CS 6381 Final Report - Apache Cassandra Team:

1. Overview

\subsection{Apache Cassandra}

Apache Cassandra is a distributed, NoSQL datastore that runs on Java. Originally designed at Facebook to combine the benefits of Google’s Bigtable and Amazon’s Dynamo, Cassandra aimed to fulfill the growing need for a large-scale and low-latency datastores that provide global availability. Over its iterations, Cassandra has included features such as: full multi-master database replication, online load balancing, a flexible schema, and several others.

Cassandra’s approach to the choices around the CAP Theorem is to guarantee availability and partition tolerance, while supporting eventual consistency across the data store. Other features include lightweight transactions, linearizable consistency, and guaranteed consistency secondary indexes.

~\cite{cassandra\_official\_site}

~\cite{apache\_cassandra\_wikipedia}

~\cite{cassandra\_configuration\_file}

~\cite{cassandra\_third\_party\_projects}

~\cite{datastax\_expert\_support}

~\cite{apache\_cassandra\_tutorial\_4}

~\cite{cassandra\_documentation}

~\cite{getting\_started\_with\_cassandra\_on\_windows}

~\cite{apache\_cassandra\_tutorial\_3}

~\cite{installing\_apache\_cassandra\_on\_a\_mac}

~\cite{running\_a\_cassandra\_application\_in\_docker}

~\cite{hub\_docker\_cassandra}

~\cite{installing\_apache\_cassandra\_3x\_on\_debian}

~\cite{setting\_up\_an\_arm\_based\_cassandra\_cluster}

~\cite{cassandra\_documentation\_hardware\_choices}

~\cite{eiger\_github}

~\cite{hands\_on\_with\_cassandra\_commands\_cqlsh}

~\cite{quora\_how\_much\_data\_can\_a\_single\_node}

~\cite{stackoverflow\_cassandra\_node\_limitations}

~\cite{datastax\_cql\_limits}

~\cite{which\_part\_cap\_theorem\_cassandra}

~\cite{youtube\_cassandra\_demo}

~\cite{object\_serialization\_vs\_relational\_data}

~\cite{clustering\_order\_in\_cassandra}

~\cite{cassandra\_data\_types}

~\cite{sorting\_and\_limiting\_results}

~\cite{cassandra\_read\_performance}

~\cite{cassandra\_faq}

~\cite{features\_of\_cassandra}

~\cite{casquatch\_an\_open\_source}

~\cite{is\_cassandra\_hard\_to\_learn}

~\cite{guide\_to\_cassandra\_with\_java}

~\cite{datastax\_java\_driver\_49}

~\cite{youclap\_kassandra}

~\cite{ruby\_driver\_for\_cassandra}

~\cite{how\_to\_start\_cassandra\_server}

~\cite{cassandra\_client\_drivers}

~\cite{cassandra\_guarantees}

~\cite{cassandra\_multi\_master\_replication}

~\cite{the\_cassandra\_query\_language}

~\cite{apache\_cassandra\_org}

~\cite{application\_connectivity\_with\_cassandra}

~\cite{data\_transmission\_between\_nodes\_and\_client\_cassandra}

~\cite{wikipedia\_gossip\_protocol}

~\cite{towards\_data\_science\_cassandra\_python}

~\cite{datastax\_node\_js\_driver\_41}

~\cite{datastax\_node\_js\_driver\_for\_apache\_cassandra}

~\cite{datastax\_c\_plus\_driver\_20}

~\cite{datastax\_cpp\_driver}

~\cite{stackoverflow\_how\_much\_data\_per\_node}

1. Experiment Criteria and Design

\subsection{Apache Cassandra Experiment Criteria and Design}

We created a set of applications to test the read-write efficiency of the Apache Cassandra database based on our overall understanding of the experiment designers’ guidelines and the time available. Each set of applications connects to a created keyspace within the Cassandra database before interactions with the database are made. On Ubuntu 18.04, we installed Cassandra and ran it using the “sudo service cassandra start” command, checked its active status using the “sudo service cassandra status” command, and restarting by running “sudo service cassandra stop” and the start command as mentioned here. With this, we were able to run our sets of applications.

One set of applications revolves around calling read and write queries to a table in the Cassandra database we called table1 using the Python file, cassandraRW.py. Each write query inserts a record to the database that contains a string parameter, message, which consists of randomly generated characters to a specified length that is supplied to the function call. Each read query selects 1 record from the database. Time differences are saved to a CSV with the title including the indication of read or write, number of iterations, host name and process ID, which are obtained using terminal commands, current timestamp, and UUID to distinguish them apart. We ran tests using shell files to call these queries a number of times based on the following conditions: Each reader and writer works independently (neither have to wait for the other to finish to start running). A write query collection contains 5 write queries that vary in terms of total character lengths as follows: 128, 254, 512, 1024, and 10000.

Our expectations for running the tests are as follows: For the 1writerTest, we run 1 write query collection once. For the 1readerTest, we run 1 read query once. For the 10writer1readerTest, we run 10 write query collections then 1 read query, 10 times. For the 1writer10readerTest, we run 1 write query collection then 10 read queries, 10 times. For the 25writer25readerTest, we run 25 write query collections then 25 read queries, 5 times.

We tried to implement a set of applications to test the CAP analysis efficiency of Apache Cassandra, but we fell short likely due to unstable connection to Cassandra. The applications revolve around capAnalysis.py and testing the efficiency of readers based on sample text files of varying character lengths. There are 5 sample files, each of which is named “sample” concatenated with a number from 1 to 5, followed by the “.txt” file extension. Among the queries, there are various parameters as follows: payload refers to the text from the sample file and recordIndex refers to the numerical value in the sample file name. There are 3 different queries, insert, update, and read, each of which is accessed by a URL request and involves looping through a folder directory to find a sample file with matching numerical value as that specified by the provided recordIndex. insertQuery() accepts a payload and a recordIndex, appends the payload to the current timestamp as a new message, and inserts the message into the database with the id equal to the recordIndex. updateQuery() acts like insertQuery() except the difference is it updates the message of the record with the id that matches the given recordIndex. readQuery() accepts a recordIndex, notes the timestamp before selecting for a record from the table where the id matches the recordIndex, gets the timestamp after, splits the message to get the timestamp from the record, and returns this information along with the string, “reader”, concatenated to the recordIndex. Time differences are saved to a CSV with the title including the number of iterations, host name and process ID, which are obtained using terminal commands, current timestamp, and UUID to distinguish them apart. The tests we run revolve around readIteration.py. We run capAnalysis.py as a server that can accept incoming URL requests. Next, we run “http://localhost:5000/u/1” to keep the write query running on the server. Then we run readIteration.py with an argument of 10 to get 10 iterations of send URL requests to the server to run read queries. We do this 10 times.

We used calculateStatistics.py to read the data from the time differences CSV files and save a CSV filled with statistics and calculateStatisticsAll.py for similar purposes with the additional feature of generating graphs based on the calculations. Here, we set a for-loop to check for CSV files in a folder. When there’s a match, we call readCSV(), which takes in the file and opens it, iterates through it, and appends the write and read times to lists as well as write and read success rates, which are based on how many time rows there are divided by the total iterations for a file. For latency and throughput, using this collected information, we get the minimum and maximum values for times by using the min() and max() functions respectively, average by summing the tuple-converted data list of float-converted times and dividing that by the length of the data list, and standard deviation by getting the variance for the result of dividing the sum of the data list by the length of the data list and setting it to the power of 0.5. These findings are added to a calculations CSV file with the current timestamp and UUID in the title to distinguish the files apart. calculateStatisticsAll.py featured generating graphs by passing in the file name that is parsed from the path name. Using matplotlib and pandas, graph settings are configured with the X-axis showing the various test conditions and the Y-axis showing the values for the parameter of interest as graphs are generated.

We used hardwareMeasure.py to continuously see the latest hardware statistics, including CPU, RAM, and storage. In a “while True” loop, we printed the functions that return a dictionary including the max average values for each of these parameters. In each function, we appended the parameter fetching using psutil to its own list and from that list, we find the maximum value using the max() function and average value by summing up the tuple-converted data list and dividing that by the length of the list.

1. Experiment Results

\subsection{Apache Cassandra Experiment Results}

The following includes our observations of trends among the graphs, which are stored in the “graphs\_ApacheCassandra1” folder. In terms of average, max, min, and standard deviation for read latency and throughput and write throughput, all results appear high around the same range in the graphs. Upon closer observation, the lowest yields come from the 1st, 2nd, and 10th 10writer1reader tests, the moderate yields come from the 3rd to 8th 10writer1reader tests, and higher yields come from the rest of the conditions. For write latency, the exception is the min, where the 1st 10writer1reader test result yields the highest result, followed by the 2nd, 3rd, 4th, and 10th, followed by the 5th, 6th, and 7th, and the rest of the conditions yield lower results around the same range. From these observations, it seems that Apache Cassandra, especially when there are more readers, performs well in throughput and has increased latency. An exception occurs at least in our case as generally when there are more readers, the write latency is lower. We think one possibility for increased latency could be due to our connection issues to Cassandra.

For the base file, it appears the hard disk memory stays roughly stable while the readIteration.py test is run. RAM decreases and CPU increases from the 1st to the 2nd load. The computer we used had other activities going on outside of the VM, so perhaps this contributed to the RAM and CPU fluctuations. The computer has a lot of hard disk memory, so changes to the memory should appear minimal as long as large-sized items are not added or removed.

graphs from “graphs\\_ApacheCassandra1” folder:

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/readLatencyAverage\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e.png}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/readLatencyMax\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/readLatencyMin\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/readLatencyStandardDeviation\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/readThroughputAverage\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/readThroughputMax\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/readThroughputMin\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/readThroughputStandardDeviation\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/writeLatencyAverage\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/writeLatencyMax\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/writeLatencyMin\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/writeLatencyStandardDeviation\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/writeThroughputAverage\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/writeThroughputMax\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/writeThroughputMin\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/writeThroughputStandardDeviation\_04-23-2021\_195445\_46df3f26-3fcc-4621-9966-71a727f2277e}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/HardwareMeasurements\_CPU.PNG}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/HardwareMeasurements\_HardDiskMemory.PNG}

\includegraphics[width=50mm,scale=0.5]{./Figs/graphs\_ApacheCassandra1/HardwareMeasurements\_RAM.PNG}

In an additional study, the Cassandra team tested out multiple benchmarks tests within the data repository, the results of such metrics were directionally indicative that the database conforms generously in a containerized ring or distributed architecture utilizing eventual consistency. Graphs for this study are stored in the "graphs\\_ApacheCassandra2" folder. In addition to having such an architecture the given application utilizing Cassandra is best suited when the needs correspond with a write heavy systems.

The benchmarks were split into two separate tests, the first test measured differing ratios in readers and writers. The second scenario we measured some of the CAP consistency benchmarks you see to understand the dynamics in the eventual consistency.

In the read write ratio test iterated through 2 different scenarios; 1 writer to 10 readers (Figure 1A), 10 writers to 1 reader (Figure 1B), and 25 writers to 25 readers (Figure 1C). The Cap consistency test measured ‘start read after writes’ and read after writes. This helped measure some of the consistency gaps noticed after writing and completing a read. All test collected averages of the various sized text that was generated for database insertion and reading. We also collected standard deviation within each test category itself, parsed by single runs with multiple rows.

The results of the 1 write to 10 readers resulted in an average of 18 ms, respectively. The metric represented a system where 1 machine was writing and 10 readers were vying for reads, this resembles a read heavy application. The 10 writers one reader only saw a minor increase in milliseconds resulting in an average of just 4 more ms, 22ms. We can see the curve being somewhat flat due to seeing about a 20% increase in time differentials but increasing the writers 10 times. When measuring 25 writers and 25 readers, we wanted to resemble a fairly balanced system, yet with high traffic usage. This averages 19ms, just slightly slower than the read heavy system.

The CAP testing resulted in graphs which looked very similar (please see below image), we measured the average once again and saw some interesting results. The ‘start read after write’(SRAW) metric averaged 4.2 ms and the ‘read after write’(RAW) had an average of 4.1ms. When looked in a visual format we notice the similar numbers, the SRAW looks like how quickly the system could start to read a consistent value after a machine had made the new insertion into the database, eg availability. The second measurement ‘RAW’ looked at how long it took a machine to completely finish that reading, eg effective consistency. We see that within the topology and provisioning chosen for our environment the results were negligible.

Cassandra CAPS Graph

\includegraphics{./Figs/graphs\_ApacheCassandra2/CapData.png}

As a single node Cassandra test its important to understand the CAP testing would most likely see an increase in ms due to network traffic variables which would be introduced. Current threading mechanism for a single system allotted for finer grain scheduling due to the homogeneity. In summary Cassandra is a great database for a distributed system that requires loose data schemas and writes very frequently. I believe the polar opposite of such a database would be SQL, which scaled better vertically and has fantastic consistency with quick reads.

Figure 1A

\includegraphics{./Figs/graphs\_ApacheCassandra2/cass1w10r.png}

Figure 1B

\includegraphics{./Figs/graphs\_ApacheCassandra2/cass10w1r.png}

Figure 1C

\includegraphics{./Figs/graphs\_ApacheCassandra2/cass25w25r.png}

1. Repository
   1. <https://github.com/jamesvu1/cs6381finalprojectapachecassandra>
2. References

@inproceedings{cassandra\_official\_site,

booktitle={Cassandra Architecture Overview},

url={https://cassandra.apache.org/doc/latest/architecture/overview.html#overview},

year={2016}

}

@inproceedings{apache\_cassandra\_wikipedia,

title={Apache Cassandra},

url={https://en.wikipedia.org/wiki/Apache\_Cassandra},

year={2021}

}

@inproceedings{cassandra\_configuration\_file,

title={Cassandra Configuration File},

url={https://cassandra.apache.org/doc/latest/configuration/cassandra\_config\_file.html},

year={2016}

}

@inproceedings{cassandra\_third\_party\_projects,

title={Third-party projects},

url={https://cassandra.apache.org/third-party/},

year={2016}

}

@inproceedings{datastax\_expert\_support,

title={Datastax Expert Support},

url={https://www.datastax.com/services/support},

year={2021}

}

@inproceedings{apache\_cassandra\_tutorial\_4,

title={Apache Cassandra - Tutorial 4 - Installing Cassandra On Linux},

author={jumpstartCS},

url={https://www.youtube.com/watch?v=pGhkX5z\_vW8},

year={2019}

}

@inproceedings{cassandra\_documentation,

title={Documentation},

url={https://cassandra.apache.org/doc/latest/getting\_started/installing.html},

year={2016}

}

@inproceedings{getting\_started\_with\_cassandra\_on\_windows,

title={Getting Started with Cassandra on Windows | DataStax},

url={https://www.datastax.com/blog/getting-started-apache-cassandra-windows-easy-way},

year={2021}

}

@inproceedings{apache\_cassandra\_tutorial\_3,

title={Apache Cassandra - Tutorial 3 - Installing Cassandra On Windows},

author={jumpstartCS},

url={https://www.youtube.com/watch?v=EEXtVn3zAqc},

year={2019}

}

@inproceedings{installing\_apache\_cassandra\_on\_a\_mac,

title={Installing Apache Cassandra on a Mac},

author={Programming with Dr. Hayes},

url={https://www.youtube.com/watch?v=JtpsOFXJUBw},

year={2020}

}

@inproceedings{running\_a\_cassandra\_application\_in\_docker,

title={Running a Cassandra Application in Docker},

url={https://www.datastax.com/learn/apache-cassandra-operations-in-kubernetes/running-a-cassandra-application-in-docker},

year={2021}

}

@inproceedings{hub\_docker\_cassandra,

title={cassandra},

url={https://hub.docker.com/\_/cassandra},

year={2021}

}

@inproceedings{installing\_apache\_cassandra\_3x\_on\_debian,

title={Installing Apache Cassandra 3.x on Debian-based systems},

url={https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/install/installDeb.html},

year={2021}

}

@inproceedings{setting\_up\_an\_arm\_based\_cassandra\_cluster,

title={Setting up an ARM Based Cassandra Cluster with Beagle Bone Black},

author={Dan Ricart},

url={https://www.linux.com/training-tutorials/setting-arm-based-cassandra-cluster-beagle-bone-black/},

year={2013}

}

@inproceedings{cassandra\_documentation\_hardware\_choices,

title={Hardware Choices},

url={https://cassandra.apache.org/doc/latest/operating/hardware.html},

year={2016}

}

@inproceedings{eiger\_github,

title={eiger},

author={Wyatt Lloyd},

url={https://github.com/wlloyd/eiger},

year={2015}

}

@inproceedings{hands\_on\_with\_cassandra\_commands\_cqlsh,

title={Hands-on with Cassandra Commands | Cqlsh Commands},

author={TechTransform},

url={https://www.youtube.com/watch?v=z0fPRyR12KA},

year={2020}

}

@inproceedings{quora\_how\_much\_data\_can\_a\_single\_node,

title={How much data can a single node in a Cassandra cluster handle? Are there configuration settings to set the data size/limit the data size on the node?},

url={https://www.quora.com/How-much-data-can-a-single-node-in-a-Cassandra-cluster-handle-Are-there-configuration-settings-to-set-the-data-size-limit-the-data-size-on-the-node/}

}

@inproceedings{stackoverflow\_cassandra\_node\_limitations,

title={cassandra node limitations},

author={Gary Lindahl},

url={https://stackoverflow.com/questions/7190573/cassandra-node-limitations},

year={2011}

}

@inproceedings{datastax\_cql\_limits,

title={CQL Limits},

url={https://docs.datastax.com/en/cql-oss/3.x/cql/cql\_reference/refLimits.html#:~:text=Table%20%2F%20CF%20name%20length%3A%2048,65535%20(2%2016%2D1)},

year={2021}

}

@inproceedings{which\_part\_cap\_theorem\_cassandra,

title={Which part of the CAP theorem does Cassandra sacrifice and why?},

author={hawkeye},

url={https://stackoverflow.com/questions/20205797/which-part-of-the-cap-theorem-does-cassandra-sacrifice-and-why},

year={2013}

}

@inproceedings{youtube\_cassandra\_demo,

title={Cassandra Demo},

author={Ali Alkarbi},

url={https://www.youtube.com/watch?v=lE8LI5BvDGA},

year={2017}

}

@inproceedings{object\_serialization\_vs\_relational\_data,

title={Object serialization vs relational data modelling

in Apache Cassandra: a performance evaluation},

author={Valdemar Johansen},

url={http://www.diva-portal.org/smash/get/diva2:839521/FULLTEXT02}

}

@inproceedings{clustering\_order\_in\_cassandra,

title={Clustering Order in Cassandra — how to achieve ordering of data in Cassandra},

author={Anirudh Kanabar},

url={https://medium.com/@anirudhkanabar/clustering-order-in-cassandra-how-to-achieve-ordering-of-data-in-cassandra-f52e8a73d5d5},

year={2019}

}

@inproceedings{cassandra\_data\_types,

title={Data Types},

author={Anirudh Kanabar},

url={https://cassandra.apache.org/doc/latest/cql/types.html},

year={2016}

}

@inproceedings{sorting\_and\_limiting\_results,

title={Sorting and limiting results},

url={https://docs.datastax.com/en/dse/5.1/cql/cql/cql\_using/useQueryColumnsSort.html},

year={2021}

}

@inproceedings{cassandra\_read\_performance,

title={Cassandra read performance},

author={alex07},

url={https://stackoverflow.com/questions/46643437/cassandra-read-performance},

year={2017}

}

@inproceedings{cassandra\_faq,

title={Frequently Asked Questions},

url={https://cassandra.apache.org/doc/latest/faq/},

year={2016}

}

@inproceedings{features\_of\_cassandra,

title={Features of Cassandra},

url={https://www.geeksforgeeks.org/features-of-cassandra/},

year={2019}

}

@inproceedings{casquatch\_an\_open\_source,

title={Casquatch: An Open Source Java Abstraction Layer for Cassandra Databases},

url={https://www.datastax.com/resources/video/datastax-accelerate-2019-casquatch-open-source-java-abstraction-layer-cassandra},

year={2019}

}

@inproceedings{is\_cassandra\_hard\_to\_learn,

title={Is Cassandra hard to learn?},

url={https://www.quora.com/Is-Cassandra-hard-to-learn}

}

@inproceedings{guide\_to\_cassandra\_with\_java,

title={A Guide to Cassandra with Java},

author={baeldung}

url={https://www.baeldung.com/cassandra-with-java},

year={2019}

}

@inproceedings{datastax\_java\_driver\_49,

title={DataStax Java Driver 4.9 (Earlier version)},

url={https://docs.datastax.com/en/developer/java-driver/4.9/manual/mapper/config/kotlin/},

year={2021}

}

@inproceedings{youclap\_kassandra,

title={Kassandra},

author={YouClap},

url={https://github.com/YouClap/Kassandra},

year={2019}

}

@inproceedings{ruby\_driver\_for\_cassandra,

title={Ruby Driver for Apache Cassandra 2.0 (Earlier version)},

url={https://docs.datastax.com/en/developer/ruby-driver/2.0/},

year={2020}

}

@inproceedings{how\_to\_start\_cassandra\_server,

title={how to start cassandra server},

url={https://stackoverflow.com/questions/36664397/how-to-start-cassandra-server},

year={2016}

}

@inproceedings{cassandra\_client\_drivers,

title={Client drivers},

url={https://cassandra.apache.org/doc/latest/getting\_started/drivers.html?highlight=drivers},

year={2016}

}

@inproceedings{cassandra\_guarantees,

title={Guarantees},

url={https://cassandra.apache.org/doc/latest/architecture/guarantees.html#what-is-cap},

year={2016}

}

@inproceedings{cassandra\_multi\_master\_replication,

title={Dynamo: Multi-master Replication: Versioned Data and Tunable Consistency},

url={https://cassandra.apache.org/doc/latest/architecture/dynamo.html#multi-master-replication-versioned-data-and-tunable-consistency},

year={2016}

}

@inproceedings{the\_cassandra\_query\_language,

title={The Cassandra Query Language (CQL)},

url={https://cassandra.apache.org/doc/latest/cql/index.html},

year={2016}

}

@inproceedings{apache\_cassandra\_org,

title={Apache Cassandra},

url={https://cassandra.apache.org/},

year={2021}

}

@inproceedings{application\_connectivity\_with\_cassandra,

title={Application connectivity with Cassandra},

url={https://www.geeksforgeeks.org/application-connectivity-with-cassandra/},

year={2019}

}

@inproceedings{data\_transmission\_between\_nodes\_and\_client\_cassandra,

title={Data transmission between nodes and client cassandra},

url={https://stackoverflow.com/questions/28127182/data-transmission-between-nodes-and-client-cassandra},

year={2015}

}

@inproceedings{wikipedia\_gossip\_protocol,

title={Gossip protocol},

url={https://en.wikipedia.org/wiki/Gossip\_protocol},

year={2021}

}

@inproceedings{towards\_data\_science\_cassandra\_python,

title={Getting started with Apache Cassandra and Python},

author={Adnan Siddiqi},

url={https://towardsdatascience.com/getting-started-with-apache-cassandra-and-python-81e00ccf17c9},

year={2019}

}

@inproceedings{datastax\_node\_js\_driver\_41,

title={DataStax Node.js Driver 4.1 (Earlier version)},

url={https://docs.datastax.com/en/developer/nodejs-driver/4.1/getting-started/},

year={2020}

}

@inproceedings{datastax\_node\_js\_driver\_for\_apache\_cassandra,

title={DataStax Node.js Driver for Apache Cassandra®},

url={https://www.npmjs.com/package/cassandra-driver},

year={2021}

}

@inproceedings{datastax\_c\_plus\_driver\_20,

title={DataStax C/C++ Driver 2.0 (Earlier version)},

url={https://docs.datastax.com/en/developer/cpp-driver/2.0/},

year={2021}

}

@inproceedings{datastax\_cpp\_driver,

title={cpp-driver},

author={datastax},

url={https://github.com/datastax/cpp-driver},

year={2021}

}

@inproceedings{stackoverflow\_how\_much\_data\_per\_node,

title={How much data per node in Cassandra cluster?},

author={user334596},

url={https://stackoverflow.com/questions/4775388/how-much-data-per-node-in-cassandra-cluster},

year={2011}

}