

# **Eiger:** Stronger Semantics for Low-Latency Geo-Replicated Storage

**Wyatt Lloyd\***

**Michael J. Freedman\***

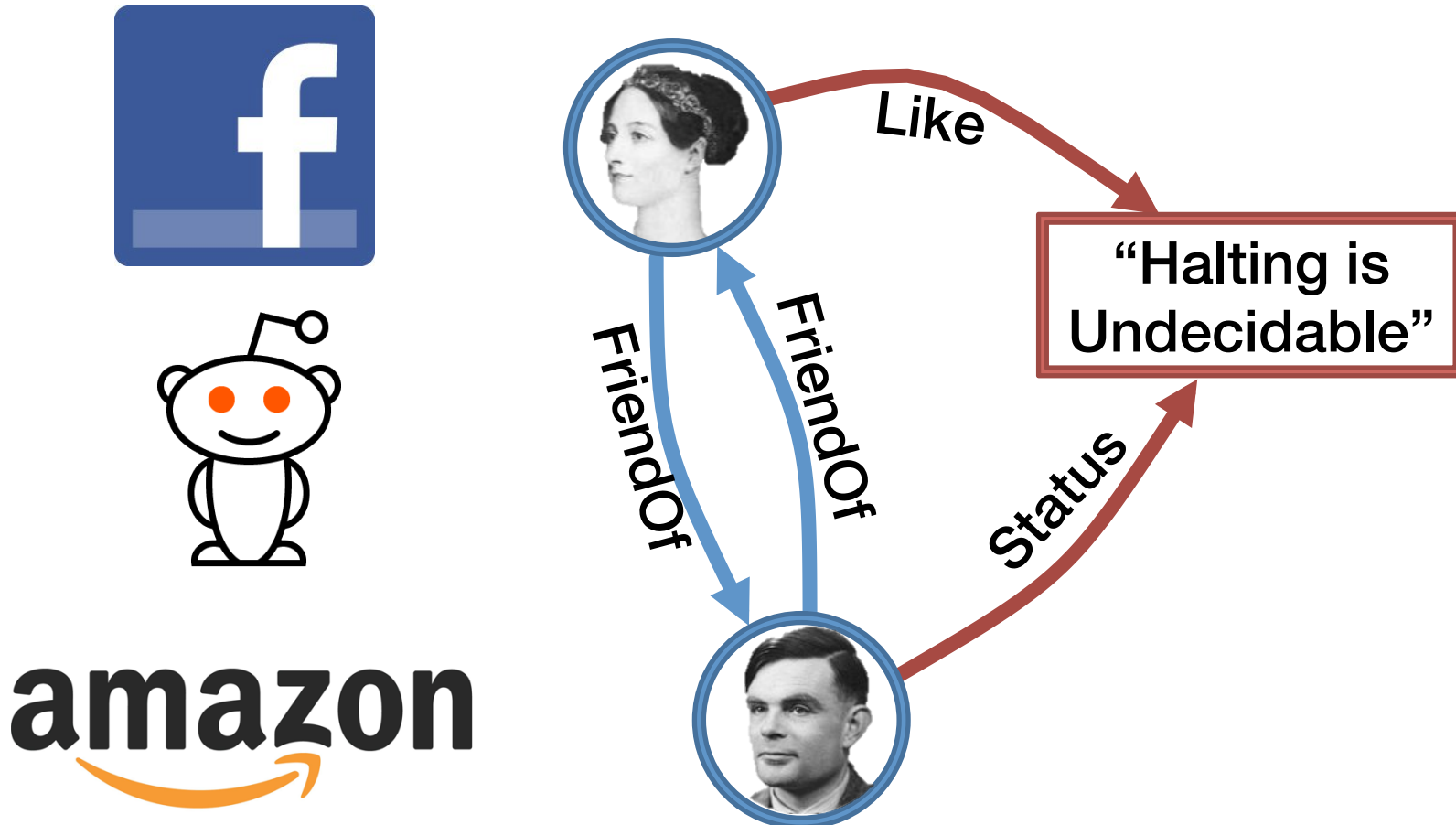
**Michael Kaminsky†**

**David G. Andersen‡**

\*Princeton, †Intel Labs, ‡CMU

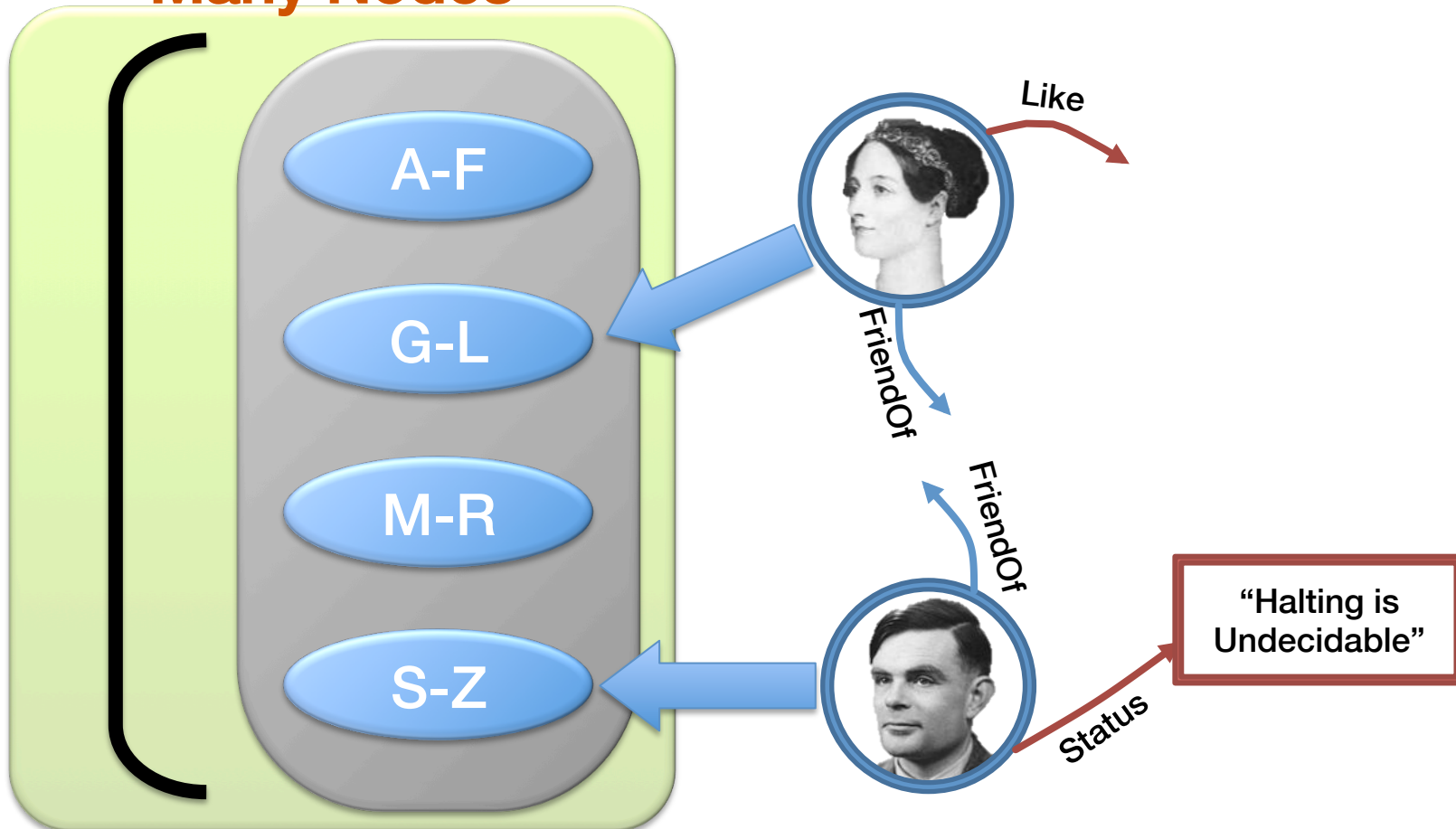
# Geo-Replicated Storage

is the backend of massive websites



# Storage Dimensions

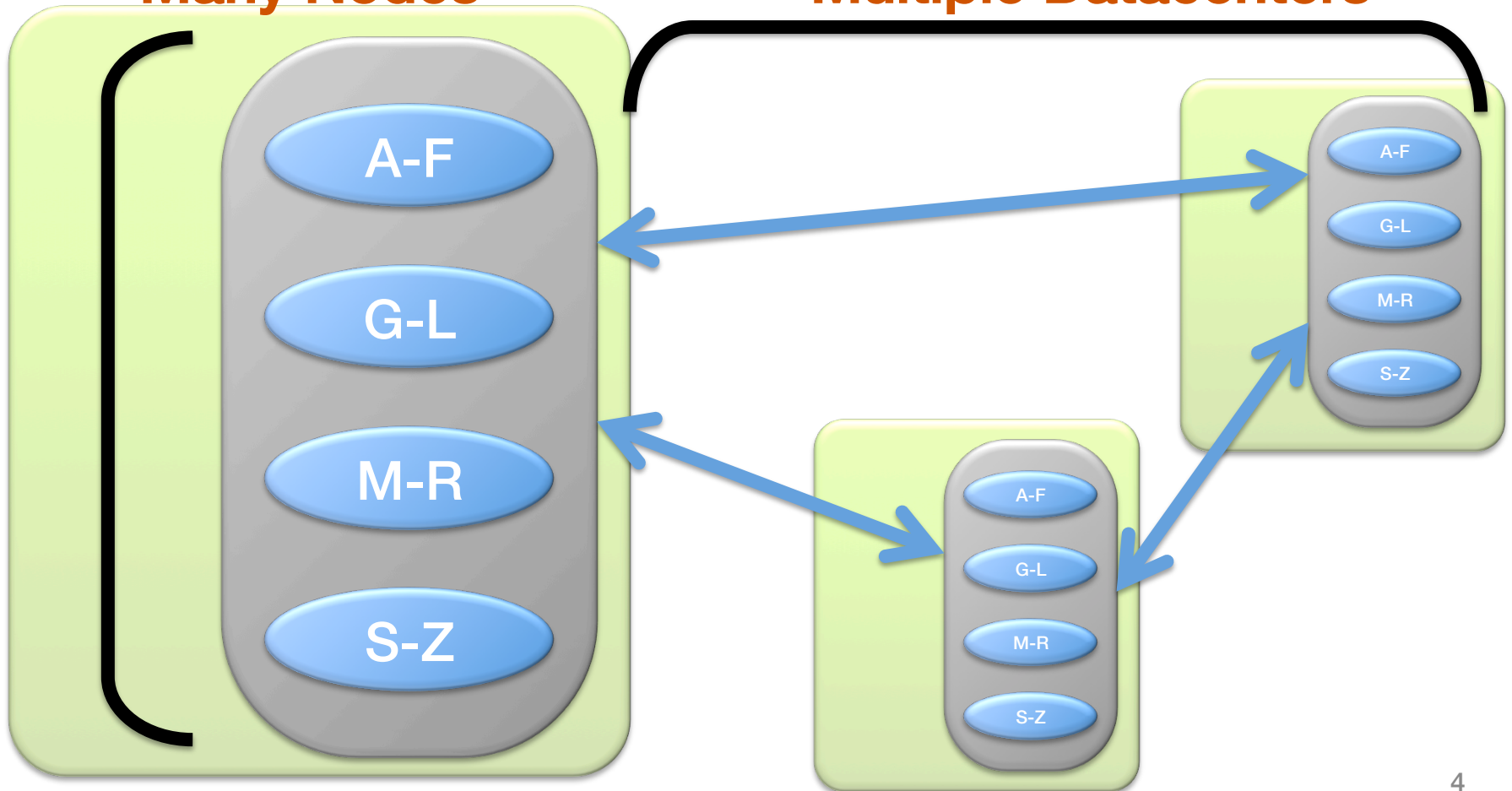
Shard Data Across  
**Many Nodes**



# Storage Dimensions

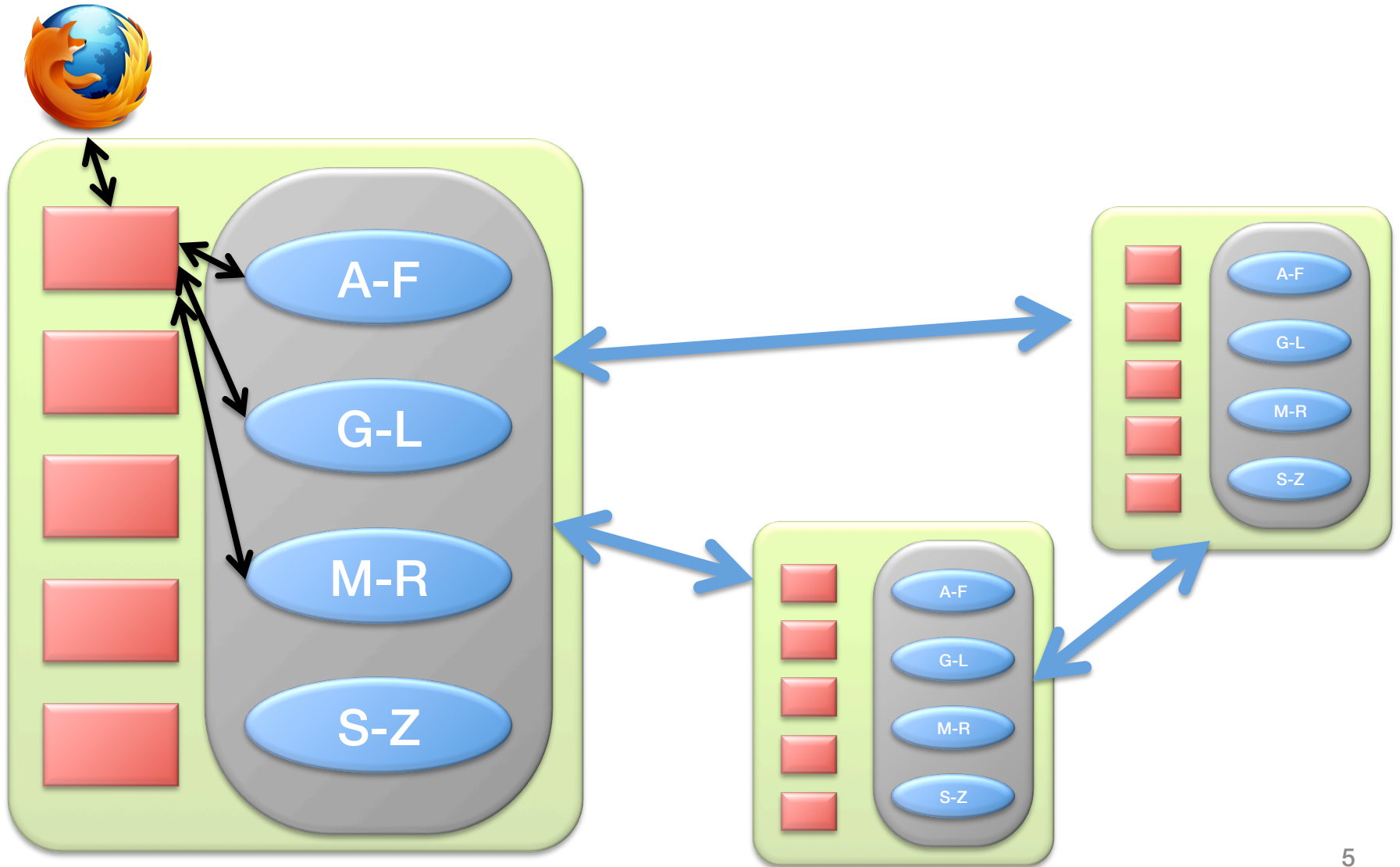
Shard Data Across  
**Many Nodes**

Data Geo-Replicated In  
**Multiple Datacenters**





# Sharded, Geo-Replicated Storage



# Strong Consistency or Low Latency

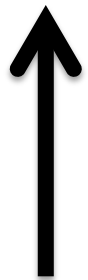


## Low Latency

- Improves user experience
- Correlates with revenue



**Fundamentally in Conflict**  
[LiptonSandberg88, AttiyaWelch94]



## Strong Consistency

- Obey user expectations
- Easier for programmers



# Strong Consistency or Low Latency

Megastore [SIGMOD '08]

Dynamo [SOSP '07]

Spanner [OSDI '12] →

← COPS [SOSP '11]

Gemini [OSDI '12] →

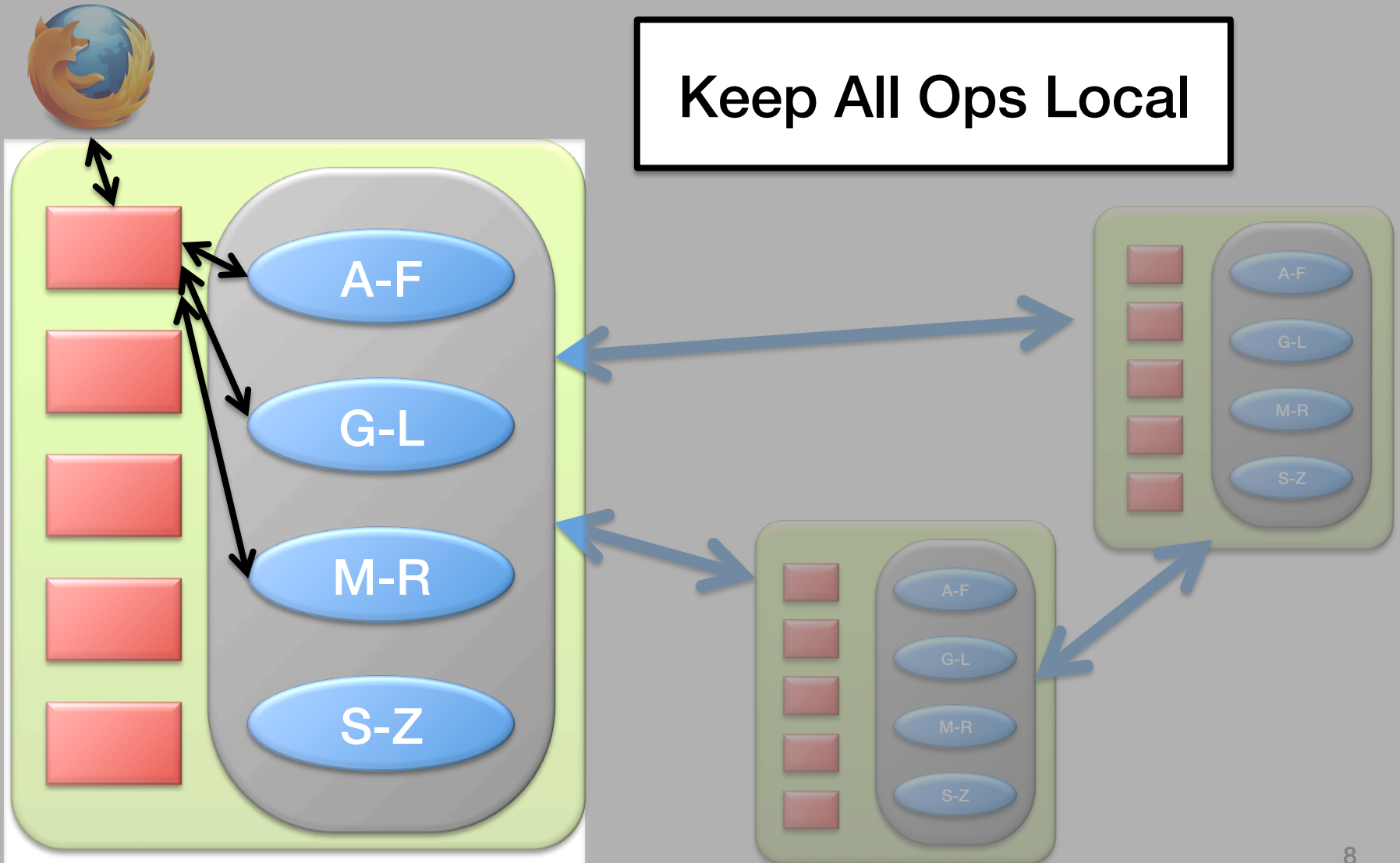
← **Eiger**

Walter [SOSP '11] →

**Causal+ Consistency**  
**Rich Data Model**  
**Read-only Txns**  
**Write-only Txns**

**Obey user expectations**  
**Easier for programmers**

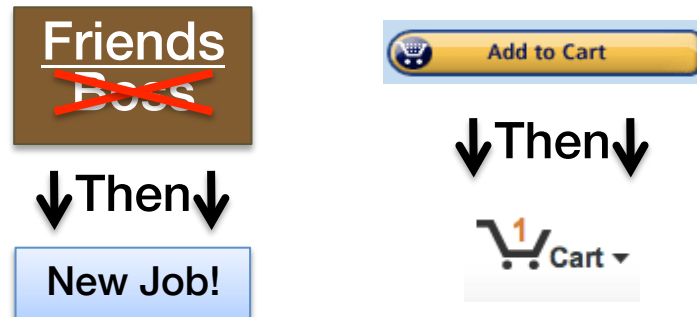
# Eiger Ensures Low Latency



# Causal+ Consistency Across DCs

- If  $A$  happens before  $B$ 
  - Everyone sees  $A$  before  $B$

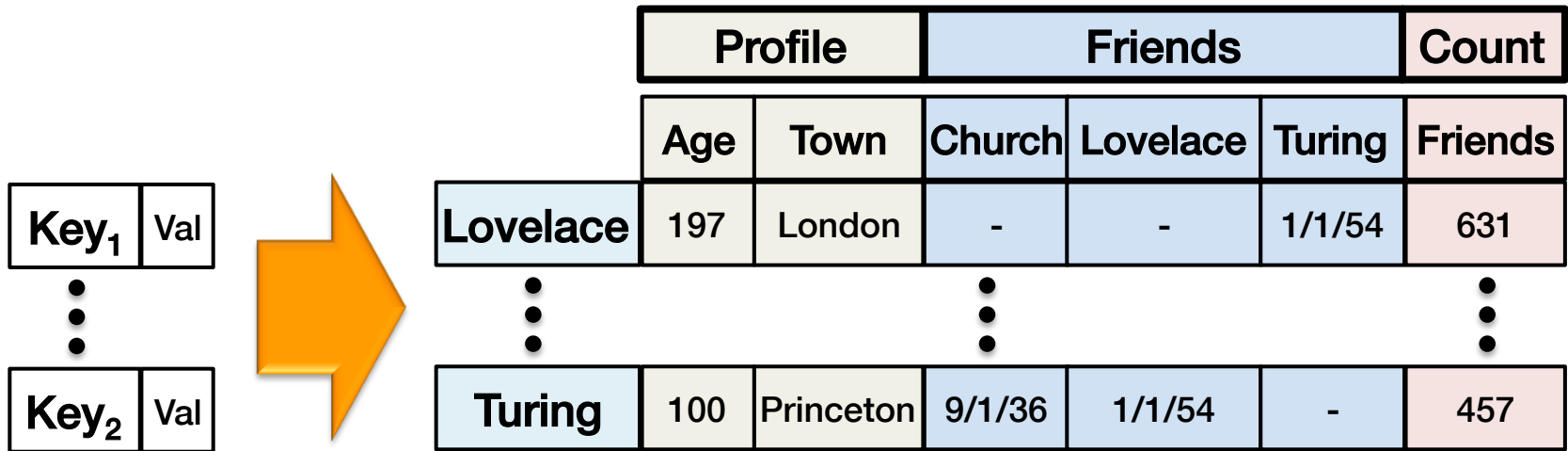
- Obeys user expectations



- Simplifies programming



# Causal For Column Families

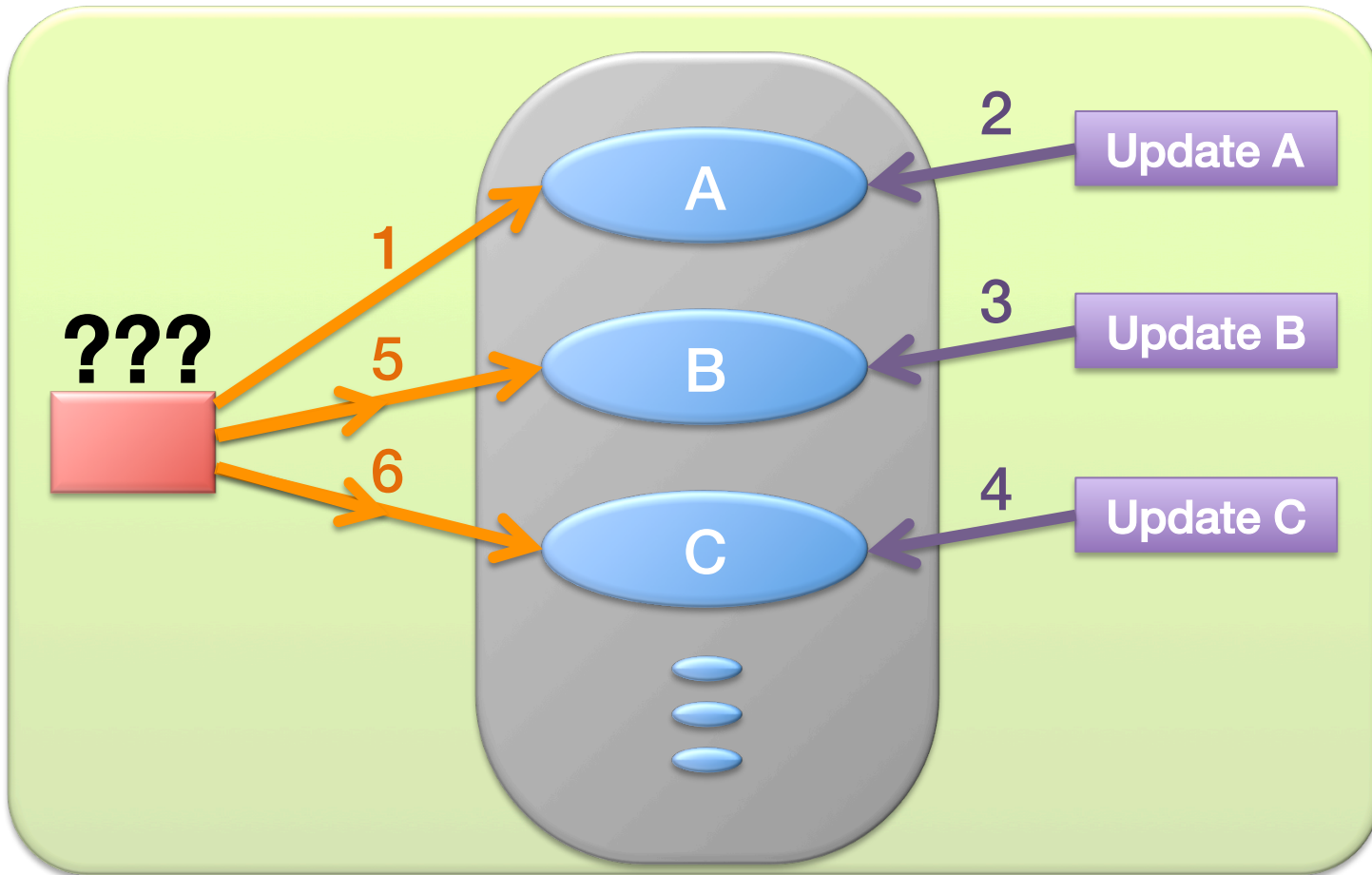


- Operations update/read many columns
- Range query columns concurrent w/ deletes
- Counter columns
- See paper for details



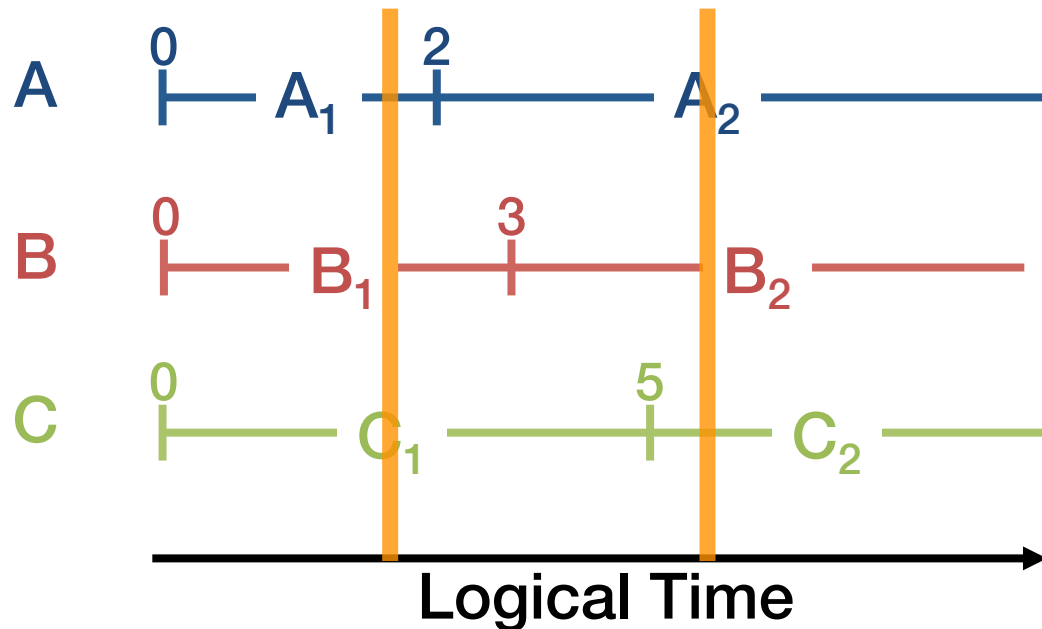
# Viewing Data Consistently Is Hard

Asynchronous requests + distributed data = ??????



# Read-Only Transactions

- Logical time gives a global view of data store
  - Clocks on all nodes, carried with all messages
- Insight: Store is consistent at all logical times

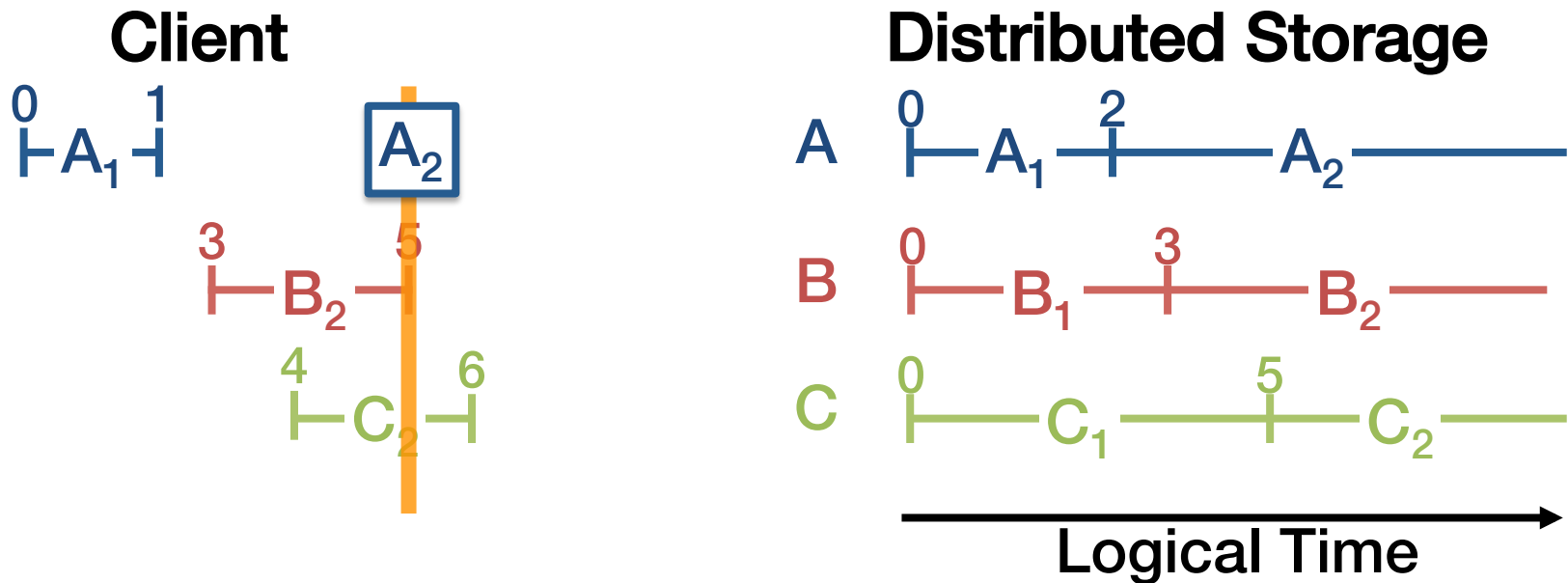


# Read-Only Transactions



- **Extract consistent up-to-date view of data**
  - Across many servers
- **Challenges**
  - **Scalability**
    - Decentralized algorithm
  - **Guaranteed low latency**
    - At most 2 parallel rounds of local reads
    - No locks, no blocking
  - **High performance**
    - Normal case: 1 round of reads

# Read-Only Transactions

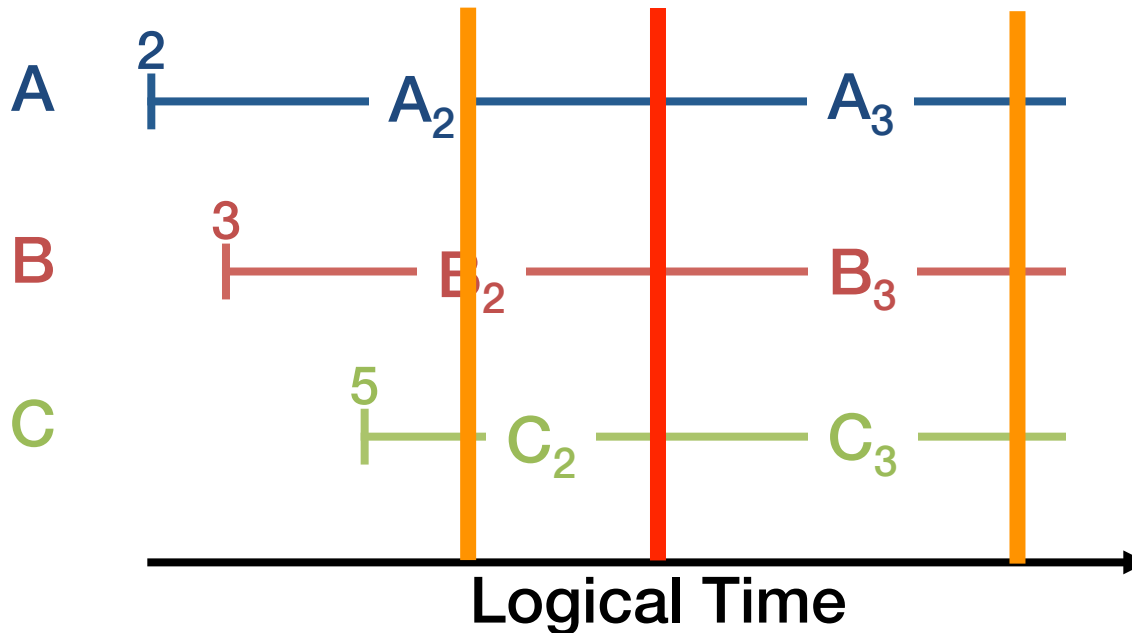
- Round 1: Optimistic parallel reads
- Calculate *effective time*
- Round 2: Parallel read\_at\_times



# Transaction Intuition

- Read-only transactions 
  - Read from a single logical time
- Write-only transactions 
  - Appear at a single logical time

**Bonus:**  
Works for  
Linearizability



# Eiger Provides

- ✓ Low latency
- ✓ Rich data model
- ✓ Causal+ consistency
- ✓ Read-only transactions
- ✓ Write-only transactions

**But what does all this cost?**

**Does it scale?**



# Eiger Implementation

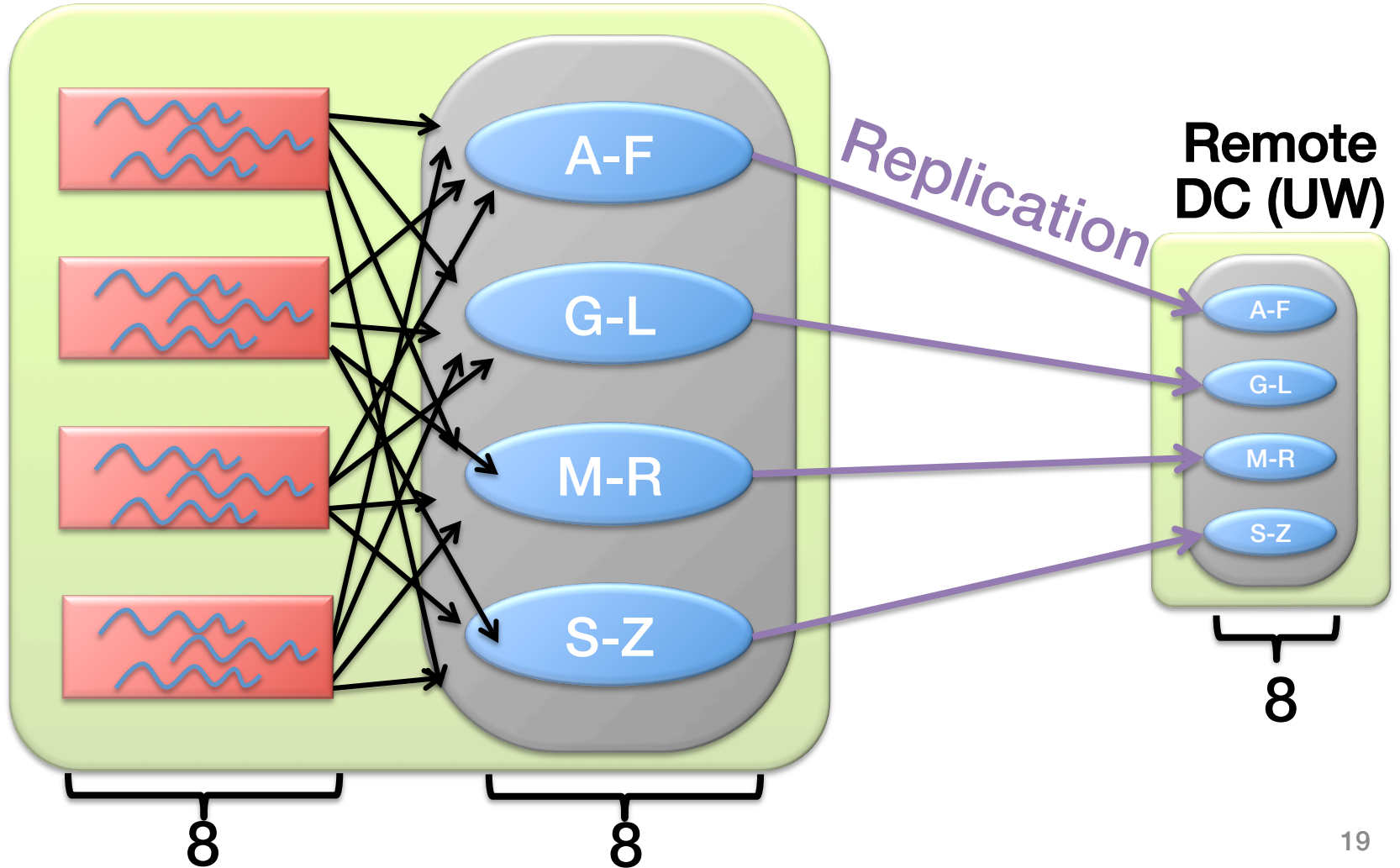
- Fork of open-source Cassandra
- +5K lines of Java to Cassandra's 75K
- Code Available:
  - <https://github.com/wlloyd/eiger>

# Evaluation

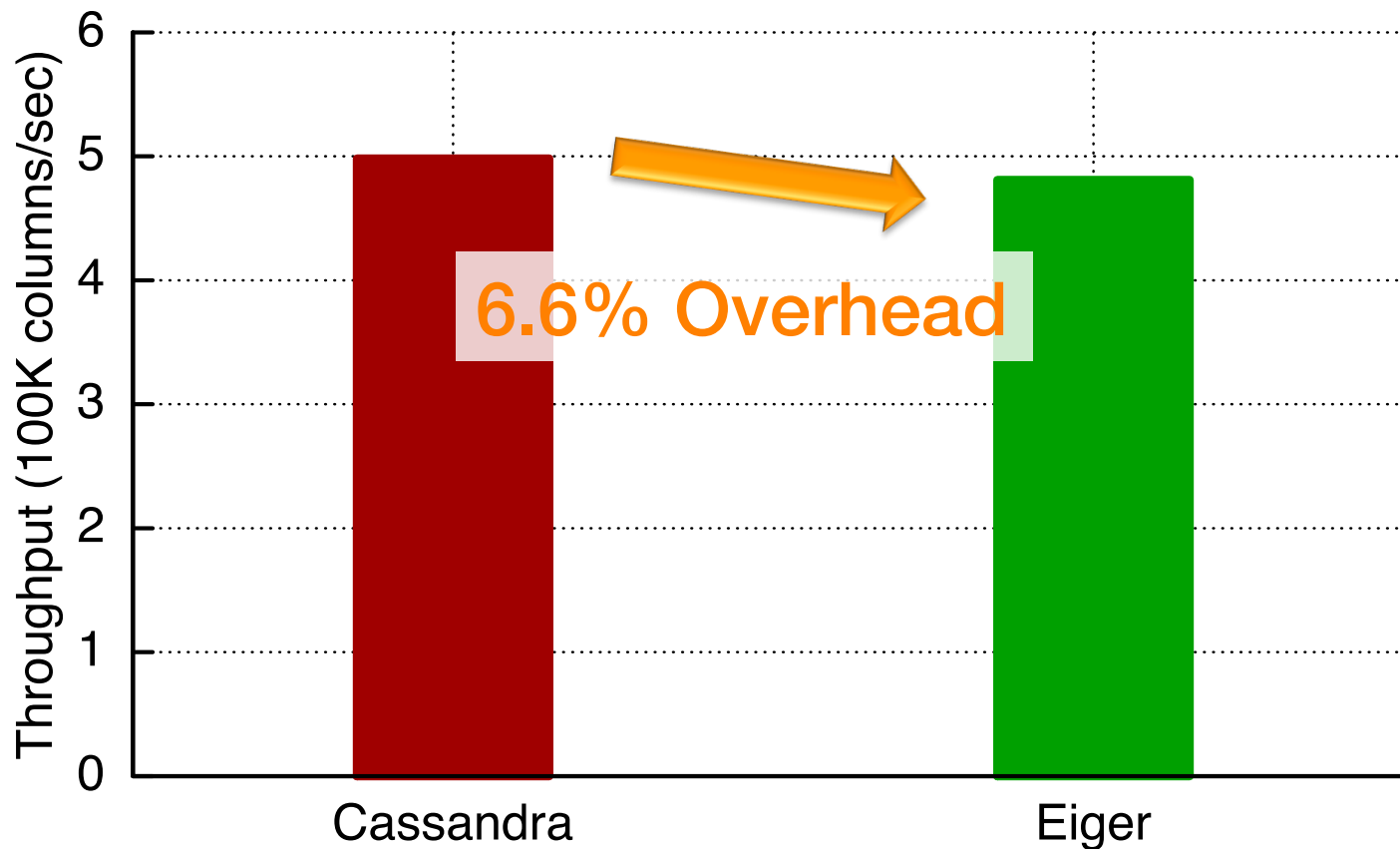
- **Cost of stronger consistency & semantics**
  - Vs. eventually-consistent Cassandra
  - Overhead for real (Facebook) workload
  - Overhead for state-space of workloads
- **Scalability**

# Experimental Setup

## Local Datacenter (Stanford)

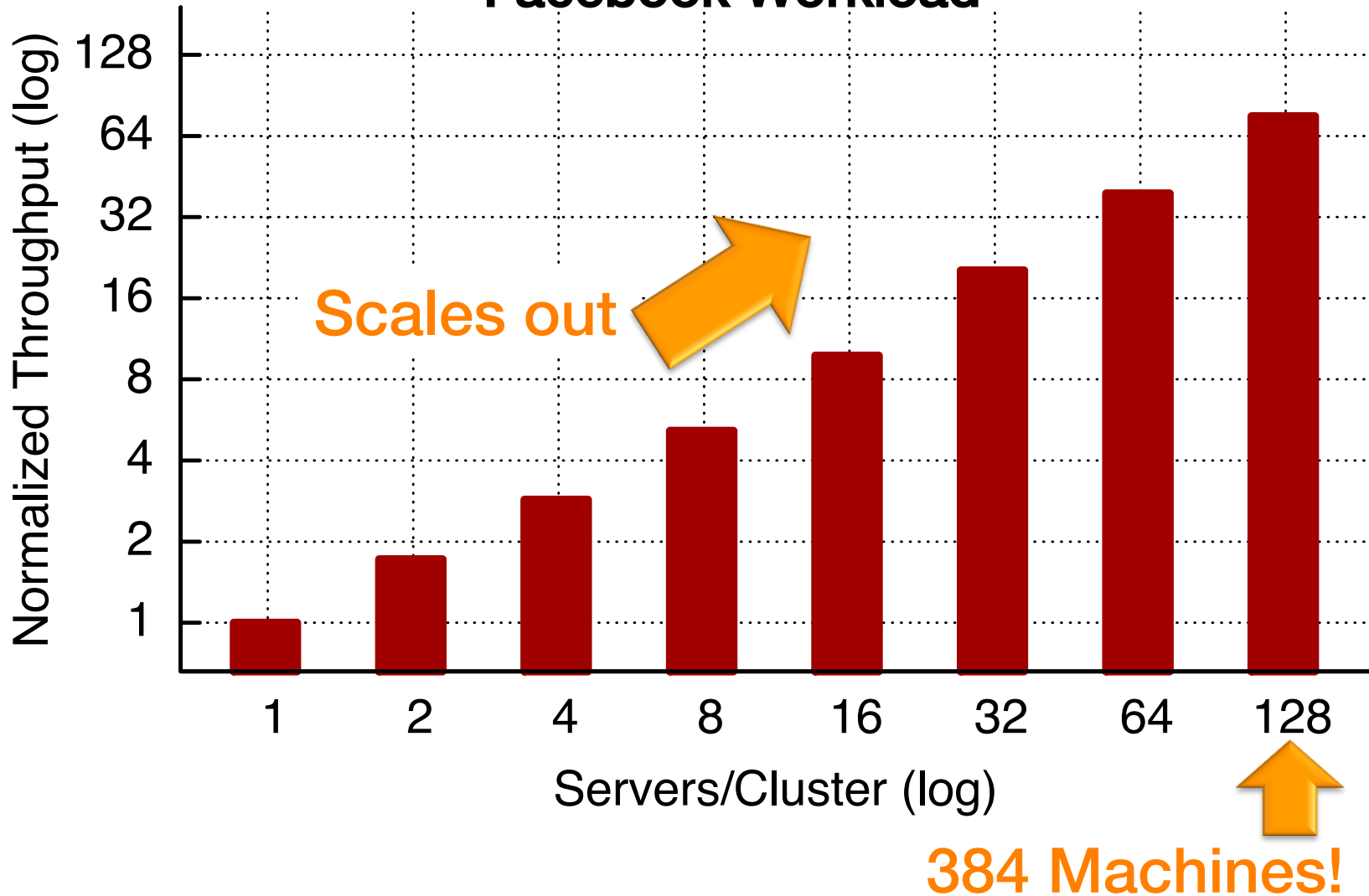


# Facebook Workload Results



# Eiger Scales

## Facebook Workload



# Improving Low-Latency Storage

COPS → Eiger

Data model	Key-Value	→	Column-Family
Read-only Txns	Causal stores	→	All stores
Write-only Txns	None	→	Yes
Performance	Good	→	Great
DC Failure	Throughput degradation	→	Resilient



# Eiger

- **Low-latency geo-replicated storage**
  - Causal+ for column families
  - Read-only transactions
  - Write-only transactions
- **Demonstrated in working system**
  - Competitive with eventual
  - Scales to large clusters
  - <https://github.com/wlloyd/eiger>