SwitchTx: Scalable In-Network Coordination for Distributed Transaction Processing

Junru Li¹, Youyou Lu¹, Yiming Zhang², Qing Wang¹, Zhuo Cheng³, Keji Huang³, Jiwu Shu¹

¹Tsinghua University
http://storage.cs.tsinghua.edu.cn/

²Xiamen University

³Huawei
Coordination in Distributed Transactions

- Network communication is a major source of coordination cost
  - Concurrency control protocols
  - Replication protocols
- Leveraging the high-speed network
  - Reduce latency
  - Shorten contention span to reduce abort rate

However, with the high-speed network, the coordination cost is still significant
Coordination Cost I

- With high-speed network, the coordination cost is still significant

Waste CPU to process coordination packets

- Waste CPU cycles
- CPU processing latency is more important with a faster network
Coordination Cost II

- With high-speed network, the coordination cost is still significant

**Semantic gap between Txn apps and network**

- Inappropriate processing order introduces extra aborts
- Redundant flow control algorithms interfere with each other
  - Admission control: controls the number of concurrent transactions
  - Congestion control: controls the number of concurrent network messages

Situation ① > ②: Client<sub>2</sub> locks A successfully
Situation ② > ①: Client<sub>2</sub> needs to retry to lock A
Opportunities from Programmable Switches

Programmable Switches

- Centralized hub
- User-defined parsers / Match-Action tables / queues
- On-chip memory
- Line-rate processing

Transaction-defined protocol

Low-overhead
Design Goals and Challenges

**Design Goals:** reduce coordination cost
- Offload coordination tasks
- Manipulate transaction traffic intelligently

**Challenges:**
- Restricted expressive power and limited on-chip memory
- Multi-switch scalability

Eris (SOSP’17)
Outline

- Background & Motivation
- SwitchTx: In-Network Transaction Coordination
- Results
- Summary
Overview

In-Network Transaction Coordination

Server 0

Worker Threads

client: Exec Txns
participant: Exec Remote Ops

Memory
primary backup buffer

Server 1

primary backup buffer

Programmable Switches

Server 2

primary backup buffer

client
participant

client
participant
Key Design

To save CPU utilization

1.1 Coordination tasks → in-switch Gather-and-Scatter (GaS)
1.2 Scalable tree-based GaS using all switches

To break the semantic gap between Txn apps and network

2.1 Semantic-aware packet priority control
2.2 Dynamic admission control
1.1) In-switch Gather-and-Scatter

- Offload coordination tasks as in-switch GaS
- An example: Txn \{ read[D_0,D_2], write[D_1,D_2] \}
1.1) In-switch Gather-and-Scatter

- **GaS (gather_group, scatter_group)**
  - Gather messages from the participants of the current phase
  - Scatter the result to the participants of the next phase
1.1) In-switch Gather-and-Scatter
1.2) Scalable Tree-based Gather-and-Scatter

- **Gather-and-Scatter tree**
  - Servers: leaf nodes
  - Switches: non-leaf nodes

![Diagram of a Scalable Tree-based Gather-and-Scatter](image-url)
2 Break the semantic gap

Client

Admission Control

To reduce both transaction aborts and network congestion

Switch

Performance Monitoring

Priority Control

Server

To shorten contention spans and reduce unnecessary abort
2.1) Semantic-aware Packet Priority Control

- Assign priorities to messages based on their types
  - **Highest:** lock releasing + messages of retrying transactions
  - **Lowest:** lock acquiring
  - **Medium:** other messages

- Implementation
  - Priority queues in switches
  - Batch-based reordering in servers
2.2) Dynamic Admission Control

- Increasing maximum number of parallel requests
  - Higher resource utilization 😊
  - Higher abort rate 😞
  - Network congestion 😞

- Signals
  - Global performance metrics
  - Individual network conditions

- Algorithm: AIMD
  - Additive increase
  - Multiplicative decrease
More Details: checkout our paper

❖ Other design details
  ❖ How to map GaS operations to Match-Action tables
  ❖ How to select switches to form the GaS tree
  ❖ How to handle packet loss and packet out-of-order
  ❖ How to handle server or switch failure
  ❖ The practicality of SwitchTx

❖ Implementation details
  ❖ Packet formats
  ❖ RMDA RAW_PACKET verbs for control messages
  ❖ RDMA WRITE_WITH_IMM verbs for data messages
  ❖ …
Outline

- Background & Motivation
- SwitchTx: In-Network Transaction Coordination
- Results
- Summary
## Experimental Setup

### Hardware Platform

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>8x Servers</td>
</tr>
<tr>
<td>CPU</td>
<td>2x Intel 12-core Xeon E5-2650 CPUs</td>
</tr>
<tr>
<td>NIC</td>
<td>100Gbps Mellanox ConnectX-5</td>
</tr>
<tr>
<td>Switch</td>
<td>Barefoot Tofino Wedge 100BF-32X (bf-sde-8.8.1)</td>
</tr>
</tbody>
</table>

### Competitors

<table>
<thead>
<tr>
<th>Competitor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SwitchTx</td>
<td>OCC + Primary-backup replication, scalable in-network coordination</td>
</tr>
<tr>
<td>FaSST [OSDI’16]</td>
<td>OCC + Primary-backup replication</td>
</tr>
<tr>
<td>Eris [SOSP’17]</td>
<td>Independent deterministic transaction, centralized in-network sequencer</td>
</tr>
</tbody>
</table>

Others: Aria [VLDB’20], Calvin [SIGMOD’12] (check our paper)

### Benchmarks: TPC-C, YCSB-T
Overall Performance

8 nodes, 24 threads per node

YCSB-T: a transaction reads/writes (50%:50%) 8 records, each record has an 8-byte key and a 16-byte value

TPC-C: 50% New-Order + 50% Payment

SwitchTx can boost the performance of distributed transactions
Scalability

8 nodes
1~24 threads per node

In-switch Gather-and-Scatter is scalable
Saved CPU Resources and Packets

YCSB-T Benchmark

SwitchTx effectively saves CPU resources and reduces network traffic
Limitation

YCSB-T: varying the number of shards accessed by each transaction
TPC-C: varying the % of remote items for New-Order transaction

SwitchTx is suitable for the transactions cross many shards
Outline

- Background & Motivation
- SwitchTx: In-Network Transaction Coordination
- Results
- Summary
Summary

❖ Goal
   ❖ Reduce coordination cost in distributed transaction processing systems

❖ Key Idea
   ❖ Using programmable switches to offload coordination tasks and manipulate transaction traffic intelligently

❖ Techniques in SwitchTx
   ❖ Scalable in-network Gather-and-Scatter
   ❖ Priority control and dynamic admission control

❖ Results
   ❖ SwitchTx outperforms state-of-the-arts
   ❖ SwitchTx is scalable to multiple switches
Thanks

SwitchTx: Scalable In-Network Coordination for Distributed Transaction Processing

Contact Information: Junru Li (http://storage.cs.tsinghua.edu.cn/~ljr/), lijr19@mails.tsinghua.edu.cn