

HBase vs. Cassandra: NoSQL Battle!

[Bradford](#)

(Note: Check out the [Drawn to Scale platform](#). Store, query, search, process, and serve **all** your data. To **all** your users. In real time).

Distributed, scalable databases are desperately needed these days. From building massive data warehouses at a social media startup, to protein folding analysis at a biotech company, “Big Data” is becoming more important every day. While Hadoop has emerged as the *de facto* standard for handling big data problems, there are still quite a few distributed databases out there and each has their unique strengths.

Two databases have garnered the most attention: HBase and Cassandra. The split between these equally ambitious projects can be categorized into Features (things missing that could be added any at time), and Architecture (fundamental differences that can’t be coded away). HBase is a near-clone of Google’s BigTable, whereas Cassandra purports to being a “BigTable/Dynamo hybrid”.

In my opinion, while Cassandra’s “writes-never-fail” emphasis has its advantages, HBase is the more robust database for a majority of use-cases. Cassandra relies mostly on Key-Value pairs for storage, with a table-like structure added to make more robust data structures possible. And it’s a fact that far more people are using HBase than Cassandra at this moment, despite both being similarly recent.

Let’s explore the differences between the two in more detail...

CAP and You

[This article at Streamy](#) explains CAP theorem (Consistency, Availability, Partitioning) and how the BigTable-derived HBase and the Dynamo-derived Cassandra differ.

Before we go any further, let’s break it down as simply as possible:

- Consistency: “*Is the data I’m looking at now the same if I look at it somewhere else?*”
- Availability: “*What happens if my database goes down?*”
- Partitioning: “*What if my data is on different networks?*”

CAP posits that distributed systems have to compromise on each, and HBase values strong consistency and High Availability while Cassandra values Availability and Partitioning tolerance. Replication is one way of dealing with some of the design tradeoffs. HBase does not have replication yet, but that’s about to change — and Cassandra’s replication comes with some caveats and penalties.

Let’s go over some comparisons between these two datastores:

Feature Comparisons

Processing

HBase is part of the Hadoop ecosystems, so many useful distributed processing frameworks support it: Pig, Cascading, Hive, etc. This makes it easy to do complex data analytics without resorting to hand-coding. Efficiently running MapReduce on Cassandra, on the other hand, is difficult because all of its keys are in one big “space”, so the MapReduce framework doesn’t know how to split and divide the data natively. There needs to be some hackery in place to handle all of that.

In fact, here’s some code from a Cassandra/Hadoop Integration patch:

```
+ /*  
  
+ FIXME This is basically a huge kludge because we needed access to  
  
+ cassandra internals, and needed access to hadoop internals and so we  
  
+ have to boot cassandra when we run hadoop. This is all pretty  
  
+ fucking awful.  
  
+  
  
+ P.S. it does not boot the thrift interface.  
  
+ */
```

This gives me The Fear.

Bottom line? Cassandra may be useful for storage, but not any data processing. HBase is much handier for that.

Installation & Ease of Use

Cassandra is only a Ruby gem install away. That's pretty impressive. You still have to do quite a bit of manual configuration, however. HBase is a .tar (or packaged by Cloudera) that you need to install and setup on your own. HBase has thorough documentation, though, making the process a little more straightforward than it could've been.

HBase ships with a very nice Ruby shell that makes it easy to create and modify databases, set and retrieve data, and so on. We use it constantly to test our code. Cassandra does not have a shell at all — just a basic API. HBase also has a nice web-based UI that you can use to view cluster status, determine which nodes store various data, and do some other basic operations. Cassandra lacks this web UI as well as a shell, making it harder to operate. (*ed: Apparently, there is now a shell and pretty basic UI — I just couldn't find 'em*).

Overall Cassandra wins on installation, but lags on usability.

Architecture

The fundamental divergence of ideas and architecture behind Cassandra and HBase drives much of the controversy over which is better.

Off the bat, Cassandra claims that “writes *never* fail”, whereas in HBase, if a region server is down, writes will be blocked for affected data until the data is redistributed. This rarely happens in practice, of course, but will happen in a large enough cluster. In addition, HBase has a single point-of-failure (the Hadoop NameNode), but that will be less of an issue as Hadoop evolves. HBase does have row locking, however, which Cassandra does not.

Apps usually rely on data being accurate and unchanged from the time of access, so the idea of eventual consistency can be a problem. Cassandra, however, has an internal method of resolving up-to-dateness issues with vector clocks — a complex but workable solution where basically the latest timestamp wins. The HBase/BigTable puts the impetus of resolving any consistency conflicts on the application, as everything is stored versioned by timestamp.

Another architectural quibble is that Cassandra only supports one table per install. That means you can't denormalize and duplicate your data to make it more usable in analytical scenarios. (*edit: this was corrected in the latest release*) Cassandra is really more of a Key Value store than a Data

Warehouse. Furthermore, schema changes require a cluster restart(!). Here's what the [Cassandra JIRA](#) says to do for a schema change:

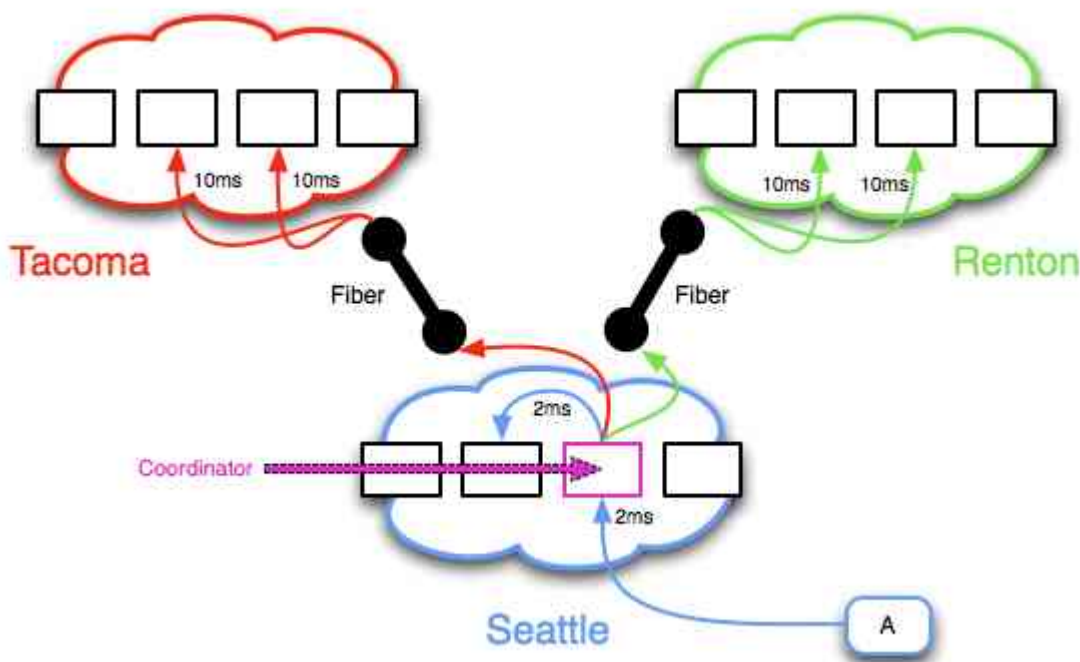
1. Kill Cassandra
2. Start it again and wait for log replay to finish
3. Kill Cassandra AGAIN
4. Make your edits (now there is no data in the commitlog)
5. Manually remove the sstable files (-Data.db, -Index.db, and -Filter.db) for the CFs you removed, and rename files for CFs you renamed
6. Start Cassandra and your edits should take effect

With the lack of timestamp versioning, eventual consistency, no regions (making things like MapReduce difficult), and only one table per install, it's difficult to claim that Cassandra implements the BigTable model.

Replication

Cassandra is optimized for small datacenters (hundreds of nodes) connected by very fast fiber. It's part of Dynamo's legacy from Amazon. HBase, being based on research originally published by Google, is happy to handle replication to thousands of planet-strewn nodes across the 'slow', unpredictable Internet.

A major difference between the two projects is their approach to replication and multiple datacenters. Cassandra uses a P2P sharing model, whereas HBase (the upcoming version) employs more of a data+logs backup method, aka 'log shipping'. Each has a certain elegance. Rather than explain this in words, here comes the drawings:



This first diagram is a model of the Cassandra replication scheme.

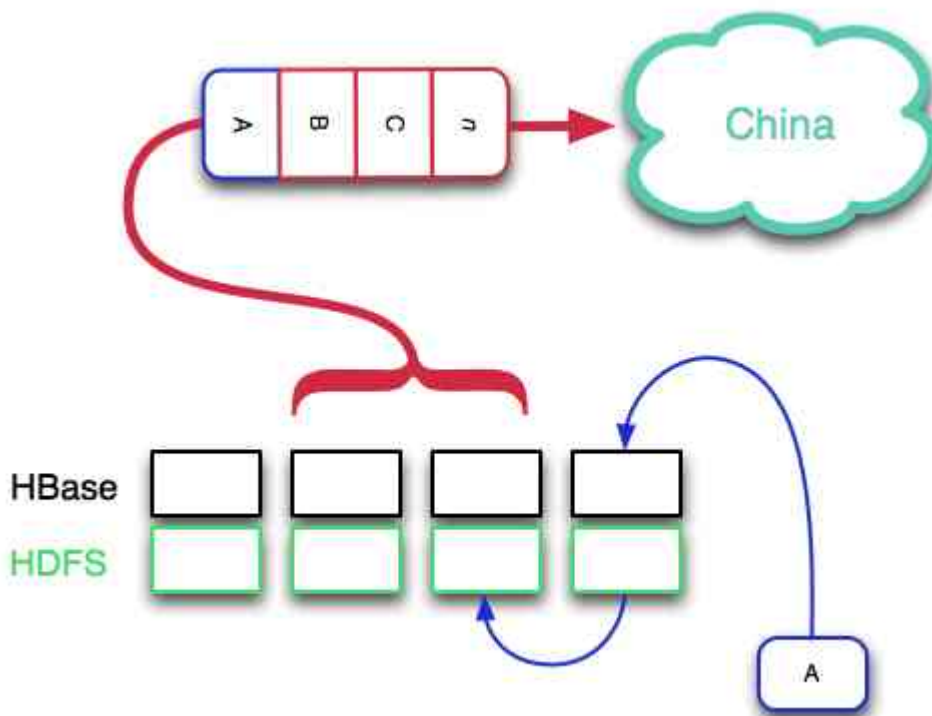
1. The value is written to the "Coordinator" node
2. A duplicate value is written to another node in the same cluster
3. A third and fourth value are written from the Coordinator to another cluster across the high-speed fiber
4. A fifth and sixth value are written from the Coordinator to a third cluster across the fiber
5. Any conflicts are resolved in the cluster by examining timestamps and determining the

“best” value.

The major problem with this scheme is that there is no real-world auditability. The nodes are eventually consistent — if a datacenter (“DC”) fails, it’s impossible to tell when the required number of replicas will be up-to-date. This can be extremely painful in a live situation — when one of your DCs goes down, you often want to know *exactly* when to expect data consistency so that recovery operations can go ahead smoothly.

It’s important to note that Cassandra relies on high-speed fiber between datacenters. If your writes are taking 1 or 2 ms, that’s fine. But when a DC goes out and you have to revert to a secondary one in China instead of 20 miles away, the incredible latency will lead to write timeouts and highly inconsistent data.

Let’s take a look at the HBase replication model (note: this is coming in the .21 release):



What’s going on here:

1. The data is written to the HBase write-ahead-log in RAM, then it is then flushed to disk
2. The file on disk is automatically replicated due to the Hadoop Filesystem’s nature
3. The data enters a “Replication Log”, where it is piped to another Data Center.

With HBase/Hadoop’s deliberate sequence of events, consistency within a datacenter is high. There is usually only one piece of data around the same time period. If there are not, then HBase’s timestamps allow your code to figure out which version is the “correct” one, instead of it being chosen by the cluster. Due to the nature of the Replication Log, one can always tell the state of the data consistency at any time — a valuable tool to have when another data center goes down. In addition, using this structure makes it easy to recover from high-latency scenarios that can occur with inter-continental data transfer.

Knowing Which To Choose

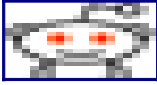






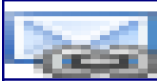


The business context of Amazon and Google explains the emphasis on different functionality between Cassandra and HBase.

Cassandra expects High Speed Network Links between data centers. This is an artifact of Amazon’s

Dynamo: Amazon datacenters were historically located very close to each other (dozens of miles apart) with very fast fiber optic cables between them. Google, however, had transcontinental datacenters which were connected only by the standard Internet, which means they needed a more reliable replication mechanism than the P2P eventual consistency.

If you need highly available writes with only eventual consistency, then Cassandra is a viable candidate for now. However, many apps are not happy with eventual consistency, and it is still lacking many features. Furthermore, even if writes do not fail, there is still cluster downtime associated with even minor schema changes. HBase is more focused on reads, but can handle very high read and write throughput. It's much more Data Warehouse ready, in addition to serving millions of requests per second. The HBase integration with MapReduce makes it valuable, and versatile.

Share and Enjoy:

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 







This entry was posted on Thursday, October 29th, 2009 at 12:42 pm and is filed under Scalability. You can follow any responses to this entry through the [RSS 2.0](#) feed. You can [leave a response](#), or [trackback](#) from your own site.

26 Responses to “HBase vs. Cassandra: NoSQL Battle!”

- [uberVU - social comments](#) Says:
[October 29th, 2009 at 1:06 pm](#)

Social comments and analytics for this post...

This post was mentioned on Twitter by LusciousPear: NEW ARTICLE! #HBase vs. #Cassandra : #NoSQL Battle. <http://bit.ly/25Td9z> #hadoop #nosqleast...

-  [chetan conikee](#) Says:
[October 29th, 2009 at 1:11 pm](#)
I cannot view the images on this page.
Is it just me ?
-  [Bradford](#) Says:
[October 29th, 2009 at 1:15 pm](#)
Hey there,
The images are working for me. Thanks for the notice, though! What browser are you on?
- [Tweets that mention HBase vs. Cassandra: NoSQL Battle! | Road to Failure -- Topsy.com](#)
Says:
[October 29th, 2009 at 1:30 pm](#)
[...] This post was mentioned on Twitter by Ali Sohani, BradfordS. BradfordS said: New article, can't remember if the correct hashtag is #cassandra db <http://bit.ly/25Td9z> :) [...]
-  [Michael Greene](#) Says:
[October 29th, 2009 at 2:00 pm](#)
Cassandra supports multiple tables, as even a cursory search reveals. I have to suspect that any article suggesting it does not was not researched very well. In the same section, “no regions” goes unexplained as a drawback of Cassandra.
I think you are being highly disingenuous with lines like \Cassandra is really more of a Key Value store than a Data Warehouse\ given your experience with HBase and BigTable. If a data store supports range queries and schemaless columns with fat tables, wide tables, and tall tables, it is not a key value store, at least not as typically considered.
You are right to point out that the prime differences are the data distribution model and how this pertains to the CAP theorem.
-  [Bradford](#) Says:
[October 29th, 2009 at 2:05 pm](#)
Michael: Thanks for the feedback. When I tried Cassandra, it did not support multiple tables. Has that changed recently?
As far as it not being a data warehouse, it's much less useful out of the box than HBase — it doesn't integrate with anything that can be used for analytics. Thanks for pointing out the “no regions”, I think I forgot a sentence there in editing. I'll fix that.
-  [ian](#) Says:
[October 29th, 2009 at 2:13 pm](#)
nice comparison and explanation of the differences between the two — good job!
-  [Igor](#) Says:
[October 29th, 2009 at 2:14 pm](#)
There are a few nasty things about Cassandra that I've discovered:
#0

A client connects to a nearest node, which address it should know beforehand, all communications with all other Cassandra nodes proxied through it.

- a. read/write traffic is not evenly distributed among nodes – some nodes proxy more data than they host themselves
- b. Should the node go down, the client is helpless, can't read, can't write anywhere in the cluster.

#1

Although Cassandra claims that “writes never fail” they do fail, at least at the moment of speaking they do. Should the target data node become sluggish, request times out and write fails. There are many reason for a node to become unresponsive: garbage collector kicks in, compaction process, whatever...

In all such cases all write/read request fail. In a conventional database these requests would have become proportionally slow, but in Cassandra they just fail.

#2

There is multi-get but there is no multi-delete and one can't truncate ColumnFamily either

#3

Should a new, empty data node enter the cluster, shares of data from the two neighbor nodes on the key-ring will be transfered only. This leads to uneven data distribution and uneven load. One should also keep track on tokens manually and select them wisely.



- [Michael Greene](#) Says:
[October 29th, 2009 at 2:15 pm](#)

Cassandra has supported multiple tables since June. The first official Apache release including this was in September, but was long recommended in IRC and the mailing lists. Given that it's now the end of October, I would have expected you to test based on 0.4, 0.4.1, or 0.5 all of which include this functionality.

I agree with Cassandra's current shortcomings regarding analytics, and enjoyed your previous article in August about the potential for an analytics DB based on HBase, but I think there are many statements in this article that are either wrong, like the above, or at least antagonistic.

- [Twitted by rgaidot](#) Says:
[October 29th, 2009 at 2:29 pm](#)

[...] This post was Twitted by rgaidot [...]

- [Twitted by andreisavu](#) Says:
[October 29th, 2009 at 4:51 pm](#)

[...] This post was Twitted by andreisavu [...]

- [Twitted by ungerik](#) Says:
[October 29th, 2009 at 10:01 pm](#)

[...] This post was Twitted by ungerik [...]

- [Twitted by tkang1](#) Says:
[October 30th, 2009 at 12:57 am](#)

[...] This post was Twitted by tkang1 [...]

- [Twitted by pogrebnyak](#) Says:
[October 30th, 2009 at 4:09 am](#)

[...] This post was Twitted by pogrebnyak [...]

- [*Twitted by abrdev*](#) Says:
[October 30th, 2009 at 1:53 pm](#)

[...] This post was Twitted by abrdev [...]
-  [*schubert zhang*](#) Says:
[November 6th, 2009 at 9:27 am](#)

To compare two different architectures, I think it should be objectively and theoretically. Some temporal un-ready features should not be the subjects. Such as the “multi-table” feature, it seems not difficult to be implemented.

It is very interesting that many people doubt Dynamo. Even guys from Facebook:
<http://jsensarma.com/blog/2009/11/dynamo-a-flawed-architecture-part-i/>

I think each architecture is designed for its usage.
-  [*Bradford*](#) Says:
[December 23rd, 2009 at 10:21 pm](#)

Thanks for the feedback. I'm going to do another end-to-end test with the next release of both products~
- [*HBase vs. Cassandra: NoSQL Battle! | Road to Failure « The other side of the firewall*](#) Says:
[February 24th, 2010 at 4:04 pm](#)

[...] February 24, 2010 at 6:03 pm · Filed under Cloud computing · Tagged Databases
 [From HBase vs. Cassandra: NoSQL Battle! | Road to Failure] [...]
- [*NoSQL Is Not SQL And That's A Problem | CloudAve*](#) Says:
[March 5th, 2010 at 7:37 pm](#)

[...] data persistence and access solutions that challenges the long lasting legacy of RDBMS. Competition between HBase and Cassandra is heating up. Amazon now supports a variety of consistency models on EC2. However none of the NoSQL [...]
- [*Configuration management and Chef » zulutime*](#) Says:
[March 21st, 2010 at 1:47 pm](#)

[...] article specifying why you the article is FUD. I have come across this before when reading about cassandra – I later discovered from an expert inside Yahoo! that a lot of the information was [...]
-  [*craig*](#) Says:
[April 1st, 2010 at 4:50 am](#)

Very impressive “note” under the blog post title. I'm curious, what sort of numbers do you have on your ingestion rate, volume/size of data, time until it's searchable, etc?
- [*Soluzioni NoSQL, Apache Cassandra | Webbare*](#) Says:
[April 5th, 2010 at 10:00 am](#)

[...] fra Cassandra e HBase: <http://www.roadtofailure.com/2009/10/29/hbase-vs-cassandra-nosql-battle/> [...]
- [*NoSQL Is Not SQL And That's A Problem « rlblogs.com*](#) Says:
[July 15th, 2010 at 2:45 pm](#)

[...] data persistence and access solutions that challenges the long lasting legacy of RDBMS. Competition between HBase and Cassandra is heating up. Amazon now supports a

variety of consistency models on [...]

- [NoSQL Daily – Sat Nov 13 › PHP App Engine](#) Says:
[November 12th, 2010 at 6:16 pm](#)

[...] HBase vs. Cassandra: NoSQL Battle! | Road to Failure: Scalability, Startups, Computer Science, and o... [...]

- [ehcache.net](#) Says:
[March 11th, 2011 at 10:37 pm](#)

HBase vs. Cassandra: NoSQL Battle!...

Distributed, scalable databases are desperately needed these days. From building massive data warehouses at a social media startup, to protein folding analysis at a biotech company, “Big Data” is becoming more important every day. While Hadoop has emer...

- [Articles I Found Useful, pt 1 « Missional Code](#) Says:
[September 6th, 2011 at 9:42 am](#)

[...] <http://www.roadtofailure.com/2009/10/29/hbase-vs-cassandra-nosql-battle/> A comparison between HBase and Cassandra (both supported by Apache). Definitely seems to be ultimately in favor of HBase, though Cassandra seems better suited in some scenarios. [...]