

Architecture, social context and interesting pull requests/issues

1. Architecture of Cassandra

To recover the as-implemented architecture of Apache Cassandra, what we learned from the class is to:

1. get a class diagram UML;
2. group classes by folders to get a folder level connection map;
3. group and make abstractions out of the folders;
4. reorganize the abstractions to make higher level groups;
5. match the high-level groups with a known architecture.

There are nearly 2000 Java files spreading in 31 packages (not including packages within these "first-level" packages). The class diagram is huge and it is impossible for us to look into each file. So we decide to skip the first step and go from the folder level. Also, to make the grouping and reorganizing steps more efficient, we did some research on:

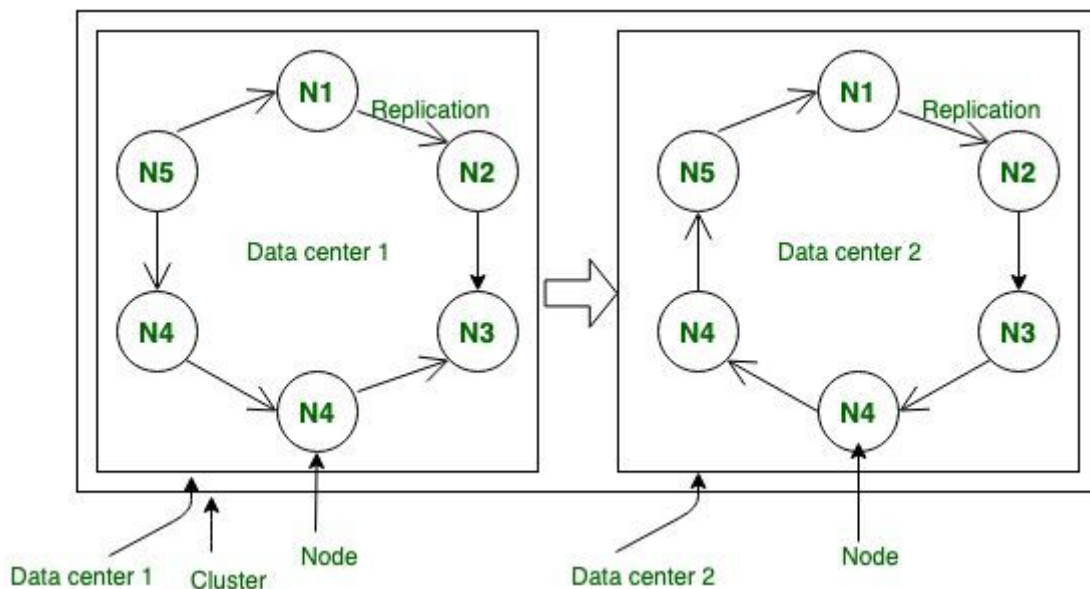
1. key concepts and components in Cassandra. The folder names are short and highly abstract, background information helps us to understand the terms and abbreviations.
2. whether there is a documented Cassandra architecture. If there is one, we can follow the document and focus on the differences instead of building the architecture from the ground up.
3. what are the common architectures for a database management system. No matter how complicated Cassandra is, it is a database management system. We can assume that it shares the common functions and architectures of other systems.

1.1. Preparation

1.1.1. Key concepts and components

Two great explanations about Cassandra concepts and components are from [GeeksforGeeks](#) and [Datastax](#). We summarize their ideas as below.

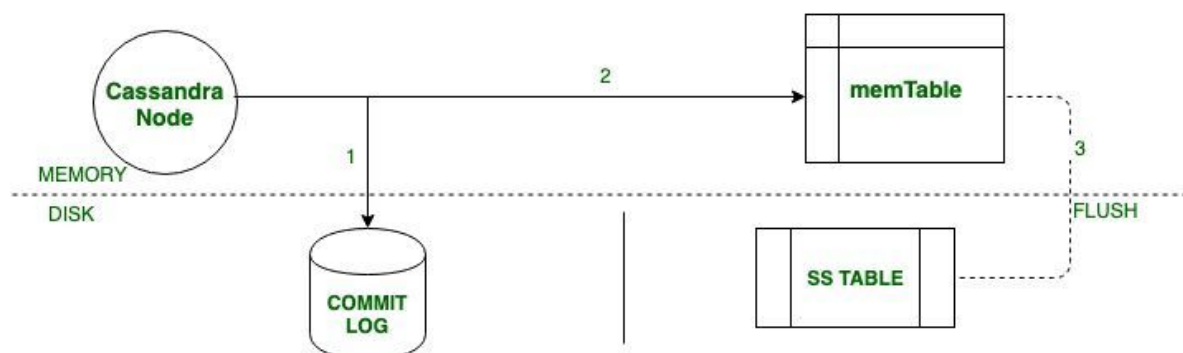
The basic infrastructure component in Cassandra is a **Node**. It is the place where data is stored. Each node frequently exchanges state information about itself and other nodes using peer-to-peer gossip communication protocol. A collection of related nodes is a **Data Center**. A datacenter can be physical or virtual, but datacenters never span physical locations. A collection of data centers (or one single datacenter) is a **Cluster**. A Cassandra cluster is visualized as a ring because it uses a consistent hashing algorithm to distribute data. ¹



(source: <https://www.geeksforgeeks.org/architecture-of-apache-cassandra/>)

The two main operations in Cassandra are **read** operation and **write** operation. Client read or write requests can be sent to any node in the cluster. When a client connects to a node with a request, that node serves as the coordinator for that particular client operation.

There are three storage engines: **Commit Log**, **Mem-table**, and **SSTable**. As soon as the request is received, it is first dumped into commit log to make sure that data is saved. Commit log is used to serve sync issues if a data node is down. After data written in commit log, data is also inserted into Mem-table that holds the data temporarily till it's full. If Mem-table reaches its threshold then data is flushed to SS Table.



(source: <https://www.geeksforgeeks.org/architecture-of-apache-cassandra/>)

1.1.2. Documented architecture

There is no documented architecture of Cassandra.

1.1.3. Common architectures for a DBMS

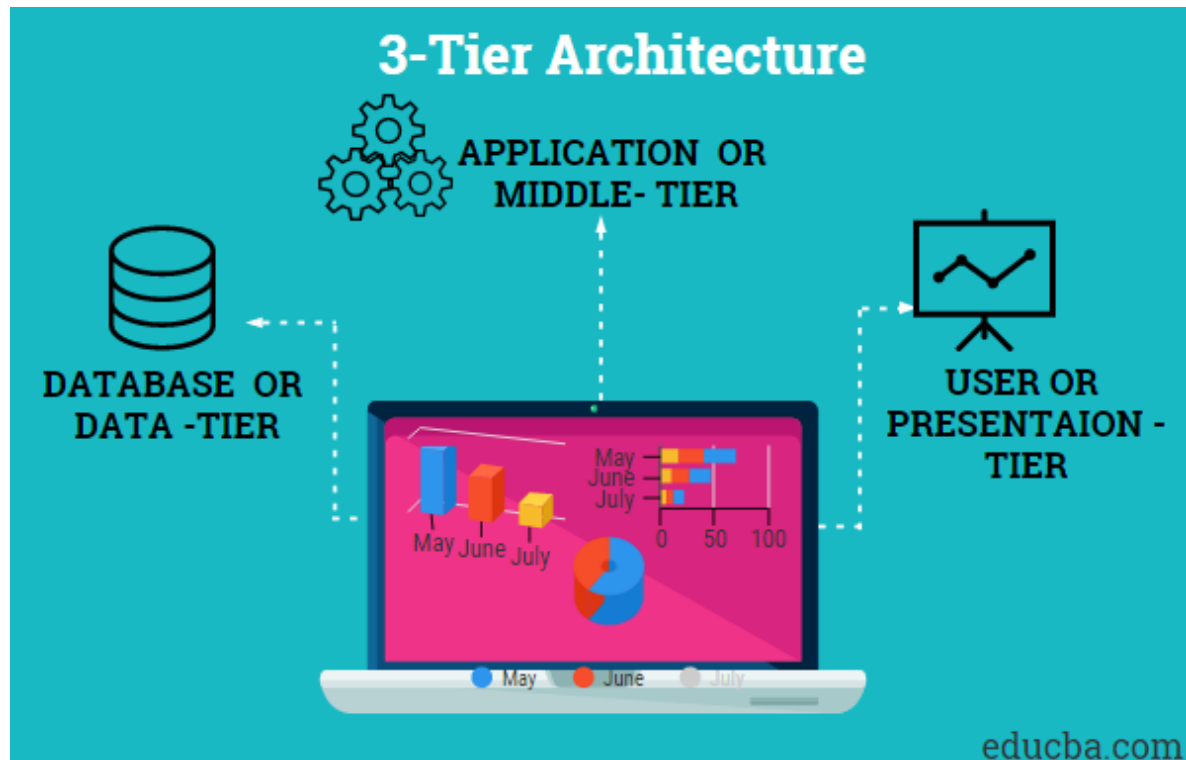
We find a great tutorial on [Introduction to DBMS Architecture](#). The content is summarized below.

There are three types of database architectures: 1-tier architecture, 2-tier architecture, and 3-tier architecture.

With 1-tier architecture, the users can access the database and use the data directly. The 1-tier architecture is mainly used for local applications as it has the client, server and database all on the same machine.

The 2-tier architecture is similar to a **client-server** architecture. Users use APIs like ODBC and JDBC on the client side to communicate with the database on the server side. The server is responsible for providing functionalities like query processing and transaction management. This architecture is used where they need to access DBMS by means of any application. The applications are independent of the database in terms of operation, design, and programming.

3-tier architecture is composed of **Database or Data-tier**, **Application or middle-tier**, and **user or presentation-tier**. In data-tier, the database works with query processing language. The middle-tier is a middleware or gateway and act as a mediator between the database and the user. When the end user accesses the database he is unaware of any existence of database being present. Similarly, the database is unaffected by the user who is accessing it. The user interacts with the database using the presentation-tier, without being aware of anything that is happening at the backend.



(source: <https://www.educba.com/dbms-architecture/>)

The 1-tier is clearly not what we expect as it requires all the server, client and database on the same machine while Cassandra is a distributed database. The 2-tier architecture is not a good choice either. Cassandra cannot be accessed by "any application" and the server cannot do the processing and management because data is physically distributed. So we take the 3-tier architecture as a candidate because it matches our understanding of the product. But we need to check with the codebase.

1.2. Group and reorganize

We first separate the 31 packages into essential functional parts and other components (we also put utility classes and helper classes here, just to make the essential parts fewer and clearer). The description of the package is in the parentheses.

Essential:

- **concurrent** (thread and executors),
- **cql3** (Cassandra Query Language),
- **db** (database),
- **dht** (distinguished hash table for the ring structure of the nodes),

- **gms** (gossip message system, gossip is the internode communication protocol),
- **index** (for read, write and search),
- **io** (reader and writer for disk),
- **locator** (find nodes position on the ring),
- **net** (for communication: encoder, decoder, protocol and versions),
- **schema**,
- **service**

Non-essential:

- **audit** (logging),
- **auth** (authorization),
- **batchlog** (logging),
- **cache** (save cache during writing),
- **config** (configurations),
- **diag** (diagnostic and testing),
- **exceptions**,
- **hadoop** (to complement hadoop),
- **hints** (indicate down rows to provide extreme write availability),
- **metrics** (performance monitor),
- **notifications**,
- **repair** (data repair, to avoid data lost when a node is down),
- **security**,
- **streaming** (transfer data out),
- **tracing**,
- **serializers** (for all data types),
- **tools**,
- **transport** (transport messages),
- **triggers**,
- **utils**

1.3. Matching

It's easy to find that the **db**, **schema** and **io** folder can be group together as the data-tier because they deal with the storage engines: commit log, mem-table and sstable.

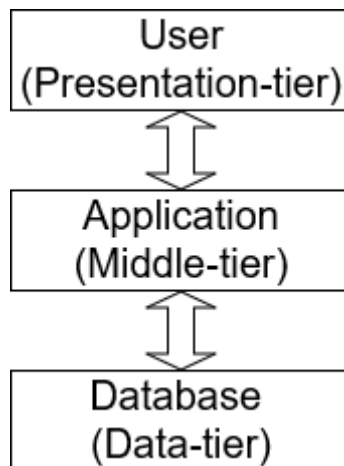
The **service** and **net** folder are the presentation-tier which interacts with the user. User talks to other parts of the database through different services, and the net is responsible for communication and generating messages.

The rest of the essential functional parts do the processing and management work to find the data nodes, read, write, search or communicate between nodes, forming a middle-tier.

Data-tier and middle-tier, and middle-tier and presentation-tier, these two pairs communicate bi-directionally so that user behaviors are processed and sent to data correctly, and data can be collected, processed and sent to users. But data-tier and presentation-tier has no communications. Users cannot work to change the data directly without queries and permissions being checked and executed by the middle-tier. And data changes cannot show up in its "raw" structure, which is impossible for users to understand.

1.4. Conclusion

The following picture shows the architecture of the Cassandra database. It generally follows the 3-tier architecture of database management systems. We did not look closely to other non-essential functionalities this time. But we believe those components just support the trunk of the system, but did not change the main architecture we present here.



2. What is the social context of Cassandra

- Who are the key developers?
- What is the state of the project?
 - level of usage
 - last release date

Cassandra's latest official release (3.11) released on 2/14/2020

<http://cassandra.apache.org/download/>

- release roadmap
 - Currently, Cassandra developers are working on the 4.0 release.
 - There is a [thread](#) of emails detailing the update process
 - The most recent update is dated 2/17/2020 and states 94 open issued pending resolution before the Cassandra 4.0 release
 - Cassandra 4.0 currently has no strict release date
- number of commits
 - Cassandra sees between 5-10 commits a month
 - However there has been a spike of activity over January and February with 10-15 commits each month
 - <https://github.com/apache/cassandra/graphs/commit-activity>
- number of open issues
 - Cassandra has over 2000 open issues of varying priorities
 - However, at this time there are 94 critical open issues against the Cassandra 4.0 release
- active developers
- Coding standards?
 - Cassandra uses [Sun's java coding conventions](#)
 - Furthermore, Cassandra has its own [dedicated coding standards](#) as well, some highlights include:
 - A specific import order
 - Non-redundant overrides

- Minimal 'this' references
 - Specific multiline spacing and splitting formats
- Testing standards:
 - Cassandra uses JUnit and recommends using a unit test when writing test cases
 - The [documentation](#) also mentions Cassandra is mock friendly and the tester should attempt to exclusively cover the test class
 - The documentation recommends using a "Cassandra Distributed Test" for integration testing
- Documentation standards:
 - Information on Cassandra documentation can be found [here](#)
 - Cassandra uses Github to house its documentation and specifies how the Github workflow should be approached
 - Cassandra uses "Sphinx", a static site generator to develop documentation
 - "Sphinx" uses "reStructuredText", a simple plaintext markup language
- Commit standards:
 - Information on how to commit for Cassandra can be found [here](#)
 - Cassandra has specifications for committing based on version as well as whether the commit is a branch or trunk commit
- What is the typical process of contributing?

Cassandra has a workflow for contribution that is summarized [here](#). The first step is to identify what the contributor's goal is. Cassandra recommends some introductory topics such as updating documentation or writing a test case. Cassandra also has a [review checklist](#) that covers some of the main points for each type of contribution. There is also information on the specific steps to take when contributing [here](#). To summarize...

- Gain insight on the issue you wish to address through the community
- Choose the right branch to work on
- Code and test!
- Create a patch

At this time the review checklist is useful to make sure the contributor has adhered to all the standards discussed above.

3. Interesting pull requests

All current pull request from Cassandra's open source repository can be found here: <https://issues.apache.org/jira/projects/CASSANDRA/issues/CASSANDRA-15273?filter=allopenissues>

3.1. Make it possible to resize concurrent read/ write thread pools at runtime

official issue link: <https://issues.apache.org/jira/browse/CASSANDRA-15277>

first attempt link: <https://github.com/apache/cassandra/pull/340>

second attempt link: <https://github.com/apache/cassandra/pull/371>

This is technically not a issue but an improvement instead. To better mitigate cluster overload, executor services for various stages should be configurable at runtime. The author who was trying to fix this issue first created a pull request that fixed only part of the problem. He used a package called "codahale metrics" and made the "read" configurable at runtime. However, the reviewer from Cassandra team noticed that the route he chose could finally solve the problem

but a large amount of refactoring of all JMX MBeans will be needed. Meanwhile the reviewer pointed out a neater way to approach this issue by doing something on Virtual Tables. This pull request got force-pushed to another pull request, which is the one he finally solve the problem in a better way.

After the author added some new classes and made some changes to it the original methods and variables, the reviewer had some interesting conversation with him before he merged the commits.

1. The variable name was not consistent. As Cassandra is a huge system, all variable name should be the same if they are under the same context with same meanings. However the author did not use the same variable name. After being pointed out, the author gladly agreed to change and make the whole system neater.



belliottsmith on Oct 28, 2019 Contributor



Could rename all of these to `maxPoolSize` for consistency - not necessary at all, but might be nice now we have concept leakage



jonmeredith on Oct 28, 2019 Author



Agreed - I don't think these changes introduce any additional concept leakage, the concepts already leaked and hopefully this tidies things up a little.

2. One of the method does not seem to create a huge value to the system. The method was going to keep the configuration up to date with the system. However, from the reviewer's point of view, we do not need to update this configuration in all places. There isn't a big value in it except when debugging. The author agreed with the reviewer, however, he still insists on keep the code as it is to avoid confusion for debugging.



jonmeredith on Oct 30, 2019 Author



Agreed it isn't uniformly updated and perfect, but keeping the Config updated with the current values may avoid confusion for debugging heap dumps as you mentioned. As you don't object, I'll keep the code as is.

3. Explanation of why interface was designed this way from the author. This interaction was pretty straightforward that the author just explained why he chose to keep the interface.

```
src/java/org/apache/cassandra/concurrent/ResizableThreadPool.java
```

```
23 + /**
24 +  * Returns maximum pool size of thread pool.
25 +  */
26 + public int getCorePoolSize();
```



belliottsmith on Oct 31, 2019 Contributor



Most of these methods aren't used. Should `LocalAwareExecutorService` extend `ResizableThreadPool` ?



jonmeredith on Oct 31, 2019 Author



Originally I added it make sure the JMX support for resizing pools was common. It's the same as the interface in `LocalAwareExecutorService` so it makes sense to use the interface. Updated.

3.2. Arithmetic operators over decimal truncate results

issue link: <https://issues.apache.org/jira/browse/CASSANDRA-15232>

pull request that got merged: <https://github.com/apache/cassandra/pull/334>

The issue was pretty straightforward. The decimal operators have their computations with a hard-code 128 bit precision, which does not make sense to have rule especially when performing multiplication and addition.

I found this pull request interesting because when I clicked in, I was expecting to see some conversation that happens between the author who created this pull request and reviewer who was reviewing the codes and leave some feedbacks there. However, the truth is that one person did both jobs and since she was able to review and merge her own code, she must be part of the Cassandra team. Firstly, she commented with a very clear comparison of what function was in the code before and what does it have now. Part of her comment can be seen below:

add (+), subtract (-) and multiply (*) operations:

- before:
 - precision of result used to be always 34 (see `MathContext.DECIMAL128`)
- after:
 - precision (number of significant digits) of result is at most 10000.
If result exceeds given precision it will be rounded using `HALF_UP` mode

division (/) operation:

- before:
 - precision used to be always 34 (see `MathContext.DECIMAL128`)
- after:
 - expected scale is set to minimum precision (32) minus estimated position of first digit in quotient
 - scale should be at least as big as maximum scale of operands
 - scale should not be less than 32
 - scale should not be bigger than 1000
 - if actual quotient scale is bigger than calculated scale then result is rounded using `HALF_UP` mode
 - trailing zeros are stripped


Meanwhile, she pointed out what has not been fixed yet regarding to the issue:

To do:

- ☒ do not truncate mod operation
- ☒ add tests for mod operation

In addition to that, she copied some code snippets and left comments about why she did such changes. Just like a normal author-reviewer interaction except she is both of them. This is an odd pull request for us at first but we understood the meaning behind it. Even it was someone within the team to fix a issue, they should also leave records and comments in its git history and explain their intention of doing so just like someone else outside the team.


```
test/unit/org/apache/cassandra/cql3/functions/OperationFctsTest.java  Outdated
196 - new BigDecimal("0.7857142857142857142857142857142857"),
197 - new BigDecimal("0.9285714285714285714285714285714286"),
196 + new BigDecimal("0.785714285714285714285714285714286"),
197 + new BigDecimal("0.928571428571428571428571428571429"),
```

 **kornilova-l** on Jul 23, 2019 • edited ▾ Author Contributor + 😊 ...

This change is because precision used to be 34 (in this case it causes 34 significant digits after dot) and now minimum scale is 32 (32 digits after dot)

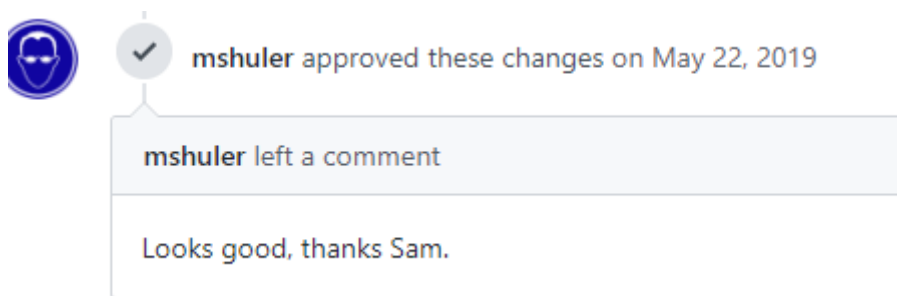
3.3. Add note regarding DROP ROLE and connected sessions

pull request: <https://github.com/apache/cassandra/pull/321>

This pull request that got merged caught my eye not because how complicated code he wrote but how simple but also important the contribution is. By fully understand what does DROP ROLE do and does not do, the author add a chunk of clear explanation regarding granting a role. It points out that DROP ROLE will not terminate any open user sessions and the currently connected sessions will remain connected. However, it will revoke a drop role and consequently, if there were still connections going on, newly granted permissions and roles will be acquired.

```
151 + .. note:: DROP ROLE intentionally does not terminate any open user sessions. Currently connected sessions will remain
152 + connected and will retain the ability to perform any database actions which do not require
      :ref:`authorization<authorization>`.
153 + However, if authorization is enabled, :ref:`permissions<cql-permissions>` of the dropped role are also revoked,
154 + subject to the :ref:`caching options<auth-caching>` configured in :ref:`cassandra.yaml<cassandra-yaml>`.
155 + Should a dropped role be subsequently recreated and have new :ref:`permissions<grant-permission-statement>` or
156 + :ref:`roles<grant-role-statement>` granted to it, any client sessions still connected will acquire the newly granted
157 + permissions and roles.
158 +
```

The note itself is not very long, but it gives a clear note to whoever is using the system about this this DROP ROLE actually works during different sessions. This kind of pull request does not contribute to the code base, however, it creates a value for users that they can understand the system better. Reviewer from Cassandra happily merged the change with gratitude.



3.4. Optimize streaming


pull request link: <https://github.com/apache/cassandra/pull/239>


This pull request is not fixing any particular issue that was posted on Cassandra official page, but an improvement on streaming.

For almost every commit on this pull request, the author was actually working with his teammate to solve the problem together. When click on one of the commits, a detailed conversation between him and his teammates can be seen. Also in one of the comments, his teammate commented that he was not familiar with a concept `LifeCycleTransaction` in Cassandra and he

tagged a code reviewer from Cassandra and hoped to get some help of explaining this for him and his team.

```
68 68 // The transaction records, this set must be ORDER PRESERVING
69 - private final LinkedHashSet<LogRecord> records = new LinkedHashSet<>();
69 + private final Set<LogRecord> records = Collections.synchronizedSet(new LinkedHashSet<>()); // TODO: Hack until we
```


 **iamaleksey** on Jul 24, 2018 Member + 😊 ...
I'm not very well familiar with `LifeCycleTransaction` code, but I'm not sure this is sufficient as a workaround. @belliottsmith Could you please confirm that this - or isn't - sufficient? Here and the equivalent change to `LogReplicaSet`.

 **dineshjoshi** on Jul 24, 2018 Author Member + 😊 ...
@iamaleksey I don't think it is sufficient but we have to address this conclusively as part of 14554. I've already spoken to @belliottsmith regarding it and have a plan to fix this.


Even though the reviewer did not reply directly on the thread, the author or the team lead talked to the reviewer and figured out the way to fix this. It's really interesting to see how the team who's working on the pull request communicate within the team and also talk with code reviewers from Cassandra when they are stuck. This was only one commit of a big improvement requests.


There are about 11 total commits and 143 comments from this pull request. The reviewer reviewed all his codes and since this was not a small fix, a lot of conversations were going on between the reviewer and the team that is working on it. The problems of the code that the reviewer pointed out include:

1. random imports
2. the implementation of new code
3. comments that does not make right sense
4. new coding style does not follow Cassandra's coding style, which is very important that even some reviewers said they've done the same mistake before (both [aweisberg](#) and [belliottsmith](#) are code reviewers from Cassandra) More than half of the review comments were based on wrong style such as "new line before brace".


 **aweisberg** on Jul 6, 2018 • edited ▾ Contributor + 😊 ...
This also doesn't follow the code style, we are supposed to use braces for single line if all the time (from the Sun Java code style).

Can you review the entire change set for style issues?

 **dineshjoshi** on Jul 10, 2018 Author Member + 😊 ...
The codebase that I've been through doesn't use braces for single statement blocks. I assumed that was the convention. I can change it in my PR to follow the Sun style but we should ideally audit and change the code style in the future to avoid confusion.

 **belliottsmith** on Jul 10, 2018 • edited ▾ Contributor + 😊 ...
I think we need to raise this on the mailing list to possibly modify the code style, because for a long time it has been standard practice for many contributors (myself included) to elide the braces for single line statements, because otherwise the code becomes illegible.

Note this all springs from the modification to the sun code style, i.e. the new line per brace. Everyone would prefer braces, but they would prefer the code to be readable too :/

 **belliottsmith** on Jul 10, 2018 Contributor + 😊 ...
Even you've done it! :)

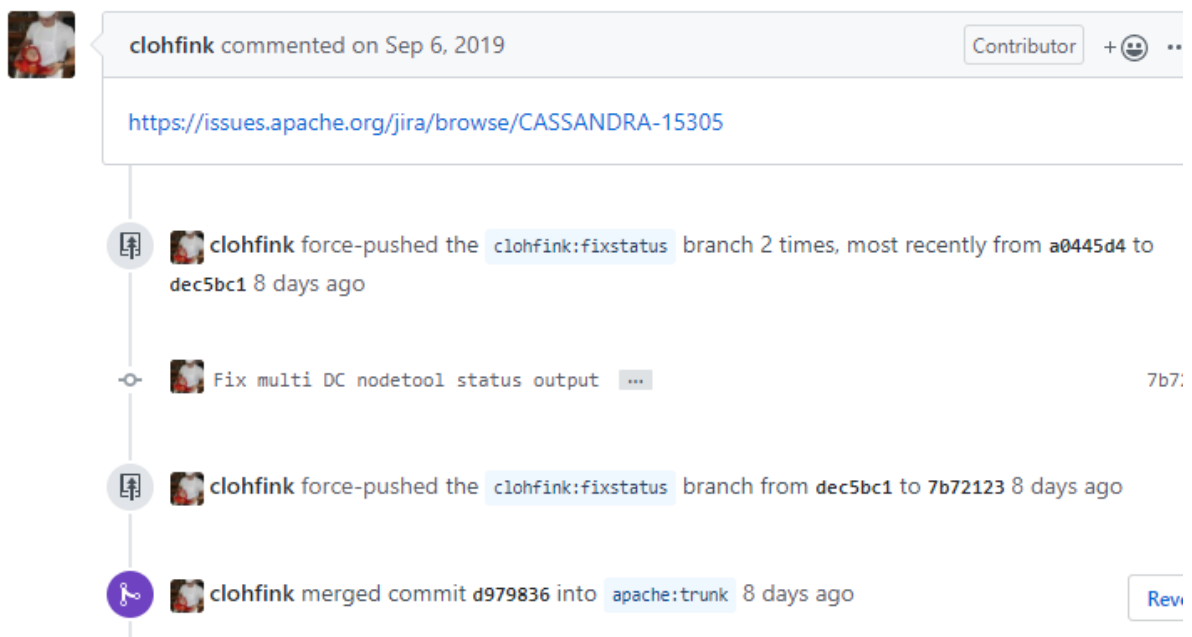
3.5. Fix multi DC nodetool status output

issue link: <https://issues.apache.org/jira/browse/CASSANDRA-15305>

pull request: <https://github.com/apache/cassandra/pull/353>

All of the previous 4 pull request got closed instead of being merged after however many discussion with the reviewer. The 5th one actually was one of the few pull requests that got merged into Cassandra trunk.

It is quite interesting because it is the reviewer who was fixing the problem himself. I was expecting some kind of comment that left for the reason of him adding or deleting some code pieces but I found none. The only information I found for how he fix the issue was the description of this commit. One reason of no comments of this pull request could be that it happened recently (8 days ago), and they haven't uploaded any document of this fix yet, otherwise this is not a good way of showing stakeholders the reason and the approach of fixing this issue.



The screenshot shows a GitHub pull request history for the issue "Fix multi DC nodetool status output". At the top, a comment by clohfink from Sep 6, 2019, links to the Jira issue. Below, the commit history is shown: clohfink force-pushed the clohfink:fixstatus branch 2 times, most recently from a0445d4 to dec5bc1 8 days ago; clohfink force-pushed the clohfink:fixstatus branch from dec5bc1 to 7b72123 8 days ago; and clohfink merged commit d979836 into apache:trunk 8 days ago.

4. Interesting issues

Cassandra issue page: <https://issues.apache.org/jira/projects/CASSANDRA/issues/CASSANDRA-15273?filter=allopenissues>

4.1. OOM when using SASI index

The issue is when running Cassandra, 2.8Gb of the heap is taken by the index data, pending flush to tables, and this caused node (where data is stored) with OOM (out of memory) issue. The question is:

1. Why can't Cassandra keep up with the inserted data and flush it ?
2. What resources/configuration should be changed to improve the performance?

The issue was shown with a picture below:

Instances of class 'org.apache.cassandra.index.sasi.conf.ColumnIndex'
 Objects: 5 / shallow size: 240 / retained size: 3.1 GB All the objects are strong reachable [Reachability scopes](#)

Class	Name	Retained Size	Shallow Size
org.apache.cassandra.index.sasi.conf.ColumnIndex		2,862,562,448	48
pendingFlush	java.util.concurrent.ConcurrentHashMap size = 61	2,862,507,472	64
tracker	org.apache.cassandra.index.sasi.conf.DataTracker	54,712	24
column	org.apache.cassandra.config.ColumnDefinition	184	64
component	org.apache.cassandra.io.sstable.Component	136	24
<class>	org.apache.cassandra.index.sasi.conf.ColumnIndex	128	72
keyValidator	org.apache.cassandra.db.marshall.Int32Type	40	24
memtable	java.util.concurrent.atomic.AtomicReference	32	16
mode	org.apache.cassandra.index.sasi.conf.IndexMode	32	32
config	java.util.Optional	16	16
isTokenized	= boolean false		1
org.apache.cassandra.index.sasi.conf.ColumnIndex		389,428,688	48
org.apache.cassandra.index.sasi.conf.ColumnIndex		30,387,568	48
org.apache.cassandra.index.sasi.conf.ColumnIndex		24,699,328	48
org.apache.cassandra.index.sasi.conf.ColumnIndex		13,039,560	48

It's interesting to read people's comments as each of them are encountering the problems under different circumstances. Some of them pointed out that this might be related to hardware issue. Since Cassandra has official recommendation of running the system is 8 cores CPU and 32 Gb RAM. Others pointed out that it works fine with only 20Gb per node, so this could be a software issue.

Right now, 4 cores CPU is below the official recommendation to run Cassandra in production, which is 8 cores CPU. Same for RAM, recommendation is 32Gb, see here: <http://cassandra.apache.org/doc/latest/operating/hardware.html>

20Gb per node should be quite fine. Did you try configuring `memtable_*heap_space_in_mb` (which is 1/4 heap by default) and `memtable_cleanup_threshold`. So technically 4Gb partition updates is still "under" the threshold. Problem is that there's some memory (also, CPU) involved into the flush process, so making it lower should help.

This issue was created in Sep, 2018 and it is still open now. Even though people has been constantly posting new findings, the essential part of this issue seem not to be found.

4.2. Python 3.8 fails to execute cqlsh

Issue page: <https://issues.apache.org/jira/projects/CASSANDRA/issues/CASSANDRA-15573?filter=allopenissues>

This issue was posted less than two weeks ago concerning the latest Python version cannot read some code lines and it's throwing some errors. One of the key committers pointed out that not just Python 3.8, if you are running earlier Python versions such as 3.6 and 3.7, similar problems will appear due to the missing version check in the DOS Batch file. He linked to another page with simliar Python version problems (<https://issues.apache.org/jira/browse/CASSANDRA-15572>), and this page leads to an open pull request that people are working on right now. (<https://github.com/apache/cassandra/pull/448>).

▼ Dinesh Joshi added a comment - 17/Feb/20 00:40

Thanks for reporting this issue. Please see my comment on [CASSANDRA-15572](#). These are some of the issues you'll run into if you use > 3.6. There is additional work to adapt SaferScanner in 3.7 and 3.8.

▼ Dinesh Joshi added a comment - 1 week ago

Circling back once more, this is again due to the missing version check in the DOS Batch file (`cqlsh.bat`) and doesn't affect Linux-like systems.

It is interesting to see the flow from which the issue was first found and documented, then someone who's familiar with this issue might point out that this issue could be related to other issues that are similar. If that's the case, people will mark this issue as `TRIAGE NEEDED`, and usually there will be a pull request, either an open or a closed one, trying to solve the problems. Another interesting fact that we noticed was that how difficult to maintain a big code base like Cassandra. As not only there are bugs, improvments to fix, but also as modern programming languages develops, there will be a lot of deprecated code lines that needs to be refractored.

4.3. Overflow of 32-bit integer during compaction

Issue page: <https://issues.apache.org/jira/projects/CASSANDRA/issues/CASSANDRA-14773?filter=allopenissues>

After a development update to decrease memory and CPU usage, a rounding error was discovered in Cassandra. When attempting to round a certain expiration time, there is a chance it hits MAX_INT. The issue is more than two years old with varying degrees of success but greater complications are introduced by classes that depend on the rounded value.

StreamingTombstoneHistogramBuilder is an example of such a class. At this time the original developer who posted this issue is no longer working on it, but there is some recent interest in addressing this issue.

4.4. Flakey Test: testAcquireReleaseOutbound

Issue page: <https://issues.apache.org/jira/projects/CASSANDRA/issues/CASSANDRA-15308?filter=allopenissues>)

This issue, which is approximately 6 months old, addresses a test case which fails on some occasions. There is an issue with resource capacity as the test runs multiple rounds of testing. As the test runs, resource capacity is depleted and ultimately might end in a failed test without proper result. This of course depends on the circumstances of the test case and how many rounds are run. The issue is partially resolved and in review currently.

4.5. Fix flaky test

Issue page: <https://issues.apache.org/jira/browse/CASSANDRA-15526>

This issue was posted recently and whoever found this issue noticed a flaky test and it seems like the code is not producing the same result. It is tricky to fix a flaky test because different from fixing source code, fixing test code need to reproduce how the code failed and record all related variables that could lead to that failure. The author who created this issue pointed out that this could be a Java version issue as it occurs with Java 11.

David Capwell added a comment - 24/Jan/20 23:41

The failure was seen on java 11

Later someone else said it could be caused from OS, because he was able to produce the same failure with Ubuntu VM.

Ekaterina Dimitrova added a comment - 07/Feb/20 18:24

Jordan West, are you looking into this one? I see it unassigned so I wasn't sure.

As I said in CASSANDRA-15527, on Ubuntu VM I am able to reproduce this one. Still no success with the other two reported by David Capwell so I think his point that this one might be the main trouble maker is a valid assumption to start with.

Do you want me to look into it?

After more and more people pitched in their thoughts, one of the active members conclude the failure is probably caused by one a Java built-in method `size()`, and the possible solution can be using a different function that can achieved the same testing scenario instead of using `size()`

Jordan West added a comment - 11/Feb/20 15:02

Ekaterina Dimitrova the use of `size()` in the code is definitely suspicious given the javadoc. Whats odd to me is `isEmpty`, `first`, `size` all rely on an internal method called `findFirst`. Unless we are deleting from the set, which right now I don't believe we are, it seems unexpected that the first two would return indicating elements are present but `size` does not – even with the javadoc warning. With that in mind I'd like to better understand the root cause of the failure. I also explored if we could get rid of using `size` but I don't see a clear path to doing so at this time.

