Eliminating the Bandwidth Bottleneck of Central Query Dispatching Through TCP Connection Hand-Over

Stefan Klauck[1], Max Plauth[1], Sven Knebel[1], Marius Strobl[2], Douglas Santry[2], Lars Eggert[2]

[1] Hasso Plattner Institute, University of Potsdam, Germany

[2] NetApp

March, 2019

Motivation

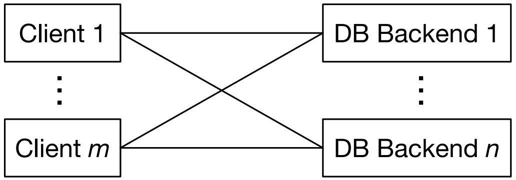
Hasso Plattner Institut

In scale-out database systems, queries must be routed to individual servers.

Motivation

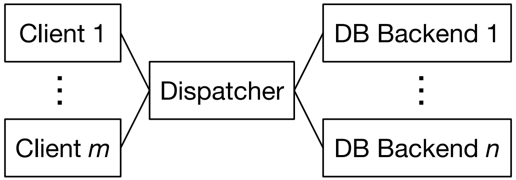
In scale-out database systems, queries must be routed to individual servers.

Direct Communication



- \+ Latency
- \- Requires smart clients or static backends

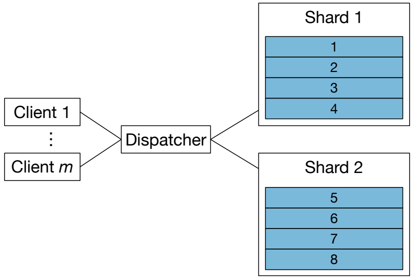
Central Dispatcher



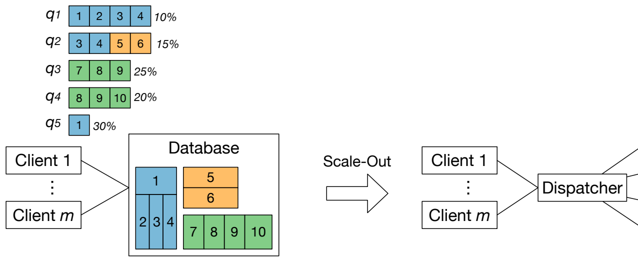
- \+ Simple clients / dynamic backends
- \- Central dispatcher is potential bottleneck

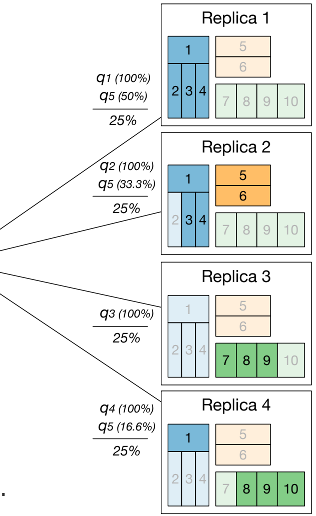
Motivation – Use Cases for Central Dispatching

- Horizontal Partitioning / Sharded Database



- Partially Replicated Database System
  - Maximize throughput
    by balancing the load evenly
    while minimizing memory footprint



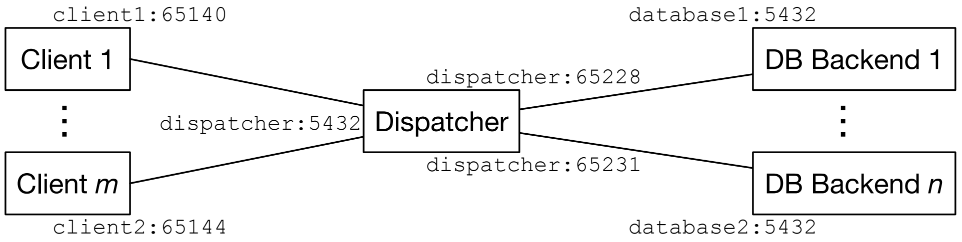


Rabl et Jacobsen. Query Centric Partitioning and Allocation for Partially Replicated Database Systems. SIGMOD 2017.
Klauck et Schlosser. Workload-Driven Fragment Allocation for Partially Replicated Databases Using Linear Programming. ICDE 2019.

Motivation – Central Dispatching from a Network Perspective

```
>>> import psycopg2
>>> conn = psycopg2.connect("dbname='tpch' host='dispatcher'")
```

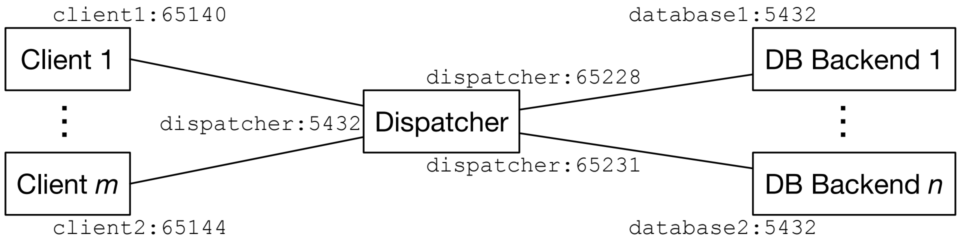
- Logical view



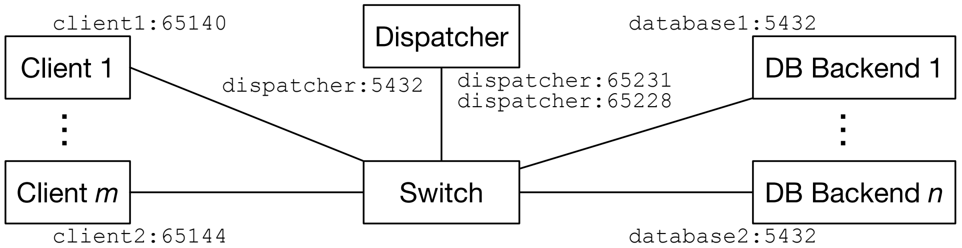
Motivation – Central Dispatching from a Network Perspective

```
>>> import psycopg2
>>> conn = psycopg2.connect("dbname='tpch' host='dispatcher'”)
```

- Logical view

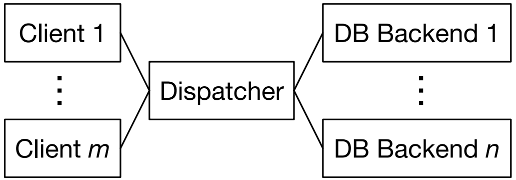


- Physical view



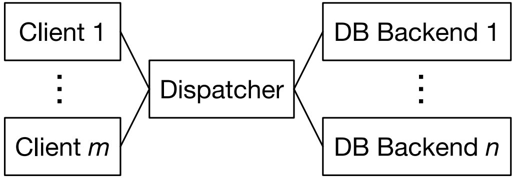
Motivation

- Whether the dispatcher becomes a bottleneck depends on the workload
  - Number and size of queries/messages
  - Ratio of processed tuples and result set size



Motivation

- Whether the dispatcher becomes a bottleneck depends on the workload
  - Number and size of queries/messages
  - Ratio of processed tuples and result set size



- "Transferring a large amount of data out of a database system to a client program is a common task."

Raasveldt et Mühleisen. Don't Hold My Data Hostage – A Case For Client Protocol Redesign. VLDB 2017.

  - Needed for statistical analyses or machine learning in clients
  - Main bottleneck is *network bandwidth*

Research Goals

- Integration of a TCP connection hand-over by means of a reprogrammable network switch into a database

- Comparison of query-based dispatching approaches in terms of
  - Throughput scaling
  - Processing flexibility

Dispatcher Implementations

- Traditional architecture with two separate TCP connections:

  client ←→ dispatcher ←→ database

  1. **HAProxy** – free and open source TCP/HTTP load balancer
  2. **Hyrise dispatcher**

https://github.com/hyrise

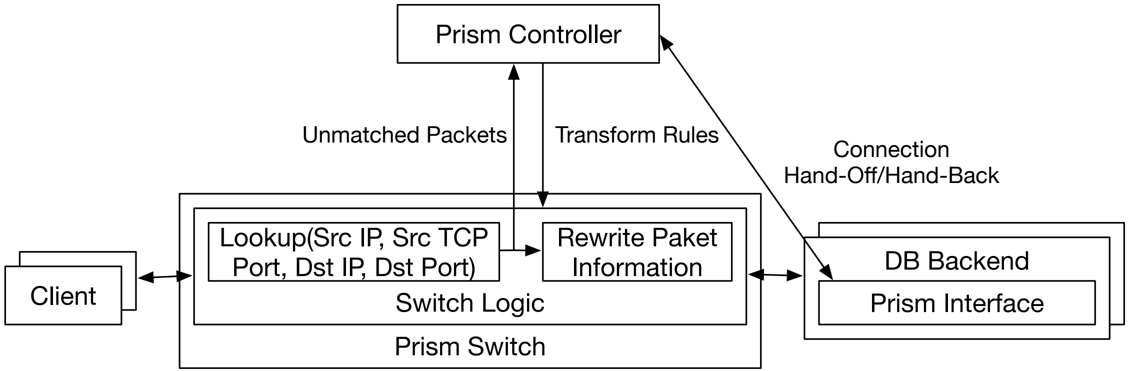
- Using a reprogrammable switch to perform TCP connection hand-over

  3. **Prism**: exchange most packets directly between client and backend

Y. Hayakawa et al. Prism: A Proxy Architecture for Datacenter Networks. SoCC 2017.

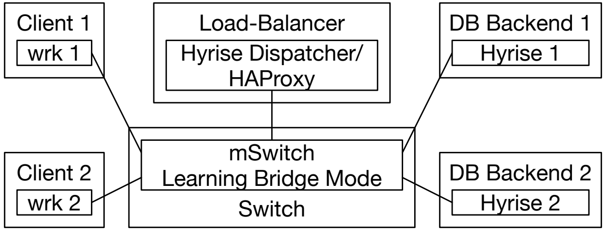
Dispatcher Implementations - Prism

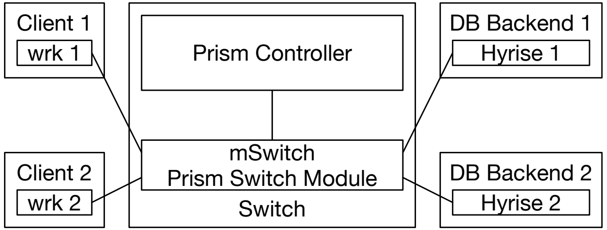
- Client query is initially sent/routed to Prism Controller
- Prism Controller **forwards connection** to an appropriate backend and **reprograms the switch**
- Backend processes query and **sends result directly to the client** (bypassing the Prism Controller)
- Backend hands back connection to Prism Controller



Experimental Evaluation

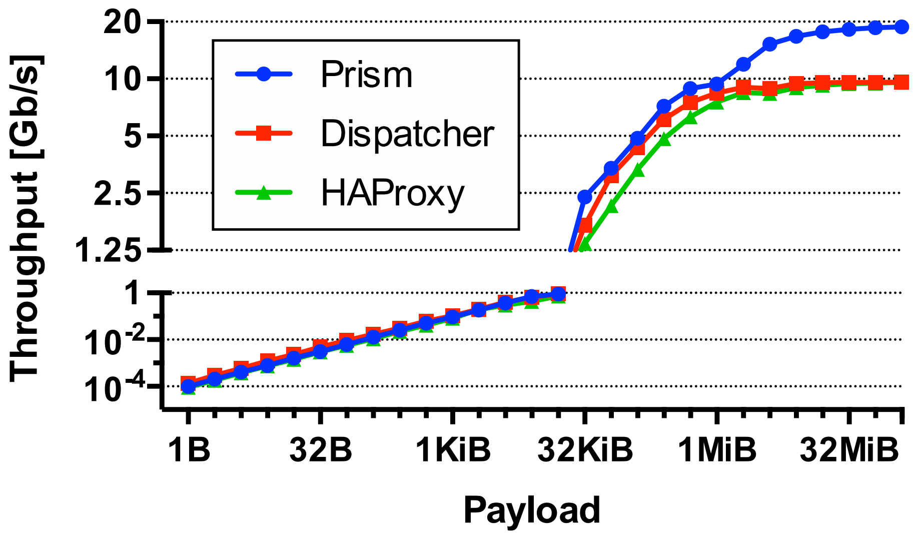
- 10Gb and 40Gb Ethernet experiments
  - Hyrise with a stored procedure — https://github.com/hyrise
  - wrk - HTTP benchmarking tool — https://github.com/wg/wrk
  - mSwitch - software switch — Honda et al. mSwitch: A Highly-Scalable, Modular Software Switch. SOSR 2015.





Experimental Evaluation with two Clients and Backends

- 10 GbE results



← scales up to bandwidth: min(Σ clients, Σ backends)

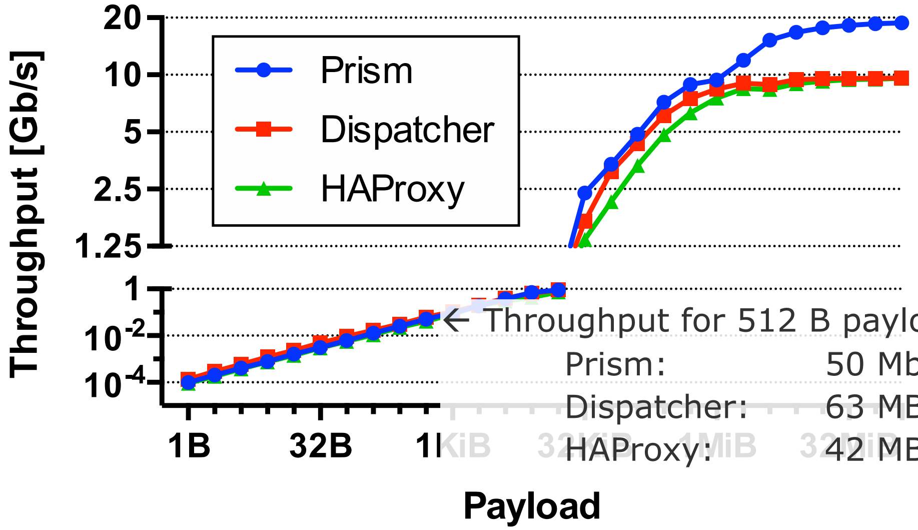
← limited by bandwidth of central dispatcher

  - Using TCP hand-over outperforms traditional approaches for large payloads

Experimental Evaluation with two Clients and Backends

Hasso Plattner Institut

- 10 GbE results



← scales up to bandwidth: min(Σ clients, Σ backends)

← limited by bandwidth of central dispatcher

← Throughput for 512 B payload

Prism:	50 Mb/s
Dispatcher:	63 MB/s
HAProxy:	42 MB/s

  - Using TCP hand-over outperforms traditional approaches for large payloads
  - Hyrise dispatcher performs best for small payload sizes up to 4kB

Other Uses of Software Defined Networking in Databases

HPI Hasso Plattner Institut

- Implement transaction ordering inside the network switch (published @ SOSP 2017)
- Offload full SQL query segments onto a programmable dataplane (published @ CIDR 2019)

Eris: Coordination-Free Consistent Transactions Using In-Network Concurrency Control

Jialin Li, University of Washington, lijl@cs.washington.edu
Ellis Michael, University of Washington, emichael@cs.washington.edu
Dan R. K. Ports, University of Washington, drkp@cs.washington.edu

ABSTRACT

Distributed storage systems aim to provide strong consistency and isolation guarantees on an architecture that is partitioned across multiple shards for scalability and replicated for fault tolerance. Traditionally, achieving all of these goals has required an expensive combination of atomic commitment and replication protocols – introducing extensive coordination overhead. Our system, Eris, takes a different approach. It moves a core piece of concurrency control functionality, which we term *multi-sequencing*, into the datacenter network itself. This network primitive takes on the responsibility for consistently ordering transactions, and a new lightweight transaction protocol ensures atomicity.

The end result is that Eris avoids both replication and transaction coordination overhead: we show that it can process a large class of distributed transactions *in a single round-trip* from the client to the storage system *without any explicit coordination* between shards or replicas in the normal case. It provides atomicity, consistency, and fault tolerance with less than 10% overhead – achieving throughput 3.6–35× higher and latency 72–80% lower than a conventional design on standard benchmarks.

CCS CONCEPTS

• **Information systems → Database transaction process-**

KEYWORDS

distributed transactions, in-network concurrency control, network multi-sequencing

ACM Reference Format:
Jialin Li, Ellis Michael, and Dan R. K. Ports. 2017. Eris: Coordination-Free Consistent Transactions Using In-Network Concurrency Control. In *Proceedings of SOSP '17, Shanghai, China, October 28, 2017,* 17 pages.
https://doi.org/10.1145/3132747.3132751

1 INTRODUCTION

Distributed storage systems today face a tension between transactional semantics and performance. To meet the demands of large-scale applications, these storage systems must be partitioned for scalability and replicated for availability. Supporting strong consistency and strict serializability would give the system the same semantics as a single system executing each transaction in isolation – freeing programmers from the need to reason about consistency and concurrency. Unfortunately, doing so is often at odds with the performance requirements of modern applications, which demand not just high scalability but also tight latency bounds. Interactive applications now require contacting hundreds or thousands of individual storage services on each request, potentially leaving individual transactions with sub-millisecond latency bud-

The Case for Network-Accelerated Query Processing

Alberto Lerner Rana Hussein Philippe Cudre-Mauroux
eXascale Infolab, U. of Fribourg—Switzerland

ABSTRACT

The fastest plans in MPP databases are usually those with the least amount of data movement across nodes, as data is not processed while in transit. The network switches that connect MPP nodes are hard-wired to perform packet-forwarding logic only. However, in a recent paradigm shift, network devices are becoming "programmable." The quotes here are cautionary. Switches are not becoming general purpose computers (just yet). But now the set of tasks they can perform can be encoded in software.

In this paper we explore this programmability to accelerate OLAP queries. We determined that we can offload onto the switch some very common and expensive query patterns. Thus, for the first time, moving data through networking equipment can contribute to query execution. Our preliminary results show that we can improve response times on even the best agreed upon plans by more than 2x using 25 Gbps networks. We also see the promise of linear performance improvement with faster speeds. The use of programmable switches can open new possibilities of architecting rack- and datacenter-sized database systems, with implications across the stack.

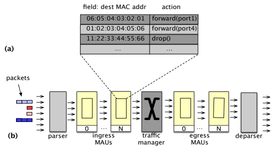


Figure 1: (a) A match-action table programmed to forward or to drop a packet according to its destination MAC address. (b) Architecture of a programmable switch dataplane holding that table.

with a row in this table using, for instance, exact matching. Other types of matches are also possible. The *action* engine executes simple instructions over a packet or table data. Examples of such instructions are simple arithmetic or moving data within a packet. The MAU is programmable in the sense that one can specify its table layout, the type of lookup to perform, and the processing done at a match event, as we illustrate in Figure 1(a). We say that a MAU

1. INTRODUCTION

Networking is an area in constant evolution. New protocols keep arising from emerging fields such as virtualization [20], cloud computing [6], or the Internet-of-Things [27]

Summary

- Scale-out database systems use central query dispatchers to hide backend complexity, but may be a bandwidth bottleneck

- We compared dispatching architectures for database systems
  - Traditional dispatcher performs best for small payload sizes
  - Prism's connection overhead pays off for larger payloads

-> Hybrid approach with on-demand connection hand-over for large results

IT Systems Engineering | Universität Potsdam

Thanks

Stefan Klauck

stefan.klauck@hpi.de

http://epic.hpi.de