AMBRY: LINKEDIN'S SCALABLE GEO-DISTRIBUTED OBJECT STORE

PRESENTED BY SANKHA
AMBRY: LINKEDIN’S SCALABLE GEO-DISTRIBUTED OBJECT STORE

BACKGROUND

- In-house system known as Media Server used previously
- It used NFS (for files), Oracle DB (for metadata)
- Not horizontally scalable, faced CPU & I/O issues
AMBRY: LINKEDIN’S SCALABLE GEO-DISTRIBUTED OBJECT STORE

GOALS

- Low latency, high throughput
- Geo-distributed operation
- Scalability
- Load balancing
WHY NOT EXISTING SOLUTIONS?

- Primary use case: distributed key-value store for blobs
- Existing distributed filesystems have too much overhead for metadata and support for all kinds I/O operations
- Key value stores are not optimized for blobs, ie. support zero-copy reads, streaming, etc.
ARCHITECTURE
**SYSTEM OVERVIEW**

- Partition: append only log in pre-allocated large file
- API: 3 operations - put, get and delete
- Load balancing with a re-balancing algorithm
CLUSTER MANAGER

- Kept in sync with Zookeeper
- Hardware Layout: Map of DCs, datanodes, disks and status
- Logical Layout: Map of partitions to the state & placement
ROUTER LIBRARY

- Policy based routing
- Chunking
- Zero cost failure detection
- Proxy requests
DATANODE LAYER

- In-memory indexing
- Exploiting OS cache
- Bloom-filters to faster access to older index segments
REPLICATION

- Find missing blob ids since the last synchronization point
- Request missing blobs and append them to the replica
EVALUATION

(a) Without rebalancing

(b) With rebalancing

(a) Throughput

(b) Latency normalized by blob size
EVALUATION

(a) Throughput (MB/s)

(b) Throughput (requests/s)

(c) Latency
COOL STUFF

- Multi-master system
- Streaming & zero-copy reads
- Zero cost health checks
- Remove consistency issues by generating ID inside Ambry