Google Spanner

2012 9

E-mail: ziyulin@xmu.edu.cn http://www.cs.xmu.edu.cn/linziyu

Spanner

Spanner API API API

Google Spanner, Bigtable, distributed database

1

Spanner

Spanner Paxos[21]

Spanner

Spanner

Spanner

F1[35]

F1 5
3-5
1-2

Spanner

BigTable[9] BigTable

[37]
Megastore[5]
Megastore Spanner
BigTable

Spanner
SQL
Spanner

Spanner Spanner

Spanner MapReduce [12]

Spanner

T1 T2
T1 T2 Spanner

TrueTime API API
Spanner API
Spanner
TrueTime API 10ms
GPS
2 Spanner 3
TrueTime API 4 Spanner TrueTime
5
Spanner TrueTime F1 6 7 8

2

Spanner

Spanner

Spanner universe Spanner
universe universe /
universe universe
Spanner zone zone BigTable
zone zone
zone
zone zone

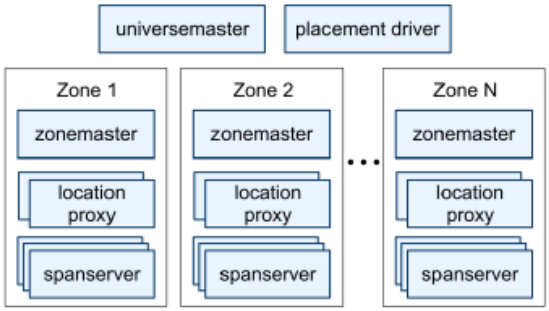

Figure 1: Spanner server organization.

1 Spanner universe zone zonemaster
spanserver Zonemaster spanserver spanserver
zone location proxy
spanserver Universe master placement driver Universe master
zone Placement
driver spanserver

2.1 Spanserver

spanserver
BigTable 2 spanserver 100-1000
tablet tablet BigTable tablet

(key:string, timestamp:int64)->string

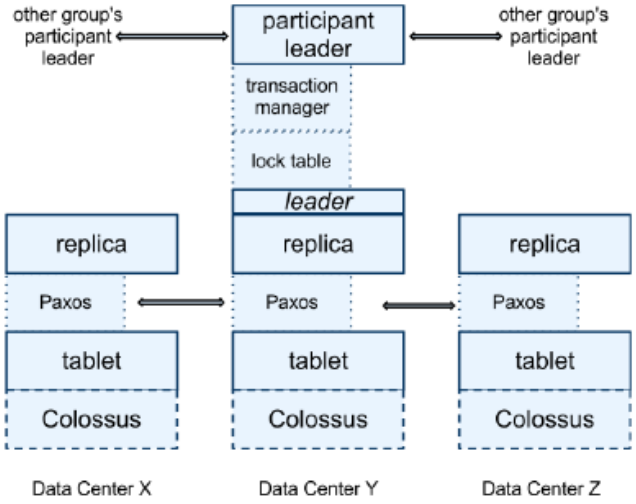

Figure 2: Spanserver software stack.

BigTable Spanner
Spanner tablet
B- (write-ahead)
Colossus Google File System
spanserver tablet Paxos
Spanner tablet Paxos

tablet Paxos
0 10 Spanner Paxos
tablet Paxos
Paxos
Spanner Paxos
Paxos
tablet Paxos
tablet Paxos
group
spanserver

Paxos BigTable Spanner

spanserver
participant leader
participant slaves Paxos
Paxos
Paxos
participant leader coordinator leader participant slaves
coordinator slaves Paxos

2.2

Spanner

2.3

Paxos 3 Spanner

Paxos

50MB

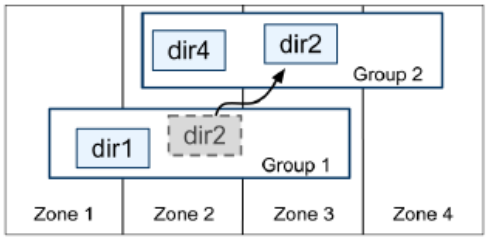

Figure 3: Directories are the unit of data movement between Paxos groups.

Paxos Spanner tablet BigTable tablet Spanner tablet

Movedir Paxos [14] Movedir Paxos [25] Spanner Paxos Movedir

Movedir (fact)

Paxos

A B 5

Spanner

Paxos

Movedir

2.3

Spanner

Megastore[5] 300 Megastore BigTable

BigTable

Megastore Gmail,Picasa,Calendar,Android Market, AppEngine Spanner

SQL Dremel[28]

BigTable

Percolator[32]

[9][10][19]

Paxos

universe

Spanner

SQL

Spanner

Spanner

NULL

```
CREATE TABLE Users {
  uid INT64 NOT NULL, email STRING
} PRIMARY KEY (uid), DIRECTORY;

CREATE TABLE Albums {
  uid INT64 NOT NULL, aid INT64 NOT NULL,
  name STRING
} PRIMARY KEY (uid, aid),
  INTERLEAVE IN PARENT Users ON DELETE CASCADE;
```

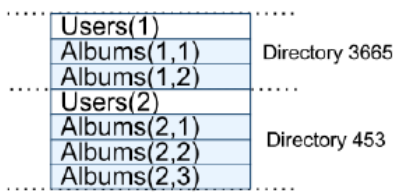

Figure 4: Example Spanner schema for photo metadata, and the interleaving implied by INTERLEAVE IN.

4 Spanner

Megastore Spanner

hierarchy INTERLEAVE IN

K K

ON DELETE CASCADE

interleaved layout Albums(2,1)

Albums user_id=2 album_id=1

Spanner

3 TrueTime

Method	Returns
TT.now()	*TTinterval*: [*earliest*, *latest*]
TT.after(*t*)	true if *t* has definitely passed
TT.before(*t*)	true if *t* has definitely not arrived

Table 1: TrueTime API. The argument *t* is of type *TTstamp*.

TrueTime API

API 1 API TrueTime

TTinterval

“ ” TTinterval TTstamp

TT.now() TTinterval TT.now()

leap-second smearing UNIX

ε, ε̄ TT.after() TT.before() TT.now()

e t_{abs}(e) TrueTime

tt=TT.now() tt.earliest≤t_{abs}(e_{now})≤tt.latest e_{now}

TrueTime GPS TrueTime

GPS

GPS

GPS

TrueTime time master timeslave

daemon master GPS master

master

Armageddon master

Armageddon master GPS master master

master

Armageddon master

GPS master

0

daemon master[29]

master master GPS master GPS master

Armageddon master Daemon Marzullo

[27] master

daemon ε

ε time master time master

ε

ε 1 7ms ε̄ 4ms Daemon

30 200 / 0 6ms

1ms time master

time master ε

ε

4

TrueTime

t t

Spanner Paxos

Paxos Paxos

Paxos

4.1

2 Spanner Spanner

retry retry loop

Operation	Timestamp Discussion	Concurrency Control	Replica Required
Read-Write Transaction	§ 4.1.2	pessimistic	leader
Read-Only Transaction	§ 4.1.4	lock-free	leader for timestamp; any for read, subject to § 4.1.3
Snapshot Read, client-provided timestamp	—	lock-free	any, subject to § 4.1.3
Snapshot Read, client-provided bound	§ 4.1.3	lock-free	any, subject to § 4.1.3

Table 2: Types of reads and writes in Spanner, and how they compare.

[6]

4.1.3

Spanner

retry loop

4.1.1 Paxos

Spanner Paxos 10

Spanner Paxos Paxos

A

Spanner Paxos slave

Spanner

s_{max}

TT.after(s_{max})

4.1.2

Spanner

Paxos Paxos

Spanner Paxos Spanner

Paxos

s s_{max} s

Spanner T_2 T_1

T_2 T_1 T_i

e_i^{start} e_i^{commit} s_i

$t_{abs}(e_1^{commit})<t_{abs}(e_2^{start})\Rightarrow s1<s2$

T_i

e_i^{server}

Start. Ti s_i TT.now().latest

TT.now().latest e_i^{server}

4.2.1

Commit Wait. T_i

TT.after(s_i) s_i T_i 4.2.1

$$
\begin{aligned}
s_1 &< t_{abs}(e_1^{commit}) && \text{(commit wait)}\\
t_{abs}(e_1^{commit}) &< t_{abs}(e_2^{start}) && \text{(assumption)}\\
t_{abs}(e_2^{start}) &\le t_{abs}(e_2^{server}) && \text{(causality)}\\
t_{abs}(e_2^{server}) &\le s_2 && \text{(start)}\\
s_1 &< s_2 && \text{(transitivity)}
\end{aligned}
$$

4.1.3

4.1.2 Spanner

t_{safe}

t t<=t_{safe}

$t_{safe}=min(t_{safe}^{Paxos}, t_{safe}^{TM})$ Paxos t_{safe}^{Paxos}

t_{safe}^{TM} t_{safe}^{Paxos} Paxos

t_{safe}^{Paxos}

0 t_{safe}^{TM}

slave t_{safe}^{TM} slave

Paxos

4.2.1

Ti g

$s_{i,g}^{prepare}$ g

si>=$s_{i,g}^{prepare}$ g g

Ti t_{safe}^{TM} =min_i($s_{i,g}^{prepare}$)-1

4.1.4

s_{read}[8] s_{read}

s_{read}=TT.now().latest 4.1.2

s_{read} t_{safe}

s_{read} s_{read}

s_{max} Spanner

4.2.2

4.2

4.2.1

Bigtable

Spanner

wound-wait [33]

Paxos

s 4.1.3

s TT.now().latest s

Paxos

TT.after(s)

4.1.2

TT.now().latest s

2*ε̄ Paxos

Paxos

4.2.2

Paxos

Spanner scope

Spanner scope

scope Paxos

Spanner Paxos leader

s_{read} Spanner

TT.now().latest LastTS() Paxos

s_{read}=LastTS()

scope Paxos

LastTS() s_{read} Spanner

s_{read}=TT.now().latest

4.2.3

TrueTime Spanner

Bigtable

Spanner

t t

t t

TrueTime t

4.2.4

t_{safe}^{TM} t_{safe}^{TM}

t_{safe}^{TM} key ranges

LastTS()

s_{read}

LastTS() key ranges

LastTS()

t_{safe}^{Paxos} Paxos t
t Paxos Spanner
Paxos t_{safe}^{Paxos}
Paxos n
MinNextTS(n) Paxos n+1
n t_{safe}^{Paxos} -1
MinNextTS() MinNextTS()

MinNextTS()
s_{max} MinNextTS()
8 MinNextTS()
Paxos slave 8
slave MinNextTS()

5.

Spanner
TrueTime F1

5.1

3 Spanner (microbenchmark)
spanserver 4GB CPU AMD Barcelona 2200MHz
zone spanserver zone
1ms
50 Paxos 2500
4KB All reads were served out of memory after a compaction
Spanner

replicas	latency (ms)			throughput (Kops/sec)		
	write	read-only transaction	snapshot read	write	read-only transaction	snapshot read
1D	9.4±.6	—	—	4.0±.3	—	—
1	14.4±1.0	1.4±.1	1.3±.1	4.1±.05	10.9±.4	13.5±.1
3	13.9±.6	1.3±.1	1.2±.1	2.2±.5	13.8±3.2	38.5±.3
5	14.4±.4	1.4±.05	1.3±.04	2.8±.3	25.3±5.2	50.0±1.1

Table 3: Operation microbenchmarks. Mean and standard deviation over 10 runs. 1D means one replica with commit wait disabled.

1 5ms Paxos 9ms
Paxos
slave
CPU

spanserver
spanserver
zone 3 5

4

3 zone zone 25 spanserver 50
99 100

participants	latency (ms)	
	mean	99th percentile
1	17.0 ±1.4	75.0 ±34.9
2	24.5 ±2.5	87.6 ±35.9
5	31.5 ±6.2	104.5 ±52.2
10	30.0 ±3.7	95.6 ±25.4
25	35.5 ±5.6	100.4 ±42.7
50	42.7 ±4.1	93.7 ±22.9
100	71.4 ±7.6	131.2 ±17.6
200	150.5 ±11.0	320.3 ±35.1

Table 4: Two-phase commit scalability. Mean and standard deviations over 10 runs.

5.2

5 Spanner

universe 5 zone Zi zone 25 spanserver
1250 Paxos 100 50K
Z1 5 zone
non-leader Z2 leader-hard Z1 leader-soft Z1

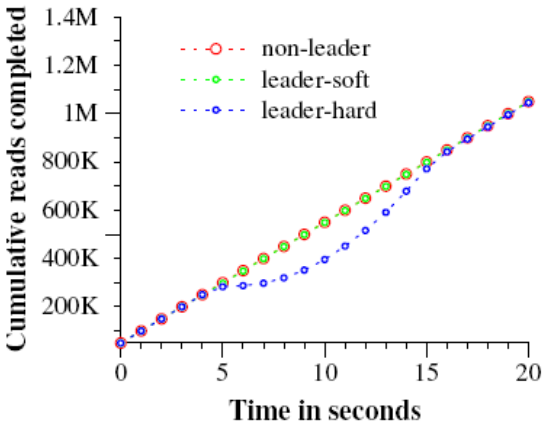

Figure 5: Effect of killing servers on throughput.

Z2 Z1
zone 3-4%
Z1 0
100K

Paxos 10ms zone
10
10

slave Paxos

5.3 TrueTime

TrueTime ε ε
200us/sec
TrueTime CPU
6
TrueTime Spanner
6 TrueTime spanserver spanserver
2200 ε 90 99 99.9
timemaster timeslave daemon
ε timemaster 0

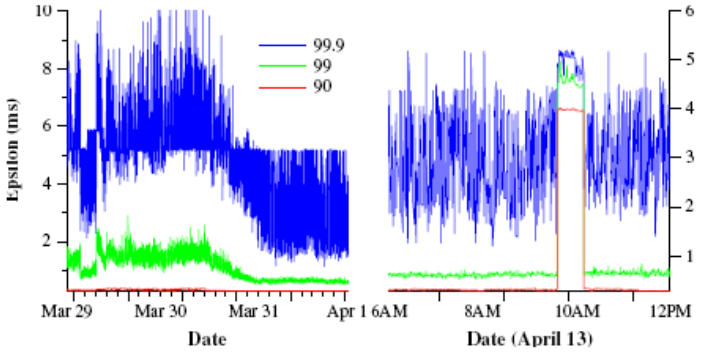

Figure 6: Distribution of TrueTime ϵ values, sampled right after timeslave daemon polls the time masters. 90th, 99th, and 99.9th percentiles are graphed.

6 ε
ε 3 30
4 13 ε
1 time master
TrueTime

5.4 F1

Spanner 2011 F1[35]
MySQL
TB NoSQL
MySQL MySQL

MySQL
Bigtable

F1 Spanner Spanner
Spanner MySQL master-slave
F1
NoSQL F1
Spanner
Spanner
F1 Spanner MySQL
F1
Spanner

F1 Spanner Paxos

Spanner F1
F1
Spanner F1

5 F1 F1

100
F1 F1

# fragments	# directories
1	>100M
2–4	341
5–9	5336
10–14	232
15–99	34
100–500	7

Table 5: Distribution of directory-fragment counts in F1.

6 F1 Spanner
Paxos 6 F1
fat
tail Paxos

1.6KB 119KB

operation	latency (ms)		count
	mean	std dev	
all reads	8.7	376.4	21.5B
single-site commit	72.3	112.8	31.2M
multi-site commit	103.0	52.2	32.1M

Table 6: F1-perceived operation latencies measured over the course of 24 hours.

6.

Megastore[5] DynamoDB[3] DynamoDB

region Spanner Megastore
Megastore Megastore
Bigtable Megastore
Paxos
Paxos
Spanner
Pavlo [31] MapReduce[12]
[1][4][7][41]

Spanner
Gifford [16] Scatter[17]
DHT Spanner
Scatter Gray Lamport[18] Paxos

Walter[36]

Calvin[40]
HStore[39] Granola[11]

Spanner
VoltDB[42]
NewSQL
SQL[38]
Marklogic[26] Oracle' Total Recall[30] Lomet Li[24]

Faresite [13]
TrueTime Farsite Spanner Paxos
[2][23]
TrueTime Paxos

7.

F1 MySQL
Spanner
Spanner

Paxos [22]34]

TrueTime ε ε 1ms
Time-master-query Time-master-query

Spanner
SQL

8.

Spanner

SQL

Spanner

5

Spanner

Bigtable

TrueTime API

API

Jon Howell Atul Adya, Fay Chang, Frank Dabek, Sean Dorward, Bob Gruber, David Held, Nick Kline, Alex Thomson, and Joel Wein.

Aristotle Balogh, Bill Coughran, Urs H ̈olzle, Doron Meyer, Cos Nicolaou, Kathy Polizzi, Sridhar Ramaswany, and Shivakumar Venkataraman.

Bigtable Megastore F1 Jeff Shute

Platforms Luiz Barroso

Bob Felderman TrueTime

Ken Ashcraft, Paul Cychosz, Krzysztof Ostrowski, Amir Voskoboynik, Matthew Weaver, Theo Vassilakis, and Eric Veach; or have joined our team recently: Nathan Bales, Adam Beberg, Vadim Borisov, Ken Chen, Brian Cooper, Cian Cullinan, Robert-Jan Huijsman, Milind Joshi, Andrey Khorlin, Dawid Kuroczko, Laramie Leavitt, Eric Li, Mike Mammarella, Sunil Mushran, Simon Nielsen, Ovidiu Platon, Ananth Shrinivas, Vadim Suvorov, and Marcel van der Holst.

[1] Azza Abouzeid et al. “HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads”. Proc. of VLDB. 2009, pp. 922–933.
[2] A. Adya et al. “Efficient optimistic concurrency control using loosely synchronized clocks”. Proc. of SIGMOD. 1995, pp. 23–34.
[3] Amazon. Amazon DynamoDB. 2012.
[4] Michael Armbrust et al. “PIQL: Success-Tolerant Query Processing in the Cloud”. Proc. of VLDB. 2011, pp. 181–192.
[5] Jason Baker et al. “Megastore: Providing Scalable, Highly Available Storage for Interactive Services”. Proc. of CIDR. 2011, pp. 223–234.
[6] Hal Berenson et al. “A critique of ANSI SQL isolation levels”. Proc. of SIGMOD. 1995, pp. 1–10.
[7] Matthias Brantner et al. “Building a database on S3”. Proc. of SIGMOD. 2008, pp. 251–264.

[8] A. Chan and R. Gray. “Implementing Distributed Read-Only Transactions”. IEEE TOSE SE-11.2 (Feb. 1985), pp. 205–212.
[9] Fay Chang et al. “Bigtable: A Distributed Storage System for Structured Data”. ACM TOCS 26.2 (June 2008), 4:1–4:26.
[10] Brian F. Cooper et al. “PNUTS: Yahoo!’s hosted data serving platform”. Proc. of VLDB. 2008, pp. 1277–1288.
[11] James Cowling and Barbara Liskov. “Granola: Low-Overhead Distributed Transaction Coordination”. Proc. of USENIX ATC. 2012, pp. 223–236.
[12] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: a flexible data processing tool”. CACM 53.1 (Jan. 2010), pp. 72–77.
[13] John Douceur and Jon Howell. Scalable Byzantine-Fault-Quantifying Clock Synchronization. Tech. rep. MSR-TR-2003-67. MS Research, 2003.
[14] John R. Douceur and Jon Howell. “Distributed directory service in the Farsite file system”. Proc. of OSDI. 2006, pp. 321–334.
[15] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. “The Google file system”. Proc. of SOSP. Dec. 2003, pp. 29–43.
[16] David K. Gifford. Information Storage in a Decentralized Computer System. Tech. rep. CSL-81-8. PhD dissertation. Xerox PARC, July 1982.
[17] Lisa Glendenning et al. “Scalable consistency in Scatter”. Proc. of SOSP. 2011.
[18] Jim Gray and Leslie Lamport. “Consensus on transaction commit”. ACM TODS 31.1 (Mar. 2006), pp. 133–160.
[19] Pat Helland. “Life beyond Distributed Transactions: an Apostate’s Opinion”. Proc. of CIDR. 2007, pp. 132–141.
[20] Maurice P. Herlihy and Jeannette M. Wing. “Linearizability: a correctness condition for concurrent objects”. ACM TOPLAS 12.3 (July 1990), pp. 463–492.
[21] Leslie Lamport. “The part-time parliament”. ACM TOCS 16.2 (May 1998), pp. 133–169.
[22] Leslie Lamport, Dahlia Malkhi, and Lidong Zhou. “Reconfiguring a state machine”. SIGACT News 41.1 (Mar. 2010), pp. 63–73.
[23] Barbara Liskov. “Practical uses of synchronized clocks in distributed systems”. Distrib. Comput. 6.4 (July 1993), pp. 211–219.
[24] David B. Lomet and Feifei Li. “Improving Transaction-Time DBMS Performance and Functionality”. Proc. of ICDE (2009), pp. 581–591.
[25] Jacob R. Lorch et al. “The SMART way to migrate replicated stateful services”. Proc. of EuroSys. 2006, pp. 103–115.
[26] MarkLogic. MarkLogic 5 Product Documentation. 2012.
[27] Keith Marzullo and Susan Owicki. “Maintaining the time in a distributed system”. Proc. of PODC. 1983, pp. 295–305.
[28] Sergey Melnik et al. “Dremel: Interactive Analysis of Web-Scale Datasets”. Proc. of VLDB. 2010, pp. 330–339.
[29] D.L. Mills. Time synchronization in DCNET hosts. Internet Project Report IEN–173. COMSAT Laboratories, Feb. 1981.
[30] Oracle. Oracle Total Recall. 2012.
[31] Andrew Pavlo et al. “A comparison of approaches to large-scale data analysis”. Proc. of SIGMOD. 2009, pp. 165–178.

[32] Daniel Peng and Frank Dabek. “Large-scale incremental processing using distributed transactions and notifications”. Proc. of OSDI. 2010, pp. 1–15.
[33] Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis II. “System level concurrency control for distributed database systems”. ACM TODS 3.2 (June 1978), pp. 178–198.
[34] Alexander Shraer et al. “Dynamic Reconfiguration of Primary/Backup Clusters”. Proc. of SENIX ATC. 2012, pp. 425–438.
[35] Jeff Shute et al. “F1—The Fault-Tolerant Distributed RDBMS Supporting Google’s Ad Business”. Proc. of SIGMOD. May 2012, pp. 777–778.
[36] Yair Sovran et al. “Transactional storage for geo-replicated systems”. Proc. of SOSP. 2011, pp. 385–400.
[37] Michael Stonebraker. Why Enterprises Are Uninterested in NoSQL. 2010.
[38] Michael Stonebraker. Six SQL Urban Myths. 2010.
[39] Michael Stonebraker et al. “The end of an architectural era: (it’s time for a complete rewrite)”. Proc. of VLDB. 2007, pp. 1150–1160.
[40] Alexander Thomson et al. “Calvin: Fast Distributed Transactions for Partitioned Database Systems”. Proc. of SIGMOD.2012, pp. 1–12.
[41] Ashish Thusoo et al. “Hive — A Petabyte Scale Data Warehouse Using Hadoop”. Proc. of ICDE. 2010, pp. 996–1005.
[42] VoltDB. VoltDB Resources. 2012.

========== ==========