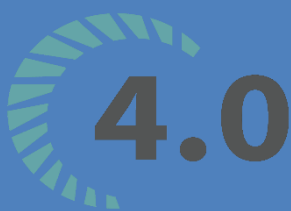


BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN  
ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH  
ĐẠI HỌC QUỐC GIA TP HCM

# HỆ CƠ SỞ DỮ LIỆU NÂNG CAO



Sinh viên thực hiện:

20C12007 – Trần Đình Lâm

20C11035 – Trương Thế Kiệt

20C11040 – Đặng Nhật Minh

ĐỒ ÁN MÔN HỌC - HỆ CƠ SỞ DỮ LIỆU NÂNG CAO

NĂM HỌC 2020-2021



## BẢNG THÔNG TIN CHI TIẾT NHÓM

<b>Mã nhóm:</b>	<b>14</b>			
<b>Tên nhóm:</b>	<b>K2014</b>			
<b>Số lượng:</b>	<b>3</b>			
<b>MSSV</b>	<b>Họ tên</b>	<b>Email</b>	<b>Điện thoại</b>	<b>Hình ảnh</b>
20C12007	Trần Đình Lâm	tdlam123@gmail.com	0383522356	
20C11035	Trương Thế Kiệt	truongthekiet709@gmail.com		
20C11040	Đặng Nhật Minh	minhdangnhat685@gmail.com		

Bảng phân công & đánh giá hoàn thành công việc			
Người thực hiện	Công việc thực hiện	Mức độ hoàn thành	Đánh giá của nhóm
20C12007 Trần Đình Lâm	Phân công các công việc cần thực hiện	90%	8/10
	Tìm hiểu và cài đặt được HBase trên lap cá nhân		
	Phối hợp với Minh viết báo cáo		
	Quay video demo sử dụng HBase Java API		
20C11035 Trương Thế Kiệt	Soạn slide theo tiến độ của báo cáo	90%	8/10
	Cắt video đã quay thành bài trình bày		
	Nộp bài		
	So sánh HBase vs Cassandra		
20C11040 Đặng Nhật Minh	Tìm hiểu Column Family	90%	8/10
	Cài đặt được HBase trên lap cá nhân		
	Quay record demo sử dụng HBase shell		
	Phối hợp với Lâm viết báo cáo		



## YÊU CẦU ĐỒ ÁN

Loại bài tập	<input checked="" type="checkbox"/> Lý thuyết <input checked="" type="checkbox"/> Đồ án
Ngày bắt đầu	22/02/2021
Ngày kết thúc	07/03/2021

**Tìm hiểu nội dung, đặc điểm, và các vấn đề liên quan đến loại lưu trữ dữ liệu Column Family, và một cơ sở dữ liệu cụ thể kèm theo (HBase):**

- Báo cáo loại lưu trữ NoSQL/NewSQL theo qui định
- Tài liệu kỹ thuật về nghiên cứu loại sản phẩm NoSQL qui định: cài đặt, các thao tác trên db: tạo db, thêm, xóa, sửa,... và các thao tác nâng cao khác (nếu có)
- Phim demo sử dụng sản phẩm (tối đa 7 phút)
- Clip nhóm tự thuyết trình (tối đa 7 phút)
- File powerpoint (và dùng báo cáo trong buổi học)

# MỤC LỤC

<b>TỔNG QUAN COLUMN FAMILY .....</b>	<b>1</b>
1. Định nghĩa.....	1
2. Đặc điểm .....	1
3. Ưu điểm.....	2
4. Nhược điểm.....	2
<b>APACHE HBASE .....</b>	<b>3</b>
1. HBase là gì? .....	3
2. Vì sao cần HBase? .....	5
3. HBase vs RDBMS.....	6
4. HBase Data Model .....	7
5. Kiến trúc Hbase.....	9
6. Thực hành cài đặt Hbase.....	12
7. Các thao tác & công cụ cơ bản.....	17
<b>KẾT LUẬN VÀ MỞ RỘNG .....</b>	<b>18</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>20</b>

## TỔNG QUAN COLUMN FAMILY

### 1. Định nghĩa

Column Family là một database object trong Column-Oriented NoSQL Database, với dữ liệu được lưu trữ và truy xuất theo các cột thay vì các hàng như trong các loại cơ sở dữ liệu quan hệ

Row-oriented			
ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented					
Name	ID	Grade	ID	GPA	ID
John	001	Senior	001	4.00	001
Karen	002	Freshman	002	3.67	002
Bill	003	Junior	003	3.33	003

Hình 1 Row-oriented và column-oriented

Các hệ CSDL dựa trên Column Family hiện nay trên cộng đồng gồm một số tên tuổi lớn như:

- Big table
- HBase
- Cassandra

### 2. Đặc điểm

- Mỗi Column Family bao gồm nhiều hàng
- Mỗi hàng có thể chứa các cột tùy ý (không cần thiết phải giống nhau giữa các hàng)

- Nhiều Column Family có liên hệ với nhau về mặt logic tạo thành 1 cơ sở dữ liệu hoàn chỉnh (Column Families)
- Được tối ưu cho các hệ thống online analytical processing (các thao tác chủ yếu là query thông tin trên các cột để phân tích), giảm khối lượng công việc và thời gian cần để thao tác với dữ liệu trên đĩa cứng
- Thích hợp với các hệ thống data warehousing và xử lý Big Data

### 3. Ưu điểm

- Compression: do dữ liệu trên mỗi Column Family chỉ gồm 1 loại, nên có thể chọn cách nén riêng cho từng Column Family, làm tăng hiệu quả
- Dễ dàng mở rộng và chia nhỏ (scalability and partitioning)
- Nhanh với những query chỉ cần dữ liệu trên 1 Column Family
- Tốc độ tính toán các phép toán cần truy xuất trên toàn bộ tập dữ liệu (dataset) như SUM, COUNT, AVG, ... nhanh và hiệu quả hơn

### 4. Nhược điểm

- Không hỗ trợ transaction
- Chậm với các thao tác insert-update-delete
- Chậm với các câu query cần truy xuất trên nhiều Column Family



## APACHE HBASE

### 1. HBase là gì?

HBase là một loại NoSQL, column-oriented Database phát hành lần đầu năm 2008, lưu trữ dữ liệu theo cột thay vì theo hàng như RDBMS. HBase có nguồn gốc từ cơ sở dữ liệu BigTable của Google, chạy trên nền Hadoop Distributed File System (HDFS), phát triển bởi Apache.

HBase được phân loại là một NoSQL lưu trữ dạng key-value. Value được định danh bởi một key, cả key và value đều được lưu trữ dạng ByteArray. Nói cách khác, HBase là một dạng NoSQL lưu trữ phi cấu trúc. Ở dạng lưu trữ có cấu trúc (RDBMS), cấu trúc dữ liệu phải được khai báo đầy đủ rõ ràng, và đối tượng được lưu trữ theo cột, dòng và có mối quan hệ chặt chẽ với nhau. Ngược lại, HBase cung cấp cách thức lưu trữ đa dạng các loại dữ liệu mà không cần khai báo tường minh trước.

### Các đặc trưng của HBase:

- Là dự án open source có khả năng scale theo chiều ngang (scale out/horizontal scale)
- Được viết bằng Java, chạy trên nền JVM
- Được thiết kế để lưu trữ, xử lý dữ liệu lớn
- Xử lý tốt các loại dữ liệu thưa (nhiều giá trị rỗng)
- HBase là database lưu trữ dạng bảng mà không cần khai báo trước schema. Tại thời điểm tạo bảng, ta chỉ cần khai báo trước column family.

### Một số tính chất của HBase:

- **Distributed:**

HBase có 2 distributed mode

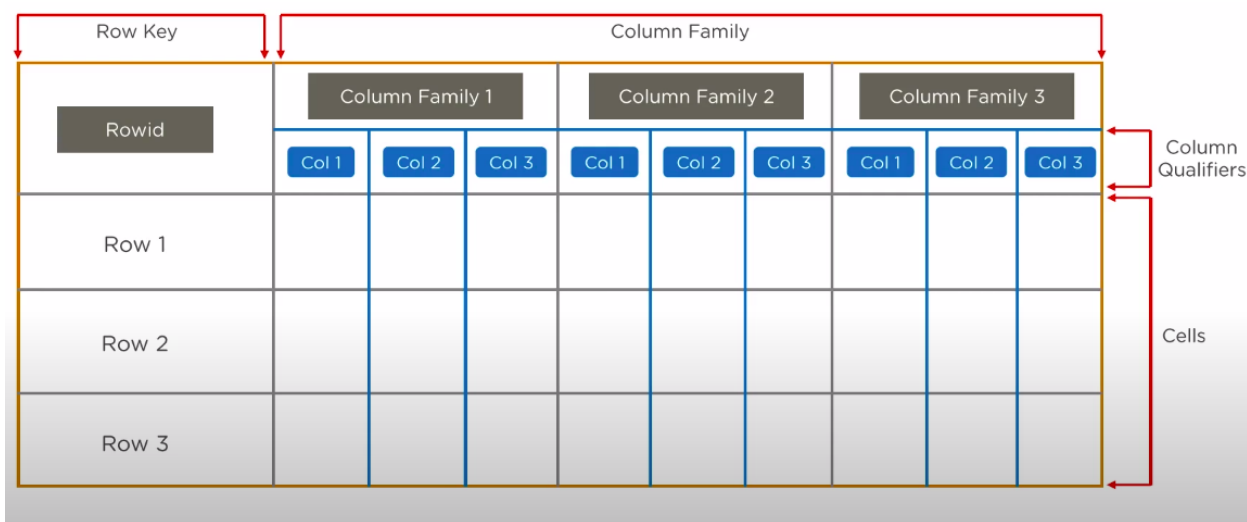
- Pseudo-distributed (giả phân tán): Mỗi thành phần của HBase là một process riêng lẻ, và đều chạy trên 1 node. Ở mode này ta có thể lưu file

local hoặc lưu trên HDFS. Khi chạy trên mode này dĩ nhiên khi node gặp sự cố, cả hệ thống sẽ bị ngưng.

- Fully-distributed (phân tán hoàn toàn): Mode này thường được dùng để vận hành sản phẩm thật vì được chạy trên một hệ thống gồm nhiều node. Ở mode này HBase còn hỗ trợ cả auto-sharding, tức tự động chia tách và phân tán các bảng dữ liệu khi bảng quá lớn.

- **Flexible Data:**

HBase được lấy ý tưởng từ Google BigTable và chạy trên nền Hadoop. Đối tượng cơ bản mà nó lưu trữ là table (bảng). Một table sẽ bao gồm các column family (họ cột), trong mỗi column family lại chứa nhiều column. Sau cùng, dữ liệu sẽ được lưu thành các row (dòng). Các dòng sẽ được sắp xếp theo thứ tự từ điển dựa vào key (khóa) tương ứng để phục vụ tối ưu khi truy vấn. Ngoài



ra, HBase không quy định trước kiểu dữ liệu, vì tất cả các loại dữ liệu đều được lưu dưới dạng ByteArray.

- **Non-Relational:** NoSQL database vận hành theo cơ chế storage-and-query, nên sẽ không tồn tại các quan hệ giữa các bảng.
- **Big data storage:** HBase lưu trữ dữ liệu trên HDFS nên cũng được thừa hưởng các đặc trưng của hệ thống này. Hệ thống có thể xử lý hàng PB dữ liệu



với độ trễ thấp, real-time. HBase được thiết kế để có thể truy vấn được các table lớn với tốc độ nhanh.

- **Scalable:** Như đã trình bày ở phần đầu, HBase có thể scale theo chiều ngang (scale-out) bằng cách gắn thêm nhiều node mới, sau đó các Region (nơi lưu trữ các table) tự động chia tách và tạo ra nhiều Region mới, tích hợp vào hệ thống.

## 2. Vì sao cần HBase?

Với sự bùng nổ của Internet những năm gần đây và trong tương lai, đặc biệt trong lĩnh vực như Social network, social media, kinh doanh online,... thì lượng dữ liệu phát sinh hàng ngày, hàng giờ là cực kỳ lớn và ngày càng gia tăng. Để đáp ứng nhu cầu thu thập, lưu trữ, truy xuất và khai thác dữ liệu lớn, các loại database mới ra đời để giải quyết các bài toán, tình huống cụ thể khi thao tác với dữ liệu lớn nói trên với performance tốt, khả năng scale tốt và dễ dàng truy xuất. Trong số đó, HBase là loại column-base database mạnh mẽ và phổ biến trên thế giới.

Các loại ứng dụng có thể dùng HBase:

- Hệ thống audit log
- Tracking user action
- Realtime counters, realtime analytics
- Monitor các hệ thống
- Hệ thống message
- Lưu trữ dữ liệu thu thập từ web
- Lưu trữ dữ liệu sparse (thưa)
- Nhiều người dùng truy cập đồng thời (stream,...)

Các loại ứng dụng không nên dùng HBase:

- Cần đến transaction hoặc các quan hệ, ràng buộc chặt chẽ
- Cần join dữ liệu
- Dữ liệu quy mô nhỏ

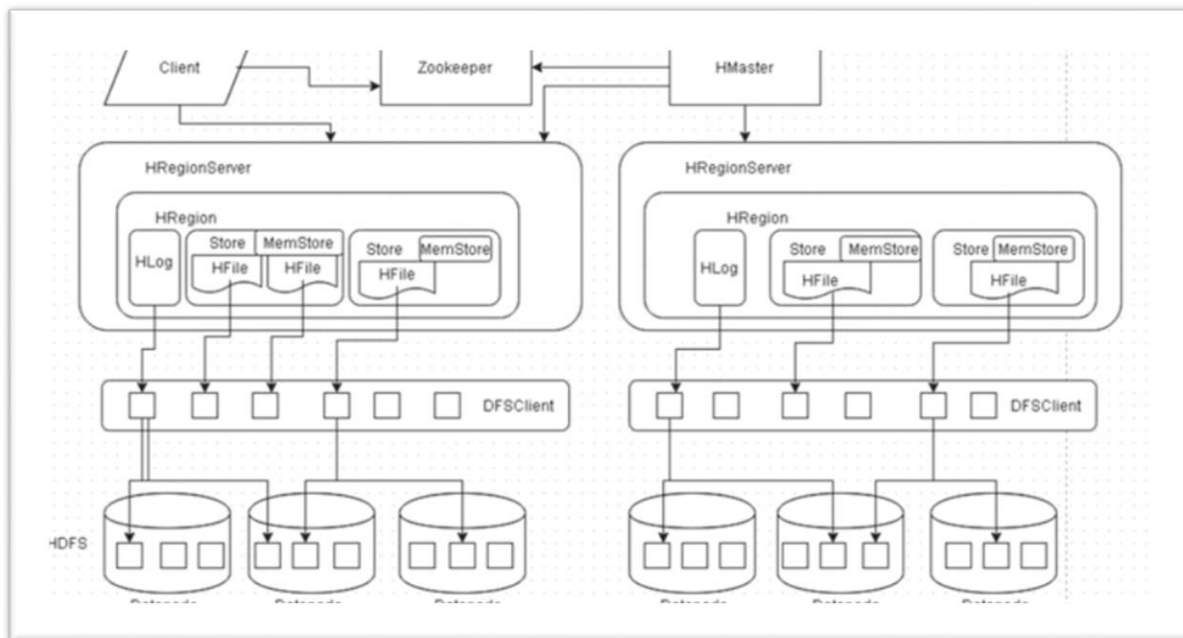
### 3. HBase vs RDBMS

Sau đây là bảng so sánh một số đặc trưng, tính chất cơ bản của HBase và RDBMS

Feature	RDBMS	HBase
Data volume	TB of data	PB of data
Primary query language	SQL	Get, Put, Scan shell
Data object	Table	Table
Relational	Yes	No
Join	Yes	No
Transactions	Supported	Not supported
Indexes	Primary, secondary, B-Tree, Clustered	Secondary indexes
Schema	Fixed schema	Schema-less
Storage model	Table spaces	StoreFiles (HFiles) in HDFS
Oriented	Row-oriented	Column-oriented
Caching	Standard data/metadata cache with query cache	In-memory caching
Architecture	Monolithic	Distributed
Fault tolerant	Some case	Highly fault tolerant
Scalability	Hard	Highly horizontal scalability
Read/write throughput	~1.000s/second	~1.000.000s/second
Write performance	Does not scale well	Scales linearly
Single point of failure	Yes	No
Sharding	Limited support. Manual server sharding. Table partitioning	Auto-sharding

#### 4. HBase Data Model

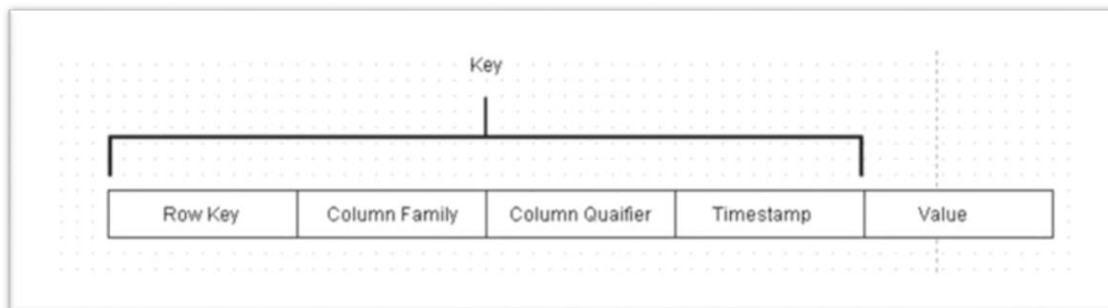
Mô hình lưu trữ dữ liệu của HBase được mô tả như lược đồ sau đây



HBase sử dụng 2 định dạng file chính là HLog và HFile, được vào các HDFS Datanode thông qua DFSCient. Điều này giúp cho HBase có thể tập trung vào việc tối ưu truy vấn và cập nhật dữ liệu, vốn không phải thế mạnh của HDFS nguyên thủy. Tập hợp một số file như trên được quản lý bởi một Region (trình bày ở phần sau), thường được sao lưu thành 3 bản lưu ở 3 datanode khác nhau. Cụ thể:

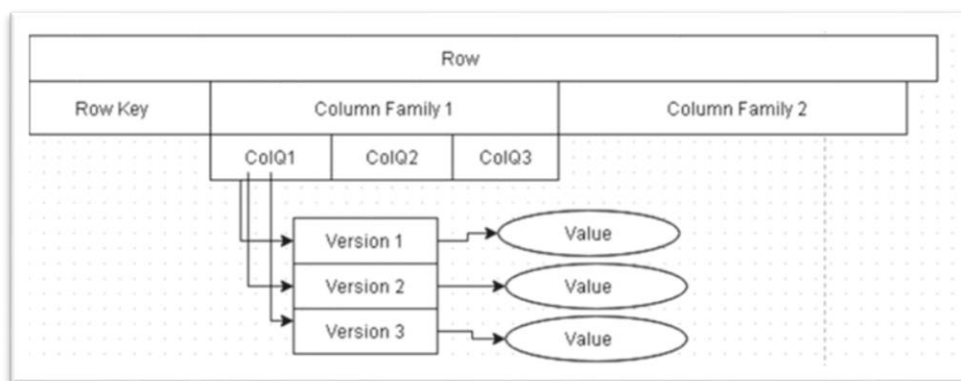
- Column family & Column & Row:** Mỗi file HFile riêng biệt sẽ chứa thông tin của một (và chỉ một) Column family. Mỗi column family bao gồm nhiều column. Giá trị cần lưu trữ trong bảng là giao điểm giữa một column và một row, được xác định bằng row key. Ở trong một table có 2 loại index theo thứ tự từ điển, index chính sẽ theo thứ tự các row key, còn index phụ sẽ index các column (theo mỗi row). Table trong HBase thường khá thưa, bởi vì mỗi row không nhất thiết phải có tất cả các column family của table đó.

- **Region:** Region là các mảnh của một table hoàn chỉnh. Tập hợp một số region sẽ được quản lý bởi một HRegionServer.
- **Key-Value:** Cấu tạo một Key được mô tả như sau:



Key của một phần tử đơn vị lưu trữ bao gồm 4 thành phần nối đuôi nhau:

- Row Key: Id của row cần lưu trữ, được sắp xếp theo thứ tự từ điển
- Column Family
- Column Qualifier
- Timestamp: Version của value, mỗi khi thêm mới hoặc cập nhật thì version mới nhất sẽ nằm trên cùng. Cụ thể như hình sau đây, một value được cập nhật 3 lần nên có 3 Version khác nhau



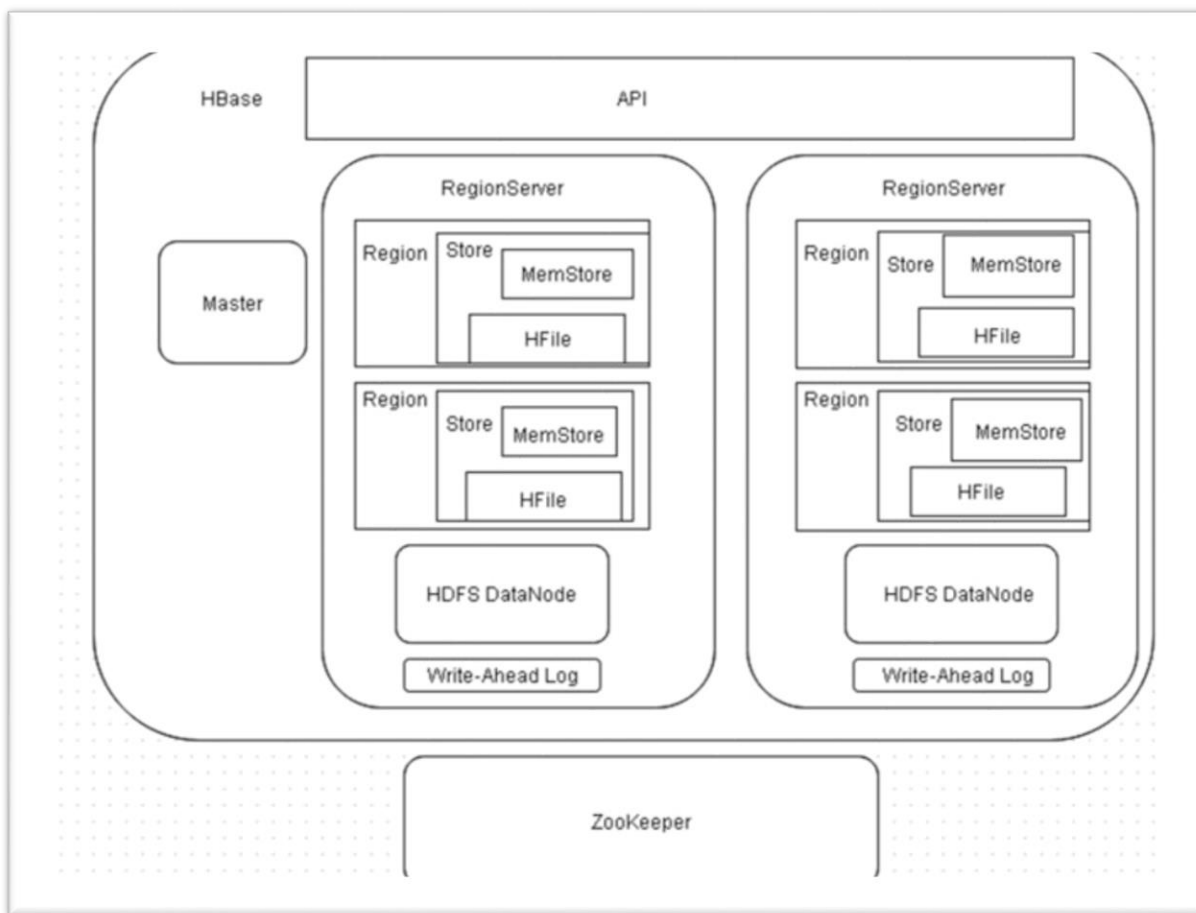
- **Row-version:** Mỗi một ô trong table khi được thêm mới hoặc cập nhật thì đều phát sinh ra một version (timestamp). Ví dụ hình trên ColQ1 được cập nhật 3 version, sắp xếp từ mới tới cũ. Hình sau đây mô tả cách thức lưu trữ vật lý trong một HFile

```
123 cf1 col1 val1 @ ts1
123 cf1 col2 val2 @ ts1
235 cf1 col1 val3 @ ts1
235 cf1 col2 val4 @ ts1
235 cf1 col2 val5 @ ts2
```

Có thể thấy HBase về bản chất lưu trữ cuối cùng ở tầng hệ thống là Key-Value

## 5. Kiến trúc Hbase

Chương này chủ yếu tìm hiểu kiến trúc HBase ở mode distributed

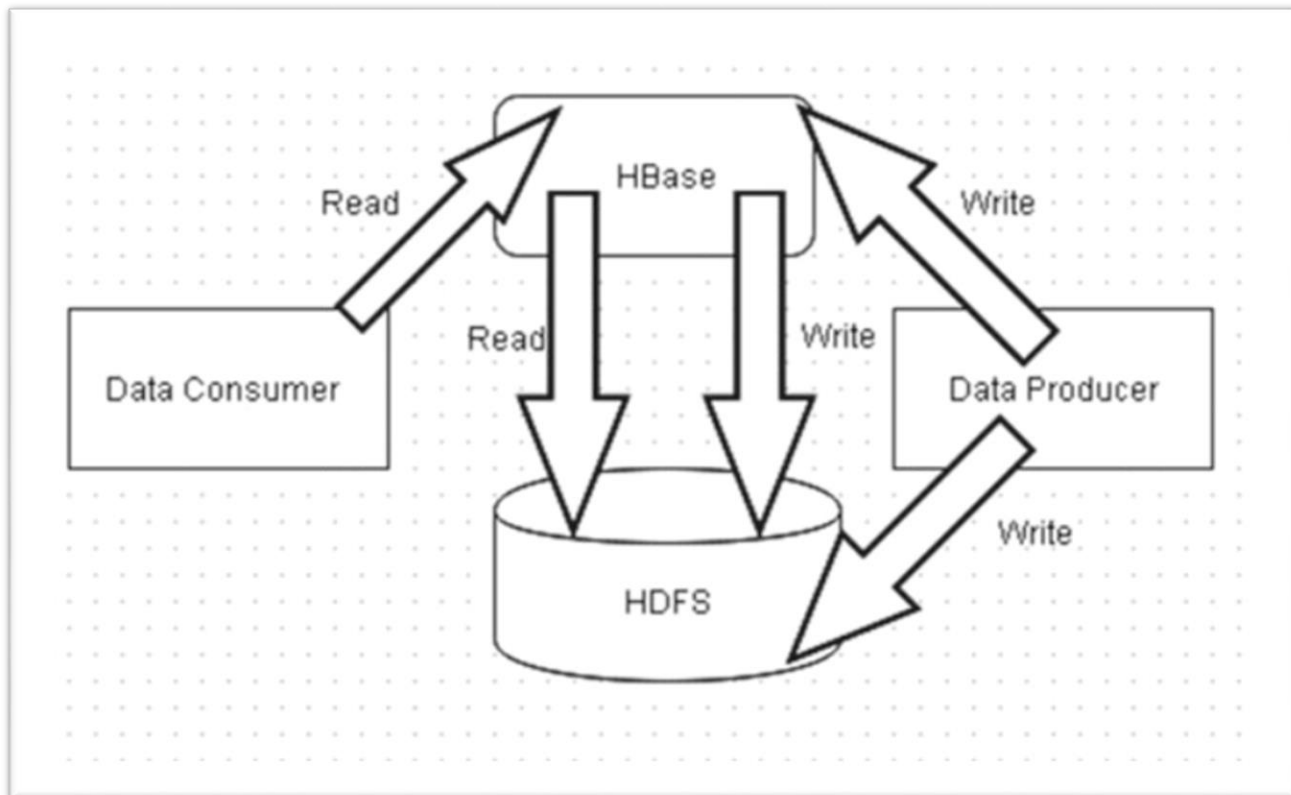


Hình trên đây là mô tả chi tiết kiến trúc cơ bản của một Hbase Cluster, bao gồm 3 thành phần chính sau:

- **Master:** Còn gọi là HMaster, chạy trên Namenode (của HDFS). Process này có nhiệm vụ quản lý toàn bộ cluster, bao gồm các công việc như:
  - Chỉ định region nào được lưu vào RegionServer nào lúc khởi tạo, hoặc lúc bị lỗi.
  - Load balancing
  - Monitor các ResgionServer
  - Quản lý các thay đổi trong metadata
- **RegionServers:** Là nơi tiếp nhận lệnh từ Master, phụ trách quản lý data. Trong đó, mỗi RegionServer lưu trữ một số lượng Region được giao cho. Mỗi phân đoạn chứa số lượng row nhất định của một table sẽ được lưu ở các RegionServer khác nhau. RegionServer bao gồm các thành phần chủ yếu sau:
  - **Region(s):** Là phân mảnh của một table cần được lưu trữ. Một region sẽ giữ một lượng row (sắp xếp theo đúng thứ tự được index) của table tương ứng. Table càng thêm nhiều row thì số region tách ra càng nhiều, và được sao lưu để phân tán qua các RegionServer khác.
  - **Write-Ahead Log (WAL):** Là nơi đầu tiên ghi dấu lại mọi cập nhật vào file HLog, trước khi cập nhật đó đi tới Memstore và tới HFile. Mục đích lưu dấu này là để tái hiện lại cập nhật trong trường hợp RegionServer bị lỗi.
  - **Store:** Gồm 2 thành phần là Memstore (lưu data trên memory) và HFile (lưu data trên file vật lý). Khi Memstore bị đầy hoặc đến ngưỡng nhất định, data sẽ được flush xuống HFile.
- **Zookeeper:** Là đơn vị quản lý vận hành của toàn bộ kiến trúc trên. Một số công việc cụ thể như:
  - Thông báo đến các đơn vị khác trạng thái hiện tại của Master
  - Lưu trữ metadata của Master, recover lại Master trong trường hợp lỗi
  - Lưu trữ danh sách tất cả các region của hệ thống

- Lưu giữ và cung cấp các file metadata giúp HregionServer định hướng được dữ liệu

### Đường đi của data:



Hình trên đây minh họa đường đi của dữ liệu trong HBase:

- **Read:** Client read data từ HBase <- HBase lấy data từ HDFS
- **Write:** Client write data vào HBase -> HBase write data vào HDFS. Bên cạnh đó, client cũng có option write data trực tiếp vào HDFS
- Quá trình giao tiếp giữa HBase với HDFS được thông qua các đối tượng HDFS Client

Ngoài ra còn một số cơ chế Block-Cache hỗ trợ truy vấn, cơ chế chia tách table thành các region (Splitting) và gộp các region (Compaction) chưa được trình bày sâu trong báo cáo này.



## 6. Thực hành cài đặt Hbase

Hướng dẫn cài đặt chi tiết các bước cài đặt cần thiết cho từng thành phần của một HBase Cluster được mô tả cụ thể trong link các link tham khảo sau:

- <http://hbase.apache.org/book.html>
- <https://computingforgeeks.com/how-to-install-apache-hadoop-hbase-on-ubuntu>

Ở quy mô của báo cáo này sẽ hướng dẫn các bước cài đặt HBase ở mode *pseudo-distributed*. Cụ thể từng bước như sau:

**Cài đặt Java:** Tham khảo các bước trong link <https://ubuntu.com/tutorials/install-jre#3-installing-oracle-jre>

- Cài đặt SSH-server, mặc định port của ssh là 22. Sau khi cài đặt và enable SSH, ta cần add ssh-key của client vào server

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys  
chmod 0600 ~/.ssh/authorized_keys
```

### Cài đặt Hadoop:

- Tải bản Hadoop version stable theo link <https://hadoop.apache.org/releases.html>, trong hướng dẫn này sẽ sử dụng version 2.10.1. Sau đó giải nén và move vào /usr/local/hadoop

```
tar -xzf hadoop-$RELEASE.tar.gz
```

```
rm hadoop-$RELEASE.tar.gz  
sudo mv hadoop-$RELEASE/ /usr/local/hadoop
```

- Tạo và cập nhật các biến môi trường cho Hadoop:



```
$ sudo vim tee /etc/profile.d/hadoop_java.sh
export JAVA_HOME=$(dirname $(dirname $(readlink $(readlink $(which javac)))))
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

source /etc/profile.d/hadoop_java.sh
```

- Các file config quan trọng trong `"/usr/local/hadoop/etc/hadoop/":`
- **core-site.xml**: file này chứa các cấu hình cơ bản để khởi động Hadoop cluster, ta gán cơ bản như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the license.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the license is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

- **hdfs-site.xml**: chứa cấu hình thư mục lưu trữ các tập tin của HDFS



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.name.dir</name>
    <value>file:///hadoop/hdfs/namenode</value>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>file:///hadoop/hdfs/datanode</value>
  </property>
</configuration>
~
~
~
```

- mapred-site.xml: Quy định framework được sử dụng cho MapReduce - mô hình tính toán của Hadoop:

```
<configuration>
$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [hbase]
hbase: Warning: Permanently added 'hbase' (ECDSA) to the list of known hosts.
</property>
</configuration>
```

- Cuối cùng khởi chạy Hadoop:
- Ta truy cập vào <http://0.0.0.0:50070> để xem màn hình quản lý Datanode của Hadoop

The screenshot shows the Hadoop Overview page for 'localhost:9000' (active). The page has a navigation bar with links: Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, and Startup Progress. The Overview section displays a table with the following information:

Started:	Wed Mar 03 21:49:41 +0700 2021
Version:	2.10.1, r1827467c9a56f133025f28557bfc2c562d78e816
Compiled:	Mon Sep 14 20:17:00 +0700 2020 by centos from branch-2.10.1



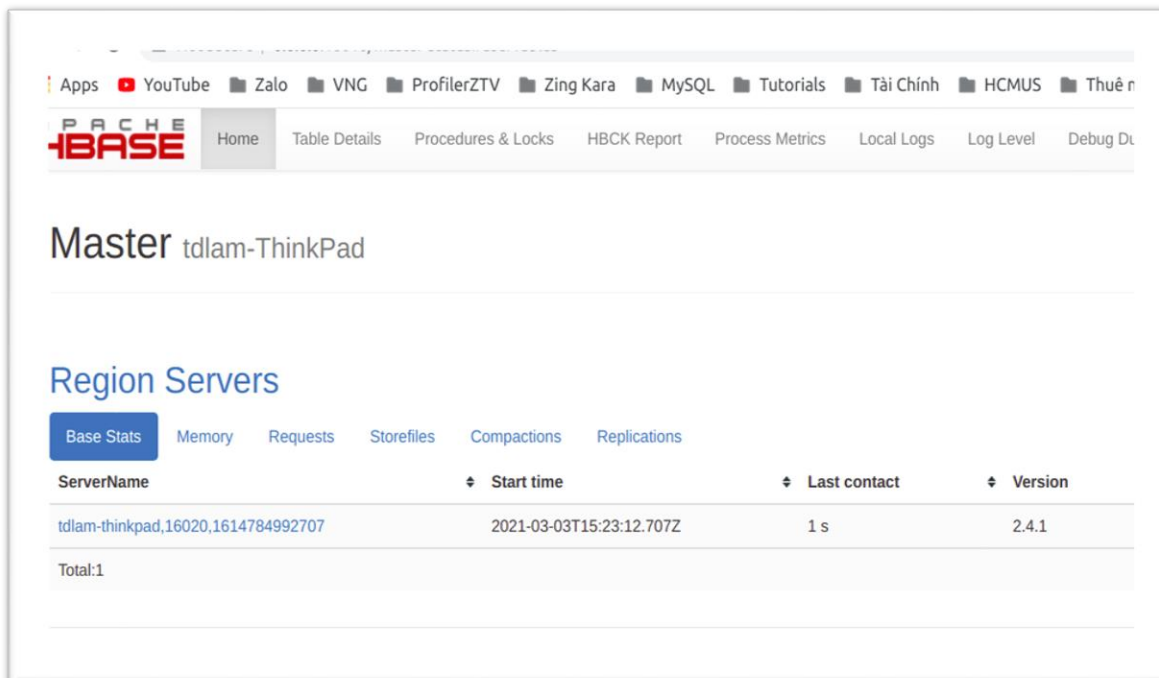
## Cài đặt HBase cluster

- Tải HBase: <https://downloads.apache.org/hbase/>
- Giải nén vào thư mục /opt/hbase
- Thêm các biến môi trường vào tập tin ~/.bash\_profile với nội dung sau:
  - export HBASE\_HOME="/opt/hbase"
  - export PATH="\$HBASE\_HOME/bin:\$PATH"
- Sửa nội dung tập tin /opt/hbase/conf/hbase-env.sh với nội dung sau:
  - export JAVA\_HOME=/usr/java/default
  - export HBASE\_MANAGES\_ZK=true
  - export HBASE\_PID\_DIR=/opt/hbase/var
- Cấu hình trong tập tin /opt/hbase/conf/hbase-site.xml với các thông tin quan trọng:

```
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://localhost:9000/hbase2</value>
</property>
<property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/hadoop/zookeeper</value>
</property>

<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
```

- Khởi chạy HBase bằng lệnh /opt/hbase/bin/start-hbase.sh



Như vậy ta đã cài đặt xong 2 thành phần quan trọng là HBase Cluster và HDFS kèm theo.

## 7. Các thao tác & công cụ cơ bản

- HBase shell là console hỗ trợ giao tiếp trực tiếp với HBase server ở local thông qua câu lệnh
- HBase Java API là Client hỗ trợ thao tác với HBase server thông qua các interface ví dụ như:
  - **org.apache.hadoop.hbase.client.Admin**: Class này dùng làm interface để quản lý các Table của HBase, và làm một số tác vụ quản trị khác như thêm, xóa, tra cứu, bật/tắt các Table; thêm/cập nhật các Column family của table. Code ví dụ được trình bày trong file UsingHBaseAdmin.java trong mã nguồn đính kèm báo cáo này.
  - **org.apache.hadoop.hbase.client.Table**: Là interface giao tiếp với một Table trong HBase. Ví dụ cụ thể mô tả trong file UsingHTable.java

## KẾT LUẬN VÀ MỞ RỘNG

### 1. So sánh HBase và Cassandra

#### 1.1 Giống nhau

##### ➤ Cơ sở dữ liệu(Database):

- Cả hai đều là cơ sở dữ liệu mã nguồn mở.
- Có thể xử lý được dữ liệu lớn, dữ liệu không quan hệ(bao gồm image, audio, video..)

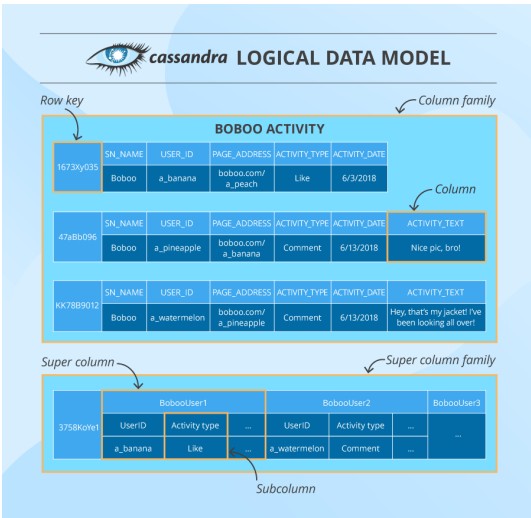
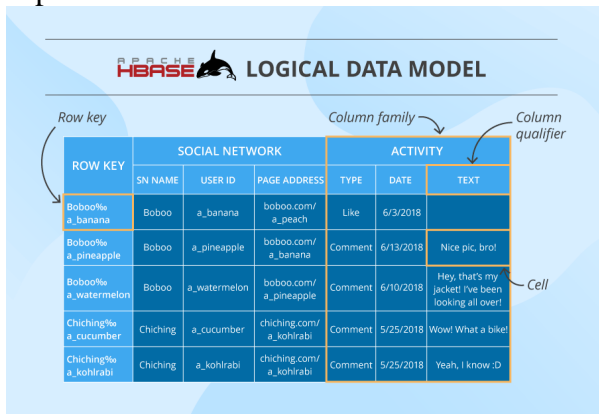
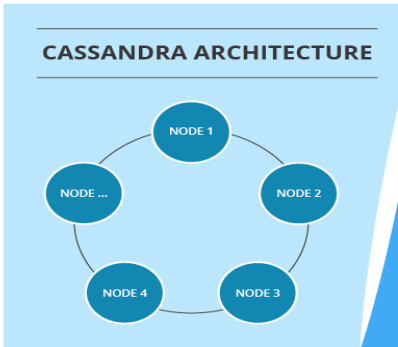
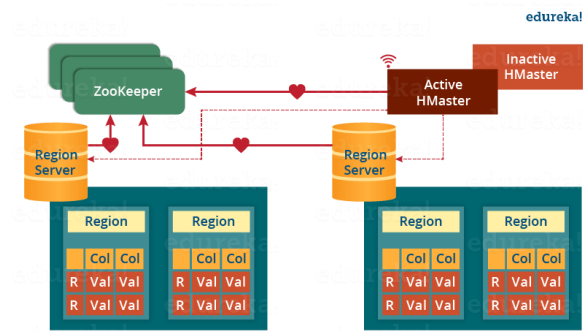
##### ➤ Khả năng mở rộng(Scalability)

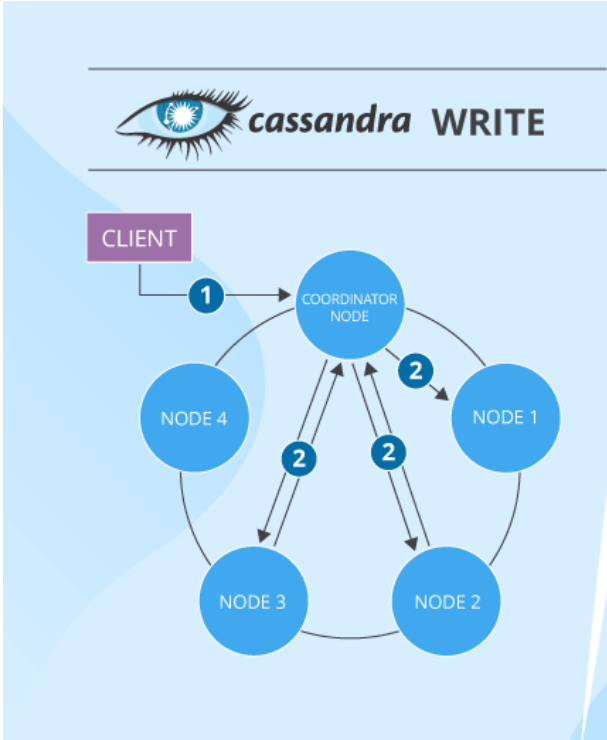
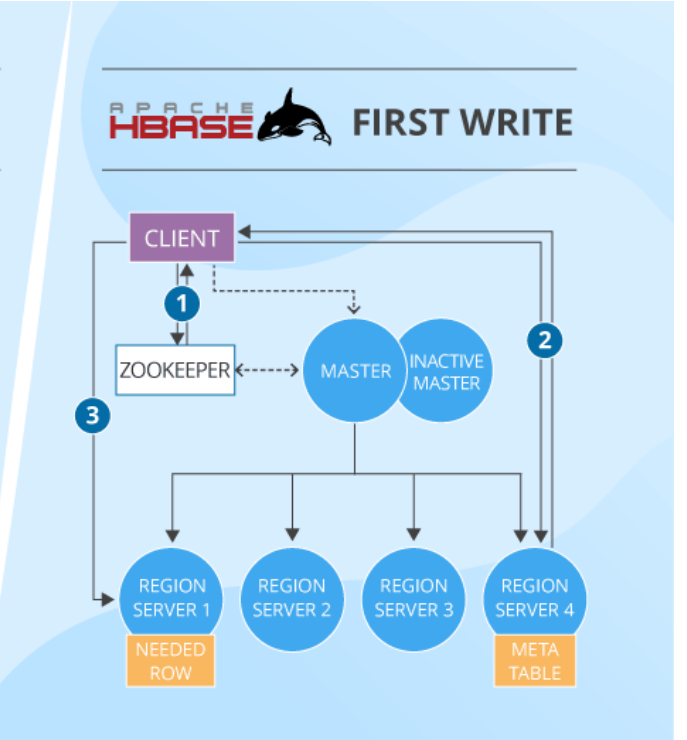
- Cả hai đều có khả năng mở rộng cao.
- Để mở rộng chỉ cần tăng số lượng node trên cluster.

##### ➤ Tạo bản sao(Replication)

- Data khi được lưu xuống node sẽ tạo bản sao ở một số node khác, nên khi xảy ra lỗi vẫn tồn tại data ở node backup để truy xuất.

#### 1.2 Khác nhau

	Cassandra	Hbase
Data Model	<p>- Column is like cell in Hbase. - Column Family gần giống với table in HBase</p> 	<p>- HBase column qualifier gần tương đối giống với super column của Cassandra.</p> 
Architecture	<p>Masterless one</p> 	<p>Master-based architecture</p>  <p>Figure: ZooKeeper as Coordination Service</p>

Performance	<p><b>Read</b></p> <ul style="list-style-type: none"> <li>- 129,000 reads/second</li> </ul> <p><b>Write</b></p> <ul style="list-style-type: none"> <li>- 326,500 operations per second</li> </ul>	<p><b>Read:</b></p> <ul style="list-style-type: none"> <li>- 8,000 reads/second in 32 nodes</li> </ul> <p><b>Write</b></p> <ul style="list-style-type: none"> <li>- 297,000 operations per second</li> </ul>
Data write flow	<ol style="list-style-type: none"> <li>1. Client connect and write to node</li> <li>2. Replicate data</li> </ol>	<ol style="list-style-type: none"> <li>1. Client connect to zookeeper lấy thông tin region server đang chứa meta data</li> <li>2. Get meta data</li> <li>3. Write data</li> </ol>
	<div style="display: flex; justify-content: space-around;"> <div style="width: 45%;">  <p><b>cassandra WRITE</b></p> </div> <div style="width: 45%;">  <p><b>APACHE HBASE FIRST WRITE</b></p> </div> </div>	
Security	- Row level access	- Cell level access
Internode Communication	- Use by Gossip Protocol	- Use by Zookeeper Protocol





## TÀI LIỆU THAM KHẢO

D. Vohra, Apache HBase Primer 2016

<https://www.edureka.co/>

<http://hbase.apache.org/book.html>

<https://blog.acolyer.org>

<https://appinventiv.com/blog/hbase-vs-cassandra>

<https://www.scnsoft.com/blog/cassandra-vs-hbase>

<https://www.edureka.co/blog/hbase-architecture/>