

The SNOW Theorem and Latency-Optimal Read-Only Transactions

Haonan Lu[✉],
Christopher Hodsdon[✉], Khiem Ngo[✉],
Shuai Mu[†], Wyatt Lloyd[✉]

[✉]University of Southern California, [†]New York University

Huge Web Services Shard Data

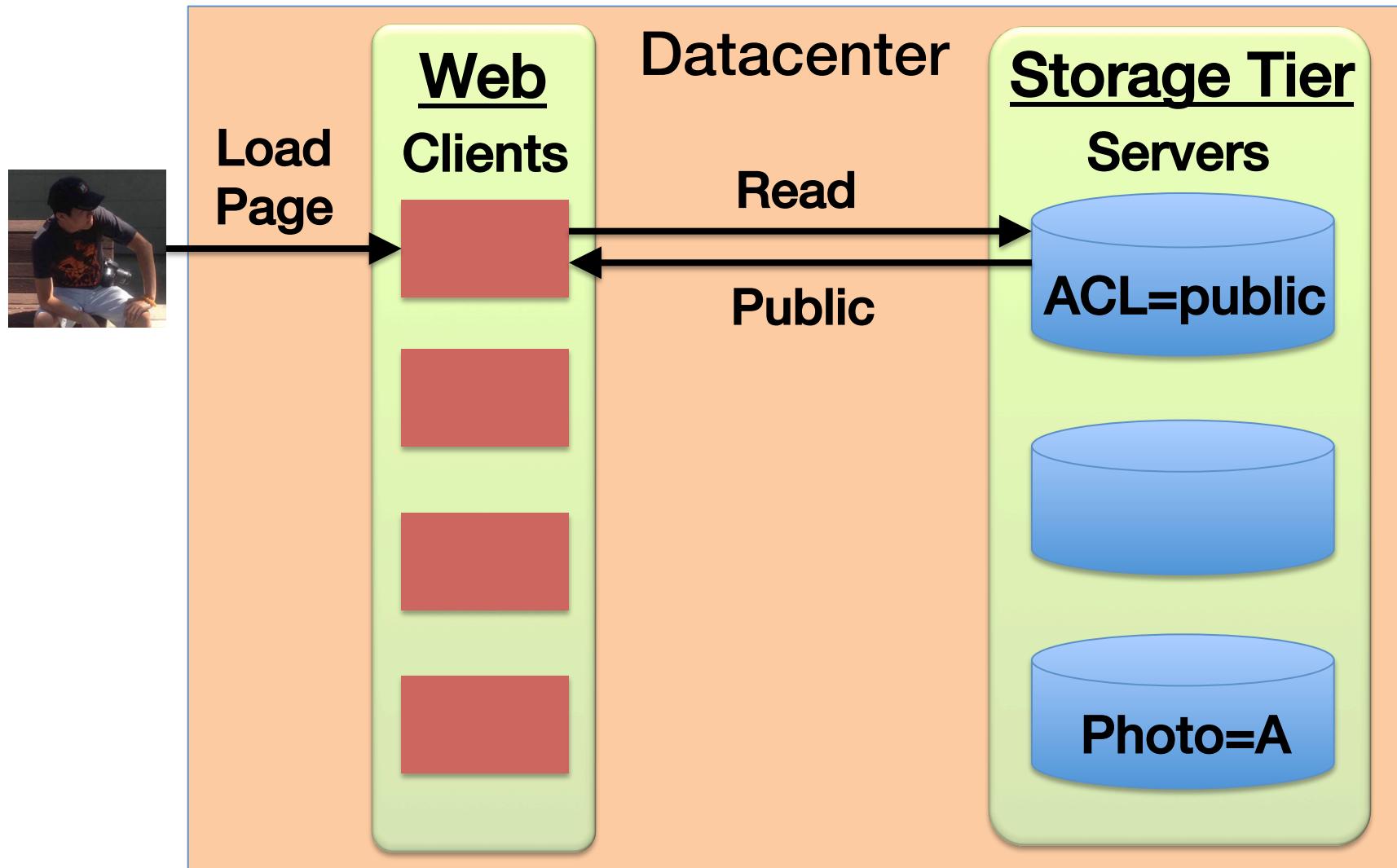


Massive amount of data

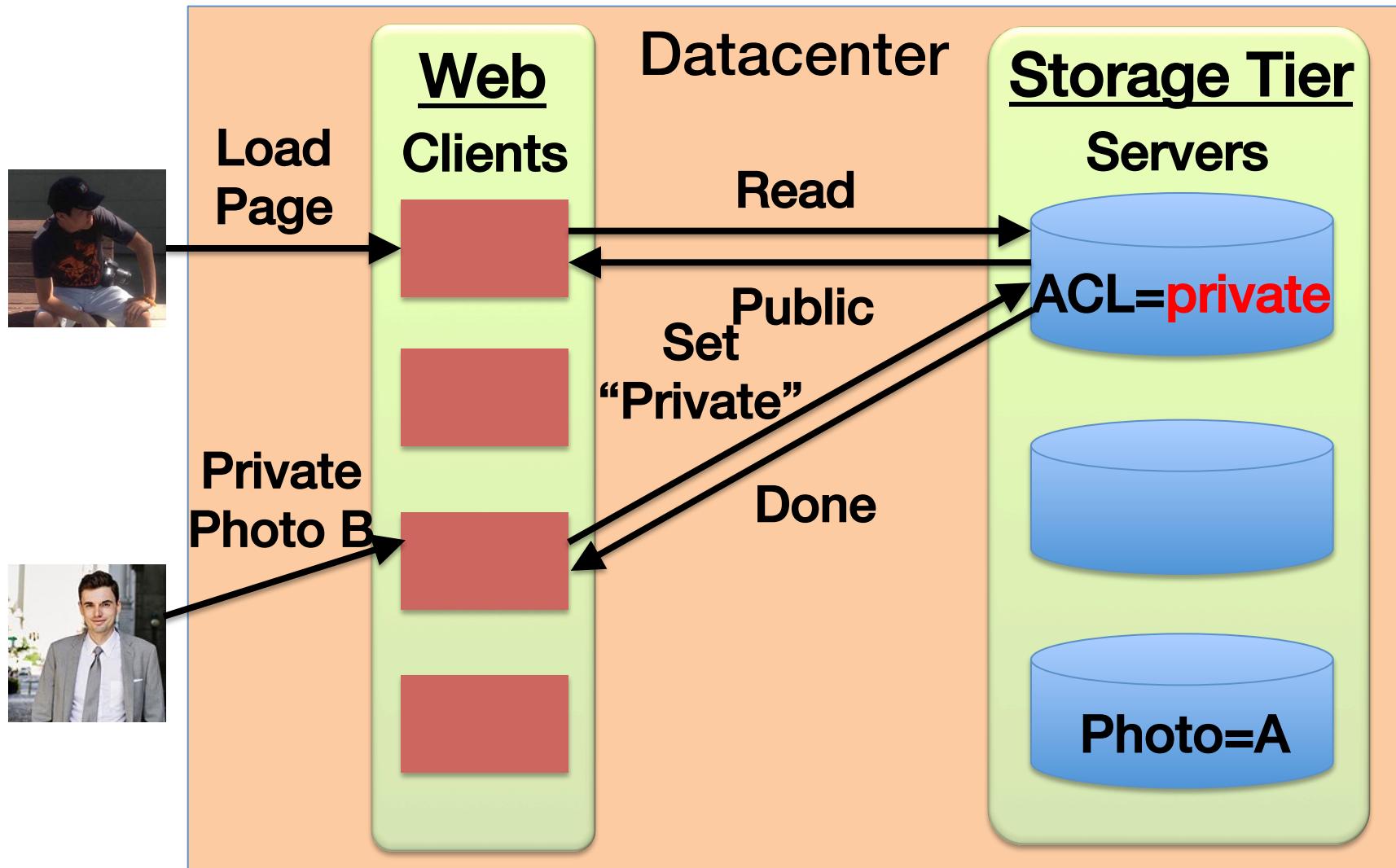
→ must be distributed across servers

Reads dominate the workloads
– need to be as fast as possible!

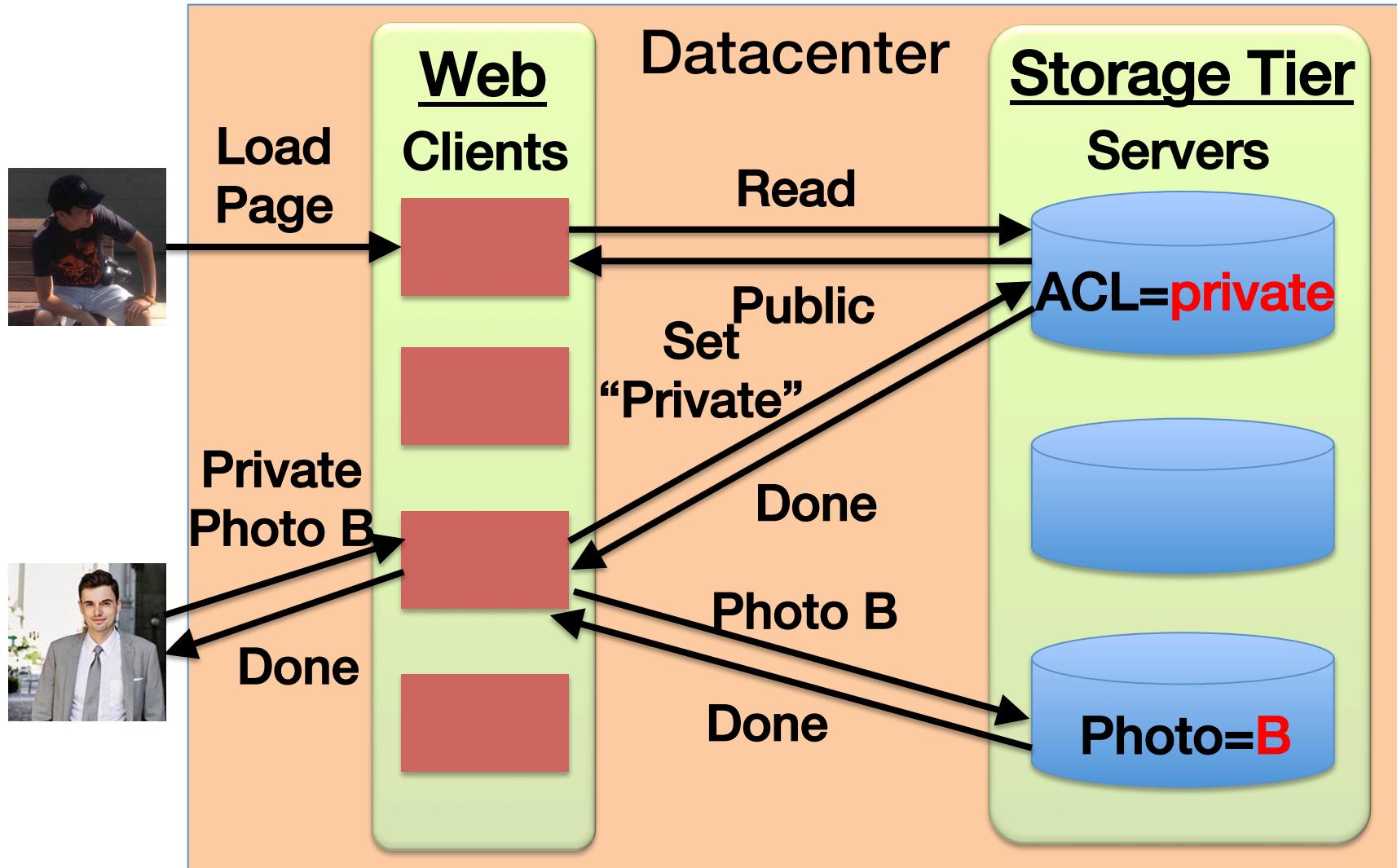
Simple Reads Are Insufficient



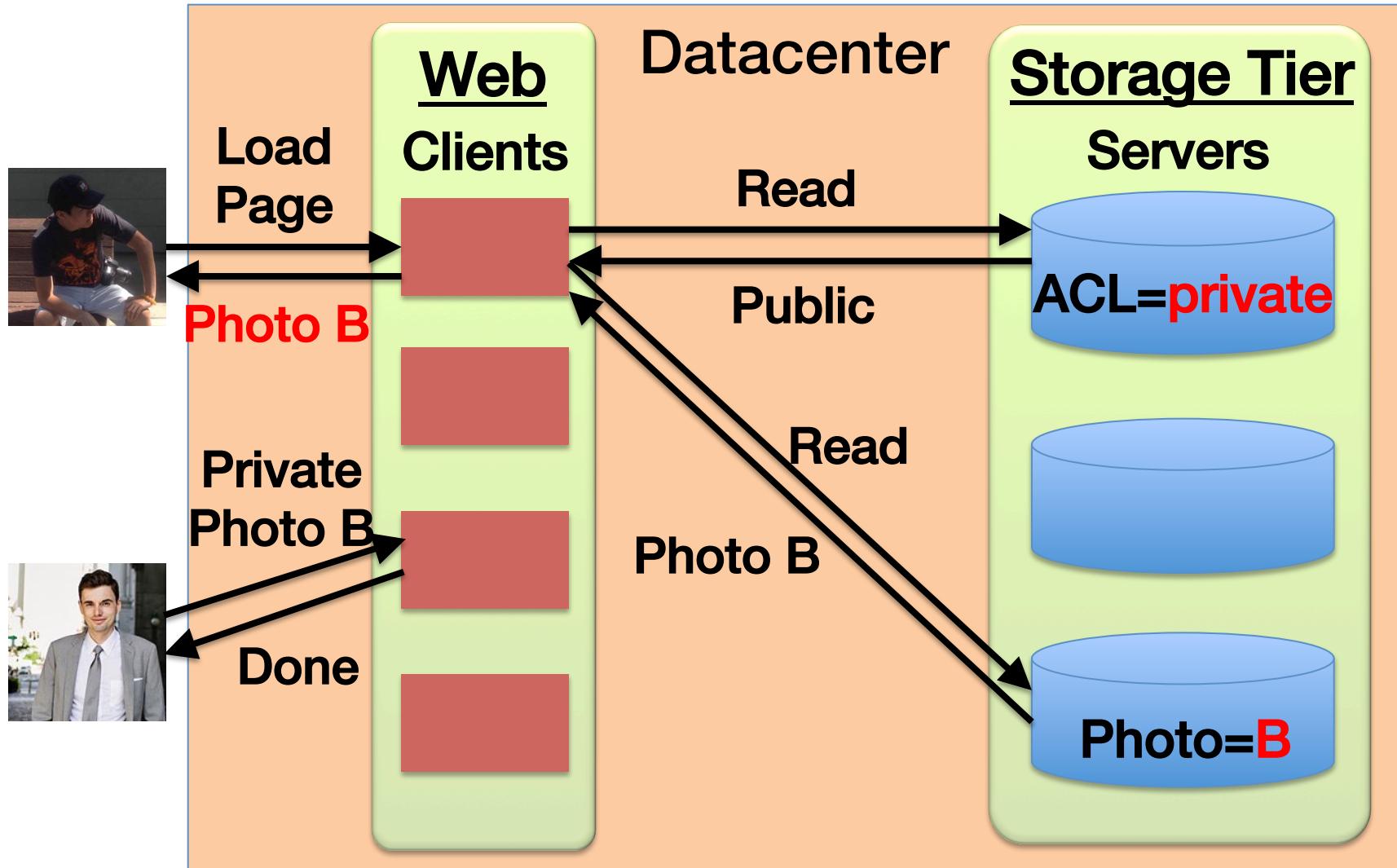
Simple Reads Are Insufficient



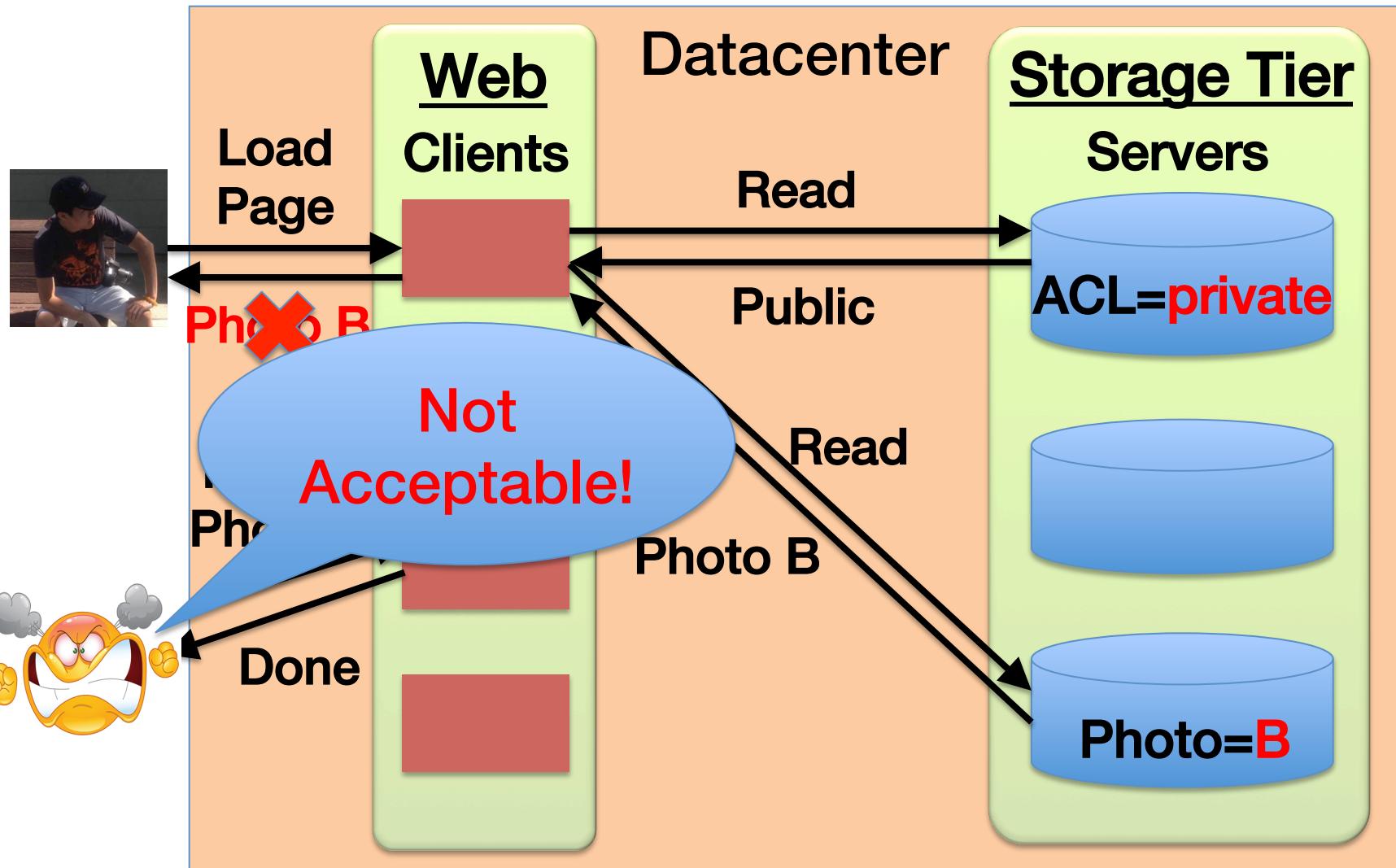
Simple Reads Are Insufficient



Simple Reads Are Insufficient



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

Intuitive Tension

High Power



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

~~Intuitive Tension~~

High Power



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

Fundamental Tradeoff

High Power



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

The SNOW Properties

[S]trict serializability

[W]rite transactions that conflict

[O]ne response per read

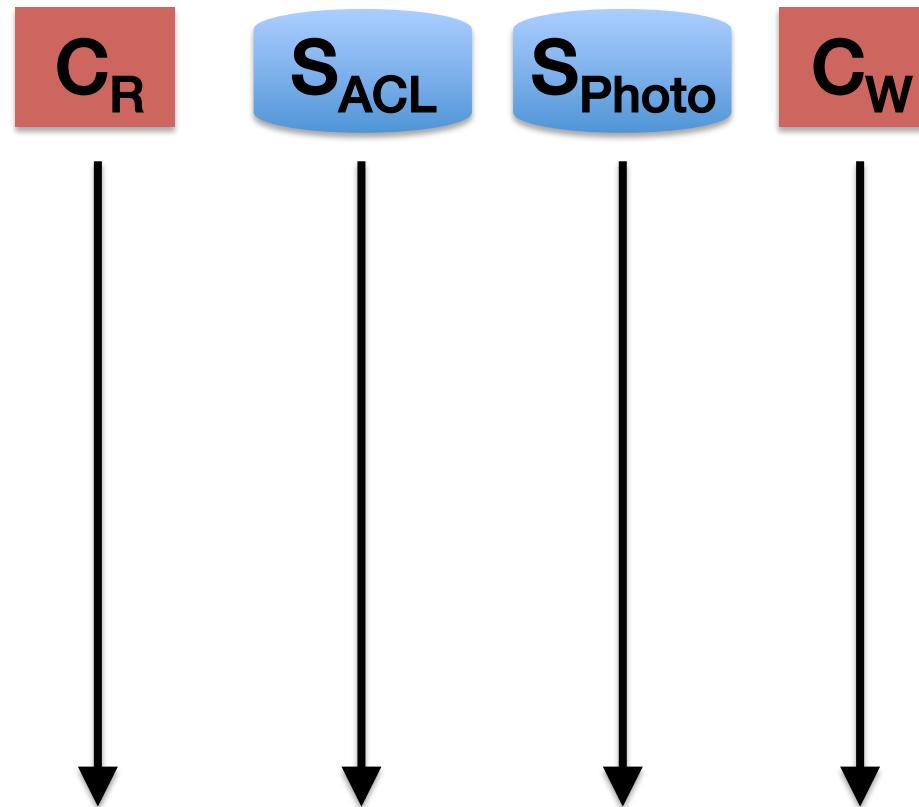
[N]on-blocking operations

Highest Power

Lowest Latency

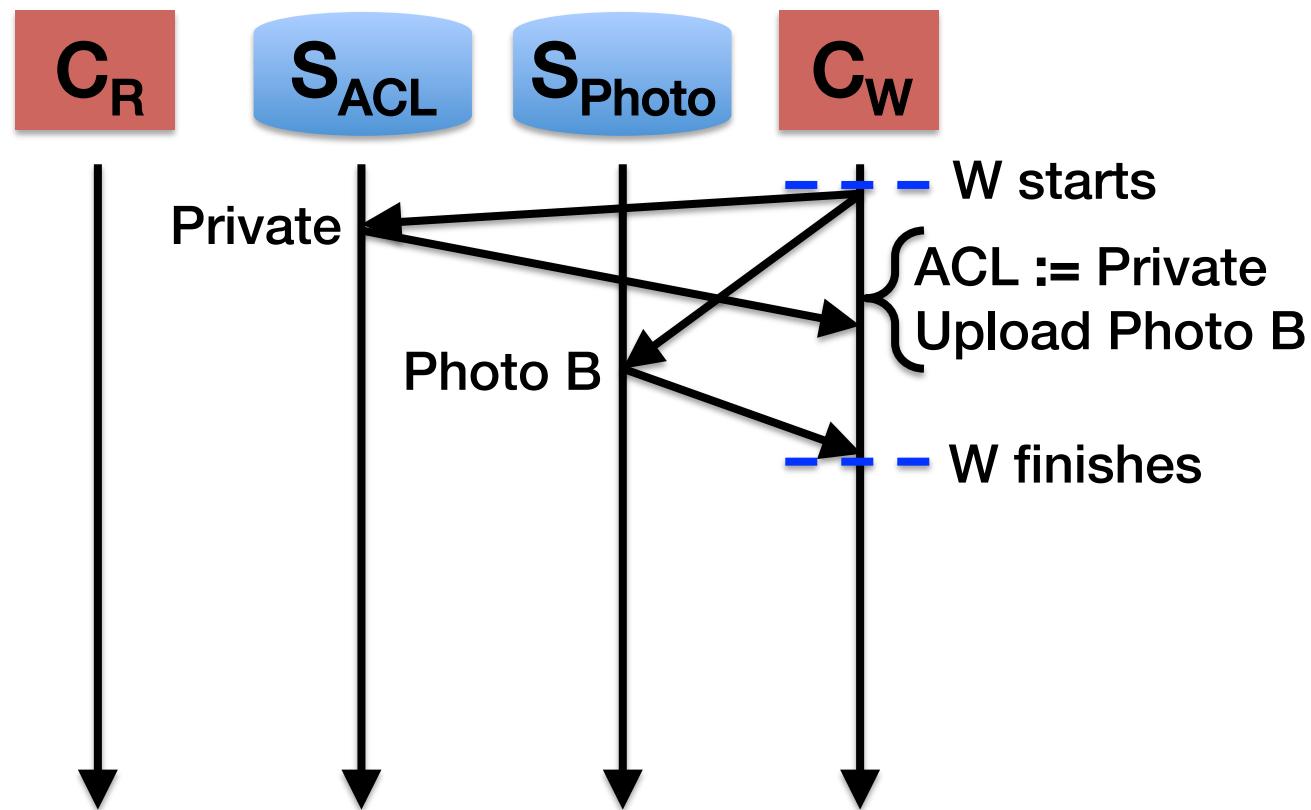
[S]trict Serializability

- Strongest model: real-time + total order



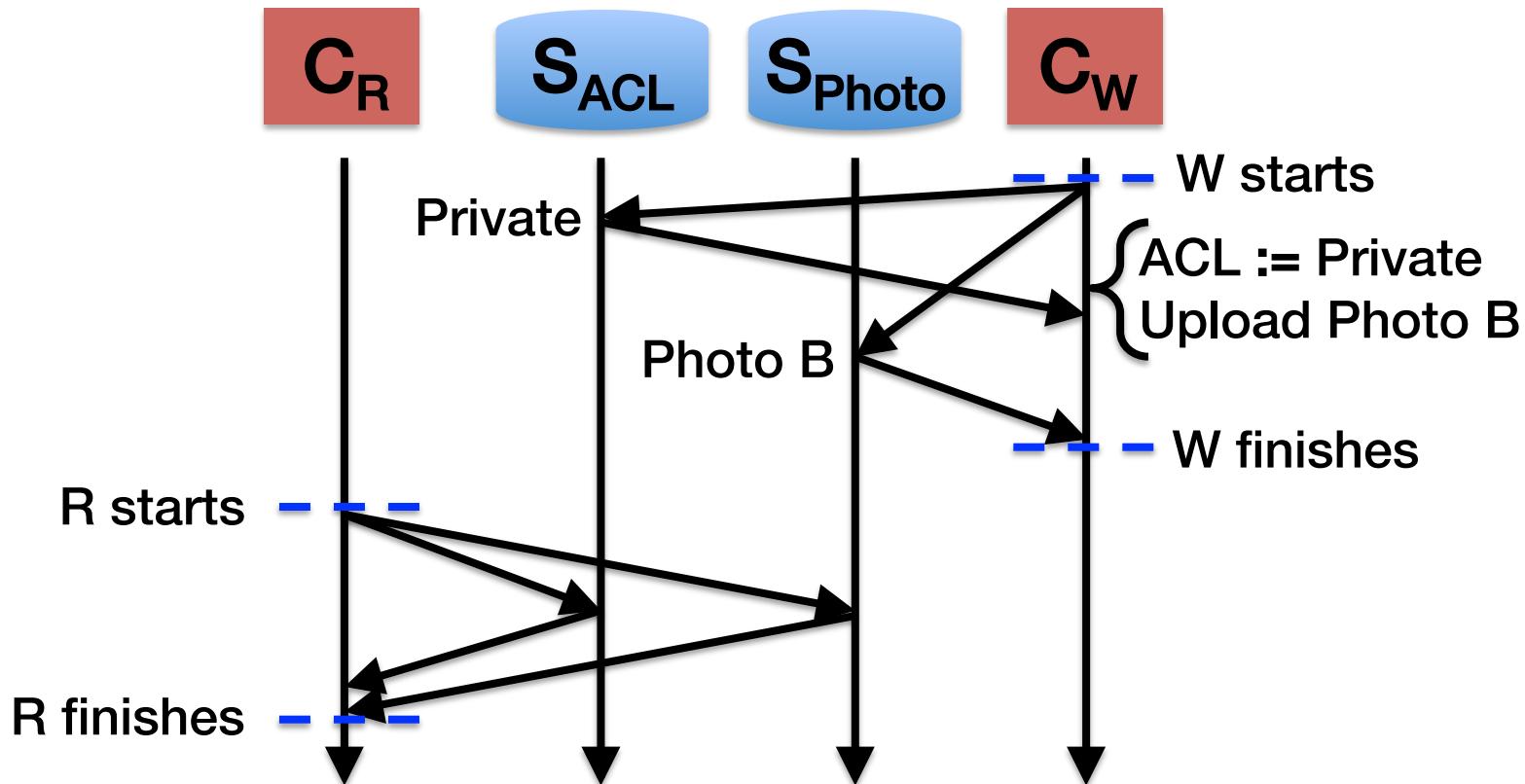
[S]trict Serializability

- Strongest model: real-time + total order



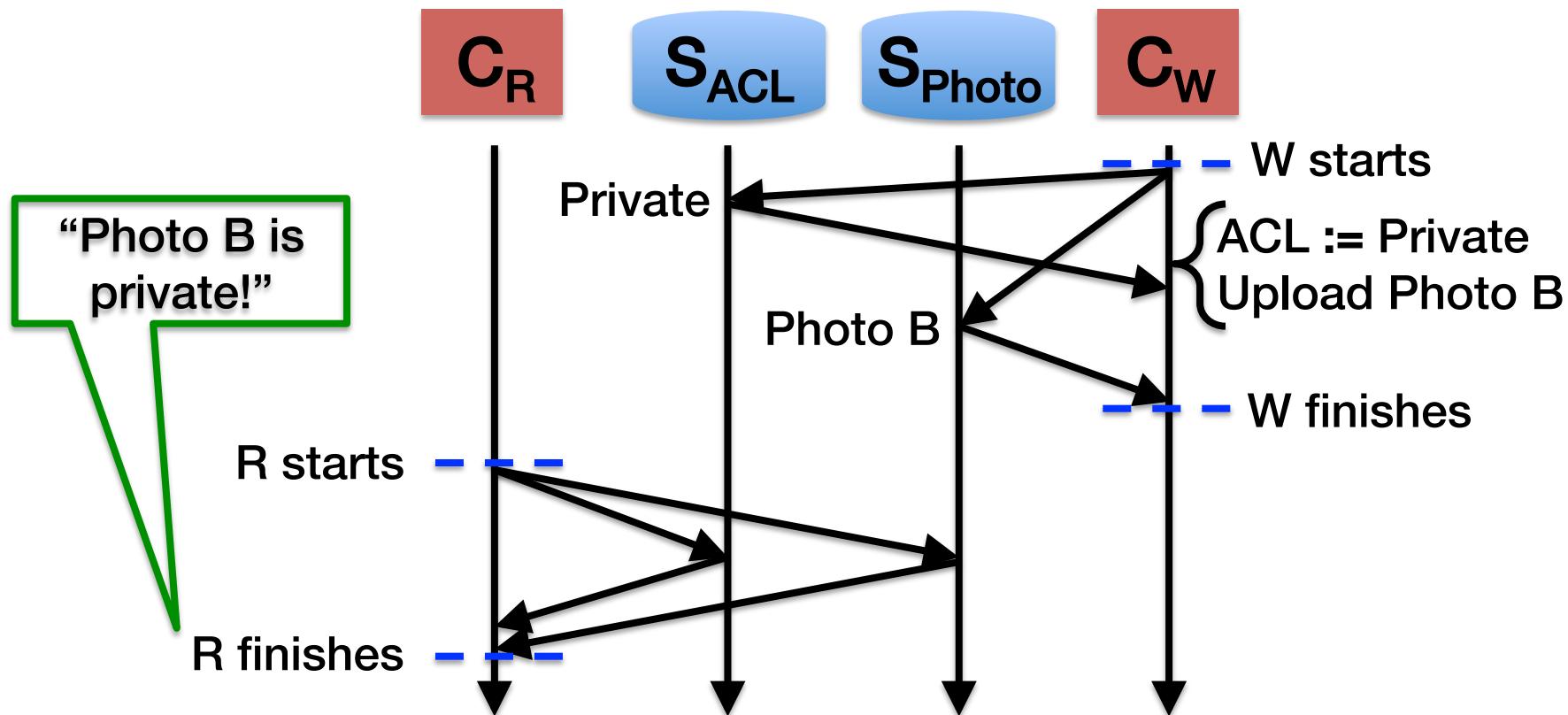
[S]trict Serializability

- Strongest model: real-time + total order



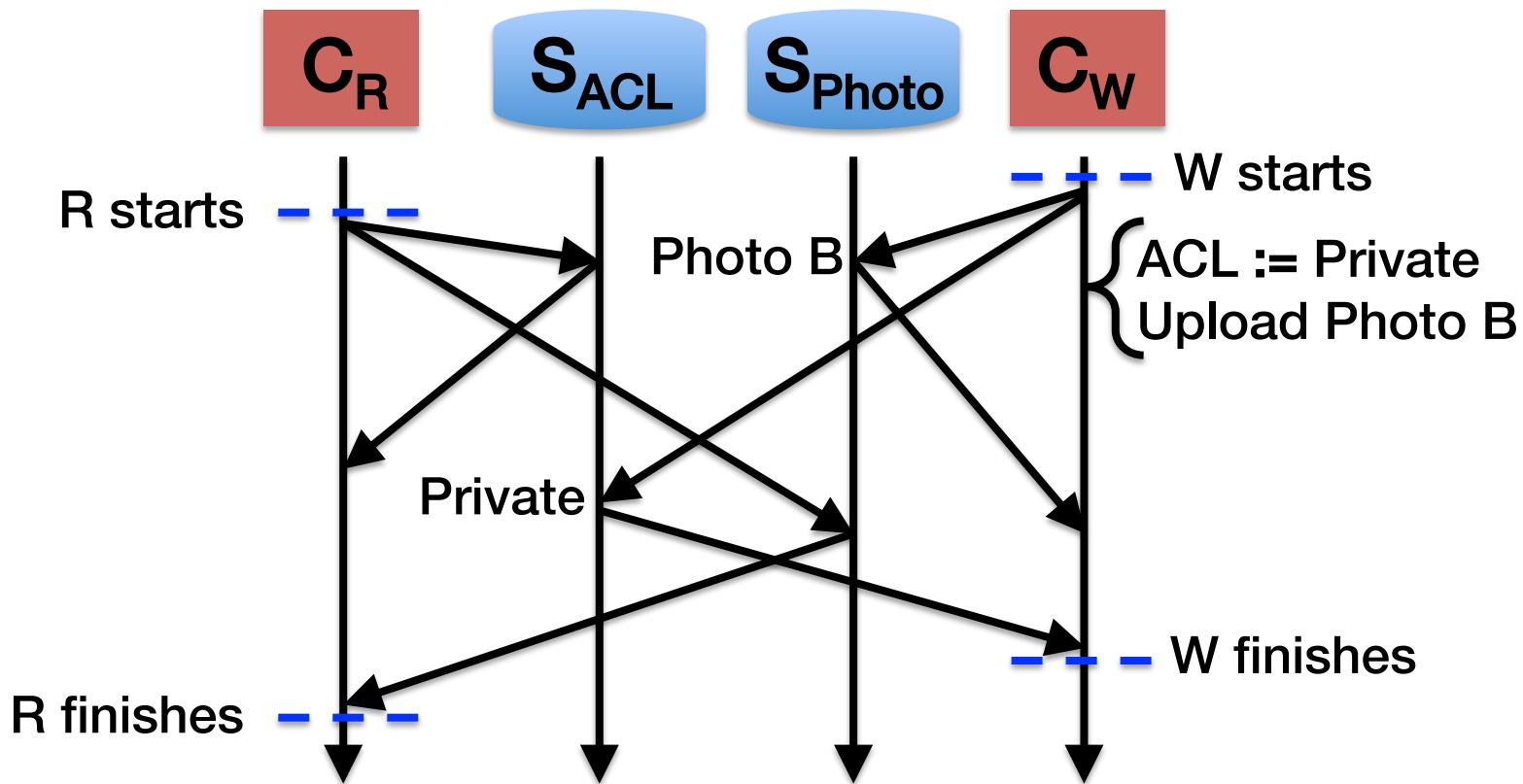
[S]trict Serializability

- Strongest model: real-time + total order



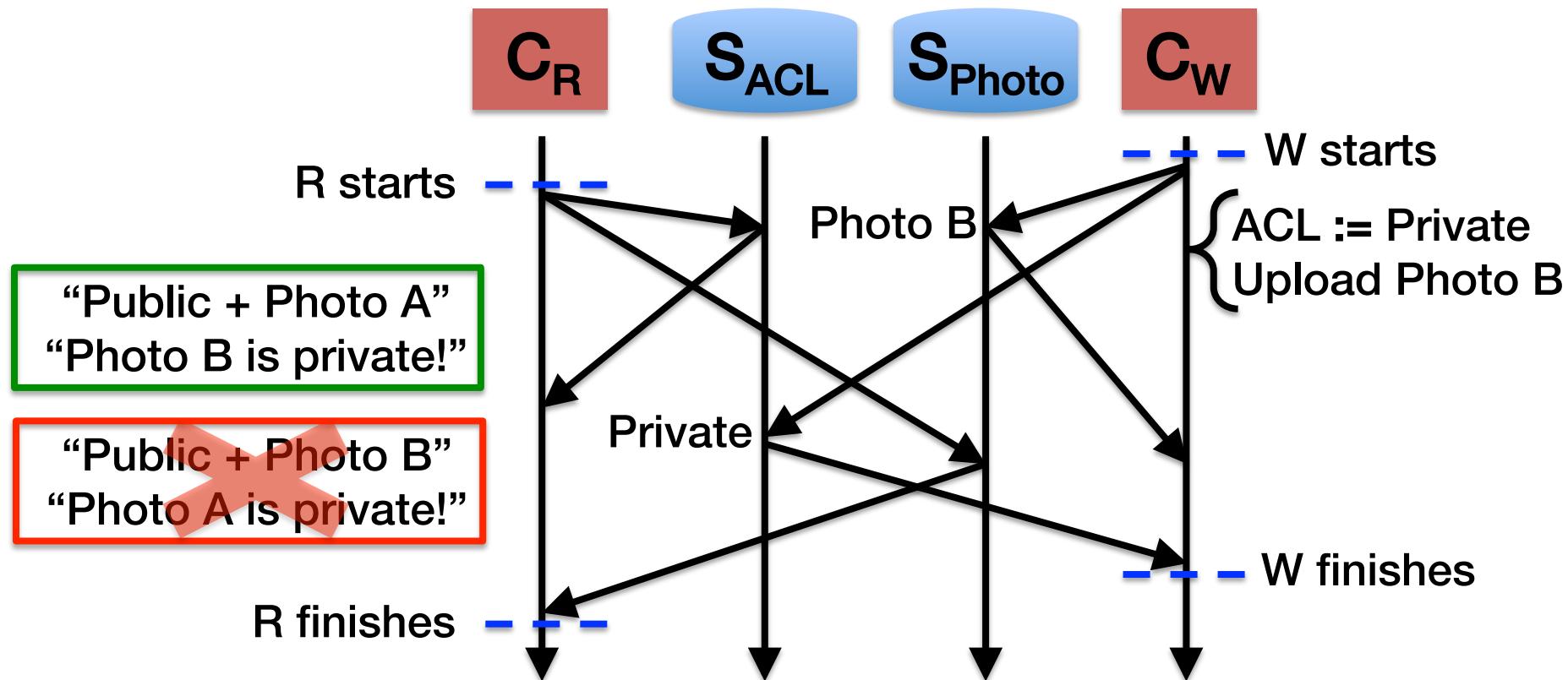
[S]trict Serializability

- Strongest model: real-time + total order



[S]trict Serializability

- Strongest model: real-time + total order



[N]on-blocking Operations

- Do not wait on external events
 - Locks, timeouts, messages, etc.
- Lower latency
 - Save the time spent blocking

[O]ne 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.

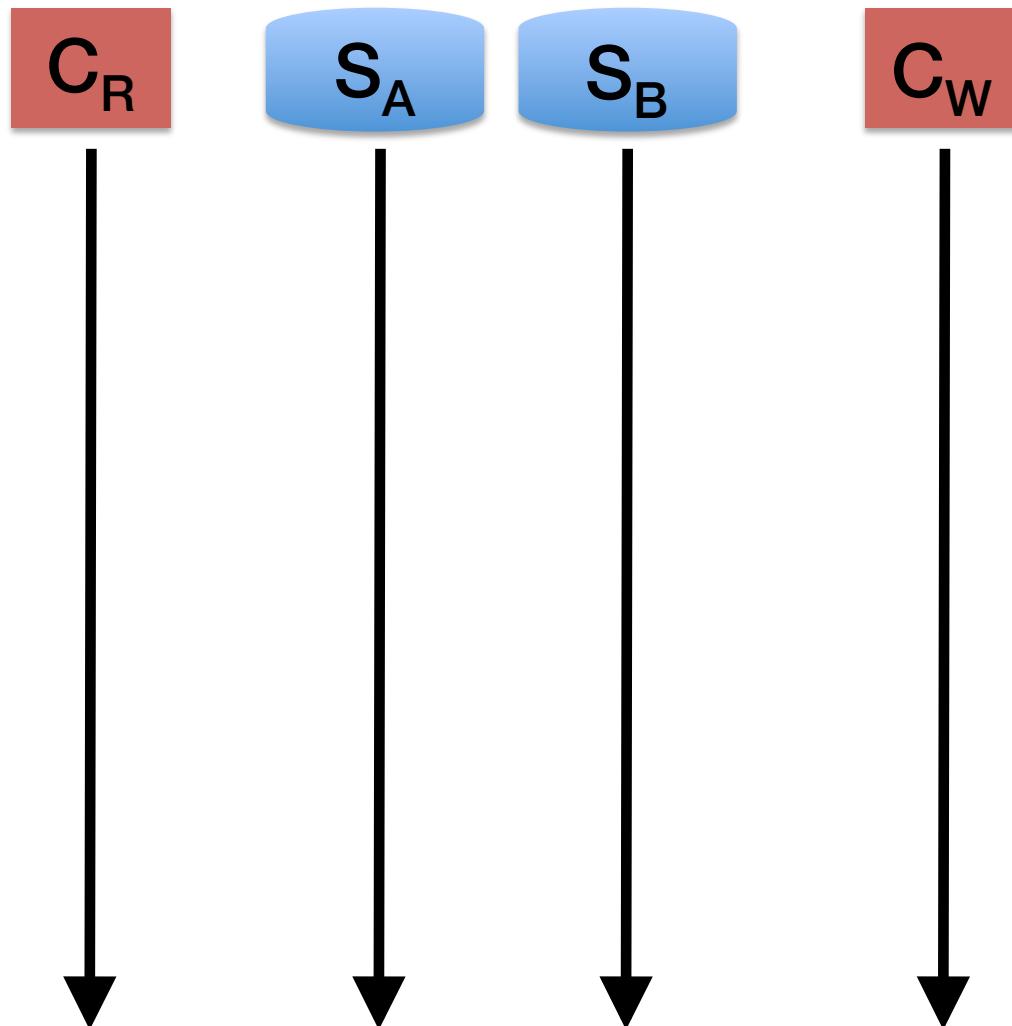
[W]rite Transactions That Conflict

- Compatible with write transactions
 - Richer system model
 - Easier to program

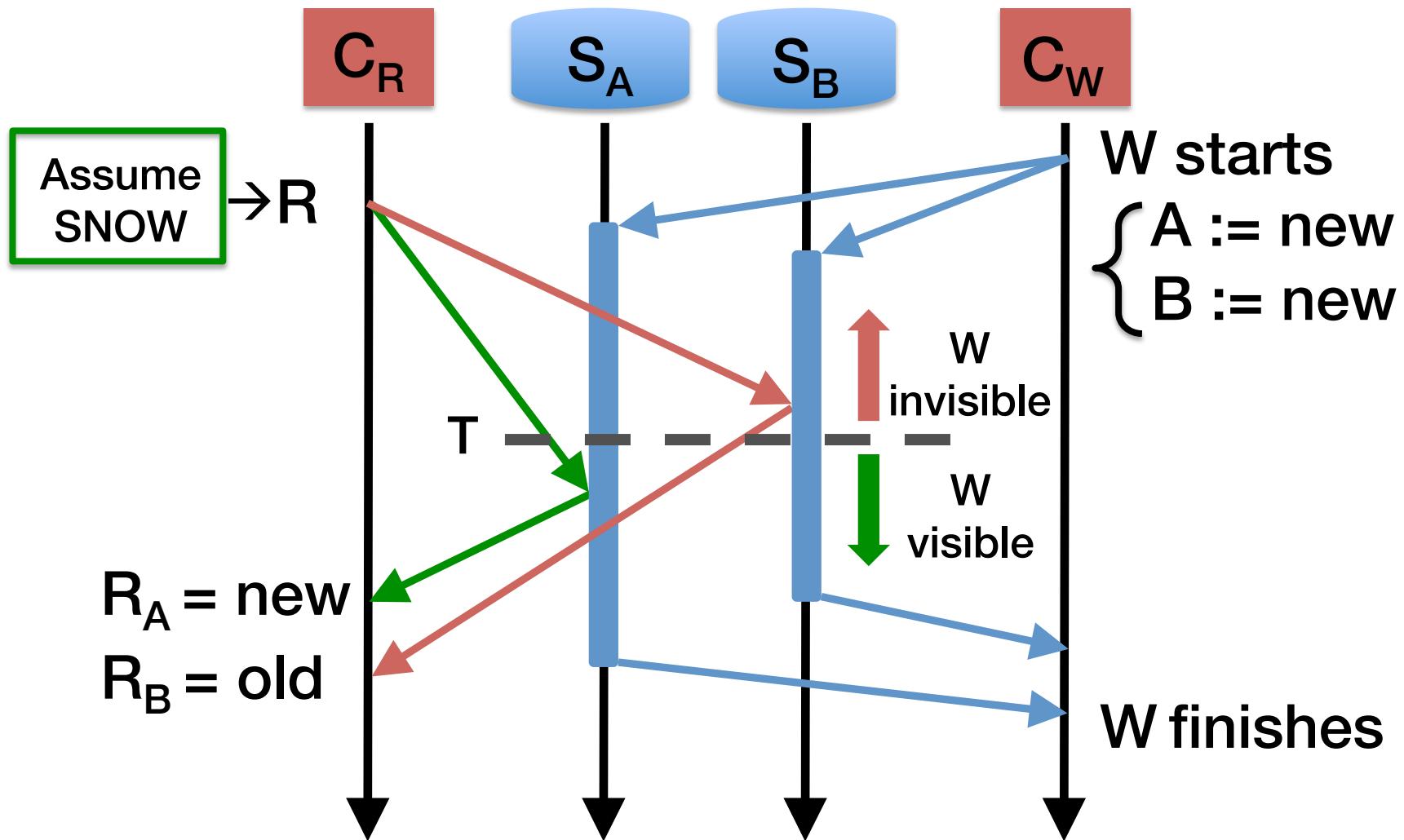
The SNOW Theorem:

Impossible for read-only transaction
algorithms to have all SNOW properties

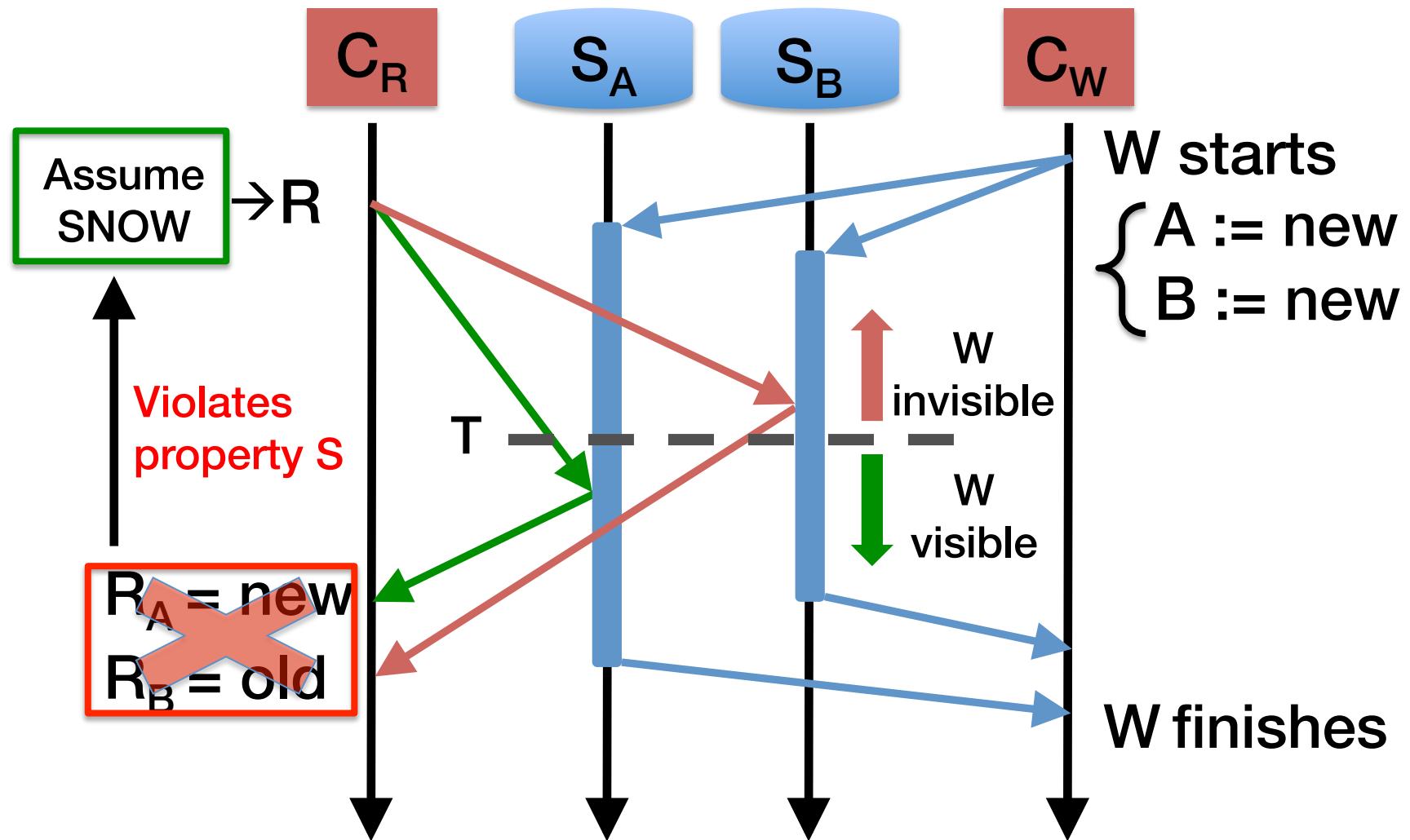
Why SNOW Is Impossible



Why SNOW Is Impossible



Why SNOW Is Impossible



A Deeper Look at SNOW

- Complete proof in the paper
- SNOW is tight
 - Any combination of 3 properties is possible
- Optimality
 - SNOW-optimal: have any 3 properties
 - Latency-optimal: have property N and O
- Spectrums of property S and O
 - Show what is possible to achieve

Study Existing Systems with SNOW

SNOW-optimal and latency-optimal

System	S	N	O	W
Spanner-Snap [OSDI '12]	*	✓	✓	✓
Yesquel [SOSP '15]	*	✓	✓	✓
MySQL Cluster	*	✓	✓	✓

Study Existing Systems with SNOW

SNOW-optimal

System	S	N	O	W
Eiger [NSDI '13]	✓	✓	≤ 3	✓
DrTM [SOSP '15]	✓	✓	≤ 1	✓
RIFL [SOSP '15]	✓	✓	≥ 2	✓
Sinfonia [SOSP '07]	✓	✓	≥ 2	✓
Spanner-RO [OSDI '12]	✓	*	✓	✓

Study Existing Systems with SNOW

Candidates for Improvement

System	S	N	O	W
COPS	*	✓	≤ 2	*
Rococo	✓	*	> 1	✓

Many more

•
•
•

Improve Existing Systems with the SNOW Theorem

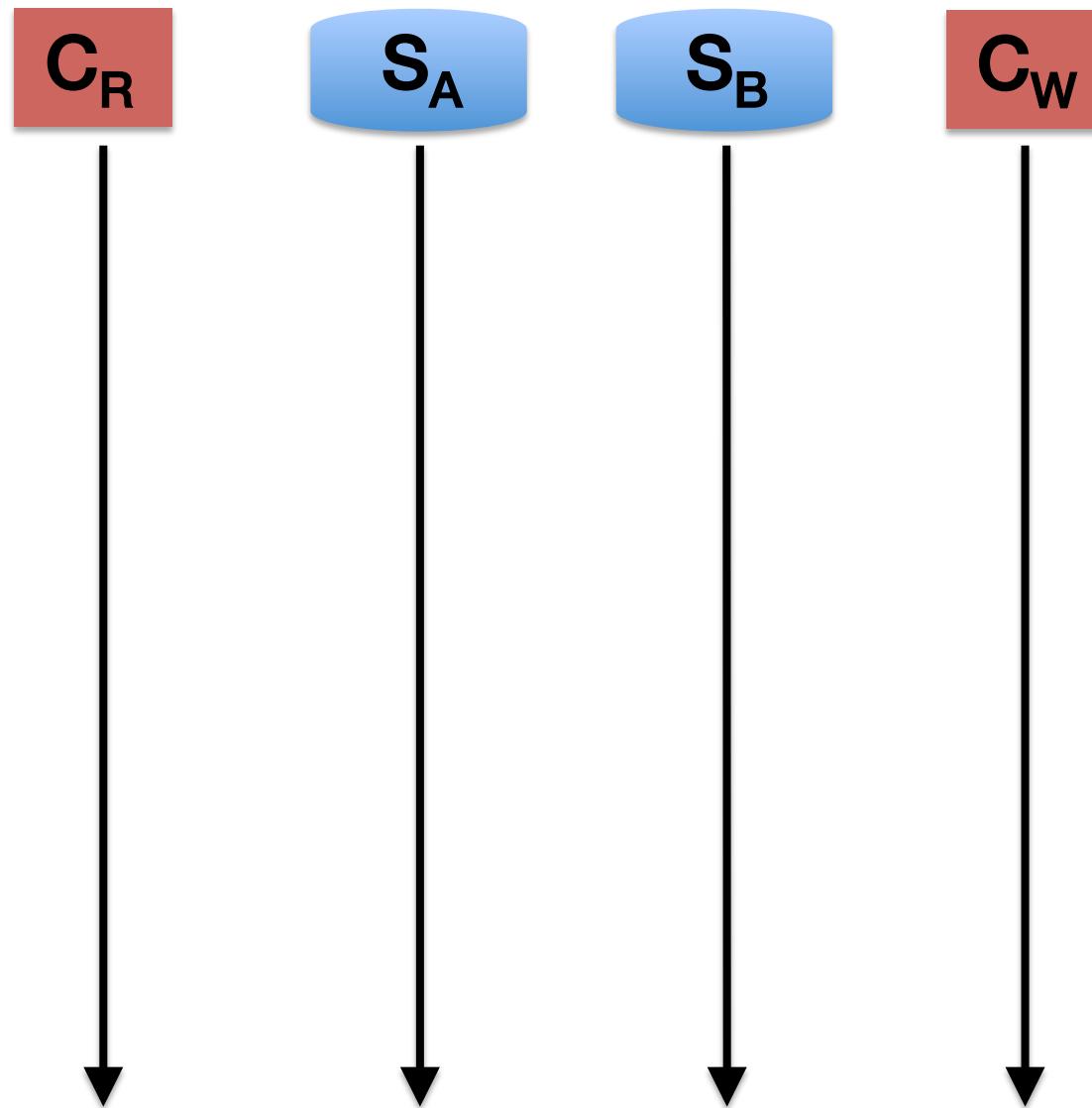
- COPS [SOSP '11]
 - Geo-replicated
 - Causally consistent
 - Read-only txn: $\cancel{S} \ N \ \cancel{\otimes} \ \cancel{W}$
- Rococo [OSDI '14]
 - Supports general transactions
 - Strictly serializable
 - Read-only txn: $S \ \cancel{\otimes} \ \cancel{\otimes} \ W$

New Algorithm Designs

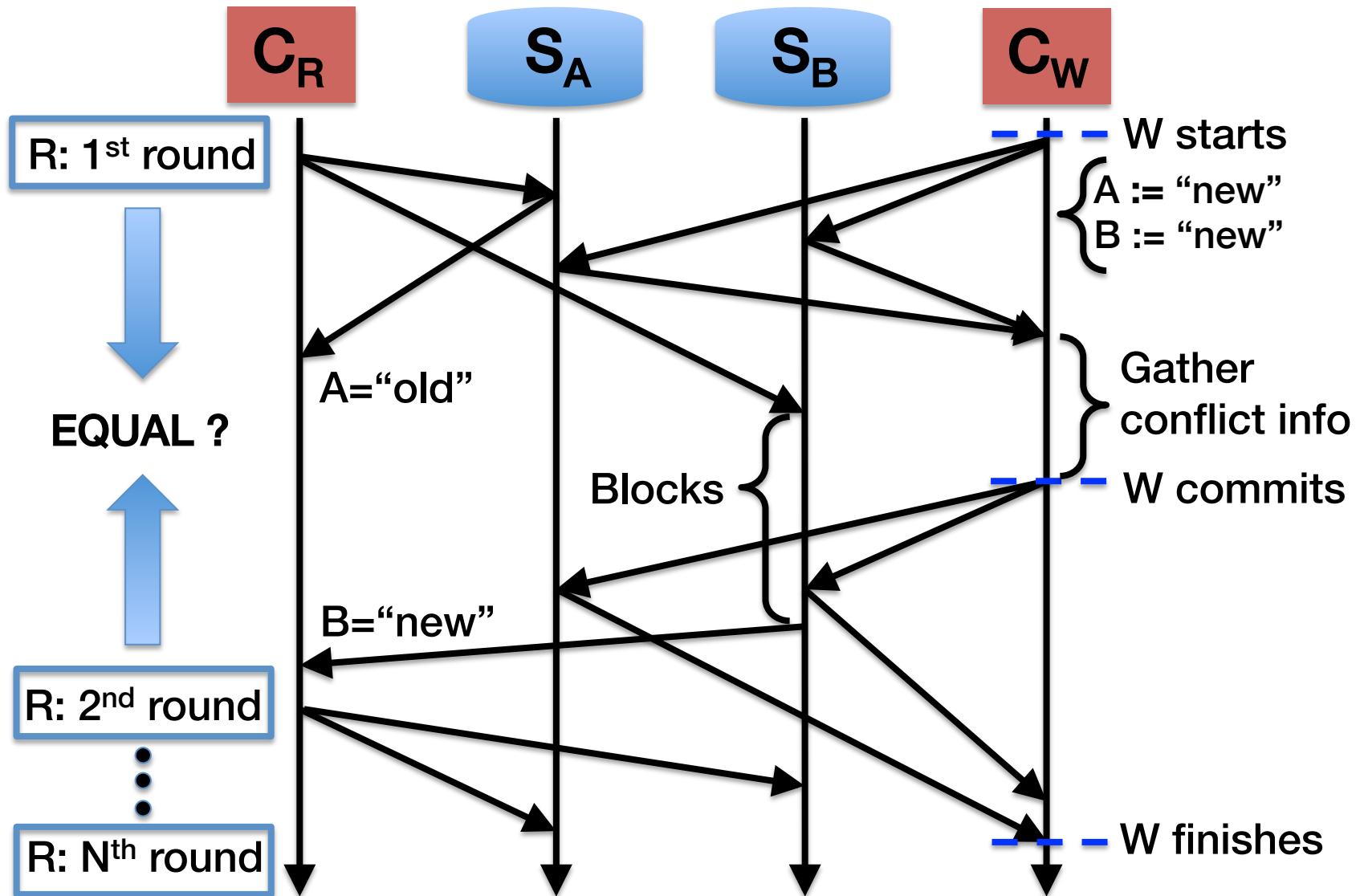
- COPS-SNOW
 - Latency-optimal ($N + O$)
- Rococo-SNOW
 - SNOW-optimal ($S + O + W$)

Design insight for optimizing reads:
shift the overhead to writes

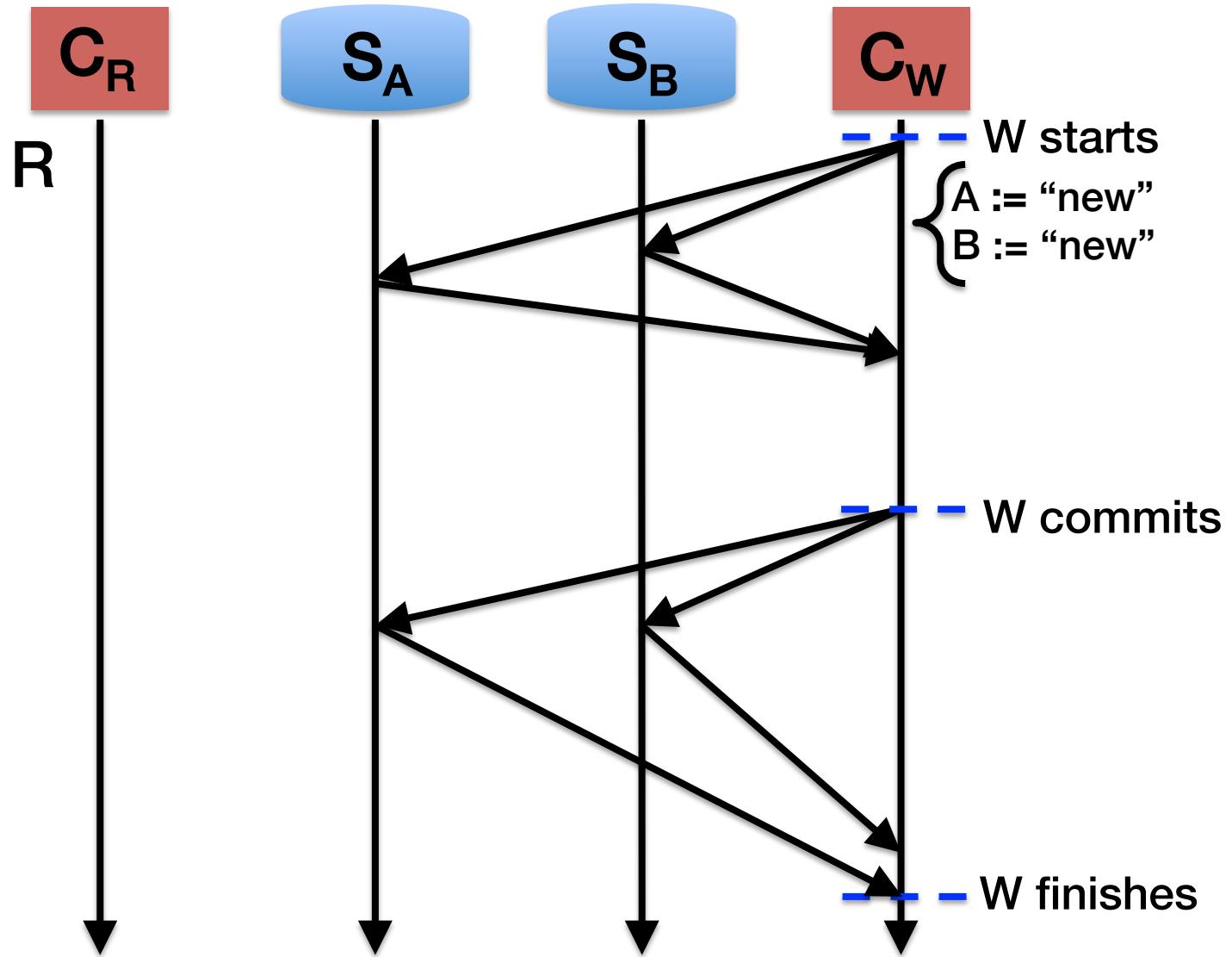
Rococo's Read-Only Txn ($S + W$)



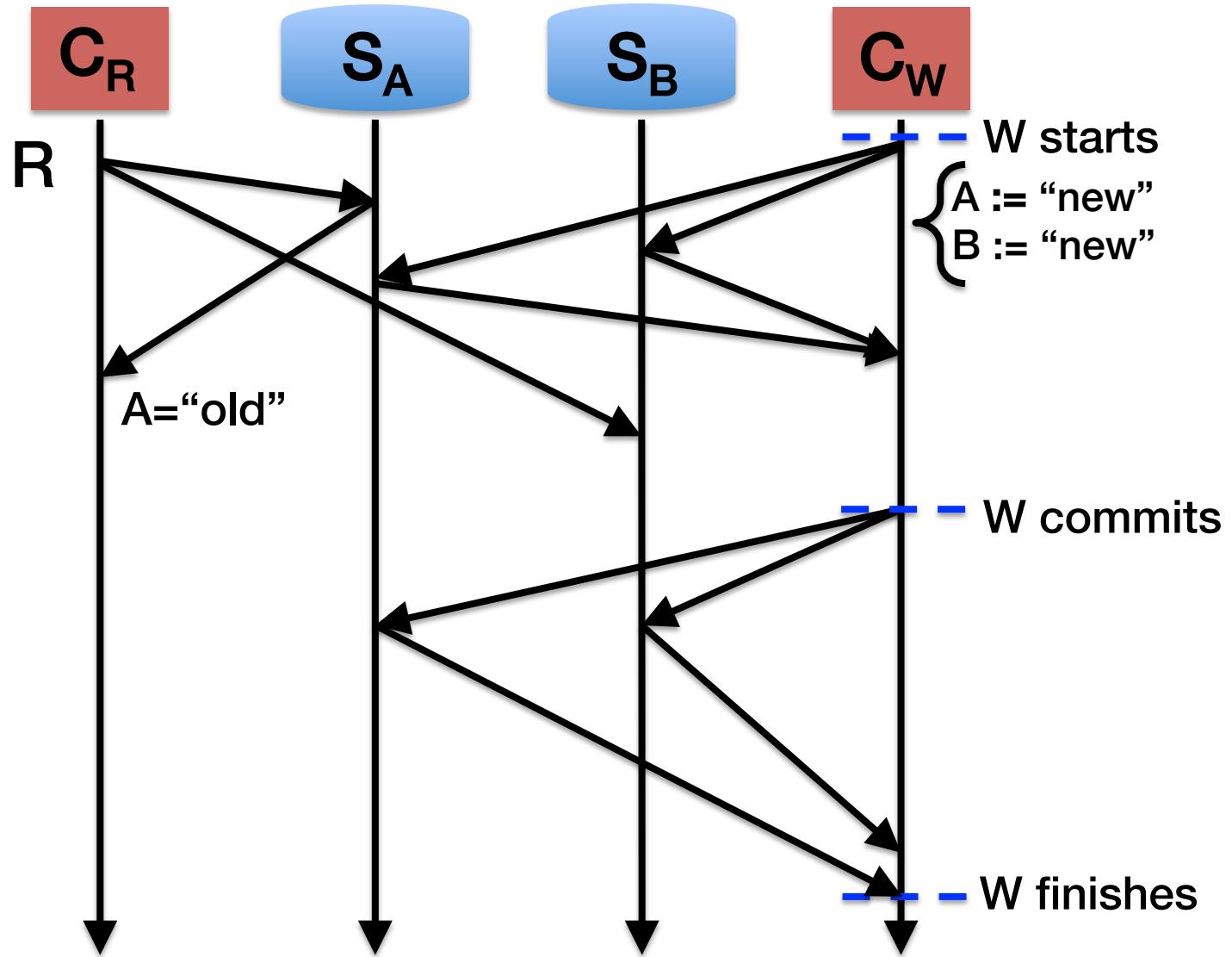
Rococo's Read-Only Txn ($S + W$)



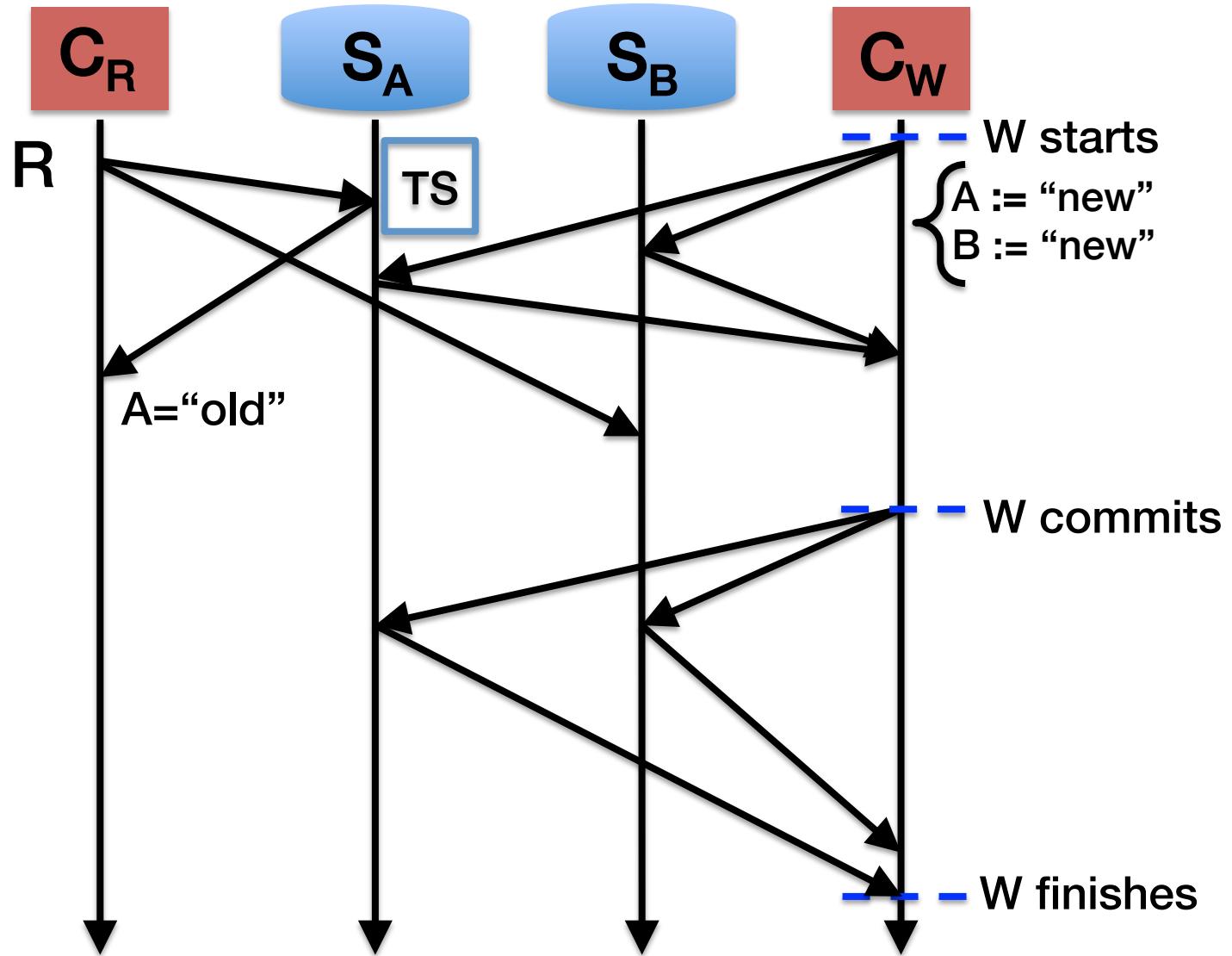
Rococo-SNOW (S+O+W)



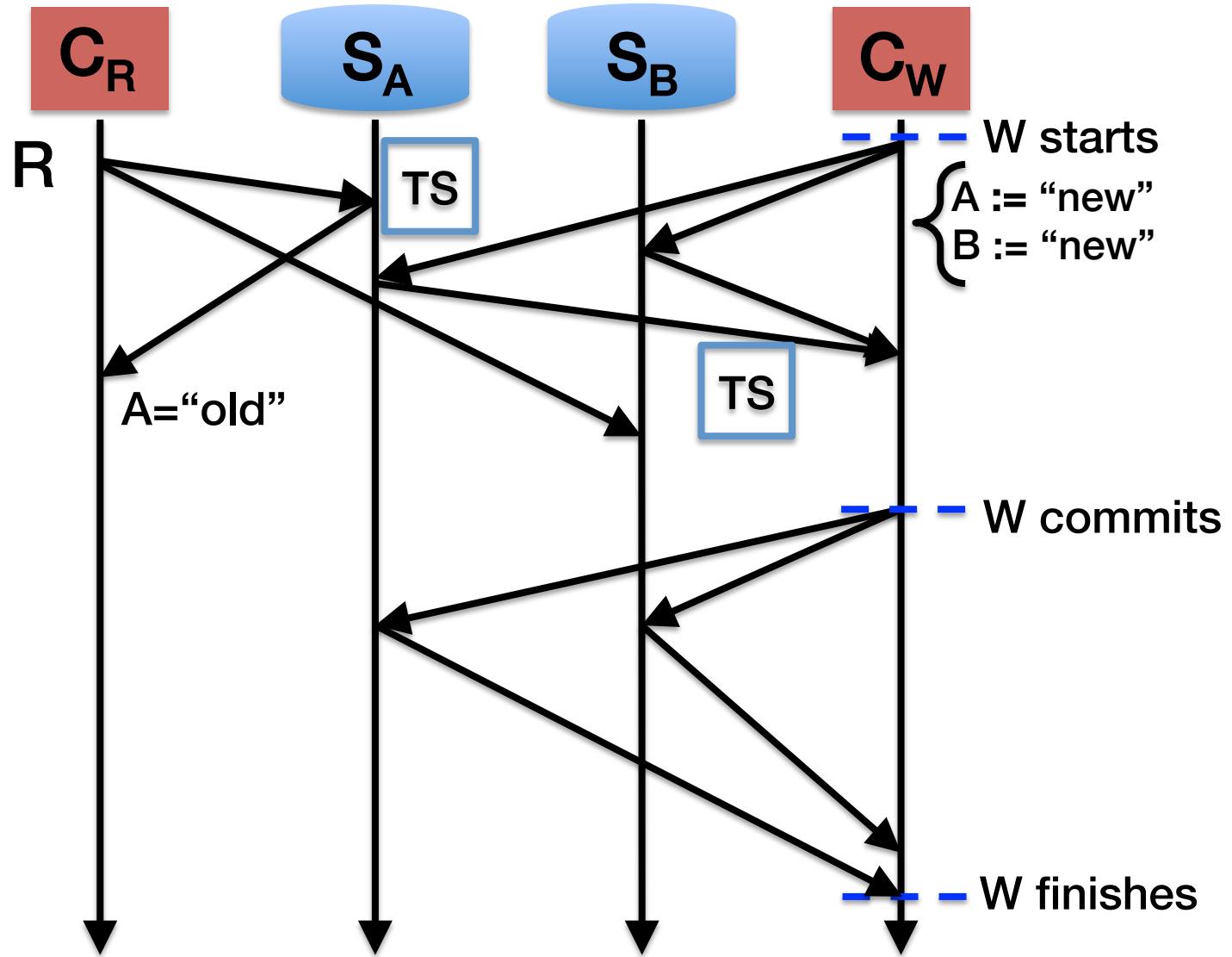
Rococo-SNOW (S+O+W)



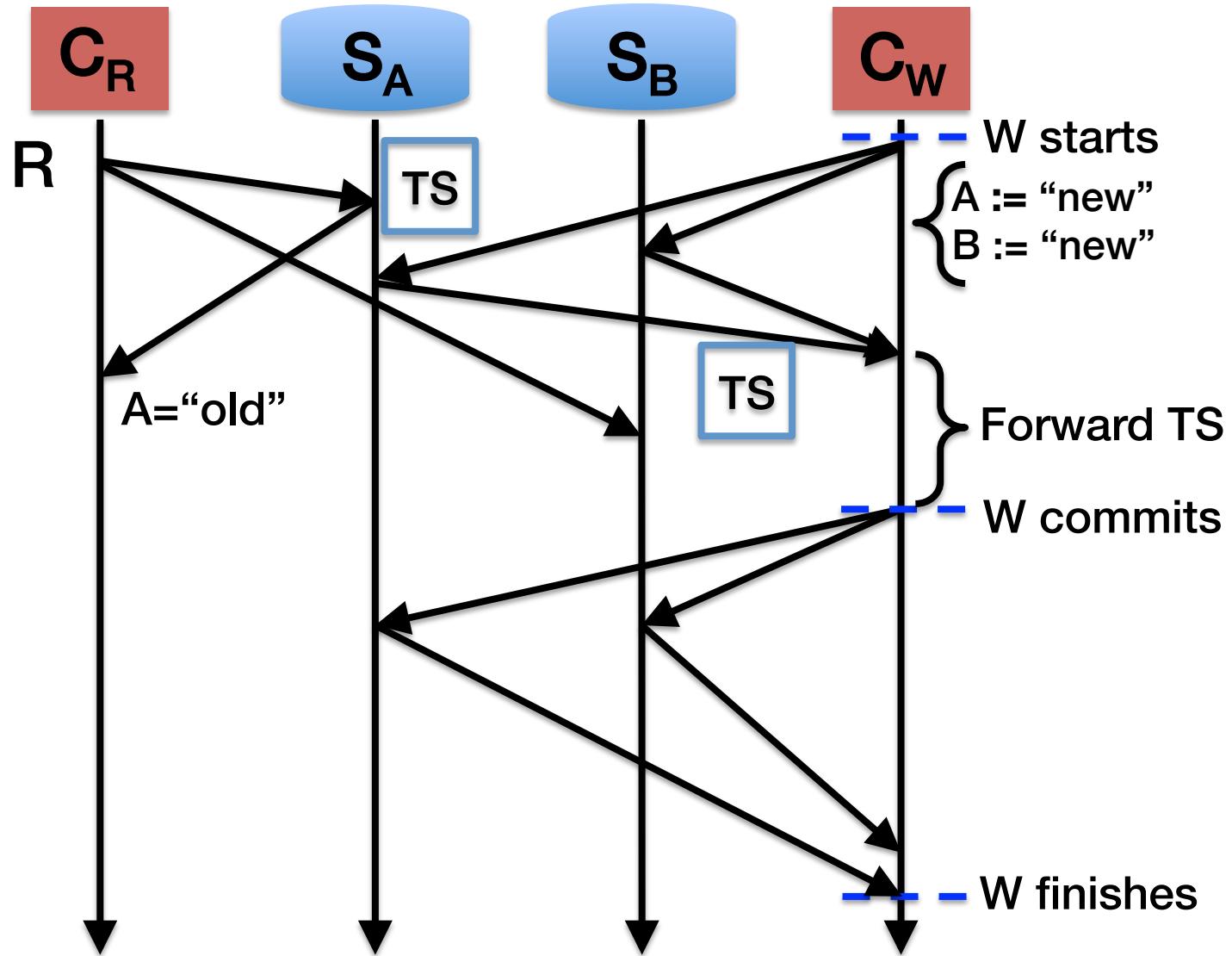
Rococo-SNOW (S+O+W)



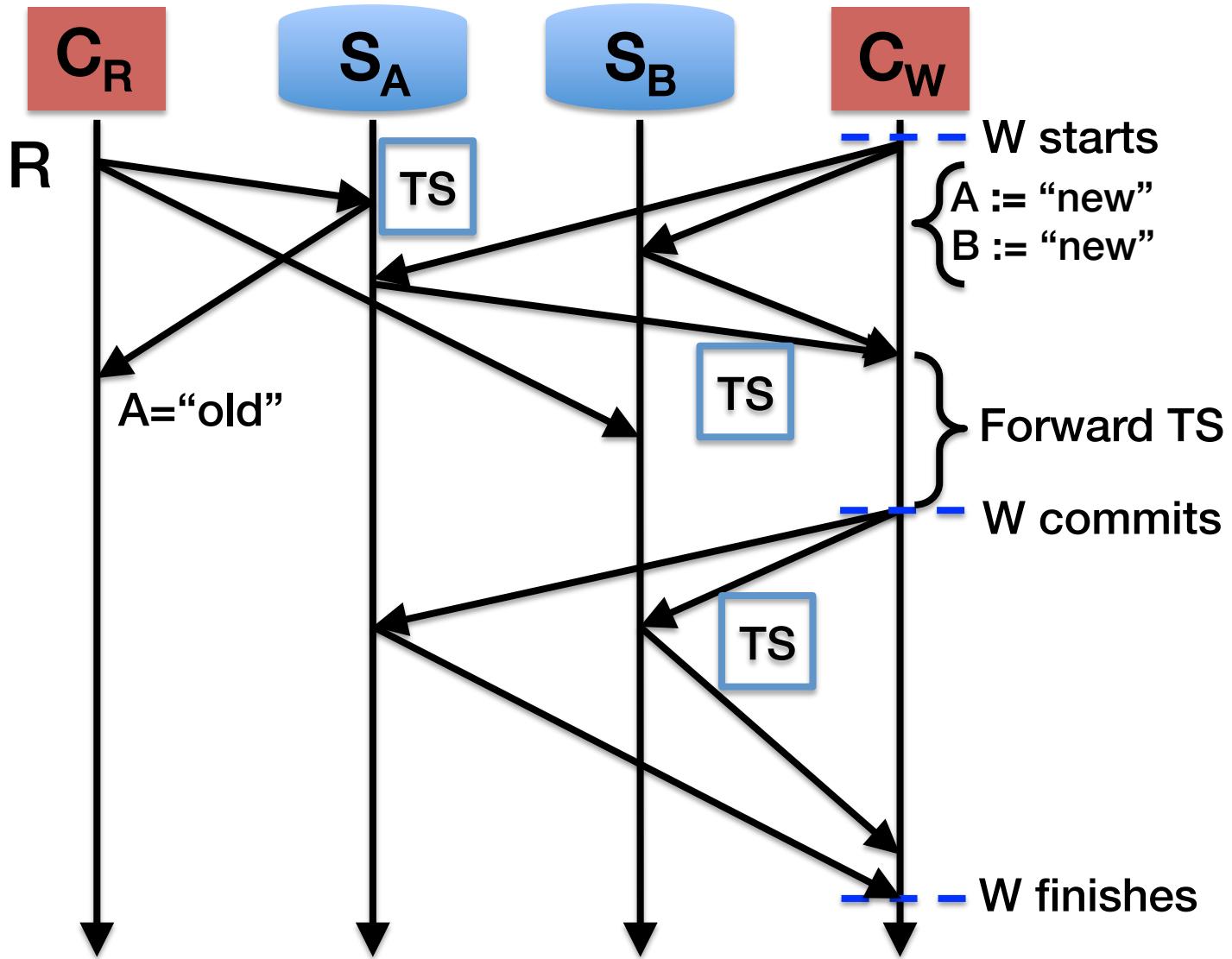
Rococo-SNOW (S+O+W)



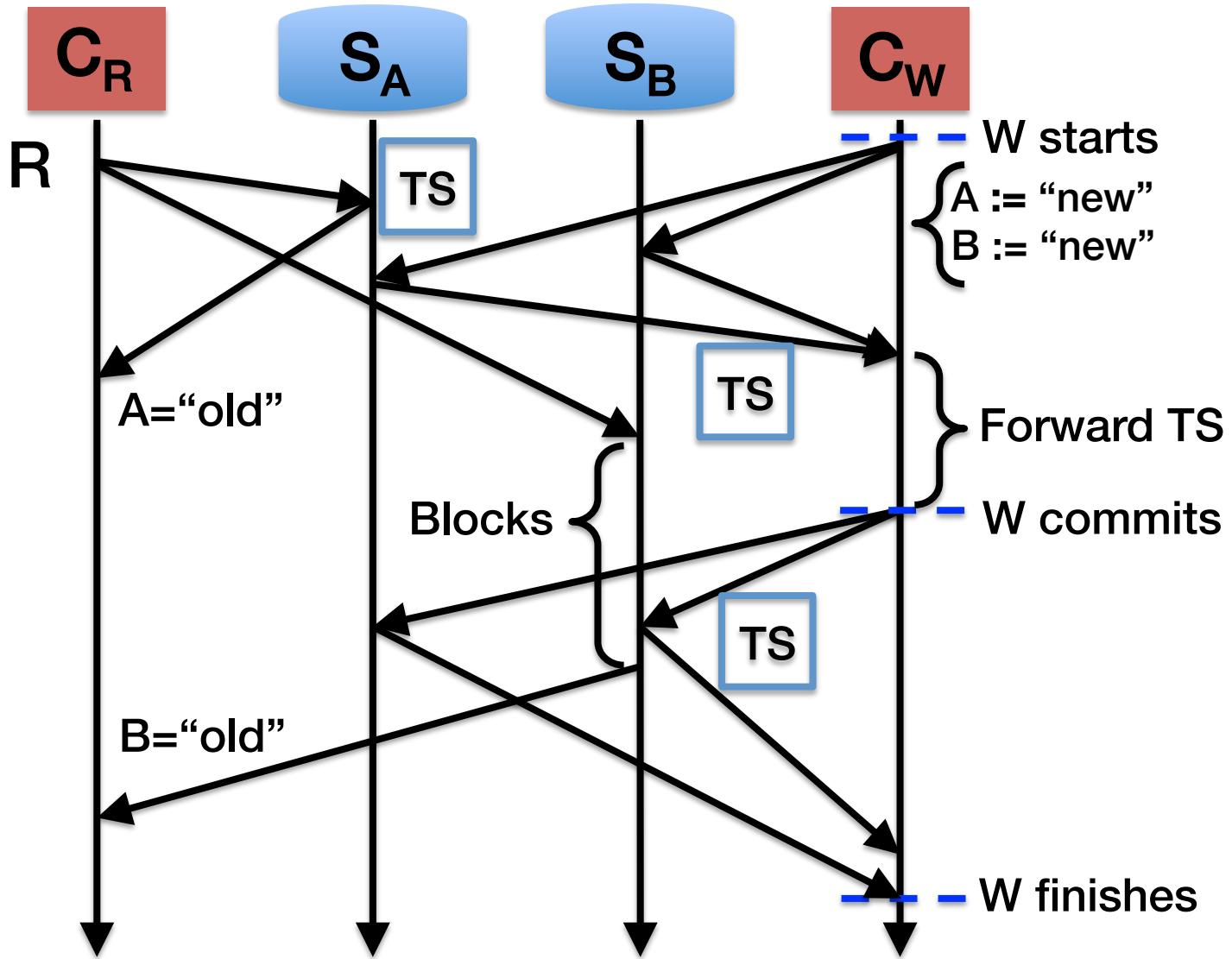
Rococo-SNOW (S+O+W)



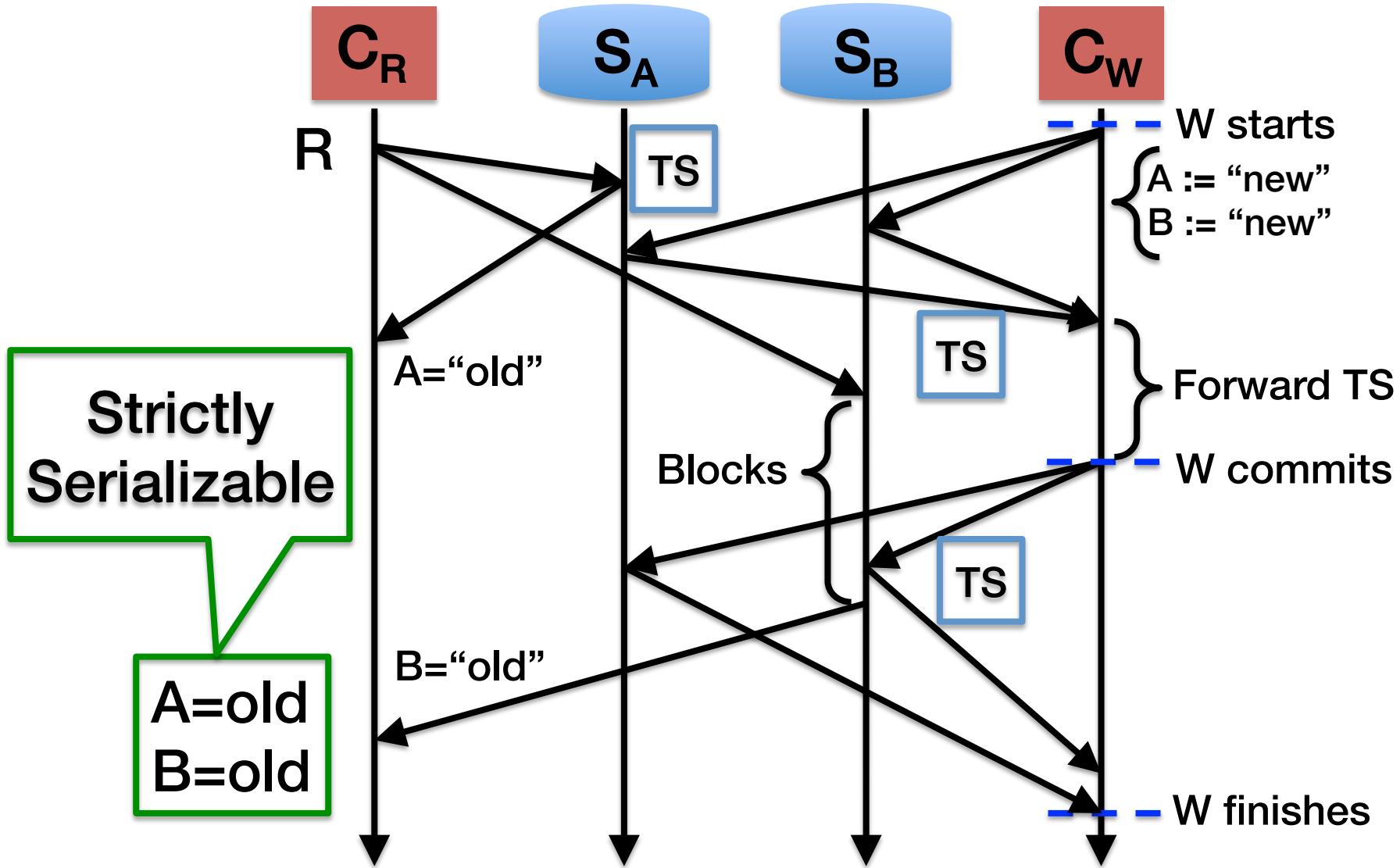
Rococo-SNOW (S+O+W)



Rococo-SNOW (S+O+W)



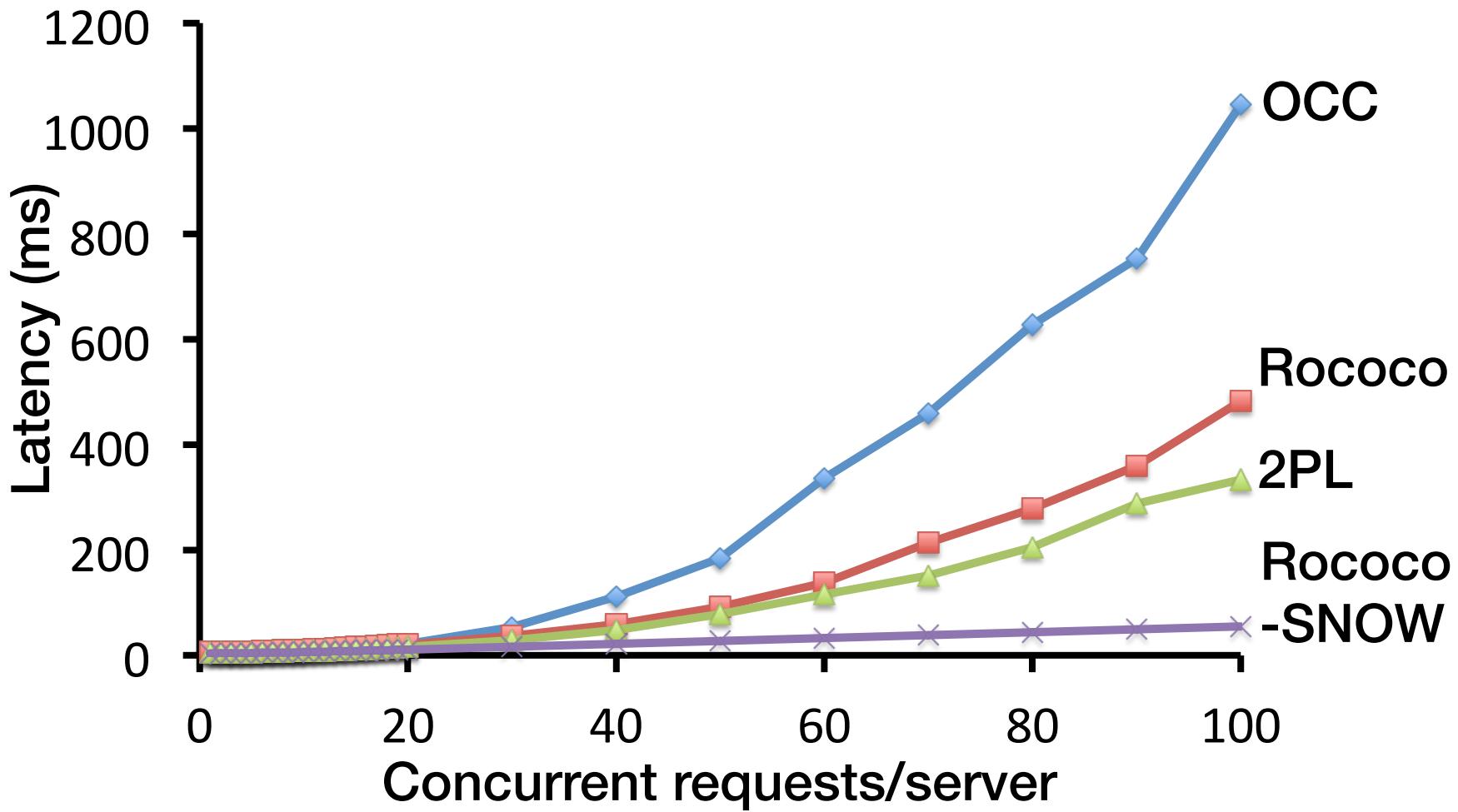
Rococo-SNOW (S+O+W)



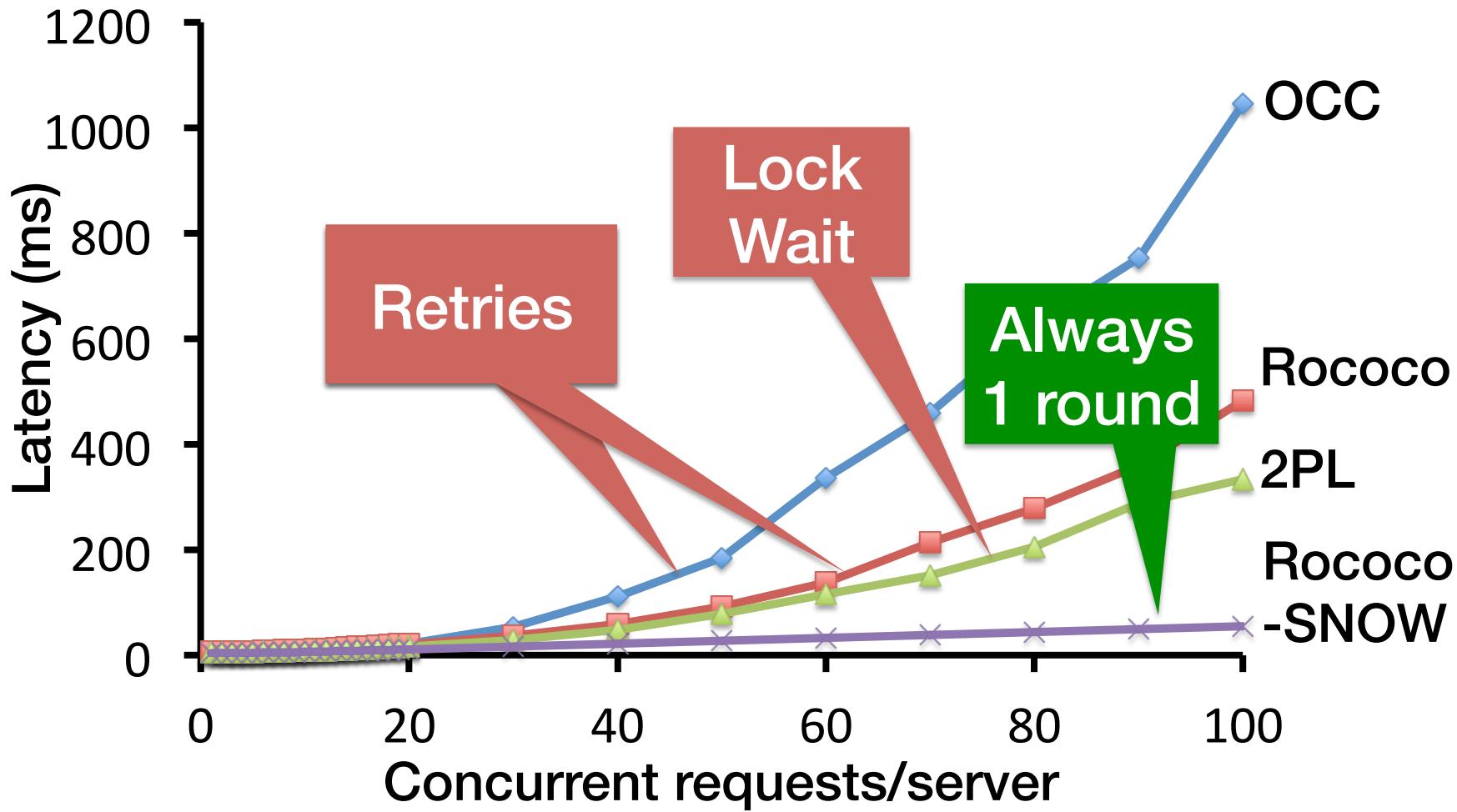
Evaluation of Rococo-SNOW

- 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>

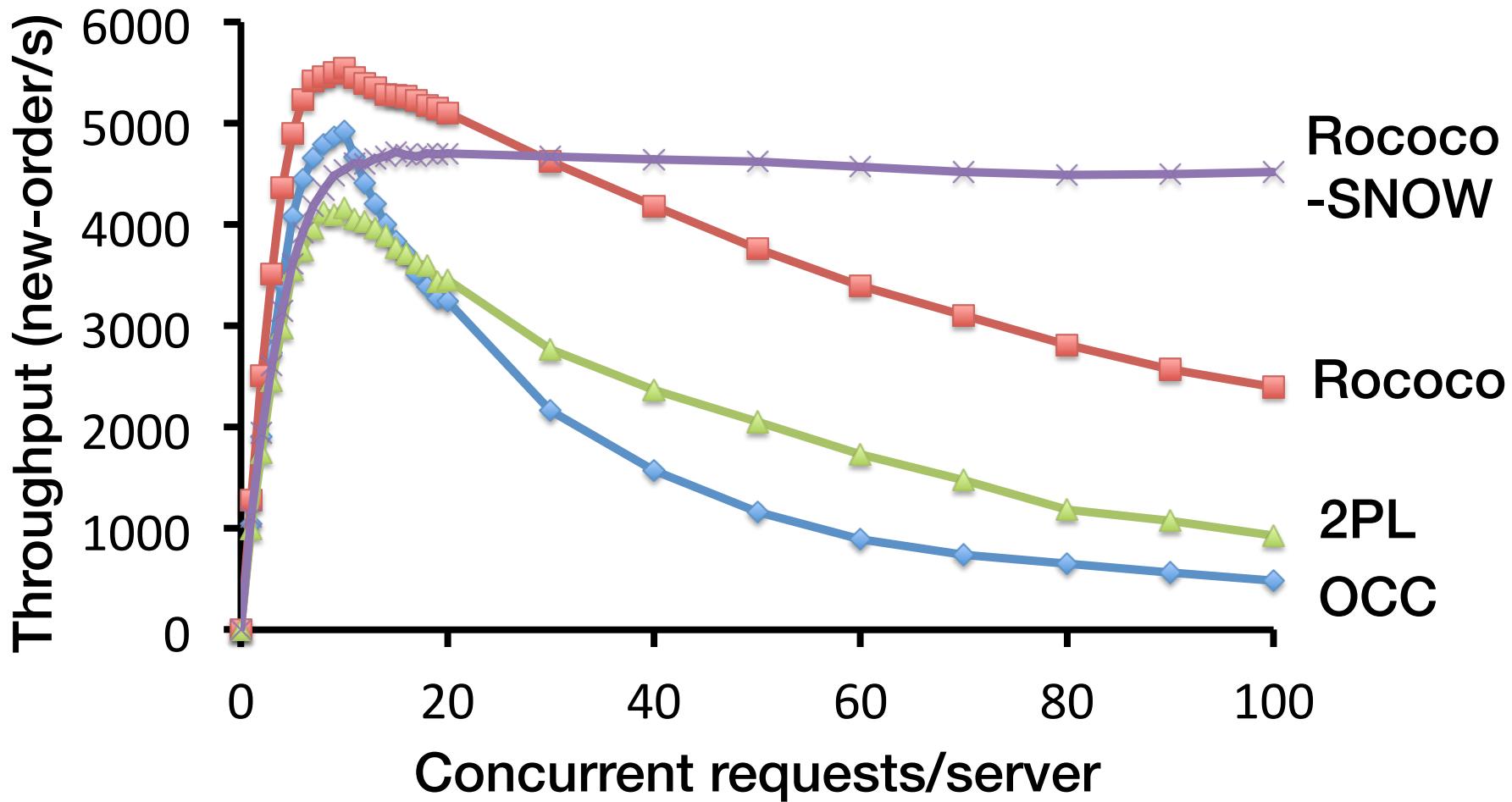
Significantly Lower Latency for Read-Only Txn



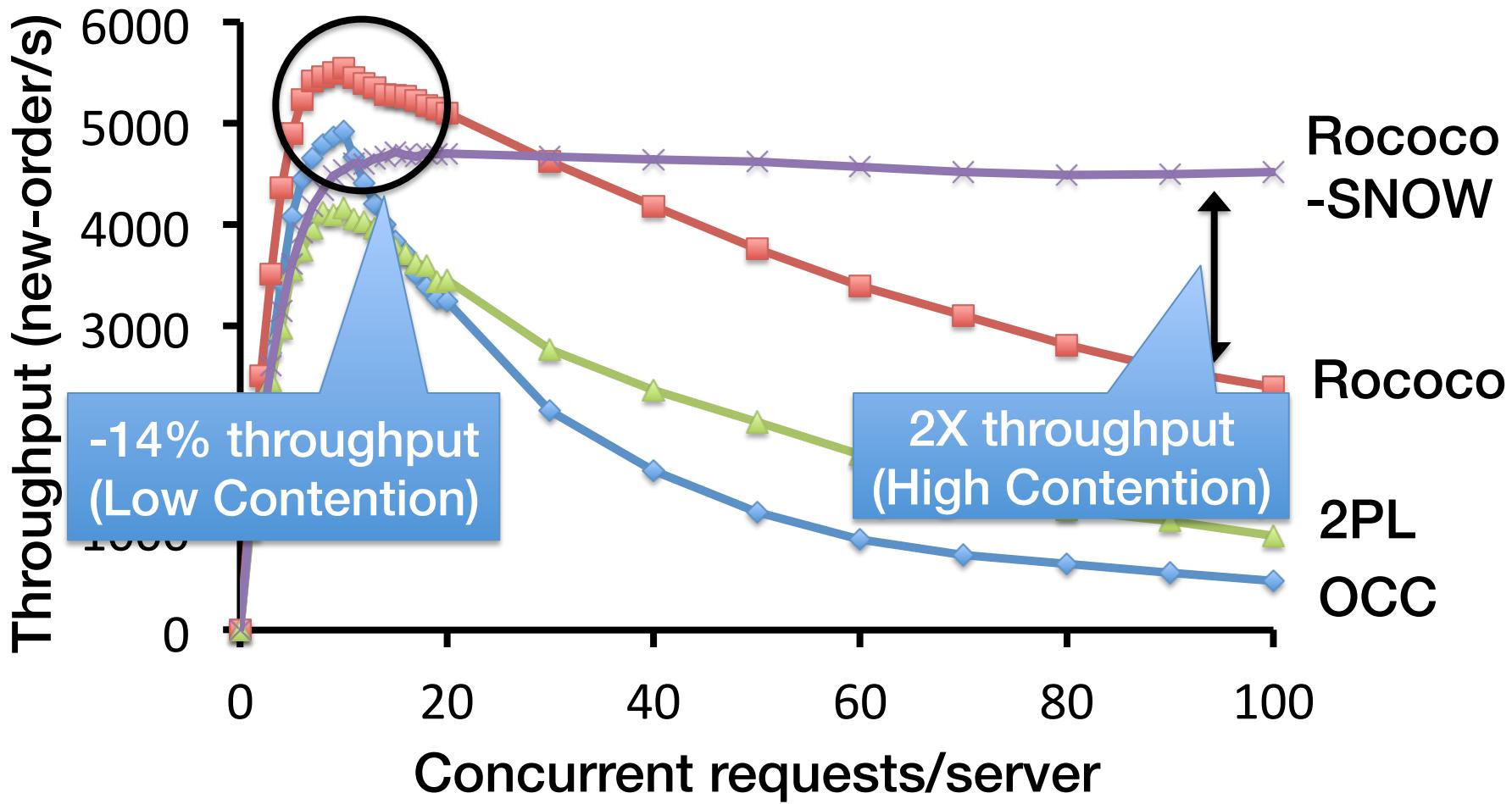
Significantly Lower Latency for Read-Only Txn



Higher Throughput under High Contention



Higher Throughput under High Contention



Conclusion

- The SNOW Theorem for read-only txns
 - Impossible to have all of the SNOW properties
- 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