Petabyte Scale Data at Facebook

Dhruba Borthakur, Engineer at Facebook, UC Berkeley, Nov 2012
<table>
<thead>
<tr>
<th></th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong></td>
<td>Types of Data</td>
</tr>
<tr>
<td><strong>2</strong></td>
<td>Data Model and API for Facebook Graph Data</td>
</tr>
<tr>
<td><strong>3</strong></td>
<td>SLTP (Semi-OLTP) and Analytics data</td>
</tr>
<tr>
<td><strong>4</strong></td>
<td>Why Hive?</td>
</tr>
</tbody>
</table>
Four major types of storage systems

- **Online Transaction Processing Databases (OLTP)**
  - The Facebook Social Graph

- **Semi-online Light Transaction Processing Databases (SLTP)**
  - Facebook Messages and Facebook Time Series

- **Immutable DataStore**
  - Photos, videos, etc

- **Analytics DataStore**
  - Data Warehouse, Logs storage
## Size and Scale of Databases

<table>
<thead>
<tr>
<th></th>
<th>Total Size</th>
<th>Technology</th>
<th>Bottlenecks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook Graph</td>
<td>Single digit petabytes</td>
<td>MySQL and TAO</td>
<td>Random read IOPS</td>
</tr>
<tr>
<td>Facebook Messages and Time Series Data</td>
<td>Tens of petabytes</td>
<td>HBase and HDFS</td>
<td>Write IOPS and storage capacity</td>
</tr>
<tr>
<td>Facebook Photos</td>
<td>High tens of petabytes</td>
<td>Haystack</td>
<td>storage capacity</td>
</tr>
<tr>
<td>Data Warehouse</td>
<td>Hundreds of petabytes</td>
<td>Hive, HDFS and Hadoop</td>
<td>storage capacity</td>
</tr>
</tbody>
</table>
## Characteristics

<table>
<thead>
<tr>
<th></th>
<th>Query Latency</th>
<th>Consistency</th>
<th>Durability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook Graph</td>
<td>&lt; few milliseconds</td>
<td>quickly consistent across data centers</td>
<td>No data loss</td>
</tr>
<tr>
<td>Facebook Messages</td>
<td>&lt; 200 millisec</td>
<td>consistent within a data center</td>
<td>No data loss</td>
</tr>
<tr>
<td>and Time Series Data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facebook Photos</td>
<td>&lt; 250 millisec</td>
<td>immutable</td>
<td>No data loss</td>
</tr>
<tr>
<td>Data Warehouse</td>
<td>&lt; 1 min</td>
<td>not consistent across data centers</td>
<td>No silent data loss</td>
</tr>
</tbody>
</table>
Facebook Graph: Objects and Associations
name: Barack Obama
birthday: 08/04/1961
website: http://... verified: 1
...
Facebook Social Graph: TAO and MySQL

An OLTP workload:

- Uneven read heavy workload
- Huge working set with creation-time locality
- Highly interconnected data
- Constantly evolving
- As consistent as possible
Data model

Content aware data store

- Allows for server-side data processing
- Can exploit creation-time locality
- Graph data model
  - Nodes and Edges: Objects and Associations
- Restricted graph API
Data model

Objects & Associations

- Object -> unique 64 bit ID plus a typed dictionary
  - (id) -> (otype, (key -> value)*)
  - ID 6815841748 -> {'type': page, 'name': "Barack Obama", ...}

- Association -> typed directed edge between 2 IDs
  - (id1, atype, id2) -> (time, (key -> value)*)
  - (8636146, RSVP, 130855887032173) -> (1327719600, {'response': 'YES'})

- Association lists
  - (id1, atype) -> all assocs with given id1, atype in desc order by time
Data model

API

- Object : (id) -> (otype, (key -> value)*)
  - obj_add(otype, (k->v)*): creates new object, returns its id
  - obj_update(id, (k->v)*): updates some or all fields
  - obj_delete(id): removes the object permanently
  - obj_get(id): returns type and fields of a given object if exists
Data model

API

- Association : (id₁, atype, id₂) -> (time, (key -> value)* )
  - assoc_add(id₁, atype, id₂, time, (k->v)* ) : adds/updates the given assoc
  - assoc_delete(id₁, atype, id₂) : deletes the given association
Data model

API

- Association : (id₁, atype, id₂) -> (time, (key -> value)*)
  - assoc_get(id₁, atype, id₂set): returns assocs where id₂ ∈ id₂set
  - assoc_range(id₁, atype, offset, limit, filters*): get relevant matching assocs from the given assoc list
  - assoc_count(id₁, atype): returns size of given assoc list
Architecture

Cache & Storage

Web servers

<table>
<thead>
<tr>
<th>TAO Storage Cache</th>
</tr>
</thead>
</table>

MySQL Storage
Sharding

- Object ids and Assoc ids are mapped to shard ids
Workload

- Read-heavy workload
  - Significant range queries
- LinkBench benchmark being open-sourced
  - [http://www.github.com/facebook/linkbench](http://www.github.com/facebook/linkbench)
- Real data distribution of Assocs and their access patterns
Messages & Time Series Database
SLTP workload
Facebook Messages

- Messages
- Chats
- Emails
- SMS
Why we chose HBase

- High write throughput
- Horizontal scalability
- Automatic Failover
- Strong consistency within a data center
- Benefits of HDFS: Fault tolerant, scalable, Map-Reduce toolset,

Why is this SLTP?

- Semi-online: Queries run even if part of the database is offline
- Light Transactions: single row transactions
- Storage capacity bound rather than iops or cpu bound
What we store in HBase

- Small messages
- Message metadata (thread/message indices)
- Search index
- Large attachments stored in Haystack (photo store)
Size and scale of Messages Database

- 6 Billion messages/day
- 74 Billion operations/day
- At peak: 1.5 million operations/sec
- 55% read, 45% write operations
- Average write operation inserts 16 records
- All data is lzo compressed
- Growing at 8 TB/day
Haystack: The Photo Store
## Facebook Photo DataStore

<table>
<thead>
<tr>
<th></th>
<th>2009</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total Size</strong></td>
<td>15 billion photos 1.5 Petabyte</td>
<td>High tens of petabytes</td>
</tr>
<tr>
<td><strong>Upload Rate</strong></td>
<td>30 million photos/day 3 TB/day</td>
<td>300 million photos/day 30 TB/day</td>
</tr>
<tr>
<td><strong>Serving Rate</strong></td>
<td>555K images/sec</td>
<td></td>
</tr>
</tbody>
</table>
Haystack based Design

- Haystack Directory
- Web Server
- Haystack Store
- Haystack Cache
- Browser
- CDN
Haystack Internals

- Log structured, append-only object store
- Built on commodity hardware
- Application-aware replication
- Images stored in 100 GB xfs-files called needles
- An in-memory index for each needle file
  - 32 bytes of index per photo
Hive Analytics Warehouse
Life of a photo tag in Hadoop/Hive storage

Periodic Analysis (HIVE)
- Daily report on count of photo tags by country (1 day)

Adhoc Analysis (HIVE)
- Count photos tagged by females age 20-25 yesterday

Hive Warehouse
- Log line reaches warehouse (15 min)

Scrapes
- User info reaches Warehouse (1 day)

MySQL DB
- User tags a photo

Scribe Log Storage (HDFS)
- Log line reaches Scribeh (10s)
- Log line generated: <user_id, photo_id>

Realtime Analytics (HBASE)
- Count users tagging photos in the last hour (1 min)

copier/loader
- puma

nocron
- nocron

RealHme AnalyHcs (HBASE)
- puma
## Analytics Data Growth (last 4 years)

<table>
<thead>
<tr>
<th>Growth</th>
<th>Facebook Users</th>
<th>Queries/Day</th>
<th>Scribe Data/Day</th>
<th>Nodes in warehouse</th>
<th>Size (Total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>14X</td>
<td>60X</td>
<td>250X</td>
<td>260X</td>
<td>2500X</td>
<td></td>
</tr>
</tbody>
</table>
Why use Hive instead of a Parallel DBMS?

- Stonebraker/DeWitt from the DBMS community:
  - Quote “major step backwards”
  - Published benchmark results which show that Hive is not as performant as a traditional DBMS

What is BigData? Prospecting for Gold..

- “Finding Gold in the wild-west”
- A platform for huge data-experiments
- A majority of queries are searching for a single gold nugget
- Great advantage in keeping all data in one queryable system
- No structure to data, specify structure at query time
How to measure performance

- Traditional database systems:
  - Latency of queries

- Big Data systems:
  - How much data can we store and query? (the ‘Big’ in BigData)
  - How much data can we query in parallel?
  - What is the value of this system?
Measure Cost of Storage

- Distributed Network Encoding of data
  - Encoding is better than replication
  - Use algorithms that minimize network transfer for data repair
- Tradeoff cpu for storage & network
  - Remember lineage of data, e.g. record query that created it
  - If data is not accessed for sometime, delete it
  - If a query occurs, recompute the data using query lineage
Measure Network Encoding

**Start the same:** triplicate every data block

**Background encoding**

- Combine third replica of blocks from a single file to create parity block
- Remove third replica
- Reed Solomon encoding for much older files

A file with three blocks A, B and C (XOR Encoding)

Measuring Data Discovery: Crowd Sourcing

- There are 50K tables in a single warehouse
- Users are Data Administrators themselves
- Questions about a table are directed to users of that table
- Automatic query lineage tools
Measuring Testability

- Traditional systems
  - Recreate load using tests
  - Validate results

- Big Data Systems
  - Cannot replicate production load on test environment
  - Deploy new service on a small percentage of service
    - Monitor metrics
    - Rolling upgrades
  - Gradually deploy to larger section of service
Fault Tolerance and Elasticity

- Commodity machines
- Faults are the norm
- Anomalous behavior rather than complete failures
  - 10% of machines are always 50% slower than the others
Measuring Fault Tolerance and Elasticity

- Fault tolerance is a must
  - Continuously kill machines during benchmarking
  - Slow down 10% of machine during benchmark

- Elasticity is necessary
  - Add/remove new machines during benchmarking
Measuring Value of the System

- cost /GB decreasing with time
  - So users can store more data
  - But users need a metric to determine whether this cost is worth it
- What is the VALUE of this system?
  - A metric that aids the user (and not the service provider)
Value per Byte (VB) for the System

- A new metric named VB
  - Compare differences in value over time
  - If VB increases with time, then user is satisfied
- You touch a byte, its VB is MAX (say 100)
- System VB = weighted sum of VB of each byte in the system
VB – Even a turtle ages with time

- The VB decreases with time
  - A more recent access has more value than an older access
  - Different ageing models (linear, exponential)
Why use Hive instead of a Parallel DBMS?

Stonebraker/DeWitt from the DBMS community:

- Quote “Hadoop is a major step backwards”
- Published benchmark results which show that Hadoop/Hive is not as performant as a traditional DBMS
- Hive query is 50 times slower than DBMS query

Stonebraker’s Conclusion: Facebook’s 4000 node cluster (100PB) can be replaced by a 20 node DBMS cluster

What is wrong with the above conclusion?
Hive/Hadoop instead of Parallel DBMS

- Dr Stonebraker’s proposal would put 5 PB per node on DBMS
  - What will be the io throughput of that system? **Abysmal**
  - How many concurrent queries can it support? **Certainly not 100K concurrent clients**
  - He is using a wrong metric to make a conclusion
- Hive/Hadoop is very very slow
  - Hive/Hadoop needs to be fixed to reduce query latency
  - But an existing DBMS cannot replace Hive/Hadoop
Future Challenges
New trends in storage software

- **Trends:**
  - SSDs cheaper, increasing number of CPUs per server
  - SATA disk capacities reaching 4 - 8 TB per disk, falling prices $/GB

- **New projects**
  - Evaluate OLTP databases that scales linearly with the number of cps
  - Prototype storing cold photos on spin-down disks
Questions?

dhruba@gmail.com

http://hadoopblog.blogspot.com/