Cassandra Offline Analytics

Dongqian Liu, Yi Liu

2017/05/02

Agenda

- Introduction
- Use Case
- Problem & Solution
- Suitable User Scenario
- Cassandra Internals
- Implementation Details
- Performance
- Similar Projects
- Quick Start
- Q&A

Cassandra

- Cassandra is low latency, high throughput big table. It’s suitable for real-time query.
- Good read/write performance
- Bad performance for some operations like scan, count, groupBy, etc.

Our Use Case

- Our team supports Feeds system for eBay Paid Internet Marketing
- We use Cassandra to build our live item snapshots and other use cases
- We need analytics on the data to better boost our business

- **Cluster**
  - 30 nodes
  - r/w throughput: avg. 40k/s, peak 100k/s
  - Data size: 8~10T

Problem

- Operations like full scan take long time. Ex. simple groupBy and count on a single table takes 10+ hrs
- Cause cluster overload
- Pollute in-memory cache that makes read request performance much worse

Solution

- Cassandra internal structure is similar with HBase from high level.
- We can do similar thing as what HBase MapReduce does. There are two gaps:
  - Cassandra sstables are on local disks of cluster nodes, the HBase HFiles are on HDFS.
  - There is no good way to read raw SSTables in MapReduce or Spark job efficiently.

- To fill in above gaps, we can upload the sstables to HDFS in parallel and write some code to let MR and Spark job be able to read the raw SSTables efficiently and finally transform it to Hadoop Table.

Suitable User Scenario

- Require offline analytics on big table, besides low latency and high throughput read/write performance.
- Find HBase can not satisfy the requirements of random read/write performance.

High level Overview

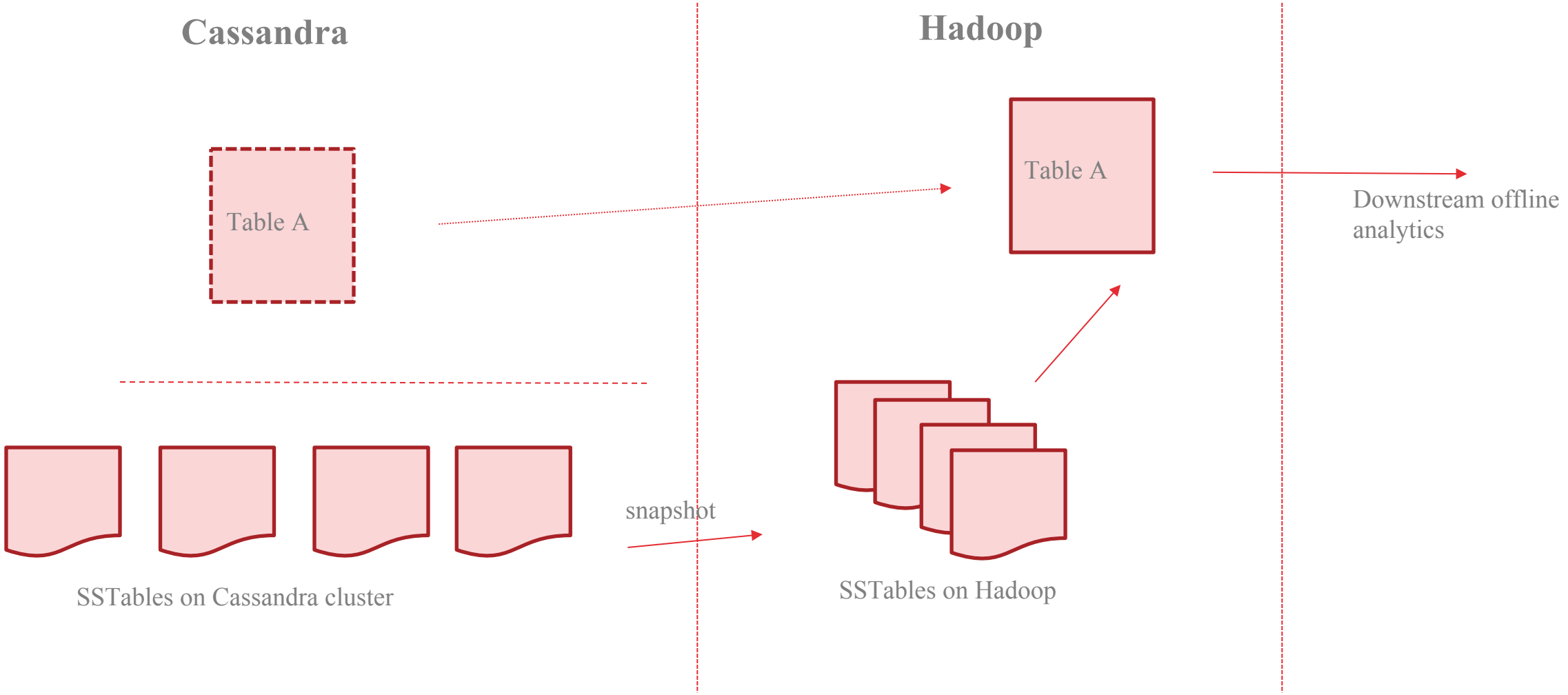


Table Transformation

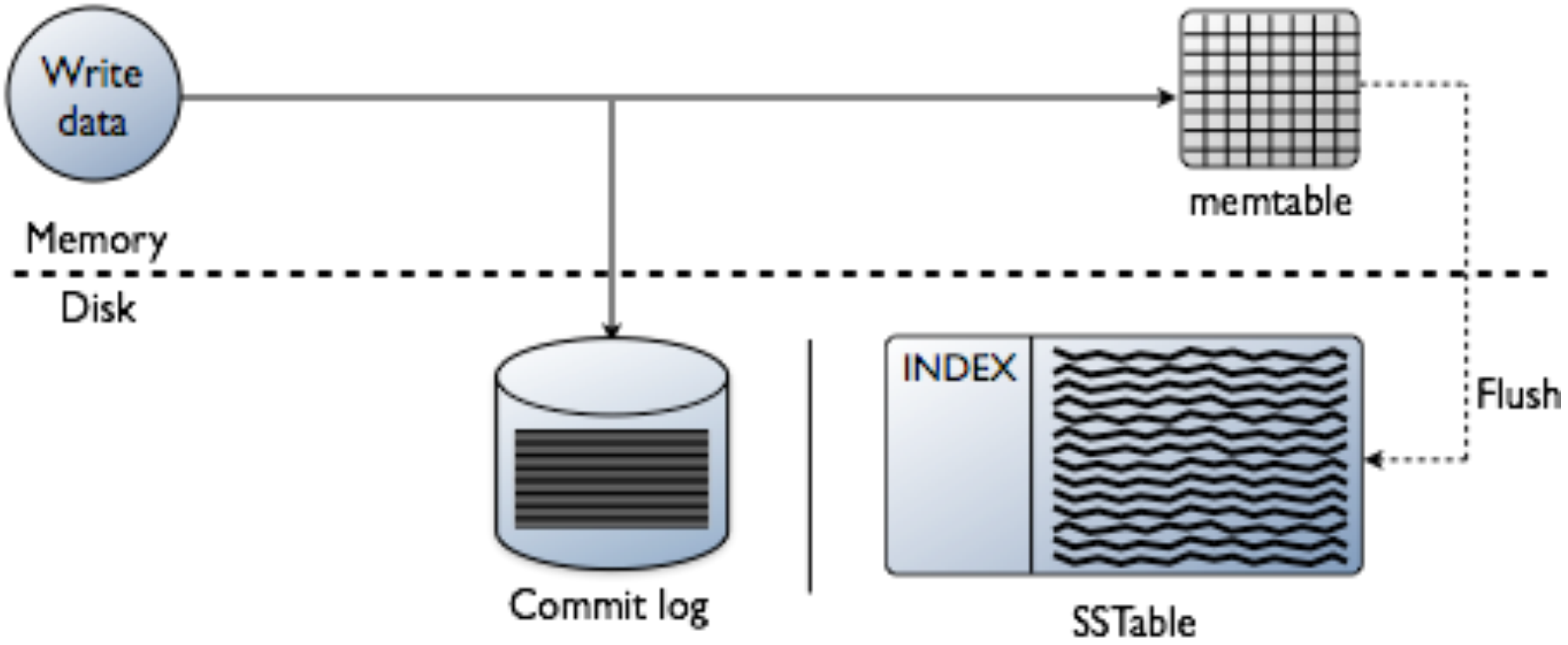
- Compaction
- Deduplication
- Consistency

JIRA in Community

- https://issues.apache.org/jira/browse/CASSANDRA-2527

Cassandra Internals

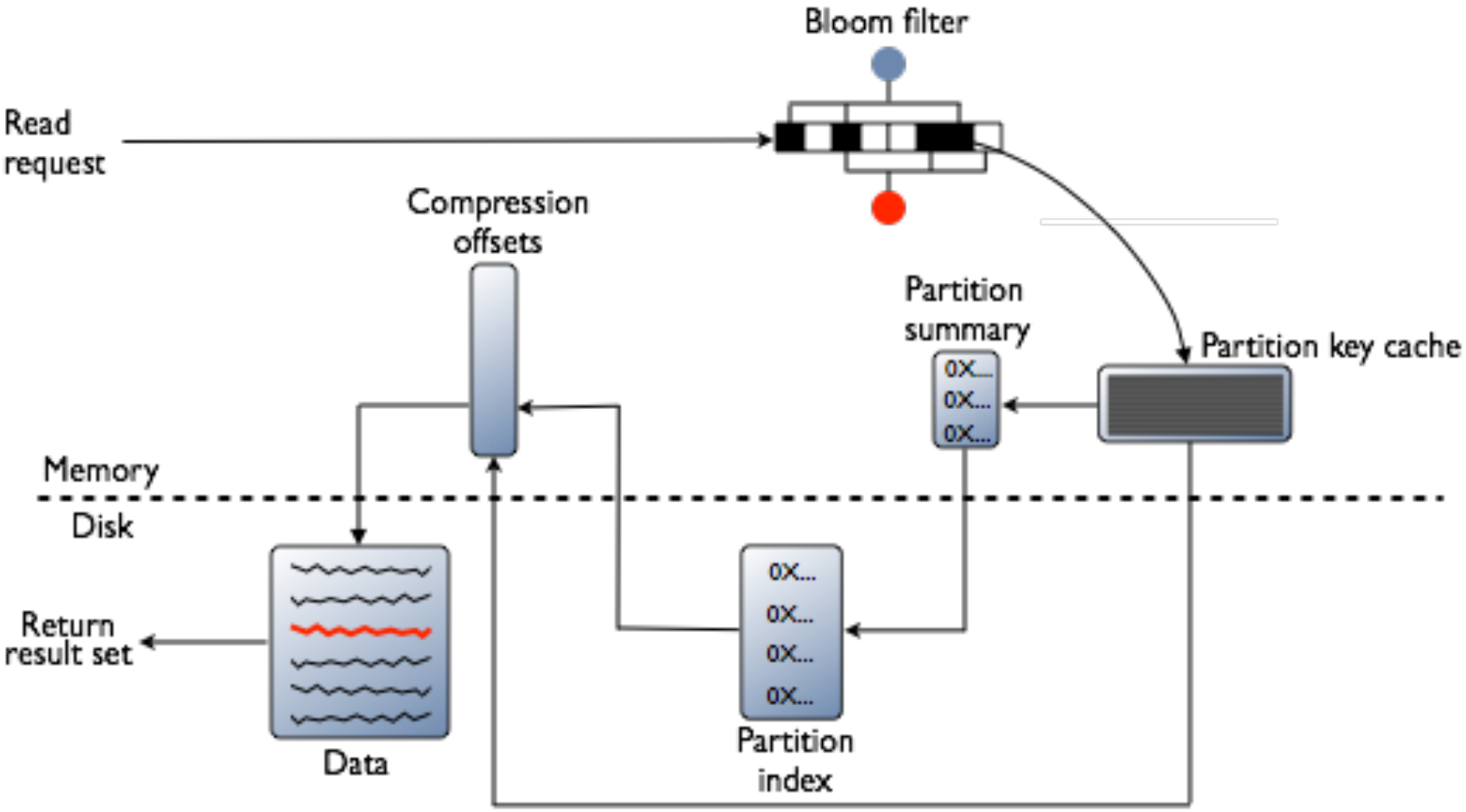
Cassandra Write



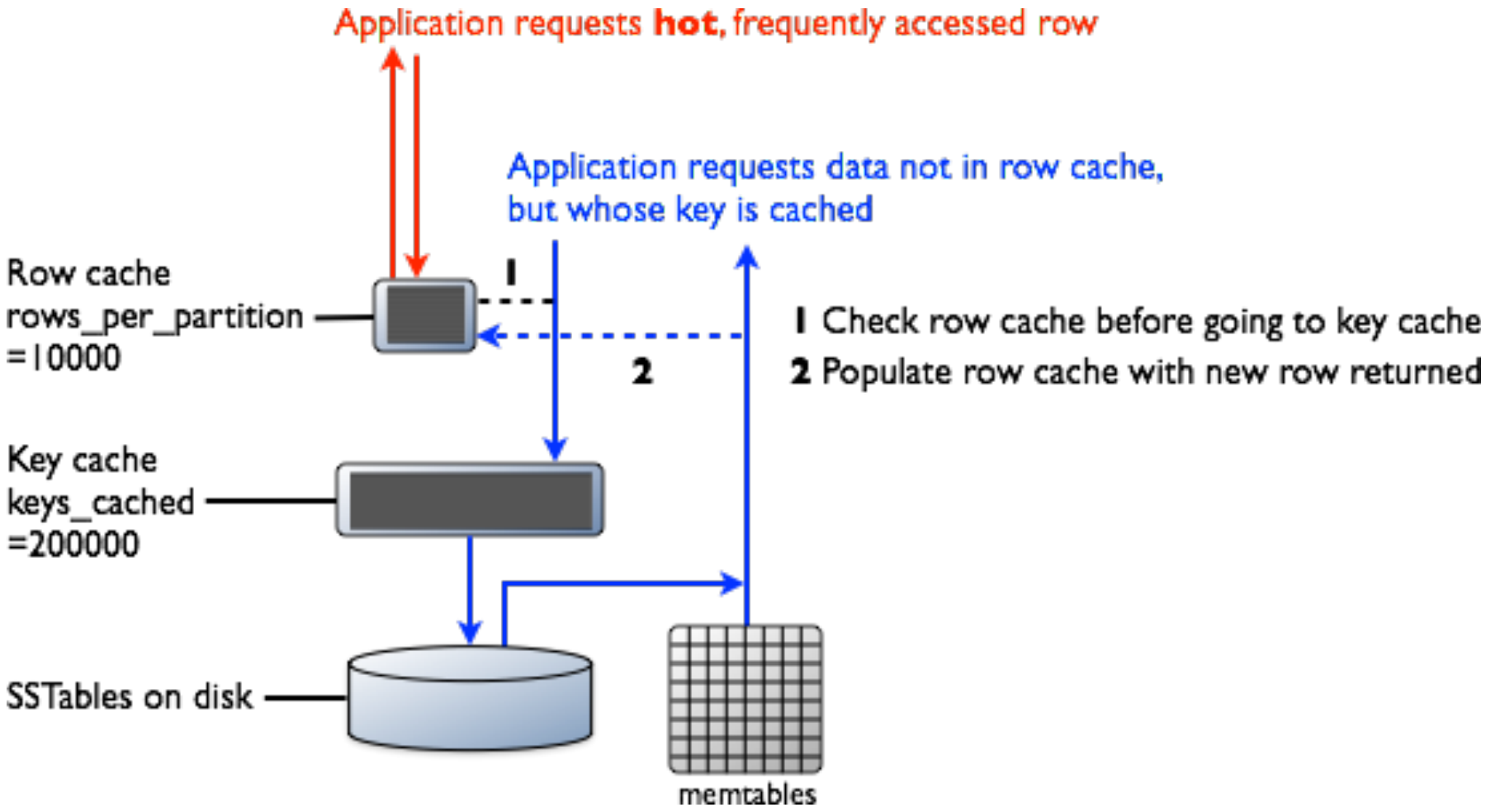
Storing Data on Disk

- **Data (Data.db):** SSTable data
- **Primary Index (Index.db):** Index of the row keys with pointers to their positions in the data file
- Bloom filter (Filter.db)
- **Compression Information (CompressionInfo.db):** A file holding information about uncompressed data length, chunk offsets and other compression information
- Statistics (Statistics.db)
- Digest (Digest.crc32 …)
- CRC (CRC.db)
- SSTable Index Summary (SUMMARY.db)
- SSTable Table of Contents (TOC.txt)
- Secondary Index (SI_.*.db)

Cassandra Read

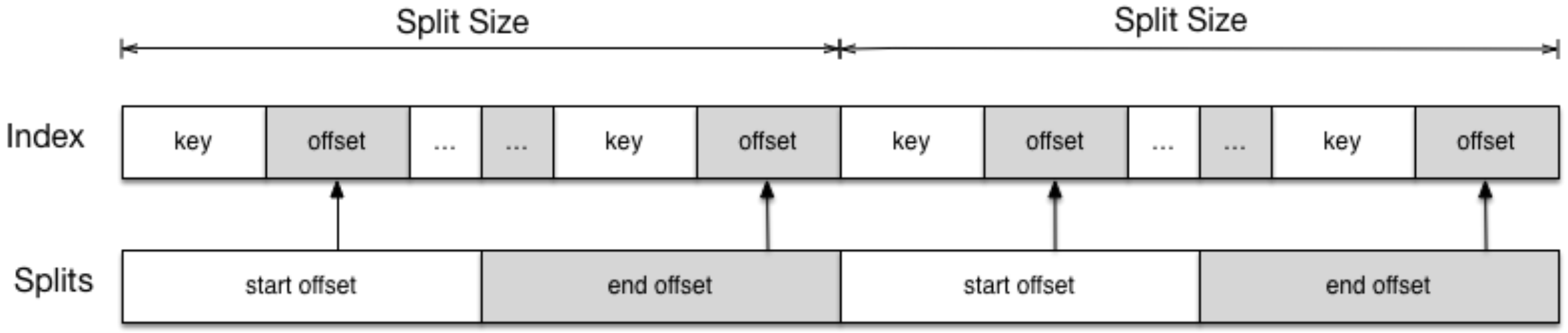


Cassandra Read



Implementation Details

Build Split index for SSTables



Compaction, Deduplication and Consistency

• Compaction works on a collection of SSTables. From these SSTables, compaction collects all versions of each unique row and assembles one complete row, using the most up-to-date version (by timestamp) of each of the row's columns.

• We use reducer to handle the compaction, deduplication and consistency, which is the same logic as C*.

Core Classes

- SSTableIndexInputFormat
- SSTableIndexRecordReader

- SSTableSplitIndexOutputFormat
- SSTableSplitIndexRecordWriter

- SSTableInputFormat
- SSTableRecordReader

- SSTableReducer

- SSTableRowWritable

SSTableIndexInputFormat

```
• protected boolean isSplitable(JobContext context, Path filename) {
  return false;
}

• protected List<FileStatus> listStatus(JobContext job) throws IOException {
  List<FileStatus> files = super.listStatus(job);
  List<FileStatus> indexFiles = new ArrayList<FileStatus>();
  for (FileStatus file : files) {
    if (file.getLen() > 0 && IS_SSTABLE_INDEX.apply(file.getPath())) {
      indexFiles.add(file);
    }
  }
  return indexFiles;
}
```

Performance

- A SSTable of size 100G takes ~10 mins to complete building a snapshot table on HDFS

Similar Projects in Industry

- https://github.com/fullcontact/hadoop-sstable
- https://github.com/Knewton/KassandraMRHelper
- https://github.com/Netflix/aegisthus
- https://github.com/Netflix/Priam

Quick Start

Configuration

- hadoop.sstable.cql="CREATE TABLE ..."
- hadoop.sstable.keyspace="<KEYSPACE>"

- mapreduce.job.reduces=<REDUCE_NUM>

Upload SSTables to Hadoop

- sstable-backup/bin/backup.sh

- **Example**:
- pssh -i -h conf/hosts -p 7 -t 0 /data/applications/sstable-backup/bin/backup.sh -k <keyspace> -cf <column_family> -s <snapshot> -o <output_dir>

Build index

- $HADOOP_HOME/bin/hadoop jar hadoop-cassandra-1.0-SNAPSHOT-allInOne.jar com.ebay.traffic.cassandra.sstable.index.hadoop.mapreduce.BuildIndex <input> <output>

SSTable to Hadoop file format

- $HADOOP_HOME/bin/hadoop jar hadoop-cassandra-1.0-SNAPSHOT-allInOne.jar com.ebay.traffic.cassandra.sstable.hadoop.mapreduce.SSTable2Hadoop -D mapreduce.job.reduces=<REDUCE_NUM> <input> <output>

Q & A

Thanks