PNUTS: Yahoo!'s Hosted Data Serving Platform

Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver and Ramana Yerneni

What is needed from current DBMS?

- Web applications need:
  - Scalability
  - Geographic scope
  - High availability
- Web applications typically have:
  - Simplified query needs
    - No joins, aggregations
  - Relaxed consistency needs
    - Applications can tolerate stale or reordered data

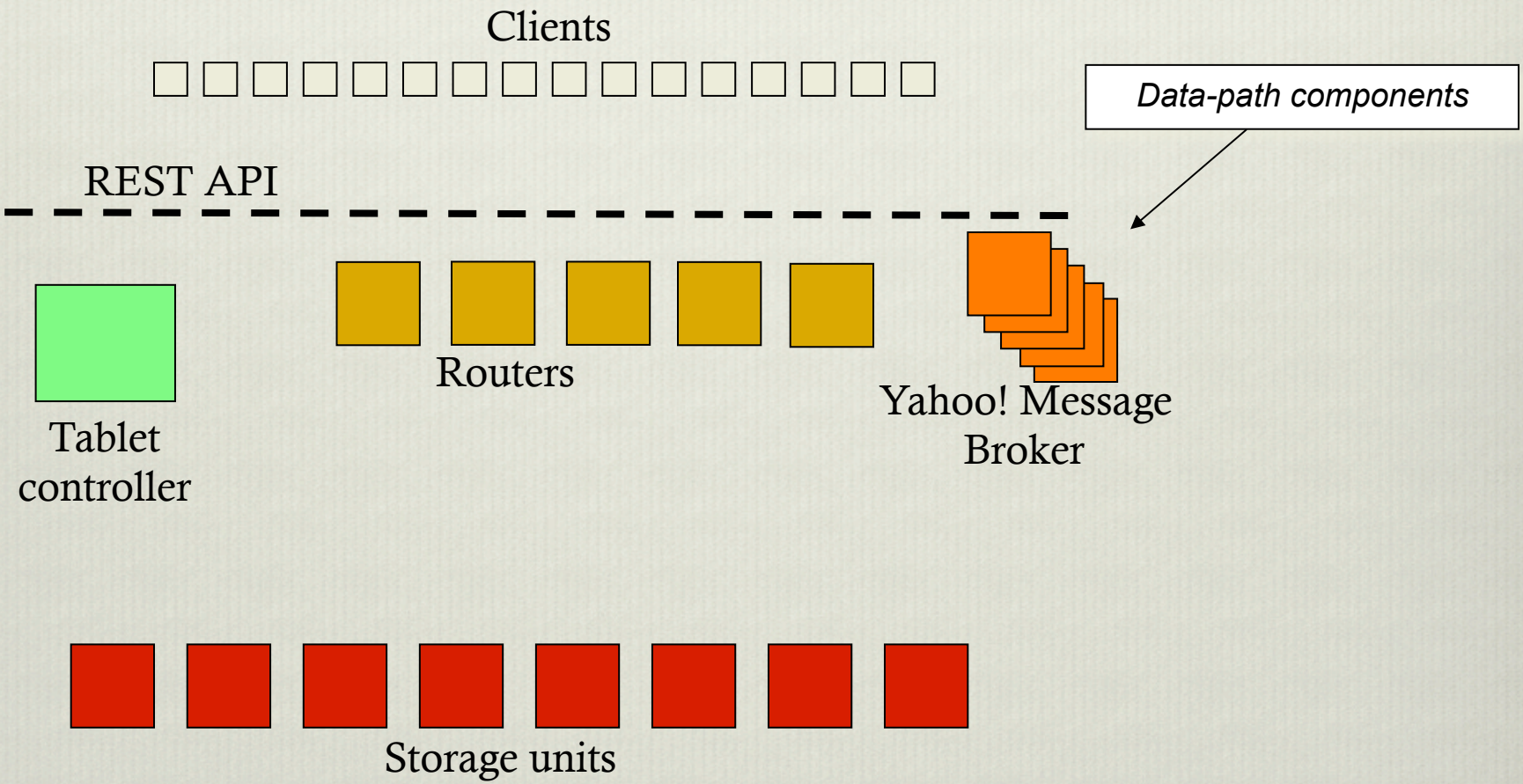
PNUTS

- PNUTS, a massively parallel and geographically distributed database system for Yahoo!'s web applications

Outline

- Architecture
- Experiment
- Future work
- Critique

Architecture



Ordered table

A

H

P

Z

Name	Price
Apple	1.2
Avocado	2.0
Banana	1.0
Grape	2.5
Kiwi	2.0
Orange	0.8
Strawberry	1.2
Watermelon	1.0

Hash table

Name
Apple
Avocado
Banana
Grape
Kiwi
Orange
Strawberry
Watermelon

Hash function

	Name	Price
ox0000	Grape	2.5
	Apple	1.2
	Banana	1.0
ox132A	Watermelon	1.0
	Kiwi	2.0
	Orange	0.8
ox2C3F	Avocado	2.0
	Strawberry	1.2

Flexible schema

- Arbitrary structures are allowed
- New attributes can be added at any time
  - Records are not required to have values for all attributes

Name	Price

Name	Price	Description

Name	Price	Country	Description

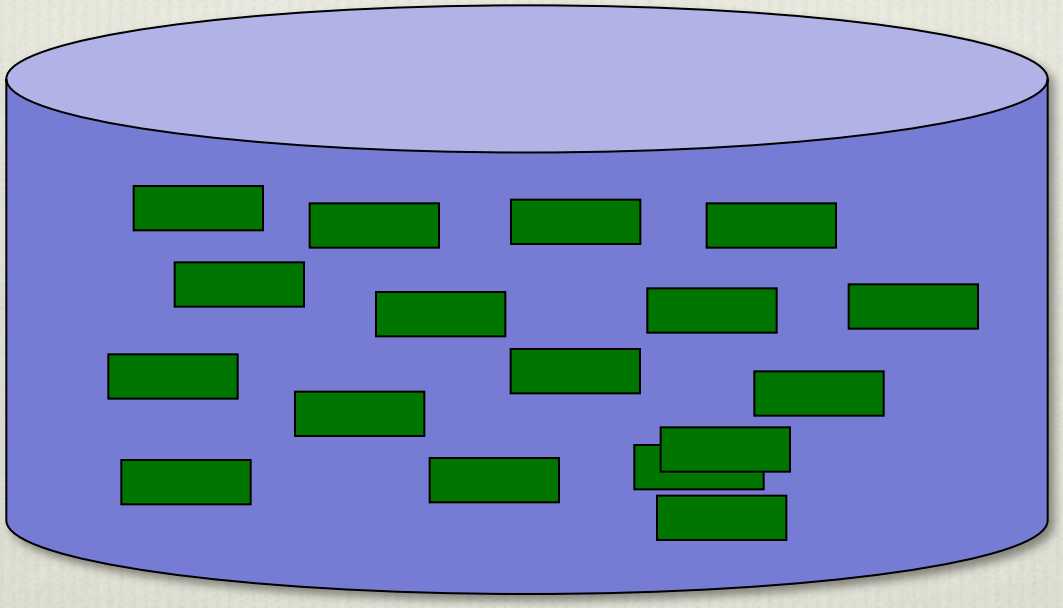
Architecture

- Routers
  - Which table -> which tablet -> which SU
  - Contains a cached copy of the interval mapping
- Tablet controller
  - Owns the mapping
  - Decides when to move tablet, split tablet
  - Not a bottleneck

Tablet Splitting & Balancing

Tables: horizontall partitioned -> tablts

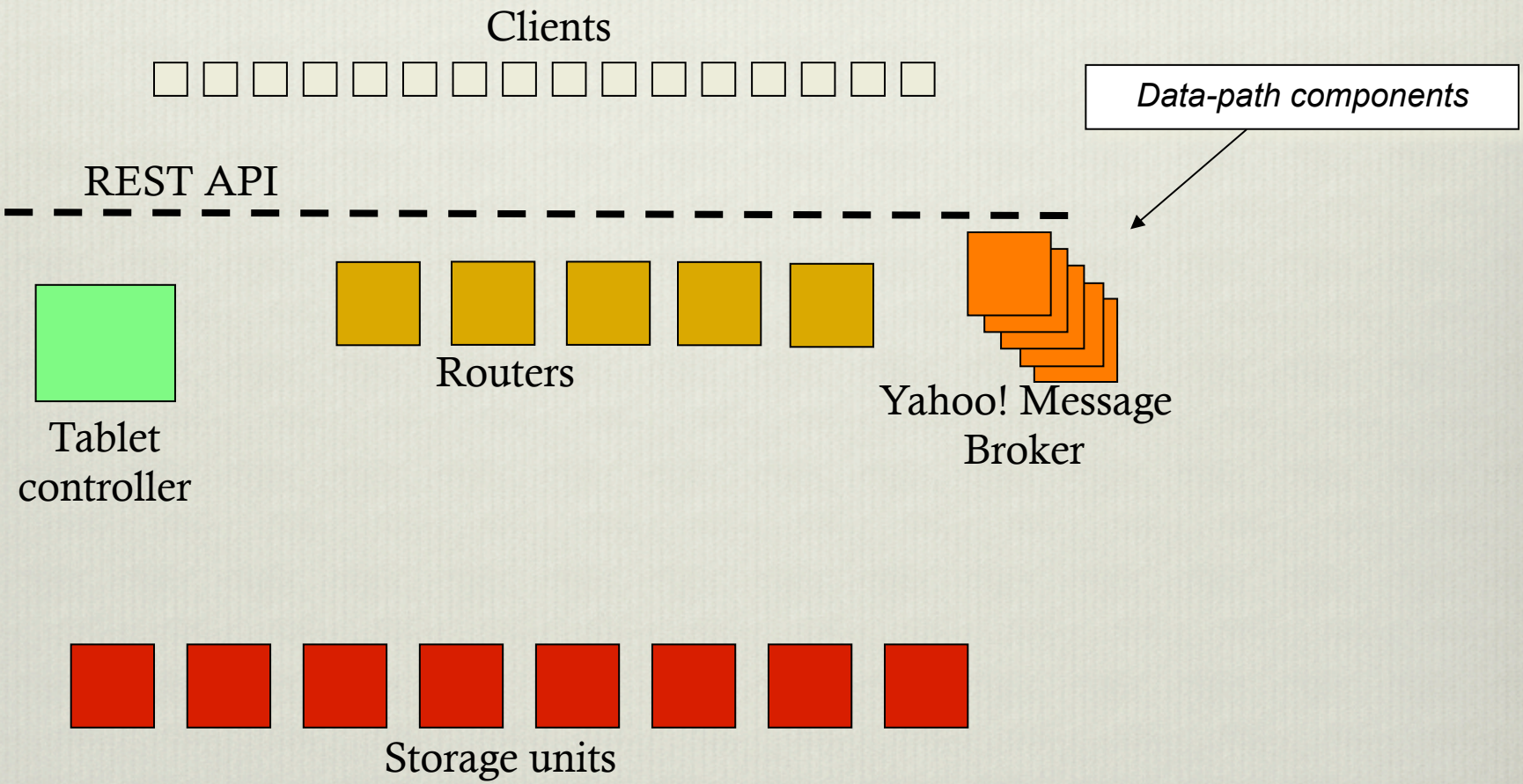
Storage unit may become a hotspot



Tablets may grow over time

Shed load by moving tablets to other servers

Architecture



Architecture - YMB

- Yahoo! Message Broker (YMB)
  - Topic-based pub/sub system
  - Data is considered ‘committed’ when they have been published to YMB.
  - Only partial ordering of published messages

Consistency model

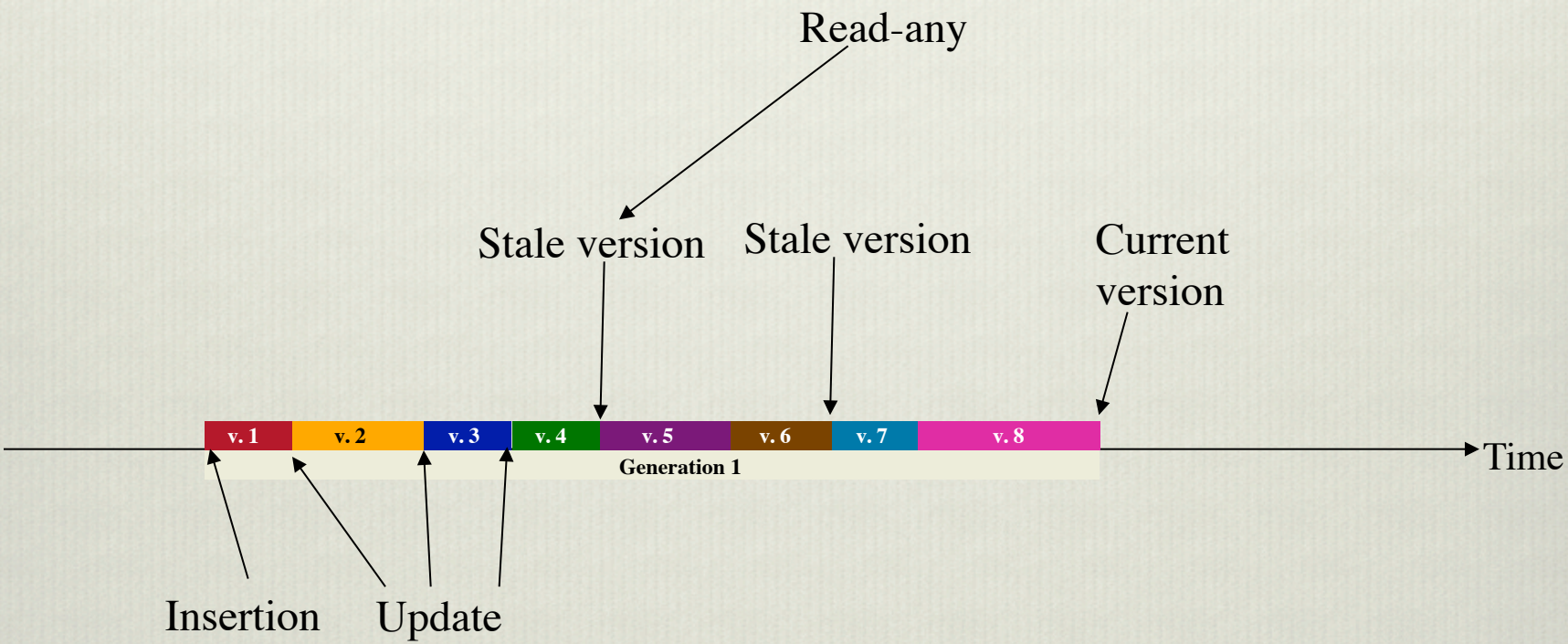
- Per-record timeline consistency

- Per-record mastering
  - Each record is assigned a “master region”
  - May differ between records
  - Updates to the record forwarded to the master region
  - Ensures consistent ordering of updates

Consistency model - API

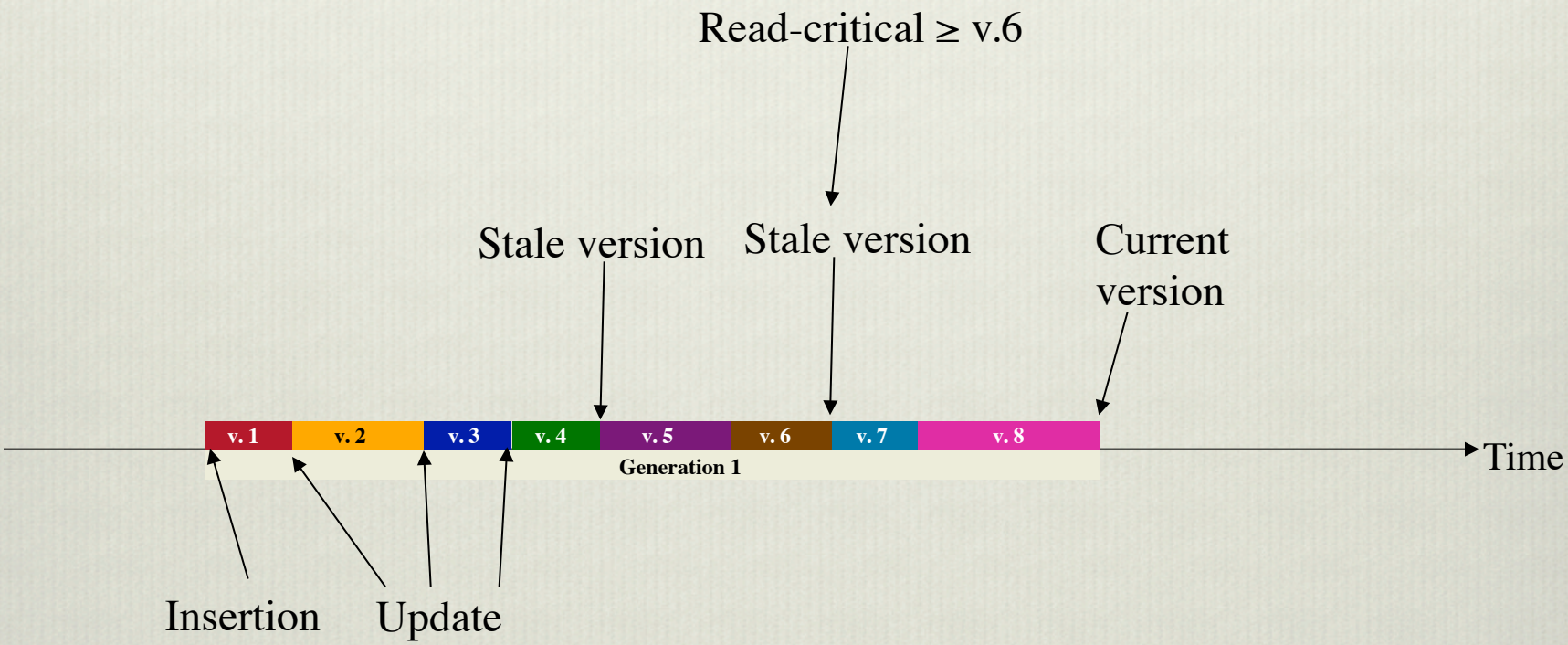
- Read-any
- Read-critical(required_version)
- Read-latest
- Write
- Test-and-set-write(required_version)

Consistency model

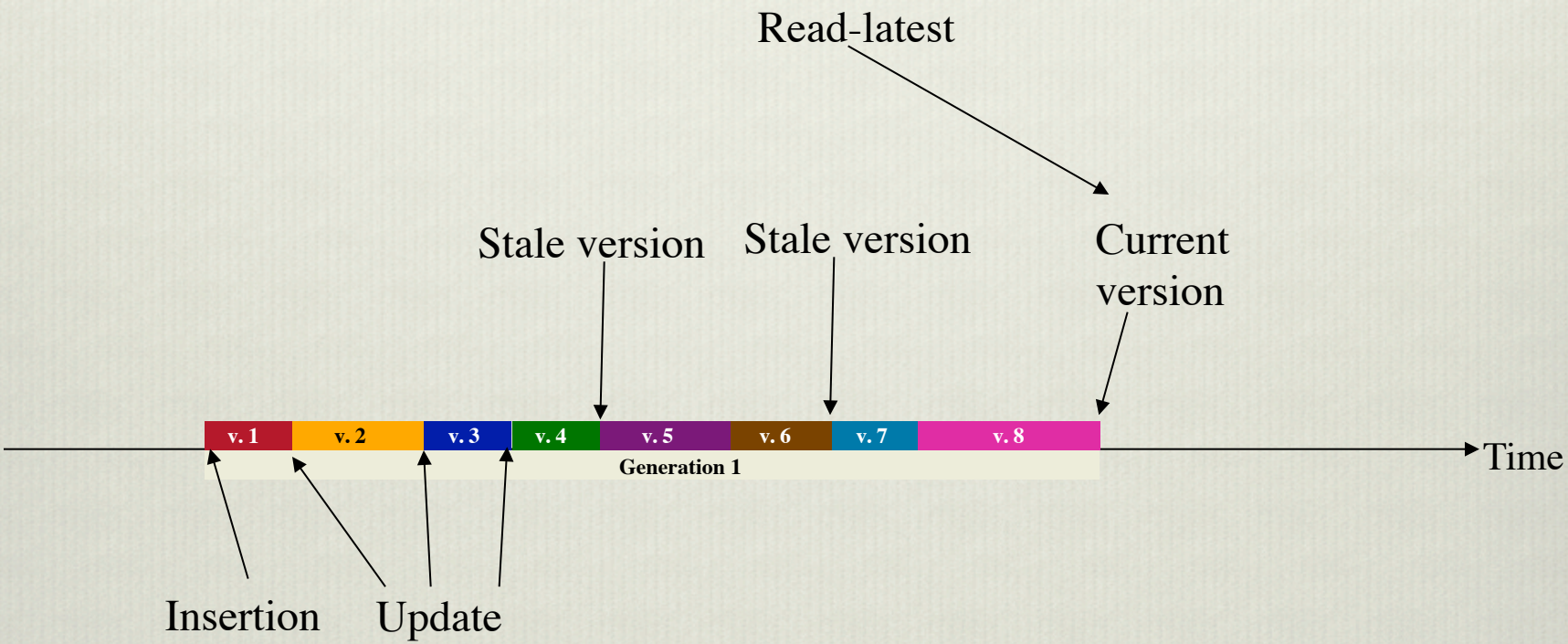


In general, reads are served using a local copy

Consistency model

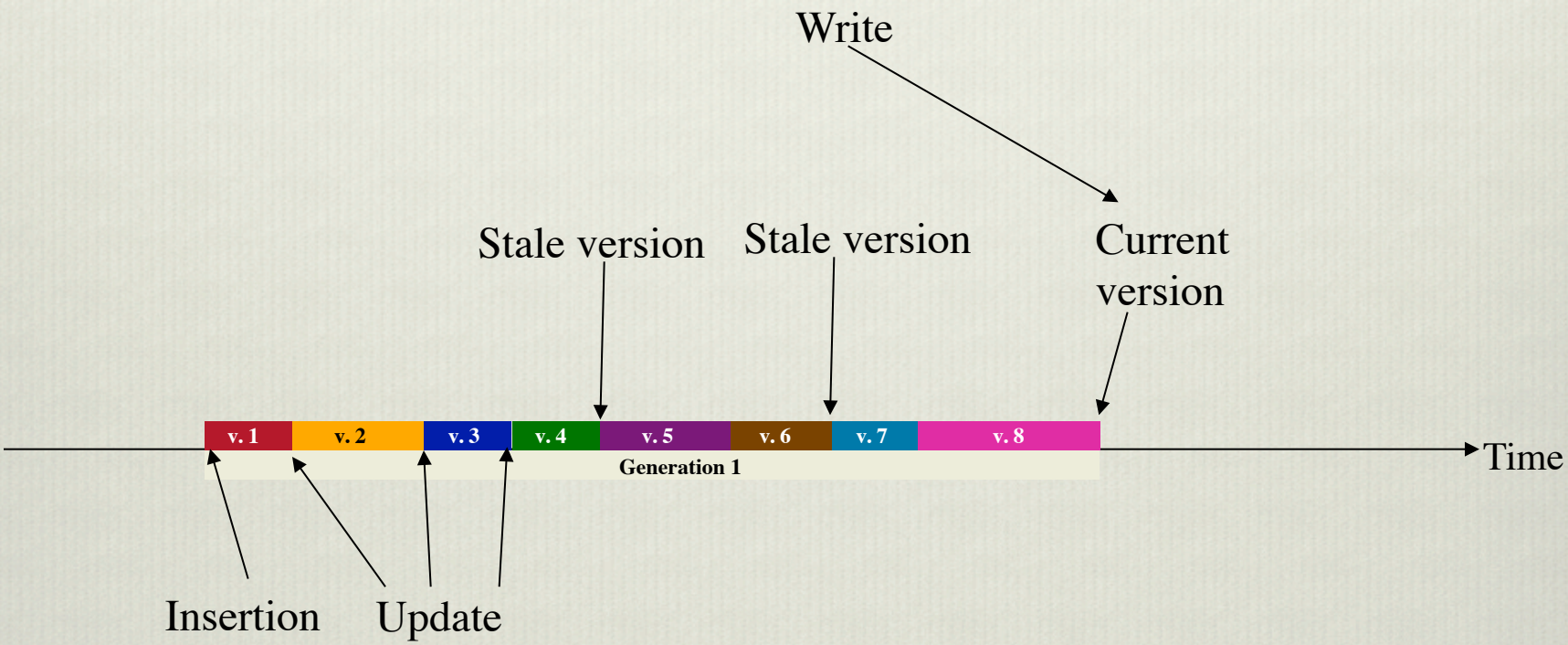


Consistency model

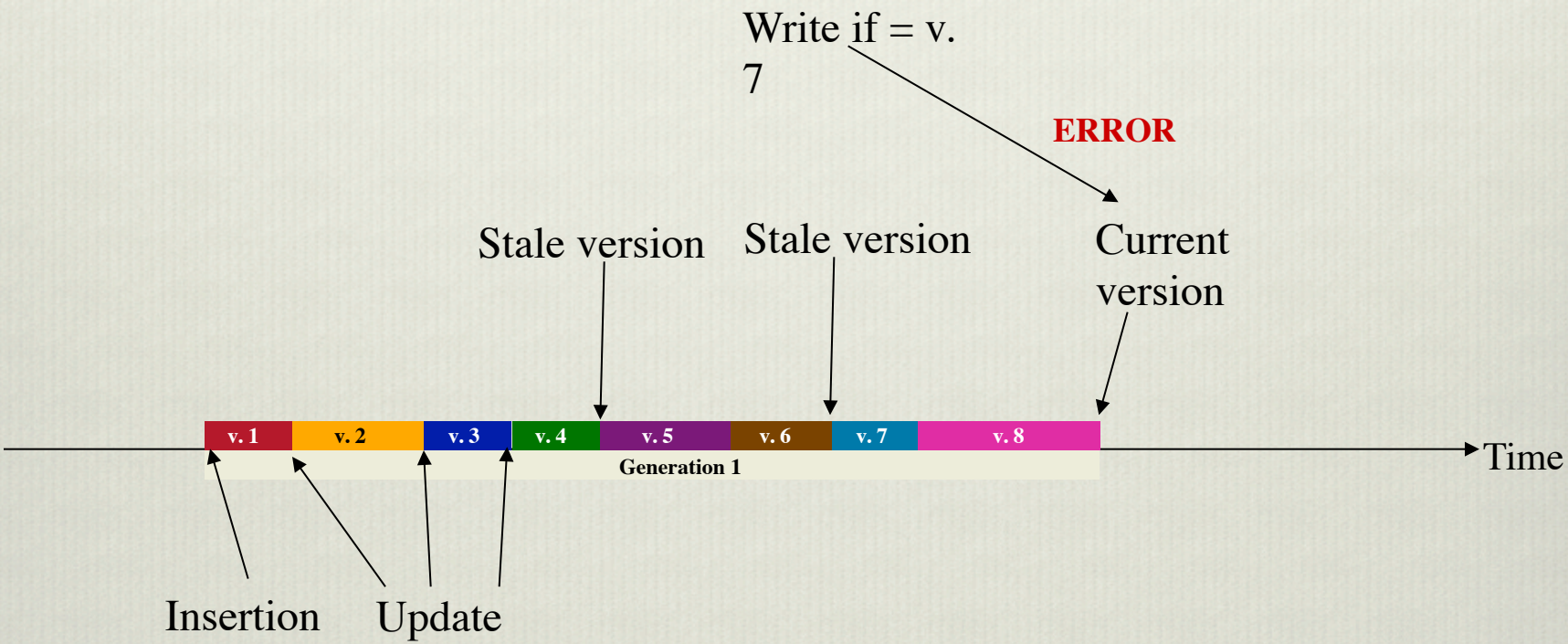


But application can request and get current version

Consistency model



Consistency model

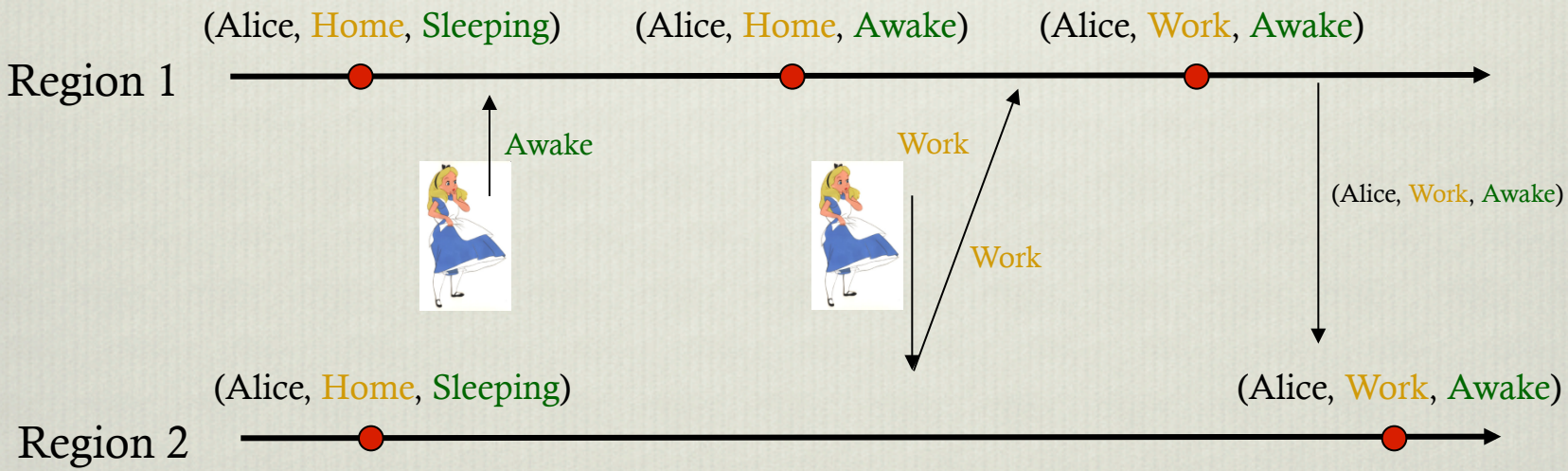


Test-and-set writes facilitate per-record transactions

Record Timeline Consistency

Transactions:

- Alice changes status from "Sleeping" to "Awake"
- Alice changes location from "Home" to "Work"



No replica should see record as (Alice, Work, Sleeping)

Eventual Consistency

- *Timeline consistency* comes at a price
  - Writes not originating in record master region forward to master and have longer latency
  - When master region down, record is unavailable for write
- We added *eventual consistency* mode
  - On conflict, latest write per field wins
  - Target customers
    - Those that externally guarantee no conflicts
    - Those that understand/can cope

Experimental setup

- Performance metric
  - Average request latency
- Three PNUTS regions

Region	Machine	Servers/region
West 1, West 2	2.8 GHz Xeon, 4GB RAM	5 SU 2 YMB 1 Router 1 Tablet controller
East	Quad 2.13 GHz Xeon	

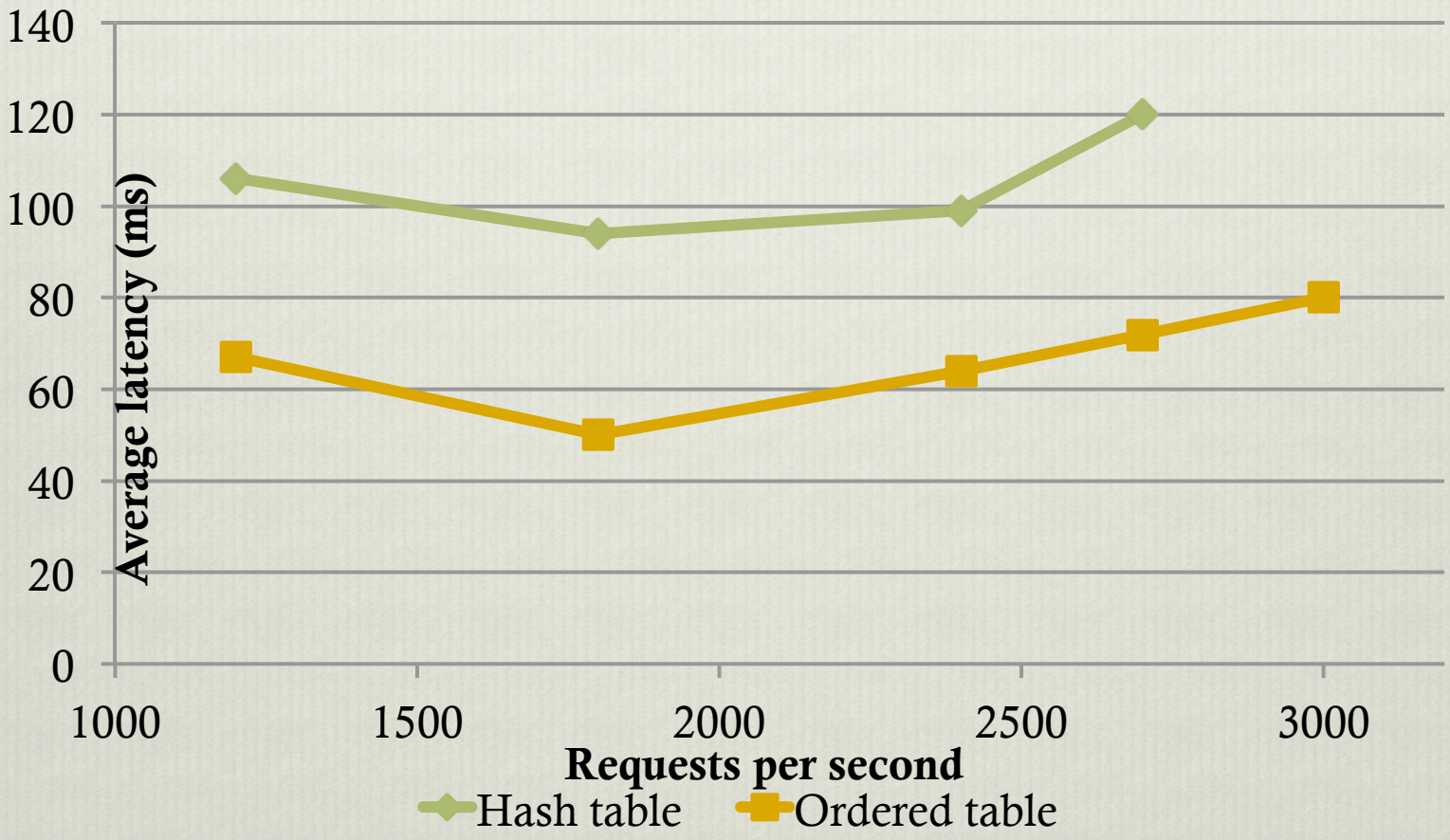
- Workload
  - 1200-3600 requests/second
  - 0-50% writes
  - 80% locality

Inserts

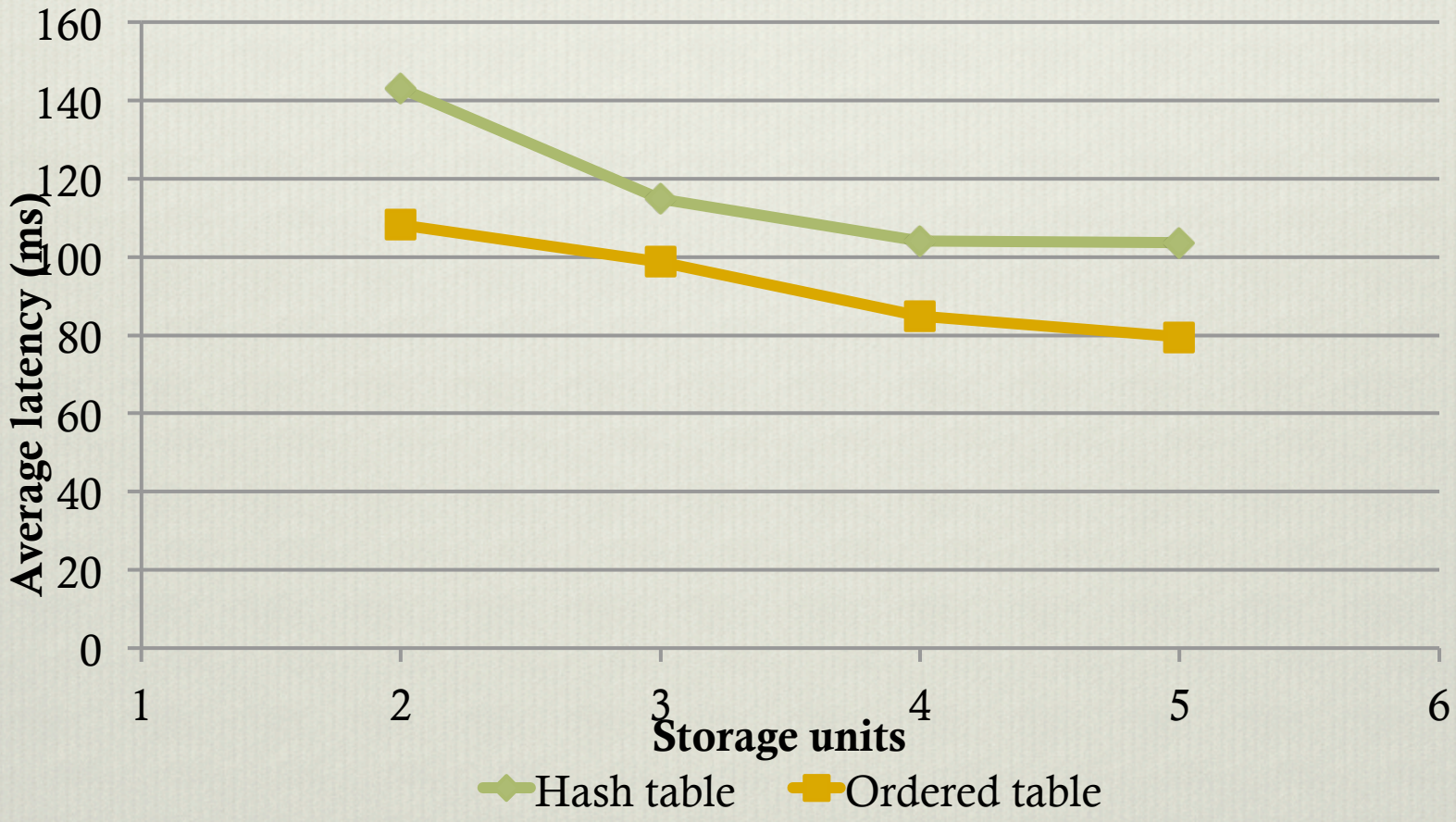
- Inserts

Region	Time (hash table)	Time (ordered table)
West 1 (master)	75.6 75.6 ms	33 ms
West 2 (non-master)	131.5 ms	105.8 ms
East (non-master)	315.5 ms	324.5 ms

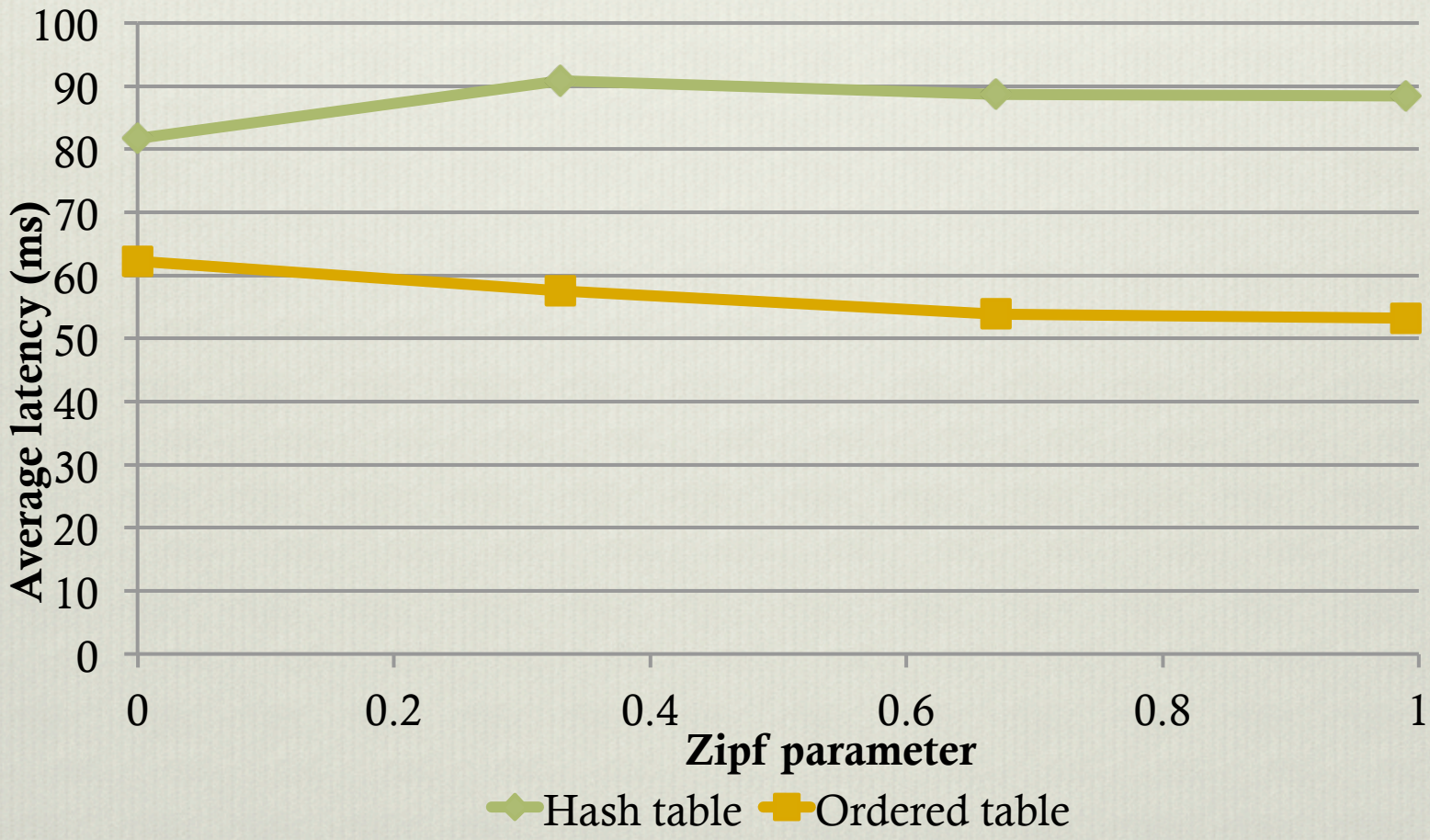
Write


140
120
100
80
60
40
20
0
Average latency (ms)
1000
1500
2000
2500
3000
Requests per second
Hash table
Ordered table

Scalability



Request skew



Future work

- Indexes
  - Efficient query processing
- Bundled updates
- Batch-query processing

Critique

- Yahoo! Message Broker
  - Multiple YMBs in one region, how to coordinate ?
  - The mechanism is rather complicated, scalability ?
  - All writes go through it, bottleneck ?

Critique

- Limited scope of the experiment
  - 5 storage units for each region
  - Testing scalability: ranging SU from 2 to 5
  - Only test latency, how about throughput ?
  - Comparison with other data storage systems, like Cassandra
    - Seems High latency

Thanks.

Questions?