High-Speed Query Processing over High-Speed Networks

- Aboli
Introduction

- Networks - no longer a bottleneck (InfiniBand TCP, RDMA)
- Only increasing bandwidth - not much benefit
- Distributed query engines should adapt

Figure 3: Simply increasing the network bandwidth is not enough; a novel RDMA-based communication multiplexer is required (HyPer, TPC-H, SF 100)
High Speed Networks

- InfiniBand
  (High bandwidth, low latency cluster connect)

- Large amount of data shuffled during joins and aggregations

- Tune existing protocol for analytical workloads
Tuning TCP

- Use data direct I/O

(a) Classic I/O involves three memory trips at sender/receiver

(b) Data direct I/O reduces this to only one memory trip each

Figure 4: Data direct I/O significantly reduces the memory bus traffic for TCP compared to classic I/O
Tuning RDMA

- Asynchronous operation
  - Infinibands verbs API - Asynchronous
  - Work requests handled async via queue

- Kernel Bypassing
  - No copying between buffers
    -> no sys calls
  - Memory regions
    -> map virtual to physical
  - Message pools
Tuning RDMA

- Channel semantics
  - Read/write needs to have memory key
  - 2 sided operations
  - No separate exchange of memory keys

- Event notification
  - Polling vs Interrupt
  - CPU use vs latency
High Speed Queries

CLASSIC EXCHANGE OPERATORS

- Introduced in Volcano - Goetz Graefe (UW-Madison)
  
  Paper link

- Allows parallel query evaluation (Exchange operator)

- Parent process consumes data from child process
  (Bushy parallelism - diff subtrees, Intra-operator - same operator on diff data)

- Threads execute parallel copies - communicate via exchange operator
Issues with classical exchange operator

- All parallel units are same - local or remote
- Every exchange operator talks to other
  - \((n \times t) - 1\) for \(n\) servers and \(t\) local exchange operators per server
- Limits use of broadcast join
- Many connections required - scalability issues

ANSWER - **Hybrid Parallelism**

- Decoupled exchange operator
- RDMA based, NUMA aware multiplexer
- Application level Network Scheduling
Decoupled Exchange Operators

- All parallel units only interact with the multiplexer
  - Minimizes the number of connections

- Locally units can steal work
  - Better load balancing

- Further optimizations
  - Efficient serialization/deserialization
  - Unnecessary columns pruned
NUMA

- Every CPU - local memory controller
- Access remote memory via QPI - slower, expensive

Query engine - CPUs must access local memory addresses as much as possible
RDMA-based, NUMA aware multiplexer

- Network thread - exchange between local and remote servers via RDMA
- Multiplexers connected together
- Maintains message pools - registered with HCA for RDMA
- One receive queue per NUMA socket
  - Work stealing - NUMA local empty -> take from remote

Figure 7: Interaction of decoupled exchange operators with the RDMA-based, NUMA-aware multiplexer
Application level network scheduling

- All to all traffic - switch contention
- Round robin algorithm
  - Send and receive from one server in each phase

(a) Round-robin scheduling with conflict-free phases; three phases for four servers
(b) Application-level network scheduling improves throughput by up to 40%
Evaluations

- HyPer - in-memory database, columnar storage
- TPC-H queries for different servers
Evaluations

- Distributed SQL systems comparison
- Spark SQL, Impala, MemSQL, Vectorwise

(a) Queries per hour for each distributed SQL system
(b) Impact of network bandwidth on TPC-H performance

Figure 12: Comparing distributed analytical SQL systems for the TPC-H benchmark (6 servers, SF 100)
Concluding Remarks

- Full fledged query engine based on RDMA - good approach
- Implementation on HyPer - other databases?
THOUGHTS?