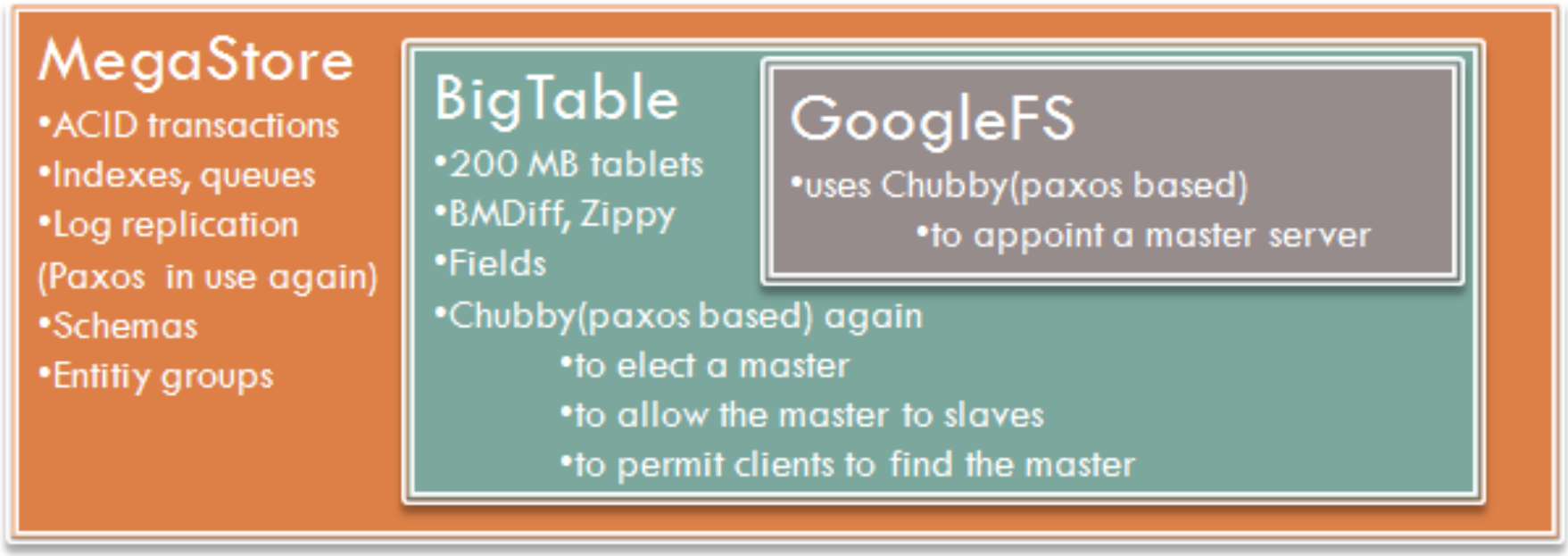
Google Megastore: The Data Engine Behind GAE

presentation by Atreyee Maiti

Carnegie Mellon University

What is it?

- Best of both worlds - NoSQL and relational
- Fully serializable ACID in fine grained data partitions
- Designed for interactive online services which pose challenging requirements



Handles more than three billion write and 20 billion read transactions daily and stores nearly a petabyte of primary data across many global datacenters. Being used for google app engine since 2009 and hundreds of google applications

Brief overview of main concepts

- Replication
- Partitioning
- Entity Groups
- Data model
- Transactions

Replication

Needed across wide geographic area

Possible Strategies:

- async master/slave
- sync master
- optimistic replication

Paxos to the rescue!

- Inherently fault tolerant
- Write ahead log replicated over peers
- Acknowledges when majority of replicas have changes others catch up when able to

Partitioning and locality

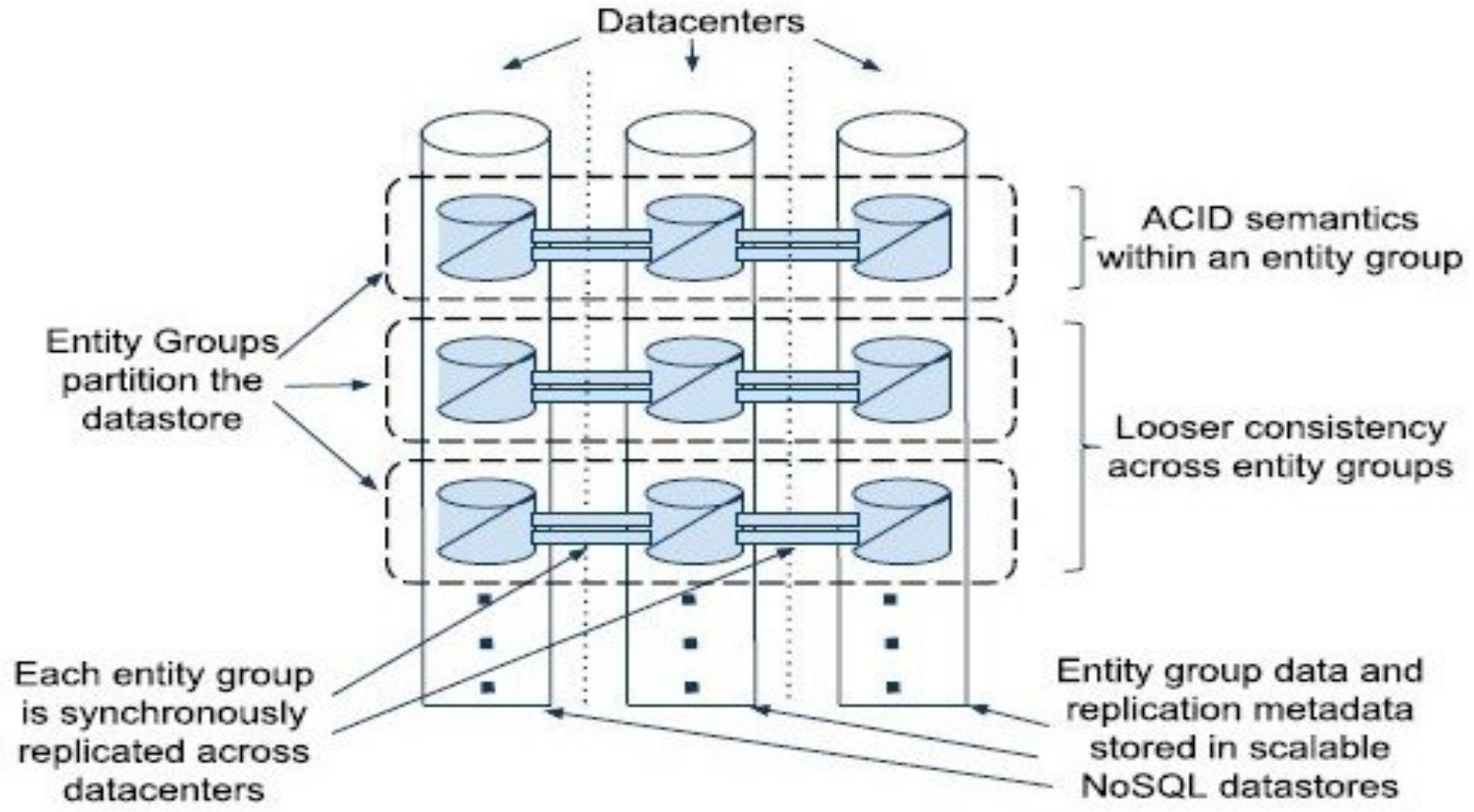


Figure 1: Scalable Replication

source: J. Baker, et al., MegaStore: Providing Scalable, Highly Available Storage For Interactive Services

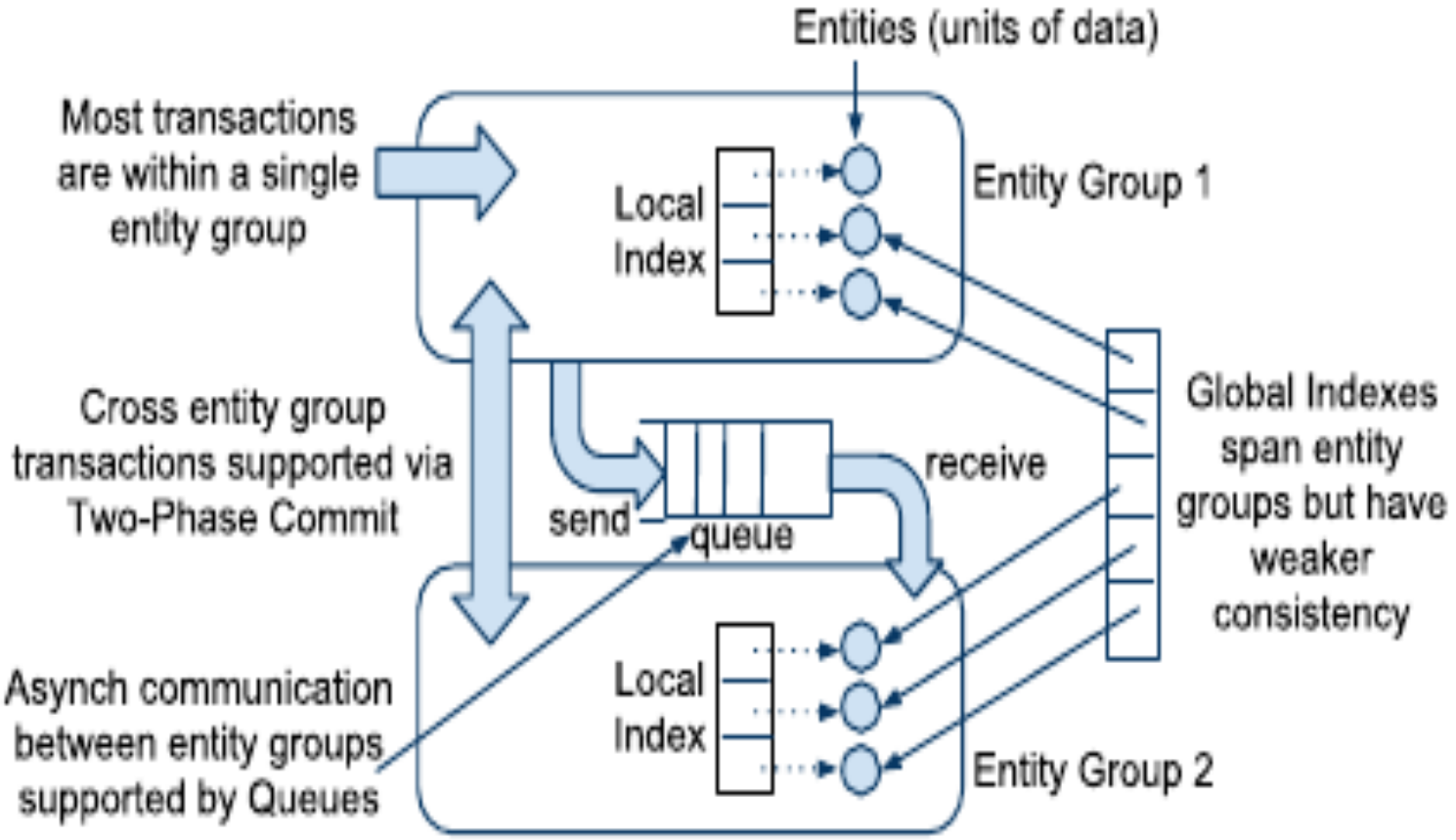


Figure 2: Operations Across Entity Groups

source: J. Baker, et al., MegaStore: Providing Scalable, Highly Available Storage For Interactive Services

Entity group boundaries

email

blogs - profiles

- Storing data - uses big table
- For low latency, cache efficiency, and throughput, the data for an entity group are held in contiguous ranges of Bigtable rows.
- Schema language lets applications control the placement of hierarchical data, storing data that is accessed together in nearby rows or denormalized into the same row.

API design philosophy

- Aim is to serve interactive apps - cannot afford expensive joins
- Move complexity to writes because reads are higher
- Joins not needed because of the hierarchical organization in big table

Data model

```
CREATE SCHEMA PhotoApp;

CREATE TABLE User {
 required int64 user_id;
 required string name;
} PRIMARY KEY(user_id), ENTITY GROUP ROOT;

CREATE TABLE Photo {
 required int64 user_id;
 required int32 photo_id;
 required int64 time;
 required string full_url;
 optional string thumbnail_url;
 repeated string tag;
} PRIMARY KEY(user_id, photo_id),
  IN TABLE User,
  ENTITY GROUP KEY(user_id) REFERENCES User;

CREATE LOCAL INDEX PhotosByTime
  ON Photo(user_id, time);

CREATE GLOBAL INDEX PhotosByTag
  ON Photo(tag) STORING (thumbnail_url);
```

Figure 3: Sample Schema for Photo Sharing Service

Carnegie Mellon University

source: J. Baker, et al., MegaStore: Providing Scalable, Highly Available Storage For Interactive Services

Indexes

- Could be on any property
- Local - to search within entity group
- Global - to find across entity groups - without knowing which group they belong to - find all photos tagged by big data
- Storing clause - add additional properties on the entity for faster retrieval
- Repeated indexes - for repeated properties
- Inline indexes - for extracting info from child entities and storing in parent for fast access - can be used to implement many to many links

Mapping to Bigtable

Megastore table name + property name = Bigtable column name

metadata maintained in same row of Bigtable - atomicity

Row key	User. name	Photo. time	Photo. tag	Photo. _url
101	John			
101,500		12:30:01	Dinner, Paris	...
101,502		12:15:22	Betty, Paris	...
102	Mary			

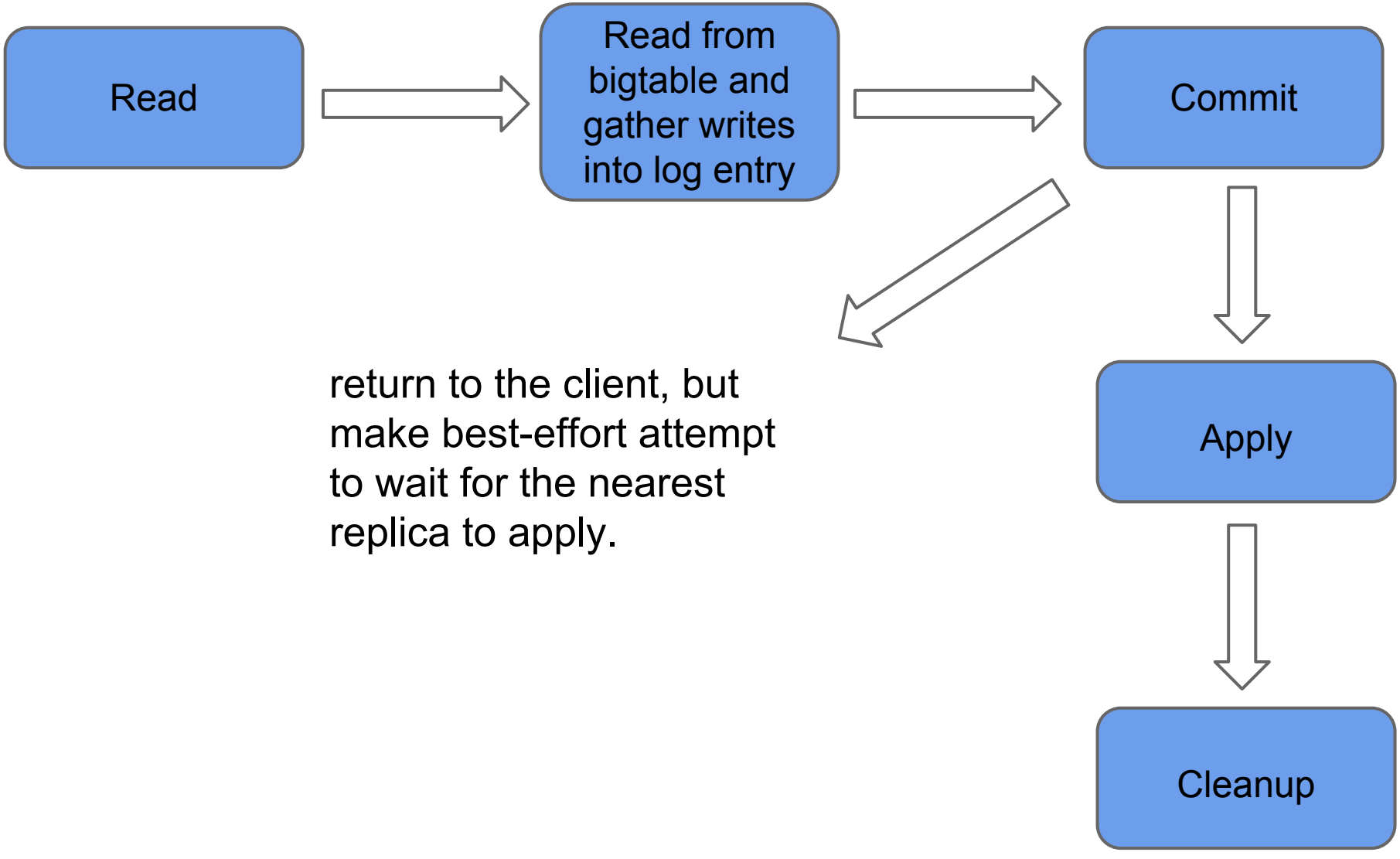
Figure 4: Sample Data Layout in Bigtable

source: J. Baker, et al., MegaStore: Providing Scalable, Highly Available Storage For Interactive Services

Transactions and concurrency control

- Each entity group like a mini db with serializable ACID semantics. A transaction writes its mutations into the entity group's write-ahead log, then the mutations are applied to the data
- Implements multiversion concurrency control (MVCC)
- Provides current, snapshot and inconsistent reads

Transaction lifecycle



return to the client, but make best-effort attempt to wait for the nearest replica to apply.

Queues

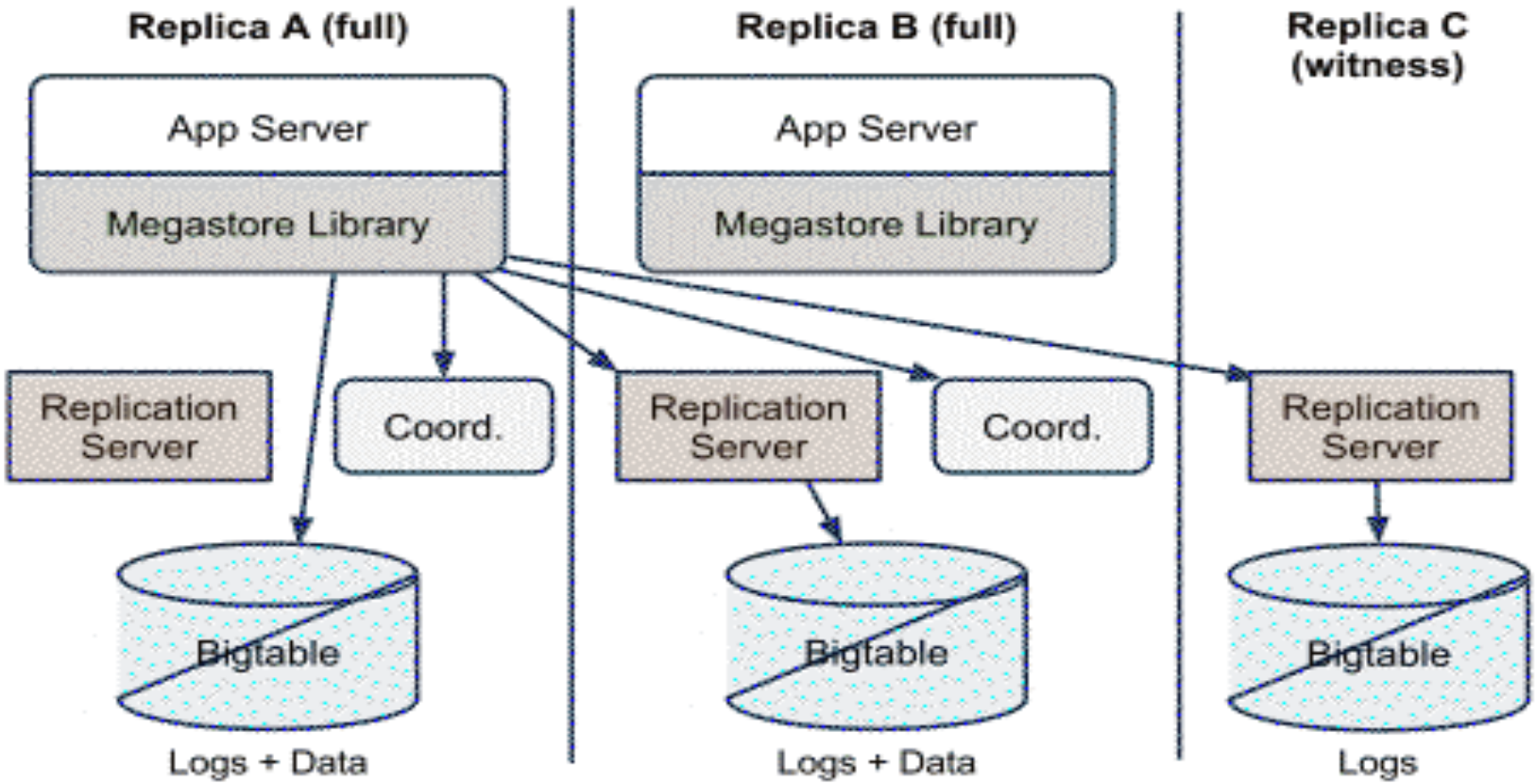
A way to batch multiple updates into a single transaction, or to defer work

For example, calendar application

Replication in detail

- Reads and writes can be initiated from any replica, and ACID semantics are preserved.
- Replication is done per entity group by synchronously replicating the group's transaction log to a quorum of replicas

Megastore's usage of paxos

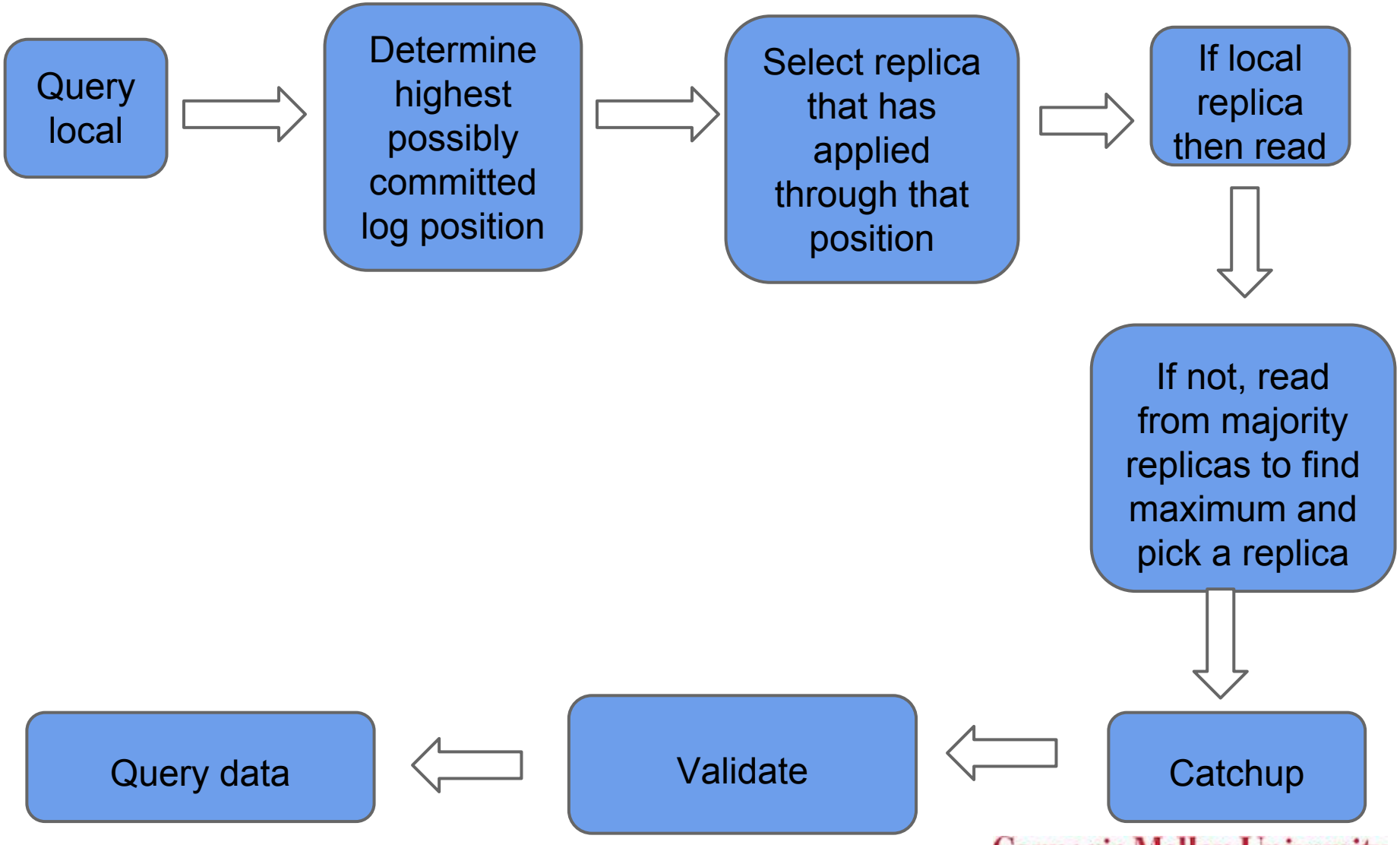


Megastore Architecture Example

Carnegie Mellon University

source: J. Baker, et al., MegaStore: Providing Scalable, Highly Available Storage For Interactive Services

Algorithms


Query local
Determine highest possibly committed log position
Select replica that has applied through that position
If local replica then read
If not, read from majority replicas to find maximum and pick a replica
Catchup
Validate
Query data

Comparison

Name of System	Difference
Bigtable, Cassandra, and PNUTS	traditional RDBMS systems properties not sacrificed synchronous replication schemes with consistency These systems often reduce the scope of transactions to the granularity of single key access and place hurdle to building applications - lack rich data model
Bigtable replication	replicates at the level of entire entity group transactions, not individual Bigtable column values.

Limitations

- Fault tolerance is fault masking
- Chain gang throttling
- Achieving good performance for more complex queries requires attention to the physical data layout in Bigtable
- Megastore does not enforce specific policies on block sizes, compression, table splitting, locality group, nor other tuning controls provided by Bigtable.

Conclusion

- As Brewer's CAP theorem showed, a distributed system can't provide consistency, availability and partition tolerance to all nodes at the same time. But this paper shows that by making smart choices we can get darn close as far as human users are concerned.
- Megastore is perhaps the 1st large-scale storage system to implement Paxos-based replication across datacenters while satisfying the scalability and performance requirements of scalable web applications in the cloud.

References / Acknowledgements

http://googleappengine.blogspot.com/2009/09/migration-to-better-datastore.html

http://googleappengine.blogspot.com/2010/06/datastore-performance-growing-pains.html

http://storagemojo.com/2011/04/20/googles-megastore/

http://www.informationweek.com/internet/google/google-spills-megastores-secrets/229205494

Resources

http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/en/us/events/io/2011/static/presofiles/more_9s_under_the_covers_of_the_high_replication_datastore.pdf

http://www.youtube.com/watch?v=tx5gdoNpcZM