On Brewing Fresh Espresso: LinkedIn’s Distributed Data Serving Platform

Thomas Marshall
Motivation

- Better performance and horizontal scalability than traditional RDBMS.
- Better consistency, transactions, and schema support than NoSQL.
- Integration into LinkedIn’s data ecosystem.
Data Model

- Nested entities and independent entities.
- Relational
  - Documents - the equivalent of rows
- Hierarchical
  - Document groups - share same partitioning key, span tables, largest unit of transactions
Secondary Indexes

- Allow for efficient lookup based on values other than the primary key.
- Local secondary indexes - apply to one document group.
- Global secondary indexes - apply across doc groups, implemented as derived tables.
Secondary Indexes

- **Lucene**
  - Inverted index.
  - Log structured.
- **Prefix**
  - Inverted index, prefixed by the partition key.
**Architecture**

- **Client** - submit requests via REST API.
- **Router** - send request to appropriate node based on partitioning protocol.
Architecture

- Helix
  - Cluster management system
  - Assigns partitions
Architecture

- Fault tolerance
  - When a master partition fails, a slave is promoted by Helix.
  - Zookeeper heartbeat and performance metrics determine failure.
Overpartitioning

- Shard data into many more partitions than there are nodes.
- Eases failover/cluster expansion.
Architecture

- **Storage node**
  - Stores partitions.
  - Performs queries.
  - Maintains log.
  - Performs background tasks.
Architecture

- **Databus**
  - Achieves replication via pub/sub
  - Ensures timeline consistency
  - Replicated for fault tolerance
Future Work

- Transactions across document groups.
- OLAP workloads.
- Multiple data center deployment.
Conclusion

- Espresso attempts to find a nice medium between traditional RDBMS and NoSQL.
- LinkedIn particularly emphasized operability - ease of schema changes, horizontal scalability, etc.