HBase VS Cassandra

RDBMS (SQL)

- Pro
  - Relational database have provided a standard persistence model
  - SQL has become a de-facto standard model of data manipulation
  - Relational database manage concurrency for transaction
  - Relational database have lots of tools
- Cons
  - scale vertically (big box)
  - difficult to partition/shard

HBase

Welcome to Apache HBase™

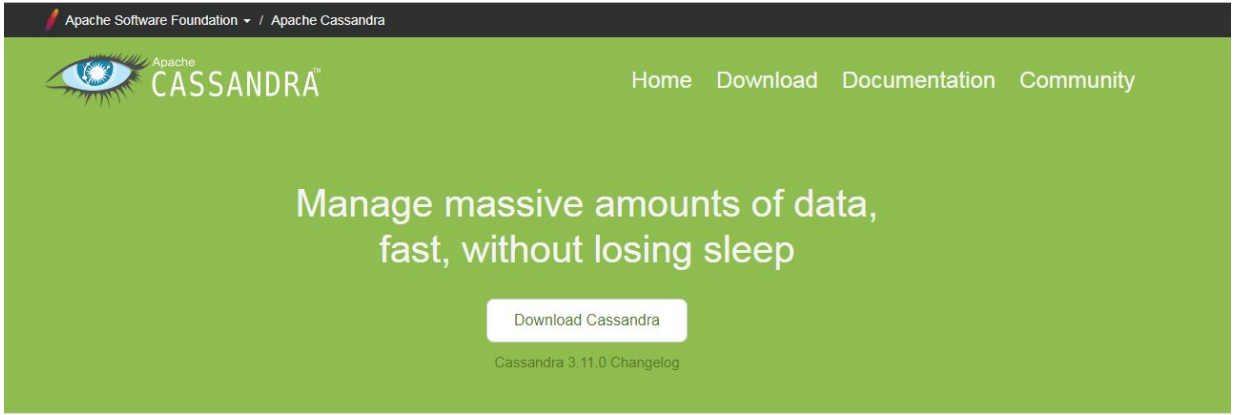
Apache HBase™ is the Hadoop database, a distributed, scalable, big data store.

Use Apache HBase™ when you need random, realtime read/write access to your Big Data. This project's goal is the hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware. Apache HBase is an open-source, distributed, versioned, non-relational database modeled after Google's Bigtable: A Distributed Storage System for Structured Data by Chang et al. Just as Bigtable leverages the distributed data storage provided by the Google File System, Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS.

Features

- Linear and modular scalability.
- Strictly consistent reads and writes.
- Automatic and configurable sharding of tables
- Automatic failover support between RegionServers.
- Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables.
- Easy to use Java API for client access.
- Block cache and Bloom Filters for real-time queries.
- Query predicate push down via server side Filters
- Thrift gateway and a REST-ful Web service that supports XML, Protobuf, and binary data encoding options
- Extensible jruby-based (JIRB) shell
- Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX

Cassandra



What is Cassandra?

The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data.Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages.

PROVEN

Cassandra is in use at Constant Contact, CERN, Comcast, eBay, GitHub, GoDaddy, Hulu, Instagram, Intuit, Netflix, Reddit, The Weather Channel, and over 1500 more companies that have large, active data sets.

FAULT TOLERANT

Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime.

PERFORMANT

Cassandra consistently outperforms popular NoSQL alternatives in benchmarks and real applications, primarily because of fundamental architectural choices.

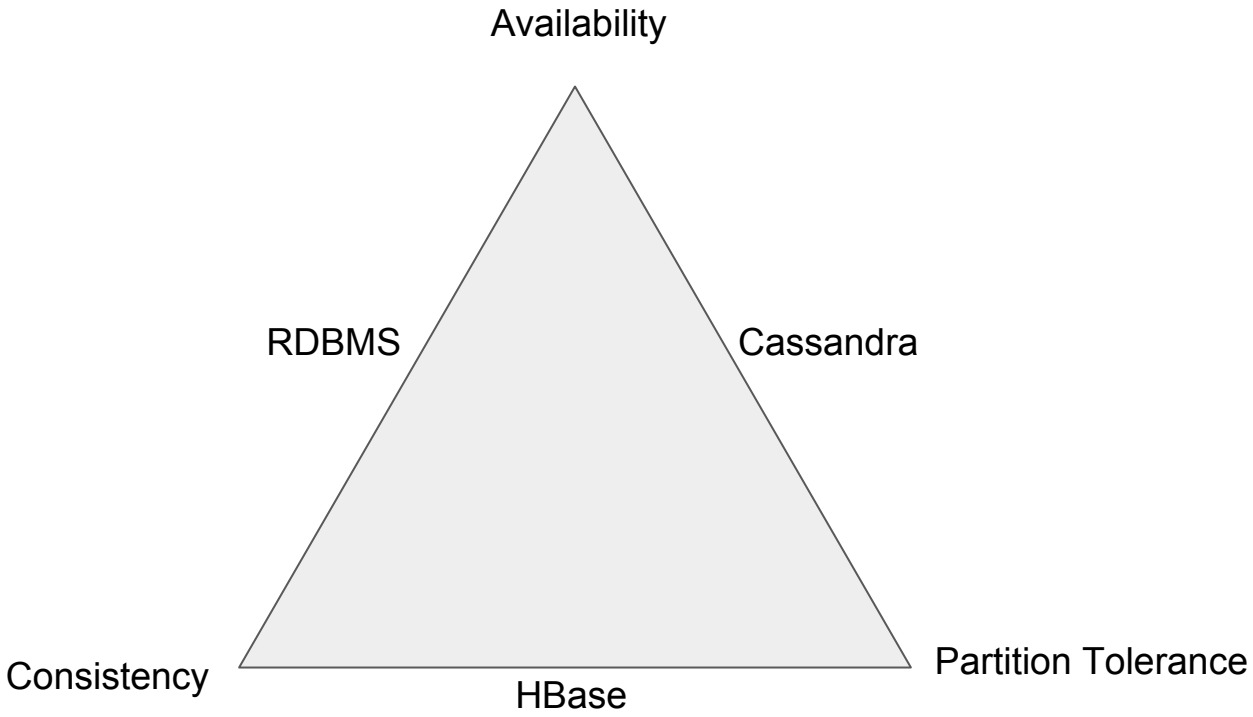
DECENTRALIZED

There are no single points of failure. There are no network bottlenecks. Every node in the cluster is identical.

SCALABLE

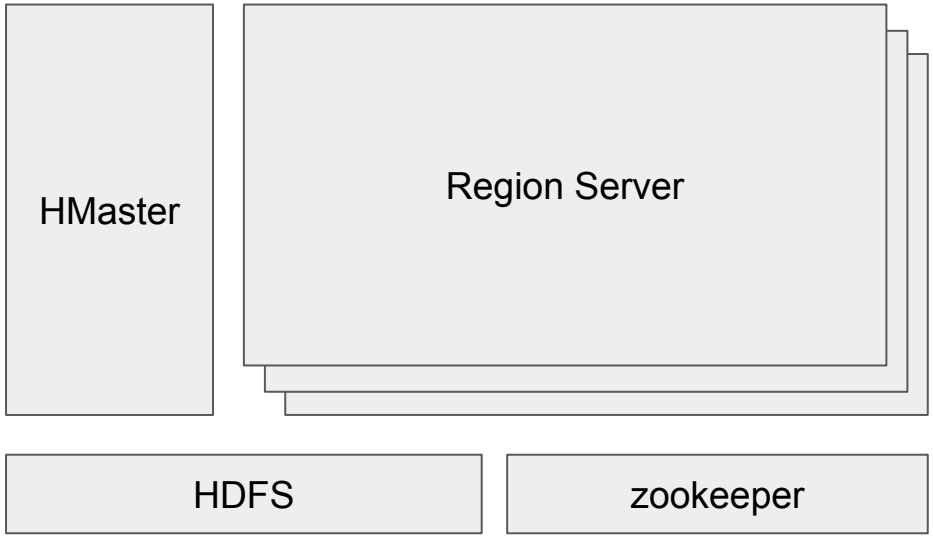
Some of the largest production deployments include Apple's, with over 75,000 nodes storing over 10 PB of data, Netflix (2,500 nodes, 420 TB, over 1 trillion requests per day), Chinese search engine Easou (270 nodes, 300 TB, over 800 million requests per day), and eBay (over 100 nodes, 250 TB).

CAP

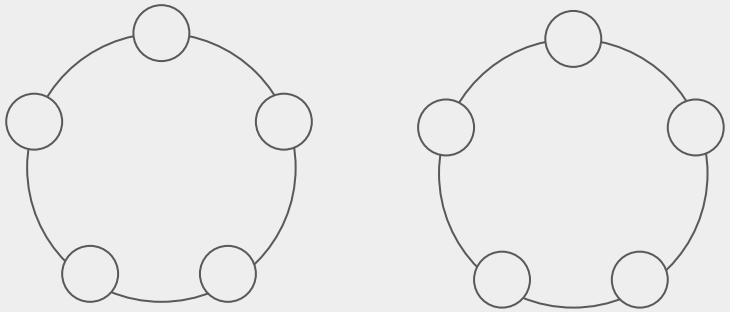

Availability
RDBMS
Cassandra
Consistency
HBase
Partition Tolerance

Architecture

HBase

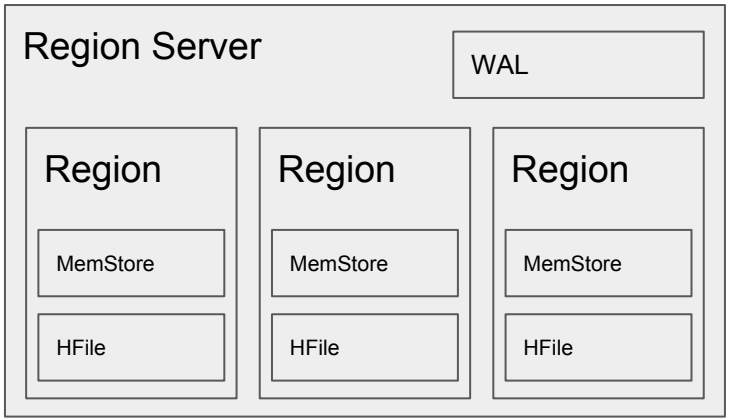


Cassandra

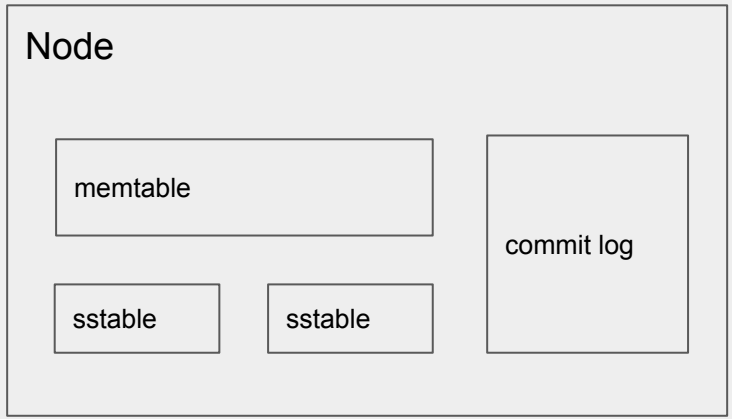


Architecture

HBase



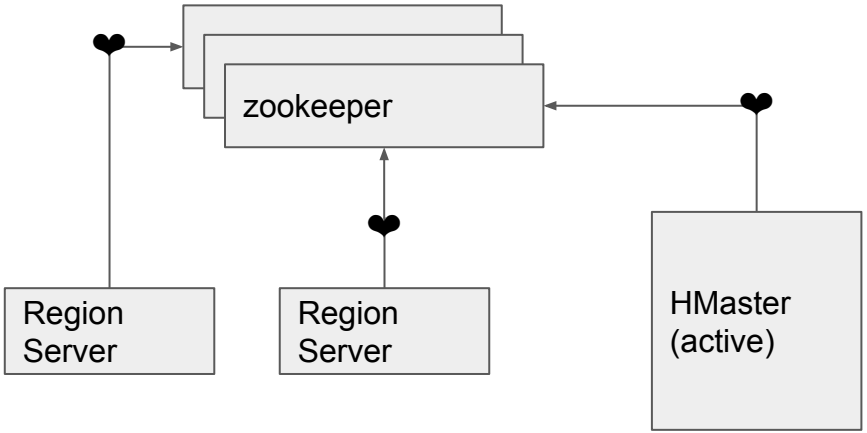
Cassandra



Health

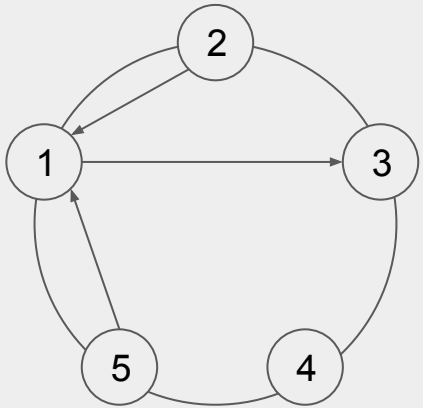
HBase

- zookeeper
- 1 active HMaster only
- heartbeat



Cassandra

- gossip protocol
- P2P
- detect slow node



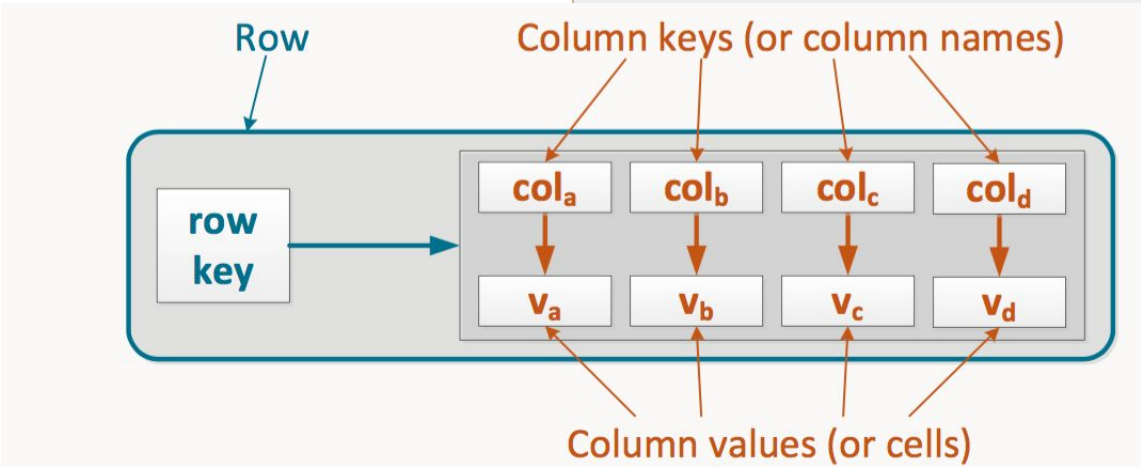
Data Model

HBase

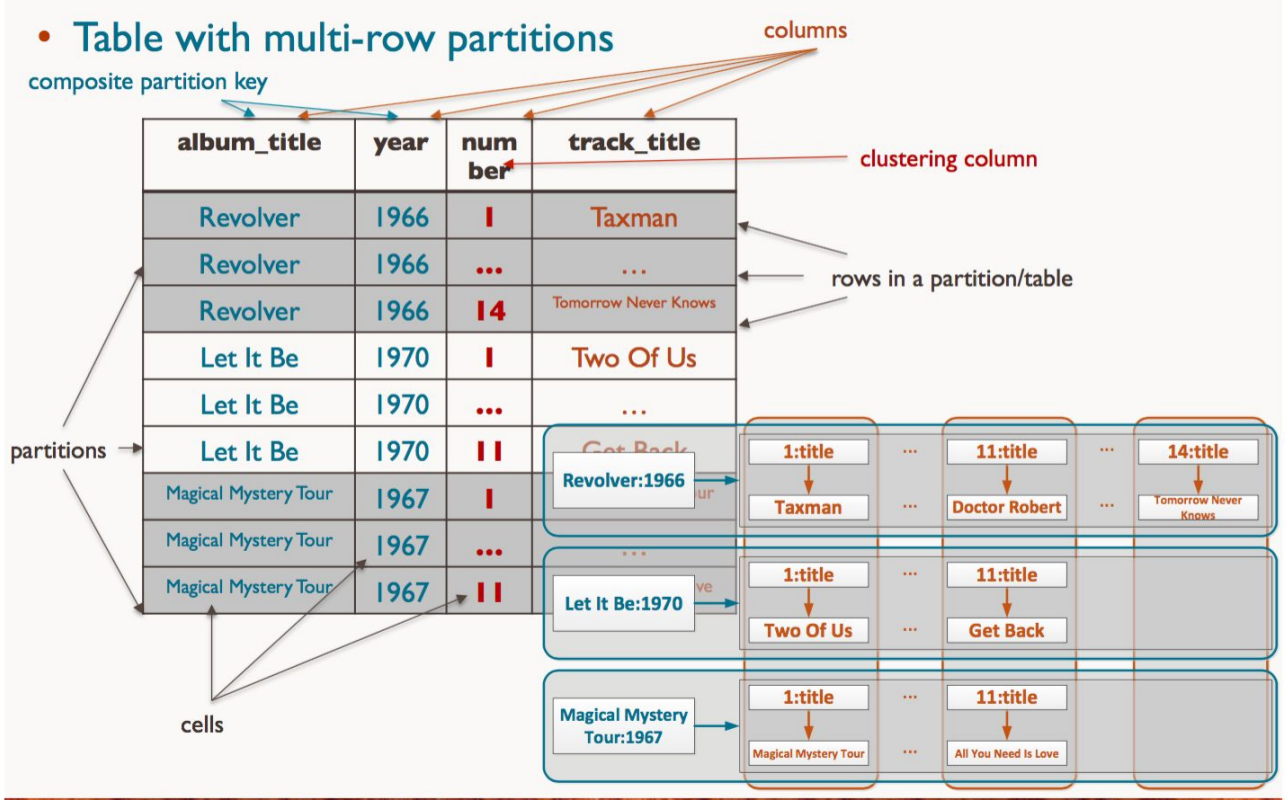
- row key
- column key
- column family

Cassandra

- partition key
- clustering key
- wide row



Wide Row


• Table with multi-row partitions
columns
composite partition key
clustering column
rows in a partition/table
partitions
cells
album_title
year
number
track_title
Revolver 1966 1 Taxman
Revolver 1966 ... ...
Revolver 1966 14 Tomorrow Never Knows
Let It Be 1970 1 Two Of Us
Let It Be 1970 ... ...
Let It Be 1970 11
Magical Mystery Tour 1967 1
Magical Mystery Tour 1967 ...
Magical Mystery Tour 1967 11
Revolver:1966
1:title
Taxman
11:title
Doctor Robert
14:title
Tomorrow Never Knows
Let It Be:1970
1:title
Two Of Us
11:title
Get Back
Magical Mystery Tour:1967
1:title
Magical Mystery Tour
11:title
All You Need Is Love

Replication

HBase

- HDFS
- Controlled by Table

Cassandra

- Controlled by keyspace
- 1 = 1 copy = no replication
- Strategy
  - simple
  - network topology
    - different rack
    - different factor per data center

Writing

HBase

- Atomic
- Append Only
- History

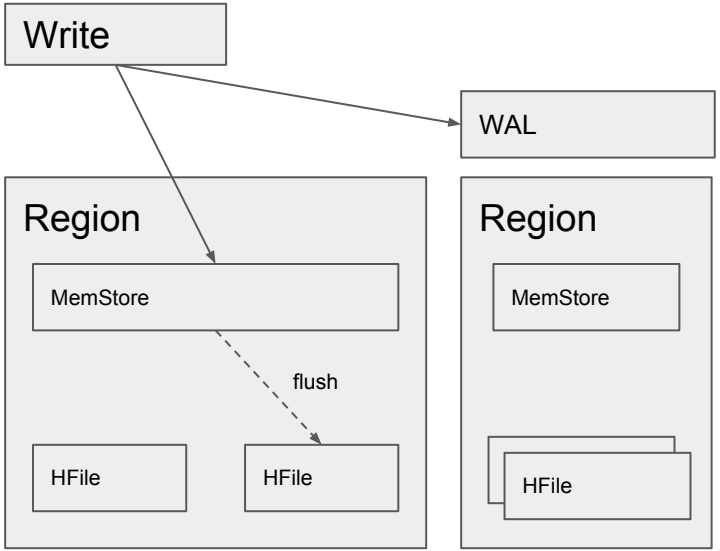
Cassandra

- Atomic
- Append Only
- Tunable consistency (ANY, QUORUM, ALL, …)
- Batch
- Lightweight Transaction
- Each value is timestamped

Writing Path

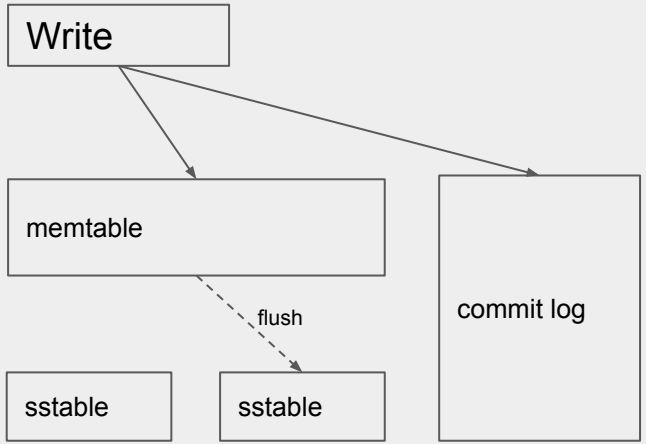
HBase

Region Server

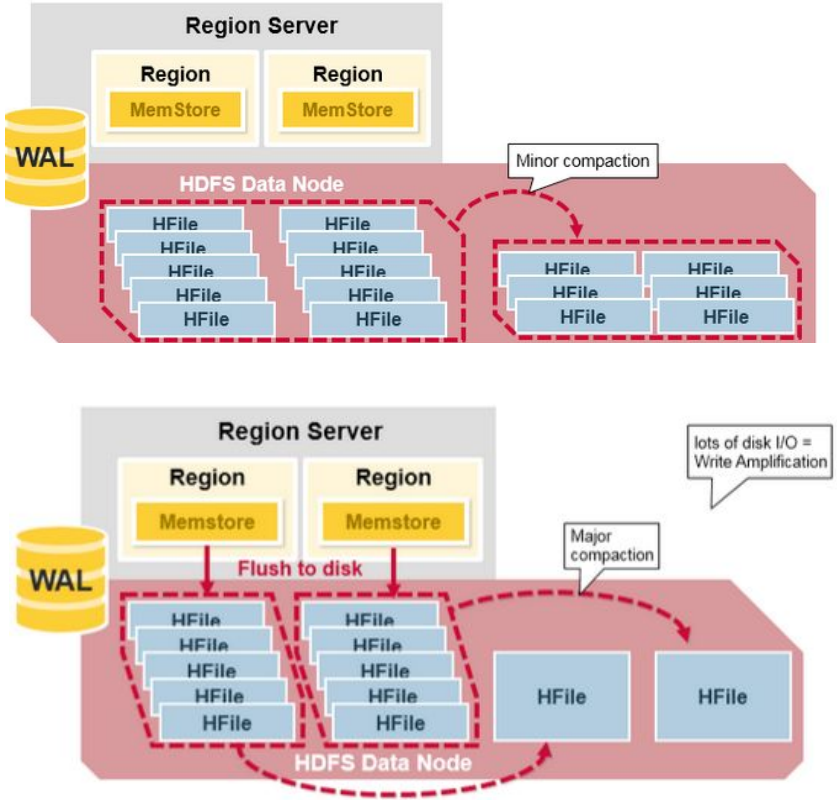


Cassandra

Node



Compaction



Slow Compaction I/O
Compaction impact Size
(x3)

Reading

HBase

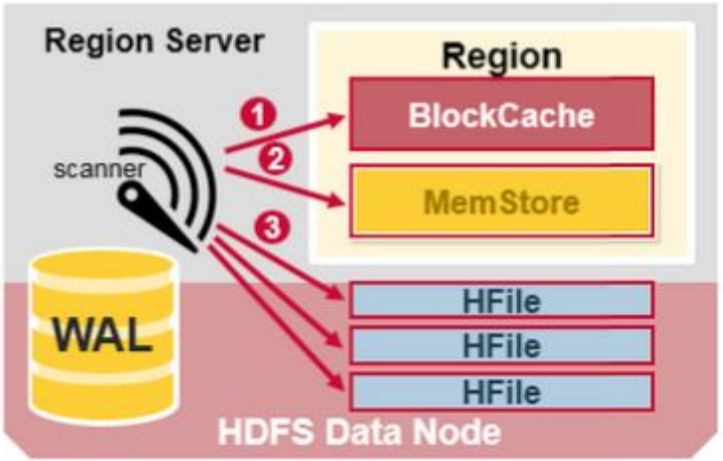
- scanner
- BlockCache
- bloom filter

Cassandra

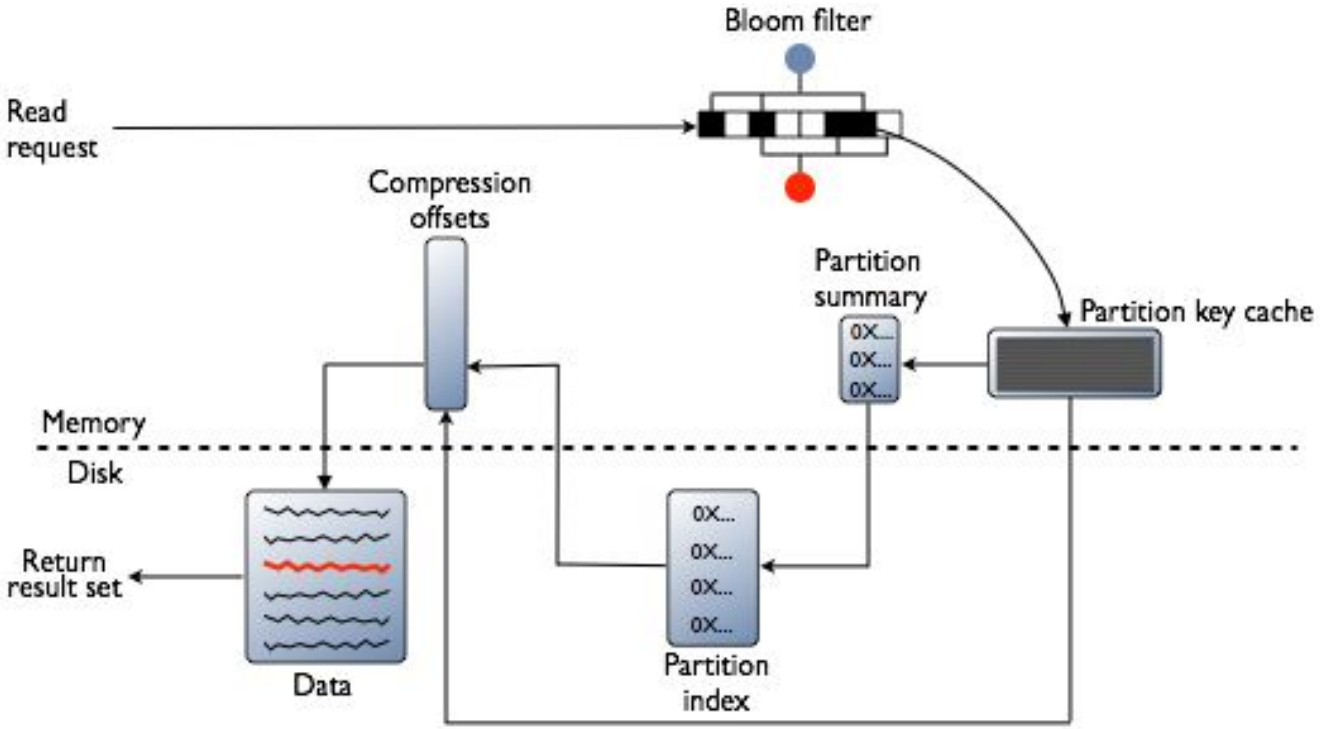
- slower than writing
- tunable consistency (ONE, QUORUM, ALL, …)
- bloom filter
- If conflict, most recent timestamp win

Reading Path HBase

1. First the scanner looks for the Row KeyValues in the Block cache
2. Next the scanner looks in the MemStore
3. If all row cells not in MemStore or blockCache, look in HFiles



Reading Path Cassandra



Query

HBase

- Java API
- REST
- Thrift Server
- ~~JOIN~~
- Key
- Key Range
- Regex on Row Column Value (low level API)
- Coprocessor

Cassandra

- CQL = ~ SQL
- ~~JOIN~~
- Key can be compound
- Key must be unique
- WHERE (on key)
- GROUP BY (on clusturing key since 2.2)
- Materialized view (since 3.0)

Benchmark

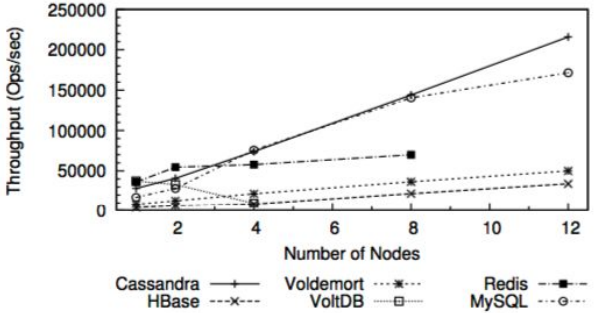


Figure 6: Throughput for Workload RW

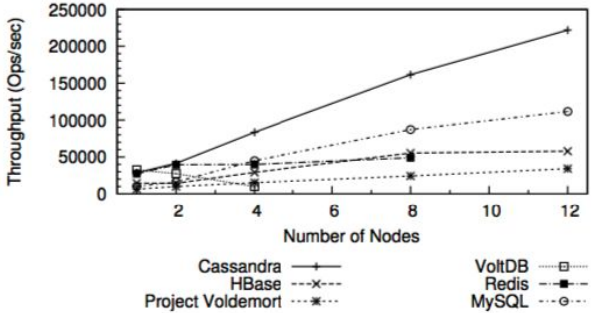


Figure 9: Throughput for Workload W

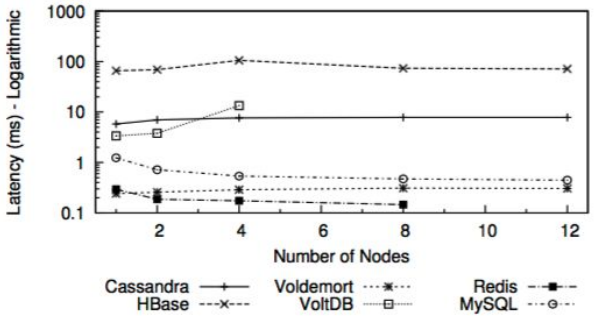


Figure 7: Read latency for Workload RW

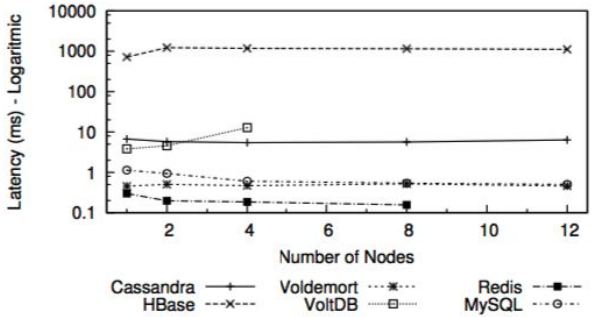
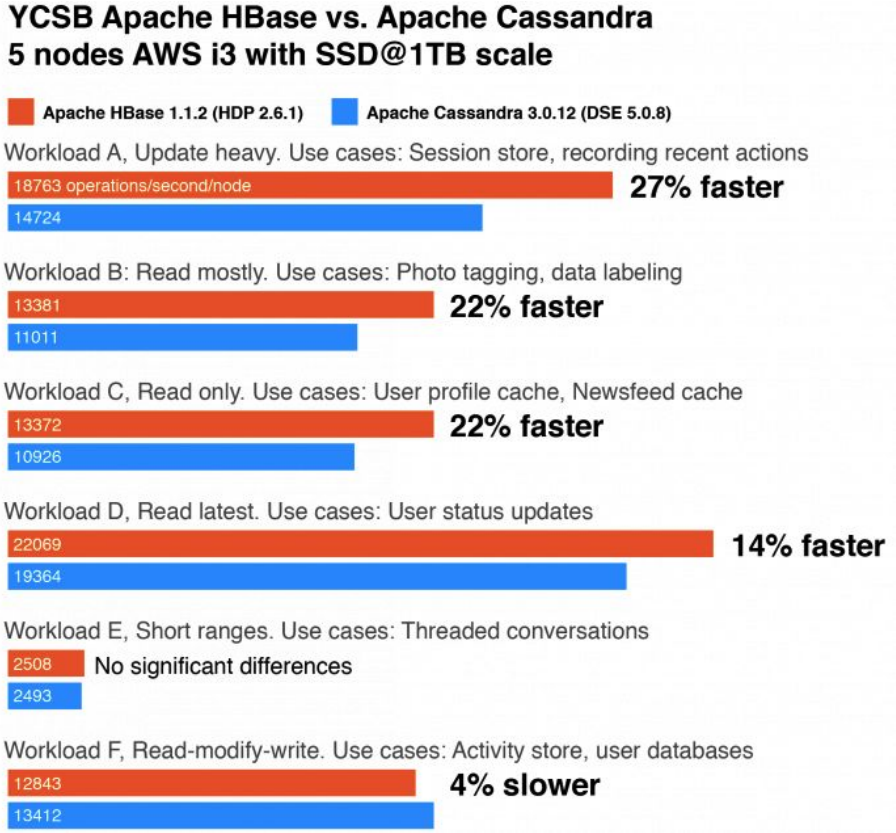


Figure 10: Read latency for Workload W

Benchmark


YCSB Apache HBase vs. Apache Cassandra
5 nodes AWS i3 with SSD@1TB scale
Apache HBase 1.1.2 (HDP 2.6.1)
Apache Cassandra 3.0.12 (DSE 5.0.8)
Workload A, Update heavy. Use cases: Session store, recording recent actions
18763 operations/second/node
14724
27% faster
Workload B: Read mostly. Use cases: Photo tagging, data labeling
13381
11011
22% faster
Workload C, Read only. Use cases: User profile cache, Newsfeed cache
13372
10926
22% faster
Workload D, Read latest. Use cases: User status updates
22069
19364
14% faster
Workload E, Short ranges. Use cases: Threaded conversations
2508
2493
No significant differences
Workload F, Read-modify-write. Use cases: Activity store, user databases
12843
13412
4% slower

Fair Bench

Backlog:

Construire une application Vision Client 360

- Restituer toutes les infos clients
- 1 seule table

Alimentation en fil de l'eau de la base

- Job en Spark Streaming

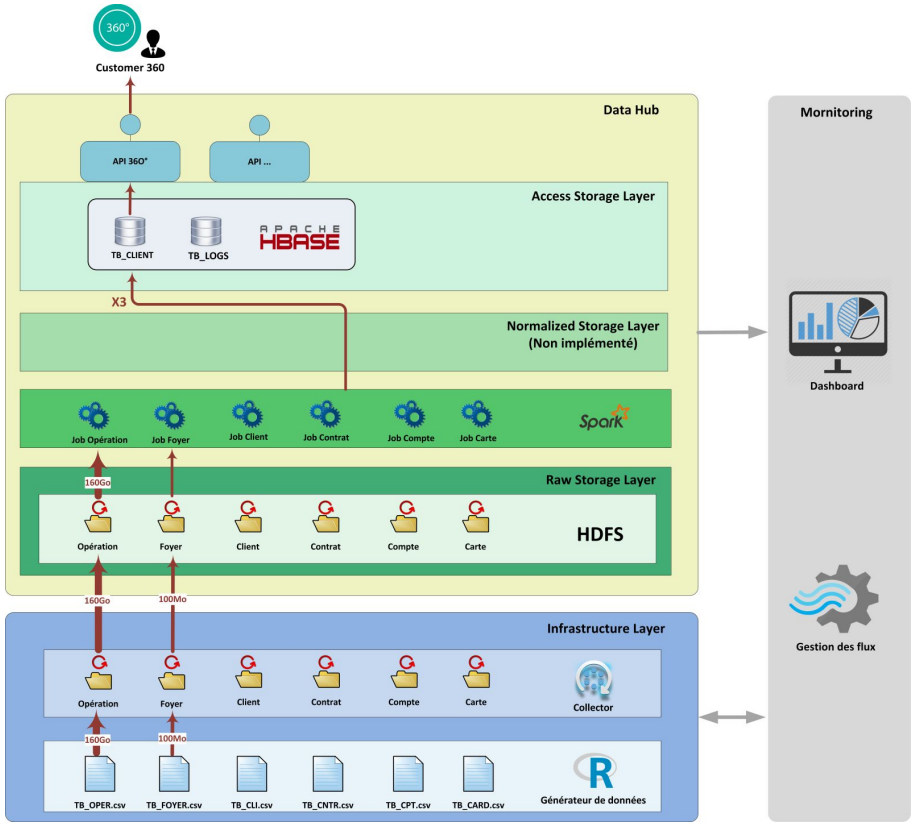
Intégration des flux

- Arrivée des flux à fréquence variable

Générer les jeux de données:

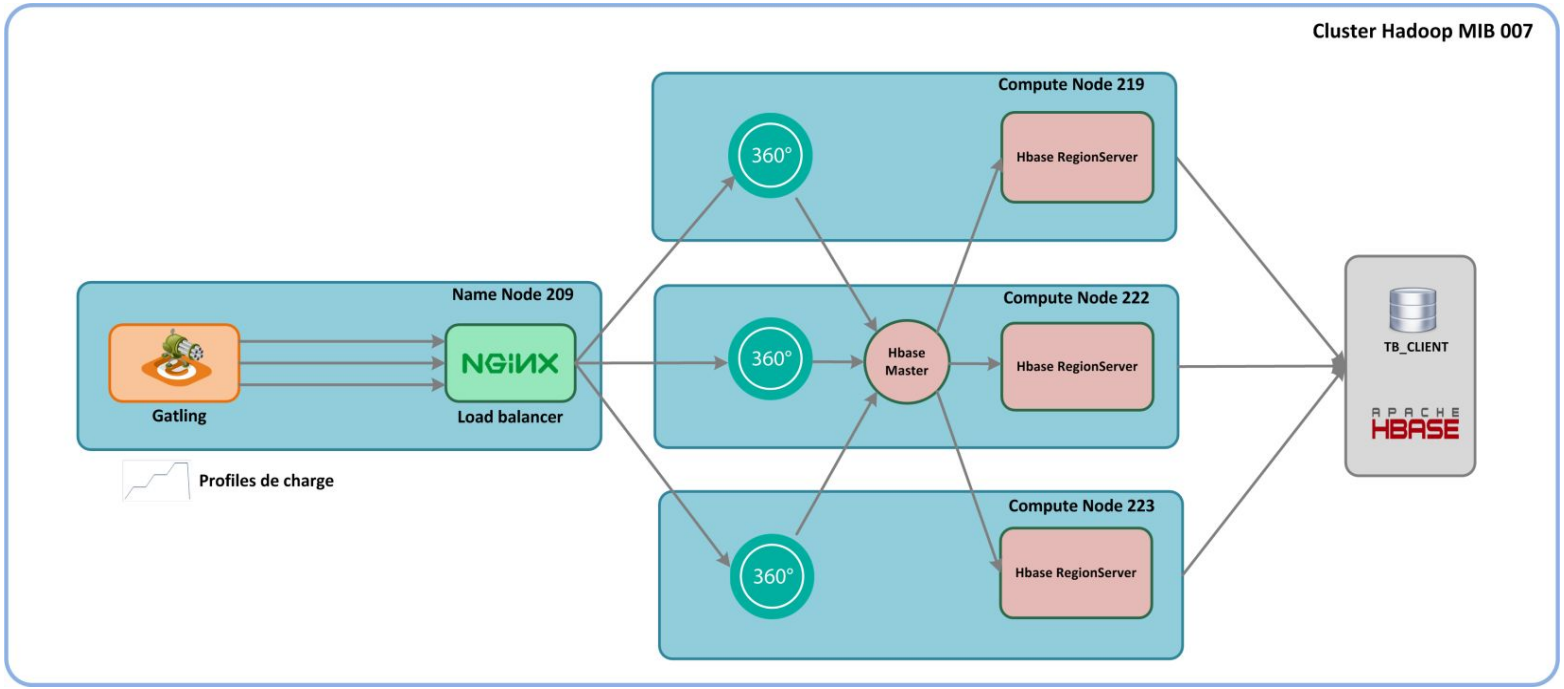
- 10 ans d'historiques
- 1 million de clients
- 1 milliard d'opérations

Infra: IPS/BP2i



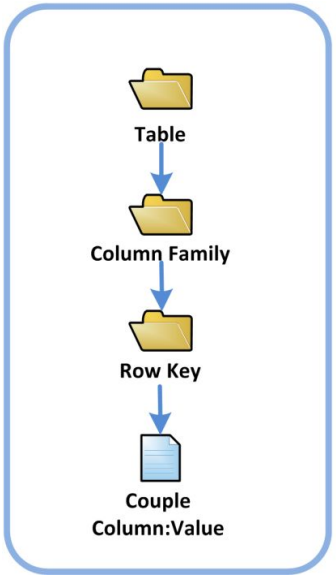
Setup: HBase

Hadoop Cluster : 5To (Isilon), 4VM -> 128Go, 16 Core

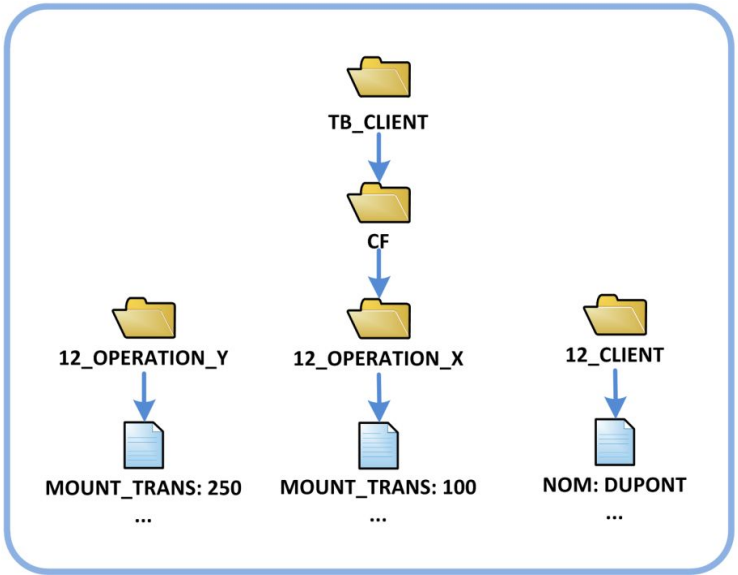


HBase DataModel

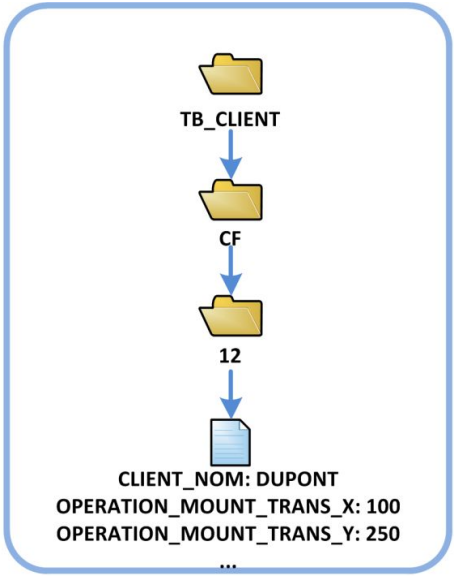
Modèle générique Hbase



Modèle Accès Multi-clés (scan)



Modèle Accès Mono-clé (Get)



Result: HBase

Scan 1500 Users:

- Avg response time: 147ms
- Throughput: 200 query/s

Result: HBase

Type	Natif	Rest	Thrift
Get	Temps Rép. Moyen: **164 ms** Débit : **26 Req/s**	Temps Rép. Moyen: **261 ms** Débit : **25,7 Req/s**	Temps Rép. Moyen: **204 ms** Débit : **25,9 Req/s**
Scan	Temps Rép. Moyen: **156 ms** Débit : **26 Req/s**		

Result: Cassandra

- Slower
- due to poor VM disk performance Vs Isilon

Use Case

HBase

- read intensive
- range scan

Cassandra

- write intensive
- geographical distribution
- evolving applications

Beyond Database

HBase

- HDFS
- Spark on YARN
- Solr (Search on Hadoop)

Cassandra

- DataStax Enterprise FS
- DSE Analytics (Spark on Cassandra)
- DSE Search (Solr)

Thank you