Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Michael Burrows, Tushar Chandra, Andrew Fikes, Robert Gruber

{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com

Google, Inc.

http://www.cs.xmu.edu.cn/linziyu

2010 7

[ChangDGHWBCFG06]Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Michael Burrows, Tushar Chandra, Andrew Fikes, Robert Gruber: Bigtable: A Distributed Storage System for Structured Data (Awarded Best Paper!). OSDI 2006:205-218.

5.4 Compactions

6 Refinements

Locatity groups

compression

Caching for read performance

Bloom filters

Commit-log implementation

Speeding up tablet recovery

Exploiting immutability

7 Performance Evaluation

Single-tablet-server performance

Scaling

8 Real applications

8.1 Google Analytics

8.2 Google Earth

8.3 Personalized Search

9 Lessons

10 Related Work

11 Conclusions

Acknowledgements

References

[
http://dblab.xmu.edu.cn/cloud_database_view]

Abstract

BigTable PB
Google BigTable WEB Google
Earth Google Finance BigTable
URL
BigTable Google
BigTable
BigTable

1 Introduction

BigTable
Google BigTable PB
BigTable Google
BigTable Google Analytics Google Finance Orkut
Personalized Search Writely Google Earth BigTable

BigTable TB
BigTable
[14] [13] BigTable
BigTable

BigTable
BigTable
BigTable

2 3 API 4
BigTable Google 5 BigTable 6
BigTable 7 BigTable
8 Google BigTable 9
BigTable 10 11

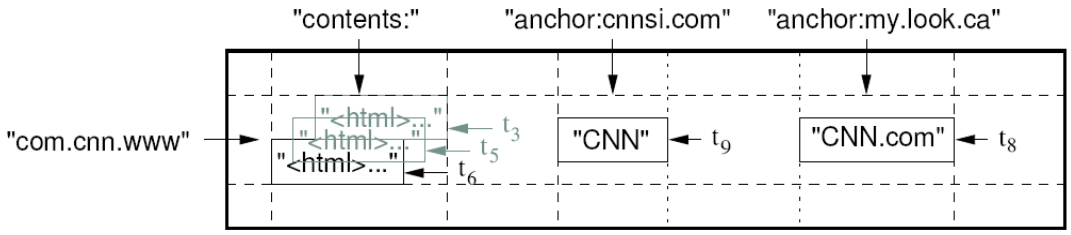
2 Data model

BigTable row key
column key timestamp

(row:string, column string, time:int64) string
BigTable

Webtable
Webtable URL

contents:column 1



1 Webtable URL contents anchor anchor CNN Sports Illustrated MY-look "anchor:cnnsi.com" "anchor:my.look.ca" anchor contents t3,t5 t6

ROWS

64KB 10-100

BigTable

Tablet

Webtable URL

com.google.maps/index.html com.google.maps/index.html

Column Families

family:qualifier

qualifier Webtable language

language

ID Webtable anchor

anchor 1 qualifier

Webtable

Timestamps

BigTable

BitTable 64 BigTable

BigTable

garbage-collect n

7

Webtable contents:column

3 API

BigTable API

BigTable

2 C++ RowMutation

Apply

Webtable anchor www.cnn.com

anchor

```
// Open the table
Table *T = OpenOrDie("/bigtable/web/webtable");
// Write a new anchor and delete an old anchor
RowMutation r1(T, "com.cnn.www");
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");
Operation op;
Apply(&op, &r1);
```

Figure 2: Writing to Bigtable.

3 C++ Scanner anchor

anchor

anchor:*.cnn.com 10 anchor

```
Scanner scanner(T);
ScanStream *stream;
stream = scanner.FetchColumnFamily("anchor");
stream->SetReturnAllVersions();
scanner.Lookup("com.cnn.www");
for (; !stream->Done(); stream->Next()) {
printf("%s %s %lld %s\n",
scanner.RowName(),
stream->ColumnName(),
stream->MicroTimestamp(),
```

```
stream->Value());
}
```

Figure 3: Reading from Bigtable.

BigTable BigTable
- - BigTable

BigTable BigTable
Sawzall Google
Sawzall API BigTable

BigTable MapReduce[12] MapReduce Google
Wrapper BigTable
MapReduce

4 Building Blocks

BigTable Google BigTable Google
GFS[17] BigTable
BigTable
BigTable

Google SSTable BigTable SSTable
BigTable
/
SSTable 64KB
SSTable SSTable

SSTable
BigTable Chubby[8] Chubby
5
Chubby Paxos [9][23]
Chubby
Chubby
Chubby Chubby session session Chubby
session session
session Chubby
Chubby callback session
BigTable Chubby 1
2 BigTable bootstrap 5.1 3 tablet
4 tablet 5 BigTable
6 Chubby BigTable
11 Chubby 14 BigTable

Chubby Chubby BigTable BigTable 0.0047% 0.0326%

5 Implementation

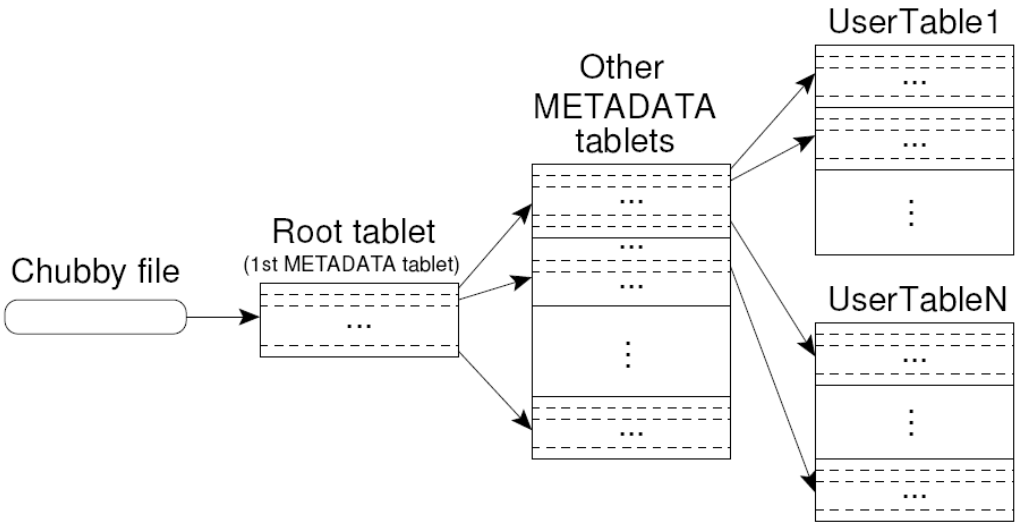
BigTable (1) 2 3 Tablet Tablet Tablet Tablet Tablet Table GFS

Tablet Tablet Tablet 10 1000 Tablet Tablet Tablet Tablet [17,21] Tablet BigTable Tablet

BigTable Tablet Tablet Tablet Tablet 100 200MB

5.1 Tablet Location

B+ 4 Tablet



4 Tablet

Chubby Toot Tablet Root Tablet Tablet METADATA METADATA user tablet Root Tablet METADATA

Tablet Tablet

METADATA Tablet Tablet
METADATA 1KB
128M METADATA Tablet
2 34 Tablet
Tablet Tablet
Tablet Tablet

Chubby
entry METADATA Tablets
Tablets GFS
tablet
METADATA Tablet
METADATA tablet
tablet

5.2 Tablet Assignment

Tablet tablet tablet
tablet tablet tablet
tablet tablet tablet
tablet tablet tablet
tablet
BigTable Chubby tablet Tablet
Chubby
tablet tablet
tablet Chubby
Chubby tablet
tablet
tablet
tablet
tablet
tablet
tablet tablet
tablet tablet tablet
tablet tablet
tablet

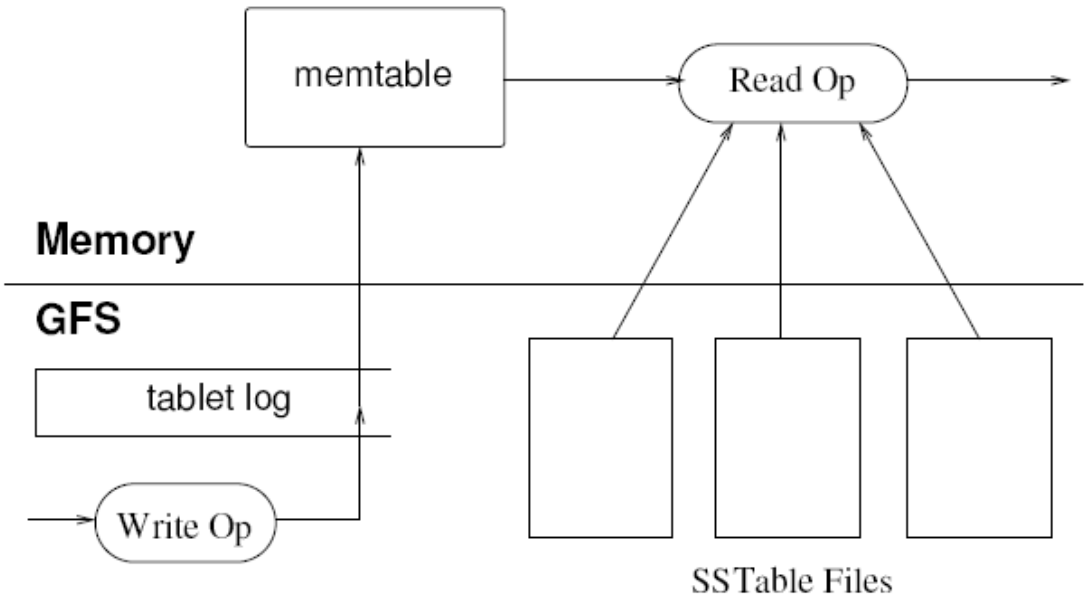
Chubby tablet
Chubby Chubby tablet
tablet

tablet tablet BigTable
Chubby Chubby
tablet
table
tablet
tablet 1
Chubby master lock 2
Chubby 3
tablet tablets tablet 4
METADATA tablets tablet
tablet tablet tablet

METADATA tablets
METADATA 4 3
root tablet root tablet tablet
root tablet root tablet METADATA tablets
root tablet METADATA tablets
tablet 1 tablet 2
tablet tablet 3 tablet
tablet Tablet tablet
Tablet tablet METADATA
tablet
tablet tablet
tablet tablet Tablet
METADATA tablet entry

5.3 Tablet Serving

tablet GFS 5
redo
memtable SSTable tablet
tablet METADATA tablet SSTable
SSTable tablet redo point redo point
tablet SSTable
memtable
tablet
Chubby
Chubby Chubby
[13,16]
memtable



5 Tablet Representation

Tablet

SSTable

memtable SSTable memtable

tablet

5.4 Compactions

memtable memtable

memtable memtable memtable

SSTable GFS minor compaction 1

tablet 2

SSTable

SSTable

SSTable memtable

SSTable SSTable

memtable

SSTable SSTable

major compaction SSTable

entry SSTable

SSTable BigTable

tablet BigTable

6 Refinements

Locality groups

locality group tablet
locality group SSTable
locality group WebTable
locality group locality
group
locality group locality
group locality group SSTable
tablet SSTable
SSTable locality group

Compression

locality group SSTable
SSTable
locality group

Bentley and McIlroy[6]

16KB
100-200MB/S 400-1000MB/S

WebTable
locality group

10 1 GZip GZip
HTML 3:1 4:1 WebTable

Bentley and McIlroy
WebTable
BigTable

Caching for read performance

tablet Scan
- - tablet SSTable Block
GFS SSTable Scan
Block

locality group

Bloom filters

5.3 tablet SSTable
SSTable
locality group SSTable
Bloom filter[7] Bloom filter SSTabble
- Bloom filter tablet
Bloom filter

Commit-log implementation

tablet GFS
GFS
tablet
tablet
tablet tablet
[18][20]

tablet tablet tablet
tablet tablet tablet tablet
tablet tablet
tablet
tablet tablet
100
tablet tablet
100

entry tablet

64MB
tablet

tablet

GFS GFS
GFS tablet

Speeding up tablet recovery

tablet tablet tablet tablet
tablet minor compaction tablet

tablet tablet tablet
tablet tablet
tablet
tablet

Exploiting immutability

SSTable BigTable
SSTable SSTable

memtable memtable
memtable ”copy-on-write”

SSTable SSTable
tablet SSTable METADATA SSTable

SSTable tablet tablet
SSTable tablet tablet SSTable

7 Performance Evaluation

BigTable N tablet N
BigTable Tablet 1GB GFS
GFS 1786 400GB IDE N
BigTable
tablet
Opteron 2GHz GB

100-200Gbps

1

Tablet GFS

GFS tablet

R BigTable R benchmark

tablet 1GB

benchmark 0 R-1 10N

N

benchmark

R benchmark

benchmark benchmark

benchmark

benchmark benchmark

benchmark

Scan benchmark benchmark BigTable API

scan benchmark RPC

Scan RPC tablet

(mem) benchmark benchmark benchmark

in-memory GFS

benchmark tablet 1GB

100MB tablet

Experiment	# of Tablet Servers			
	1	50	250	500
random reads	1212	593	479	241
random reads (mem)	10811	8511	8000	6250
random writes	8850	3745	3425	2000
sequential reads	4425	2463	2625	2469
sequential writes	8547	3623	2451	1905
scans	15385	10526	9524	7843

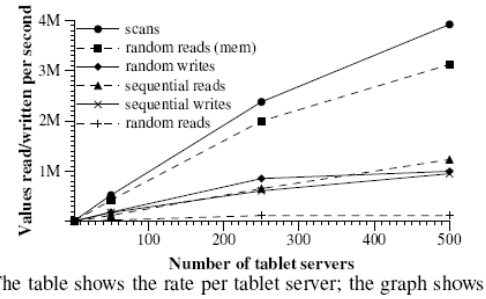


Figure 6: Number of 1000-byte values read/written per second. The table shows the rate per tablet server; the graph shows the aggregate rate.

6 1000 BigTable benchmark

tablet

Single tablet server performance

tablet

64KB SSTable

GFS tablet 1000 Tablet
1200 GFS 75MB/s
BigTable 8KB
1000 tablet
GFS 64KB
tablet
GFS

GFS 64KB SSTable
64
Scan RPC tablet RPC

Scaling

tablet 1 500
100 tablet 500
300 benchmark tablet
CPU
benchmark tablet 1
50
CPU

tablet tablet
benchmark
benchmark 500
100 1000
64KB GB

# of tablet servers	# of clusters
0 .. 19	259
20 .. 49	47
50 .. 99	20
100 .. 499	50
> 500	12

Table 1: Distribution of number of tablet servers in Bigtable clusters.

Project name	Table size (TB)	Compression ratio	# Cells (billions)	# Column Families	# Locality Groups	% in memory	Latency-sensitive?
Crawl	800	11%	1000	16	8	0%	No
Crawl	50	33%	200	2	2	0%	No
Google Analytics	20	29%	10	1	1	0%	Yes
Google Analytics	200	14%	80	1	1	0%	Yes
Google Base	2	31%	10	29	3	15%	Yes
Google Earth	0.5	64%	8	7	2	33%	Yes
Google Earth	70	–	9	8	3	0%	No
Orkut	9	–	0.9	8	5	1%	Yes
Personalized Search	4	47%	6	93	11	5%	Yes

Table 2: Characteristics of a few tables in production use. *Table size* (measured before compression) and *# Cells* indicate approximate sizes. *Compression ratio* is not given for tables that have compression disabled.

8 Real Applications

2006 8 388 BigTable Google
24500 tablet 1 tablet
14
8069 tablet 120 RPC
741MB/s RPC 16GB/s 2

BigTable

8.1 Google Analytics

Google Analytics
URL

Javascript
Javascript
Google Analytics Google Analytics

Google Analytics 200TB
session
WEB session
14%
20TB
MapReduce MapReduce
session GFS 29%

8.2 Google Earth

Google
Google Maps Google Earth

70TB
BigTable

BigTable MapReduce
MapReduce 1MB/
GFS
500GB
tablet

8.3 Personalized search

Personalized search
Google
Google
Personalized search BigTable
ID ID

BigTable Personalized search BigTable MapReduce profile
profile
Personalized Search BigTable
Personalized Search BigTable

Personalized Search
Google
BigTable

9 Lessons

BigTable

Chubby
GFS
RPC

checksumming

Chubby

API

BigTable

BigTable

BigTable BigTable RPC

RPC RPC

tablet BigTable

METADATA tablet METADATA

BigTable Chubby

10

tablet

tablet tablet

Chubby

Chubby

10 Related Work

Boxwood [24] Chubby GFS BigTable Boxwood

B Boxwood

Google Boxwood

BigTable

CAN[29] CHORD[32] Tapestry[37]

Pastry[30] BigTable

BigTable

B

- -

-

Oracle Real Application Cluster [27] BigTable GFS BigTable Chubby IBM DB2 Parallel Edition[4] [33] BigTable DB2 IBM DB2 Parallel Edition Oracle Real Application Cluster

BigTable C-store[1,34] Sybase IQ[15,36] SenSage[31] KDB+[32] MonetDB/X100[38] ColumnBM AT&T Daytona [19] CPU Ailamaki[2]

BigTable memtable SSTable tablet Log-Structured Merge Tree[26]

C-Store BigTable

API C-Store BigTable C-Store DBMS BigTable

BigTable [11,35]

1

2

3

11 Conclusions

BigTable Google 2005 4 BigTable 7 person-years 2006 8 60 BigTable BigTable

BigTable BigTable BigTable Google BigTable BigTable BigTable BigTable BigTable

BigTable Google BigTable

BigTable

BigTable Google

Acknowledgements

We thank the anonymous reviewers, David Nagle, and our shepherd Brad Calder, for their feedback on this paper. The Bigtable system has benefited greatly from the feedback of our many users within Google. In addition, we thank the following people for their contributions to Bigtable: Dan Aguayo, Sameer Ajmani, Zhifeng Chen, Bill Coughran, Mike Epstein, Healfdene Goguen, Robert Griesemer, Jeremy Hylton, Josh Hyman, Alex Khesin, Joanna Kulik, Alberto Lerner, Sherry Listgarten, Mike Maloney, Eduardo Pinheiro, Kathy Polizzi, Frank Yellin, and Arthur Zwiegincew.

References

[1] ABADI, D. J., MADDEN, S. R., AND FERREIRA,M. C. Integrating compression and execution in columnoriented database systems. Proc. of SIGMOD (2006).

[2] AILAMAKI, A., DEWITT, D. J., HILL, M. D., AND SKOUNAKIS, M. Weaving relations for cache performance. In The VLDB Journal (2001), pp. 169–180.

[3] BANGA, G., DRUSCHEL, P., AND MOGUL, J. C. Resource containers: A new facility for resource management in server systems. In Proc. of the 3rd OSDI (Feb.1999), pp. 45–58.

[4] BARU, C. K., FECTEAU, G., GOYAL, A., HSIAO,H., JHINGRAN, A., PADMANABHAN, S., COPELAND, G. P., AND WILSON, W. G. DB2 parallel edition. IBM Systems Journal 34, 2 (1995), 292–322.

[5] BAVIER, A., BOWMAN, M., CHUN, B., CULLER, D., KARLIN, S., PETERSON, L., ROSCOE, T., SPALINK, T.,AND WAWRZONIAK, M. Operating system support for planetary-scale network services. In Proc. of the 1st NSDI (Mar. 2004), pp. 253–266.

[6] BENTLEY, J. L., AND MCILROY, M. D. Data compression using long common strings. In Data Compression Conference (1999), pp. 287–295.

[7] BLOOM, B. H. Space/time trade-offs in hash coding with allowable errors. CACM 13, 7 (1970), 422–426.

[8] BURROWS, M. The Chubby lock service for looselycoupled distributed systems. In Proc. of the 7th OSDI (Nov. 2006).

[9] CHANDRA, T., GRIESEMER, R., AND REDSTONE, J. Paxos made live — An engineering perspective. In Proc. of PODC (2007).

[10] COMER, D. Ubiquitous B-tree. Computing Surveys 11, 2 (June 1979), 121–137.

[11] COPELAND, G. P., ALEXANDER, W., BOUGHTER, E. E., AND KELLER, T. W. Data placement in Bubba. In Proc. of SIGMOD (1988), pp. 99–108.

[12] DEAN, J., AND GHEMAWAT, S. MapReduce: Simplified data processing on large clusters. In Proc. of the 6th OSDI (Dec. 2004), pp. 137–150.

[13] DEWITT, D., KATZ, R., OLKEN, F., SHAPIRO, L., STONEBRAKER, M., AND WOOD, D. Implementation techniques for main memory database systems. In Proc. of SIGMOD (June 1984), pp. 1–8.

[14] DEWITT, D. J., AND GRAY, J. Parallel database systems: The future of high performance database systems. CACM 35, 6 (June 1992), 85–98.

[15] FRENCH, C. D. One size fits all database architectures do not work for DSS. In Proc. of SIGMOD (May 1995), pp. 449–450.

[16] GAWLICK, D., AND KINKADE, D. Varieties of concurrency control in IMS/VS fast path. Database Engineering Bulletin 8, 2 (1985), 3–10.

[17] GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S.-T. The Google file system. In Proc. of the 19th ACM SOSP (Dec.2003), pp. 29–43.

[18] GRAY, J. Notes on database operating systems. In Operating Systems — An Advanced Course, vol. 60 of Lecture Notes in Computer Science. Springer-Verlag, 1978.

[19] GREER, R. Daytona and the fourth-generation language Cymbal. In Proc. of SIGMOD (1999), pp. 525–526.

[20] HAGMANN, R. Reimplementing the Cedar file system using logging and group commit. In Proc. of the 11th SOSP (Dec. 1987), pp. 155–162.

[21] HARTMAN, J. H., AND OUSTERHOUT, J. K. The Zebra striped network file system. In Proc. of the 14th SOSP (Asheville, NC, 1993), pp. 29–43.

[22] KX.COM. kx.com/products/database.php. Product page.

[23] LAMPORT, L. The part-time parliament. ACM TOCS 16,2 (1998), 133–169.

[24] MACCORMICK, J., MURPHY, N., NAJORK, M., THEKKATH, C. A., AND ZHOU, L. Boxwood: Abstractions as the foundation for storage infrastructure. In Proc. of the 6th OSDI (Dec. 2004), pp. 105–120.

[25] MCCARTHY, J. Recursive functions of symbolic expressions and their computation by machine. CACM3, 4 (Apr. 1960), 184–195.

[26] O'NEIL, P., CHENG, E., GAWLICK, D., AND O'NEIL,E. The log-structured merge-tree (LSM-tree). Acta Inf.33, 4 (1996), 351–385.

[27] ORACLE.COM. www.oracle.com/technology/products/-database/clustering/index.html. Product page.

[28] PIKE, R., DORWARD, S., GRIESEMER, R., AND QUINLAN, S. Interpreting the data: Parallel analysis with Sawzall. Scientific Programming Journal 13, 4 (2005),227–298.

[29] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SHENKER, S. A scalable content-addressable network. In Proc. of SIGCOMM (Aug. 2001), pp. 161–172.

[30] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems. In Proc. of Middleware 2001 (Nov. 2001), pp. 329–350.

[31] SENSAGE.COM. sensage.com/products-sensage.htm. Product page.

[32] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proc. of SIGCOMM (Aug. 2001), pp. 149–160.

[33] STONEBRAKER, M. The case for shared nothing. Database Engineering Bulletin 9, 1 (Mar. 1986), 4–9.

[34] STONEBRAKER, M., ABADI, D. J., BATKIN, A., CHEN, X., CHERNIACK, M., FERREIRA, M., LAU, E., LIN, A., MADDEN, S., O'NEIL, E., O'NEIL, P., RASIN, A., TRAN, N., AND ZDONIK, S. C-Store: A columnoriented DBMS. In Proc. of VLDB (Aug. 2005), pp. 553–564.

[35] STONEBRAKER, M., AOKI, P. M., DEVINE, R., LITWIN, W., AND OLSON, M. A. Mariposa: A new architecture for distributed data. In Proc. of the Tenth ICDE(1994), IEEE Computer Society, pp. 54–65.

[36] SYBASE.COM. www.sybase.com/products/databaseservers/sybaseiq. Product page.

[37] ZHAO, B. Y., KUBIATOWICZ, J., AND JOSEPH, A. D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, CS Division, UC Berkeley, Apr. 2001.

[38] ZUKOWSKI, M., BONCZ, P. A., NES, N., AND HEMAN, S. MonetDB/X100—A DBMS in the CPU cache. IEEE Data Eng. Bull. 28, 2 (2005), 17–22.

[
http://dblab.xmu.edu.cn/cloud_database_view]