Google Megastore

Jasper Bernhardt
Elias Grünewald
Fabian Lehmann

Technische Universität Berlin
Mobile Cloud Computing
EINSTEIN CENTER Digital Future

Megastore: Providing Scalable, Highly Available Storage for Interactive Services

Jason Baker, Chris Bond, James C. Corbett, JJ Furman, Andrey Khorlin, James Larson, Jean-Michel Léon, Yawei Li, Alexander Lloyd, Vadim Yushprakh
Google, Inc.

{jasonbaker,chrisbond,jcorbett,jfurman,akhorlin,jimlarson,jm,yaweili,alloyd,vadimy}@google.com

ABSTRACT

Megastore is a storage system developed to meet the requirements of today's interactive online services. Megastore blends the scalability of a NoSQL datastore with the convenience of a traditional RDBMS in a novel way, and provides both strong consistency guarantees and high availability. We provide fully serializable ACID semantics within fine-grained partitions of data. This partitioning allows us to synchronously replicate each write across a wide area network with reasonable latency and support seamless failover between datacenters. This paper describes Megastore's semantics and replication algorithm. It also describes our experience supporting a wide range of Google production services built with Megastore.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed databases; H.2.4 [**Database Management**]: Systems—*concurrency, distributed databases*

General Terms

Algorithms, Design, Performance, Reliability

Keywords

Large databases, Distributed transactions, Bigtable, Paxos

1. INTRODUCTION

Interactive online services are forcing the storage community to meet new demands as desktop applications migrate to the cloud. Services like email, collaborative documents, and social networking have been growing exponentially and are testing the limits of existing infrastructure. Meeting these services' storage demands is challenging due to a number of conflicting requirements.

First, the Internet brings a huge audience of potential users, so the applications must be *highly scalable*. A service can be built rapidly using MySQL [10] as its datastore, but scaling the service to millions of users requires a complete redesign of its storage infrastructure. Second, services must compete for users. This requires *rapid development* of features and fast time-to-market. Third, the service must be responsive; hence, the storage system must have *low latency*. Fourth, the service should provide the user with a *consistent view of the data*—the result of an update should be visible immediately and durably. Seeing edits to a cloud-hosted spreadsheet vanish, however briefly, is a poor user experience. Finally, users have come to expect Internet services to be up 24/7, so the service must be *highly available*. The service must be resilient to many kinds of faults ranging from the failure of individual disks, machines, or routers all the way up to large-scale outages affecting entire datacenters.

These requirements are in conflict. Relational databases provide a rich set of features for easily building applications, but they are difficult to scale to hundreds of millions of users. NoSQL datastores such as Google's Bigtable [15], Apache Hadoop's HBase [1], or Facebook's Cassandra [6] are highly scalable, but their limited API and loose consistency models complicate application development. Replicating data across distant datacenters while providing low latency is challenging, as is guaranteeing a consistent view of replicated data, especially during faults.

Megastore is a storage system developed to meet the storage requirements of today's interactive online services. It is novel in that it blends the scalability of a NoSQL datastore with the convenience of a traditional RDBMS. It uses synchronous replication to achieve high availability and a consistent view of the data. In brief, it provides fully serializable ACID semantics over distant replicas with low enough latencies to support interactive applications.

We accomplish this by taking a middle ground in the RDBMS vs. NoSQL design space: we partition the datastore and replicate each partition separately, providing full ACID semantics within partitions, but only limited consistency guarantees across them. We provide traditional database features, such as secondary indexes, but only those features that can scale within user-tolerable latency limits, and only with the semantics that our partitioning scheme can support. We contend that the data for most Internet services can be suitably partitioned (e.g., by user) to make this approach viable, and that a small, but not spartan, set of features can substantially ease the burden of developing cloud applications.

Contrary to conventional wisdom [24, 28], we were able to use Paxos [27] to build a highly available system that pro-

5^{th} *Biennial Conference on Innovative Data Systems Research (CIDR '11)* January 9-12, 2011, Asilomar, California, USA.

Technische Universität Berlin

Baker, Jason, Chris Bond, James C. Corbett, JJ Furman, Andrey Khorlin, James Larson, Jean-Michel Leon, Yawei Li, Alexander Lloyd, and Vadim Yushprakh. 2011. "Megastore: Providing Scalable, Highly Available Storage for Interactive Services." In *Proceedings of the Conference on Innovative Data system Research (CIDR),* 223–234. `http://www.cidrdb.org/cidr2011/Papers/CIDR11_Paper32.pdf`.

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

In a nutshell

High Replication Datastore

Published in 2008

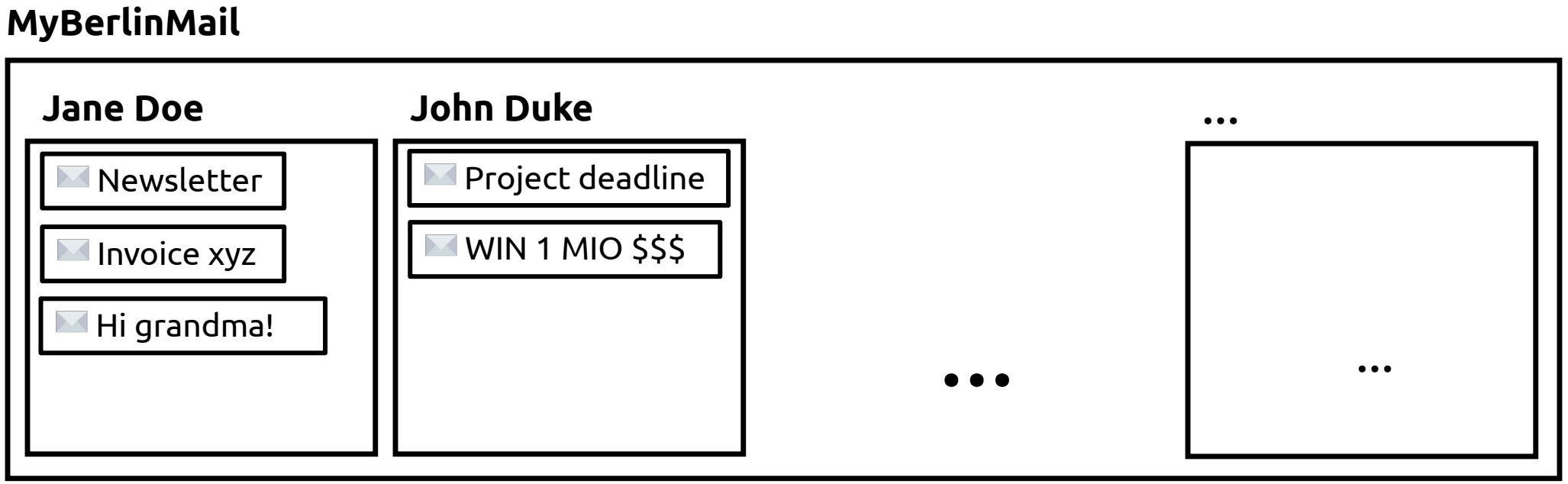
Megastore, BigTable ⇒ Spanner

Hybrid: RDBMS/Data store

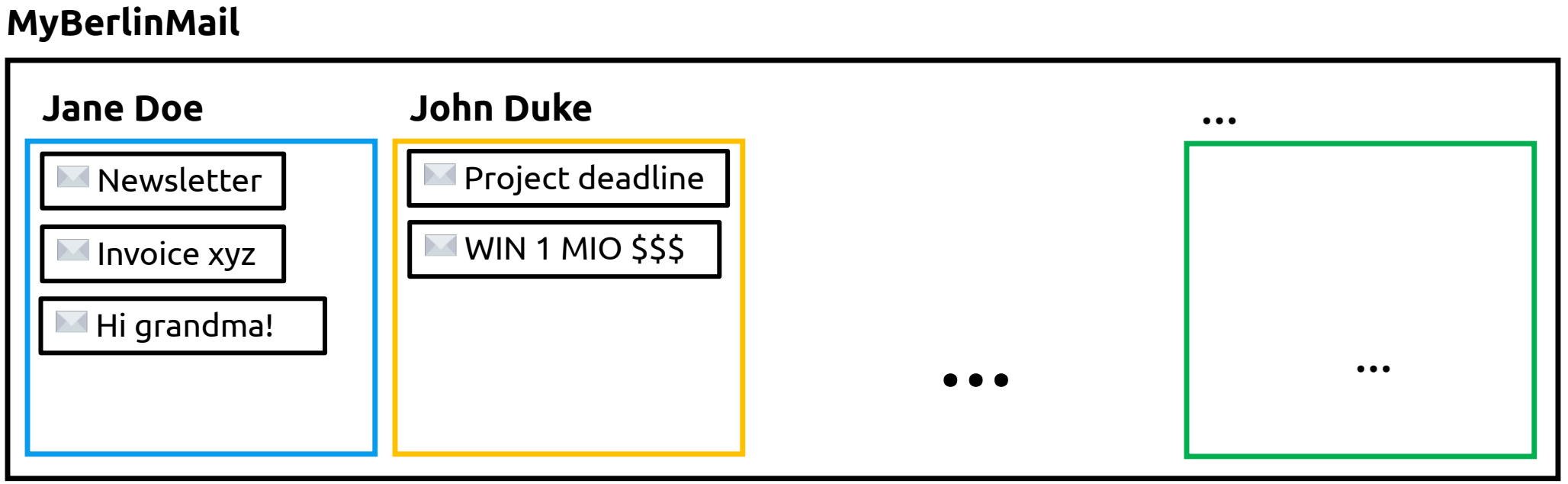
Technische Universität Berlin

Availability	Scalability
~~Asynchronous Master/Slave, Synchronous Master/Slave, Optimistic replication~~	data partitioned into several smaller databases (“entity groups”)
Paxos: synchronous, fault tolerant log replicator	underlying data is stored in NoSQL data store in each data center
any node may initiate reads and writes ⇒ majority vote	ACID transaction within one entity group; then log replication

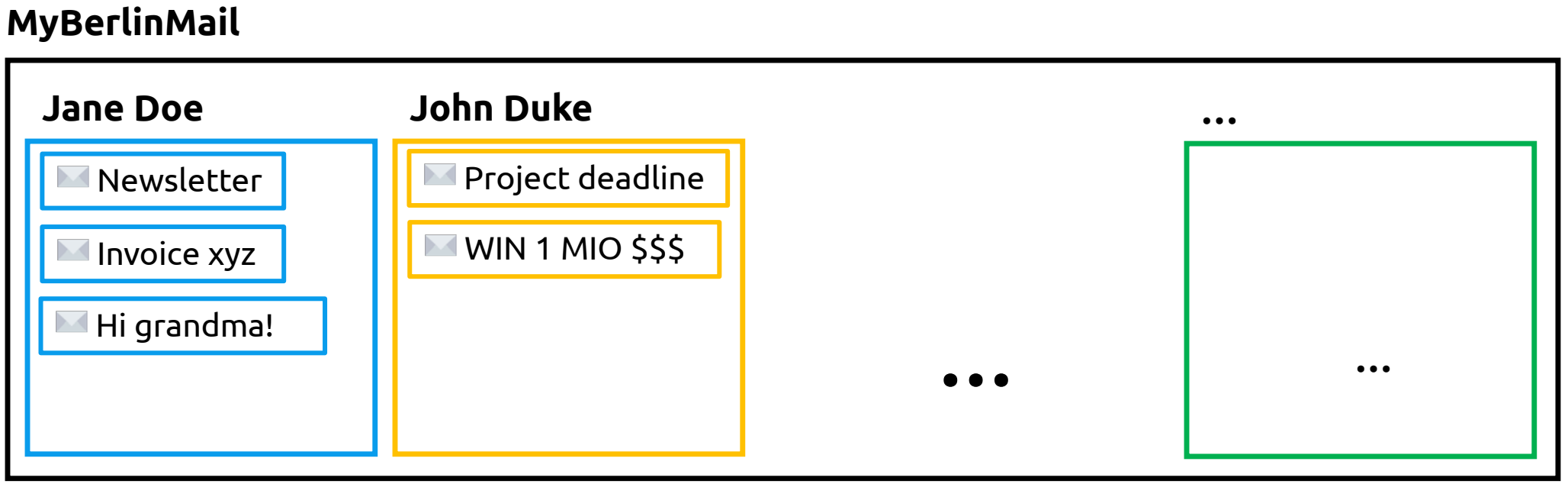
Example: Partitioning



Example: Partitioning



Example: Partitioning



Entity group 1

Entity 1.1
Entity 1.2
Entity 1.3

Entity group 2

Entity 2.1
Entity 2.2

Entity group *n*

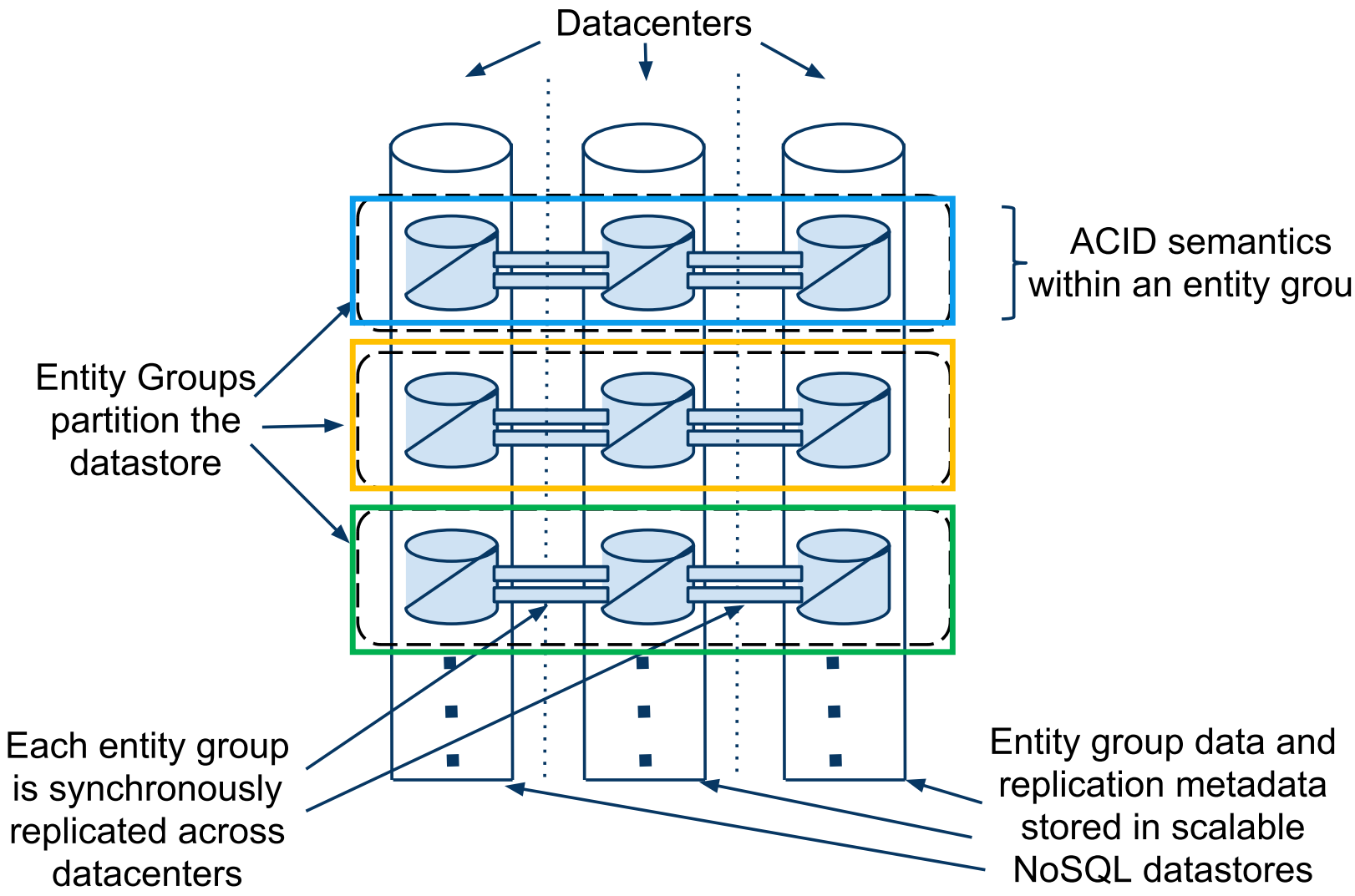
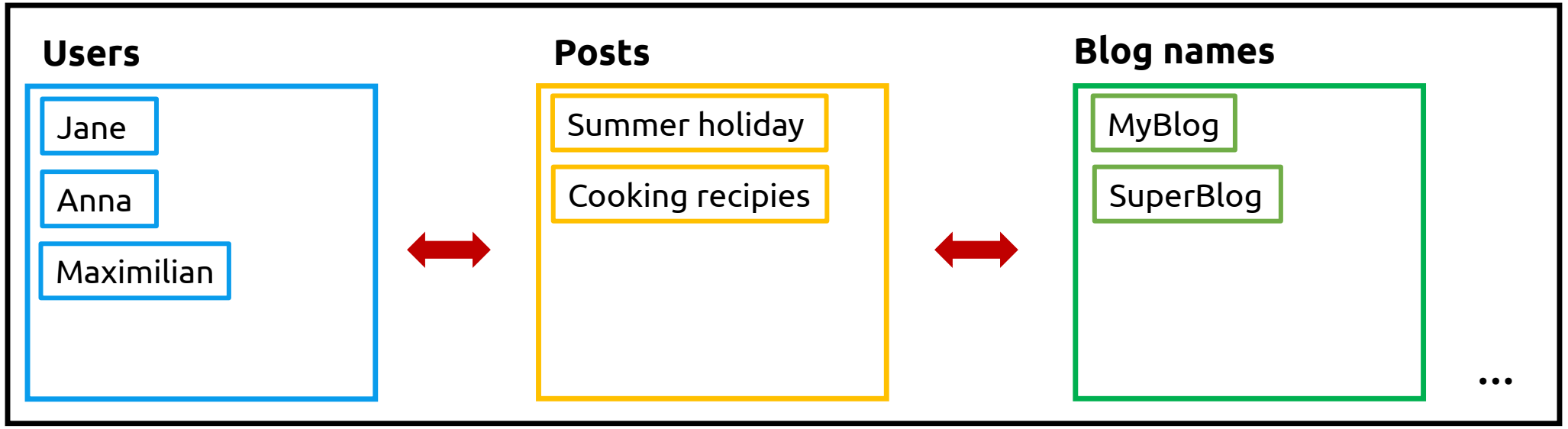


Figure 1: Scalable Replication

Example: Partitioning

Technische Universität Berlin

Blogging application



Mobile Cloud Computing

EINSTEIN CENTER Digital Future

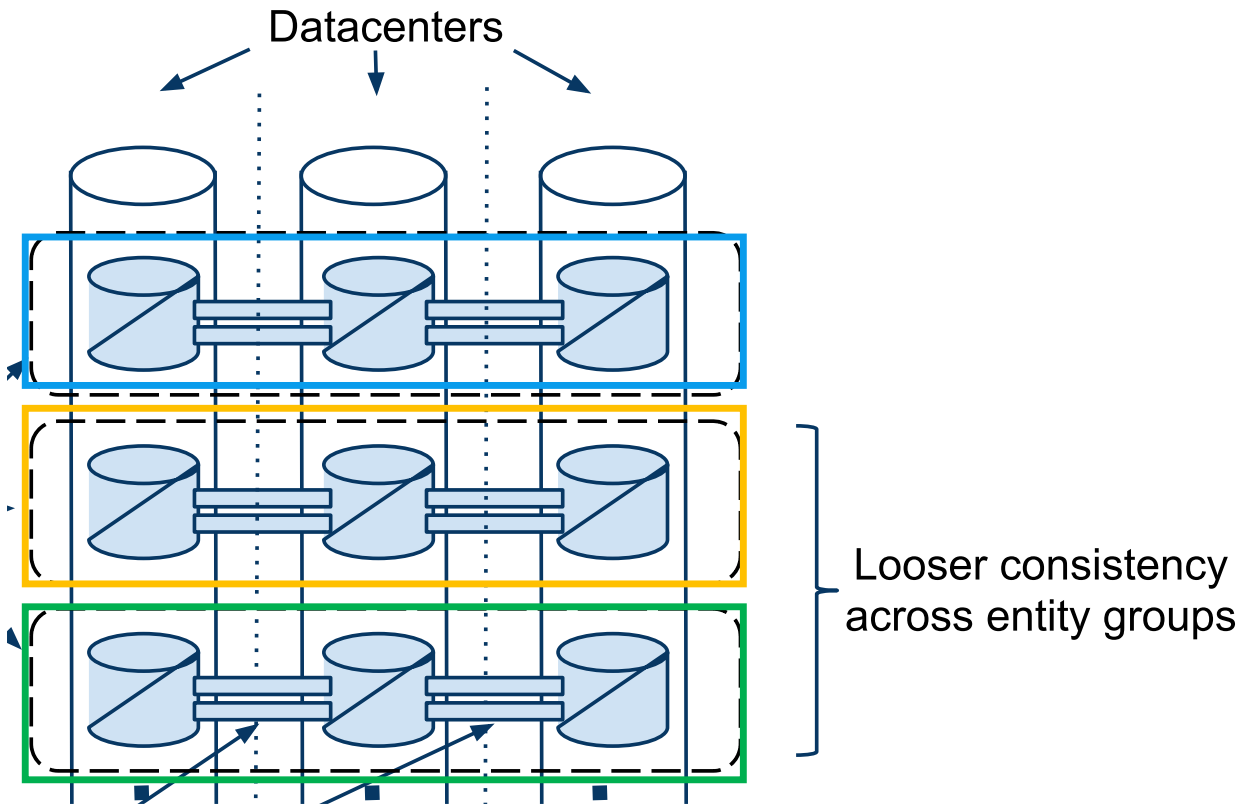


Figure 1: Scalable Replication

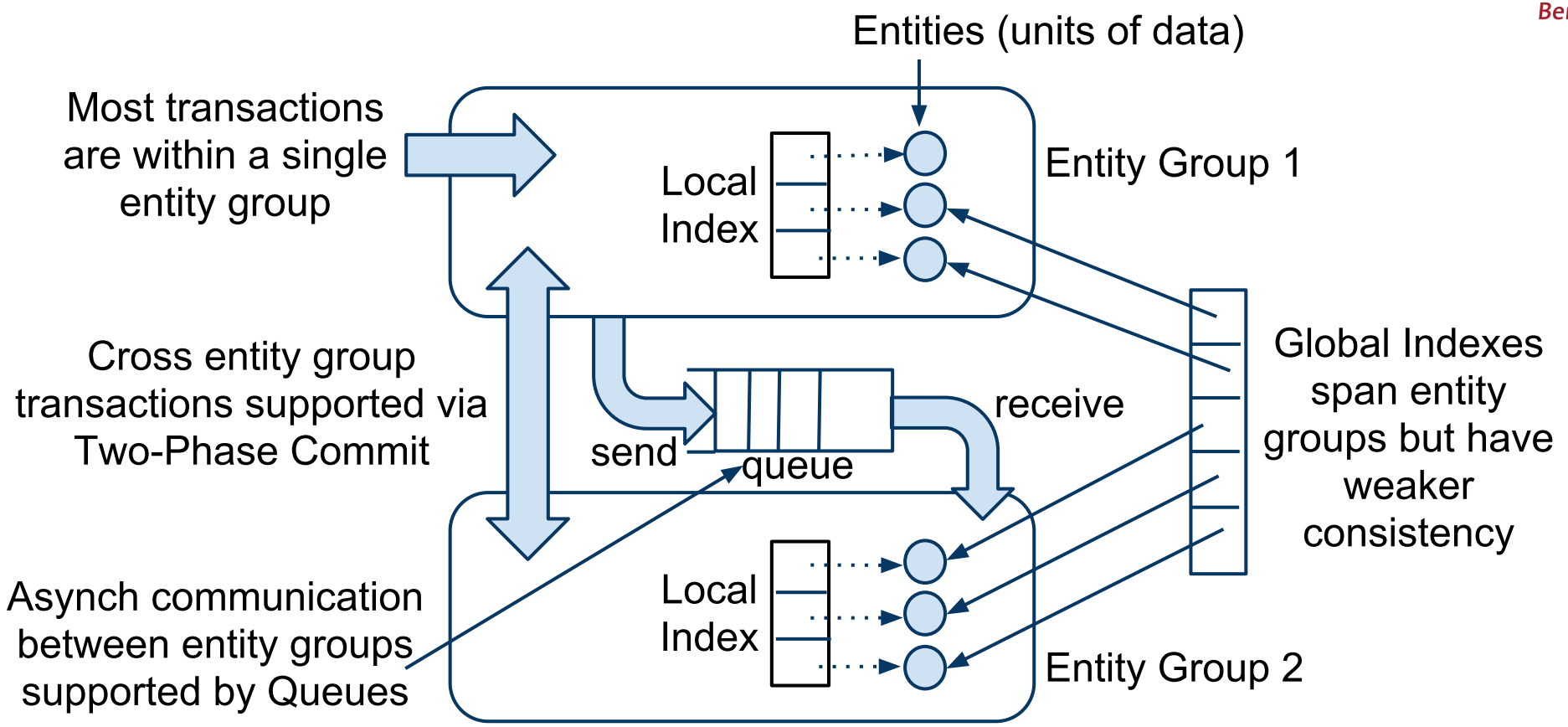


Figure 2: Operations Across Entity Groups

Technische Universität Berlin

Data model: RDBMS + NoSQL?

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

```
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

Transactions and concurrency

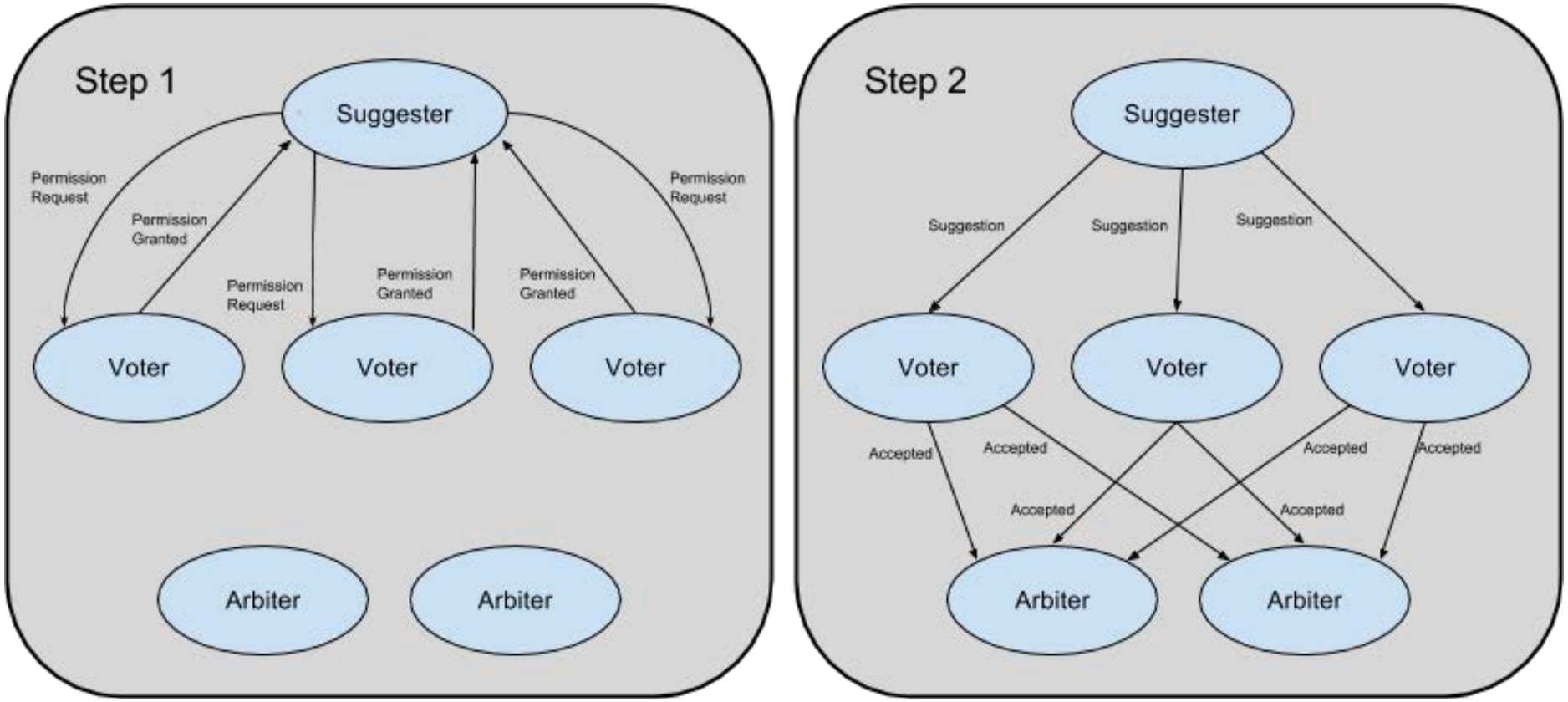
Row key	User. name	Photo. time	Photo. tag	Photo. _url
101	John			
101,500		12:30:01	Dinner, Paris	...
101,502		12:15:22	Betty, Paris	...
102	Mary			

Figure 4: Sample Data Layout in Bigtable

1. Read
2. Application logic
3. Commit
4. Apply
5. Clean up

Multi-version concurrency control

Replication: Paxos



https://understandingpaxos.files.wordpress.com/2016/01/paxos_concomp.png

(modified)
consensus algorithm

Replication

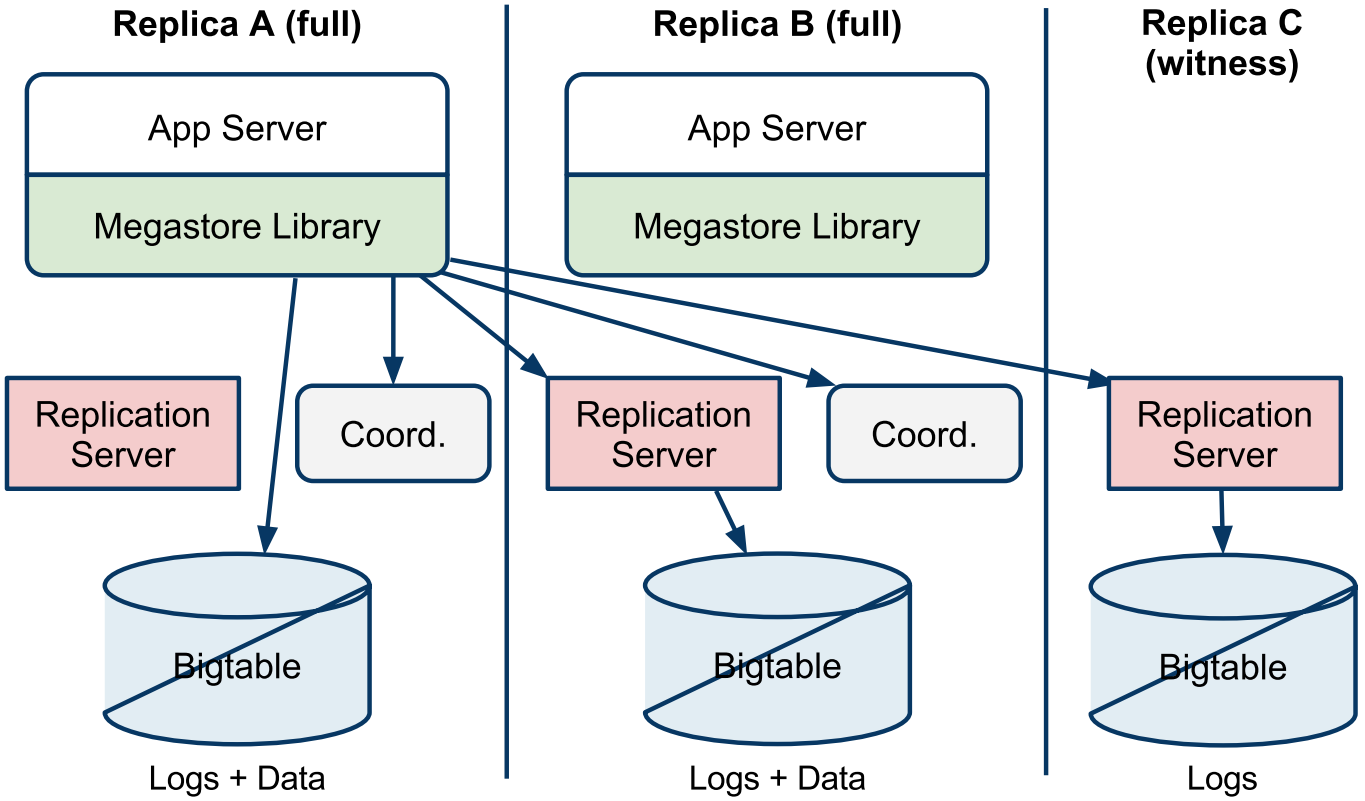


Figure 5: Megastore Architecture Example

- Fast (local) reads
- Fast writes
- Replica types

Replicated logs

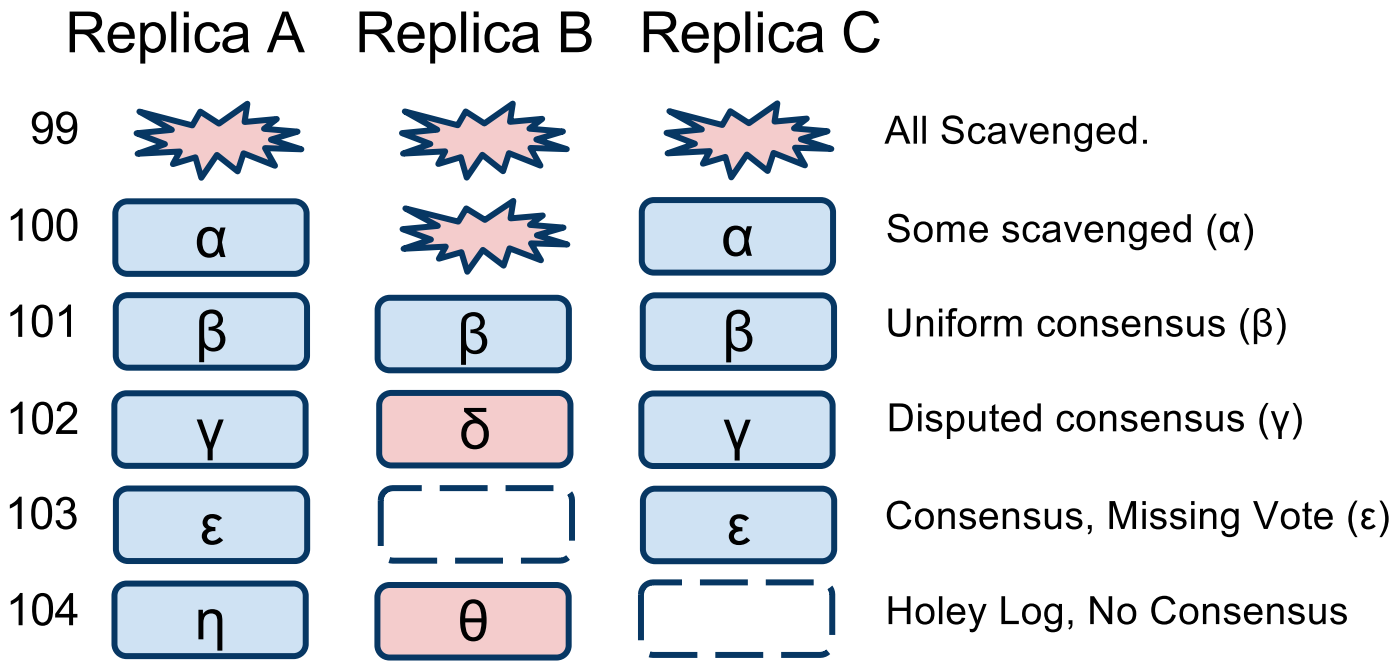


Figure 6: Write Ahead Log

Timelines

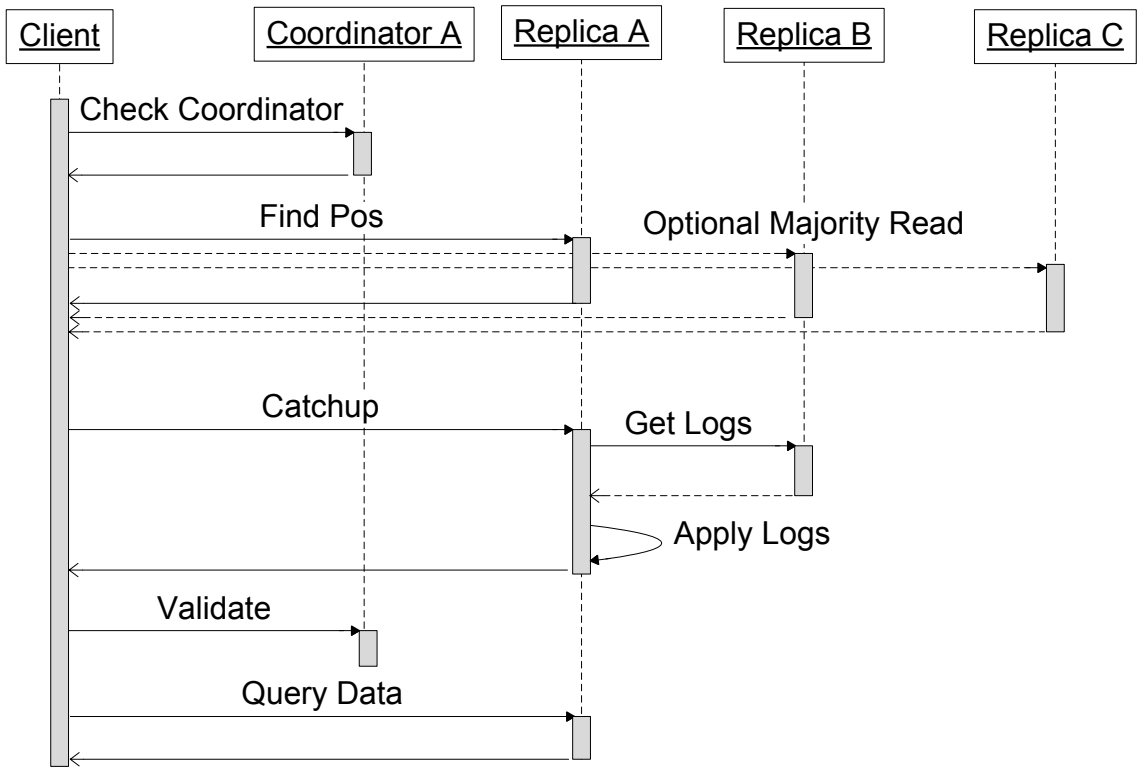


Figure 7: Timeline for reads with local replica A

Timelines *(contd.)*

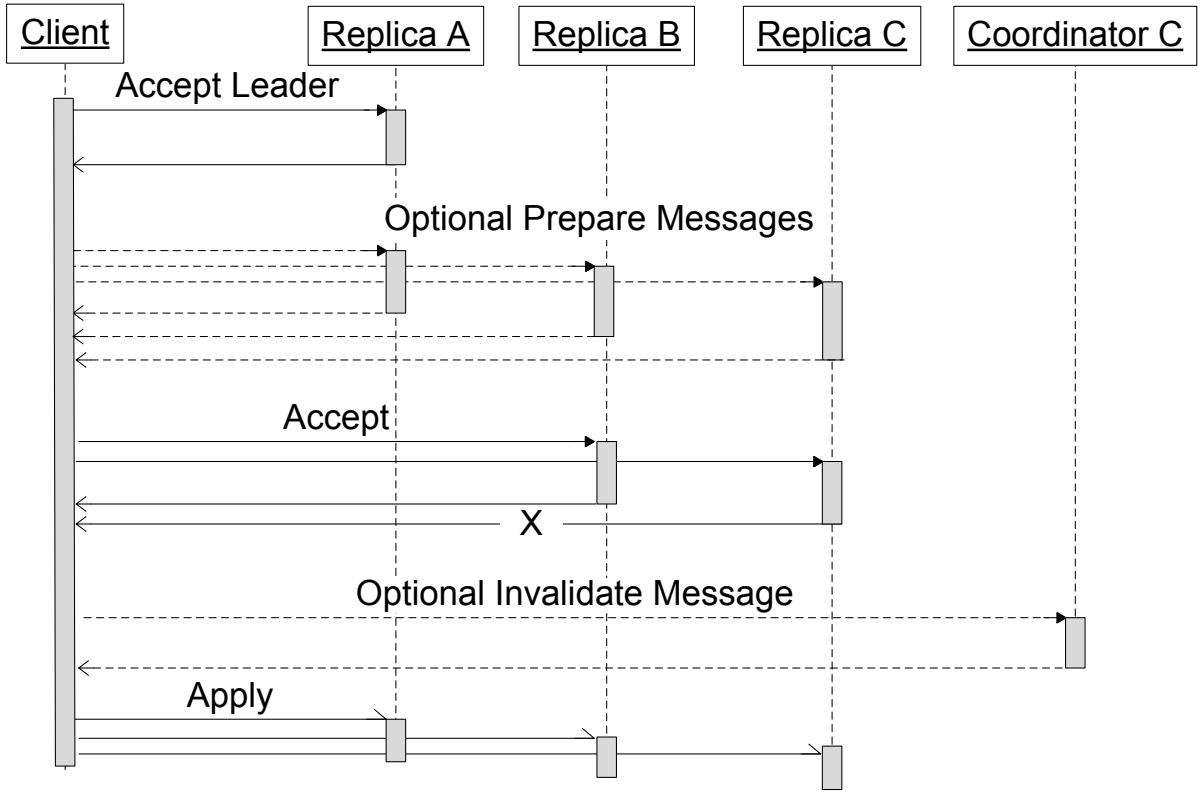


Figure 8: Timeline for writes

Technische Universität Berlin

Scalable + **Tx consistent** + **Wide-area replication**

= ?

https://de.slideshare.net/bergwolf/google-megastore-9042861

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Technische Universität Berlin

“deployed for several years within 100 production applications” @ Google

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Evaluation

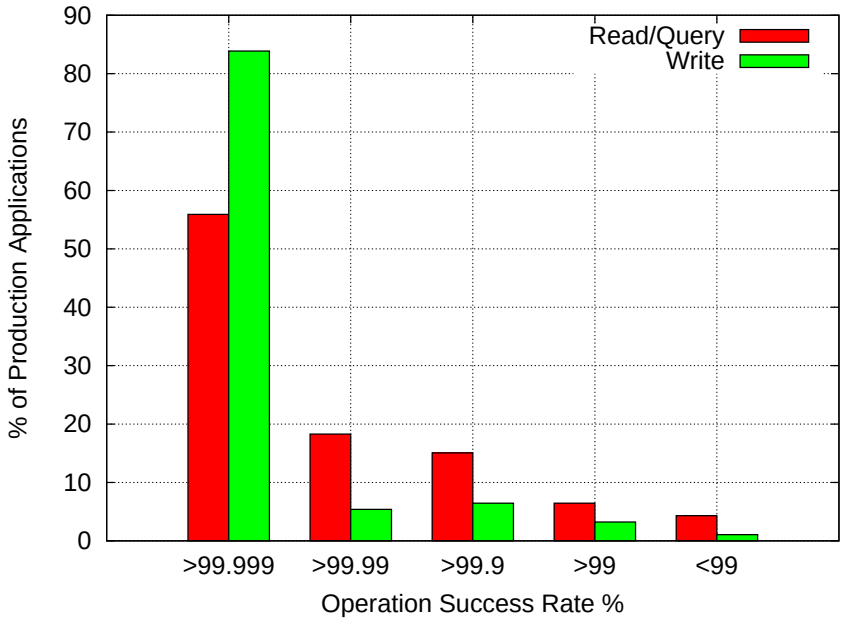


Figure 9: Distribution of Availability

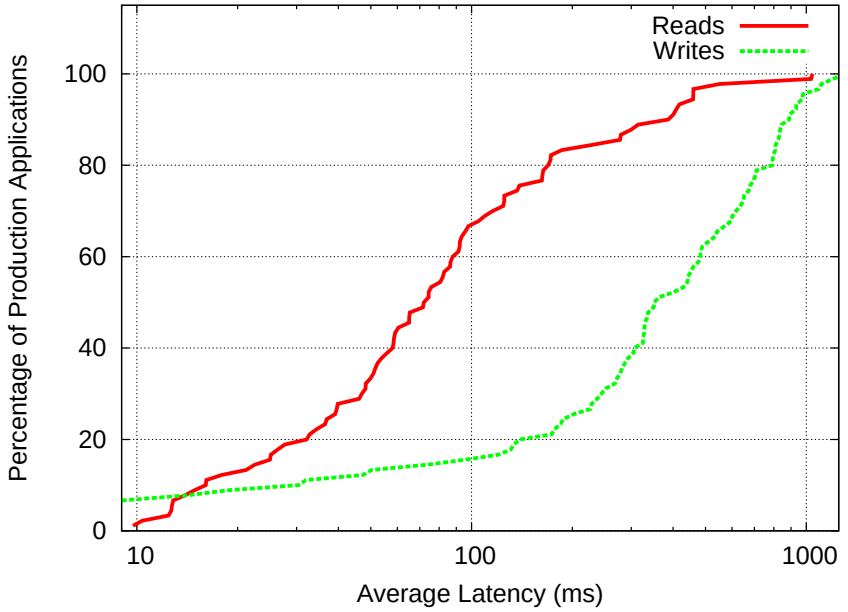


Figure 10: Distribution of Average Latencies

Related work

Yahoo PNUTS

Spanner: Google's Globally Distributed Database

JAMES C. CORBETT, JEFFREY DEAN, MICHAEL EPSTEIN, ANDREW FIKES, CHRISTOPHER FROST, J. J. FURMAN, SANJAY GHEMAWAT, ANDREY GUBAREV, CHRISTOPHER HEISER, PETER HOCHSCHILD, WILSON HSIEH, SEBASTIAN KANTHAK, EUGENE KOGAN, HONGYI LI, ALEXANDER LLOYD, SERGEY MELNIK, DAVID MWAURA, DAVID NAGLE, SEAN QUINLAN, RAJESH RAO, LINDSAY ROLIG, YASUSHI SAITO, MICHAL SZYMANIAK, CHRISTOPHER TAYLOR, RUTH WANG, and DALE WOODFORD, Google, Inc.

8

Spanner is Google's scalable, multiversion, globally distributed, and synchronously replicated database. It is the first system to distribute data at global scale and support externally-consistent distributed transactions. This article describes how Spanner is structured, its feature set, the rationale underlying various design decisions, and a novel time API that exposes clock uncertainty. This API and its implementation are critical to supporting external consistency and a variety of powerful features: nonblocking reads in the past, lock-free snapshot transactions, and atomic schema changes, across all of Spanner.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Concurrency, distributed databases, transaction processing*

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Distributed databases, concurrency control, replication, transactions, time management

ACM Reference Format:
Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., Hsieh, W., Kanthak, S., Kogan, E., Li, H., Lloyd, A., Melnik, S., Mwaura, D., Nagle, D., Quinlan, S., Rao, R., Rolig, L., Saito, Y., Szymaniak, M., Taylor, C., Wang, R., and Woodford, D. 2013. Spanner: Google's globally distributed database. *ACM Trans. Comput. Syst.* 31, 3, Article 8 (August 2013), 22 pages.
DOI: http://dx.doi.org/10.1145/2491245

1. INTRODUCTION

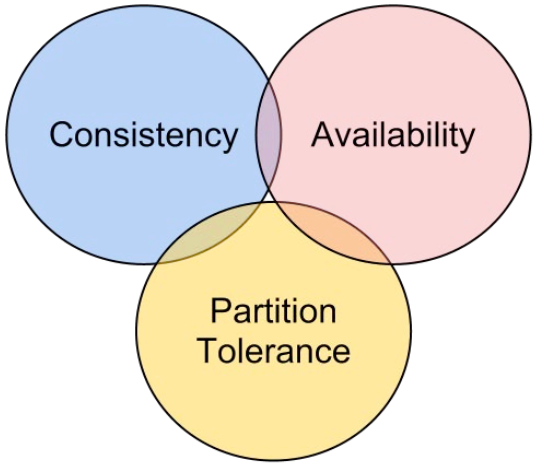
Spanner is a scalable, globally distributed database designed, built, and deployed at Google. At the highest level of abstraction, it is a database that shards data across many sets of Paxos [Lamport 1998] state machines in datacenters spread all over the world. Replication is used for global availability and geographic locality; clients automatically failover between replicas. Spanner automatically reshards data across machines as the amount of data or the number of servers changes, and it automatically migrates data across machines (even across datacenters) to balance load and in

---

This article is essentially the same (with minor reorganizations, corrections, and additions) as the paper of the same title that appeared in the Proceedings of OSDI 2012.

Authors' address: J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh (corresponding author), S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, Google, Inc. 1600 Amphitheatre Parkway, Mountain View, CA 94043; email: wilsonh@google.com.

0734-2071/2013/08-ART8
DOI: http://dx.doi.org/10.1145/2491245



Technische Universität Berlin

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Can the approach be used in Fog Computing?

Technische Universität Berlin

Pro

- ”Natural” entity groups for layouting *(cf. 2.2.2)*
- Minimized latency, data *near* users *(cf. 2.2.3)*
- *Reads* dominate writes *(cf. 3.1)*
- Current, snapshot or inconsistent reads *(cf. 3.3)*

Con

- Location-awareness for moving objects? *(cf. 4.4.2)*
- Limited *write throughput* *(cf. 4.8)*
- Operational issues due to *full replica failures* *(cf. 4.9)*
- *Today’s adoption, topicality, documentation and proprietary software*

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

References

Baker, J., Bond, C., Corbett, J. C., Furman, J. J., Khorlin, A., Larson, J., ... & Yushprakh, V. (2011). Megastore: Providing scalable, highly available storage for interactive services.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., ... & Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, *26*(2), 4.

Furman, J., Karlsson, J. S., Leon, J. M., Lloyd, A., Newman, S., & Zeyliger, P. (2008, June). Megastore: A scalable data system for user facing applications. In *ACM SIGMOD/PODS Conference*.

Lakshman, A., & Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, *44*(2), 35-40.

Pokorny, J. (2013). NoSQL databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, *9*(1), 69-82.