# Skew-resilient Query Processing for Fast Networks

**(Extended Abstract)**

Tobias Ziegler[1] Carsten Binnig[1] Uwe Röhm[2]

## 1 Introduction

**Motivation:** Scalable distributed in-memory databases are at the core of data-intensive computation. Although scaling-out solutions help to handle large amounts of data, more nodes do not necessarily lead to improved query performance. In fact, recent papers have shown that performance can even degrade when scaling out due to higher communication overhead (e.g., shuffling data across nodes) and limited bandwidth [Rö15]. Thus, current distributed database systems are built with the assumption that the network is the major bottleneck [BH13] and should be avoided at all costs.

In recent years, high-speed networks (e.g., InfiniBand (IB)) with a bandwidth close to the local memory bus [Bi16] have become economically viable. These network technologies provide Remote Direct Memory Access (RDMA) to allow direct memory access to a remote host and also reduce the latency of data transfer through bypassing the remote's CPU [In17, Gr10]. Therefore, the assumption that the network is the bottleneck no longer holds.

Consequently, recent research has focused on integrating RDMA-enabled high-speed networks into existing database systems designed along a *Shared-Nothing Architecture (SN)* [Rö16, LYB17]. This architecture co-locates computation and data to reduce the communication overhead in a cluster. Although combining a SN with IB's higher network bandwidth enables scalability to a certain extent, this approach fails if the data or workload is skewed and cannot be evenly partitioned. The root cause is that classical query execution schemes assume that each partition is processed by one node. Since nodes with larger partitions must process more data, they may become a bottleneck and hinder the overall scalability. In consequence, only utilizing the higher bandwidth without adapting the database architecture and query execution, does not automatically lead to improved scalability [Bi16].

**Contributions:** In this paper, we present a new approach to execute distributed queries on fast networks with RDMA. Our main contribution is a novel execution strategy, which enables collaborative query processing by remote work stealing to mitigate skew, as this is a common issues that hinders scalable query execution [WDJ91, Ly88]. Moreover, we implement this execution strategy in our prototype engine I-Store and show that it introduces almost no overhead to handle skew.

[1] TU Darmstadt, Data Management Lab - Informatik, Germany, firstname.lastname@cs.tu-darmstadt.de
[2] University of Sydney, School of Computer Science, Australia, uwe.roehm@sydney.edu.au

doi:10.18420/btw2019-ws-06

## 2 System Overview

I-Store builds upon an architecture specifically designed along fast networks — called the network-attached-memory (NAM) architecture [Bi16, Sa17]. The NAM architecture logically decouples *compute nodes* from *storage nodes* and uses RDMA for communication between all nodes as shown in Figure 1. The idea is that storage nodes provide a shared distributed memory pool that holds all the data and auxiliary data structures, which can be accessed via one-sided RDMA from all compute servers. In contrast to the traditional SN database architecture which physically co-locates the query execution with the storage location, the NAM architecture separates them. Due to the separation of compute and storage servers, computation can be executed independently of its storage location. Thus the computation is less sensitive to workload skew, since in case of a straggling compute server any other compute server can help. Therefore, in the NAM architecture data locality is not a hard requirement but only a tuning parameter that can be added to speed-up workloads.

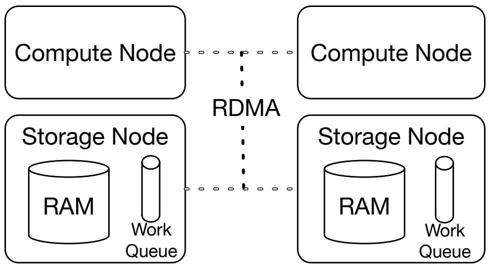

Fig. 1: The NAM Architecture

Our skew-resilient execution engine I-Store relies on fine-grained work elements that are stored inside *work queues* as depicted in Figure 1. A query execution is broken down into multiple work elements, similar to the morsel-driven execution on single node database systems as proposed in [Le14]. The queue-based execution, in combination with the possibility to access every data item via RDMA, allows load balancing by *work stealing*. However, not every data access between a compute and a storage node needs to be via RDMA. If we allow the (logical) compute and storage nodes to be co-located on the same (physical) cluster node, I-Store can use local memory access instead of RDMA for all local available data.

## 3 Remote Work Stealing

**Queue-based Query Execution:** I-Store implements a *queue-based query execution* strategy that allows fined-grained execution by organizing work in smaller chunks, namely *work elements*. A work element encodes the operation (e.g., an operator) and on which part of the data the operation is executed. The work elements are then stored in work queues, which are placed on the storage nodes as shown in Figure 1. Each work queue manages work elements which belong to the same partition as indicated in Figure 2. In order to process a query, a compute node is initially assigned to one work queue, i.e., to one partition. A compute node pops the work elements sequentially from its respective queue. Based on the information in the work element, a compute node processes the specified data pages. Once the assigned queue of a compute node is empty (i.e., if all work elements have been processed), this node starts stealing work elements remotely from other straggling compute nodes (i.e., from their work queues).

**NAM-Partitioning:** A problem of remote work stealing is that multiple compute nodes may try to steal data from the same straggling compute node, which would cause the available bandwidth of the storage node to be shared among them. To achieve a more balanced network usage, I-Store implements a novel partitioning scheme, called *NAM partitioning*, which distributes data equally among all storage nodes independent of the data distribution. In NAM partitioning, a partition is split into many small *pages*. These pages are then distributed evenly in a round-robin fashion to all storage nodes. To maintain a logical partition, the pages are linked together via a remote pointer to form a distributed linked list of pages, as indicated in Figure 2. To avoid pointer chasing, I-Store implements a *prefetching* mechanism: Using the remote pointer from an already fetched data page, I-Store can overlay computation with data retrieval by exploiting the RDMA-network card as a co-processor to prefetch the next page.

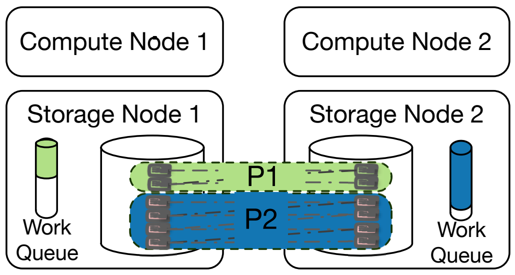

Fig. 2: The NAM-Partitioning

## 4 Experimental Evaluation

In the following, we present the results of a small experimental performance evaluation of I-Store to validate the performance benefits of work stealing and NAM-partitioning. The evaluation was conducted on a four-node cluster connected via a single InfiniBand FDR 4X switch using one Mellanox Connect-IB card[3]. Each server had two Intel Xeon E5-2660 v2 processors (20 cores in total) and ran on Ubuntu 14.01 Server Edition (kernel 3.13.0-54-generic). I-Store was compiled using gcc 4.8.5.

To be able to assess workload skew, we generated two synthetical datasets consisting of four relations (A, B, C, D), one where the partition key is following a uniform distribution (i.e., all partitions have the same size), while the second dataset followed a Zipf distribution with $z = 1.25$ (i.e., one partition dominates the others in size). Each record in the dataset consisted of three attributes $(PK, payload, FK)$ similar to [Rö16], with a tuple width of 24 Bytes. In total, each table contained 420M records, which yields a total size of about 10 GB per table. The query workload consisted of SQL queries that execute three joins (i.e., A⋈B⋈C⋈D) with an additional selection on each inner relation (A⋈ $\sigma_X$(B)⋈ $\sigma_Y$(C)⋈ $\sigma_Z$(D)). We mainly concentrated on joins since these operations are widely used in many analytical SQL workloads and we thus can show the effects of or work stealing algorithms for a wide class of analytical SQL queries.

We assess the runtime of this workload on two system architectures: The baseline is the *shared-nothing* architecture with co-partitioned (A,B) tables to minimize network transfers, while I-Store implements a *NAM architecture*. We used the same cluster with four physical nodes for this experiment. We configured I-Store to co-locate one compute and one storage

[3] Theoretical bandwidth of 6.8 GB/s per incoming and outgoing link

node on each single physical node. We measured I-Store in four different configurations: Plain I-Store without further optimizations, with work-stealing (WS), with NAM-partitioning (NAM-Part), and with local-access optimization (LocOpt) enabled (i.e., data accesses do not use RDMA but local memory accesses). For this paper, work stealing was done on the granularity of the selection operators (scan and pre-filtering of data pages).

As expected, the uniform dataset is the ideal case for shared-nothing, however I-Store with all optimizations shows a similar performance (1220 ms vs 1251 ms). Interestingly, work stealing can improve query execution even for homogeneous clusters: I-Store WS was 5% faster than plain I-Store. This shows that even if the dataset follows a uniform distribution, it can happen that individual nodes become slower than others due to external factors, for example in a shared experiment cluster like ours. NAM-partitioning (I-Store WS+NAM-Part) further decreases the runtime by another 5% since it balances network access among storage nodes and avoids delays due to network congestion. The last optimization leverages local-access (I-Store with LocOpt) and performs similarly as the baseline.

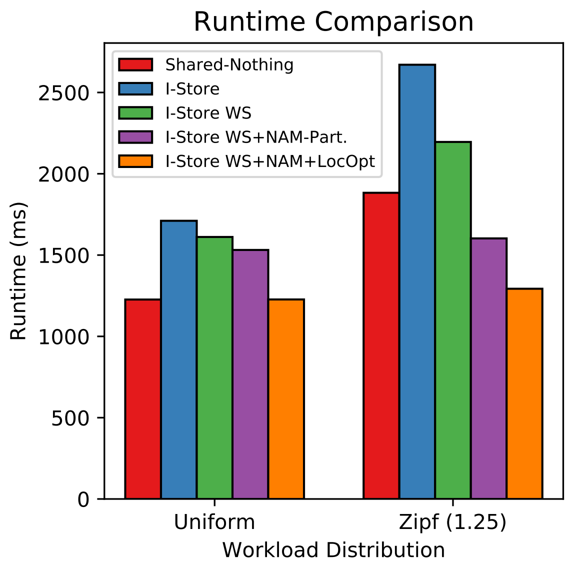

Fig. 3: Performance of Different Execution Strategies on Uniform and Skewed Data

For the skewed distribution, the runtime of shared-nothing (1917 ms) was dominated by the slowest node which needed to process 4.1 GB per partition. I-Store without any optimizations takes the longest to finish (2699 ms), but if work-stealing is enabled the runtime is close to our baseline. With NAM-Partitioning enabled, I-Store outperforms shared-nothing and shows the effect of network congestion. The runtime with NAM-partitioning compared to the vanilla work stealing approach is reduced by a 35%. In both distributions I-Store with all optimizations performs best. Additionally, the overhead induced by the skewed workload is only around 60 ms compared to the uniform execution.

## 5 Conclusion

This paper explored techniques to better align query execution with direct memory access over high-speed networks. We presented I-Store, a novel queue-based query execution engine that efficiently supports load balancing via NAM-aware data partitioning and work stealing. In a short evaluation we showed that I-Store can handle skew with almost no overhead. As an avenue of future work, we plan to implement different work stealing strategies and show that our work stealing approach is applicable to a variety of operators.

## References

[BH13] Babu, Shivnath; Herodotou, Herodotos: Massively Parallel Databases and MapReduce Systems. Found. Trends databases, 5(1):1–104, November 2013.

[Bi16] Binnig, Carsten; Crotty, Andrew; Galakatos, Alex; Kraska, Tim; Zamanian, Erfan: The End of Slow Networks: It's Time for a Redesign. Proc. VLDB Endow., 9(7):528–539, March 2016.

[Gr10] Grun, Paul: Introduction to infiniband for end users. White paper, InfiniBand® Trade Association (IBTA), 2010.

[In17] InfiniBand® Trade Association (IBTA): , Infiniband Roadmap. `http://www.infinibandta.org/content/pages.php?pg=technology_overview`, 2017. Accessed: 2017-10-19.

[Le14] Leis, Viktor et al.: Morsel-driven parallelism: a NUMA-aware query evaluation framework in the many-core age. In: ACM SIGMOD. 2014.

[Ly88] Lynch, Clifford A.: Selectivity Estimation and Query Optimization in Large Databases with Highly Skewed Distribution of Column Values. In: Proceedings of the 14th VLDB. VLDB '88, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 240–251, 1988.

[LYB17] Liu, Feilong; Yin, Lingyan; Blanas, Spyros: Design and Evaluation of an RDMA-aware Data Shuffling Operator for Parallel Database Systems. Proceedings of the Twelfth European Conference on Computer Systems - EuroSys '17, pp. 48–63, 2017.

[Rö15] Rödiger, Wolf; Mühlbauer, Tobias; Kemper, Alfons; Neumann, Thomas: High-speed Query Processing over High-speed Networks. Proc. VLDB Endow., 9(4):228–239, dec 2015.

[Rö16] Rödiger, Wolf; Idicula, Sam; Kemper, Alfons; Neumann, Thomas: Flow-Join: Adaptive skew handling for distributed joins over high-speed networks. 2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016, pp. 1194–1205, 2016.

[Sa17] Salama, Abdallah; Binnig, Carsten; Kraska, Tim; Scherp, Ansgar; Ziegler, Tobias: Rethinking Distributed Query Execution on High-Speed Networks. IEEE Data Eng. Bull., 40(1):27–37, 2017.

[WDJ91] Walton, Christopher B.; Dale, Alfred G.; Jenevein, Roy M.: A Taxonomy and Performance Model of Data Skew Effects in Parallel Joins. In: Proceedings of the 17th VLD. VLDB '91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 537–548, 1991.