The **SNOW** Theorem and Latency-Optimal Read-Only Transactions

Haonan Lu*, Christopher Hodsdon*, Khiem Ngo*, Shuai Mu†, Wyatt Lloyd*

*University of Southern California and Princeton University
†New York University
Web Services Are Huge
Web Services Are Huge

2.5 B – content items shared
2.7 B – “likes”
300 M – photos uploaded
105 TB – data scanned
500 TB – new data ingested

Huge Web Services Shard Data

Massive amount of data

→ must be distributed across servers

**Reads** dominate the workloads
– need to be as fast as possible!
Simple Reads Are Insufficient
Read-Only Transactions

• Transactions that do not modify data

• Consistently read data across servers
The Power of Read-Only Txn

• Consistency restricts what can be read
  – Eliminates unacceptable combinations

• Compatibility enables write transactions
  – Write transactions atomically update data

• Higher power → more useful
  – Stronger consistency → higher power
  – Compatibility → higher power
Fundamental Tradeoff

High Power vs Low Latency

- Reduces anomalies (the ACL – Photo example)
- Easier to reason about

- Better user experience
- Higher revenue

Our study proves: highest power + lowest latency is impossible
The SNOW Properties

[S]trict serializability

[N]on-blocking operations

[O]ne response per read

[W]rite transactions that conflict

\{\text{Highest Power}\}

\{\text{Lowest Latency}\}
[S]trict Serializability

- Strongest model: real-time + total order

“Photo B is private!”

ACL := Private
Upload Photo B

W starts

W finishes

R starts

R finishes
[S]trict Serializability

- Strongest model: real-time + total order
Non-blocking Operations

• Do not wait on external events
  – Locks, timeouts, messages, etc.

• Lower latency
  – Save the time spent blocking
One Response

• One round-trip
  – No message redirection
    • Centralized components: coordinator, etc.
  – No retries
  – Save the time for extra round-trips

• One value per response
  – Less time for transmitting, marshaling, etc.
[W]rite Transactions That Conflict

• Compatible with write transactions
  – Richer system model
  – Easier to program
The SNOW Theorem: **Impossible** for read-only transaction algorithms to have all SNOW properties
Why SNOW Is Impossible

Assume SNOW

Violates property S

\[ \begin{align*}
A & := \text{new} \\
B & := \text{new}
\end{align*} \]

\[ \begin{align*}
W & \text{ starts} \\
W & \text{ finishes}
\end{align*} \]
SNOW Is Tight

SNOW-optimal: have any 3 properties

Latency-optimal: have N and O
# Study Existing Systems with SNOW

SNOW-optimal and latency-optimal

<table>
<thead>
<tr>
<th>System</th>
<th>S</th>
<th>N</th>
<th>O</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spanner-Snap [OSDI '12]</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Yesquel [SOSP '15]</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>MySQL Cluster</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
## Study Existing Systems with SNOW

### SNOW-optimal

<table>
<thead>
<tr>
<th>System</th>
<th>S</th>
<th>N</th>
<th>O</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Eiger</strong> [NSDI ’13]</td>
<td>✔</td>
<td>✔</td>
<td>≤ 3</td>
<td>✔</td>
</tr>
<tr>
<td>DrTM [SOSP ’15]</td>
<td>✔</td>
<td>✔</td>
<td>≥ 1</td>
<td>✔</td>
</tr>
<tr>
<td>RIFL [SOSP ’15]</td>
<td>✔</td>
<td>✔</td>
<td>≥ 2</td>
<td>✔</td>
</tr>
<tr>
<td>Sinfonia [SOSP ’07]</td>
<td>✔</td>
<td>✔</td>
<td>≥ 2</td>
<td>✔</td>
</tr>
<tr>
<td>Spanner-RO [OSDI ’12]</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
## Study Existing Systems with SNOW Candidates for Improvement

<table>
<thead>
<tr>
<th>System</th>
<th>S</th>
<th>N</th>
<th>O</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPS</td>
<td>✗</td>
<td>✓</td>
<td>≤ 2</td>
<td>✗</td>
</tr>
<tr>
<td>Rococo</td>
<td>✓</td>
<td>✗</td>
<td>&gt; 1</td>
<td>✓</td>
</tr>
</tbody>
</table>

Many more

•

•

•
Improve Existing Systems with the SNOW Theorem

• COPS [SOSP ’11]
  – Geo-replicated
  – Causally consistent
  – Read-only txn: 🎣 N 🎣 W

• Rococo [OSDI ’14]
  – Supports general transactions
  – Strictly serializable
  – Read-only txn: S 🎣 W
New Algorithm Designs

• COPS-SNOW
  – Latency-optimal ($N + O$)

• Rococo-SNOW
  – SNOW-optimal ($S + O + W$)

Design insight for optimizing reads:
shift the overhead to writes
Rococo's Read-Only Txn (S + W)

W starts
\{ A := \text{"new"} \\
B := \text{"new"} \}

Gather conflict info

W commits

W finishes

R: 1\textsuperscript{st} round

\begin{align*}
A &= \text{"old"} \\
B &= \text{"new"}
\end{align*}

R: 2\textsuperscript{nd} round

R: N\textsuperscript{th} round
Rococo-SNOW (S+O+W)

R starts
W starts
\{ A := “new” \}
\{ B := “new” \}
Forward TS
W commits
W finishes

\begin{align*}
A &= “old” \\
B &= “old” \\
A &= old \\
B &= old
\end{align*}
Evaluation of Rococo-SNOW

• To understand
  – Latency of read-only transactions
  – Throughput of other types of transactions

• Experiment configuration
  – Identical to Rococo’s
  – TPC-C workloads

• https://github.com/USC-NSL/Rococo-SNOW
Significantly Lower Latency for Read-Only Txn

Latency (ms)

Concurrent requests/server

OCC
Rococo
2PL
Rococo-SNOW

Retries
Lock Wait
Always 1 round

High Contention
Higher Throughput under High Contention

-14% throughput (Low Contention)

2X throughput (High Contention)
Conclusion

• The SNOW Theorem for read-only txns
  – Impossible to have all of the SNOW properties
  – The SNOW Theorem is tight
  – Understands what is possible

• SNOW helps understand existing systems
  – Many are not yet optimal

• Rococo-SNOW
  – SNOW Theorem guided SNOW-optimal design
  – Significantly higher throughput and lower latency under high contention