CS 245: Database System Principles

Notes 13:
BigTable, HBASE, Cassandra

Hector Garcia-Molina

Sources

Lots of Buzz Words!
• “Apache Cassandra is an open-source, distributed, decentralized, elastically scalable, highly available, fault-tolerant, tunably consistent, column-oriented database that bases its distribution design on Amazon’s dynamo and its data model on Google’s Big Table.”
• Clearly, it is buzz-word compliant!!

Basic Idea: Key-Value Store

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>v1</td>
</tr>
<tr>
<td>k2</td>
<td>v2</td>
</tr>
<tr>
<td>k3</td>
<td>v3</td>
</tr>
<tr>
<td>k4</td>
<td>v4</td>
</tr>
</tbody>
</table>

API:
- lookup(key) → value
- lookup(key range) → values
- getNext → value
- insert(key, value)
- delete(key)

Each row has timestamp
Single row actions atomic (but not persistent in some systems?)
No multi-key transactions
No query language!

Fragmentation (Sharding)

```
server 1
k1 v1
k2 v2
k3 v3
k4 v4
server 2
k5 v5
k6 v6
k7 v7
k8 v8
server 3
k9 v9
k10 v10
```

- use a partition vector
- "auto-sharding": vector selected automatically
Tablet Replication

- Cassandra:
  - Replication Factor (# copies)
  - R/W Rule: One, Quorum, All
  - Policy (e.g., Rack Unaware, Rack Aware, ...)
  - Read all copies (return fastest reply, do repairs if necessary)
- HBase: Does not manage replication, relies on HDFS

Need a “directory”
- Table Name: Key → Server that stores key
  → Backup servers
- Can be implemented as a special table.

Tablet Internals

Design Philosophy (?): Primary scenario is where all data is in memory. Disk storage added as an afterthought.

Column Family

- for storage, treat each row as a single “super value”
- API provides access to sub-values (use family:qualifier to refer to sub-values e.g., price:euros, price:dollars)
- Cassandra allows “super-column”: two level nesting of columns (e.g., Column A can have sub-columns X & Y)
Vertical Partitions

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>a1</td>
<td>b1</td>
<td>c1</td>
<td>d1</td>
<td>e1</td>
</tr>
<tr>
<td>k2</td>
<td>a2</td>
<td>null</td>
<td>c2</td>
<td>d2</td>
<td>e2</td>
</tr>
<tr>
<td>k3</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>d3</td>
<td>e3</td>
</tr>
<tr>
<td>k4</td>
<td>a4</td>
<td>b4</td>
<td>c4</td>
<td>e4</td>
<td>e4</td>
</tr>
<tr>
<td>k5</td>
<td>a5</td>
<td>b5</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

can be manually implemented as

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>a1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k2</td>
<td>a2</td>
<td>null</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k3</td>
<td>null</td>
<td>null</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k4</td>
<td>a4</td>
<td>b4</td>
<td>c4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k5</td>
<td>a5</td>
<td>b5</td>
<td>null</td>
<td>null</td>
<td></td>
</tr>
</tbody>
</table>

Vertical Partitions

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>a1</td>
<td>b1</td>
<td>c1</td>
<td>d1</td>
<td>e1</td>
</tr>
<tr>
<td>k2</td>
<td>a2</td>
<td>null</td>
<td>c2</td>
<td>d2</td>
<td>e2</td>
</tr>
<tr>
<td>k3</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>d3</td>
<td>e3</td>
</tr>
<tr>
<td>k4</td>
<td>a4</td>
<td>b4</td>
<td>c4</td>
<td>e4</td>
<td>e4</td>
</tr>
<tr>
<td>k5</td>
<td>a5</td>
<td>b5</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

Failure Recovery (BigTable, HBase)

ping

memory → tablet server → master node → spare tablet server

write ahead logging

log → GFS or HFS

Failure recovery (Cassandra)

- No master node, all nodes in "cluster" equal

server 1

server 2

server 3

Failure recovery (Cassandra)

- No master node, all nodes in "cluster" equal

access any table in cluster at any server

server 1

server 2

server 3

that server sends requests to other servers

Bonus Slides*: Are Traditional Databases Dead?

- Heard on Twitter:
  - noSQL rules
  - new DB systems scale better than old ones
  - DBMS too slow ...
- Therefore, need new, revolutionary technology!!

* WARNING: Author may be biased :-3
Cautionary Tale

- Lawrence Richard Walters, nicknamed "Lawnchair Larry" or the "Lawn Chair Pilot", (April 19, 1949 – October 6, 1993) was an American truck driver who took flight on July 2, 1982 in a homemade aircraft. Dubbed Inspiration I, the "flying machine" consisted of an ordinary patio chair with 45 helium-filled weather balloons attached to it. Walters rose to an altitude of over 15,000 feet (4,600 m) and floated from his point of origin in San Pedro, California into controlled airspace near Los Angeles International Airport.

Parallels

- **Lawnchair Larry**
  - Wanna fly
  - Can't afford airplane
  - I can do myself!
  - I am off!!

- **T-Gen**
  - Wanna DB services
  - Can't afford real DBMS
  - I can do myself!
  - I am off!!
  - How do I land???
  - How to talk ATC?
  - How to navigate?
  - Need oxygen!!!

Keep Lawnchair Larry in Mind

- Does DBMS technology not cut it and we need to start from scratch??
  - Or are you just being cheap? 😊
- If you think you need a subset of DBMS, will needs change over time?