EPISODE 1063

[INTRODUCTION]

[00:00:00] JM: Redis is an in-memory object storage system that is commonly used as a cache for web applications. This core primitive of in-memory object storage has created a larger ecosystem encompassing a broad set of tools. Redis is also used for creating objects such as queues, streams and probabilistic data structures. Machine learning systems also need access to fast in-memory object storage, and Redis AI is a newer module for supporting machine learning tasks.

For serverless computing, Redis Gears allows for the execution of functions close to a Redis instance, and Redis Edge allows for edge computing with Redis.

Alvin Richards returns to the show to discuss the expansion of Redis to becoming a broad suite of in-memory tools as well as the resiliency properties of Redis and usage patterns for the tool. Redis Labs is a sponsor of Software Engineering Daily and Redis Conf is a virtual conference around REdis that runs May 12th through 13th. If you're interested in Redis, you can check out Redis Conf for free by going to redisconf.com. That's R-E-D-I-S-C-O-N-F.com.

[SPONSOR MESSAGE]

[00:01:17] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it's definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[INTERVIEW]

[00:02:55] JM: Alvin Richards, welcome back to the show.

[00:02:57] AL: Great to be here. How are you today?

[00:03:00] JM: I'm doing great. How about you?

[00:03:02] AL: It's week 593 of lockdown and I'm still vaguely sane. So I think I'm in good shape. Thank you.

[00:03:10] JM: In a previous episode that you were on, we went through an overview of Redis. Could you give us a reminder of the basic use case of Redis?

[00:03:21] AL: Great question. Redis for those of you who've not come across it is primarily an in-memory database. It was designed to deal with very low-latency and high throughput traffic. The sort of original use cases are things like session stores, or leaderboards, or other activities where you have a highly volatile data where you want to look at it, analyze it quickly.

As the product has matured, so have the use cases. We have customers from financial services, healthcare, telco who use this all for a variety of cases where they want to look at large amounts of data very, very quickly with very low latencies.

[00:04:05] JM: Does Redis sit entirely on memory or is there a disk component as well?

[00:04:10] AL: You could see the way that Redis uses memory. Predominantly on the caches cases, people just use DRAM, but what you can do with Redis Enterprise is tear the data between DRAM, persistent memory and SSD so that you essentially can use SSD as an extension of DRAM and therefore deal with much larger datasets, but also change the price per gigabyte as well. So you can store greater datasets more cost-effectively.

[00:04:42] JM: How have the applications of Redis evolved beyond the traditional use case, which was an in-memory object store? That was the first way that people used it typically, was just to cache objects. But how have the use case evolved since then?

[00:04:57] AL: The use case evolved in a number of ways. Some of that is just people getting successful using it as an in-memory cache and realizing they can use it for other things. Some of it is also the extensions that have been put in to Redis overtime. Redis is inherently extensible, and so through the module interface, you can go and extend additional functionality. There're a lot of modules that have been created, for example, to extend the types of data structures Redis can support. Things like vortex search, time series, graph, AI and so on and so forth, but there are also extensions to Redis that allow for geo-replication across data centers, and that's a technology we called active-active, and that's using CRDT, a conflict-free replicated data types. This allows you to have essentially a logical cluster that spans data centers with each one of those data centers replicating changes. Allowing changes to be accepted in any data center and then moved around that global cluster so that you always got a consistent view.

[00:06:06] JM: When is it useful to have a Redis instance beyond a single instance? Sorry. So, a Redis cluster I guess you would say. When is it useful to be replicating your Redis into multiple places?

[00:06:22] AL: Well, there are two parts to scale out, and they solve different parts of the problem. The first is essentially it's not exactly true, but it's logically true, which is Redis is a single-threaded process. There are only so many operations a second that it can process. The first part of scale-out is what's called sharding. It's the ability to take your logical dataset, partition it into slots or sets and then allow multiple Redis processes to operate across that entire dataset by creating these logical slices. That's one way you get scalability if you run the limits of a single process.

Now, at the same time, you will want availability and resilience and say what happens if a single process fails? Well, you want that data to be replicated elsewhere so that if a process fails, you still have that data in-memory and you just have to deal with the failover time in order to still get access to that data. You can get from a single Redis process, you scale-out horizontally by partitioning or sharding the data and then you make each one of those shards reliable by putting in replication.

[00:07:38] JM: When you have a multi-node Redis cluster, how do transactions work?

[00:07:45] AL: Good question. There are a couple of constructs in Redis that allow you to build essentially transactions with multiple operations in, and this is through the multi-statement and the watch statement. Essentially what you can do is in multi, you can build, "Here's a series of commands I want to execute," and they're executed as a single operation.

The watch statement allows you to get a notification or to essentially fail the transaction if a key you are watching gets modified since you added that watch. That gives you the ability to deal with, A, batching up these changes into a single block that gets executed, but also notifications that the data that you're dependent on, or the keys you're dependent on may have changed under the covers since you started your operation. This allows you to build-out applications where you're relying on multiple operations to be executed.

[00:08:47] JM: Before we get into some of the other data structures in Redis, I just want to continue talking a little bit about the clustered configuration, cluster properties. Tell me about how to deploy a high-availability Redis cluster.

[00:09:02] AL: There are a couple of ways you can do it. You can take the native open source. It's got some native clustering and high-availability and you can roll your own, and a lot of people do that because they don't mind taking on that administrative burden. If you don't want the administrative burden, then obviously you could use a managed cloud service from all of your favorite cloud providers including Redis Labs, or if you don't want to go with a managed cloud service, you can use – We provide a Kubernetes operator, and that's another way to orchestrate the deployment of a cluster, or you use Redis Enterprise, which is not only got a built-in clustering. It’s got the automatic failover and recovery as well, again, to simplify the operations. The real question is, is how much do you want to take on yourself? How much do you want the automation through the product or through a service offering?

[00:10:03] JM: What are some of the other options for configuring a Redis cluster? Like if I'm scaling up, what are the kinds of decisions I'm going to be making across this cluster?

[00:10:15] AR: It will come down to having that good understanding of what is the type of data you want to operate against? How many operations a second do you think you're going to need? What are the latencies that you're after for the application? I would always recommend everybody would build the proof of concept to understand for their particular use case what is the ideal topology.

Now, we are four things that help people do that sizing calculation so that you can set up the right number of shards, the right kind of replication to satisfy your use case. But like any tool, any technology, you got to go from the sort of generic sizing formulas that any vendor or any project can give you, but to really understand what it means to you in your code, in your application. I would always recommend people actually build a many proof of concept and actually try it out and understand what it means for that particular context.

[00:11:19] JM: What about failure scenarios? What are the different recovery strategies in case a Redis node fails?

[00:11:27] AR: There are number of options. Redis provides persistence out-of-the-box, right? Most people think of it as in-memory cache, but it actually persistence. There’s a couple of persistence options, RDP files, AOF files. There’s a way of getting the data to the disk.

Now, obviously if you set up replication and there is another copy in a secondary or a slave node, and so there’s another copy of data there and that will provide, if you’re using Redis Enterprise, automatic failover if you lose that Redis process.

Now, the next stage on for that is what happens if you lose a whole availability zone or a whole data center? Well, this is where the active-active technology of Redis Enterprise comes in, because this essentially allows replication between clusters in different geographies. Depending on what kind of scenarios that you are trying to defend against, you've got a sort of playbook of deployment options that allow you to meet ultimately what your demand is.

[00:12:38] JM: Let's talk about some of the data structures that one can create beyond just the simple object cache. Redis has a number of different data structures that you can instantiate with it. How is Redis used for performing search?

[00:12:54] AR: Search is super interesting. We've had a module called Redis for two, three years now, and we have a bunch of customers who he use it for extraordinary mission-critical activities in ecommerce and in financial services, and it fits this sort of set of use cases where you need some of the characteristics of full-text search, right? You want to do skip words. You want to do locality and proximity of words to each other. But in Redis search, you can extend it. You can do numeric range scans. You can to geospatial queries, but all with the characteristics that you expect of Redis, which is very low-latency access to that data.

One of the obvious use cases that people can go read about is GAP, who use Redis search as part of their inventory management system. When you are looking for products online that's using Redis search to figure out where is the inventory? What's closest to you? What is the best shipping method to give you the shortest possible delivery times? They've put that together using Redis search, because it allows that flexibility of looking at many different attributes of that data structure not just a single value and doing a comparison against it.

[SPONSOR MESSAGE]

[00:14:31] JM: Apache Cassandra is an open source distributed database that was first created to meet the scalability and availability needs of Facebook, Amazon and Google. In previous episodes of Software Engineering Daily we have covered Cassandra's architecture and its benefits, and we're happy to have DataStax, the largest contributed to the Cassandra project since day one as a sponsor of Software Engineering Daily.

DataStax provides DataStax Enterprise, a powerful distribution of Cassandra created by the team that has contributed the most to Cassandra. DataStax Enterprise enables teams to develop faster, scale further, achieve operational simplicity, ensure enterprise security and run mixed workloads that work with the latest graph, search and analytics technology all running across hybrid and multi-cloud infrastructure.

More than 400 companies including Cisco, Capital One, and eBay run DataStax to modernize their database infrastructure, improve scalability and security, and deliver on projects such as customer analytics, IoT and e-commerce. To learn more about Apache Cassandra and DataStax Enterprise, go to datastax.com/sedaily. That's DataStax with an X, D-A-T-A-S-T-A-X, @datastax.com/sedaily.

Thank you to DataStax for being a sponsor of Software Engineering Daily. It’s a great honor to have DataStax as a sponsor, and you can go to datastax.com/sedaily to learn more.

[INTERVIEW CONTINUED]

[00:16:10] JM: There are also Redis streams and people who think streaming, streaming abstraction, they probably are thinking Kafka. How do Redis streams compare to Kafka?

[00:16:24] AR: Redis streams have got a couple of I think truly unique characteristics. One of them takes advantage of you getting very low-latency in-memory access. In a normal streaming model, yes, you’ve got data that's being fed in. You got the ability to create consumer groups so that you can have multiple processes consume that data as it's coming in and processing it.

Redis streams puts all of those basic things you'd expect, but on top of that in the Redis Enterprise 6.0 release, which has just come out, we support streams in active-active mode, which means that you can have this geographically replicated stream and consumer groups go and processing in different data centers all in parallel looking at a consistent copy of that data. That combined with the very low-latency access allows it to be used in a number of really interesting ways in not just the sort of IoT space, but in telco, in finance where people want to do a have very high-ingestion rates with low-latencies and processing of string type data.

[00:17:43] JM: Great. How are Redis streams implemented under the hood? I mean, we know that the Redis in-memory object cache is mostly an in-memory system, but it can be persisted to disk. How does that compare to how Redis streams are represented?

[00:18:02] AR: Good questions. Redis has had a pub/sub for many, many years and people use that for message buses and all the sort of typical things that you use a pub/sub model for. The challenge with pub/sub is that there was no guaranteed delivery of message. There was no guaranteed order. One of the things that streams did was to add those concepts in so that you could restart your consumer at the last point it got to and still get all those messages because you weren't connected. You weren't in the position where you'd lose those messages, which is the case for pub/sub. Streams is a much more reliable implementation in order to meet those, that criteria of ensuring that you do see each one of those messages over each pieces of data.

[00:18:54] JM: We actually didn't discuss how search is implemented under the hood. How much do you know about the implementation of Redis search?

[00:19:03] AR: Yeah. Redis search is going under a fairly large refactor, and so the new version, 2.0, incorporates a lot of that refactoring, and this came back from internal discussions as well as discussions with customers.

Let me sort of frame the problem, which is in the original implementation, what Redis search was doing was in holding all of that sort of – That breakdown of the words and their currencies and where they were into a series of internal data structures. That meant that you basically had to take your data, put it into Redis search in order to index it. That meant for some of our customer scenarios that they had multiple copies of the same data which is represented differently.

What we're doing in Redis search 2.0 is that we use in the native hash as the backing store of the data, and then the index is built on top of that pre-existing hash. That allows you to add Redis search capability to existing data by just pointing at the hashes you want indexed. That means that your code that then manipulates those hashes remains unchanged, but the full-text search capability of the index is maintained in the background.

We think this, A, reduces the footprint of how much copies of the data you have to hold that simplifies the adaption from the developer's perspective, because all the code that's maintaining that data just uses all the commands that they've been using for many place in the hashes. They want to do a full-text search. They then just go through the full-text search query API to go and do the querying in order to think back to the hash.

We think this sort of internal architecture allows for a more efficient and effective use. It's also have this really nice side effect, is that it's encapsulated search into a piece of library code that we can now embed in graph, in JSON all the other modules going down the road.

[00:21:13] JM: Okay. How much sharing is there between the different teams who are implementing these different data structures? Are there sub-modules in these different systems that get reused, like something that's reusable in the implementation of Redis streams that can be used in the implementation of Redis search?

[00:21:33] AR: Absolutely, and that's something that we're getting better at. I think what we try to do with the modules to begin with is to use it as a playground for people to experiment and understand how different data structures can be processed within Redis. As those modules mature, we make them available in the Redis enterprise, and one of the things about maturity is ensuring that you have high- quality codebases where you have the least amount of duplication, because you have the least amount of bugs introduced that way. I think as those modules naturally mature, some of those things come into play.

[00:22:19] JM: And probabilistic data structures, Redis has these implemented as well. Explain what a probabilistic data structure is.

[00:22:28] AR: These are things like Bloom Filters and TopK, these sort of – The short answer is sometimes an approximation is a good enough answer. Sometimes you don't need a complete accuracy in order to get a meaningful response. You could use some of these probabilistic data structures in ways that – In gaming, in leader boards where having a good approximation quickly is good for the particular use case.

If you need absolute accuracy, then you need the other data structures, but there is a potential performance penalty if you're dealing with a very large dataset or a very large graph of values. So it's a tradeoff. There are many cases where an approximation is good enough.

[00:23:23] JM: Can you give an example use case?

[00:23:26] AR: That's a great question. Like I said, things like it lends itself to places like leaderboards, where an approximation of what that leaderboard looks like is probably good enough, because at any instance of time, it's changing. So an approximation is good. It's like how many likes supposed this got? Do you need 100% accurate number with the fact that it's 35 good enough? Any of these cases where you can forgo accuracy to save computational cost, they're worth looking at.

[00:24:06] JM: These data structures, Redis has been developing these newer data structures throughout the years, and then more recently there's some other systems, like Redis AI is a newer system. Redis AI is a Redis module for deep learning and machine learning modules. Why did Redis start working on an AI module, or I guess more accurately a machine learning module?

[00:24:31] AR: Yeah, great question. We've had a machine learning module for a while. We got Redis Conf coming up next week and one of the announcements is Redis AI becoming a GA module. The rationale for those modules and frankly any module is the desire to bring those speed, high-performance, low-latency characteristics to different domains.

In AI, we integrate with Tensorflow, so it's a great way to be able to process and look at that data to do some of the training, to do some the analysis by looking at data in real-time. An alternative way of thinking about Redis is whilst it's a very high-performance keyvalue store, it's also great when you have a metric or a value or analytically you've got an SLA, right?

Wanting to get a finite piece of information really quickly is where Redis excels. That's why we have all the existing data structure type, so that's why we're adding things like AI because it's just another way of bringing those qualities of having a very tight SLA on a different type of processing with a different type of data.

[00:25:54] JM: What are the technical requirements for a backing storage system for my machine learning data? Why is it any different than just using in-memory cash or using a database?

[00:26:09] AR: I think it comes down to the particular use case, right? If you think historically, training was done as a big offline process. Wind back the time machine, you had a big Hadoop cluster processing this data, it crunch away. It would generate these models and then these models will be served by a process or a technology that was closer to where those models were needed.

But as people have found, as you get into sort of hyper personalization, you need to more and more real-time rather than batch processing. This is where a technology like Redis and the extension of Redis AI come into play. It comes down to does this answer have – Does it have an SLA and how tight is that SLA? How quickly do you want to be able to respond to change and reprocess that data in order to result in a better outcome?

[00:27:12] JM: Can you describe the architecture of Redis AI in more detail?

[00:27:16] AR: Yup. We work with a technology partner on the guts and internals of this, and so what we've done is we've incorporated some of their core technology into the Redis ecosystem so that through the AI module, that particular module and those particular algorithms can get access to that data with all those sort of Redis-y type attributes with very low-latency.

[00:27:46] JM: What about the API? What's the API for a machine learning developer who's interacting with Redis AI?

[00:27:54] AR: The API, some of it comes down to what is your personal choice? If you're using Tensorflow, you're going to us that tool chain in order to execute the processing in that particular type of framework. That the goal is to provide something as close to being as native experience with developer while still retaining that very quick access to the data involved.

[00:28:24] JM: How does it integrate with machine learning frameworks? Does it have the APIs into the frameworks themselves or are they just language level APIs?

[00:28:35] AR: They're just language level APIs. ML was the first module in this space and we learned a lot from that. From that we sort of branched out into AI, because we felt it gave a simpler way for the people who were dependent on those frameworks to get the value out of Redis and the performance that they could get out of Redis, i.e., to be able to manipulate that data faster.

[00:29:01] JM: How much data would somebody store in a Redis AI instance and what kind of operations would they be performing over that data?

[00:29:10] AR: Good question. AI, like I said, we got Redis Conf coming up next week. The 1.0 is going GA. I think it's early days. We do see customers who have been experimenting in a number of different and disparate use cases and industries. So handle [inaudible 00:29:33]. What are the key sort of use cases? We're going to finding out. We see what the early customers are doing. That doesn't necessarily mean that's where the sot of the majority of the use cases are going to ultimately be.

I think it's one of the things that come to Redis Conf and see what people are talking about now. Come back in 12 months' time and see what people have found out in those 12 months. I think that the world will change.

[00:29:59] JM: Another newer system in Redis is Redis Gears, which is a serverless engine as an adjunct to Redis. This is a system for executing functions on top of the core Redis infrastructure. How does Redis Gears compare to a serverless system like AWS Lambda?

[00:30:18] AR: Good question. We're super excited about Gears for a number of reasons, and some of this is – Again, it's sort of an extensibility of Redis. Sometimes you want to extend Redis but only a small bit. Essentially what Gears allows you to do is drop in code that runs in Redis as close to the data as possible.

The 1.0 is going to support Python natively and then we're going to extend out through Java and Scala going forward. This helps in use cases where you want to do some more sophisticated processing. You don't necessarily want to do it in a Lua script, but you don't want to transfer that data from Redis across the network and process client-side.

It's got some of that, the benefits of Lambda in terms of its serverless. You don't need to privation it. It just runs in your cluster, but it's also very different because it's manipulating the data that's already there. You don't have to transfer the data in or out in order to manipulate it.

[00:31:32] JM: Why would I use Redis gears instead of Lambda? I think it comes down to a couple things, which is do you want to program essentially in a proprietary framework? Whether it's Lambda or something else. Do you want to retain that code in a language that you're familiar with, like Python or Java, and you don't want to go to a scripting language you're not familiar with like Lua? Do you want to have that processing function operate with very low-latency because it's got access to that data and it's got that locality of the access to the data?

I think answering some of those questions get you ultimately to choose the right technology to solve your problem. There is nothing wrong with AWS Lambdas, but they've got some challenges that if you don't architect for, you may get tripped up. You may not be able to build something as performant, and if performance is the key thing, then you look for alternatives. Gears is just a really great alternative way of doing this.

[00:32:36] JM: Can you explain in more detail how a Redis Gears script would execute? I've got a Python script that I want to run as a gear, how is that executing?

[00:32:47] AR: Good question. Essentially, there are phases to this. The first is the deployment phase. First of all is essentially registering the script, understanding dependencies, because what the cluster needs to do is ensure that the script and the dependencies are spread throughout the cluster, right? Because when you execute the function and you got a sharded system where the data is partitioned, that function is got to be available wherever that data is available.

The first part is deployment and ensuring that code is present everywhere. The second part is now that the function has been registered and all the dependencies are installed and set up, it's now executing that function. One way of thinking about what Gears got to do is, it's a sort of distributed MapReduce, right? You got to ensure that that function is operating across all the places where the data is present and then pulling all those results back and presenting them back as a single set of results back to the client that invoked the function.

[SPONSOR MESSAGE]

[00:34:05] 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 softwareengineeringdaily.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. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:35:54] JM: Another newer feature is a Redis Edge. This is an edge computing system. Can you explain what Redis Edge is? One thing I'm curious about is what's the difference between deploying Redis Edge versus just putting a Redis instance on a cluster at the edge would be?

[00:36:17] AR: Yeah. Redis Edge is a project that we did to essentially package Redis and a couple of modules, timesavers and IA, so that he could deploy them not necessary on a super edge device like a phone, but perhaps one or two levels back from that so that they're acting as the aggregator for many of these devices. So allow the Redis Edge to basically collect and manipulate data on behalf of many of those devices. It's a packaging exercise that allows for simple consumptions for those types of models, or a touch of deployments I should say.

[00:37:10] JM: Is the Redis Edge database significantly different than a normal Redis database?

[00:37:18] AR: It's essentially dependent on the configuration, but typically it's a single Redis process. It's a very much smaller, more compact packaging. Because of the compute on the – The further way you get from the core, the compute is clearly less powerful. So you need to slim it down as much as possible.

[00:37:40] JM: With all these different data structures and systems, have you seen some new patterns developer on how people are using Redis?

[00:37:49] AR: Yeah. I mean, I think the great think about talking to the community and talking to customers is just you're constantly finding out new and different ways people are using your product, and I think this is one of the reasons I love to go to conferences or talks is just to sort of hear that.

As I said, people would start off traditionally with session stores and that type of functionality or use case, but increasingly we see people using it for very sophisticated analytics for fraud detection, credit scoring, personalization, anywhere where you need to look at a large graph of data really quickly. I think sort of what's happening in terms of the industry is there's a realization that the better data you've got, the more data you can look at, the better outcomes of the service or the code that you write are because you've got a better way of influencing others behaviors or you provide things that are richer, more actionable for the consumers of your service.

I think that's kind of where the real growth of Redis will be, is the realization that we had this odd notion that data got stored on disk and had disk access. I think the notion of Redis is that all data is available all the time with ultralow latency access. I think that becomes a very significant change in how people think and operate and manipulate data and how deep insights that they can create because they're not limited to disk accesses. When everything is now memory accessed, things look very different.

[00:39:44] JM: Do you know much about the cost differences between let’s say like a gigabyte of RAM versus a gigabyte of disk space these days?

[00:39:53] AR: I’m always really good at misquoting numbers like that, but there is a significant. Is it orders of magnitude? Probably. Disk on [inaudible 00:40:04] gigabyte, right? Now, what's happening in- memory architectures is not really much has changed in the last 15 years, 20 years. DRAM is DRAM. Yes, it get faster, it’s still very expensive. Intel launched persistent memories of final GA product last year. We now have a tear between DRAM and SSD of persistent memory, and as soon as persistent memory was launched, all the DRAM manufactures dropped their prices, right?

It's great having new technologies as competition because it starts driving the price down. Now, is traditional disk drives going to be cheap? Yes, they are for this foreseeable future. But if you look at what's happened with SSDs, what's likely to happen with persistent memory and therefore DRAM, the prices is going to drop.

Architecturally, I’ve got to a stage where I think that essentially memories for all intents and purposes free. If it's all intents and purposes free, what can I now do with it? How has that changed the problem I’m trying to solve? Now, in reality, it's not free. But if you look at the compute prices in your favor crowd vendor, look at how much memory you’re getting for very little dollars. The question is what can you now do with that memory? Terabytes of DRAM are going to be really commonplace. What do you do with a terabyte of DRAM?

[00:41:34] JM: Very interesting. What would you do? What do you think? What do you think will be some of the applications that become more feasible as RAM becomes more accessible and more cost-effective?

[00:41:45] AR: Well, I mean, not too long ago, a terabyte relational database was vast. I'm sure you've got close to a terabyte of storage on your phone. I remember my first iPod squeezing my hundred CDs into it. On my phone I updated, I've got my entire CD collection in lossless on my phone with some space left over.

I think what really happens is it unlocks the imagination of people not just in the data that they collect, but the data they want to manipulate and how deep they want to go in that data in order to look at more results, right?

This is why things like Redis Gears is really interesting, because you can start combining search, and graph and time series and look at those results in entirety in order to actually process different projections of data. You can combine these aspects together. Perhaps a full-text search drives – The results of that drives the first query into graph to give you those initial nodes and then you can do a graph computation on the vertices of those graphs in order to actually then look at the deepest set of data that you then feed into a time series, right?

Gears is a technology that allows you to combine those intrinsic atomic modules together. I think, thinking about memory is ubiquitous, allows you to start thinking about the data in very different ways and what you can process. I'm fascinated to see what people do with it. I'm truly fascinated.

[00:43:41] JM: Yeah, same here. I just wonder what aspects of my applications would become faster. I guess everything.

[00:43:48] AR: Well, I think speed is one of those things, but we deal with humans, right? Humans have got a blink time. The other way to think about it is all the machine-to-machine processes, right? Can you feed more data into those other processes to the machines that are consuming that data? I think the human aspect here is the slow part of it.

Wight the slow organic creature in the process, the other Silicon that can consume more and richer data or more augmented data may be able to do something quite radically different with it.

[00:44:25] JM: I know Redis Conf is coming later this month, and because of the crazy times, it's a virtual conference. What has it been like organizing a virtual conference?

[00:44:38] AR: It’s been honestly a huge challenge for our marketing team. They had their agenda. How they were going to organize it, and overnight it got ripped up and they had to start all over again. Yes, it's a virtual conference. We wanted to create something that was going to still be practitioners being able to tell their story for other practitioners, right?

A lot of what we've had to do is figure out how to organize those speakers to get their content in such a way that it's consumable in a virtual conference. We’re doing a mixture of some live events. Some are recorded. We have a Ask The Expert Session where you can book a session with an engineer and talk about your specific problem. We’re trying a whole bunch of things to try and engage the audience, but ensure they get high-value and high-quality content that, as I said, is delivered by practitioners for practitioners.

Some of those talks are going to be given by Redis Labs engineers. There are people from Apple and many top tier companies who are talking about their real world experience and use case of using Redis in production.

[00:45:53] JM: Are there any particular subjects or areas that you're anticipating seeing covered there that you’re excited about?

[00:46:01] AR: I think there’s a ton of exciting things. I mean, we’ve got everything from talks about Redis Graph, and [inaudible 00:46:08] who is the famous or infamous author of [inaudible 00:46:13] talking about work he's been doing with us proving out that module and the consistency of Redis Graph, all the way through what's new in Redis 6. Things like active-active or streams. We’ve got all the security features by ACLs a RBAC. Then we’ve got a whole bunch of technology promise. Talking about how Redis is being incorporated into their cloud infrastructure or the technologies that they're working with. For example, we got a great talk by one of our partners on in-memory encryption. The guys from [inaudible 00:46:49]. This is the next generation of what you can do with encryption, right?

Encryption at rest solve problem. Encryption in-transit solve problem. What about encryption in use? There are a lot of people who are talking about some of the kind of more future stuff, which will be super exciting to look through as well as the concrete stuff of what can I use right now?

[00:47:12] JM: Very interesting. As I was looking through the newer systems that Redis has been building, I understood the licensing deliberations a little bit better. I mean, there's just a lot of work, a lot of features that you guys have been building. Does the licensing get – Licensing stuff get discussed at all within Redis?

[00:47:33] AR: I think in our company and every other open source projects and every open source company based on an open source project, I mean I think we are all living in – Even before COVID, we're living in an interesting and challenging times, which is how do you hold the values of open source about the openness and the transparency but also ensure the longevity of those projects and build a community around it?

There's a sort of natural hesitation between those characteristics and the commercialization of it, and that compounded with offering open source projects as cloud services I think added this sort of third dimension that all the projects that have to deal with it. It's not just Redis. Cassandra's had to go through this. Mongo's had to go through, Elastic, and everybody is wrestling with how do they ensure the health of the community and the longevity of their project whilst needing to meet some commercial criteria so that they can generate the revenue to make all those things happen?

Redis is our own version of source available and so do many of the other vendors, and I think two, three years down the road, we will look at this time going, "Wow! Everybody had this sort of idea and then we slowly coalesced back into a sort of common principle like we were 5, 10 years ago. But I think it's still in flight for everybody about how best to work with this.

[00:49:15] JM: Yeah, definitely. What are the other Redis products that are being worked on right now?

[00:49:21] AR: Good question. One of the things we're talking about at Redis Conf, and it's free for every developer to download and use is called Redis Insight. Back in the SQL days, people used things like Toad, because they wanted a visual way to look at the data, inspect the data or run queries and so on and so forth.

Redis Insight provides a way to look at your data inside the Redis instance, be able to visualize that if it's a graph or JSON and so and so forth and be able to manipulate and do things like memory analysis and look for slow operations and so on and so forth. It's a desktop tool. It just allows for the developer a simple way to inspect and play with the data.

[00:50:03] JM: Cool. The Redis Conf again is a virtual conference. What are you going to be doing at the conference?

[00:50:11] AR: Yeah. It's a virtual conference. It's May the 12th. Free to register. What am I doing? I'm giving a couple of keynotes about some of the new products that we are excited to talk about. I'd love to tell you all about them, but I have to wait until next Tuesday. I'd appreciate to have a follow-up after that, but there are some great projects that we're announcing. I'm there mostly kind of to hang out in there, ask the experts and try and hang out as much with the community as I can do. In this virtual environment they've created, you can go play virtual pool with people. So perhaps you'll find me hanging out with the pool table.

[00:50:51] JM: Okay. Well, Alvin, thanks for coming back on the show. It's been great talking to you about Redis.

[00:50:55] AR: Thanks so much. It's been great to speak to you again, and be safe out there.

[END OF INTERVIEW]

[00:51:08] JM: Vettery makes it easier to find a job. If you are listening to this podcast, you are probably serious about software. You are continually learning and updating your skills, which means you are staying competitive in the job market. Vettery is for people like you. Vettery is an online hiring marketplace that connects highly qualified workers with top companies. Workers and companies on the platform are vetted, and this vetting process keeps the whole market high-quality.

Access is exclusive and you can apply to find a job through Vettery by going to vettery.com/sedaily. That’s V-E-T-T-E-R-Y.com/sedaily. Once you are accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you. If you have the right skills, you have access to a better hiring process. You have access to Vettery. So check out vetter.com/sedaily and get $300 signup bonus if you accept a job through Vettery. Vettery is changing the way that people hire and the way that people get hired. Check out vetter.com/sedaily an get a $300 signup bonus if you accept a job through Vettery.

Thanks to Vettery for being a sponsor of Software Engineering Daily.

[END]