EPISODE 783

[INTRODUCTION]

[0:00:00.3] JM: Monitoring tools are used by every area of an organization. Business development teams use monitoring to understand the metrics for product performance. Finance teams need to understand how the costs of cloud computing resources are changing. Site reliability engineers use monitoring dashboards to ensure that applications are up and running without problematic latency.

Product managers evaluate the results of AB tests based off of the monitoring data of how users are reacting to new features. A monitoring system needs to be able to handle the large volumes of data that are being generated at a high-velocity. The data needs to be queryable in an aggregated format, which might require an ETL system for getting data into columnar format.

Alexandre Pucher is an engineer at LinkedIn where he works on a monitoring platform called ThirdEye. ThirdEye is built on top of Apache Pinot, a distributed columnar storage engine that ingests data and serves analytical queries at low-latency. Pinot is comparable to Apache Druid. Alexander joins the show to discuss ThirdEye and explain why Pinot is a useful building block for monitoring infrastructure.

[SPONSOR MESSAGE]

[0:01:29.1] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with container technologies like Docker and Kubernetes, so you can monitor your entire container cluster in real-time.

See across all of your servers, containers, apps and services in one place with powerful visualizations, sophisticated alerting, distributed tracing and APM. Now, Datadog has application performance monitoring for Java. Start monitoring your microservices today with a free trial. As a bonus, Datadog will send you a free t-shirt. You can get both of those things by going to softwareengineeringdaily.com/datadog. That's softwareengineeringdaily.com/data.dog.

Thank you, Datadog.

[INTERVIEW]

[0:02:23.4] JM: Alexander Pucher, welcome to Software Engineering Daily.

[0:02:26.2] AP: My pleasure, Jeff.

[0:02:28.2] JM: You're an engineer here at LinkedIn and monitoring is one of the things that you're focused on most right now. Monitoring is a very broad term. What are the different users at LinkedIn that need access to monitoring tools?

[0:02:42.1] AP: Oh, virtually everyone. At LinkedIn of course, we strive to do or make data-driven decisions. If you do not have monitoring tools, or you don't collect the data, you're flying by it. There's various tools of course on various levels of the organization. In the lowest level, you have performance monitoring. You go a step higher, maybe system health, like some key business metrics, how many users do we have viewing your pages, signing up.

Then of course in the higher level, you have the generic, say user engagement, user satisfaction, maybe enterprise revenue. All of these different kinds of monitoring require different systems, or different tools and different ways of doing it. One thing that stands out is that oftentimes, there is a lot of data that exists in different parts of the organization that other parts are not even aware of. Then the question becomes well, if I notice that something is going wrong with my systems, how does this impact the business or vice versa? How does say a product decision impact the technology? Tying all of these together is one of the most interesting questions that I'm trying to answer and that I'm working on.

[0:03:54.1] JM: The data that turns into “monitoring data” can come from a wide array of sources. Describe some of the sources that you're getting data from.

[0:04:05.5] AP: Yeah. Again, it depends on which part of the organization you're talking about. For example, there's basic performance counters. Your CPU and memory and disk

utilization, there are business metrics, user signups and so on. These things usually get collected. We are, say Kafka, right? Which is a link that is very famous for that.

Then on the high-level, you have maybe quantitative but also qualitative data, but maybe what is your user engagement, but also what are the different holidays that are going on around the world and how are they impact your business? What is product decisions that are made, maybe AP tests and so on? There's a wide variety of sources where a data can come from.

[0:04:43.9] JM: You have a high-level metrics, like users visiting pages, lower level metrics like CPU load or disk utilization, stuff like that. If I understand correctly, a lot of this data gets shuttled to Kafka as the primary bus of maybe you would call the data lake, or an in-between the data lake. Could you talk more about how Kafka fits into LinkedIn's infrastructure?

[0:05:12.4] AP: Sure. Kafka attracts most of the key business metrics, so to say. Usually, high-level interaction that happens from say members, or even by employees inside of LinkedIn with the infrastructure they're collected. If you assume, for example you're a user, you go to LinkedIn because you've got a notification from a recruiter and you open this message and you interact with them, then of course, first of all we collect information, so that we don't modify it again.

Then the second thing is well, you take this information and you say like, "Hey, maybe this message was useful to you. You may want to collect, or receive something like this in the future again." Then this information can also be collected and say like, "Hey, here's a valuable user engagement," right? This is something that you care about. This information gets emitted via Kafka and then processed through a whole pipeline of different systems, which is of course there's Hadoop, there is Samza and so on. Then there's of course data platform at LinkedIn, which is known as UNP and that then builds on other systems like Pinot, like ThirdEye states the other – the reading end of this whole process.

[0:06:18.3] JM: Kafka was originally created at LinkedIn, so we should just talk – we'll talk more about Kafka, obviously in massive use in data infrastructure today across the world. In fact, the use cases only seem to be growing and the proliferation of Kafka seem to be growing.

Can you talk about the access semantics and the write and read semantics of Kafka? What exactly are we using Kafka for?

[0:06:43.3] AP: Kafka is really for high-throughput and also reliable delivery of messages, or events, whatever you want to call this. If you build a infrastructure be it for tracking, or be it for any collecting any business metric, then really what you use Kafka for is to make sure that when you emit data, this data makes it at very high bandwidth through a system and it doesn't get lost. Then ultimately of course, you collect this data in one way or another. By the way you just write it to HDFS, or you write it directly into your analytic systems, you end up usually with structures like lambda-styled architectures. Then on the back-end of this, you analyze what's coming off this.

[0:07:28.8] JM: If every piece of data makes its way through Kafka as a delivery system, could we just use Kafka as the data lake, as the source of truth for everything?

[0:07:41.9] AP: Kafka is really the delivery mechanism, right? It makes sure that the data that should be tracked is tracked, it doesn't get lost and then it is the transport mechanism that shifts, collects this data that comes from many different sources, many different services, servers, maybe data centers and delivers them into one place, or multiple places if you are looking for a replication. Then once this delivery has happened right then, there is a second step after this, which is the actual analytics on top of this data.

[0:08:14.4] JM: In terms of where the data winds up after being in Kafka, it's often HDFS, the Hadoop Distributed File System, because storage there is going to be very cheap. Can you talk more about the consumers of Kafka? We're writing all of our data from all of these different sources into Kafka and then Kafka is the means by which data gets from point A to point B. What are the different places where we are throwing data from Kafka?

[0:08:43.8] AP: I mean, there's lots and lots of different Kafka consumers that are altering the ecosystem, right? I mean, LinkedIn itself is probably famous for its data infrastructure, because most of it is actually open source, right? Of course, you already mentioned HDFS, or Hadoop is one place where this data goes, which is mostly just for box storage, the offline processing.

Additionally to this, there is other systems use LinkedIn for example, Samza, right? That allows you to take this streaming data and maybe combine it with offline patch data and then join this data together, so that you can ingest it into other systems, like for example, Apache Pinot, which is an OLAP system. Then you can combine whatever data sources you have, whether it's coming from Hadoop, or it's coming from Kafka, or maybe it's ingested from something else. It might be some performance counters for example that go into say RocksDB. Then combine all these data together to run your analytics.

[0:09:39.7] JM: You've mentioned a bunch of different components of data infrastructure that can be used to build a monitoring platform, for example. We'll talk about ThirdEye, which is a monitoring platform that you've been working on at LinkedIn. These different data sources, whether we're tlaking about HDFS, or Samza, or Kafka, they have different latency characteristics, and they have different write and read performance characteristics. Describe how these different performance characteristics make them a good, or a bad fit for monitoring applications?

[0:10:15.5] AP: It really depends on what kind of data you're collecting. If you have say high-frequency time series, maybe just BM performance counters, then you're emitting at high speed and you usually want to have this data available with very low latency, like maybe within just seconds of being emitted you want to know hey, how is your server from performing?

If you have data that has more detailed, say like high-dimensional information, which may be say I use a signup. If a new user comes to LinkedIn, they sign up, they decide well, did you get invited from a friend, or did you just sign up because they found LinkedIn useful directly? What browser are they using? Are they coming from say the US, Europe, right? Some location information. Just high-dimensional information is added. This is what Kafka really shines at.

You have tracking events that have lots of additional, maybe domain specific information that needs to be collected. Usually that means that you get high-quality information relatively or near real-time speed, but usually now you're not talking about seconds, but you're talking about minutes if you need to ingest this with a longer pattern.

Then if you go to a very high-level, like there might be some data that you really just collect manually, that maybe I feel and say okay, is there maybe configuration changes that are planned in the system, or are there any upcoming holidays for the business on the high level, this information might actually be entered by a human into a database.

Of course, all of these cases you talk about different types of data and different latency, but also different depths of information. Then when you join this data together, you ask yourself the question well, what is it that you're really after? Do you care about the high-speed, maybe aggregated crosshair information, or do you care very much about the detail and the tech and say the dimensionality of the information?
Then usually the more depth goes into it, the more delays is introduced into this whole process going from origination of the data to or even pretty much processing it in the end and analyzing it.

[0:12:28.4] JM: Now, describing the latencies of the data is one direction go in, but there's also the shape of the data. A JSON document for example might have many, many different fields. If we wanted to process a large series of JSON documents in order to do an aggregation, or do something at low latency for our monitoring purposes, this might not be so easy. Oftentimes, we are taking this highly dimensional data and turning it into some columnar format. Can you talk about the differences between columnar and document, or row-wise data?

[0:13:13.6] AP: I mean, I have worked mostly on the OLAP, or the analytics side. The analytic side is usually more about read performance and the performance of aggregating data. If you're in an OLTP setting, or say a transactional information, usually data that you say is the source of truth, then people tend to use roll-based storage are coming up better, this is like my SQL, or this is pretty much database systems used inside of LinkedIn. Usually it means that when you have an entity like, I don't know, a company, then all the attributes are stored together, right? The name of the company, may be some description, additional information like what is that website and so on and so forth.

When you have this OLTP setting, then consistency is of utmost importance, right? You don't want to lose information. If you send a message, you don't want the message to disappear, or beep later to something like this. Ultimately though, if you go to the space of analytics, you say

well, I'm not a source of truth, right? Usually, I'm taking the data from these OLTP systems and then somehow ETLing or transferring them into my analytic systems. That introduces some delay, but ultimately, it allows you to usually restructure this data in a different way so that you can read it or aggregate it much, much faster, as compared to OLTP systems.

[0:14:40.0] JM: Help us explore this from an application standpoint. We've got this data that maybe we are writing to a log file and we're writing a bunch of different fields about the state of a system, or maybe we're logging user behavior, high-dimensional user behavior and the data gets logged into Kafka in this row-wise format, or this document style format. When do we want to turn it into columnar data and what is the procedure for doing – what kinds of systems do we want to use for that?

[0:15:15.5] AP: At LinkedIn, really the biggest systems for these are both Hadoop and Samza, right? The idea is to process this data and then structure it. For example, if you have an OLAP system like Pinot, it takes tables and these tables have different chunks or parts to it. They're called segments. This is really just optimized columnar storage format, where you try to combine all the counts of page views, or the counts of signups together, instead of really going row-by-row, or document-by-document.

When you have a system like Hadoop, of course, you just have the batch processing you can usually do a highly efficient optimization of these segments. If you combine this with streaming, then usually you have to make this trade-off between high-optimization. It takes time. The other side is probably what is the latency of the data? Then usually when you run an OLAP system that maybe serves production use cases, then this trade-off becomes very delicate, where you try of course to be as close to real-time as possible, or maybe in the order of minutes, like 5 to 10 minutes. At the same time, also have high-performance when these aggregation requests come in.

[0:16:29.2] JM: Let's start to talk about this from an application point of view. If we are building a monitoring system and we've got these different data components that we could use, whether we're talking about Samza, or talking about HDFS. or talking about Kafka and we want to build this monitoring application, we realized there is some gap in the systems that we have

available, and so we end up building a new system, Apache Pinot. This originated at LinkedIn, right?

[0:16:58.4] AP: Yes.

[0:16:59.9] JM: Okay. Explain what Apache Pinot is and how it differs from other data infrastructure tools.

[0:17:07.0] AP: Sure. I think the most similar thing to Pinot is probably Druid, Apache Druid. In case you're not familiar with this, it's essentially – it's an OLAP system, but Pinot was really built for extreme scale and low latency at LinkedIn. Many of both internal and external use cases that are covered, really both from delivering statistics to users. Say the company pages how many employees work at LinkedIn, or Apple, or Microsoft. Other use cases internally that may be used for business intelligence, or even anomaly detection, AP testing and so on.

Really, one of the core ideas of building a system like Apache Pinot is that you want to be able to process large amounts of OLAP data at really low latency. We're talking datasets at the size of terabytes and you want responses to increase in the order of tens of milliseconds, or even less.

[0:18:08.6] JM: Why can't we accomplish that with something like Samza, or one of these systems that existed before?

[0:18:15.5] AP: Samza is really a tool to process your data. Pinot itself is the actual database system that takes in queries, essentially SQL-like queries. Then it gives you responses, usually optimized for aggregations, saying number of views per company, these kind of things; averages, sums, minimums, maximas and these things.

[SPONSOR MESSAGE]

[0:18:46.0] JM: We are all looking for a dream job. Thanks to the internet, it's gotten easier to get matched up with an ideal job. Vettery is an online hiring marketplace that connects highly qualified jobseekers with inspiring companies. Once you have been vetted and accepted to

Vettery, companies reach out directly to you, because they know you are a high-quality candidate. The Vettery matching algorithm shows off your profile to hiring managers looking for someone with your skills, your experience and your preferences. Because you've been vetted and you're a highly qualified candidate, you should be able to find something that suits your preferences.

To check out Vettery and apply, go to vettery.com/sedaily for more information. Vettery is completely free for job seekers. There's 4,000 growing companies, from startups to large corporations that have partnered with Vettery and will have a direct connection to access your profile. There are fulltime jobs, contract roles, remote job listings with a variety of technical roles in all industries, and you can sign up on vettery.com/sedaily and get a $500 bonus if you accept a job through Vettery.

Get started on your new career path today. Get connected to a network of 4,000 companies and get vetted and accepted to Vettery by going to vettery.com/sedaily. Thank you to Vettery for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:20:30.6] JM: We've talked about the role that Kafka plays. You've also mentioned the term lambda architecture. The lambda architecture was something that people were talking about a lot maybe five years ago, or three years ago. The lambda architecture was this idea that you have a slow leg of data and a fast leg of data. The faster leg of data may not be completely consistent, or even it may get events out of order. The slow leg of data ends up compensating for the inconsistencies of the fast leg of data. In exchange for that, you pay the penalty of some latency. Can you talk about the lambda architecture and how newer components were able to overcome some of the limitations of the lambda architecture?

[0:21:21.7] AP: I think lambda architecture is really – I mean, it's a see, a new term for an old problem. Even at LinkedIn that was just a few days ago, we put out a blog post about Apache Calcite and thought makes it possible for you to easily handle this slow and fast lag of data.

Ultimately, the real problem that lambda architecture seems to try to solve is to say well, we have a trade-off usually between consistency, or latency and performance under some resource constraint. This problem goes back I mean, way to the 90s, or even before, right, with parallel databases volcano and so on. Where depending on what is the current bottleneck in the technology that's available, but it's the speed of your disk, or the amount of memory that's available, or a bisection bandwidth of your network and so on.

You try to design a system that is optimized to take advantage of whatever is the resource that is the bottom line, right? You all put the miles for this one thing. As technology changes, people change their designs, or their architectures to optimize for different bottlenecks. I mean, probably one of the most famous examples over the past probably decade has been the Hadoop versus Spark thing, right? Where Hadoop was the system that allows you to well, farm out requests across large amounts of data that maybe feed only on hundreds of thousands of servers and still come to a result reasonably quickly.

Then as memory becomes more readily available and you suddenly go to the next one saying well, I don't really want to wait on all these disks. I want to train in-memory and maybe I want to take more advantage of my faster network. Therefore, I need a batch processing system that now allows me to do lots of computation in-memory.

This space keeps evolving, right? In a way that when you look at the system, even Apache Pinot, then you see that there again, LinkedIn started to take advantage of say the availability of really fast, like SSDs, the ability to process and ingest lots of data both from batch and from streaming in parallel, for example with a system like Samza. Then farm result across a bunch of servers and then they'll do some intelligent optimization of the way that you store your data and the way that you balance load, so that you can achieve really extremely low-latency for aggregation queries, even on massive scales of data.

[0:24:02.3] JM: Now we've talked about some different infrastructure components. Can you give an example of a data type that you would want to record and that would make its way from being recorded into Kafka and through Pinot and eventually make its way to a user's dashboard on a monitoring system?

[0:24:23.6] AP: What exactly do you mean by data type?

[0:24:25.7] JM: Let's just say a user profile view, or CPU load measurement, anything that we would want to use as monitoring data.

[0:24:33.8] AP: Sure. Well, let's take that profile view for example. Say some of your friends visit your LinkedIn profile, then this information is collected by both to show you that somebody visited your profile and then also to know what are people interested in general. If there's a visit to a profile page, then the fact that this visit happened is immediately a Kafka. Then Kafka serves this piece of information both to some of the OTP systems, right, so they can update the status, but also into this whole data pipeline, pretty much UNP.

There's two parts or two paths in this data pipeline that the same takes. One is of course Kafka writes it to HDFS, and so that you have a persistent log of this information. Also because you want to have up-to-date, or near real-time analytics information, you also send this Kafka event to say a processing system like Samza.

Then it is the job of Samza to combine whatever has been in offline storage, plus the incoming event. Join it together and turn it into say a segment for a system like Pinot. That then gets added to the system. From the moment where the segment is loaded into the state of a system, any query that is Pinot will return the aggregate data plus whatever was added. This then might be served back, for example to you as the user again and you see that the number of visitors of your profile page went up.

[0:26:05.1] JM: You mentioned that this data might be quickly written to OLTP. When the data is created to quickly written to an OLTP, online transaction processing system, and then much of the remainder of the data pipeline might be for OLAP purposes, online analytic processing. Can you describe the difference between OLTP and OLAP?

[0:26:30.9] AP: OLTP is really about transactional consistency. You want to make sure that to find both a trade-off between the performance of writes and reads, you need to keep up with updates. On the other hand, you want to make sure that any change that you make is consistent and durable. I mean, there's this famous example of the bank account. If you send money from

your bank account to another bank account, you either want this money to show up on the other bank account, or the transaction did not go through. The last thing that you want is your money to disappear and then never come back again.

In OLAP, you really try to optimize for read performance. You accept the fact that your OLAP system is not going to be a source of truth. Instead, it's usually the OLTP system. Then you try to take the data out of the OLTP system in a way that doesn't overload the original source system. The other way still allows you to be pretty much up to date, at least in near real-time.

This transformation of data, this loading of data, usually it takes an additional amount of time, which is why then OLAP systems are usually close to the state of the real world, maybe within a few seconds at best, or usually minutes, or maybe hours, but allow you to then create very effectively without sharing any negative impact on your production or OLTP system.

[0:27:53.8] JM: Just to fill people in on the acronyms, this is one that I didn't know for a while, the process of getting from OLTP to OLAP is often called ETL, the extract transform load procedure.

Now let's take this from a different angle, because I want to gradually illustrate what role Pinot plays here. Now let's look at this from a top-down perspective. Let's say I'm using a monitoring platform like ThirdEye that you've built at LinkedIn, I want to sit down in front of ThirdEye and be able to look at a dashboard that represents the user's profile views over time. Let's say all users in the United States. I want to be able to have this dashboard just in front of me that's just how frequently are people looking at, let's say their own profiles, the classic LinkedIn use case. People who are looking at their own profiles in the United States and we want to be able to build a dashboard around that. What's going to be the query path for this dashboard?

[0:28:55.6] AP: Sure. Actually, one of the funny things is that it's not really a dashboard tool, but you will still look at this data usually because you see some unexpected outliers. To just roll with this example, I would say, let's assume that –

[0:29:09.2] JM: Wait, what do you mean it's not actually a dashboard example?

[0:29:11.3] AP: The system it visualizes data, but the idea is really to surface relevant data. If you had your classic company dashboards that might have hundreds of even thousands of metric on there, then the idea behind that is to really say well, instead of letting you scroll through these thousands of metrics and then manual figure out what's going on, we will try to figure out what is interesting aspects of this data and then surface that to you. You can of course still manually add metrics in that you want to look at.

To just go back to the example, so let's just say the number of views on your profile page plummet dramatically and this is an outlier or an anomaly. Then you would look at this information infer. Of course, you can look at this information in various ways. You could do some simple aggregation, say the number of use this week is lower than last week. I think actually at LinkedIn, usually it sends you these summary e-mails that says how many people looked at your profile page.

Another way of doing this would be to do it as a time series. Well, you could look at this say per day, or per month, or per hour probably, depending on how active your profile is. The way that this information is surfaced is that and thought I would simply go identify what is the name of the table and the metric that you're interested in, maybe apply the filters. Of course, it's your profile and maybe you only care about views from a certain geography. Then you take this information, you turn it into a SQL-like query and you pass the screen to Pinot.

Then Pinot, as of course it's charted, but this should be the database system, right? We have a broker that then takes this query and decodes it, figures out which of its working notes have the relevant data and then it farms all these requests to be processed. Then different charts, it just process the data and this gets aggregated in a whole and then eventually just sent back to ThirdEye.

Of course, the take away there is that all of this information lives in Pinot, so I don't need to actually go back to the real database system that stores your profile to scan three entire table and figure out who viewed your profile. It means that this information has been pre-processed and is readily available and doesn't really have any negative impact on what's going on in the real world.

[0:31:35.3] JM: It's pre-processed, sitting in columnar data files on HDFS?

[0:31:42.2] AP: Pinot uses its own format. It does essentially a segment format. It's just then, think of it as a chunk of a table that is highly optimized for aggregation, also depending on the use case.

[0:31:55.3] JM: Okay. It's sitting on disk. It's not in-memory?

[0:31:59.0] AP: Yes. It's actually both on disk and in-memory, right? Obviously you have disk, usually SSDs, but there's also a conventional, like the rotating, spinning disks. The other part is that you of course take advantage of memory as much as you can. That's of course some caching mechanism.

[0:32:16.1] JM: Can you talk about that in more detail? How does the latency characteristics of that data – obviously if it's in-memory, it's going to be served much faster than if it's sitting on disk. Do you ever need to specify to the system like, we need this dashboard to be up-to-date much more rapidly, so maybe the profile data should all sit in-memory? Can you specify at what layer of the cache hierarchy you want your Pinot data to be stored?

[0:32:46.4] AP: Let's put it this – the theory is that the system should figure this all automatically. In practice, obviously you will input your domain knowledge and tell the system that certain data sets, like for example, we have these company pages with the statistics of number of employees, they serve production traffic and they should never leave cache, because it's important.

[0:33:08.2] JM: For this application that we've described, we could have just used some other columnar data storage format, like Apache Parquet. We could have just ETL'd these from Kafka, stored them in Apache Parquet files and have some query system like Pig or Hive, right, to query these columnar data sources. Why do we need a brand new system like Pinot?

[0:33:33.1] AP: First of all, I mean, even at LinkedIn you use many different columnar formats, right? You mentioned Hive [inaudible 0:33:38.2], this of course comes from the Hadoop side there. Really, the idea where Pinot is well, you optimize this for fast, excessive query. You could

process this data, which means usually you don't just have the raw data. You have an index, or usually in just one index, but different types of indices, right? So that you can both filter the data quickly, aggregate it quickly.

Pinot then does another optimization in the segment format, which is known as star cubing. You pre-aggregate parts of the data, so that you're ready to respond extremely quickly to queries that aggregate across different slices, or different dicing of the data, maybe at the cost of additional storage space, but really to optimize for a low, low latency.

[0:34:27.4] JM: Okay. Is it the querying system that is optimizing for the low latency, or it's the storage format? Can you talk a little bit more about that?

[0:34:35.4] AP: Yeah, it has to play together, right? The one thing is well first, you need to have the data accessible in a way so that you avoid scanning through say terabytes of data in order to give a quick response.

[0:34:48.5] JM: Even if it's columnar.

[0:34:49.9] AP: Even if it's columnar, exactly. This is usually what star cubing is about.

[0:34:54.3] JM: Star cubing? Is that what you call?

[0:34:55.8] AP: Yes.

[0:34:56.6] JM: Never heard that term. Can you define that term?

[0:34:58.4] AP: Pinot has one of the index format set it use is star tree. The idea is that if – Maybe let me give you a simple example. I look at page views; they might come from a different browser and they might come from a different platform, say mobile and desktop in the simplest sense. Now if I were to ask the question how many page views are there in my system? The conventional way of doing this would be to just go through and scan all the rows that I have. Say okay, well here's all the views that I have on Chrome on mobile, plus all the views that I have on

Safari on mobile, plus all the views that I have on Firefox on mobile and then the same thing for desktop.

That way when you have high-dimensional data, that doesn't just have two fields, but maybe 20, or 100, or a 1,000, then even if you do a simple aggregation that just says like, “Hey, what is the total?” You end up scanning through thousands, ten thousands, or billions of rows depending on what's the cardinality of your data. The idea of it, roughly speaking, the idea of it star cubing is to say I want to avoid making these scans across billions of rows. Therefore, I will pre-aggregate parts of this data.

For example, if I just care about hey, what is the number of page views on mobile? Then I will not just have well, page views on mobile on Chrome and page views on mobile on Safari and pages on mobile Firefox. Instead, I have this concept of a star column, like the all-encompassing, or the aggregate dimension. Therefore, I can now ask the system well, what is the number of page views on mobile? Then the system will go and say like, “Okay, mobile and the browser is the star,” which is just the aggregation of the previous columns. This aggregation can happen before the system actually issues the query, right? Then you generate these segments. This is roughly what star cubing is for.

[0:37:05.1] JM: It's a way of doing – I think this term is sometimes called roll-ups, or pre-aggregation, right? I've got my data that I'm collecting about user profiles. That data is getting ETL’d into my OLAP system, which in this case is Apache Pinot. How does Apache Pinot know what roll-ups to do? Because there are many different roll-ups. I mean, you've got all these columns you're saving, you've got the different profile view instances, you've got I don't know, maybe the age of the person who viewed the profile, you've got all these different things. If it were to just randomly generate aggregations and combinations of data in the system, it would be fairly random, how does it smartly aggregate different columns?

[0:37:54.3] AP: There's multiple ways of doing this. Sometimes this happens literally by configuration. Somebody figured out, “Oh, we’ll need to start off aggregation. Therefore, generate a start dimension for this.” Another way of doing this is more automatic, or heuristically almost, where you can identify depending on how the data set looks like, what aggregations can I have?

Then for example, one way to decide which things to pre-aggregate and which things to just leave us is to simulate how many rolls would you scan? Give them a certain aggregation query. If this surpasses a certain threshold, then you start to pre-aggregate.

[0:38:37.1] JM: There can be some combination of automatic discovery of what should be aggregated and there can also be a manual specification there.

[0:38:45.2] AP: Of course, you can really go off the rails with this. You can go and do all kinds of predictions of how this should be done and so on. Roughly speaking, it's either heuristics, or some manual specification.

[0:38:59.3] JM: Yeah. Now let's revisit this latency question. One way of revisiting it is the discussion of batch versus streaming. Streaming data, you could say is coming from one source and it's being pulled into another source on a data point by data point basis, and then the batch might be we let the data points accumulate in Kafka on maybe on a 24-hour basis and we batch these 24-hour collections of data points into the data lake or into Apache Pinot.

In this end-to-end system that we're describing, the generation of data from profile views through Kafka, into Apache Pinot and eventually to the monitoring system, the end monitoring system ThirdEye, are there separate data paths for batch versus streaming?

[0:39:56.3] AP: There is different data paths and actually they happen in parallel. However, it really all ends in Pinot, so to say. Where if you have your Kafka events that get collected and obviously, you have an offline storage and you'll have some regular Hadoop job running through that just collects say 24 hours of those and then turns them into say these Pinot segments.

When a Hadoop does this, it runs all kinds of optimization to make the segment as efficiently packed and pre-aggregated as possible. That of course takes some time. Therefore, the second and parallel path to this is the streaming part, right, as you indicated. Typically for streaming, it means that I process these Kafka events in micro-batches. That typically entails just waiting for certain periods of time. Maybe I wait for a couple of seconds or minutes or maybe an hour. That

way, I generate segments that are still collections of a number of events, but of course there's more segments, so usually creating them become slower.

Then both these large offline segments and maybe smaller streaming segments both get loaded into Pinot. Then it is up to Pinot to decide when a query comes in, can I query some of these highly optimized offline segments, or is there only some up-to-date data that I can only get from these streaming or online segments that are generated? Usually, this is also split across different servers, so that they don't interfere with each other. It really comes down to optimizing or wisely choosing which segments to access at query time, depending on what's available in the system at that moment.

[0:41:36.7] JM: Let's come back to ThirdEye. Earlier when I gave this simple use case of wanting to know about the aggregation of people viewing profile pages in the United States, you pointed out that ThirdEye is actually useful for surfacing insights that you didn't know about across the data, rather than just configuring dashboards about things that you already know you want to monitor. Describe some of the discovery features of ThirdEye and then we'll get into the engineering behind them.

[0:42:11.3] AP: Sure. That's really two main tasks that it fulfills today at LinkedIn. One is pretty much the anomaly or outlier detection. The second part is then what goes more into the follow-up use case of anomaly detection, which is I found that there's an anomaly. Now what caused it, right? I try to infer potential root causes.

Now for anomaly detection, it typically means that I need to be somewhat choosy in what I monitor. If you monitor billions of metrics, each of which have tens or maybe hundreds of sub-dimensions, you will always find some form of outlier, maybe just because it is noise. There has to be some human discretion of what parts of the data are monitored and which parts are not. Of course, then different time series of different metrics, those behave in different ways and sometimes there's say like to me, a specific business logic that applies. A team that looks at say, ads and click-through rates has a different way of determining what is an anomaly, than say a team that looks at disk utilization and just system stability.

Now ThirdEye takes these various metrics and then trains models against them. In the simplest case, it can be say a user-defined rule. It can get complicated enough to, say different e-mail techniques. There can be say, regression models, various say spline regressions or other ways. If I find an anomaly, then I will notify the user and then the user can come into the system and say, “Okay, well now I know there's an outlier, but what data exists in a context of this outlier?” If I'm looking at ad click-through rates, then maybe the number of ad impressions and ad clicks might be relevant. ThirdEye collects all kinds of these relationships between metrics between systems and then tries to intelligently and on demand walk through this dependency graph and the available data to give you other related metrics that also show surprising behavior during the same time window to help you as a user figure out where should I go with my investigation.

[SPONSOR MESSAGE]

[0:44:39.9] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It’s open source, it’s free to use and GoCD has all the features that you need for continuous delivery. You can model your deployment pipelines without installing any plugins, you can use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your cloud native project.

With GoCD on Kubernetes, you define your build workflow. You let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, and they have talked in such detail about building the product in previous episodes of Software Engineering Daily. ThoughtWorks was very early to the continuous delivery trend and they know about continuous delivery as much as almost anybody in the industry.

It’s great to always see continued progress on GoCD with new features, like Kubernetes integrations, so you know that you’re investing in a continuous delivery tool that is built for the long-term. You can check it out for yourself at gocd.org/sedaily.

[INTERVIEW CONTINUED]

[0:46:10.5] JM: One thing you mentioned there, we covered this in a couple previous shows about anomaly detection is that you can have these automatically generated anomaly detection systems, but such a high-throughput system, it's hard to create a general rule for what constitutes an outlier, or an anomaly. You might want to have people describe to the system what does an anomaly look like.

However, if we're talking about building a monitoring platform that's useful for ads operation specialists, back-end data infrastructure SREs, designers, all these different use cases, you don't want to give them a super complicated interface for describing anomalies. What's the interface that you want to present to the user, or what is your anomaly?

[0:47:05.9] AP: I totally agree with you. It is essentially impossible to do it right for everyone. I mean, probably one other thing to say is that no matter how well you're doing at anomaly detection, there's always a way to improve it. There's always something that you missed, or there's always something that's too noisy.

The other part to this is that I think about the history of ThirdEye, or just even as it's used today, it is primarily about integrating data from different sources across the organization. You will still have monitoring systems that are specific for say ads operations, or monitoring systems that are specific to the SREs and monitoring systems are specific to how the users actually use my platform. Then the task of ThirdEye is to be able to integrate data across these platforms and have some out-of-the-box tools that will cover hopefully 90% of your use cases.

The idea is not to really replace any of these individual systems, because there's always some domain-specific thing that you can do better if you build a system just for your SREs, your designs, your ads. On the other hand, what ThirdEye provides might be more than good enough for simple use cases, so that you don't need to build another dedicated system. Instead, you can go to this platform that shows you this integrated data. You have the additional benefit that now you do not need to go to three different systems, be as the SREs and the design interaction to figure out how did one thing impact the other, but instead you get it in one place.

You also have the advantage that if in any one of these specific parts you see a problem, you can go to ThirdEye and because it is aware of these relationships between these different parts of the organization or the metrics, it can point you to the relevant other parts of the org.

[0:49:06.5] JM: LinkedIn has been around for a while. I'm sure since the very early days, there were different monitoring systems. Could you explain how ThirdEye contrasts with previous monitoring systems and maybe illustrate that. Do you have maybe an application that you can illustrate like, here's what ThirdEye was able to enable and this contrasts with older monitoring infrastructure?

[0:49:34.0] AP: First of all, I mean, every company has monitoring systems, even LinkedIn today has many different monitoring systems. If you ever attempt to write the one monitoring system that solves all the problems, then you end up in N plus one monitoring systems. That's just how these things go.

Instead, what we try to do with ThirdEye is to integrate with these various domain-specific systems. We of course also take advantage of Apache Pinot. At LinkedIn, that has lots and lots of that quantitative data already in it and we can access it effectively. Pulling all of this into this platform, you want to integrate data but still provide the backlinks to the various systems that he gets this data from if somebody needs to dig even deeper.

Now a classic use case that further enables that was not really possible before is to get an end-to-end view of how, say LinkedIn as an organization interacts as a whole. To give a classic example, I think we also wrote this in our blog post. There were cases where actually the ads team noticed that, or that certain parts of the feed, some sponsored content was not displayed correctly that should have been more. The whole team was investigating, well where does this come from?

Obviously, there was some change was made somewhere in the large distributed system of LinkedIn, but nobody really knew. Now, the ironic thing about this whole thing is that the monitoring infrastructure of course of the ads team was somehow connected to the monitoring infrastructure of say, the feed team. The feed team itself also has its integration with say, some

security components. Because this integration was on a high and an aggregated level, it just indicated well, there's a problem somewhere.

Independently of this, somebody actually sat down and was trying out well, how does ThirdEye work? What's the benefit of pretty much hooking ourselves into this platform? What ended up happening is that pretty much immediately as this data gets hooked up further finds, there are some related problems in the feed and actually in some security components. There's other information that gets fed into 32, like what is cold deployments, right? What is a AP test and all things around?

I go one of the things that showed up was a new code deployment, really it's one of the security components. Because you can relate this data and you can walk this dependency graph and say in LinkedIn, or just in an organization, you become aware of potential problems much faster. Then they just sat down. They looked at this code change, figured out that oh, for security reasons where we are dropping our own ads, we could simply roll the back-end that really fixed the problem.

[0:52:26.9] JM: Can you zoom in on that a little bit more? Because what you're describing here is a connection between data sources that were probably owned by two different teams, being related to the timing of a deployment. These are these super disparate data sources. Talk in a little more detail about how ThirdEye was able to find a correlation between these highly disparate data sources.

[0:52:58.1] AP: I mean, there's multiple ways of going about this. In reality, even in say in a large organization like LinkedIn, of course lots of code is deployed and lots of changes happen. Usually when you see a problem occur, you can walk back to where it started. Maybe if you see it in something like your click-through rate and your ads, this problem might actually come from somewhere else, like maybe the number of impressions went down, which was the case there.

Then you can usually see well, when did this trend of lower impressions start, right? Then you get a typically a time a point, or a point in time. When you have this point in time, you can see well, what is other things that happened around the specific point in time that could have cost us? Did somebody start an AB experiment and somebody changed the code? Did somebody

change the configuration of the system? There's much various events that happened. Maybe they're more qualitative in information, but they still tell what in what point in time they happen and maybe in what part of the system they happen.

If you have high-dimensional information in Pinot, then you can say, "Well, I see it as a trend of lower as impressions." Most of them are related to say, "My feed, they're not coming from in all the – like an ad somewhere in the model platform, but they're embedded in your feed." When you have this information, you know okay, it must be related to the feed. Additionally I have the time dimension that says when it happened. Therefore, if there is a code change that is close in time and related to the feed, it is probably relevant.

[0:54:36.7] JM: Let's switch back to talking about the data infrastructure at a little bit of a lower level. You mentioned Druid earlier. Druid is a database that's growing in popularity, Apache Druid. You said Pinot was perhaps the closest relative to Pinot in terms of architecture might be Druid. Can you give a comparison between Pinot and Druid?

[0:55:00.2] AP: I can say one thing. I think Apache Druid is awesome and it has a really great community. Probably at smaller scale, Druid totally gets the job done. There's some differences in say all the data structure and how the segments are stored and so on. Really, I think the core takeaway is that Pinot works at an entirely different scale. I mean, if you're really interested in nitty-gritty numbers and so on, there's comparison benchmarks, there was a segment paper about Pinot. You can just dig into this.

Just to give you a feel for how Pinot operates at LinkedIn, right? It serves both internal use cases, right, as the anomaly detection and root cause analysis and further, but also much remember facing or external, or use cases like statistics about well, the companies that brought this up many times already.

Pinot as a whole system, of course multiple clusters of those at LinkedIn, they operate – I took some notes where it said for example, right now we're running about 60,000 clicks per second on Pinot. Or it ingests over 1.4 billion records, like Kafka events per second. These are updates in near real-time to the system. Ultimately, when you're serving all this data because it faces the

members of LinkedIn, it also has to be really, really fast. You're talking about tail latencies of less than 10 milliseconds.

[0:56:33.0] JM: I find it interesting because I've been covering these different data infrastructure tools. What seem to be the case is if you look into any problem that seems to be a niche in data infrastructure, if you look a little bit closer it's actually a gigantic opportunity. When Druid first came out it was like, “Oh, it's this database that’s just for operational analytics or whatever.” It actually turns out to solve a gigantic array of use cases. Do you have a perspective on what the broader array of use cases for Apache Pinot is? Can you describe in some detail what are the use cases where Pinot is really the niche that is worth using for this use case?

[0:57:21.4] AP: Yeah, it turned out that Pinot is essentially the niche to this entire platform at LinkedIn. Of course, it started for high-dimensional data, but now it pretty much covers any time series that there is in key business metrics at LinkedIn. Even you store data in there that are related to say machine learning models, like types of features that you store. Sometimes you use Pinot just to somehow put your data, so that you can explore it interactively, right? That fast, speed so that you don't have to sit down and essentially wait for some Hadoop process to run through to answer your queries.

You end up using Pinot directly, or maybe a system like Presto, right? That federates queries across different platforms. Pinot has really grown to be one of the core elements in LinkedIn's data infrastructure.

[0:58:15.5] JM: I want to begin to wrap-up. You were the creator of ThirdEye?

[0:58:17.9] AP: Not really. I mean, it takes a village, right? First of all, it started sometime before I joined. Essentially, the funny part about this that ThirdEye pretty much started almost as a demo for some of the capabilities of Pinot, mostly it's real-time slicing and dicing of data. For data set that is tens of gigabytes or even terabytes in size. Then you can aggregate and drill down in it and render heat maps and other things within seconds. It's essentially using interaction.

This demo immediately actually got traction, because it well, delivered a lot of value for people that were doing these also are just looking at – it's just key business metrics they were doing monitoring tasks. It really snowballed from there and this became its own effort. When I joined the project, that was just over two years ago, really the first main use case was this anomaly detection and it is up to this point. Then the part that I contributed a lot to was then this interactive root cause analysis; really the ability to go from okay, here's an outlier in one metric that might be interesting to saying here's the other related things that either show outliers, or have some surprising behavior and then also of course, integration with all kinds of other systems like hey, here's your JIRA tickets, here's your configuration changes, your code deployments, external holidays and so on, things that might explain why this outlier happened.

[0:59:42.8] JM: Okay. Well, we're basically out of time. Maybe you can close up by just giving any vision for the future of these two projects that you're building in the near future, either Pinot or ThirdEye?

[0:59:56.7] AP: I mean, ThirdEye has itself a lot of avenues. Of course, you end up providing business intelligence inside of LinkedIn. Well, it turns out that this business intelligence might also be interesting for actual enterprise customers of LinkedIn, for example. This is evidence that is being explored.

Another thing is of course for Pinot, there's always another way to improve scale or reduce the amount of data latency and so on. There's lots and lots of work to do and we're certainly not going to get bored.

[1:00:29.0] JM: Well Alexander, thank you for coming on the show. It's been really great talking to you.

[1:00:31.4] AP: Thank you, Jeff.

[END OF INTERVIEW]

[1:00:36.8] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can data-driven websites and professional web apps very quickly.

You can store and manage unlimited data. You can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your site's functionality using Wix code APIs and your own Java script. You don't need HTML or CSS.

With Wix code's built-in database and IDE, you've got one-click deployments that instantly updates all the content on your site. Everything is SEO-friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There is plenty that you can do without writing a line of code, although of course, if you are a developer then you can do much more. You can explore all the resources on the Wix code site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix code experts.

Check it out for yourself at wix.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you could do with Wix code today.

[END]