Performance-Optimal Read-only Transactions

Haonan Lu*
Siddhartha Sen⁺, Wyatt Lloyd*

*Princeton University, ⁺Microsoft Research
Distributed Storage Systems
Enable Today’s Web Services
Distributed Storage Systems

Reads Dominate Workloads

Jack's Page

Mia's Page

Friend Lists

Load Page

Reads

Writes

Storage
Distributed Storage Systems
Simple Reads Are Insufficient
Read-Only Transactions

• A group of simple reads sent in parallel

• Do not write data
  – Writes are allowed in the system

• Coordinate a consistent view across shards

Coordination overhead causes higher latency and lower throughput
Goal:
Read-only transaction performance as close as possible to simple reads
Goal:
Read-only transaction performance as close as possible to simple reads

We answer:

• What does optimal performance mean for read-only transactions?

• When is optimal performance achievable?

• How can we design performance-optimal read-only transactions?
Performance Factors
Engineering vs. Algorithmic

- Coordination
- Batching
- Networking
- Hardware

Algorithmic Properties
- Focus on the algorithmic properties due to coordination

Engineering Factors
- Equally impact simple reads and read-only transactions
- Abstract engineering factors by comparing to simple reads
Performance Factors
Algorithmic Properties

Simple Read
Simple Read

Blocking

Page
Friends

Algorithmic Properties
Performance Factors
Algorithmic Properties

Simple Read

Blocking

Messages

Simple Read

Page

Friends

Algorithmic Properties

Performance Factors
Performance Factors
Algorithmic Properties

Algorithmic Properties
- Blocking
- Messages
- Metadata

Simple Read

Simple Read

Timestamp

Page

Friends

Timestamp
Performance Factors
Coordination Is Algorithmic

Algorithmic Properties
Coordination Overhead

Simple Read
Simple Read
Read-Only Transactions
Optimal Performance

Algorithmic Properties

Blocking

N
Messages
O

Metadata
C

Performance-optimal
Read-only Transactions
(N,O,C)

Simple Read

Simple Read
Non-Blocking Reads

- Do not wait on external events
  - Distributed locks, timeouts, messages, etc.

- Lower latency
  - Avoid any time spent blocking

- Higher throughput
  - Avoid CPU cost of context switches
One-Round Communication

• One-round on-path reads
  – Succeed in one round, i.e., no retries

• No off-path messages
  – Required by reads but off the critical path

• Lower latency
  – Avoids time for extra on-path messages

• Higher throughput
  – Avoids CPU cost of processing extra messages
**Constant Metadata**

- **Metadata**
  - Information used to find a consistent view
  - Timestamps, transaction IDs, etc.

- **Size of metadata remains constant regardless of contention**

- **Higher throughput**
  - Avoids CPU cost of processing extra data
Performance-optimal read-only transactions are **NOC**: **N**on-blocking messages that complete in **O**ne-round with **C**onstant metadata
Strict Serializability

• The strongest consistency model
  – Writing applications made easy
• Requires a total order + real-time order
The NOCS Theorem:

**Impossible** for read-only transaction algorithms to achieve performance-optimality \([N,O,C]\) and strict serializability \([S]\)
Proof Intuition of NOCS

Svr-1: Coordination Free
Svr-2: Coordination Required
Svr-3: Coordination Free
Svr-4: Coordination Required

- Stable
- Unstable

Finalized Write
Unfinalized Write

now
Proof Intuition of NOCS

Must give up either N, O, or C

Svr-1
Svr-2
Svr-3
Svr-4
NOC Designs

By the NOCS Theorem

Our new design: PORT

MySQL Cluster

Weak Consistency

Strict Serializability

Process-order Serializability

Causal

Read Committed

Weak Consistency

Strong
Design Insight
Capturing the Stable Frontier

Svr-1

Svr-2

Svr-3

Svr-4

Stable Frontier (SF)

now

stable

unstable
Version Clock

• A type of logical clock
  – Specialized for distributed storage systems

• Treat reads and writes differently
  – Enable optimizations for reads and writes

• Capture the stable frontier
PORT Overview

Jack

Web Client

Storage Server
PORT Overview

Version Clock

Key A

\[ [A_X]_0 \ [A_Y]_1 \ [A_Z]_2 \]
PORT Overview

Version Stamp (VS)

Version Clock

Jack

Key A

\[ [A_0] \rightarrow [A_1] \rightarrow [A_2] \]

VS
Write in PORT

Jack

Write \[
\begin{cases}
A := A_Y \\
VS = 2
\end{cases}
\]

“Done”

Key A

\([A_X]_0 [A_Y]_2\)

Version clocks tick on writes
Read in Port

Read \{ \begin{align*}
A &= ? \\
VS &= 2
\end{align*} \}

A = A_Y

No tick on reads

Jack

Key A

[\[A_X\]]_0 [\[A_Y\]]_2 [\[A_Z\]]_5
Read Promotion
Ensures a Total Order

Key A

[A_x]_0 [ ? ]_2
Read Promotion
Ensures a Total Order

Jack

Read \{ 
\begin{align*}
A &= \text{?} \\
VS &= 2 \\
A &= A_x
\end{align*}
\}

[A_x]_0

Immutable

Key A
Read Promotion
Ensures a Total Order

Write \( A := A_Y \)
VS = 2

“Done”

Key A

\([A_X]_{0\rightarrow 2} \quad [A_Y]_3\)
Track Stable Frontier

SF Map

- **SF** = 3
- **SF\(_A\)** = 3
- **SF\(_B\)** = 3
- **SF\(_C\)** = 5

Advance to stable frontier

Mia

Read/Write

SF\(_A\) = 3

Key A

\([A_x]_{0\rightarrow2}\) \([A_x]_{3}\)
Read-Only Transaction Logic

SF Map
- SF = 3
- SF_A = 3
- SF_B = 3
- SF_C = 5

Jack

Read \{A = ?\}
VS = 3

Key A
- \[A_X\]_0 [A_Y]_3 [A_Z]_7

Read \{B = ?\}
VS = 3

Key B
- \[B_X\]_0 [B_Y]_1 [B_Z]_3
Read-Only Transaction Logic

SF Map
- SF = 3
- SF_A = 3
- SF_B = 3
- SF_C = 5

Jack

Read
\[
\begin{align*}
A &= ? \\
VS &= 3
\end{align*}
\]

Key A
- \([A_X]_0\)
- \([A_Y]_3\)
- \([A_Z]_7\)

Read
\[
\begin{align*}
B &= ? \\
VS &= 3
\end{align*}
\]

Key B
- \([B_X]_0\)
- \([B_Y]_1\)
- \([B_Z]_3\)
Read-Only Transaction Logic

SF Map
- SF = 3
- SF_A = 3
- SF_B = 3
- SF_C = 5

Jack

A = A_Y, SF_A = 7
B = B_Z, SF_B = 3

Key A

[A_X]_0 [A_Y]_3 [A_Z]_7

Key A

[B_X]_0 [B_Y]_1 [B_Z]_3
PORT Is NOC

- Reading at the stable frontier ensures reads are non-blocking (N)

- Client pre-determined snapshot with VS ensures one-round communication (O)

- One VS per read request ensure constant metadata (C)
PORT Systems

• Scylla-PORT
  – Base system: ScyllaDB (non-transactional)
    • Highly optimized → sensitive to overhead
  – NOC + Process-ordered serializability
  – Supports simple writes (not write transactions)

• Eiger-PORT
  – Base system: Eiger (N, ☒, ☒)
    • Existing read-only and write transactions
  – NOC + Causal consistency
  – Supports write transactions
Evaluation of Scylla-PORT

• To understand
  – Overhead in latency and throughput compared to simple reads
  – Performance advantages compared to other protocols, e.g., OCC.

• Experiment configuration
  – YCSB benchmark with customized parameters for skew and read-to-write ratios
  – Evaluated latency, throughput, scalability, freshness
Latency-Throughput
Uniform, 5% Writes

Average Latency (ms)

Throughput (K Txn/s)

Scylla-OCC
Scylla-PORT
ScyllaDB

Higher Throughput
Lower Latency
Latency-Throughput

Zipf = 0.99, 5% Writes

Average Latency (ms)

Throughput (K Txn/s)

Scylla-OCC
Scylla-PORT
ScyllaDB

8%
Conclusion

- Performance-optimal read-only transactions: NOC

- The NOCS Theorem for read-only transactions
  - Impossible to have all of the NOCS properties

- The design of PORT
  - NOC with the strongest consistency to date

- Scylla-PORT
  - Minimum performance overhead compared to simple reads
  - Significantly outperforms the standard OCC

Contact Information
Haonan Lu
haonanl@cs.princeton.edu