Reliable data propagation between SQL and NoSQL databases using Aesopop

Regunath B
I’ll talk about …

Aesop change data capture and propagation

• Evolution of Flipkart data stores
• Polyglot persistence, tiered data stores
• Data consistency challenges
• Aesop Change data propagation design
• Aesop features
Flipkart in recent times

- Leading eCommerce player in India
- 10M page visits, 2M shipments a day
- 30 million products across more than 70 categories
- Big Billion Days ($300M sales, top ranked app on Google Play Store)
- Ping - social collab in eCommerce
- Progressive web app - native like experience on browser (debuted at #chromedevsummit 2015)
Evolution of Flipkart online data stores
Standard RDBMS scaling

and meanwhile… traded Consistency for Availability

Source: http://www.slideshare.net/slashn/slash-n-tech-talk-track-2-website-architecturermistakes-learnings-siddhartha-reddy
Scaling online data stores (polyglot persistence)

<table>
<thead>
<tr>
<th>Data</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>User session</td>
<td>Memcached, HBase</td>
</tr>
<tr>
<td>Product</td>
<td>HBase, Elastic Search, Redis</td>
</tr>
<tr>
<td>Cart</td>
<td>MySQL, MongoDB</td>
</tr>
<tr>
<td>Notifications</td>
<td>HBase</td>
</tr>
<tr>
<td>Orders</td>
<td>MySQL</td>
</tr>
<tr>
<td>Search</td>
<td>Solr, Neo4j</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Hadoop MR, Redis/ Aerospike</td>
</tr>
<tr>
<td>Promotions</td>
<td>HBase, Aerospike</td>
</tr>
</tbody>
</table>

- Flash sales
- 50K units in 5 seconds
- Product Search cluster is 1000 nodes
- Catalog Service
- 3 M catalog views/sec (200 Gbps)
- 3 TB data, 2B documents

Pattern: Serving Layer + Source of truth data store
Scaling payments
tiered data store(s)
<table>
<thead>
<tr>
<th>Table</th>
<th>Columns</th>
<th>Source</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransactionMaster</td>
<td>txn_id, merchant_id, amount, type</td>
<td>Payments DB (online, live TXNs)</td>
<td>MySQL</td>
</tr>
<tr>
<td>TransactionSummary</td>
<td>txn_id, status, message</td>
<td>Payments Denormalised DB (Refunds, Consoles)</td>
<td>MySQL</td>
</tr>
<tr>
<td>TransactionSaleDetail</td>
<td>txn_id, payment_method, bank_name</td>
<td>Archive DB (Regulatory Req.)</td>
<td>HBase</td>
</tr>
<tr>
<td>TransactionDetails</td>
<td>txn_id, merchant_id, amount, type, status, bank_name</td>
<td>Analytics &amp; Reporting DB (Reports, Queries)</td>
<td>HBase</td>
</tr>
</tbody>
</table>
Data consistency in polyglot & tiered persistence
Caching/Serving Layer challenges

There are only two hard things in Computer Science: cache invalidation and naming things.
-- Phil Karlton

- (Low Cache TTLs + Lazy caching + High Request concurrency) results in:
  - Thundering herds
  - Exposes Availability of primary data store
- (Cache size + distribution + no. of replicas) results in:
  - Hard to implement write-through mechanisms to maintain consistency

Serving Layer is Eventually Consistent, at best
Eventual Consistency

- Replicas converge over time
- Pros
  - Scale reads through multiple replicas
  - Higher overall data availability
- Cons
  - Reads return live data before convergence
  - Achieving Eventual Consistency is not easy
  - Trivially requires Atleast-Once delivery guarantee of updates to all replicas
Aesop : Change data capture, propagation
Need for updates & its challenges

- Multiple applications
- Heterogenous data stores
- Reliable delivery
- Varying application speeds
Producer, Relay, Consumer

Producers

MySQL

Txn Logs

HBase

WAL edits

Relay (Data store agnostic m-mapped file containing data update events - Avro serialized)

Consumers

Subscriber

Subscriber

Subscriber

Subscriber

Subscriber

Transform & Map

Transform & Map

Transform & Map

Transform & Map

Transform & Map
Log Mining
(Old wine in new bottle?)


• “The contents of the DB are a cache of the latest records in the log. The truth is the log. The database is a cache of a subset of the log.” - Jay Kreps (creator of Kafka, 2015)

• WAL (write ahead log) ensures:
  • Each modification is flushed to disk
  • Log records are in order
Reliable delivery

![Diagram of a distributed system with nodes labeled as Transaction Logs, Data store, Producer, Tailer (Slave/Replica), Event Ring, Buffer, Relay, Subscriber, and Application Data stores. The diagram illustrates the flow of transaction logs and the roles of different components in ensuring reliable delivery.]
Reliable delivery

• Store application markers in persistent storage
• Recover application from stored markers
Event consumption

- Data transformation (Data Layer): Multiple destinations
  - MySQL
  - Elastic Search
  - HBase
  - Kafka

- Clustering

```json
MAPPER_CONFIG
{
  or_test:
  {
    "Person":{
      "destinationNamespace":"or_test",
      "destinationEntity":"PERSONPARALLEL",
      "columnMap": {
        "id":"pid",
        "firstName":"FirstName",
        "lastName":"lastname",
        "birthDate":"birthdate",
        "deleted":"deleted"
      }
    },
    "primaryKeyList": ["txn_id"],
    "groupNo": "1"
  },
  {
    "mapAll":true,
    "groupNo": "1"
  }
}
```
• Sequential Consistency (updates applied in the same order, sequentially):
  • Says nothing about time - there is no reference to the “most recent” write operation
  • Mechanisms for Sequential Consistency: Primary-based replication protocols (used by Aesop)
Summary

Aesop - pub-sub like Change Capture, propagation system

What it is

• Supports multiple data stores
• Delivers updates reliably - at least once, in-order
• Supports varying consumer speeds

Performance

• Relay: 1 XL VM (8 core, 32GB)
• Consumers: 4 XL VM, 200 partitions
• Throughput: 30K Inserts per sec (MySQL to HBase)
• Data size: 800 GB

What it isn't

• Not exactly-once delivery
• Not a storage system
• No global ordering across different data-stores (consumers)
More details..

Project

• Open Source: https://github.com/Flipkart/aesop
• Support: aesop-users@googlegroups.com
• Multiple production deployments at Flipkart

Related Work

• LinkedIn Databus
• [2] Facebook Wormhole

References