Bigtable: A Distributed Storage System for Structured Data

Course: CS655

Rabiet Louis

Colorado State University

Thursday 17 October 2013

Plan

Reminders on relational database model

SQL family

Reminders on relational database model

SQL family

First order predicate

Reminders on relational database model

SQL family

First order predicate

- Based on two values: true and false.

Reminders on relational database model

SQL family

First order predicate

- Based on two values: true and false.
- Use tuples and relation

Reminders on relational database model

SQL family

First order predicate

- Based on two values: true and false.
- Use tuples and relation
  - Tuples: (Louis, CS Student, French) - (Louis, GRA)

Reminders on relational database model

SQL family

First order predicate

- Based on two values: true and false.
- Use tuples and relation
  - Tuples: (Louis, CS Student, French) - (Louis, GRA)
  - Relation = Set of attributes and sets of tuples.

Reminders on relational database model

SQL family

First order predicate

- Based on two values: true and false.
- Use tuples and relation
  - Tuples: (Louis, CS Student, French) - (Louis, GRA)
  - Relation = Set of attributes and sets of tuples.
    Attr. = (Id, Major, Nationality).

Reminders on relational database model

SQL family

First order predicate

- Based on two values: true and false.
- Use tuples and relation
  - Tuples: (Louis, CS Student, French) - (Louis, GRA)
  - Relation = Set of attributes and sets of tuples.
    Attr. = (Id, Major, Nationality).
- A query = a formula

Reminders on relational database model

SQL family

First order predicate

- Based on two values: true and false.
- Use tuples and relation
  - Tuples: (Louis, CS Student, French) - (Louis, GRA)
  - Relation = Set of attributes and sets of tuples.
    Attr. = (Id, Major, Nationality).
- A query = a formula
  $\forall students, Nationality = French \ \&\& \ Position = GRA$

Hierarchical Databases

Hierarchical Databases

- ✗ Tree structure.

Hierarchical Databases

- ✗ Tree structure.
- ✓ Efficient implementations.

Hierarchical Databases

- ✗ Tree structure.
- ✓ Efficient implementations.

Object Databases

Hierarchical Databases

- ✗ Tree structure.
- ✓ Efficient implementations.

Object Databases

- ✓ Close to programming langages.

Hierarchical Databases

- ✗ Tree structure.
- ✓ Efficient implementations.

Object Databases

- ✓ Close to programming langages.
- ✗ Overkill if the data has simple relations.

Why is this not enough ?

How to express query on a text ?

How to express query on a text ?

You have to reconstruct some kind of relation more or less manually even if some solutions exists [JDG08].

How to express query on a text ?

You have to reconstruct some kind of relation more or less manually even if some solutions exists [JDG08].

How do you do if a single server is not enough ?

How to express query on a text ?

You have to reconstruct some kind of relation more or less manually even if some solutions exists [JDG08].

How do you do if a single server is not enough ?

- Partioning $\rightarrow$ write expensive

How to express query on a text ?

You have to reconstruct some kind of relation more or less manually even if some solutions exists [JDG08].

How do you do if a single server is not enough ?

- Partioning $\rightarrow$ write expensive
  small scheme showing why it requires two-phase commit

How to express query on a text ?

You have to reconstruct some kind of relation more or less manually even if some solutions exists [JDG08].

How do you do if a single server is not enough ?

- Partioning $\rightarrow$ write expensive
  small scheme showing why it requires two-phase commit
- Replications $\rightarrow$ write expensive

Plan

BigData

Usage of the architecture

- URLs

BigData

Usage of the architecture

- URLs
- Locations

BigData

Usage of the architecture

- URLs
- Locations
- Data Personalized: settings, search

BigData

Usage of the architecture

- URLs
- Locations
- Data Personalized: settings, search

BigData

Usage of the architecture

- URLs
- Locations
- Data Personalized: settings, search

Some ideas

- Goal is to let users handle data storage structure

BigData

Usage of the architecture

- URLs
- Locations
- Data Personalized: settings, search

Some ideas

- Goal is to let users handle data storage structure
- Locality is important

BigData

Usage of the architecture

- URLs
- Locations
- Data Personalized: settings, search

Some ideas

- Goal is to let users handle data storage structure
- Locality is important
- A data = an uninterpreted string

BigData

Usage of the architecture

- URLs
- Locations
- Data Personalized: settings, search

Some ideas

- Goal is to let users handle data storage structure
- Locality is important
- A data = an uninterpreted string
- Goes nicely with Map Reduce.

Challenges

- Unstructured Data.

Challenges

- Unstructured Data.
- Scale.

Challenges

- Unstructured Data.
- Scale.
- Continuous Update (crawling).

Challenges

- Unstructured Data.
- Scale.
- Continuous Update (crawling).
- Read should be allowed at any time.

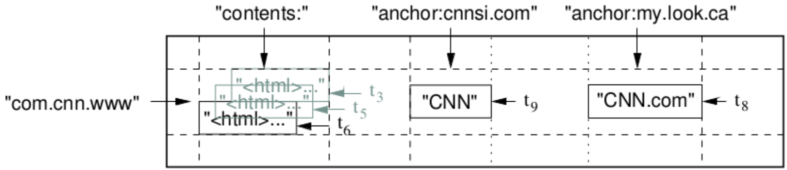
Challenges

- Unstructured Data.
- Scale.
- Continuous Update (crawling).
- Read should be allowed at any time.
- Very high rates of accesses $\rightarrow$ load-balancing.

Challenges

- Unstructured Data.
- Scale.
- Continuous Update (crawling).
- Read should be allowed at any time.
- Very high rates of accesses $\rightarrow$ load-balancing.
- Failure resilient (adding and removing servers at any time).

Data Model


"contents:"
"anchor:cnnsi.com"
"anchor:my.look.ca"
"com.cnn.www"
"<html>..."
"<html>..."
"<html>..."
t_3
t_5
t_6
"CNN"
t_9
"CNN.com"
t_8

Column-oriented database

Properties

Column-oriented database

Properties

- Row writes are atomic.

Column-oriented database

Properties

- Row writes are atomic.
- Row ranges (tablet) are grouped together dynamically

Column-oriented database

Properties

- Row writes are atomic.
- Row ranges (tablet) are grouped together dynamically
- Sorted using lexicographic order

Column-oriented database

Properties

- Row writes are atomic.
- Row ranges (tablet) are grouped together dynamically
- Sorted using lexicographic order
  - close number $\rightarrow$ physically close.

Column-oriented database

Properties

- Row writes are atomic.
- Row ranges (tablet) are grouped together dynamically
- Sorted using lexicographic order
  - close number $\rightarrow$ physically close.
- Column family is other group.

Column-oriented database

Properties

- Row writes are atomic.
- Row ranges (tablet) are grouped together dynamically
- Sorted using lexicographic order
  - close number $\rightarrow$ physically close.
- Column family is other group.
- Several versions: usage of timestamp.

Column-oriented database

Properties

- Row writes are atomic.
- Row ranges (tablet) are grouped together dynamically
- Sorted using lexicographic order
  - close number $\rightarrow$ physically close.
- Column family is other group.
- Several versions: usage of timestamp.
  - last n versions.

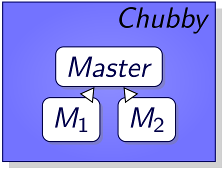
Column-oriented database

Properties

- Row writes are atomic.
- Row ranges (tablet) are grouped together dynamically
- Sorted using lexicographic order
  - close number $\rightarrow$ physically close.
- Column family is other group.
- Several versions: usage of timestamp.
  - last n versions.
  - fresh enough (age limit).

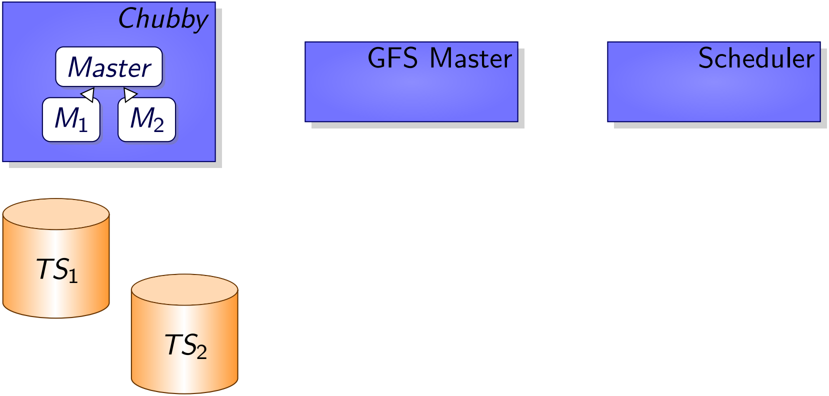
Architecture

Architecture

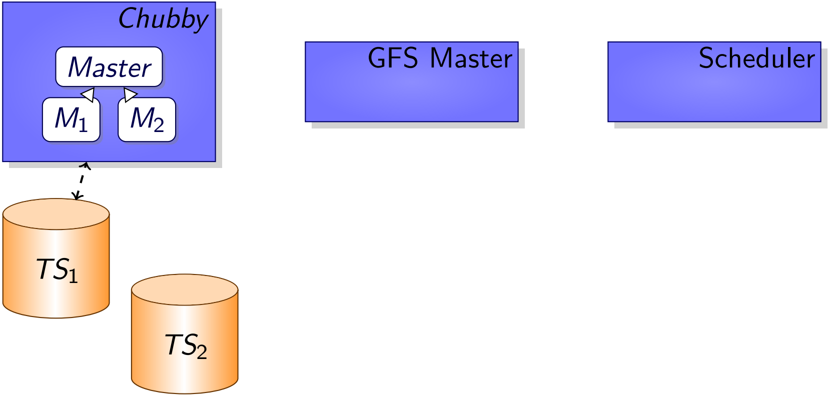


Architecture

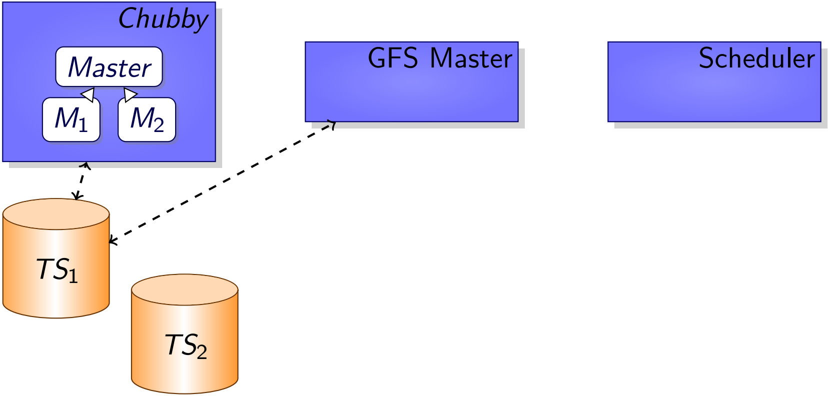
Architecture



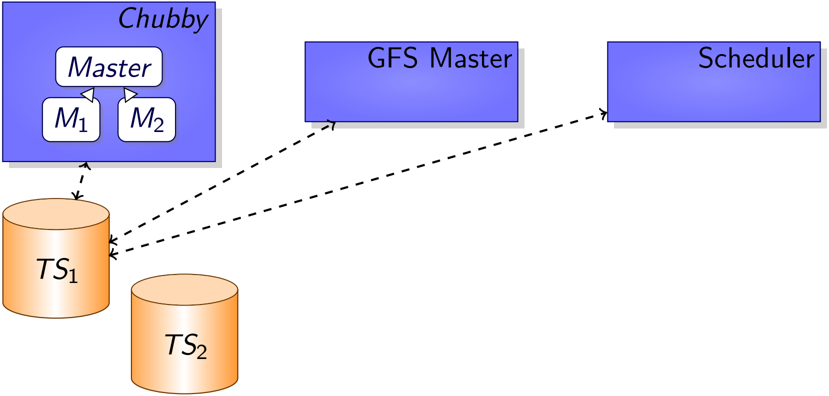
Architecture



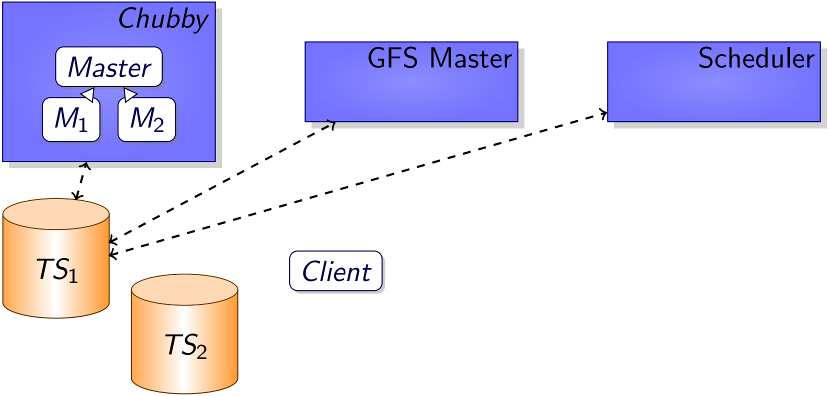
Architecture



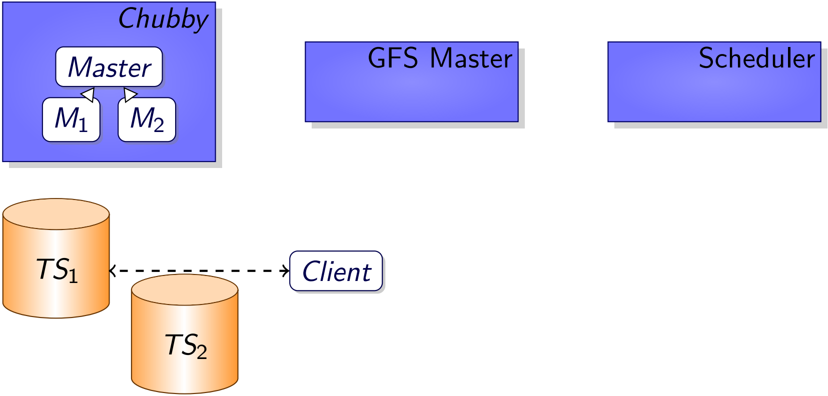
Architecture



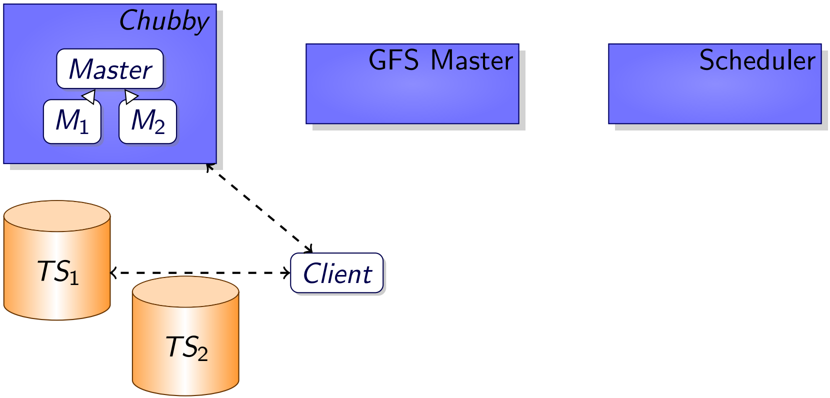
Architecture



Architecture



Architecture



Internal Storage

Key	Value	Key	Value	Key	Value	Index

Internal Storage

Key Value **Key** Value **Key** Value **Index**

- Use a key:value storage format (SStable: Sorted String Table).

Internal Storage

Key	Value	**Key**	Value	**Key**	Value	**Index**

- Use a key:value storage format (SStable: Sorted String Table).
- SStable are immutables.

Internal Storage

Key	Value	**Key**	Value	**Key**	Value	**Index**

- Use a key:value storage format (SStable: Sorted String Table).
- SStable are immutables.
  - Every time a write is done: a new table is created.

Internal Storage

Key	Value	**Key**	Value	**Key**	Value	**Index**

- Use a key:value storage format (SStable: Sorted String Table).
- SStable are immutables.
  - Every time a write is done: a new table is created.
  - ✓ Concurrency is easier: No synchronisation, No need to control the concurrent accesses.

Internal Storage

Key	Value	**Key**	Value	**Key**	Value	**Index**

- Use a key:value storage format (SStable: Sorted String Table).
- SStable are immutables.
  - Every time a write is done: a new table is created.
  - ✓ Concurrency is easier: No synchronisation, No need to control the concurrent accesses.
  - ✓ Can help for serialization, for snapshot.

Internal Storage

Key	Value	**Key**	Value	**Key**	Value	**Index**

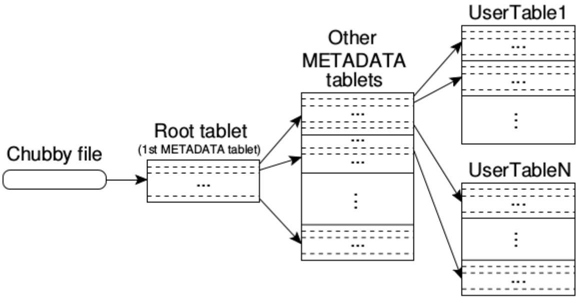
- Use a key:value storage format (SStable: Sorted String Table).
- SStable are immutables.
    - Every time a write is done: a new table is created.
    - ✓ Concurrency is easier: No synchronisation, No need to control the concurrent accesses.
    - ✓ Can help for serialization, for snapshot.
    - ✗ Increase the number of SSTables.

Internal Storage

Key	Value	Key	Value	Key	Value	Index

- Use a key:value storage format (SStable: Sorted String Table).
- SStable are immutables.
  - Every time a write is done: a new table is created.
  - ✓ Concurrency is easier: No synchronisation, No need to control the concurrent accesses.
  - ✓ Can help for serialization, for snapshot.
  - ✗ Increase the number of SSTables.
  - ✗ Persistence can be achieved by differential instead of using a full copy.

Tablets Location



B-Tree+ to store tablet location.

Tablets Assignment and Serving

- No duplication of tablets.

Tablets Assignment and Serving

- No duplication of tablets.
- Master keeps track of assignements to reallocate tablets if needed.

Tablets Assignment and Serving

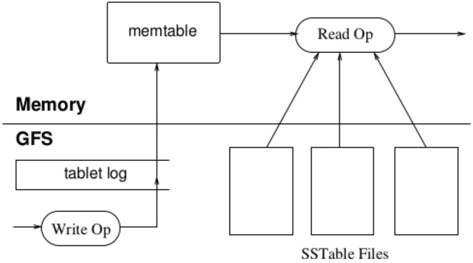
- No duplication of tablets.
- Master keeps track of assignements to reallocate tablets if needed.
- A TS acquires a lock in Chubby per tablet.

Tablets Assignment and Serving

- No duplication of tablets.
- Master keeps track of assignements to reallocate tablets if needed.
- A TS acquires a lock in Chubby per tablet.
- Authorization is given by the writers list in Chubby.

Tablets Assignment and Serving

- No duplication of tablets.
- Master keeps track of assignements to reallocate tablets if needed.
- A TS acquires a lock in Chubby per tablet.
- Authorization is given by the writers list in Chubby.



Reminder on GFS

- Moderate number of Huge Files

Reminder on GFS

- Moderate number of Huge Files
- A lot of failures

Reminder on GFS

- Moderate number of Huge Files
- A lot of failures
- Appending $\neq$ random write

Reminder on GFS

- Moderate number of Huge Files
- A lot of failures
- Appending $\neq$ random write
- Concurrent write must be supported

Reminder on GFS

- Moderate number of Huge Files
- A lot of failures
- Appending $\neq$ random write
- Concurrent write must be supported

Chubby in BigTable

Used for:

Chubby in BigTable

Used for:

- Electing a unique master.

Chubby in BigTable

Used for:

- Electing a unique master.
- Discover tablets.

Chubby in BigTable

Used for:

- Electing a unique master.
- Discover tablets.
- Relocate tablets if needed.

Chubby in BigTable

Used for:

- Electing a unique master.
- Discover tablets.
- Relocate tablets if needed.
- ACL

Chubby in BigTable

Used for:

- Electing a unique master.
- Discover tablets.
- Relocate tablets if needed.
- ACL
- Column family lists.

Some details

- 5 replicas; one will be the Chubby master.

Some details

- 5 replicas; one will be the Chubby master.

Some details

- 5 replicas; one will be the Chubby master.
- Give Locks to clients.

Some details

- 5 replicas; one will be the Chubby master.
- Give Locks to clients.

Some details

- 5 replicas; one will be the Chubby master.
- Give Locks to clients.
- Based on the Paxos algorithm.

Some details

Session

A client will need to maintain a session with the Chubby service.

Some details

Session

A client will need to maintain a session with the Chubby service. KeepAlive packets are sent by the client to the master.

Some details

Session

A client will need to maintain a session with the Chubby service.
KeepAlive packets are sent by the client to the master.
RPC are blocked by the master until the lease is closed to expire.

Some details

Session

A client will need to maintain a session with the Chubby service.
KeepAlive packets are sent by the client to the master.
RPC are blocked by the master until the lease is closed to expire.

- ✗ slow failure detection by the client.

Some details

Session

A client will need to maintain a session with the Chubby service.
KeepAlive packets are sent by the client to the master.
RPC are blocked by the master until the lease is closed to expire.

- ✗ slow failure detection by the client.
- ✓ few redondant packets

Some details

Session

A client will need to maintain a session with the Chubby service.
KeepAlive packets are sent by the client to the master.
RPC are blocked by the master until the lease is closed to expire.

- ✗ slow failure detection by the client.
- ✓ few redondant packets
- ✗ why clients should fail ?

Some details

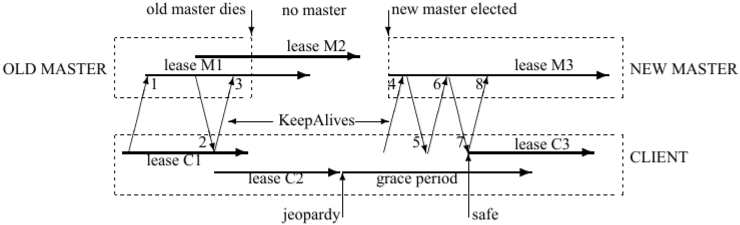
Failovers

If a master fails a new master need to take over the system.

Some details

Failovers

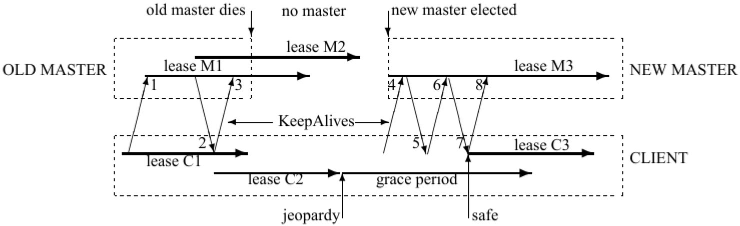
If a master fails a new master need to take over the system.



Some details

Failovers

If a master fails a new master need to take over the system.



✗ Complicated.

Paxos algorithm

Originaly for parliament decree.

Paxos algorithm

Originaly for parliament decree.

Properties wanted

Paxos algorithm

Originaly for parliament decree.

Properties wanted

- At most one decree decided (safety)

Paxos algorithm

Originaly for parliament decree.

Properties wanted

- At most one decree decided (safety)
- At least one decree (liveness)

Properties on ballots

Make liveness possible and ensure consistency (safety property)

Properties on ballots

Make liveness possible and ensure consistency (safety property)

- Each ballot has a unique number.

Properties on ballots

Make liveness possible and ensure consistency (safety property)

- Each ballot has a unique number.
- Every quorum has at least one priest in common.

Properties on ballots

Make liveness possible and ensure consistency (safety property)

- Each ballot has a unique number.
- Every quorum has at least one priest in common.
- For every ballot B, if a priest in the quorum has voted in a earlier ballot then the decree in B is equal to the lastest ballot where the priest voted.

Back to Paxos in Chubby

Consensus

Back to Paxos in Chubby

Consensus

- Several coordinators will propose a value.

Back to Paxos in Chubby

Consensus

- Several coordinators will propose a value.
  It works because:

Back to Paxos in Chubby

Consensus

- Several coordinators will propose a value.
  It works because:
  - Each coordinator will generate an unique number with their propositions. (ex: $s \mod n \equiv id$)

Back to Paxos in Chubby

Consensus

- Several coordinators will propose a value.
  It works because:
  - Each coordinator will generate an unique number with their propositions. (ex: $s \mod n \equiv id$)
  - Promise (ack from replicas that will ignore older coordinator) will include the latest value proposed.

Back to Paxos in Chubby

Consensus

- Several coordinators will propose a value.
  It works because:
  - Each coordinator will generate an unique number with their propositions. (ex: $s \mod n \equiv id$)
  - Promise (ack from replicas that will ignore older coordinator) will include the latest value proposed.

Back to Paxos in Chubby

Consensus

- Several coordinators will propose a value.
  It works because:
  - Each coordinator will generate an unique number with their propositions. (ex: $s \mod n \equiv id$)
  - Promise (ack from replicas that will ignore older coordinator) will include the latest value proposed.
- Since we want to have a consensus on several values, Paxos will be repeated: Need a catch-up mechanism for slow machines.

Properties of Paxos/Chubby

Properties of Paxos/Chubby

✓ Handle disk corruption

Properties of Paxos/Chubby

✓ Handle disk corruption

Properties of Paxos/Chubby

✓ Handle disk corruption by being a non voting member for a whole cycle.

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large

Properties of Paxos/Chubby

✓ Handle disk corruption by being a non voting member for a whole cycle.

✗ Logs can become large

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help
  Still problematic

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help
  Still problematic
- ✗ Implementation difficult

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help
  Still problematic
- ✗ Implementation difficult
- ✗ Read = a Paxos instance

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help
  Still problematic
- ✗ Implementation difficult
- ✗ Read = a Paxos instance

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help
  Still problematic
- ✗ Implementation difficult
- ✗ Read = a Paxos instance → expensive

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help
  Still problematic
- ✗ Implementation difficult
- ✗ Read = a Paxos instance → expensive
  master lease can help.

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help
  Still problematic
- ✗ Implementation difficult
- ✗ Read $=$ a Paxos instance $\rightarrow$ expensive
  master lease can help.
  But problem of unstability if network is unstable.

Properties of Paxos/Chubby

- ✓ Handle disk corruption by being a non voting member for a whole cycle.
- ✗ Logs can become large
  Snapshots can help
  Still problematic
- ✗ Implementation difficult
- ✗ Read = a Paxos instance → expensive
  master lease can help.
  But problem of unstability if network is unstable.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.

The goal of compaction is to reduce the size of this table.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.

The goal of compaction is to reduce the size of this table.

1. minor compaction: after reaching a threshold: freeze and create a new memtable. the old table is transform into a SSTable.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.

The goal of compaction is to reduce the size of this table.

1. minor compaction: after reaching a threshold: freeze and create a new memtable. the old table is transform into a SSTable.
   - ✓ reduce memory usage.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.

The goal of compaction is to reduce the size of this table.

1. minor compaction: after reaching a threshold: freeze and create a new memtable. the old table is transform into a SSTable.
   - ✓ reduce memory usage.
   - ✓ increase recovery speed.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.
The goal of compaction is to reduce the size of this table.

1. minor compaction: after reaching a threshold: freeze and create a new memtable. the old table is transform into a SSTable.
   - ✓ reduce memory usage.
   - ✓ increase recovery speed.
   - ✓ read and write can be done concurrently.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.

The goal of compaction is to reduce the size of this table.

1. minor compaction: after reaching a threshold: freeze and create a new memtable. the old table is transform into a SSTable.
   - ✓ reduce memory usage.
   - ✓ increase recovery speed.
   - ✓ read and write can be done concurrently.
   - ✗ number of SSTables is still increasing.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.

The goal of compaction is to reduce the size of this table.

1. minor compaction: after reaching a threshold: freeze and create a new memtable. the old table is transform into a SSTable.
   - ✓ reduce memory usage.
   - ✓ increase recovery speed.
   - ✓ read and write can be done concurrently.
   - ✗ number of SSTables is still increasing.
2. merging compaction: reduce the number of SSTables.

Compaction or Garbage Collection

Concept

The Memtable (logs recent changes) will become big.

The goal of compaction is to reduce the size of this table.

1. minor compaction: after reaching a threshold: freeze and create a new memtable. the old table is transform into a SSTable.
   - ✓ reduce memory usage.
   - ✓ increase recovery speed.
   - ✓ read and write can be done concurrently.
   - ✗ number of SSTables is still increasing.
2. merging compaction: reduce the number of SSTables.
3. major compaction: merging compaction using all the SSTables.

Locality

Grouping column families into a locality group.

Locality

Grouping column families into a locality group.
Each locality group will have an SSTable.

Locality

Grouping column families into a locality group.
Each locality group will have an SSTable.

✓ Will provide performance improvement if columns that are not accessed together are on seperate SSTables.

Locality

Grouping column families into a locality group.
Each locality group will have an SSTable.

- ✓ Will provide performance improvement if columns that are not accessed together are on seperate SSTables.
- ✗ Actually difficult to know how to do it dynamically.

Compression and caching

- Clients can decide what compresion scheme to use for SSTables (portion of it, different per SSTable).

Compression and caching

- Clients can decide what compresion scheme to use for SSTables (portion of it, different per SSTable).
  Typical compression:

Compression and caching

- Clients can decide what compresion scheme to use for SSTables (portion of it, different per SSTable).
  Typical compression:
  - First alogrithm will look for similarities over a large window.

Compression and caching

- Clients can decide what compresion scheme to use for SSTables (portion of it, different per SSTable).
  Typical compression:
  - First alogrithm will look for similarities over a large window.
  - Second algorithm will look for common string in a small window 16KB.

Compression and caching

- Clients can decide what compresion scheme to use for SSTables (portion of it, different per SSTable).
  Typical compression:
  - First alogrithm will look for similarities over a large window.
  - Second algorithm will look for common string in a small window 16KB.
- Tablet server will use cache to improve latency.

Logging

How would you store the logs about the tablets ?

Logging

How would you store the logs about the tablets ?

- A seperate log for each tablet ?

Logging

How would you store the logs about the tablets ?

- A seperate log for each tablet ?

Logging

How would you store the logs about the tablets ?

- A seperate log for each tablet ? GFS is used for accessing a moderate number of files...
- One huge log ?

Logging

How would you store the logs about the tablets ?

- A seperate log for each tablet ? GFS is used for accessing a moderate number of files...
- One huge log ?

Logging

How would you store the logs about the tablets ?

- A seperate log for each tablet ? GFS is used for accessing a moderate number of files...
- One huge log ? What about failure recovery time ?

Logging

How would you store the logs about the tablets ?

- A seperate log for each tablet ? GFS is used for accessing a moderate number of files...
- One huge log ? What about failure recovery time ?
- Two logs are used (only one is active) and they are sorted using table id, row name and sequence number

Bloom filters

Problem

Read requires acessing all SSTables inside a tablet.

Bloom filters

Problem

Read requires acessing all SSTables inside a tablet.
Stressful for the disk.

Bloom filters

Problem

Read requires acessing all SSTables inside a tablet.
Stressful for the disk.

- A blooming filter can be used to help locate the data when knowing the column and the row.

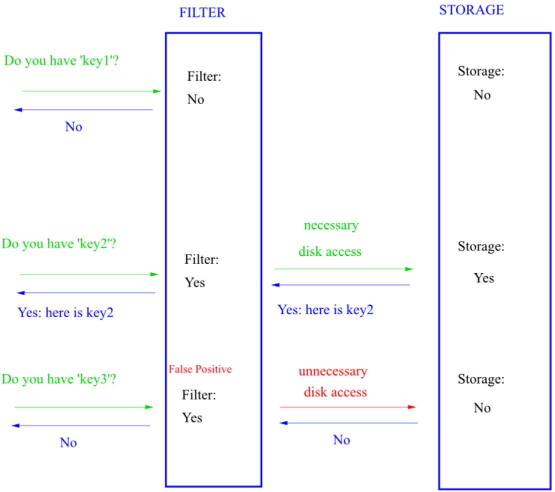
Bloom filters

Problem

Read requires acessing all SSTables inside a tablet.
Stressful for the disk.

- A blooming filter can be used to help locate the data when knowing the column and the row.
- A blooming filter is an improved hashing method.

Bloom filters



Plan

Lore[MAG+97] and [Abi97]

Semistructured database is an old problem.

Lore[MAG+97] and [Abi97]

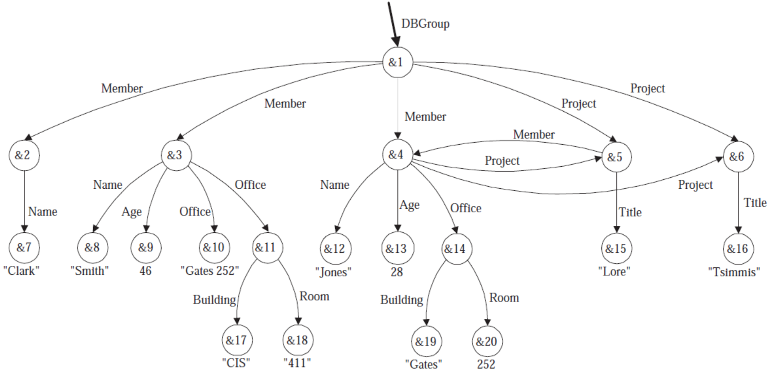
Semistructured database is an old problem.

Data model: Object Exchange Model

Lore[MAG+97] and [Abi97]

Semistructured database is an old problem.

Data model: Object Exchange Model



Lore

Properties about Lore

- Very general: graph with label.

Lore

Properties about Lore

- Very general: graph with label.

Lore

Properties about Lore

- Very general: graph with label. (˜tree)
- Hide irregularities in the structure when doing queries.

Lore

Properties about Lore

- Very general: graph with label. (˜tree)
- Hide irregularities in the structure when doing queries.
- Pattern match possible.

Lore

Properties about Lore

- Very general: graph with label. (~tree)
- Hide irregularities in the structure when doing queries.
- Pattern match possible.
- Merging new data.

Some Comparison

Some Comparison

✓ Query will be optimized. (˜compilation)

Some Comparison

- ✓ Query will be optimized. (˜compilation)
- ✓ More general than Column-key model.

Some Comparison

- ✓ Query will be optimized. (˜compilation)
- ✓ More general than Column-key model.
- ✓ Join support.

Some Comparison

- ✓ Query will be optimized. (˜compilation)
- ✓ More general than Column-key model.
- ✓ Join support.
- ✓ Virtualization of data placement

Some Comparison

- ✓ Query will be optimized. (˜compilation)
- ✓ More general than Column-key model.
- ✓ Join support.
- ✓ Virtualization of data placement

Some Comparison

- ✓ Query will be optimized. (˜compilation)
- ✓ More general than Column-key model.
- ✓ Join support.
- ✓ Virtualization of data placement At the time of the paper
- ✓ Dataguides: visualization of database.

Some Comparison

- ✓ Query will be optimized. (˜compilation)
- ✓ More general than Column-key model.
- ✓ Join support.
- ✓ Virtualization of data placement At the time of the paper
- ✓ Dataguides: visualization of database.
- ✓ Code length (60 000 lines of C++ vs 550 000 for MongoDB)

Some Comparison

- ✓ Query will be optimized. (˜compilation)
- ✓ More general than Column-key model.
- ✓ Join support.
- ✓ Virtualization of data placement At the time of the paper
- ✓ Dataguides: visualization of database.
- ✓ Code length (60 000 lines of C++ vs 550 000 for MongoDB)
- ✗ Performance Issue. (˜traversal of graphs)

Cassandra [LM10]

Very similar to Bigtable (column-oriented).

Cassandra [LM10]

Very similar to Bigtable (column-oriented).

Some features

- Dynamic partionning of data.

Cassandra [LM10]

Very similar to Bigtable (column-oriented).

Some features

- Dynamic partionning of data.
- Consistent hashing for distributing the data.

Cassandra [LM10]

Very similar to Bigtable (column-oriented).

Some features

- Dynamic partionning of data.
- Consistent hashing for distributing the data.
- Replication is done by data replication on N nodes.

Cassandra [LM10]

Very similar to Bigtable (column-oriented).

Some features

- Dynamic partionning of data.
- Consistent hashing for distributing the data.
- Replication is done by data replication on N nodes.
- Global Knowledge of the network (hashing)

Cassandra [LM10]

Very similar to Bigtable (column-oriented).

Some features

- Dynamic partionning of data.
- Consistent hashing for distributing the data.
- Replication is done by data replication on N nodes.
- Global Knowledge of the network (hashing)
- Failure detection.

Cassandra [LM10]

Very similar to Bigtable (column-oriented).

Some features

- Dynamic partionning of data.
- Consistent hashing for distributing the data.
- Replication is done by data replication on N nodes.
- Global Knowledge of the network (hashing)
- Failure detection.
- Efficient anti-entropy gossiping protocol

Consistency

Strong Consistency

```
#Writers + #Readers > NbReplication
```

Consistency

Strong Consistency

`#Writers + #Readers > NbReplication`

Consistency level

Read and write can have differents level of consistency (1 node to respond, majority, all).

Some Comparison

Some Comparison

✓ P2P structure

Some Comparison

- ✓ P2P structure
- ✓ Super Family.

Some Comparison

- ✓ P2P structure
- ✓ Super Family.
- ✓ Load balancing (move lightly loaded nodes in the "ring")

Some Comparison

- ✓ P2P structure
- ✓ Super Family.
- ✓ Load balancing (move lightly loaded nodes in the "ring")
- ✓ Some locality knowledge in replication (rack aware, datacenter aware)

Some Comparison

- ✓ P2P structure
- ✓ Super Family.
- ✓ Load balancing (move lightly loaded nodes in the “ring”)
- ✓ Some locality knowledge in replication (rack aware, datacenter aware)
- ✓ Consistent hashing (reduce cost if changed)

Some Comparison

- ✓ P2P structure
- ✓ Super Family.
- ✓ Load balancing (move lightly loaded nodes in the "ring")
- ✓ Some locality knowledge in replication (rack aware, datacenter aware)
- ✓ Consistent hashing (reduce cost if changed)
- ✗ (Eventual) Consistency

MongoDB (No precise article)

- Scability by sharding

MongoDB (No precise article)

- Scability by sharding
- Document oriented

Amazon Dynamo[DHJ+07]

- Weak Consistency.

Amazon Dynamo[DHJ+07]

- Weak Consistency.
- Consistent hashing.

Amazon Dynamo[DHJ+07]

- Weak Consistency.
- Consistent hashing.
- Object versionning

Amazon Dynamo[DHJ+07]

- Weak Consistency.
- Consistent hashing.
- Object versionning
- Decentralized

Amazon Dynamo[DHJ+07]

- Weak Consistency.
- Consistent hashing.
- Object versionning
- Decentralized
- Replication using quorum

Amazon Dynamo[DHJ+07]

- Weak Consistency.
- Consistent hashing.
- Object versionning
- Decentralized
- Replication using quorum
- Failure detection

Amazon Dynamo[DHJ+07]

- Weak Consistency.
- Consistent hashing.
- Object versionning
- Decentralized
- Replication using quorum
- Failure detection
- Merkle tree for Eventual Consensus (Used in Cassandra)

Some Comparison

Some Comparison

✗ Caching ?

Some Comparison

- ✗ Caching ?
- ✗ Snapshot ?

Some Comparison

- ✗ Caching ?
- ✗ Snapshot ?
- ✗ (original article) key/value schema will affect speed if value are huge (to write into the data you need to read it).

VoltDB `http://voltdb.com`

- Based on H-Store[SMA+07]

VoltDB http://voltdb.com

- Based on H-Store[SMA+07]
- Idea is to use main memory as storage.

RAM databases

✗ Cost ?

- ✗ Cost ?
- ✗ Size ?

- ✗ Cost ?
- ✗ Size ?
- ✗ RAM is not persistent.

Plan

Mapreduce

Using HBase.

Mapreduce

Using HBase.
One map-reduce job per tablets.

Mapreduce

Using HBase.
One map-reduce job per tablets.

Compaction

Size-Tiered

Compaction is done if the number of sstables hits a threshold.

- ✗ Need a lot of space to do the copy (and the size of sstables increases).

Compaction

Size-Tiered

Compaction is done if the number of sstables hits a threshold.

- ✗ Need a lot of space to do the copy (and the size of sstables increases).
- ✗ Columns of a row could be spread across the different sstables

Compaction

Size-Tiered

Compaction is done if the number of sstables hits a threshold.

- ✗ Need a lot of space to do the copy (and the size of sstables increases).
- ✗ Columns of a row could be spread across the different sstables

Compaction

Size-Tiered

Compaction is done if the number of sstables hits a threshold.

- ✗ Need a lot of space to do the copy (and the size of sstables increases).
- ✗ Columns of a row could be spread across the different sstables $\rightarrow$ Irregular performance.

Leveled compaction

SSTable are smaller and grouped by levels.

Compaction

Size-Tiered

Compaction is done if the number of sstables hits a threshold.

- ✗ Need a lot of space to do the copy (and the size of sstables increases).
- ✗ Columns of a row could be spread across the different sstables → Irregular performance.

Leveled compaction

SSTable are smaller and grouped by levels.
Inside a level, sstables will not overlap.

Compaction

Size-Tiered

Compaction is done if the number of sstables hits a threshold.

- ✗ Need a lot of space to do the copy (and the size of sstables increases).
- ✗ Columns of a row could be spread across the different sstables $\rightarrow$ Irregular performance.

Leveled compaction

SSTable are smaller and grouped by levels.
Inside a level, sstables will not overlap.

- ✗ More I/O

Compaction

Size-Tiered

Compaction is done if the number of sstables hits a threshold.

- ✗ Need a lot of space to do the copy (and the size of sstables increases).
- ✗ Columns of a row could be spread across the different sstables → Irregular performance.

Leveled compaction

SSTable are smaller and grouped by levels.
Inside a level, sstables will not overlap.

- ✗ More I/O

Compaction

Size-Tiered

Compaction is done if the number of sstables hits a threshold.

- ✗ Need a lot of space to do the copy (and the size of sstables increases).
- ✗ Columns of a row could be spread across the different sstables → Irregular performance.

Leveled compaction

SSTable are smaller and grouped by levels.
Inside a level, sstables will not overlap.

- ✗ More I/O

How to choose from this two policies ?

SSTable assignment

- The master is capable of maintining the list: $Tl[TS_{id}]$ = all tablets handle by TS_{id}.

SSTable assignment

- The master is capable of maintining the list:
  $Tl[TS_{id}]$ = all tablets handle by TS_{id}.
- Creations, Merges and Deletions are handle by the master.

SSTable assignment

- The master is capable of maintining the list: $Tl[TS_{id}] =$ all tablets handle by TS_{id}.
- Creations, Merges and Deletions are handle by the master.
- Split are initiated by the tablet server but it notifies the master.

✗ No Join

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency
- Problem that arise with semi structured data (texts):

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency
- Problem that arise with semi structured data (texts):

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency
- Problem that arise with semi structured data (texts):
  no schema possible ? (relationnal data. use null value)

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency
- Problem that arise with semi structured data (texts):
  no schema possible ? (relationnal data. use null value)
  dynamic schema
- ✓ Nice for Map reduce.

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency
- Problem that arise with semi structured data (texts):
  no schema possible ? (relationnal data. use null value)
  dynamic schema
- ✓ Nice for Map reduce.
- Load balancing: an way of scaling SQL server.

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency
- Problem that arise with semi structured data (texts):
  no schema possible ? (relationnal data. use null value)
  dynamic schema
- ✓ Nice for Map reduce.
- Load balancing: an way of scaling SQL server.
- ✗ Paxos

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency
- ■ Problem that arise with semi structured data (texts):
  no schema possible ? (relationnal data. use null value)
  dynamic schema
- ✓ Nice for Map reduce.
- ■ Load balancing: an way of scaling SQL server.
- ✗ Paxos
- ✗ Why use a tree for locating tables ?

- ✗ No Join
- ✗ ACID (Atomicity, Consistency, Isolation, Durability): eventual consistency
- ■ Problem that arise with semi structured data (texts):
  no schema possible ? (relationnal data. use null value)
  dynamic schema
- ✓ Nice for Map reduce.
- ■ Load balancing: an way of scaling SQL server.
- ✗ Paxos
- ✗ Why use a tree for locating tables ?
- ✗ No real solution for the number of tablets.

Plan

Conclusion

NoSQL $\neq$ SQL.

Conclusion

NoSQL $\neq$ SQL.

Old problem but new data pattern.

Conclusion

NoSQL $\neq$ SQL.

Old problem but new data pattern.

Conclusion

Thank You for your attention !

Bibliography

Serge Abiteboul.
Querying semi-structured data.
In Foto N. Afrati and Phokion G. Kolaitis, editors, *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, volume 1186 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 1997.

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels.
Dynamo: amazon's highly available key-value store.
SIGOPS Oper. Syst. Rev., 41(6):205–220, October 2007.

Alpa Jain, AnHai Doan, and Luis Gravano.
Optimizing sql queries over text databases.
In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 636–645, Washington, DC, USA, 2008. IEEE Computer Society.

Avinash Lakshman and Prashant Malik.
Cassandra: a decentralized structured storage system.
SIGOPS Oper. Syst. Rev., 44(2):35–40, April 2010.

Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom.
Lore: A database management system for semistructured data.
SIGMOD Record, 26:54–66, 1997.

Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, and Pat Helland.
The end of an architectural era: (it's time for a complete rewrite).
In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 1150–1160. VLDB Endowment, 2007.