EPISODE 1414

[INTRODUCTION]

[00:00:00] JM: Couchbase is a distributed NoSQL cloud database. Since its creation, Couchbase has expanded into edge computing, application services, and most recently a database as a service called Capella. Couchbase started as an in-memory cache and needed to be re architected as a persistent storage system.

In this episode, I interview Ravi Mayuram, SVP of Products and Engineering at Couchbase, about the architecture and history of Couchbase. To learn more about Couchbase, check out couchbase.com/sedaily.

[INTERVIEW]

[00:00:32] JM: Ravi, welcome to the show.

[00:00:35] RM: Thank you for having me, Jeff. Good to be here.

[00:00:38] JM: You have had a range of experiences in database systems and other back end application systems, spanning more than 20 years. I'd love to get your perspective on how you've seen database architectures change over that period of time.

[00:00:57] RM: A great question to start with. From the standpoint of databases, they have been in existence since the ‘60s, so to say, from the mainframe, down to mid-range and mini computers down to now commodity hardware. This has sort of been the journey from the standpoint of being single monolithic mainframes, which were application software databases, system software, all in one, to them being teased apart to multiple tiers and layers. And with the latest innovations that we have here is basically one of going into what we call as distributed computing from a single monolithic piece of hardware to a cluster of servers. So, it's called distributed databases.

So that's been the evolution. On the one hand, from the standpoint of how it has evolved from super expensive pieces of hardware to sort of commodity boxes in which we can run this. That's the democratization of that from that perspective. On the other axis is the sort of development paradigms that have changed along these last years, and how relational databases or database field in general has sort of gone about it is the other evolution from starting with the codicil systems of the past, to hierarchical databases of the past came the major innovation on this journey, which is basically relational systems. And this happened in the late ‘70s, early ‘80s. Since then, we have been sort of all on this relational journey.

While that was going, there have been other innovations, which had not somehow succeeded, but they were the object-oriented databases, object relational databases, XML databases. These are all some of the evolutions that are sort of happened. And finally, all of that innovation, not completely thrown away, all of that emerge this sort of NoSQL paradigm, which is, from our perspective, what extending were licensed systems have been all these years and sort of removing some of those limitations that the relational systems have placed. Because now, the modern requirement of data and the database management systems has completely changed to performance and scale and low latencies. The issues which did not exist to this level, back when the main objective was to go from physical records to electronic ledgers. This is where the relational journey began.

So, that in a nutshell, the hardware, networks and memory and disks, they've all evolved tremendously. That's one axis of evolution, which the databases have to adapt to. Second is the development paradigm has completely changed. There was no worldwide web and there was no object-oriented programming, when many of the database innovations of the past had happened. Finally, the consumption model has changed, which is all cloud and consumption-based computing as opposed to the good old days of you having to run on dedicated hardware. That type of computing paradigm has completely changed.

After this mobile as a component, the edge as a component, then the landscape has completely changed. And yet, in many instances, we are still using technologies that were built in the ‘70s and ’80. This is the modernization journey, or the database that we are here sort of making happen in Couchbase.

[00:04:21] JM: Give me your perspective on why NoSQL? We've covered this in other episodes, but I just love to get your personal perspective on why NoSQL has grown so much in popularity, as transactional database?

[00:04:40] RM: Sure. I think the main sort of change is in sort of where the data is getting zoomed in one sense. When the lacing systems were built, they were sort of consumed by people who are well trained. Now, societally what we have gone through is this disintermediation of the intermediary. You don't have a travel agent anymore, if you look around, mostly where the travel agent is, is you and your two thumbs on a mobile phone. Similarly, all those sorts of agents who are in between the system and the end user has basically been sort of subsumed by the technology itself in one sense. It's the direct to consumer movement, in one sense, that's what's generally happening.

So, in this, the first problem basically becomes is one of scale, where earlier you had your systems, enterprise systems have to serve to your 300,000-employee company, the biggest ones are of that size. To now, all of a sudden, you already have to start serving for tens of millions of users, your end users directly, this has been the journey in the last 15, 20 years. So, all those systems which were built, start to show their age, because they just cannot cater to this sort of scale, both in terms of accessibility, as well as geo distribution of this data, where it is required for faster interaction.

That's one side of the equation. The other is, the type of data that we are dealing with is no longer the rigid, predetermined, premeditated, sort of data. It’s very dynamic and fluid. So, this is the variety of data that we have to say. The famous way of sort of describing this as three V's, the velocity, variety, and volume of data that we're dealing with all have changed. So, the variety of data is where there's NoSQL systems brought about the sort of innovation because now it's in the database terms, it's called schema flexibility. This variety of data that you have to sort of deal with is no longer the classic, relational type model stuff, this is the newer, modern, JSON flexible type of data, and so that required a modern way of thinking. So, if you asked many of the enterprises, will tell you in the last 10 years, any new products that we have started, we have never started assuming it will be a relational system. Though some of the workloads still sits on relational systems, more and more of what people are building on based on these, the newer workloads, newer types of engagement-oriented applications, people are building, they're all come from the sort of the NoSQL head of the house.

So that’s, in a brief nutshell, a lot of details and is where NoSQL systems, they solve the scale problem, not all but some of them, but also the schema flexibility problem. And eventually, by solving both these, it finally ends up being a cost, which is cheaper for the engineering, the development teams to support such systems down the classic relational systems.

[00:07:43] JM: What are the ways in which the Couchbase architecture has evolved over time?

[00:07:50] RM: When we started about good seven, eight years ago, we were a simple key value store. Our origins, our DNA starts with one of the best highest performing caches that is very popular, which is called mem cache D. So, we were in memory distributed sort of a cache. And from there, the evolution was to sort of write through cache. What if this cache is performing so well can be persisted? So, then it becomes a persistent cache, and from there now, can you query this? Can you analyze stuff in this? Can you search the data that is sitting in this? This is how you sort – we have evolved it to say, “How can we offer those blazing performance, that is built into our DNA, and then give you all the semantics of a database all the way from the durability guarantees, the acid capabilities, and yet maintain the schema flexibility that is inherent in these JSON document databases that we have built for the NoSQL world.”

So, it has evolved from being a simple key value store to a full-fledged database, which can offer you durability guarantees, can offer you the full atomicity consistency in isolation level guarantees. So, it can give you the transactions for which people were using the older systems. And when they were using these modern systems for performance scale and low latency. Now, Couchbase is one system that can offer you both the guarantees that you want out of your relational systems that you're used to. And when you come to this site, you can get the performance and the scale and allow you to do distribute the data, yet have all your performance and durability guarantees. So, that's the evolution of the Couchbase from being a simple cache to a full-fledged geo distributed database.

[00:09:42] JM: In that evolution from cache to distributed database, I am assuming there was a lot of work that went into taking what was probably purely an in-memory layer, and figuring out how to make that persistent.

[00:10:00] RM: We of course, have our own storage engine, and the transactional system that we have to build in between to offer these guarantees of keeping the data no matter what, the durability guarantee so the one is to having the storage engine, and then to have our sort of transaction processing capabilities, which gives you these guarantees. So, it can actually persist stuff and this can give you a return of receipts, so to say, that the data is actually sitting on disk. These are all sort of classic database innovations, stuff that simple cache doesn't have to deal with, because it's just dealing with in-memory.

Now, there are a lot of assumptions you can make when you're just in-memory, because you're just, you know, operating between two memories and a network in between. But when you put a storage system, a transaction processing engine, query engine, a query language, so you have to build a lot more components to it, because a simple key value store, to that we added a secondary index, to that we have added a storage engine, which can actually sort of perform really well both when you go from in-memory – it’s tunable – can go from memory to disk at runtime without you ever having to take a system down.

So, these are all dynamic capabilities, which we sort of say late binding, as and you can do many of this sort of – we can treat the system as a cache, if you want, and go all the way to achieving full transactional guarantee. And in this process, we can interleave your transactional stuff with non-transactional stuff. So that gives you all the dynamism you want in a system. In order to build this thing, we have to do it in a sort of logical order of taking the cache, adding a secondary index, adding a query language, adding transactional capabilities for your documents, multi document transactional guarantees, multi SQL statement transactional guarantees on a storage engine, which is robust to handle this. One of our classic innovation here, just on the sort of on our cloud side, we have a similar innovation going on on edge also.

But on the cloud side, one of the main things we did was to put SQL back into this. And it's almost like I use the analogy of a Tesla. If you know how to drive a car should be able to get into a Tesla and drive. You don't need to know all the details of how the engine has changed underneath the covers. That's how we build this. We could easily have given you a new API's, a new way of sort of dealing with our database. But we took great pains to go and study the papers and relational architectures and query languages, and we made a – our query language which we call nickel, to work exactly to the specs of SQL. So, we are an extension to SQL. So, if you know how to do SQLs and databases, you can do the same in a NoSQL system of Couchbase. From the perspective of the data model, we support and the scalability, we are a NoSQL system. But from the standpoint of programming familiarity, we are a SQL system. So, if you know how to write SQL, come program on us, just like the way if you know how to drive, can drive a Tesla. Underneath the covers were very different. We are a distributed document model database, and not a shared disk, a relational, topple model database of relational systems.

So, underneath the covers were very different, which is what gives us the performance and the scale. And yet, from a programming familiarity, we deliberately put the steering wheel and the gas pedal back in the right place, so you can drive with the same comfort that you're used to the regular cars, which is SQL in the case of databases. It doesn't give you a sense of how we have gone about this evolution.

[00:13:35] JM: Yes. Can you tell me about like the query engine? So, if I want to program a document style query, versus programming, a SQL style query against the database, does that change the performance? Or how does the query parser treat those?

[00:13:59] RM: Yeah, so in one sense, from the standpoint of you writing a sequel, you do the same thing, select star from some table where, and the full SQL pipeline is available to you, in terms of aggregation, in terms of sub query group and joined by and user defined functions. All the sort of bells and whistles that you're sort of used to in the SQL world is available here. Underneath the covers, it's a document model, which is not your classic relational things. So, for that, we have built a query engine, which is fully distributed, and that which means that if you're running it on a 100-node cluster or 50-node cluster, this query engine is distributable across all these. It doesn't have to be in all these boxes. You can choose a subset of these boxes to run just the query service, but it's fully distributed, which means it can paralyze queries and give you the performance that you really want to achieve out of this.

One of the classic problems with relational systems which are distributed was the query service. They all bottleneck. But the way we have sort of built this is linearly scalable, that our performance sort of characteristics and numbers that we have. But because of the distributed query engine, which has got the parallelism capabilities built into it, entire query processing pipeline that you were alluding to, there are many places where we paralyze it, and there are places, let's say, if you want to sort stuff to bring it back to serialization, because you have to bring that data into one place before we can start of sort them. So wherever possible, we paralyze. Wherever it requires serialization, we do it. That's all in our query planning, cost based optimization that we are built. We are both rule based on cost based optimizing. These are major innovations in the database world.

In a flexible schema system, in a static schema, you can do many of this stuff easily. But when your schema itself changes, when you're sort of the data, the look or the shape of the data itself changes, doing all this stuff is a major step of innovation and that's what we have basically built on the cover. So, you get the same performance, and this familiar programming field. But underneath the covers, you get the full benefit of schema, flexibility and scalability.

[00:16:97] JM: So, if I spin up a Couchbase cluster, like a Couchbase cloud cluster these days, can you give me a sense of the infrastructure that that's going to be running on?

[00:16:21] RM: Yeah. So, I think from an infrastructure perspective, there are various optimizations that we do in terms of the sizes of the disks that is available for us to run the zone. But one of the major things that we do is, in order for you to reduce the cost is that, let's say you have a 10-node cluster that you're spinning up and running in our, let's say, the AWS infrastructure. We take that 10-node cluster, and we run query on a few of these, run data service on a few of this, and run the indexing service on a few of these. So, it's dynamically sort of scale. It's asymmetric now. Because of that, you basically get the performance benefit of not wasting SSDs, or fast disks, on all these sort of 10 nodes. Maybe it's sitting on only two nodes, and the query services running on three nodes, and you can just simply feed it more CPU capabilities. So, you can sort of, now, pack more compute on the same nodes. And then for the indexing side, we just feed it more memory. Because most indexes should be sitting in memory, and you're hitting the disk, only for the data fetching.

So, we have sort of right size. Imagine it's like a train and different cars exist in the train. There are pantry cars, and there are sleeper cars, and the sitting sort of cars. Based on the needs of the compute, we sort of choose various instances that's available, which gives us the cheapest cost of running this stuff. But the fundamental infrastructure allows you to sort of scale the system differently, so we could be running on 5x large of the storage, or the smaller version of it with 128 gigs of RAM, or we could go higher, depending upon each of these boxes. So, I'm not able to sort of give you a very precise sort of an answer, because it's very dynamically scaled, so that it's almost like in some cases could be used Spot Instances, that's the sort of sort of discussion we are having. That way, we reduce the cost and you get the most benefit out of it.

That architecture supports this sort of – the reason you want to go to the cloud is elasticity. And so, we are using the elastic scaling capabilities that is sort of available in the cloud. For our database, this is a big thing to do, because classic databases cannot sort of do this. Our fundamental architecture allows us to be sort of elastic in sort of consuming resources. Today, if you have high throughput, you can recruit more memory. Tomorrow, if that thing goes down, because today is let's say, the Black Friday and the day after Black Friday is like 1/10 the traffic, you can scale it back. So, that way you're not paying the price. It's almost pay as you go, sort of the model, which is a little bit difficult to do in databases, but we are sort of pulling that off.

[00:19:06] JM: You mentioned edge computing a little bit earlier. Can you describe some of the applications of Couchbase for edge computing and what you've had to build to enable that?

[00:19:19] RM: Absolutely. So, let's talk about a couple of these sort of use cases. One that really comes to my mind is like PepsiCo. They use our edge computing capability for their delivery. They have about 20,000 of people who are delivering their goods to the retail stores. So, all of them carry as you can see, in many cases people, carry devices with them, which can sort of monitor inventory as well as all the order processing of delivery, so to say, is sort of handled through that. That handheld has all those information and that runs on Couchbase with edge database on the device with the syncing capability with which it can interact with their database in the cloud, which has got all their information of inventory and catalog and their order scheduling, all those sorts of systems come into play, in order for them to – these 20,000 delivery people do sort of function effectively on a daily basis.

Another example would be a company, which allows you to pick up deliveries in locations, which are most convenient to you. It could be in a mall, or could be in your subway station. So, they have these kiosks where they can say drop my stuff here, so I can pick it up. These are places where internet connectivity is very low, and you can basically – it's a way for you to sort of pick up your goods that have been delivered to you in locations, which is most convenient for you to pick up. But those are the places where internet is sparse, as you can imagine in a concrete building, in a basement or in a subway station. So, they use this thinking capability to get the information of what delivery is going to be picked up from there, and you can go pick it up from there.

Another place where you can see is in certain airlines, again, apologies for not using these are marquee names in the industry. But let me just give the example as opposed to using the brand names. So, this is an airline, which does transatlantic flights. So, 8-hour or 10-hour flights, where actually when you sort of request a beverage or food service, the typical service would be for someone to take a note, go back to the galley, prepare it and bring it back. What they do now is have a handle in which the capturing of your order is done, which is synced with the back-end system which is sitting in the – it's only within the plane, this data is not going anywhere. They use the internal Wi Fi and to make this thing happen.

So, on the back, in the galley, there is a system which now syncs with this information, and someone else prepares the food and brings it. What they found was that their customer engagement scores went up really, and when they quiz their stewards as to how this happened, one of the things that they mentioned was that they actually were feeling much more refreshed because they don't have to go back and forth. And when they actually did the calculation on it, they were actually walking three less miles, by not having to just repeat this process of taking the order or going back and coming back, the net amount of time saved and the sort of less walking, they were more engaged with the customer. So, their customer stat scores went really up.

Another case is all these cruise lines. They use us in a big way, which is basically imagine a cruise ship to be a device on the edge. Many of these cruise services have these days. They give you NFC oriented sort of a gadget which you can wear on your person and that now can track a lot of stuff. You can sit in your state room, order a drink, walk down to the pool, the drink will follow you. You don't have to sit there and wait for the drink to show up. And your movement, your preferences, your dinner reservations, your evening entertainment, everything is sort of catered through sort of your personal preference which you can put on that application or on using that sort of that gadget that you have, you're on the on the transit there.

The most important thing is they're going to use this data to see the patterns of people's movement and help design their ships more effectively. So, that's one side of it. The other side of this usage is that where they're monitoring and sensing their effluence, both, they can sort of – how do I say this. Their effluence into the ocean into the air, this information is monitored and captured on the edge and then eventually synced when they come back to the shore. And that information is used to see how we can tune and be more environmentally conscious and friendly and pay less fines for any violations that they may have cost.

So, this is a way where they are both influencing the top line as well as the bottom line of their business. There are many such fantastic use cases. There is another one that I can mention is concussion detection, where people can wear these, imagine Oculus type devices, which has got a mobile phone in which you could run Couchbase edge database, and it captures the eye movement and stores that in the local, and immediately you can use that and it syncs with a laptop or an iPad sort of device that a physician can be carrying on the sidelines who can immediately tell you whether they've had a concussion or not. The care comes to you based on that information rather than you having to set up an appointment and meet the doctor three days later where the symptoms have gone away. This information is then synced to the cloud where they can do analytics, and now they are sort of doing cross pollination of, "Hey, football, these sort of patterns we are seeing how is that and can be used in some other sport, or even this is being used in scenarios like PTSD detection and stuff like that, which are all based on the same sort of underneath technology."

So, these are some of the edge use cases that sort of Couchbase plays a major role, industries being like transportation, hospitality, healthcare, these are all – and of course, mobile payments is another area where we play a big role in many of the financial institutions. So, these are all some of the examples of where Couchbase, the edge portion of the product. Because edge is not just disconnected. It is occasionally connected. So, you have to think that in the edge, you want to do something, but it has got to come back to the cloud, only then it has got meaning.

So, that's what we enable with our sync capabilities. Now, this works a lot of mobile Backend as a Service, a lot of other extraneous pieces that typically we had put in the past to make this happen. Now, it simplifies the entire architecture, and the deployment and less moving parts. So, it becomes a robust solution for our customers.

[00:26:16] JM: There's been a lot of application level functionality built around the core Couchbase database. So, things like Text Search, and Eventing. What's the binding – when you build this kind of features, how do you bind those to the database from an internal architecture perspective? Where do you insert that functionality to keep it low latency?

[00:26:44] RM: A great question, actually, because it's a very foundational architectural decision that we have sort of made. This is what I sort of call sort of late binding. What it is, is that there is a core set of database capabilities. We are modernizing the database. You have to look at it like to use an analogy is like when you were to modernize the phone, if it just did the same job as the dial toned oriented phone, then it is not truly a progress. When it's like a smartphone, it's 40 devices into one because it can sort of not just make phone calls, but it can text, it in capture video, it can capture audio, it can do GPS oriented stuff, it can do your calendaring. So, in that, lies the sort of the beauty of modernization, when you have a platform type approach. That's exactly what we have done with Couchbase.

The basic thing is that we are a database, which where it is data first and schema next. All major databases otherwise request you to define the schema first, before we can provide a single byte of data. Now, that we have liberated from there, I can bring different types of processing capabilities to that same data. I can be a key value store, I can be a classic database, which can do your SQL queries and transactions. On that, there is further integration that I can do, or I can use it independently like search. Because now a JSON document, if you see, it's human readable, then you should be able to search on that too, just like, I don’t know, I assume it's like a Word document or any other human readable document. I should be able to do search.

Now, what this means in the database parlance is that I'm looking for tokens, and this token that I'm looking for, this phrase that I'm looking for, can be anywhere in the schema. This is what it really means. It's schema agnostic, while the schema is there. So, our query engine understands that there is a schema and late binds to the schema, and gives you all the transactional query capability. Otherwise, it is just a JSON that is sitting in a database. Now, you should be able to do multiple types of processing on it, and search is a very classic example that you can imagine. Text is a field in itself. It's information retrieval, we have a full textual parsing pipeline, which can do the stemming, and vector distancing and fuzzy logic and all that stuff, which you can do on that same data.

So, the benefit here is that you're not copying data. With Couchbase, you're a cache, you're a database, you're a search, you can do the search on the same data. So, write the JSON once, and you can do your key value store type of lookups, your complex query, your search, the analytics, and sort of the Eventing pieces, which is basically where you can write your own custom logic, JavaScript logic on the data that is sitting there. So obviously, JavaScript is the most easy programming language on top of that, so you can fire your custom logic on the same data. These are the five different sort of processing capabilities on the same data. Otherwise, in a classic way that we used to use these systems, you will copy the same data five times. The data is sitting in database, now I need to optimize for the read, read scalability. Let me copy to a cache. Now, I want to do search on this data. Okay, let me write a connector to move it to another search system, and then copy this data into that search system the way it wants to operate.

Well, I want to do some streaming of this data. Okay, let me copy to a streaming queueing system. I want to do analytics on this data. Fine, let me just take this data, do a full ETL on top of all of this, and then move to an analytical system. I want to do some edge computing. God help us. There are like five different other services you need to build on top of this and then move the data. So, all these pieces, we have sort of reimagine, put all those services, we have sort of micro servicified the stuff underneath the database, if you will. And now you write the data once, but you can now do a key value lookup for login kind of stuff, or you can do your full-on inventory management system or customer relationship management system using queries on it. You can do a full search. You can do analytics on it. And then finally, you can stream the data out. We are built on a streaming protocol, which is through, which you can just simply – and high availability and disaster recovery is built into this system.

So, with a click of a button, you can create another cluster across a geography, move the data there, you can filter the data while moving. So, only the data that should be in the geo can move there and yet bring all this processing capability on top of that. That's the general innovation and it’s taken good number of years to sort of – in the logical journey of a key value store to where we are, we have added these capabilities in a fairly brisk space in the last few years. And now, you have a system which can be a simple key value store cache, or a full asset transactional system with high availability and disaster recovery built into it, and extend this all the way to an analytic system.

So, it's one platform in which you can do microsecond lookups. And people use this for 20 odd million ops per second, sort of lookup speeds that they need for airline reservation systems and other stuff. All the way to you can do full analytics on this one, without you ever having to sort of make it into a star, or snowflake schemas and all that stuff. They're not to do that, because there was no schema to begin with. So now you can run. Without doing any ETL, your analytical queries, we have a separate engine for it, which is the MPP engine where you can do all the analytical queries, which could take minutes and seconds to run. Same data, microsecond response times when you're doing key value queries, and may take a few minutes to run some of these complex analytical, high cardinality queries. So, both are possible, one not impeding the other, and that's the beauty of the architecture underneath the covers.

[00:32:48] JM: How has the emergence of the Kubernetes ecosystem affected how you run Couchbase in the cloud, and how you see customers wanting to run Couchbase?

[00:33:02] RM: I mean, our innovation started about almost, I want to say four years ago, three plus for sure. And that was mainly at that point, in one sense as much as driven by our desire to have a single orchestration platform for the on prem and cloud, as much as our customers wanting to deploy this in their data centers in a big way. So, we chose Kubernetes, as opposed to the other vendors who support on top of this. We take the Kubernetes, the basic so to say, the open source version, and we build on top of that, and then we are certified across like Red Hat OpenShift and the other versions that were there.

You basically have a single orchestration tier for the cloud and on prem, because many of our customers want both. Now, you are able to sort of orchestrate on prem and cloud using a single paradigm and so, our basic innovation out there was all this Kubernetes talk and so far even now, in the majority, they are all about stateless systems, a database anything but stateless.

This is the place where we maintain state. So, we have done innovation in the Kubernetes controller, extended that so underneath the surface we have a very good and we have the full topology visible to us. When a port goes down, another one comes up, which disk should attach to what? All that orchestration we handle, becomes easy for you to deploy and scale, and we are doing further innovation into that where we can do – this is what we call as Couchbase autonomous operator, which is a layer on top of our Couchbase server, which is what is used both in the cloud, as well as in the on prem deployments that if customers so choose to use. We can do stuff like threshold-based scaling, which is that you can simply say, “Hey, CPU spec at 80%, recruit another box, install Couchbase server and bring it up.” Now, you can distribute the data to all those new nodes recruited into this cluster. That's how you had capacity. All of this is programmable.

So far, databases have been the place where CI/CD came to a grinding halt. You could do all your CI/CD stuff and application tier, but we had to make a database change, it would say that, “Well, let's go talk to the other team and then let them make those changes before I can deploy this.” Because of the Couchbase core architecture, because there are schema and auto distribution of data, auto sharded, and all the stuff is already built in, with the additional capability of the Kubernetes operator sitting on top of it, now you can code your deployment changes along with your application changes. So, you go to your YAML file, make all your changes, and you can basically – when they same infrastructure as code, I want to say this is database as code. So, you can code your deployment along with your application changes. It’s just easy to manage, administer, and patch, upgrade all that stuff. You can basically now code if you use Kubernetes as your – and virtual deployment is what your system is going. Of course, running on bare metal too. But with Kubernetes, this is the major advantage, CI/CD comes with database built into your CI/CD pipeline.

[00:36:23] JM: I'd love to know how the maintenance of a large database that’s been around for a while, the maintenance and engineering is structured throughout the company, how you divide engineering teams and what their responsibilities are.

[00:36:43] RM: Yeah, I think basically there is the other part of the change that's happening is when people have sort of – many of our enterprise customers do follow that model, but more and more of the modern ones are more and more of that transformation out there is to go from this sort of IT and R&D type of mindset, that's a development team and a separate IT team to a DevOps model. So, in this model, both these things sort of come together in the same team. So, there is no separate sort of a maintenance versus a separate production on ops team. It's sort of fused into one organization, so that is both accountability, simplicity, as well as, how do you say, fusion of these services in one place. So, the services, uptime and runtime are all guaranteed by a collection of people. This is where the DevOps model sort of – these are the tools that help this DevOps model, so to say, stuff like Kubernetes, and running it on containers, and a database that is easy to run on containers and manage in autonomous manner. These are the crutches and tools that these organizations have requested with us or worked with us to enable their organizational shift on the back of technology shift that is required for such things to happen.

So, running a large 100-node or 500-node cluster is not easy. We make those things simple, because you can upgrade a Couchbase cluster, without taking a downtime. Your uptime is there, so let's say you're doing a 500-node upgrade, you can do those in batches, and then it can do in a rolling fashion. So, you can run in what we call as a mixed cluster deployment, which is that if you have 100 nodes, and 50 of them could be in the newest version, the rest of them are sitting in the older version. And until they all catch up, the newer functionality is not available until then your existing application runs. Then once the database is completely, all your 100 nodes in that cluster or into the newer version, then you can deploy a newer functionality so that the cluster is smart enough to detect that and tell you that I'm in a mixed deployment mode, when I'm on fully the newer version, newer functionality is available to you.

So, this is all built into the system, and you can do this maintenance and upgrade while your front-end load is sort of – you can still take the same front-end load while this upgrade has happened. But this we have to take into account, because these days, there's no more maintenance windows pager carrying people on weekends. It's gone away from that to a virtual or software defined deployments, and this is what this model is for.

[00:39:32] JM: I'm assuming that nobody uses or very few people use Couchbase for OLAP queries. Is that correct? Does it ever use an analytical database?

[00:39:45] RM: Yes, we definitely use that as an OLAP database as well. There's a famous food delivery company which sort of does all their data science team completely depends on Couchbase that are sort of some of these aforementioned companies use cases that talked about the use that for the OLAP thing. The main sort of – what we hear back from them is that two things. One is that analytics on the JSON itself. This is not an easy thing to pull off, so to say. I mean, because people what they do is, as you know, JSON is a sort of lingua franca of the web. All the data is sort of in a JSON format. Now, I have to convert that into a star schema or snowflake schema based on whatever sort of OLAP system I'm using, and there's an process I have to go through and then I have to dump it into that.

Now, I queried. When I bring it back to my operational system for insights, I had to redo this whole thing. So, this entire process is convoluted and complicated. That's one part of the equation for them. The other is the fact that if my data is already sort of sitting in operational database as a JSON, now in near real time, I can give you analytics because I just have to sort of issue the analytics query on the same sort of JSON without me having to ever touch that JSON at all. And out comes the insight which I can feed it back into the operational side. It does not just for reporting purposes, which is, of course, where it is mainly sort of, in the majority we're used for to find insights and reports. But you can sort of feed that information back to the operational systems so it can now react to the insights you are gathering in near real time.

So, that's the fundamental benefit. An example is like an ad targeting system, an ad serving system. Let's say you are now sitting in your browser and looking for, let's say, shoes. These days, what happens is that you will search for shoes today. For the next week, you are haunted by shoe ads in any website that you go to, that's because the insight that you are looking for shoes is something that they come to which many cases takes sometimes two hours, sometimes 24 hours, which is a vast improvement than the way it used to be. The OLAP systems were two, three months behind, not too long ago, then where the OLTP systems are.

Now, you can do it within a day. But that is because you're taking all the data and the operational system, moving it into your Hadoop or any of the analytical side, all the ETL it goes through. Now, there it runs the compute to see what is the right thing or rally searching for shoes, let me place this Nike ad over there. They are the ones who are giving me the most bang for the buck based on the keyword searches or whatever. You sort of send that information back, which is again batched back to the operational system, because it's the fastest ad serving system so to say. So, it comes back here.

By the time it comes back, you're done with your shoe purchase. But when you use a system like Couchbase, this is just another query that is sort of running while you're sitting in this thing. So, even if that query takes tens of seconds, you're still in that browser page, and you can actually see the ad serve. While you're actually serving, there's a much better probability of you clicking, if there is a deal there that is appealing to you or a style there that is appealing to you. It actually comes to you at your point of engagement rather than later. And that's the benefit of the whole knowing No ETL biz I was just talking about, is that it reduces the time to insight from days and hours to seconds and milliseconds.

[00:43:32] JM: Cool. Yeah, I didn't know that. So, as we begin to wind down the conversation, I'd like to talk about the more recent product, which is Capella, which is a database as a service. What's the difference between using a database as a service versus just using cloud managed instances?

[00:43:57] RM: So, a good question there. We offer you the full choice of deployment, however you want it. You can run it on prem, and manage it yourself. You can run it on the cloud. We have cloud-based instances. You can run it on the cloud and manage it yourself. Because many enterprises go in that direction, as you can imagine. And then finally, it's fully managed database as a service by Couchbase. So, you don't have to worry about any of the administration, maintenance, any operational, which is what typically are the, so to say, administrative functions. They're all taken over by Couchbase.

So, you simply as a customer, have to worry about your application running there. Develop your application as fast as you can and deploy it there and the rest is Couchbase and it will manage the scaling of it, the performance of it, the the administration of it, the maintenance patch, upgrade, all is taken care of by Couchbase. So, that's database as a service. The other two options I described earlier, which is, imagine from this vantage point, more and more administration is on your hand versus Couchbase, fully administering it for you.

[00:45:11] JM: Gotcha. What has been some of the engineering challenges of getting a database as a service running?

[00:45:22] RM: Yeah, several challenges. One as a product or as a platform versus other as a full-blown service. So, it comes down to orchestration. There is a data plane and a control plane, if you really look at it. It is the control plane that we had to brand new, fresh, imagine build, and tune the sort of the data plane with the control plane. So now, many of these administrative capabilities are sort of either managed within by Couchbase has an ability for the customers to influence. They can say, "Hey, I want the CPU consumption to be always be at 60% or 80%. Never exceed that." So that gives further information internally for Couchbase to see how to scale the system, or how to be more cost effective. If systems are not using the capacity that they wanted to do. So, that way, the net savings can go back to the customer or they're using more than metered billing. These are sort of the capabilities that are sort of net new that we basically had to build on top of the platform to make it simple for people to monitor, observe, metered billing, demand base scaling. These are sort of the the capabilities that you have to add to a product to make it into a service.

And then of course, we have to back it with a set of people who are both a combination of operational reliability engineers, as well as support engineering, which basically has to work with customers in case they run into issues and stuff like that. So, this is, in addition to the platform that we sell. These are the capabilities we have to build into the product and have teams to stand behind it to ensure that the service, has the same uptime guarantees that the customers demand, and that they are used to.

[00:47:12] JM: Cool. Well, I'd love to just close off, get your perspective on where Couchbase is headed as a company and where the product is focused on right now?

[00:47:22] RM: Yeah, so a lot of interesting stuff that we're bringing to the market going forward in the next release, which will be on the mobile side. A lot of great innovation there, which is in the areas of embedding our database in a lot more edge cases, a lot more edge scenarios. Let me leave it broad for now, as we are already closer to the earliest enhancement, I can give you more. But it's about us being able to embed the database and more edge use cases. That's one. And then make that whole management administration security to be really enterprise grade, because the data when it is sitting in the edge, it's got to be secured differently and there is a higher bar for it, because you don't want any data to be sort of lost, or at least fall in the wrong hands and stuff like that.

So, the security is paramount. Administering and managing that itself is a huge piece of innovation we are put in here. That's on the mobile, next mobile release. And on the Kubernetes operator side, a lot of great innovations that we're putting to make the demand base scaling capabilities for various of our services, the fine-grained nature of things we want to handle there. As a team, that's where the work is. And then on our Couchbase server side, that is innovation, every service, in the case of analytics, there's a lot of work that we have done for easy consumption of this data. We have more connectors and more tooling for visualizing this data and integrations with some of the very popular tools that are out there. That's a major piece.

There is a high availability capability in the analytics piece, which was something that we had worked on that will be geared in the next release. On the sort of general transactional side of the house, we have built the user defined function, which is one of the great pieces of innovation in the database world, which sort of happened in the early 2000s, if you will. And a similar one we are bringing to Couchbase as well. So, with this, there is sort of full parity in terms of what you can do in Couchbase in relation to what you can do with your classic databases. This user defined functions is a great piece of functionality, which allows you to write JavaScript logic in the query. In the middle of a query, you can have a JavaScript function fire the same time. Our query language can be called from JavaScript functions. So, it gives you both – it’s a very powerful capability where you can call it from a scripting engine, and at the same time, call the scripting engine from within the query engine. So, those capabilities that are sort of coming up next.

We are building sort of a storage engine, which will give us tremendous leg up in terms of the more capacity of data that we can manage by the end of the day. It should be even more cost effective for our customers. So, those are sort of some of the main innovation that is sort of coming. And of course, with every release, we put a lot of emphasis on security. So, there's a lot of security-oriented innovation that is sort of going into this next major release. As we move forward, there will be lot more innovation in terms of our ability to sort of process different types of data, that would be one major theme. One of the other themes would be on the sort of how to make the database more and more friendly for AI workloads. And at the same time, how can the database itself become more autonomous, and you know, I always say databases, I've gone from manual to where we are now, which is automatic, and where we need to go next is autonomous.

So, it's self-healing, self-managing, self-optimizing, those sorts of capabilities will be sort of bringing as a team on the server side, and similar team, if you can extend them, wherever applicable to the edge, that's where the edge sort of will extend to. All the while making it cheaper for our customers to run it on Couchbase. And most importantly, the , ease of programmability, that we put a lot of work into that so that it's in SQL familiarity or your familiarity with the various frameworks that you use, those are the sort of make it easier and easier for developers to use Couchbase on the cloud, and most of our emphasis will be to make it for you to have a very easy on prem to the cloud. You can sort of check that out in couchbase.com/sedaily is our website where they can go check.

We also have a free to play developer site called the Developer Playground. If you go to that site, you can find both these things and people can just sort of bring your browser and start developing. That's why we sort of want to take it to. So, you don't have to worry about all the install, config, plug in all that stuff. You can start there, and of course, when you're a sophisticated enterprise, you're not going to develop on the browser. But you can learn a lot and play and do your prototypes very easily. So, that on the cloud with Capella in the back, and the Developer Playground on the front, you can sort of do some serious development and programming and make it easier and easier for you to bring more application logic and remove all the worries of managing, scaling, upgrading, deploying, and all that stuff that we will take care of. So, that's generally the sort of set of themes that we'll be working towards the future.

[00:52:51] JM: Cool. Well, thank you so much for coming on the show. It's a real pleasure talking to you.

[00:52:54] RM: Fantastic. Thank you so much, Jeff.

[00:52:57] JM: Thanks for listening to today's episode about Couchbase. To learn more about Couchbase and look at their products, check out couchbase.com/sedaily.

[END]