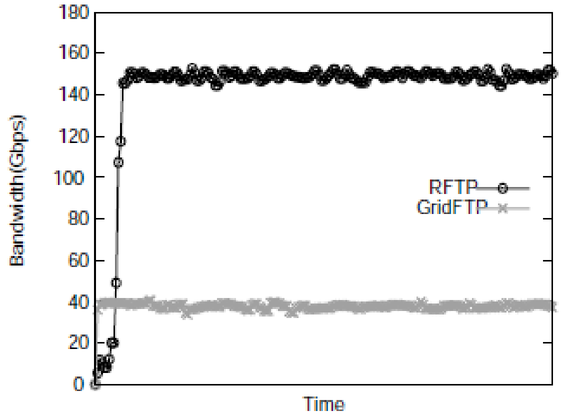


Figure 11: Throughput of bi-directional end-to-end data transfer over 50 minutes.

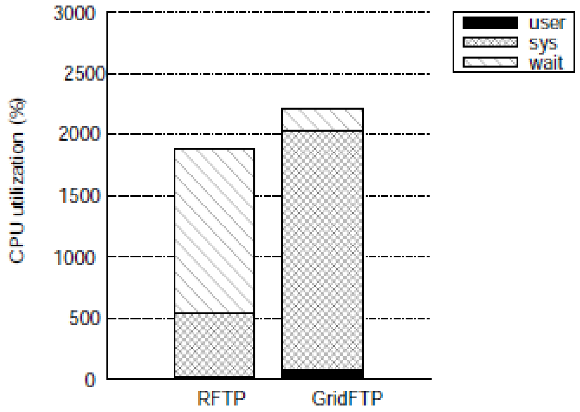


Figure 12: CPU utilization breakdown for RFTP and GridFTP bi-directional test.

Experiment 3

3) Network performance over a 40 Gbps RoCE long distance path in wide-area networks

- Issue: How to achieve 100+ Gbps on RoCE links
- Solution: Replace traditional network protocols with RFTP
- Assumption: If RFTP performs well over RoCE links, full end-to-end transfer system will perform equally well (exclude protocol overhead)
- RFTP utilizes 97% raw bandwidth
- Control message processing overhead ~ 1 / (Message block size)

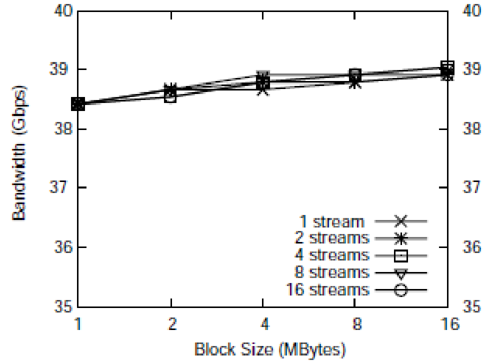
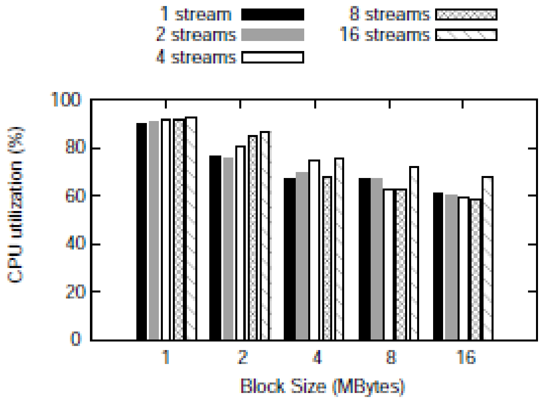
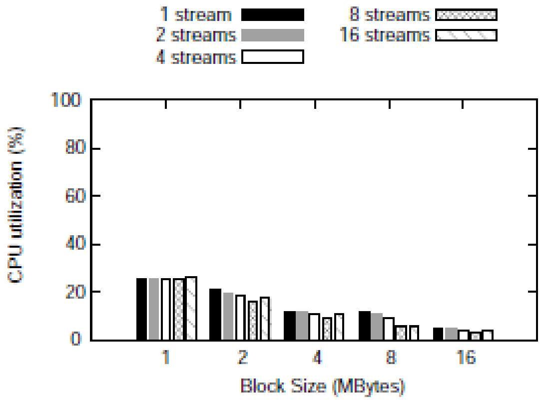


Figure 13: RFTP bandwidth with various block sizes and numbers of streams.



(a) RFTP sending side CPU utilization



(b) RFTP receiving side CPU utilization

Figure 14: RFTP CPU utilization with various block sizes and numbers of streams.

Experiment 3

3) Network performance over a 40 Gbps RoCE long distance path in wide-area networks

- Control message processing overhead ~ 1 / (Message block size)
- Therefore, increased bandwidth and lower CPU consumption as message block size increases
- Network data transfer performance not affected by long latency (due to RFTP)
  - Can scale to 1000+ servers and long-haul (inter-data center) network links
  - Used for DOE's National Laboratories and cloud data centers

Conclusion

- Need to transfer large amounts of data long distances (end-to-end high performance data transfer)
- Tested using LANs and WANs, evaluating:
  1) Back-end system performance with NUMA-aware tuning
     - Improve write operation bandwidth by ~20%
     - Utilizes CPU up to three times less
  2) Application performance in end-to-end LAN
     - RFTP (parallelized) has lower I/O-request overhead than GridFTP (single-threaded)
     - Full bi-directional bandwidth impossible due to resource contention
  3) Network performance over a 40 Gbps RoCE long distance path in wide-area networks
     - Message block size inversely proportional to bandwidth,CPU utiliz.
     - RFTP can be scaled to more servers and longer distance