WREN:
A Fast and Scalable Transactional Causally Consistent Geo-Replicated Key-Value Store

Diego Didona, Kristina Spirovska, Willy Zwaenepoel
RainbowFS workshop
Paris, May 3rd
Geo-replicated partitioned key-value data store
Consistency

- Eventual Consistency
- Causal Consistency
- Linearizability (Strong Consistency)

Performance vs. Consistency Guarantees
I’ll miss the train ;(

Robert: Happy for you!

Alice: Yes! Caught it 😊

Bob: Happy for you!
- Transactions read from a **causally consistent snapshot**

- Transactions updates are visible **atomically** in a DC
Outline

- Introduction
- Our work - Wren
- Related work
- Summary
Limitations of State-of-the-art Systems

- Dependency tracking
  - Metadata scalability

- Update timestamping
  - Clock skew induced latency
## Main Contributions

<table>
<thead>
<tr>
<th>1. Constant Metadata</th>
<th>2. Low Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hybrid Stable Time (HST)</strong></td>
<td><strong>Hybrid Logical/Physical Clocks</strong></td>
</tr>
<tr>
<td>A novel dependency tracking and stabilization protocol</td>
<td>Loose synchronization of physical clocks without suffering from clock skew</td>
</tr>
<tr>
<td>Addresses metadata scalability</td>
<td>Addresses clock skew induced latency</td>
</tr>
</tbody>
</table>
Main Contributions

1. Constant Metadata

Hybrid Stable Time (HST)
A novel dependency tracking and stabilization protocol

2. Low Latency

Hybrid Logical/Physical Clocks
Loose synchronization of physical clocks without suffering from clock skew
Hybrid Stable Time (HST)

- Only two scalar timestamps

1. Local Dependency Time
   - Tracks the dependencies on local items

2. Remote Stable Time
   - Summarizes dependencies on remote items
Existing dependency tracking methods

### Dependency vectors

- **Cure [ICDCS’16]**
  - Fresh, non-blocking snapshot
  - Metadata size $O(#DCs)$

### Single timestamp

- **GentleRain [SOCC’14]**
  - Metadata size $O(1)$
  - Inter DC sync to install snapshot
Single Scalar: GentleRain

DC 0

DC 1

DC 2

Client

Lower Bound:

Write

Read

CLIENT dependency time

5

O(1)

15

BLOCK until:
lower bound >= 15

15

15

15
Dependency vectors: Cure

DC 0

DC 1

DC 2

Client

Lower Bound: [5, 7, 15]

O(n)

Client dependency time
Dependency vectors: Cure

Client dependency time

Client

DC 0

DC 1

DC 2

Write

Read

Lower Bound: [5, 7, 15]

O(n)

[5, 7, 15]

NO BLOCKING

Client Write Read

5 7 15

[5, 7, 15]
Local and Remote Time: Wren

Client dependency time

Lower Bound: [5, 15]

O(1)
Local and Remote Time: Wren

Client dependency time

15

[5, 15]

O(1)

Lower Bound:

[5, 15]

NO BLOCKING

Client

DC 0

DC 1

DC 2
Hybrid Stable Time (HST)

• Only two scalar timestamps

1. Local Dependency Time
   Tracks the dependencies on local items

2. Remote Stable Time
   Summarizes dependencies on remote items
Remote Stable Time (RST)

• Computed periodically

• Lower bound on updates from remote DCs

• No additional inter DCs sync
• Hits sweet spot in the meta-data size vs performance spectrum
Main Contributions

1. Constant Metadata
   Hybrid Stable Time (HST)
   A novel dependency tracking and stabilization protocol

2. Low Latency
   Hybrid Logical/Physical Clocks
   Loose synchronization of physical clocks without suffering from clock skew
Existing update timestamping methods

**Physical Clocks PC**

+ (Loose) clock synchronisation enables efficient dependency tracking

- Clock skew introduces uncertainty (latencies)

**Logical Clocks LC**

+ Capture dependency among events easily

- No synchronisation makes dependency tracking more costly
Hybrid Logical/Physical clocks (HCL)

- *Logical Physical Clocks*, OPODIS’14

- Best of both worlds
  - Captures the causality relationship
  - Inherits loose synchronization of PC
  - No clock skew as LC
Hybrid Logical/Physical clocks (HCL)

Physical component

Logical component

\[ p : l \]
Clock skew problem

Invariant: timestamps must reflect causality

Client’s dep. time < x2’s update time

Write(x2)

Wait until \( t \) < \( x2 \)'s update time

Physical clock: \( t \)

Client dependency time: \( t \)
Clock skew solution

**Invariant**: timestamps must reflect causality

Client’s dep. time < x2’s update time

Client

Write(x₂)

10:0

p

5:0

Update hybrid clock

physical clock

hybrid clock

client dependency time
Clock skew solution

Client

\text{Write}(x_2)

\text{physical clock}

\text{hybrid clock}

\text{client dependency time}
Availability

**HST** always tracks stable remote dependencies

The system will continue to work, everything is there locally

Do not block in case of network partitions
Outline

- Introduction
- Our work - Wren
- Related work
- Summary
<table>
<thead>
<tr>
<th>System</th>
<th>Low Latency</th>
<th>Metadata</th>
<th>Availability (Transact.)</th>
<th>Snapshot freshness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cure [ICDCS’16]</td>
<td>Clock skew</td>
<td>#DCs</td>
<td>Yes</td>
<td>Higher</td>
</tr>
<tr>
<td>GentleRain [SOCC’14]</td>
<td>Clock skew, Inter-DC syn</td>
<td>O(1)</td>
<td>No</td>
<td>Lower</td>
</tr>
<tr>
<td>Occult [NSDI’17]</td>
<td>Inter DC sync</td>
<td>O(#DCs)</td>
<td>No</td>
<td>Highest</td>
</tr>
<tr>
<td>Wren</td>
<td>Yes</td>
<td>O(1)</td>
<td>Yes</td>
<td>Lower</td>
</tr>
</tbody>
</table>
Our work: Wren

The first transactional causally consistent geo-replicated system that at the same time has:

- Constant metadata
- Low latency
- Always-available
• Wren: The first transactional causally consistent geo-replicated system that at the same time has:
  • Constant metadata
    • Hybrid Stable Time
  • Low latency
    • Hybrid Logical/Physical Clocks
  • Always-available

Thank you!