Serving Large-scale Batch Computed Data with Project Voldemort

Roshan Sumbaly, Jay Kreps, Lei Gao,
Alex Feinberg, Chinmay Sonam, Sam Shah

Challenges of Serving Data-Derived Features

- Requires an almost complete refresh of the data
- **Problem 1:** Serving latency degradation during bulk load
  - Due to updates on serving index
- **Problem 2:** Potential long time in error state
  - Error in algorithm = Bulk load bad data = Bad state till next load

Motivation

Serving system with fast bulk loads and minimal read latency penalty

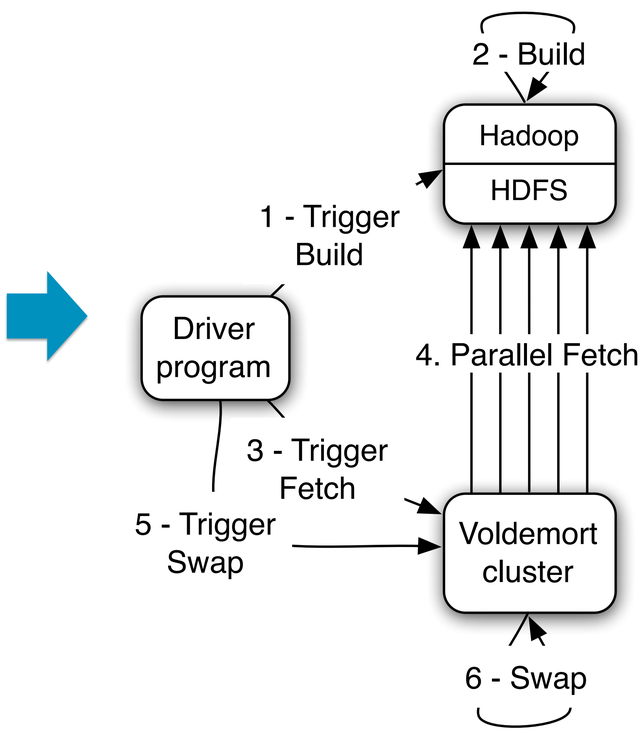
Voldemort

- Distributed key/value system
- Pluggable storage layer
- A cluster has multiple *stores* (~ tables)

Hadoop / HDFS

- Suitable for batch algorithms
- De-facto for storing large logs

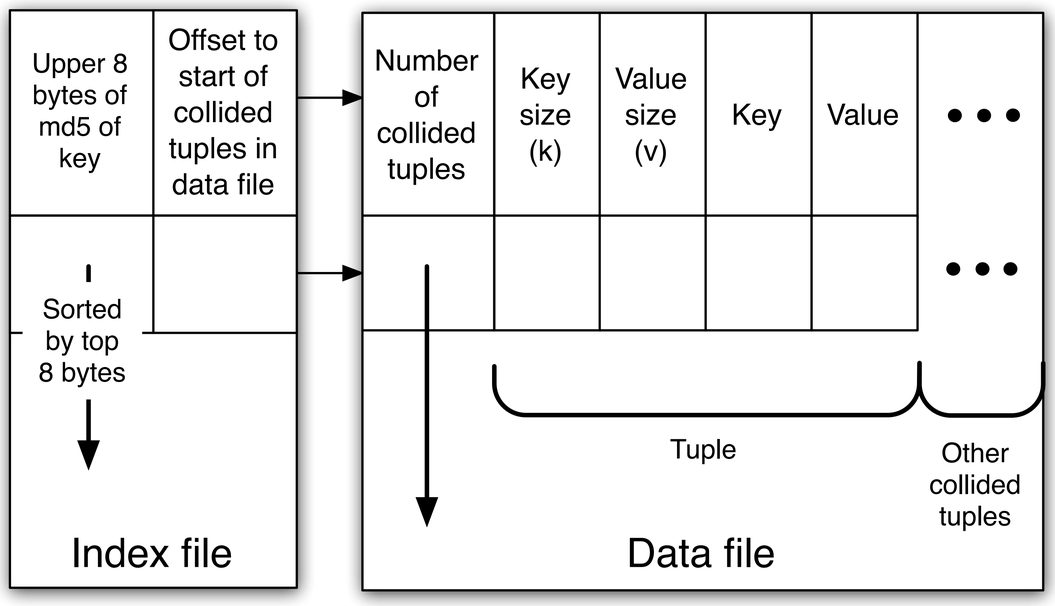
Our solution



- Custom storage engine leveraging Hadoop
- Building store offline solves online performance problem
- Store kept on HDFS in directories per Voldemort node

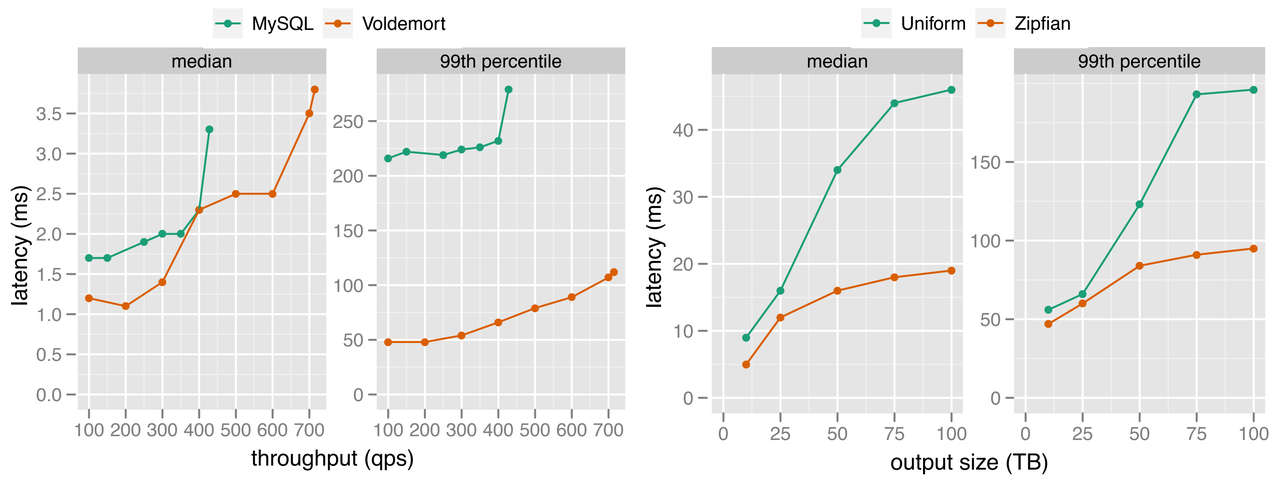
- Voldemort nodes pulls store data in parallel
- Every store saved into multiple versioned directories
- One version of every store is serving, rest for rollback
- Swap = close old store files, open new ones
- Rollback = close new store files, open old ones

Chunk set



- A store consists of multiple chunk sets (data + index file)
- Increase parallelism by increasing reducers
- Swap = memory map index files of all chunk sets
- Key search = binary search in index + jump in data file
- Subset of key of key in index = Increase cache locality

Results



Single node (24 GB RAM) latency for 100 GB data

- Scales to twice the throughput of MySQL while maintaining latency

32 node latency for varying data size

- Latency grows linearly with data size

- Running at LinkedIn for the past 2 years
- Pushing ~ 4 TB of new data to production daily
- Open source (Apache License) – http://project-voldemort.com