RDMA in large scale databases

Qinghe Jing & Jiacheng Xia
Data centers
Networking in data centers
Outline

• What is RDMA?
• RDMA-accelerated KV-stores
• RDMA in database designs
• RDMA-optimized join operations
• Conclusion
What is RDMA?

• Abbr. for Remote Direct Memory Access
• First adapted by supercomputing society
• As name suggests, reads remote memory directly.
• Higher throughput, lower latency and lower CPU burden than traditional networking stack (i.e. TCP/IP)
RDMA vs TCP/IP

RDMA doesn’t need many memory copies, saves time and CPU resources
High overhead of TCP
Low overhead at high bandwidth

Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz, two sockets 28 cores

RDMA: Single QP, 88 Gb/s, 1.7% CPU

TCP: Eight connections, 30-50Gb/s, Client: 2.6%, Server: 4.3% CPU
Connections between servers in RDMA

- UD: unreliable datagram
- UC: unreliable connection
- RC: reliable connection
RDMA requires specialized hardwares
Relevant RDMA operations

• Send/recv: Requires both local and remote CPU to join, mimics “connection”
• Read/write: Bypass remote CPU, read/write to remote memory directly.
• Atomic operations: not discussed today
InfiniBand and RoCE
Outline

• What is RDMA?
  • RDMA-accelerated KV-stores
• RDMA in database designs
• RDMA-optimized join operations
• Conclusion
In-Memory Key-Value Stores

• Interfaces: GET, PUT
• Requirements: low latency, high request rate
Related works

• Most related works come from System/networking conferences.
  • Pilaf [ATC’ 13]
  • Fast Remote Memory [NSDI’ 14]
  • General idea: Use of RDMA READs to bypass remote CPU.
  • Average cost: $\geq 1$ READ to get address, 1 READ to get value.
Related works

• Most related works come from System/networking conferences.
  • Pilaf [ATC’ 13]
  • Fast Remote Memory [NSDI’ 14]
  • HERD [SIGCOMM’14]
HERD general ideas

- Using RDMA WRITEs to write request to server, involve server CPU to send response (For lower latency).
- Optimization on low level verbs to increase throughput.
- Use datagram to increase scalability.
- The evaluations are comparing “peak” performance.
RDMA Write is faster than Read

Hardware Reason: PCI-e write is faster than read.
Increased throughput
Evaluation: Latency

48 byte items, GET intensive workload

Latency (microseconds) vs. Throughput (Mops)

- Emulated Pilaf
- Emulated FaRM-KV
- HERD

95th percentile
5th percentile

Low load, 3.4 μs
12 Mops, 8 μs
26 Mops, 5 μs
Evaluation: throughput
Lessons learned

• Traditionally, distributed systems view networking as a pipe, and send data generally via TCP/IP.
• With advances of RDMA, different solutions available.
• RDMA verbs should be selected according to the distributed systems requirement.
Outline

• What is RDMA?
• RDMA-accelerated KV-stores
• RDMA in database designs
• RDMA-optimized join operations
• Conclusion
Papers related

• The End of Slow Networks: It’s Time for a Redesign (Binnig et al, VLDB’ 16)
• The End of a Myth: Distributed Transactions Can Scale (Zamanian et al, VLDB’17)
• Accelerating Relational Databases by Leveraging Remote Memory and RDMA (Li et al, SIGMOD’ 16)
Motivating question

• With the changing features in network, should database systems be redesigned?

• Probably yes, previous systems built on the assumption that networking is the main bottleneck.

• To utilize the performance of high-speed network, redesign can be necessary.
Believed: Distributed transactions can’t scale

- CPU is busy dealing with TCP/IP stack, leaving little time for “real” work
- Increased contention likelihood
- IPoIB, a technology that adapts Infiniband devices to socket protocols brings heavy burden to CPU; performs bad when messages are small.
Figure 2: Network Throughput and Latency

Figure 3: CPU Overhead for Network Operations
Shared-nothing architecture

• Node has direct access to local memory, between nodes communicate with sockets.
• In real world, hard to avoid communications between nodes.
• Distributed join and transactions can be expensive.
• Substitute with IPoIB can lead to even worse performance! (as of previous page)
Shared-memory architecture

• Seems a natural solution to apply RDMA directly
• Remote memory is still different with local memory:
  • Cache coherence
  • Need to carefully design sharable memories
  • Local memory keeps only one copy of data, distributed memory creates one copy individually.
  • Garbage collection scheme, buffer management, consistency
Traditional architectures

• Shared-Nothing:
  • IPoIB cannot fully utilize the network.

• Shared-Memory:
  • Remote side can create a totally independent copy of memory.
  • Local and remote memory in one abstraction can be the wrong way.

• Proposed: Network-Attached Memory (NAM)
  • Separate local compute and store
NAM-DB

• Memory servers: holds all data of the database system
  • States for transaction execution
  • Provides memory capacity, handles remote memory allocation calls.

• Compute servers: actually executes the transaction.
Example: distributed transactions

• Traditionally, distributed transactions can not scale.
  • Limit of network bandwidth.
  • TCP/IP occupies CPU
    • High latency $\rightarrow$ High contention likelihood, timeout.

• With RDMA, these problems can be mitigated,

• Problem now: Global timestamps have scalability issue

• Skip for time reason, will present in the report.
NAM-DB scales well
Another question

• In relational databases, what to use when local memory is insufficient?
  • Local disk? (Much slower access speed)
  • Remote disk? (Made possible by RDMA)
What can remote memory bring?

• Extending the caches
• Spilling temporary data
• In-memory semantic caching
• Priming the buffer pool
Design

• Each server in the cluster reports the available memory.

• A database server with unmet memory demand can request for memory.

• Memory broker is an “arbiter” handling these allocations
Brief evaluation results

Figure 3: I/O micro benchmark throughput.

Figure 4: I/O micro benchmark latency.

Figure 5: I/O performance with multiple memory servers.

Figure 6: I/O performance with multiple database servers.
Lessons learned

• High speed network technologies can break some “assumptions” made as common belief.
  • Networking is bottleneck?
  • Distributed transactions can bring high burden to CPU?
  • Can we access Remote memory?
More in the report

• Comparison of sort-merge JOIN and hash JOIN in RDMA.
• RDMA optimized shuffle operator.
Conclusion

• Network has long been assumed to be the bottleneck in distributed database design and implementation.
• Recent development of RDMA seems to be removing this assumption.
• It is non-trivial task to apply RDMA to distributed database.
• Partial re-design is often required.
More on RDMA applications

• Explore RDMA on TensorFlow

• [https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/gdr](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/gdr) or search tensorflow contrib/gdr on google

• Towards Zero Copy Dataflows using RDMA, Bairen Yi, Jiacheng Xia, Li Chen and Kai Chen, Sigcomm posters and demos’ 17
More on RDMA devices