# PNUTS

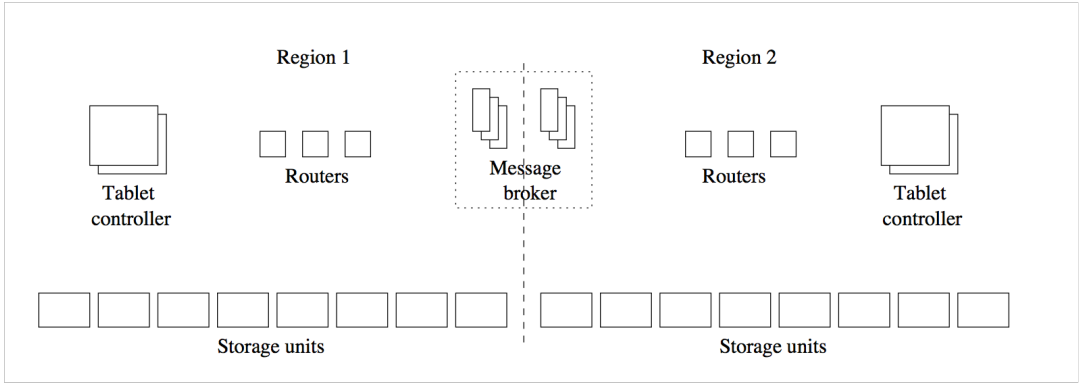
# Motivation

- Scalability
- Response time and geographic scope
- High availability and fault tolerance
- **Relaxed consistency guarantees**

# Data model

record [  key  attribute1, attribute2, ...]

- supports reads, writes, scans on primary key
- can operate on select attributes
- tables partitioned horizontally into tablets


Region 1
Region 2
Tablet controller
Routers
Message broker
Routers
Tablet controller
Storage units
Storage units

# Components

- Tablet controller -- real mapping for partition information
- Router -- caches partition information, stateless
- Storage unit -- stores many tablets
- YMB (message broker)

# Consistency

## per-record timeline consistency

all replicas of a given record apply all updates to the record in the same order

# API

- read-any
- read-critical(required_version)
- read-latest
- write
- test-and-set-write(required_version)

# Yahoo Message Broker (YMB)

- publish & subscribe system
- redo log + replication mechanism
- data updates are considered “committed” when they have been published to YMB
- record-level mastering
- 85 percent of the writes to a given record originated in the same datacenter

# YMB

- a master publishes its updates to a single broker, and thus updates are delivered to replicas in commit order.
- each record has the current master
- master replica can be updated as user moves
- tablet master resolves inserts of the same key

# Recovery

- Subscribe to YMB messages for that tablet
- Request a copy from remote replica (source tablet)
- Sends “*checkpoint”* to YMB
- Updates since “checkpoint” applied to source tablet
- Source tablet is copied over
- Resumes applying YMB messages

# Extra stuff

**scatter-gather engine** -- multi-record requests, range queries

**notification service** -- uses YMB

# Sample question (6.824 2011s)

You're running a PNUTS system (see the paper by Cooper et al.). Records X and Y both start with value zero. Here are two functions that use the API described in Section 2.2 of the PNUTS paper:

# Sample question

fn1:
x1 = read-any(X)
x1 = x1 + 1
write(X, x1) // X = x1
write(Y, x1) // Y = x1

fn2:
x1 = read-any(X)
x2 = read-latest(X)
y1 = read-any(Y)
print x1, x2, y1

# Sample question

You execute two calls to `fn1`, at different sites, at the same time. After both calls to `fn1` have returned, you execute `fn2` at a third site. There is no activity in the system other than described here, and no crashes or network failures.

**Q: What output is it possible to see from `fn2`?**

# Sample question

**fn1:**
x1 = read-any(X)
x1 = x1 + 1
write(X, x1) // X = x1
write(Y, x1) // Y = x1

**fn2:**
x1 = read-any(X)
x2 = read-latest(X)
y1 = read-any(Y)
print x1, x2, y1

A. 2, 2, 1
B. 1, 2, 2
C. 1, 1, 2
D. 2, 1, 1
E. 0, 0, 0

# Sample question

**fn1:**
x1 = read-any(X)
x1 = x1 + 1
write(X, x1) // X = x1
write(Y, x1) // Y = x1

**fn2:**
x1 = read-any(X)
x2 = read-latest(X)
y1 = read-any(Y)
print x1, x2, y1

**A. 2, 2, 1**
**B. 1, 2, 2**
C. 1, 1, 2
D. 2, 1, 1
E. 0, 0, 0