**Big Data Engineering Management Culture Among the FAANG Corporations: Tracing Apache Cassandra’s Legacy at Facebook and its Peer Companies**

Shane Cohen

IST 769 – Advanced Big Data Management

December 16, 2023

Apache Cassandra is a NoSQL-based data engineering software that Facebook developed in the 2000s. My original reason for choosing this software to study and explore was that the software stood out to me visually among all the ten or so different software models that this class introduced me to. The wide-column database returned different columns in different colors, and whether this was a feature of the specific code that the labs provided or a unique feature inherent to the Cassandra packaging, it helped contribute toward making me have a memorably positive initial impression of Cassandra. UX/UI interface features are very important among the factors that make a software effective.

As I started preparing to write this paper and plan my strategy for this data engineering project, I did an extensive literature review and uncovered an interesting business context and history about how Cassandra has been used among different companies. The research started casually. The professor and I had discussed a project where I simulated a scene where a novice or junior engineer accidentally corrupted some data through a command misfire, a lack of proper use of views as opposed to the original table, and or some similar reason. In this scenario, it was my task as a more senior engineer to try to use adept SQL commands to recover a previous version of the data and repair the integrity of the dataset.

I decided to take the project one step higher in complexity by going onto Stack Overflow and researching some actual real-world reported instances of data corruption. How does it actually look when Apache Cassandra databases get corrupted? What happens and why? What common use cases are people reporting, and where does progress stand in the coding community’s collective “hive mind” efforts to patch and debug these problems? That brought the project up to a new level of complexity and ambition. I didn’t see any reported complaints that matched the scenario I thought of, where I had imagined the junior engineer trying to do data cleaning and make a change on one dataframe cell and ending up instead making a massive accidental set of spelling changes or data erasures to a large number of dataframe cells. That doesn’t mean such things don’t happen—it just means that other problems had a greater provenance in the Stack Overflow dataset.

The Stack Overflow results revealed a node of aggregation around a common bug: error messages indicating that the SSTable had become corrupted. Here is one example of what that corruption message looks like:

java.lang.RuntimeException: org.apache.cassandra.io.sstable.CorruptSSTableException: Corrupted: /cassandra/data/data/ams/mydata\_attr\_v1-de4f9960a01711e783ea2bd3a6beadcf/mc-2925-big-Data.db at org.apache.cassandra.service.StorageProxy$DroppableRunnable.run(StorageProxy.java:2490) ~[apache-cassandra-3.9.jar:3.9] at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) ~[na:1.8.0\_72] at org.apache.cassandra.concurrent.AbstractLocalAwareExecutorService$FutureTask.run(AbstractLocalAwareExecutorService.java:164) ~

(NN -- <https://stackoverflow.com/questions/69416334/cassandra-sstable-corruption-how-to-prevent>).

The error message seems to come up for multiple reasons. Forum responses indicate that possible causes of the error can include disk failure, power failure, other improper shutdown of the Cassandra service, attempting to drop a column from the Cassandra table without fully deleting the data from the table at a deeper level, and attempting to recreate a new Cassandra column in place of a deleted column without properly erasing the data from table at a deeper level. As one source writes:

Dropping a column from a table in Cassandra doesn't drop the data. The dropped column is just marked as deleted in a special table, but the data for that column is still there; they are removed before sending a response back to the driver. So when you add the column with the same but another type, when trying to read data, Cassandra will use the wrong codec and fail to decode the data, leading to errors.

The actual removal of the data for the deleted column happens during the compaction process, so you can add a column with the same name only after you make sure that all old data are removed by compaction.

(Source: <https://stackoverflow.com/questions/77359167/cassandra-behavior-when-a-column-dropped-and-reinserted-with-different-datatyp>)

These experienced professionals’ suggested repairs for corrupted SSTables included a lot of commands that did not get mentioned in the live class lecture, the asynchronous video recording, or the labs. There was a lot of mention of shutting down the Cassandra nodes and/or cleaning and resetting the Cassandra software’s memory at a deeper level than what we were exposed to in this class, using commands like *nodetool drain*, *nodetool scrub, sstablescrub*, and *nodetool repair keyspace*.

(Source: <https://stackoverflow.com/questions/26746846/why-is-data-corruption-happen-in-cassandra-1-2>, <https://stackoverflow.com/questions/69416334/cassandra-sstable-corruption-how-to-prevent>, <https://www.gosquared.com/blog/dealing-corrupt-sstable-cassandra>,

I realized at this point that the entire question of Cassandra database corruption was more complicated than I had imagined. There’s an entire other level of code that we did not cover in this class, which makes sense because it’s only a masters program. As I left Stack Overflow behind and searched more broadly for other comments on Cassandra table repair, corruption, and troubleshooting, I found some interesting stories about some innovations Yelp had made to repair its own significant encounters with Cassandra data corruption, as well as some academic writing, some official Apache documentation, and some other extended writing about strategic approaches to Cassandra debugging. Still curious even beyond that point, I went onto Google Scholar to learn more about both the historiography and the advanced methodology of Cassandra. I’ve assembled a lot of those links in the annotated NN bibliography.

As I applied to computer science PhD programs and learned more about what kind of knowledge production these doctorate holders are actually doing, I learned that there’s more than one way to be a computer science scholar. Some people are scholars of machine learning and artificial intelligence; others, robotics; still others, computer vision and computational neuroscience; still yet others, data science. There are some professors who specialize in developing expertise in a specific technology and its methodology, and the exposure that I got to some research like that is largely what motivated the structure of this paper. When the professor challenged us to write a data engineering paper rather than a data science paper, that was a new concept for me. It was really enlightening to learn more about the differences between data engineering and data science by consulting the corpus of published peer-reviewed research.

My USC PhD application challenged me to propose what topic I would choose for my dissertation if I had to decide today. I responded that I would take this paper and expand it, titling the full dissertation very similarly to what I have titled the present paper. The Google Scholar results about Apache Cassandra reveal an interesting story. Cassandra, when Facebook invented it, was a fundamental part of the Facebook private messenger feature. There’s a story behind why Facebook killed the software off and rejected it, and there’s a narrative of why Apple, Yelp, and other companies picked it up and continued to run with it—and even why Facebook itself, at least at one point, resumed using Cassandra again. There’s an interesting narrative about different financial spending power and different hiring practices and engineering productivity/ process cultures among the different FAANG companies, and Cassandra’s features play into that—both its strengths and its weaknesses, such as the SSTable corruption problem that I described above. Together with the historiography, there’s also clear justification for using Apache Cassandra’s features to aggregate social network data en masse and perform complex network theory analysis about idea dissemination and group behavior on these social networks. One of my potential PhD advisors, Dr. Carter Butts of UC-Irvine, has done a lot of research on the uses of computer software to analyze social network website behavior through a prism informed by the sociological theory known as network theory. I’ve represented some of his work in the bibliography, together with much else.

The scope and the complexity of the sources I’ve assembled here could very easily become a dissertation, and I’ve wrestled with a problem that I often wrestle with at Syracuse, which is that the masters program is not designed to help students produce peer-reviewed research, and the majority of my classmate peers, as far as I know are not actively pursuing PhDs or trying to transform their research papers into accepted submissions at peer-reviewed journals. I’ve tried to strike a balance between fleshing out the literature review in depth and conforming more to the general tenor of the default Syracuse data science masters knowledge production culture. I haven’t done a perfect job; yet I am very pleased with the effort, and I think the final product is more impressive than I’m going to be able to realize or appreciate in the present moment. That’s often the case with some of the more ambitious intellectual risks I take.

The following section of the paper demonstrates proficiency in Apache Cassandra, its concomitant SQL operations, and PySpark, by manipulating a dataset of Dave Matthews Band concert data to answer a research question about NN

I took risks

Development in my progress as an academic and a data engineer from 2020 to present

The experience of encountering Facebook’s org behavior culture here is similar to what it felt like when I browsed Goldman Sachs’ departmental map welcome page for undergrads as a college senior and took their personality test advising me which 3 departments within GS I best matched with. I took that test in 2014 and ended up actually working for GS in 2020. So, who knows—maybe this experience was a prelude to ending up at Facebook or another FAANG company one day. I definitely understand why it’s so stressful there that people need to be paid $400,000 a year, etc.

Boundary objects within the data – what might that mean

Which DMB songs have the broadest diversity in the variations of durations which they have been played live at in concerts? Top 10.

How about the most consistency in live song length? Top 10.

Among rare songs that have only been played once, what’s their average length—are they long or short?

Dave Matthews’ performances at the Gorge in central Washington State every year are so famous that Labor Day Weekend at the Gorge is now informally known as Labor Dave Weekend. When was the first year that Labor Dave Weekend started?

I know that Dave Matthews has overall stopped playing at Salt Lake City and other Utah venues. What are the other cities where Dave Matthews has stopped playing?

Does a Dave Matthews and Tim Reynolds variant of a song tend to be longer than a Dave solo version of a song?

Boyd Tinsley, the violinist, was fired from Dave Matthews Band. I was surprised to see his solo performances post-firing included in the database. What are the names of some of his songs, and are they longer or shorter than the overall average in the data corpus?

-Data cleaning for sheet names 🡪 this became dividing each sheet into its own csv file and making sure the column names have no spaces within them DONE

-add a column to the master record file with an indicator mark indicating that it’s one of the original records DONE

-write GPT-4 code for how to create 2-3 times the original content of the master record through random replication DONE

-purge out the 2-3 times extra created content

-merge all the csvs into one using join statements

-run data science inquiries as appropriate

I took some screenshots of some aspects of my work process. There is extensive further commentary in the two Jupyter Notebooks, dmb.ipynb and GPT-4 Consult.ipynb

Une image contenant texte, logiciel, Logiciel multimédia, Appareils électroniques

Description générée automatiquementUne image contenant Appareils électroniques, Logiciel multimédia, logiciel, capture d’écran

Description générée automatiquementUne image contenant texte, logiciel, Logiciel multimédia, capture d’écran

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, logiciel, Appareils électroniques, Logiciel multimédia

Description générée automatiquementUne image contenant texte, logiciel, Logiciel multimédia, Appareils électroniques

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, Logiciel multimédia, Police, logiciel

Description générée automatiquementUne image contenant texte, capture d’écran, Page web, logiciel

Description générée automatiquementUne image contenant texte, capture d’écran, Police, Page web

Description générée automatiquementUne image contenant texte, logiciel, nombre, capture d’écran

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

**Annotated Bibliography**

*First Round of Research: Consultation of primarily StackOverflow sources to learn more about ways that Cassandra tables can become corrupted. Learned also about best practices wisdom on how to repair the corrupted tables.*

(A) SSTable corruption can occur due to: (1) Abrupt shutdown of Cassandra node due to power failure or manual shutdown. (2) Disk failure. (3) Failing to use the *nodetool drain* command before stopping Cassandra manually. (We didn’t use *nodetool drain* in our labs and everything worked just fine, although we never had to re-open the SSTables from our labs after the first shutdown—so maybe that’s why we didn’t encounter a problem.)

<https://stackoverflow.com/questions/69416334/cassandra-sstable-corruption-how-to-prevent>

(B) One of several mentions of the *nodetool drain* command as a way to repair corrupted Cassandra memory and debug the SSTable corruption error.

<https://stackoverflow.com/questions/26746846/why-is-data-corruption-happen-in-cassandra-1-2>

(C) Dropping a column from a table in Cassandra doesn't drop the data, which can cause the SST Table corruption error. The theme as a whole among the sources is that the SSTable corruption error message is very common.

<https://stackoverflow.com/questions/77359167/cassandra-behavior-when-a-column-dropped-and-reinserted-with-different-datatyp>

(D) Instructions for scrubbing and removing corrupted tables

<https://www.gosquared.com/blog/dealing-corrupt-sstable-cassandra>

(E) Guidance for repairing a corrupted Cassandra table

<https://dba.stackexchange.com/questions/330388/problem-with-cassandra-data-corruption-inconsistency>

*Second Round of Research: Continued research into existing documentation on Cassandra debugging, repair, recovery, and version control procedures*

(F) Backup and Recovery Mechanisms of Cassandra Database: A Review

<https://commons.erau.edu/cgi/viewcontent.cgi?article=1613&context=jdfsl>

(G) Yelp Rebuilds Corrupted Cassandra Cluster Using Its Data Streaming Architecture <https://www.infoq.com/news/2023/07/yelp-corrupted-cassandra-rebuild/>

(H) Guide to Apache Cassandra® Data and Disaster Recovery <https://www.instaclustr.com/blog/apache-cassandra-data-and-disaster-recovery/>

(I) Rebuilding a Cassandra cluster using Yelp’s Data Pipeline <https://engineeringblog.yelp.com/2023/01/rebuilding-a-cassandra-cluster-using-yelps-data-pipeline.html>

(J) Cassandra Operating and Repair Instructions from Official Apache Documentation <https://cassandra.apache.org/doc/stable/cassandra/operating/repair.html>

(K) Troubleshooting Cassandra

<https://docs-previous.pega.com/decision-management/86/troubleshooting-cassandra>

(L) Troubleshooting Cassandra File System <https://www.datastax.com/blog/troubleshooting-cassandra-file-system>

*Third Round of Research: Google Scholar Results Re: Apache Cassandra Methods and History*

(M) Apache Cassandra -- this article mentioned it was developed and then abandoned by Facebook, so in future writing I can work in one of my potential PhD advisor's articles about Facebook and talk about the network theory relevance of Cassandra <https://immagic.com/eLibrary/ARCHIVES/GENERAL/WIKIPEDI/W120911A.pdf>

(N) A Big Data Modeling Methodology for Apache Cassandra (2015) <https://ieeexplore.ieee.org/abstract/document/7207225>

Abstract: Apache Cassandra is a leading distributed database of choice when it comes to big data management with zero downtime, linear scalability, and seamless multiple data center deployment. With increasingly wider adoption of Cassandra for online transaction processing by hundreds of Web-scale companies, there is a growing need for a rigorous and practical data modeling approach that ensures sound and efficient schema design. This work i) proposes the first query-driven big data modeling methodology for Apache Cassandra, ii) defines important data modeling principles, mapping rules, and mapping patterns to guide logical data modeling, iii) presents visual diagrams for Cassandra logical and physical data models, and iv) demonstrates a data modeling tool that automates the entire data modeling process.

(O) **Streaming Data Analysis using Apache Cassandra and Zeppelin** (2015) <https://ijiset.com/vol3/v3s10/IJISET_V3_I10_02.pdf>

Abstract: Big data is a popular term used to describe the large volume of data which includes structured, semi-structured and unstructured data. Now-a-days, unstructured data is growing in an explosive speed with the development of Internet and social networks like Twitter,Facebook & Yahoo etc., In order to process such colossal of data a software is required that does this efficiently and this is where Hadoop steps in. Hadoop has become one of the most used frameworks when dealing with big data. It is used to analyze and process big data. In this paper, Apache Flume is configured and integrated with spark streaming for streaming the data from twitter application. The streamed data is stored into Apache Cassandra. After retrieving the data, the data is going to be analyzed by using the concept of Apache Zeppelin. The result will be displayed on Dashboard and the dashboard result is also going to be analyzed and validating using JSON.

(P) Real-Time Data Analysis with Cassandra and Spark (2018) <https://link.springer.com/chapter/10.1007/978-981-13-0550-4_11>

**Abstract**: The dynamic progress in the nature of pervasive computing datasets has been main motivation for development of the NoSQL model. The devices having capability of executing “Internet of Things” (IoT) concepts are producing massive amount of data in various forms (structured and unstructured). To handle this IoT data with traditional database schemes is impracticable and expensive. The large-scale unstructured data required as the prerequisites for a preparing pipeline, which flawlessly consolidating the NoSQL storage model such as Apache Cassandra and a Big Data processing platform such as Apache Spark. The Apache Spark is the data-intensive computing paradigm, which allows users to write the applications in various high-level programming languages including Java, Scala, R, Python, etc. The Spark Streaming module receives live input data streams and divides that data into batches by using the Map and Reduce operations. This research presents a novel and scalable approaches called "**Smart Cassandra Spark Integration (SCSI)**” for solving the challenge of integrating NoSQL data stores like Apache Cassandra with Apache Spark to manage distributed systems based on varied platter of amalgamation of current technologies, IT enabled devices, etc., while eliminating complexity and risk. In this chapter, for performance evaluations, SCSI Streaming framework is compared with the file system-based data stores such as Hadoop Streaming framework. SCSI framework proved scalable, efficient, and accurate while computing big streams of IoT data.

(Q) Twitter Data Modelling and Provenance Support for Key-Value Pair Databases (2021)   
<https://link.springer.com/chapter/10.1007/978-3-030-69377-0_8>

Abstract: In Big Data environments, reliability of data plays an important role to determine trustworthiness of the outcomes of an analysis. Big data provenance ensures the reliability of data by providing details about the origin and historical paths of data. In recent years, the preponderance of big data and its applications are increasingly using Apache Cassandra due to its high availability and linear scalability. In this paper, we present a data provenance framework for Key-Value Pair Databases using the concept of Zero-Information Loss Database (ZILD). A large volume of real-time social media data is fetched from the Twitter’s network through live streaming with the help of Twitter Streaming APIs, and then modelled in Apache Cassandra based on a Query-Driven approach. This framework provides efficient provenance capturing support for select, aggregate, update, and historical queries. We evaluate the performance of proposed framework in terms of provenance capturing and querying capabilities using appropriate query sets.

(R) An Evaluation of Cassandra for Hadoop (2013) <https://ieeexplore.ieee.org/abstract/document/6676732>

Abstract: In the last decade, the increased use and growth of social media, unconventional web technologies, and mobile applications, have all encouraged development of a new breed of database models. NoSQL data stores target the unstructured data, which by nature is dynamic and a key focus area for "Big Data" research. New generation data can prove costly and unpractical to administer with SQL databases due to lack of structure, high scalability, and elasticity needs. NoSQL data stores such as MongoDB and Cassandra provide a desirable platform for fast and efficient data queries. This leads to increased importance in areas such as cloud applications, e-commerce, social media, bioinformatics, and materials science. In an effort to combine the querying capabilities of conventional database systems and the processing power of the MapReduce model, this paper presents a thorough evaluation of the Cassandra NoSQL database when used in conjunction with the Hadoop MapReduce engine. We characterize the performance for a wide range of representative use cases, and then compare, contrast, and evaluate so that application developers can make informed decisions based upon data size, cluster size, replication factor, and partitioning strategy to meet their performance needs.

(S) A Deep Learning Approach to Fake News Detection (2020) <https://link.springer.com/chapter/10.1007/978-3-030-59491-6_11>

Excerpt: The uncontrolled growth of fake news creation and dissemination we observed in recent years causes continuous threats to democracy, justice, and public trust. This problem has significantly driven the effort of both academia and industries for developing more accurate fake news detection strategies. Early detection of fake news is crucial, however the availability of information about news propagation is limited. Moreover, it has been shown that people tend to believe more fake news due to their features [[11](https://link.springer.com/chapter/10.1007/978-3-030-59491-6_11" \l "ref-CR11)]. In this paper, we present our framework for fake news detection and we discuss in detail a solution based on deep learning methodologies we implemented by leveraging Google Bert features. Our experiments conducted on two well-known and widely used real-world datasets suggest that our method can outperform the state-of-the-art approaches and allows fake news accurate detection, even in the case of limited content information

(T) Cassandra is a Better Option for Handling Big Data in a No-SQL Database (2022) <https://www.researchgate.net/profile/Ramesh-Byali/publication/363653323_Cassandra_is_a_Better_Option_for_Handling_Big_Data_in_a_No-SQL_Database/links/6328275f873eca0c009b2585/Cassandra-is-a-Better-Option-for-Handling-Big-Data-in-a-No-SQL-Database.pdf>

(U) Scalable Archiving with the Cassandra Archiver for CSS (2013) <https://accelconf.web.cern.ch/ICALEPCS2013/papers/tuppc004.pdf>

Abstract: An archive for process-variable values is an impor- tant part of most supervisory control and data acquisition (SCADA) systems, because it allows operators to investi- gate past events, thus helping in identifying and resolving problems in the operation of the supervised facility. For large facilities like particle accelerators, there can be more than one hundred thousand process variables that have to be archived. When these process variables change at a rate of one Hertz or more, a single computer system can typi- cally not handle the data processing and storage. The Cas- sandra Archiver has been developed in order to provide a simple to use, scalable data-archiving solution. It seam- lessly plugs into Control System Studio (CSS), providing quick and simple access to all archived process variables. An Apache Cassandra database is used for storing the data, automatically distributing it over many nodes and provid- ing high-availability features. This contribution depicts the architecture of the Cassandra Archiver and presents perfor- mance benchmarks outlining the scalability and compar- ing it to traditional archiving solutions based on relational databases.

(V) NoSQL Databases as Social Networks Storage Systems (2017) <https://hrcak.srce.hr/ojs/index.php/entrenova/article/view/14100>

Abstract: The paper presents analysis of the storage systems used by social network sites. Namely, the social networks are one of the main driving forces behind the NoSQL database development. Facebook and Twitter were, together with other the Big Data players like Google and Amazon, first faced with the limitations of relational databases in solving their needs related to unprecedented transaction volumes, expectations of low-latency access to massive datasets, and nearly perfect service availability while operating in an unreliable environment. The first NoSQL databases arose as internal solutions created out of necessity, and not with the intention to abandon relational databases. But the main question is if, after more than ten years of development, NoSQL databases proved that they could be valuable storage solutions for social networks’ data. The paper shows that there is still a lot of room for improvement in the use of NoSQL in social networks and provides some suggestions on how NoSQL databases can bring additional value to social network sites.

(W) Scalability Issues in Online Social Networks (2016) <https://dl.acm.org/doi/abs/10.1145/2968216>

* + Abstract: The last decade witnessed a tremendous increase in popularity and usage of social network services, such as Facebook, Twitter, and YouTube. Moreover, advances in Web technologies coupled with social networks has enabled users to not only access, but also generate, content in many forms. The overwhelming amount of produced content and resulting network traffic gives rise to precarious scalability issues for social networks, such as handling a large number of users, infrastructure management, internal network traffic, content dissemination, and data storage. There are few surveys conducted to explore the different dimensions of social networks, such as security, privacy, and data acquisition. Most of the surveys focus on privacy or security-related issues and do not specifically address scalability challenges faced by social networks. In this survey, we provide a comprehensive study of social networks along with their significant characteristics and categorize social network architectures into three broad categories: (a) centralized, (b) decentralized, and (c) hybrid. We also highlight various scalability issues faced by social network architectures. Finally, a qualitative comparison of presented architectures is provided, which is based on various scalability metrics, such as availability, latency, interserver communication, cost of resources, and energy consumption, just to name a few.

(X) Out in the Open: The Abandoned Facebook Tech That Now Helps Power Apple <https://www.wired.com/2014/08/datastax/> and <https://www.dataversity.net/cassandra-went-abandoned-facebook-powering-likes-apple/>

* + [Klint Finley of Wired.com](http://www.wired.com/2014/08/datastax/) reports, “Though Facebook has all but abandoned Cassandra, the technology has gone on to power critical web infrastructure at companies like Twitter, Netflix, even Apple. And DataStax has built a version of the tool for all sorts of other businesses. Having raising over $84 million, the startup now spans over 300 employees, and it’s [well on its way to an IPO](http://techcrunch.com/2013/07/23/with-ipo-in-its-sights-datastax-raises-45m-for-fast-growing-nosql-database-platform/), landing over 500 customers, including 25 of the Fortune 100, according to Ellis.”
  + Finley goes on, “Facebook engineers Avinash Lakshman and Prashant Malik originally built Cassandra to power the engine that let you search your inbox on the social network. Like other so-called “NoSQL” databases, it did away with the traditional relational model—where data is organized in neat rows and columns on a single machine—in order to more easily scale across thousands of machines. That’s vitally important for a growing web service the size of Facebook. Lakshman had worked on Amazon’s distributed data storage system called Dynamo, but the two also drew inspiration from a paper Google published in 2006 describing its internal database BigTable.”
  + He adds, “Mark Zuckerberg and company open sourced Cassandra in the summer 2008, and it helped kick off the now enormous NoSQL movement, along with other databases like CouchDB and MongoDB. Rackspace hired Ellis that very year to evaluate options for a next-generation database, and he tried all the various NoSQL databases available at the time. None, he says, could top Cassandra. ‘Facebook open sourced it, but weren’t moving it forward,’ he says. ‘But the technical foundations were ahead of everyone else’.”

(Y) Facebook’s financials versus other FAANG companies’ financials  
<https://dealbreaker.com/2018/06/there-are-major-differences-between-facebook-and-the-other-faang-stocks-just-look-at-the-financials#:~:text=Returns%20compared%20to%20Alphabet%2C%20Netflix%2C%20Apple%2C%20and%20Amazon.&text=As%20you%20can%20see%2C%20Facebook,remainder%20of%20the%20FAANG%27s%20median.>

*Fourth Round of Research: Facebook’s Engineering Culture*

(Z) Inside Facebook's Engineering Culture: Part 1 <https://newsletter.pragmaticengineer.com/p/facebook>

(AA) Inside Facebook's Engineering Culture: Part 2 <https://newsletter.pragmaticengineer.com/p/facebook-2>

(AB) Engineering at Meta: Engineering Culture: Code ownership <https://engineering.fb.com/2014/10/28/culture/engineering-culture-code-ownership/>

(AC) Facebook's 6 Staff Engineer Archetypes <https://www.linkedin.com/posts/jgalvin_its-often-difficult-for-those-with-no-experience-activity-6978326270086389761-A6J1#:~:text=The%206%20Staff%2B%20Engineer%20Archetypes,Engineer%20newsletter%20by%20Gergely%20Orosz.>

(AD) Engineering at Meta Culture page   
<https://engineering.fb.com/category/culture/>

(AE) YCombinator Facebook culture review (2017) <https://news.ycombinator.com/item?id=13316569>

(AF) Engineering Culture and Hiring at Facebook <https://www.wecreateproblems.com/blog/engineering-culture-and-hiring-at-facebook>

(AG) How Facebook works: Comparing its engineering process to Google, Microsoft, and Amazon   
<https://www.geekwire.com/2021/facebook-works-comparing-engineering-process-google-microsoft-amazon/>

(AH) Secret to Facebook's Hacker Engineering Culture <https://launchdarkly.com/blog/secret-to-facebooks-hacker-engineering-culture/>

*Fifth Round of Research: What did Meta replace Cassandra with?*

(AI) The Underlying Technology of Messages (2010) <https://engineering.fb.com/2010/11/15/core-infra/the-underlying-technology-of-messages/>

(AJ) Quora: Why did Facebook move from Apache Cassandra to MySQL and then again to Apache Cassandra in their backend?   
<https://www.quora.com/Why-did-Facebook-move-from-Apache-Cassandra-to-MySQL-and-then-again-to-Apache-Cassandra-in-their-backend>

*Sixth Round of Research: Inquiry into the Network Theory/ Social Network Studies Research of a potential PhD advisor, Dr. Carter Butts of UC-Irvine. Tying in this theory adds some oomph to a paper about studying Apache Cassandra in the context of its Facebook history*

(AK) [A relational event framework for social action](https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9531.2008.00203.x)  
  
(AL) [Practical recommendations on crawling online social networks](https://ieeexplore.ieee.org/abstract/document/6027868/)

(AM) [A walk in facebook: Uniform sampling of users in online social networks](https://arxiv.org/abs/0906.0060)

(AO) [Walking in facebook: A case study of unbiased sampling of osns](https://ieeexplore.ieee.org/abstract/document/5462078/)

*Seventh Round of Research: Inquiry resulting from Learning about the Dissertation Called “Boundary Objects: The Key to Hidden Knowledge in Social Media Streams” (2016)*

(AP) “Boundary Objects: The Key to Hidden Knowledge in Social Media Streams” (2016) <https://www.proquest.com/openview/2c40452a59cfb021b74d5b5fd0a6f5c4/1?pq-origsite=gscholar&cbl=18750>

(AQ) Revisiting the notion of boundary object <https://journals.openedition.org/rac/18243>

(AR) The emerging concept of boundary objects in knowledge management <https://realkm.com/2019/12/06/the-emerging-concept-of-boundary-objects-in-knowledge-management/>

(AS) Boundary Objects, Social Meanings and the Success of New Technologies <https://journals.sagepub.com/doi/10.1177/0038038510387196>