Megastore: Providing Scalable, Highly Available Storage for Interactive Services

J. Baker, C. Bond, J.C. Corbett, JJ Furman, A. Khorlin,
J. Larson, J-M Léon, Y. Li, A. Lloyd, V. Yushprakh
Google Inc.

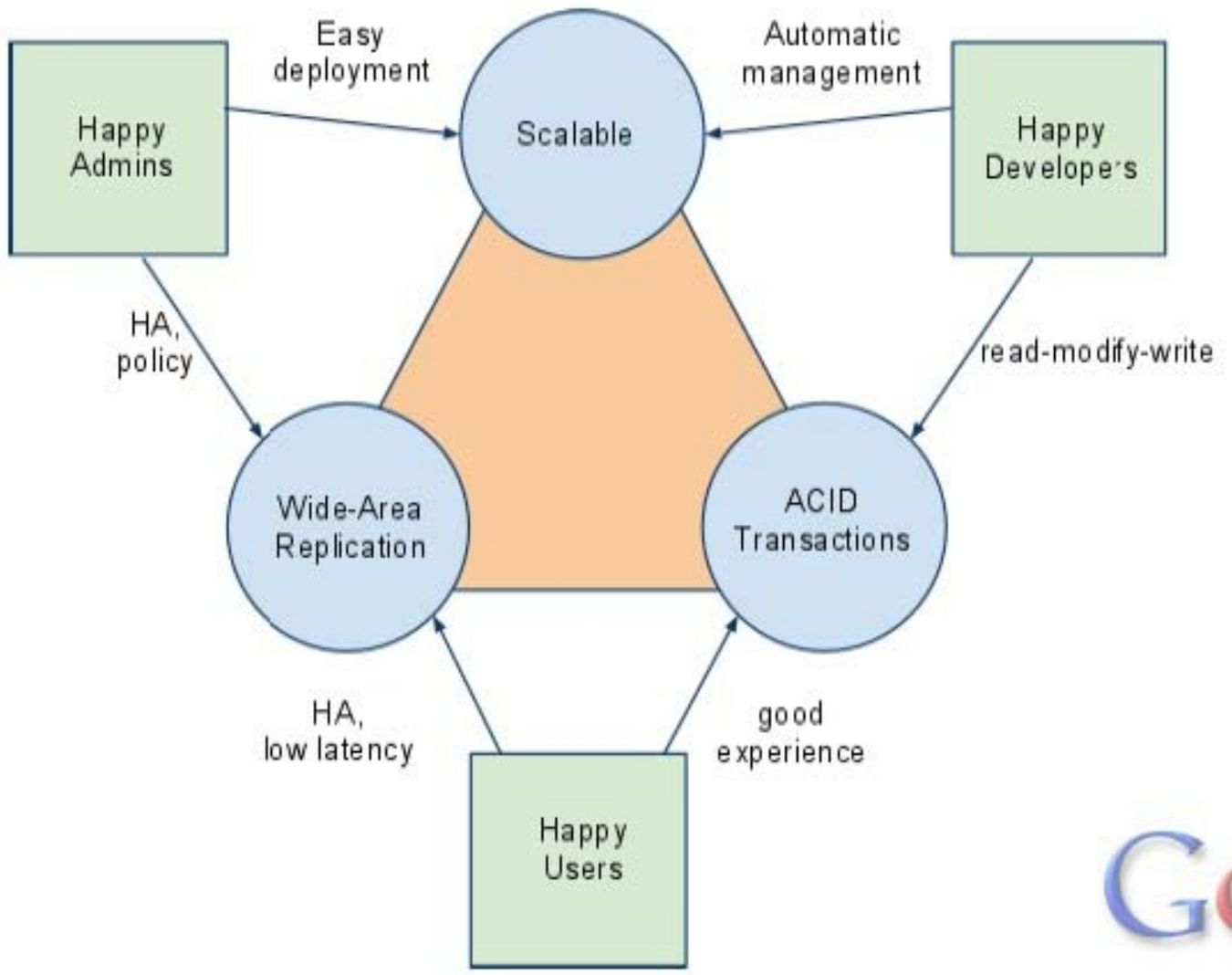
Originally presented at CIDR 2011 by James Larson

Presented by George Lee

With Great Scale Comes Great Responsibility

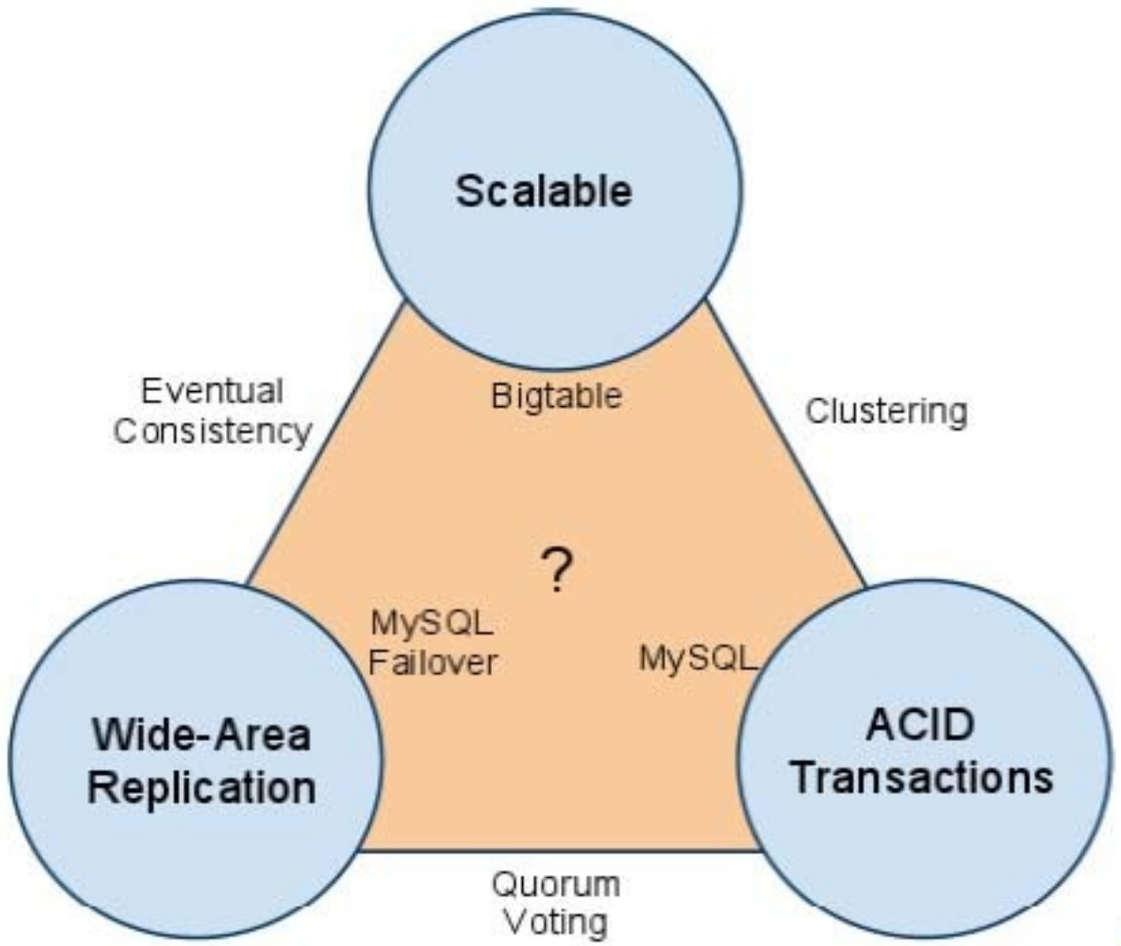
- A billion Internet users
  - Small fraction is still huge
- Must please users
  - Bad press is expensive - never lose data
  - Support is expensive - minimize confusion
  - No unplanned downtime
  - No planned downtime
  - Low latency
- Must also please developers, admins

Making Everyone Happy



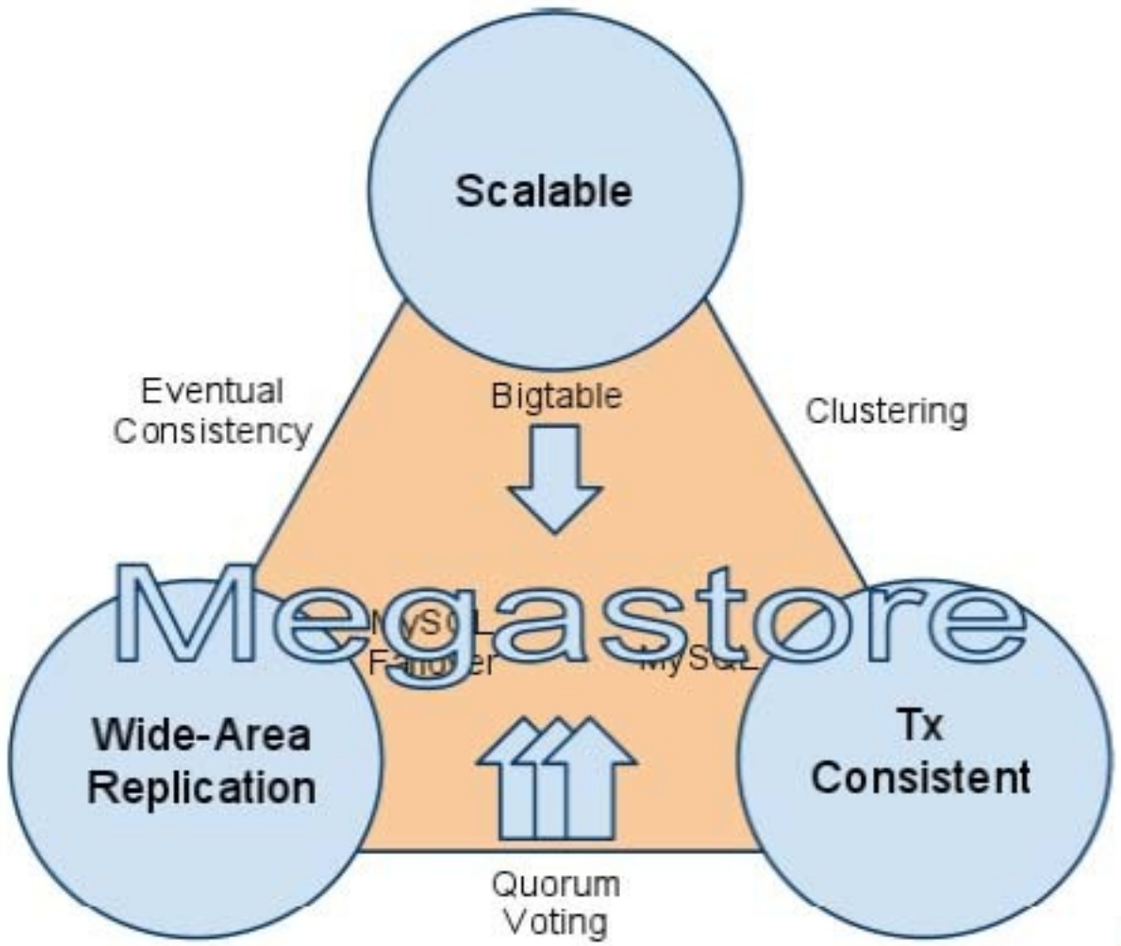
Google

Technology Options



Google™

Technology Options



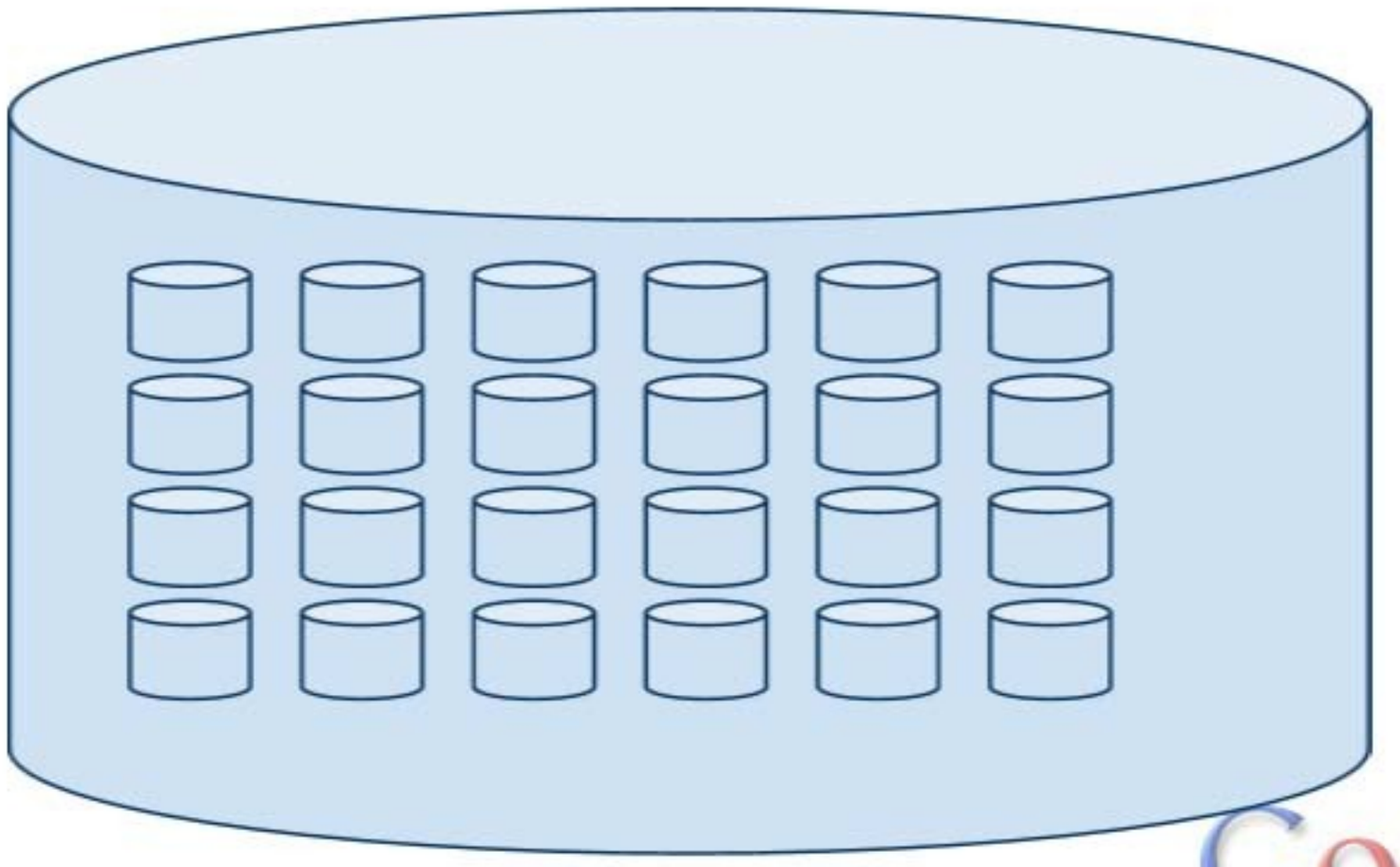
Google™

Megastore

- Started in 2006 for app development at Google
- Service layered on:
  - Bigtable (NoSQL scalable data store per datacenter)
  - Chubby (Config data, config locks)
- Turnkey scaling (apps, users)
- Developer-friendly features
- Wide-area synchronous replication
  - partition by "Entity Group"

Entity Groups

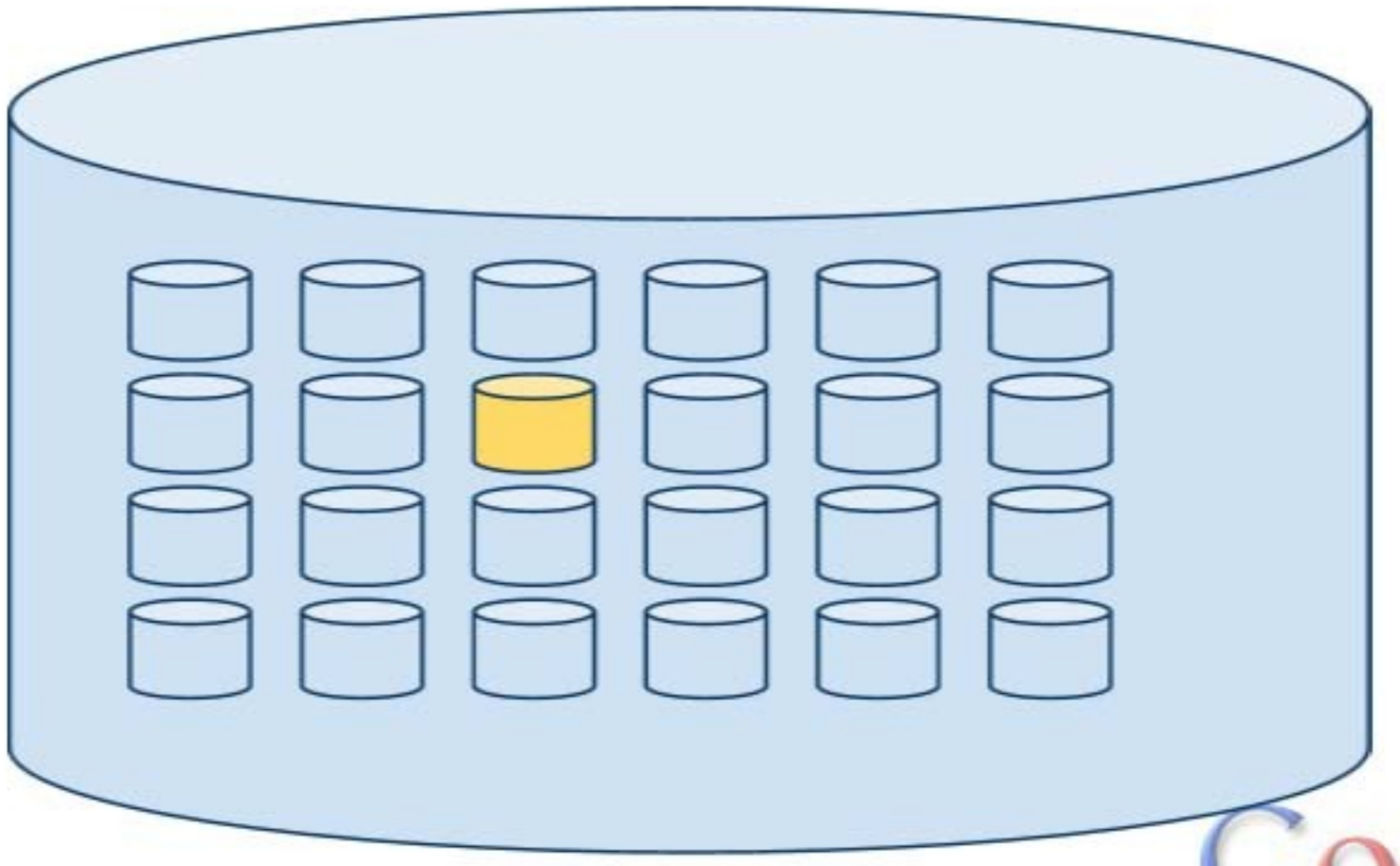
Entity Groups are sub-databases



Google™

Entity Groups

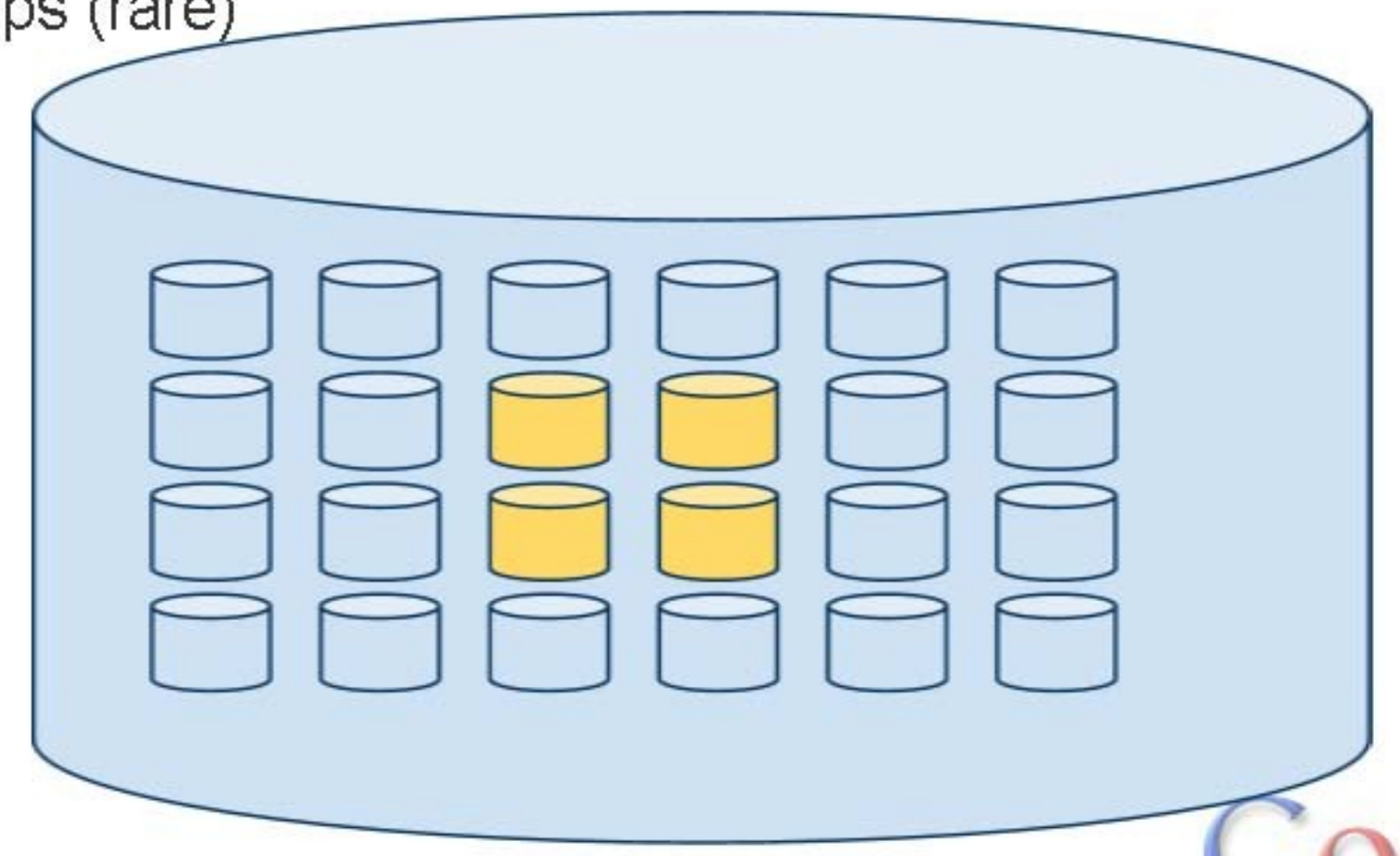
Cheap transactions within an entity group (common)



Google™

Entity Groups

Expensive or loosely-consistent operations across Entity Groups (rare)



Google™

Entity Group Examples

Application	Entity Groups	Cross-EG Ops
Email	User accounts	none
Blogs	Users, Blogs	Access control, notifications, global indexes
Mapping	Local patches	Patch-spanning ops
Social	Users, Groups	Messages, relationships, notifications
Resources	Sites	Shipments

Achieving Technical Goals

- Scale
  - Bigtable within datacenters
  - Easy to add Entity Groups (storage, throughput)
- ACID Transactions
  - Write-ahead log per Entity Group
  - 2PC or Queues between Entity Groups
- Wide-Area Replication
  - Paxos
  - Tweaks for optimal latency

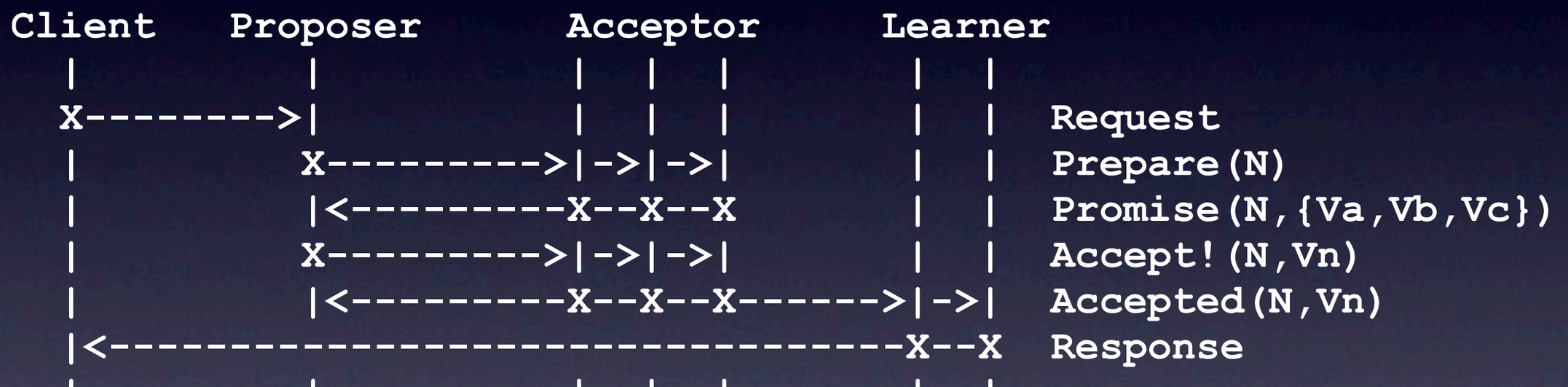
Two Phase Commit

- Commit request/Voting phase
  - Coordinator sends query to commit
  - Cohorts prepare and reply
- Commit/Completion phase
  - Success: Commit and acknowledge
  - Failure: Rollback and acknowledge
- Disadvantage: Blocking protocol

Basic Paxos

- Prepare and Promise
  - Proposer selects proposal number N and sends promise to acceptors
    - Acceptors accept or deny the promise
- Accept! and Accepted
  - Proposer sends out value
  - Acceptors respond to proposer and learners

Message Flow: Basic Paxos



Paxos: Quorum-based Consensus

"While some consensus algorithms, such as Paxos, have started to find their way into [large-scale distributed storage systems built over failure-prone commodity components], their uses are limited mostly to the maintenance of the global configuration information in the system, not for the actual data replication."

-- Lamport, Malkhi, and Zhou, May 2009

Paxos and Megastore

- In practice, basic Paxos is not used
- Master-based approach?
- Megastore's tweaks
  - Coordinators
  - Local reads
  - Read/write from any replica
  - Replicate log entries on each write

Omissions

- These were noted in the talk:
- No current query language
  - Apps must implement query plans
  - Apps have fine-grained control of physical placement
- Limited per-Entity Group update rate

Is Everybody Happy?

- Admins
  - linear scaling, transparent rebalancing (Bigtable)
  - instant transparent failover
  - symmetric deployment
- Developers
  - ACID transactions (read-modify-write)
  - many features (indexes, backup, encryption, scaling)
  - single-system image makes code simple
  - little need to handle failures
- End Users
  - fast up-to-date reads, acceptable write latency
  - consistency

Take-Aways

- Constraints acceptable to most apps
  - Entity Group partitioning
  - High write latency
  - Limited per-EG throughput
- In production use for over 4 years
- Available on Google App Engine as HRD (High Replication Datastore)

Questions?

- Rate limitation on writes are insignificant?
- Why not lots of RDBMS?
- Why not NoSQL with “mini-databases”?