Bigtable

15799 - Advanced Topics in DB

Mu Li
muli@cs.cmu.edu

October 2, 2013

stolen slides from Jeff Dean (a lot) and Edward Yoon (one)
Typical New Engineer

- Never seen a petabyte of data
- Never used a thousand machines
- Never really experienced machine failure

*Our software has to make them successful.*
Data Storage: BigTable

What is it, really?

- 10-ft view: Row & column abstraction for storing data
- Reality: Distributed, persistent, multi-level sorted map
Comparing with Dynamo

- Data Model: Table vs. Key-Value
- Consistency: Atomic row mutation vs. record-at-a-time and eventual consistency
- How to run: Centralized management vs. each application run its own instance
- Access control vs. No
- Focus: Easy to use vs. availability
Topics will be covered today:

- Data Model and API
- System Overviews
- Implementation of Tablet Servers
- Current State of Bigtable

Things are ignored

- Refinement to improved the performance
- Experiments
BigTable Data Model

- Multi-dimensional sparse sorted map
  
  $(row, column, timestamp) \Rightarrow value$
BigTable Data Model

- Multi-dimensional sparse sorted map
  \((row, column, timestamp) \Rightarrow value\)
BigTable Data Model

- Multi-dimensional sparse sorted map
  \[(row, column, timestamp) \Rightarrow value\]
BigTable Data Model

- Multi-dimensional sparse sorted map
  \( (row, column, timestamp) \Rightarrow value \)
BigTable Data Model

- Multi-dimensional sparse sorted map
  
  \((row, column, timestamp) \rightarrow value\)
BigTable Data Model

- Multi-dimensional sparse sorted map
  \((row, column, timestamp) \Rightarrow value\)

```
Rows

Columns

“contents:”

```

```
Timeslabels

“www.cnn.com”

```

```

“<html>...”

```
BigTable Data Model

- Multi-dimensional sparse sorted map
  \[(row, column, timestamp) \Rightarrow value\]
API

▶ Atomic single row mutation

```cpp
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);
```

▶ scanning cells by rows, column, and timestamp

```cpp
Scanner scanner(T);
ScanStream *stream;
stream = scanner.FetchColumnFamily("anchor");
stream->SetReturnAllVersions();
scanner.Lookup("com.cnn.www");
for (; !stream->Done(); stream->Next()) {}
```

▶ server-side scripts: Sawzall

▶ integration with other products: Mapreduce, Pregel, Parameter Server, ....
Current Design

- In-house rack design
- PC-class motherboards
- Low-end storage and networking hardware
- Linux
- + in-house software
Tablets (cont.)

```
"aaa.com"
"cnn.com"
"cnn.com/sports.html"
"website.com"
"yahoo.com/kids.html"
"zuppa.com/menu.html"
```

```
language:
contents:

EN
<html>...
```
Tablets (cont.)

“language:”

“aaa.com”
“cnn.com”
“cnn.com/sports.html”

“contents:”

Tables

“website.com”
“yahoo.com/kids.html”
“yahoo.com/kids.html”
“zuppa.com/menu.html”
Typical Cluster

Machine 1
- Scheduler slave
- GFS chunkserver
- Linux

Machine 2
- Scheduler slave
- GFS chunkserver
- Linux

Machine N
- Scheduler slave
- GFS chunkserver
- Linux
Typical Cluster

Cluster scheduling master
Chubby Lock service
GFS master

Machine 1

Scheduler slave
GFS chunkserver
Linux

Machine 2

Scheduler slave
GFS chunkserver
Linux

Machine N

Scheduler slave
GFS chunkserver
Linux
Bigtable System Structure

Bigtable Cell

Bigtable master

Bigtable tablet server  Bigtable tablet server  ...  Bigtable tablet server
Bigtable System Structure

Bigtable Cell

Bigtable master
performs metadata ops + load balancing

Bigtable tablet server  Bigtable tablet server ... Bigtable tablet server
Bigtable System Structure

Bigtable Cell

- **Bigtable master**
  - performs metadata ops + load balancing

- **Bigtable tablet server**
  - serves data

- **Bigtable tablet server**
  - serves data

- **Bigtable tablet server**
  - serves data
Bigtable System Structure

**Bigtable Cell**

- **Bigtable master**
  - performs metadata ops + load balancing

- **Bigtable tablet server**
  - serves data

- **Bigtable tablet server**
  - serves data

- **Bigtable tablet server**
  - serves data

---

- **Cluster scheduling system**
- **GFS**
- **Lock service**
Bigtable System Structure

Bigtable Cell

Bigtable master
performs metadata ops + load balancing

Bigtable tablet server
serves data

Bigtable tablet server
serves data

Bigtable tablet server
serves data

Cluster scheduling system
handles failover, monitoring

GFS

Lock service
Bigtable System Structure

Bigtable Cell

Bigtable master
performs metadata ops + load balancing

Bigtable tablet server
serves data

Bigtable tablet server
serves data

Bigtable tablet server
serves data

Cluster scheduling system
handles failover, monitoring

GFS
holds tablet data, logs

Lock service
Bigtable System Structure

**Bigtable Cell**

- **Bigtable master**: performs metadata ops + load balancing
- **Bigtable tablet server**, serves data
- **Bigtable tablet server**, serves data
- **Bigtable tablet server**, serves data

**Cluster scheduling system**: handles failover, monitoring

**GFS**: holds tablet data, logs

**Lock service**: holds metadata, handles master-election
Bigtable System Structure

**Bigtable Cell**

- **Bigtable master**
  performs metadata ops + load balancing

- **Bigtable tablet server**
  serves data

- **Cluster scheduling system**
  handles failover, monitoring

- **GFS**
  holds tablet data, logs

- **Lock service**
  holds metadata, handles master-election
Bigtable System Structure

Bigtable Cell

- Bigtable master
  - performs metadata ops + load balancing

- Bigtable tablet server
  - serves data

- Bigtable tablet server
  - serves data

- Bigtable tablet server
  - serves data

- Cluster scheduling system
  - handles failover, monitoring

- GFS
  - holds tablet data, logs

- Lock service
  - holds metadata, handles master-election

Open()
Bigtable System Structure

Bigtable Cell

Bigtable client
Bigtable client library

Bigtable master
performs metadata ops + load balancing

Open()
read/write

Bigtable tablet server
serves data

... 

Bigtable tablet server
serves data

Bigtable tablet server
serves data

Cluster scheduling system
handles failover, monitoring

GFS
holds tablet data, logs

Lock service
holds metadata, handles master-election
Find Tablet Location

- Three-level hierarchy analogous to a B+ tree
  - 1st level: bootstrapped from chubby
  - 2nd level: use META0 tablet to find the owner of appropriate META 1 tablets
  - 3rd level: META1 table holds locations of all other tablets
Master assigns tablets to tablet servers

- Master keeps track of the set of live tablet servers the current assignment of tablets to region servers, including which tablets are unassigned.

1) Start a server
2) Create a lock
3) Acquire the lock
4) Monitor
5) Assign tablets
6) Check lock status
7) Acquire and Delete the lock
8) Reassign unassigned tablets
A tablet is maintained by one tablet server.
A tablet consists of several SSTable blocks with an index to store the first and last key of the block, and stored in GFS.

- **SSTable 64Kb**
- **SSTable 64Kb**
- **SSTable 64Kb**

**index**
Tablet Serving

- Updates are stored in tablet log and
  - new ones in memtable (in memory)
  - old ones in SSTables
- For read, first check memtable, then SSTables
Compaction

minor compaction
  ▶ convert memtable into an new SSTable and write into disk
  ▶ save memory

merging (major) compaction
  ▶ read several SSTable and memtable and merging into a few (exact one) SSTable
  ▶ save disk due to high compression rate, remove deleted entries
BigTable Status

- Production use for 100+ projects:
  - Crawling/indexing pipeline
  - Google Maps/Google Earth
  - My Search History
  - Google Print
  - Orkut
  - Blogger
  - ...

- Currently ~500 BigTable clusters

- Largest cluster:
  - 70+ PB data; sustained: 10M ops/sec; 30+ GB/s I/O
BigTable: What’s New Since OSDI’06?

- Lots of work on scaling
- Service clusters, managed by dedicated team
- Improved performance isolation
  - fair-share scheduler within each server, better accounting of memory used per user (caches, etc.)
  - can partition servers within a cluster for different users or tables

- Improved protection against corruption
  - many small changes
  - e.g. immediately read results of every compaction, compare with CRC.
    - Catches ~1 corruption/5.4 PB of data compacted
BigTable Replication (New Since OSDI’06)

- Configured on a per-table basis

- Typically used to replicate data to multiple bigtable clusters in different data centers

- *Eventual consistency model*: writes to table in one cluster eventually appear in all configured replicas

- Nearly all user-facing production uses of BigTable use replication
BigTable Coprocessors (New Since OSDI ’06)

• Arbitrary code that runs next to each tablet in table
  – as tablets split and move, coprocessor code automatically splits/moves too

• High-level call interface for clients
  – Unlike RPC, calls addressed to rows or ranges of rows
    • coprocessor client library resolves to actual locations
  – Calls across multiple rows automatically split into multiple parallelized RPCs

• Very flexible model for building distributed services
  – automatic scaling, load balancing, request routing for apps
Example Coprocessor Uses

- Scalable filesystem metadata management for Colossus (next gen GFS-like file system)
- Distributed language model serving for machine translation system
- Distributed query processing for full-text indexing support
- Regular expression search support for code repository
- ...