# EPISODE 874

[INTRODUCTION]

**[0:00:00.0] JM:** Facebook generates high volumes of data at a rapid pace. Dhruba Borthakur joined Facebook in 2008 to work on data infrastructure. His early projects at Facebook were around Hadoop, the distributed file system and map reduced computation platform that laid the foundation for the big data movement.

At the time, Facebook was generating as much data as any other startup and the company needed to stay at the leading edge of scalability techniques for its Hadoop distributed file system cluster. Traditionally, Hadoop managed its file system by synchronizing the coordination of the different data nodes with the help of a single master node.

At Facebook, the scale of the data was such that the HDFS cluster had thousands of data nodes. This was too much volume for a single master node to handle. Dhruba helped implement redundancy at the master node to create a more resilient system. The early days of the big data movement were focused on batch processing.

A company like Facebook would gather large amounts of data into databases and HDFS and run offline analytics workloads to gather reports on an hourly, daily, or weekly basis. Over time, data infrastructure has moved closer to a real time processing model. Data infrastructure does not only support batch offline reporting, it also supports machine learning jobs that need to be run on a more frequent basis. These jobs have lower latency requirements and have driven the adoption of in-memory stream processing systems like Spark and Flank.

Dhruba joined the show to discuss his time at Facebook building data infrastructure. He takes us through the major projects he worked on, including the early Hadoop infrastructure. The refactoring of online user workloads to be more pull based than push based and the creation of RocksDB. A storage engine he helped create at Facebook.

Today, Dhruba is the CTO and cofounder of Rockset. A company that builds data infrastructure and database API's on top of RocksDB. Rockset is building infrastructure for modern technology companies. Many of which are facing problems that bear significant resemblance to the ones that Facebook encountered as it scaled.

We have new apps for iOS and Android. These apps are a great listening experience for Software Engineering Daily. We've spent a lot of time refactoring them and improving them and they include all 1,000 of our old episodes, including all of our shows about Facebook and its history. You can comment on episodes, you can have discussions with members of the community. I will be commenting on each episode, going forward for a while. If you have some commentary, you have some feedback, you have some thoughts on the shows, I would love to get into a discussion with you.

You can jump on the apps or jump on softwaredaily.com to share your thoughts and if you want to become a paid subscriber to get ad-free episodes, you can go to softwareengineeringdaily.com/subscribe. Also, FindCollabs is a place to find collaborators and build projects. FindCollabs is the company I'm building and we're having an online hackathon with 2,500 in prizes.

If you are looking for a collaborator or a cofounder or you just have a cool project that you want to share with people and maybe it's open source, maybe it's not, maybe it's about software, maybe it's about music. Whatever it is, you can find people to show it to or work on it with, at FindCollabs.

With that, let's get on to today's show.

[SPONSOR MESSAGE]

**[0:03:49.1] JM:** When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product like me or, an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer, we've also done several shows with the people who run G2i. Gabe Greenberg and the rest of his team, these are engineers who know about the React ecosystem about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget and the G2i staff are friendly and easy to work with. They know how product development works, they can help you find the perfect engineer for your stack and you can go to softwareengineering.com/g2i to learn more about G2i. Thank you to G2i for being a great supporter of software engineering daily, both as listeners and also as people who have contributed code that have helped me out in my projects.

If you want to get some additional help for your engineering projects. Go to softwareengineeringdaily.com/g2i.

[INTERVIEW]

**[0:05:40.5] JM:** Dhruba Borthakur, you are the CTO at Rockset, you are a long time engineer at Facebook, welcome back to Software Engineering Daily.

**[0:05:47.4] DB:** Thank you, thanks a lot Jeff.

**[0:05:49.0] JM:** You joined Facebook in 2008. Describe the open source data ecosystem back then?

**[0:05:57.2] DB:** Facebook used quite a few pieces of open source software at the time, I was mostly focused on the big data side of things at the time so my focus was mostly about Hadoop and Hive and some other related big data technologies. First we became a very big user of Hadoop which is open source software and also it became a good contributor of code to Hadoop as well as Hive. The Hive project was an open source project that was actually founded at Facebook and it was an SQL query engine built on top of Hadoop. It became very popular over the last 10 years.

**[0:06:33.7] JM:** You've been a committer to the HDFS project for 13 years. Describe how HDFS has matured.

**[0:06:42.5] DB:** Yeah, HDFS has been a very exciting project, in my mind. I started working on the HDFS, which is the Hadoop File System project. In 2006, even before I joined Facebook, I was at Yahoo and Yahoo was trying to figure out what is the best way to compete with all the search technology that Google was building at the time. And so they decided to build this Hadoop or decided to fund this project called the Hadoop project.

I was probably the first or second guy inside Yahoo who was a committer toward the Hadoop project at the time. The project started very small, I still remember the first day when there was an intern in the Yahoo search team and he was the first user of Hadoop and I was working with him closely to see how he can get some on the search indices that's built using the Hadoop technology.

I spent my early year or two at Yahoo, building a lot of Hadoop file system stuff. This was a time when probably only like 10 people in the world were using Hadoop or like creating Hadoop software, all in Yahoo and then after that, I move to Facebook and then Facebook became a very big user of Hadoop.

The team grew from like probably, grew up to a size of 10 within a year or so. All of them doing Hadoop development, Hadoop support, setting up clusters and Hadoop, the first Hadoop cluster Facebook was all under 20 nodes I think. I remember that time, there was the work that we were doing, using Hadoop at Facebook, the first piece of work was some reporting software and that reporting software used to take many hours using [inaudible].

When we moved to the Hadoop software, it probably took around maybe 20 minutes or so. People realized the impact of Hadoop that saying hey, we can process a lot of data in quite a short time period. That was exciting.

**[0:08:35.4] JM:** The Facebook Hadoop clusters had tens of petabytes. What kinds of scalability bottlenecks do you hit when you deploy a Hadoop cluster of that scale?

**[0:08:47.2] DB:** Yeah, that is a great question. Like I said, in the beginning, Hadoop was actually designed to scale out, right? The whole point of Hadoop was that hey, you can store a lot of data, you can process a lot of data. The focus wasn't on how quickly you could process but you can actually process terabytes and petabytes of data.

Design was pretty clear cut when we installed it on say, like the first pseudo production system. Like Siri, give the 20 node Hadoop cluster, each node used to have like a one terabyte of disk, it was like 20 terabytes, that was very big at the time, right? I'm talking about like 2006 probably or 2007 and then disks started to become very big, it became three terabytes per disk and then eight terabytes per disk and then also, Hadoop had the capability to scale out as far as the number of nodes were concerned.

At Facebook, I think the largest Hadoop cluster at that time, again, some of these things might have been superseded now but in the early – around 2010 or 2011, the largest sizes were probably like 3,000 or 4,000 nodes in the Hadoop cluster. Each one of them having maybe eight, like six disks each of eight terabytes each. That's a lot of storage. That's many terabytes of data which you would process using Hadoop map reduced jobs.

Scalability challenges, obviously Hadoop in the early days had a single thing called name node, which was the Hadoop master node and that was a scalability challenge because if you have 4,000 nodes, all of them talking to one master node, sometimes the master node doesn't scale very well. There are a lot of work that we did to kind of offload software work area from the master node to do more on the slaves or do more on the secondary node processing and stuff like that.

That time, the master node also did not have a replica which means that if the master died, it was a single point of failure so it was a problem for scaling to large systems. Very quickly we built high availability system for the master node of which means that if the master dies, some secondary main node takes over its work, that was very basic scalability challenge.

The other ones are more about how can you keep – when you have 4,000 nodes or when you have 5,000 nodes, there are always some nodes that need repair. For example, right? You need to like replace disks or you need to fix its memory. How can I do that in an online system so that the rest of the system can continue to work.

This becomes more and more important when you scale out because if you have two nodes, it's very easy to keep the two nodes up and alive, when you're 4,000 nodes, it's just not possible to keep all 4,000 nodes up and alive all the time.

Scalability issues came up where you need to do online repair of systems, there are features that are built in to Hadoop called like decommissioning of nodes. You can run some software, you can give a command saying hey, Hadoop, I'm going to remove this node in the next 10 minutes, please do whatever you need to do to make systems stable enough.

There was manual commands over time, those manual commands became automatic, which means that operator for a hardware operator to run a Hadoop cluster became easier and easier. Yeah, there were many scalability challenges at the time. I just spoke about two of them now but there were plenty more others.

**[0:12:07.1] JM:** When you joined the Facebook, you had been in the software industry for several years, before Facebook, including Yahoo as you mentioned. Were there unique aspects of Facebook's workloads, was there something new about the volume of data or the type of data coming in to Facebook that was unique compared to what you had seen before?

**[0:12:29.5] DB:** Yeah, how could I categorize it, that's very unique. The most unique part of this data that was coming – that Hadoop was handling or some of these big data systems were handling were very different from data that other systems were handling previous to the web kind of period, right?

All your days, before I started to work at Hadoop, at Yahoo and Facebook. Most people, when they talk about data, there's talk that data needs – if somebody's generating data, that needs to be stored all the time and it cannot afford to lose any of these data, you cannot have some loss on any of these data, that's how echo and other systems became very popular in the early days.

Saying that hey, if you have data, you need to store it reliably, availability, performance and all that stuff. For Hadoop, the use case was very different, this was essentially large volumes of data, which was likely lower in quality, which means that even if you lose .1% of the data, it's fine because most of the time, you are doing analytics on this data. You see what I'm saying?

**[0:13:29.0] JM:** Absolutely.

**[0:13:30.5] DB:** The characters think of this data was very different from previous data, that previous generation database systems used to store. This day are more about click logs, people, when they go to their Yahoo website or the Facebook website, they click on things and those generated click logs, they need to be stored because these companies, they need to mine these click logs and find out what people are doing, what people are more interested in. How can it made their old products better by having a very data driven decision making process.

To do this decision making processes, they needed to collect all these data bunt then, if a little bit of data gets lost or gets corrupted or gets missing, they don't really care because if you have 99.9% of the remaining data, you can make good analytics decision out of that data.

Hadoop was built for this purpose. A very different system from previous generation database systems. Like I said, he just made things very easy to store a large volume of data.

**[0:14:26.9] JM:** The last decade has been a golden age of data infrastructure projects. It hasn't just been Hadoop and HDFS. We've seen Kafka and Storm and Spark and Flank and Airflow. There were so much technology that needed to be built to deal with this world of big data. How did you prioritize which areas of infrastructure to focus on within Facebook at a given time?

**[0:14:55.3] DB:** That's a great question. In the beginning part of the Facebook life history, the focus was all about growth which means that how can you make Facebook faster or the Facebook be adopted by more and more users. That was the focus of the company, how can you make hundred million people use Facebook, the focus was more about growing the number of users by adding value to these users lives.

The focus of data collection was more about how can you improve the product, how can you make the facebook product matter? How do you know whether people are looking more at photos or people are looking more at videos for example, right?

Are people doing more of comments on your friend's post or are the most reading news articles at Facebook. This kind of decision making process was very useful to make the Facebook product better. This is where Hadoop came into the picture, saying that hey, I can actually look at all these analytics and make the Facebook app better and better over time.

Even things like how fast are you, these Facebook pages loading on a web browser. At that time, web browsers were quite popular. We also found that using Hadoop that if you can make your page loading 10 times faster, you'll get hundred times more users into your system. It was amazing to find these things.

You could also find clustering of users. Can you cluster all the people in the world, find like places where there's not much connection between this cluster of people and that cluster of people? This is the connectivity papers that came out after post analytics. All these analytics were done using data that was collected.

The focus of this data was all about making product better. Towards the later part of Facebook's history when I was there, that focus continued but then related focus was, how can you find like more deep insights into your users for example. Hey, this is a user who likes sports pages or this is the user who likes fitness regime. These kinds of meta data questions which are even higher level.

Needed to be built so that you can kind of monetize the product better. To do those, a lot of processing is required. The first level like I said, the first version of the software was built to collect a lot of data and make kind of lightweight predictions and more deterministic predictions but then the next generation of data that got collected and processed, needed a lot of CPU also, along with storage.

To be able to find these meta data trends as I call it. Things that are hidden in this data but they need a lot of processing to find these things or find these relationships or find these decision making process. The processing kind of changed over time, that's how far Hadoop is concerned. Other big data change that's happening. Facebook was there.

In the early days of Hadoop, the data was more about batch analytics which means he run jobs on data that is collected maybe yesterday. And you run all your analytics on yesterday's data. That's what I call batch analytics and this is what Hadoop is very good at.

Over time, Facebook also started to use a system called Log Device, which is similar to a [inaudible] system but it lets you do more real time analytics, which means that you can actually [inaudible] data that got produced maybe an hour back or maybe 10 minutes back. So it's more about real time analytics that people has moved onto and then recently in last maybe two or three years before I quit Facebook, that time the focus was more about live analytics.

Which means that can you run analytics on data that just got produced immediately in the last few minutes or few seconds. So the kind of systems that were built to process live analytics was very different from Hadoop and the Kafka like systems, which are most to very far batch and real time analytics in the earlier time sense.

[SPONSOR MESSAGE]

**[0:18:56.0] JM:** ExpressVPN is a popular virtual private network. ExpressVPN is useful for getting a private, secure, anonymous connection for your internet browsing. It encrypts your data and it hides your public IP address. You've got easy to use apps that run seamlessly in the background on your computer, or your phone, or your tablet, and turning on the ExpressVPN protection only takes a single click.

If you use ExpressVPN, you can safely surf on public Wi-Fi without being snooped on or having your personal data or your bitcoin account information stolen. For less than $7 a month, you can get ExpressVPN protection. ExpressVPN is the number one VPN service rated by Tech Radar, and it comes with a 30-day money-back guarantee. You can also support Software Engineering Daily, if you check out expressvpn.com/sedaily, you would get three months free.

Everybody needs a VPN at some point in their lives. So, if you want to get your ExpressVPN subscription for free for three months while also trying out ExpressVPN and supporting Software Engineering Daily, you can kill all those birds with one stone by going to expressvpn.com/sedaily. Thanks to ExpressVPN for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

**[0:20:34.9] JM:** Today, Facebook feels like a completely live application. So I interact with it and very quickly, the entire application changes its data model in response to my interaction and it feels seamless but back in the day, in 2008, it was not that seamless. It would take a while for your interactions to be reflected in terms of maybe how your feed gets constructed. There was a longer delay between an interaction and the feedback of the application.

But today, machine learning models are clearly working on that data very quickly. Describe some of the engineering difficulties in creating Facebook as a fast data application in contrast to the slower batch application that it was in the past.

**[0:21:28.5] DB:** Oh yeah, this is a great question again because I remember very clearly how this transitioned happened. So in the very early days, the Facebook feed, this is the things that you see in the Facebook app, they were bait as a push system, which means that if I post a message, the message goes to the Facebook backend data engine and then it tries to find out who are the friends that you are connected to and then post this message to your friends right then and there, right?

And then it doesn't matter whether your friends are logged in or not. They just posted to some queues that are waiting. So as soon as your friend logs in, you can see those posts. So it is kind of a push based system. Essentially the processing happens when you post new data or when new content is created. This is what I mean by more like a real time system where things happen when new data gets created, right? But then this process doesn't scale out and it gives you hiccups.

Essentially when this queue, the pending queue that you produced is long and when some user logs in, he has to drain this queue before he sees the most recent items that his friends posted because those are at the bottom of your queue. So over time, Facebook moved to a more live system, which means that when somebody posts a message it just gets deposited in your database. It doesn't get posted to any of your friend's queues.

It just gets posted to a database and when a friend logs in, at the time the queries made to all the database is saying, "Tell me all the latest things that all my friends have done and then write them online." So this is what I mean by live system where it's more like a pull based system. It is kind of the processing is happening one at each time to read the data when people need to access information not at the time when this data was created. Do you see the difference between these two models?

**[0:23:15.2] JM:** Of course.

**[0:23:15.6] DB:** Yeah, the older version was more push-based. When you get it rising, do a lot of processing and keep it ready so that when somebody needs it, it's ready for him. That's more like a streaming real time system that people are familiar with now but that system doesn't scale and it gives you a little bit of scale data when there's a lot of rights coming into your system, right? And also you do a lot of processing that goes to waste.

Because if none of your friends are logged in there is no need to build queues for them because they will never log in for the next seven days, right? And so the queue that you have built is all wasted on compute and resources. Whereas if you move to a more pull base system, which is a live system where the Facebook newsfeed gets generated when somebody logs in. So when somebody logs in you make a query to all your data sources and at that time, rank them online within a few milliseconds.

Sort them and get some relevance, run some user go fish and our guardians to figure out which posts are relevant to the user and then show them at the time of when they are refreshing their feed for example. So these two models, the pull based model really let Facebook data be very live, which means that the moment you go to your page, you see the most relevant date, things that are posted by your friends within the last few seconds or maybe five seconds.

So that really helped get engagement up because people love to see things that happened immediately. It is more like a messaging system now rather than static webpage system that was there in the earlier days.

**[0:24:48.2] JM:** You have as much experience in data infrastructure, modern data infrastructure as anybody I have spoken to. Do you think that all batch system will eventually be replaced by streaming systems?

**[0:25:03.7] DB:** That's a great question again because I think a lot of systems will remain as batch systems because there are certain reporting things that you need to do. Let's say you are a big enterprise and you are trying to find the total revenue you made in the last one day, last seven days, last six months or your quarter, right? So those are reports that need to be generated every day or every week or every quarter or every year.

So there is no reason to make those real time because those are fixed schedules. They are based on some certain schedules that you need to do reporting on. So those I think will remain as batch system for the lifetime of these things because batch systems are definitely much cheaper from resources perspective compared to real time system, right? So now there are other things that are becoming more real time. For example, you need personalization, right?

So somebody walks into a store, anyone showing the advertisements when you walk into a store. You cannot have a pre-canned list of advertisements because you want to personalize those advertisements based on a person's profile for example, right? So there are a lot of things that moved into real time systems in the recent past and I think the struggle continue but I never see – I don't think the batch system will ever go away because they can process a lot of data with very few resources compared to real time systems.

But again, I think real time systems is just kind of the middle ground. I also think that a lot of systems will move to a more live systems, which is the next generation of analytics in my mind. So batch analytics, real time analytics and live analytics and live analytics is a stuff of what Facebook newsfeed does for example, where there is nothing streaming going on. It is just when something needs to be displayed or somebody needs access to this information.

You create this information on the fly by queuing all of your raw data sources and providing that decision based on the data you path. So yeah, I think it's just a nature progression of events. I don't think any of these things will ever go away. There will be batch analytics, there will be real time analytics and then there will be live analytics because the customer system increases when you go from one range to the other. So not everything will get converted from one system to the other in my mind.

**[0:27:23.5] JM:** You are the founding engineer of the RocksDB project at Facebook. Tell me what that was like, did you start RocksDB as a side project or did you just jump into it full time because you knew this is going to be really important?

**[0:27:36.7] DB:** So yeah, truly this question also touches a little bit on the Facebook culture at the time. So the Facebook culture at that time was it enabled the engineers to try something new and we created a model where you can try different ideas and it can fail fast, which means you try something if it doesn't work, it is not a bad block on your carrier. It is a great thing to try something and fail. If you try something and succeed that's great too.

But if you try five things and four of them fails that's fine. Nobody is going to come and say, "Hey you are a bad engineer working on useless stuff," right? So the culture of Facebook really played a part here I feel. So Facebook had a lot of data on disk storage systems at the time and then just around 2010 or 11, 2011 Facebook moved a lot of their online data systems to flash storage. So when they moved to flash storage, the question that came out in my mind and there were some other people in the team was that, "Hey, for flash storage if we continue to use existing data software systems isn't this optimally use a flash?"

And so this is where we started the RocksDB project. We say that, "Hey, is there a better way for us to optimize this flash hardware by writing a new piece of software rather than using traditional B3's or traditional tree structures, which are essentially random write structures for data store." So we started the RocksDB project and the first used case again was I started this project and then there is one more person who came into my team because he also thought that, "Oh yeah, this is a good exciting project. Let me try my hand on it for some time."

So we tried this and there was a used case where we built a backend system, which was indexing a lot of Facebook data because data was stored in a certain way in the minuscule databases and then we needed to build a secondary index on this data so that we can find the same piece of data using some other indexing columns. For example let's say you store photos based on the user, right?

So the index is a user and if you have the user ID it is very easy to find the photos that the user published but now, somebody might want to find all the photos taken in the Golden Gate Bridge. So now you need to do is to build an index where the key is the Golden Gate Bridge and it will have a long list of photos hanging off in the database. So that is a secondary index. So RocksDB was kind of built – was used, the first used case was used to build a secondary index on your data.

So you can find these other things in our data very efficiently when the queries are from a different dimension like places or locations or time or stuff like that and then I found that it became very popular because it is very easy to use. So a lot of engineers adopted this without much trouble and the used cases was very different. There was a used case which was used, which was using RocksDB to publish data to notification to Facebook mobile phones for example later.

That was data system build using RocksDB interface and then there is a cold fishing tier, which is essentially finding the relationship or the closeness between two Facebook users using a machine learning or rule based engine and that needed to process a lot of data and that was data stored at RocksDB. So you basically qualified it in five different used cases, which are very different from one another and at that time I think we realized that yeah, this is good technology that's really useful for a large set of things.

**[0:31:05.5] JM:** You are now the co-founder of Rockset. You went from creating RocksDB within Facebook to eventually having an idea for how to productize it. How did you time at Facebook informed the goals of the company that you ended up starting?

**[0:31:22.0] DB:** At Facebook, I had the opportunity to look at these data systems from very close by and I saw that hey, RocksDB is actually used inside Facebook to run a lot of machine learning models on large data sets. It's more – so RocksDB was not used to build these models but RocksDB was used to serve these models. Like I explained, if there is a rule based mechanism to find the closeness of two users, when a user logs in it is good to find out who are his most closest friends by using this relevance mechanism.

And that needed to process a lot of data quickly so that was at RocksDB database. So those kind of used cases really kind of influenced my decision, my thinking that hey, this is a technology that we can help people, users outside of Facebook and that's what motivated us to build this company called Rockset. So at Rockset, the backend technology is very much a RocksDB data store but it has an SQL interface to it. So you can actually serve a lot of your models or evaluate a lot of your models on large datasets here.

Which gets stored in RocksDB internally but you can use SQL to interact with this system and so a lot of developers, what we are finding is a lot of developers like this system because they get the best of performance as offered by RocksDB on flash and RAM systems but they also can use SQL, which is a well-known language and easy to learn and adopt and they find it comfortable saying that, "Hey, I don't have to learn some new technology. I can leverage new technology using traditional systems like SQL and make my model serving easier."

So that's how I think our current set of users are actually leveraging the product RocksDB using Rockset technology.

**[0:33:11.3] JM:** Dhruba Borthakur, thank you for coming back on Software Engineering Daily. It's been fun talking to you.

**[0:33:15.1] DB:** Hey, thank you. Thanks a lot Jeff.

[END OF INTERVIEW]

**[0:33:21.5] JM:** GoCD is a continuous delivery tool from ThoughtWorks. If you have heard about continuous delivery, but you don't know what it looks like in action, try the GoCD Test Drive at gocd.org/sedaily. GoCD's Test Drive will set up example pipelines for you to see how GoCD manages your continuous delivery workflows. Visualize your deployment pipelines and understand which tests are passing in which tests are failing. Continuous delivery helps you release your software faster and more reliably. Check out GoCD by going to gocd.org/sedaily and try out GoCD to get continuous delivery for your next project.

[END]