Hadoop and Hive Development at Facebook

Dhruba Borthakur, Zheng Shao
{dhruba, zshao}@facebook.com
Presented at Hadoop World, New York
October 2, 2009
Joe Pasqua wanted a simple digital watch. Ended up ordering a Casio with a ton of bells and whistles. I have no resistance to gadgetry.

Pallavi Tekriwal had dinner at that spice...

Vishu Gupta this is more interesting than i thought

Michelle Bostock ad hoc in Yountville, delish

Tridisha Goswami for all d frndz..that make life more colourful,"HAPPY FRIENDSHIP' DAY"
Who generates this data?

- Lots of data is generated on Facebook
  - 300+ million active users
  - 30 million users update their statuses at least once each day
  - More than 1 billion photos uploaded each month
  - More than 10 million videos uploaded each month
  - More than 1 billion pieces of content (web links, news stories, blog posts, notes, photos, etc.) shared each week
Data Usage

- **Statistics per day:**
  - 4 TB of compressed new data added per day
  - 135TB of compressed data scanned per day
  - 7500+ Hive jobs on production cluster per day
  - 80K compute hours per day

- **Barrier to entry is significantly reduced:**
  - New engineers go though a Hive training session
  - ~200 people/month run jobs on Hadoop/Hive
  - Analysts (non-engineers) use Hadoop through Hive
Where is this data stored?

- **Hadoop/Hive Warehouse**
  - 4800 cores, 5.5 PetaBytes
  - 12 TB per node
  - Two level network topology
    - 1 Gbit/sec from node to rack switch
    - 4 Gbit/sec to top level rack switch
Data Flow into Hadoop Cloud

- Web Servers
- Scribe MidTier
- Hadoop Hive Warehouse
- Oracle RAC
- MySQL
- Network Storage and Servers
Hadoop Scribe: Avoid Costly Filers

Web Servers → Scribe MidTier → Realtime Hadoop Cluster

Oracle RAC ↔ Hadoop Hive Warehouse

MySQL

HDFS Raid

- 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
  - Apache JIRA HDFS-503
- DiskReduce from CMU
  - Garth Gibson research

A file with three blocks A, B and C

Archival: Move old data to cheap storage

http://issues.apache.org/jira/browse/HDFS-220
Dynamic-size MapReduce Clusters

- Why multiple compute clouds in Facebook?
  - Users unaware of resources needed by job
  - Absence of flexible Job Isolation techniques
  - Provide adequate SLAs for jobs

- Dynamically move nodes between clusters
  - Based on load and configured policies
  - Apache Jira MAPREDUCE-1044
Resource Aware Scheduling (Fair Share Scheduler)

- We use the Hadoop Fair Share Scheduler
  - Scheduler unaware of memory needed by job
- Memory and CPU aware scheduling
  - RealTime gathering of CPU and memory usage
  - Scheduler analyzes memory consumption in realtime
  - Scheduler fair-shares memory usage among jobs
  - Slot-less scheduling of tasks (in future)
  - Apache Jira MAPREDUCE-961
Hive - Data Warehouse

- Efficient SQL to Map-Reduce Compiler
- Mar 2008: Started at Facebook
- May 2009: Release 0.3.0 available
- Now: Preparing for release 0.4.0

- Countable for 95%+ of Hadoop jobs @ Facebook
- Used by ~200 engineers and business analysts at Facebook every month
Hive DDL

- **DDL**
  - Complex columns
  - Partitions
  - Buckets

- **Example**
  - CREATE TABLE sales (
    id INT,
    items ARRAY<STRUCT{id:INT, name:STRING}>,
    extra MAP<STRING, STRING>
  ) PARTITIONED BY (ds STRING)
  CLUSTERED BY (id) INTO 32 BUCKETS;
Hive Query Language

- SQL
  - Where
  - Group By
  - Equi-Join
  - Sub query in from clause

- Example
  - SELECT r.*, s.*
    FROM r JOIN (  
      SELECT key, count(1) as count  
      FROM s  
      GROUP BY key) s  
    ON r.key = s.key  
  WHERE s.count > 100;
Group By

- 4 different plans based on:
  - Does data have skew?
  - partial aggregation
- Map-side hash aggregation
  - In-memory hash table in mapper to do partial aggregations
- 2-map-reduce aggregation
  - For distinct queries with skew and large cardinality
Join

- Normal map-reduce Join
  - Mapper sends all rows with the same key to a single reducer
  - Reducer does the join

- Map-side Join
  - Mapper loads the whole small table and a portion of big table
  - Mapper does the join
  - Much faster than map-reduce join
**Sampling**

- **Efficient sampling**
  - Table can be bucketed
  - Each bucket is a file
  - Sampling can choose some buckets

- **Example**
  - `SELECT product_id, sum(price) 
    FROM sales TABLESAMPLE (BUCKET 1 OUT OF 32) 
    GROUP BY product_id`
FROM users

INSERT INTO TABLE pv_gender_sum
    SELECT gender, count(DISTINCT userid)
    GROUP BY gender

INSERT INTO DIRECTORY '/user/facebook/tmp/pv_age_sum.dir'
    SELECT age, count(DISTINCT userid)
    GROUP BY age

INSERT INTO LOCAL DIRECTORY '/home/me/pv_age_sum.dir'
    SELECT country, gender, count(DISTINCT userid)
    GROUP BY country, gender;
File Formats

- **TextFile:**
  - Easy for other applications to write/read
  - Gzip text files are not splittable

- **SequenceFile:**
  - Only hadoop can read it
  - Support splittable compression

- **RCFile: Block-based columnar storage**
  - Use SequenceFile block format
  - Columnar storage inside a block
  - 25% smaller compressed size
  - On-par or better query performance depending on the query
SerDe

- Serialization/Deserialization
- Row Format
  - CSV (LazySimpleSerDe)
  - Thrift (ThriftSerDe)
  - Regex (RegexSerDe)
  - Hive Binary Format (LazyBinarySerDe)
- LazySimpleSerDe and LazyBinarySerDe
  - Deserialize the field when needed
  - Reuse objects across different rows
  - Text and Binary format
**Features:**
- Use either Java or Hadoop Objects (int, Integer, IntWritable)
- Overloading
- Variable-length arguments
- Partial aggregation for UDAF

**Example UDF:**
```java
public class UDFExampleAdd extends UDF {
    public int evaluate(int a, int b) {
        return a + b;
    }
}
```
### Hive - Performance

<table>
<thead>
<tr>
<th>Date</th>
<th>SVN Revision</th>
<th>Major Changes</th>
<th>Query A</th>
<th>Query B</th>
<th>Query C</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/22/2009</td>
<td>746906</td>
<td>Before Lazy Deserialization</td>
<td>83 sec</td>
<td>98 sec</td>
<td>183 sec</td>
</tr>
<tr>
<td>2/23/2009</td>
<td>747293</td>
<td>Lazy Deserialization</td>
<td>40 sec</td>
<td>66 sec</td>
<td>185 sec</td>
</tr>
<tr>
<td>3/6/2009</td>
<td>751166</td>
<td>Map-side Aggregation</td>
<td>22 sec</td>
<td>67 sec</td>
<td>182 sec</td>
</tr>
<tr>
<td>4/29/2009</td>
<td>770074</td>
<td>Object Reuse</td>
<td>21 sec</td>
<td>49 sec</td>
<td>130 sec</td>
</tr>
<tr>
<td>8/5/2009</td>
<td>801497</td>
<td>Lazy Binary Format *</td>
<td>21 sec</td>
<td>48 sec</td>
<td>132 sec</td>
</tr>
</tbody>
</table>

- Query A: `SELECT count(1) FROM t;`
- Query B: `SELECT concat(concat(concat(a,b),c),d) FROM t;`
- Query C: `SELECT * FROM t;`
- Map-side time only (incl. GzipCodec for comp/decompression)
- * These two features need to be tested with other queries.
Hive - Future Works

- Indexes
- Create table as select
- Views / variables
- Explode operator
- In/Exists sub queries
- Leverage sort/bucket information in Join