Episode 10: An Open Road to Swift Dataframe Scaling
Host: Nicole Huesman, Intel
Guests: Devin Petersohn, Intel; Alex Baden, OmniSci

Nicole Huesman: Welcome to *Code Together*, an interview series exploring the possibilities of cross-architecture development with those who live it. I'm your host, Nicole Huesman.

Today's world is fueled by vast amounts of data and the rise of data science. Understanding a typical data science pipeline is key to deriving maximum insight from this data.

Over 60% of a data scientist's time is spent pre-processing and cleaning data. Wrangling this data is difficult and time-consuming. And to make things even more challenging, these developers are continually bombarded with new tools. We're joined by two experts to help us better understand this end-to-end workflow and to share strategies to address some of these challenges.

Devin Petersohn, machine learning engineer at Intel, joins us today to talk about his experiences. Hi, Devin.

Devin Petersohn: Hi. Thanks for having me.

Nicole Huesman: And Alex Baden, technical director at OmniSci, is also with us to share his insights. Welcome, Alex.

Alex Baden: Hi, Nicole. Great to be here.

Nicole Huesman: Devin, Alex, I'll hand the discussion off to you. Can you share your insights into the typical data science pipeline, some of the challenges you're seeing and some of the solutions you propose?

Devin Petersohn: Yeah, great, thanks Nicole. The typical end-to-end data science pipeline often starts with data coming in from the wild, and sometimes that data is dirty or messy and it's not super well-structured.

Data scientists actually spend the majority of their time, as you mentioned, cleaning and preprocessing their data just so that they can use it with the tools that exist. So, once the data is cleaned, it then goes through some machine learning pipelines for model building, and then eventually model serving in the production environment.

Alex, do you want to talk a bit about what you've seen in terms of the end-to-end pipeline and how you are dealing with it at OmniSci?

Alex Baden: Sure. So, you know, in our experience, 60% of the time just getting started with a new analysis product is spent cleaning the data, organizing the data, getting the data loaded. And there are a number of ecosystems that have developed around these processes.

You know, one of the most prolific ecosystems is Pandas, the Python dataframe API. And these are software packages that are used by, I don't know, Devin, hundreds of thousands of people probably across the entire planet. And, you know, they're phenomenal because they're community supported, they solve problems across a wide variety of software

---

domains, and there's a wide variety of packages available. So, if you want to do something, it's there.

These tools do have a downside though, which is, as they've gotten bigger and older and more mature, they are more rigid, and they don't scale. And so, this is where a product like OmniSci comes in.

Devin Petersohn: It's interesting that you've also noticed this challenge that data scientists have where tools are kind of getting more and more difficult to use as they scale. Pandas is used, you know, very, very widely and actually ends up being more of like a Swiss Army Knife where it does a lot of things, but it doesn't do any of them the best, right? So, it solves a lot of different problems, but it's not necessarily the fastest tool for any individual one of those tasks.

And so, the fact that it doesn't scale and the fact that there's kind of this missing link in the data science pipeline, where we need a scalable Swiss Army Knife, if you will, that solves a wide variety of problems and can fit that need that people have for being able to clean their data and interact with their data.

Another interesting observation I've had is that data science tools are more and more being developed without giving the user a real feel for having control of their data or a solid grasp on their data. So, data scientists are more and more having to wield these tools that it's kind of abstract what's going on underneath the hood and they don't really get any insight into how their data's actually being manipulated.

So, I think these challenges are really kind of interesting because it's not a really common problem that a lot of people are talking about.

Alex Baden: Yeah. And part of this problem is that we have these mature software platforms that people are using that have problems, and often the solution to a problem spawns a new product or ecosystem, right?

So, I lead an engineering team here at OmniSci, and one of the most important parts of my job is, tongue in cheek, when to say "No," right? But when it comes to doing software design, you know, I think one of the most important things is to avoid the urge to throw everything away and start from scratch, and to kind of look holistically and say, okay, what do I have that's working? What do I have that I can build on? How can I incrementally move sometimes big increments, right, but how can I incrementally move to something that meets the new needs? Because what happens is you end up building something new, which is really cool, but it comes back to that 60% of cleaning and ETL or ELT.

And you want to be able to plug into that ecosystem because that's the kind of hard, gritty, not sexy work that is just unbelievably important in being able to actually use this stuff as a data scientist or an end user, an analyst, whoever you happen to be, right?

Devin Petersohn: Yeah, completely agree. The fact that so much of the time is spent by the data scientists actually looking at intermediate results and building their queries and cleaning things, those really don't have anything to do with optimizing execution.

Of course, execution time is extremely important. We should obviously try to optimize our systems to run fast, but so much of that time is actually the data scientists kind of doing everything by hand, that the system performance is almost inconsequential in terms of how much time the user is actually spending at the computer, right?

They're spending a lot more time thinking about the next query or looking at this plot that they've just created. And they're not spending as much time necessarily waiting on that compute because, you know, as we mentioned, things are getting fast, they're just getting a lot less usable.

Alex Baden: So Devin, you know, my background is in databases and sort of old school computer science, you know, parallel programming. And one of the great things about working with you and your colleagues at Intel is all of the knowledge and experience you bring around dataframes. Can you walk us through what a dataframe is, how it's different from databases?

Devin Petersohn: Yeah, so, I'm happy to talk about the differences here.

This is work that I have been doing as a grad student at UC Berkeley as well. And working with a lot of the people at Berkeley, you know, we brought in a bunch of different people from different domains, databases, spreadsheets, machine learning. And we kind of came to this conclusion about what dataframes actually are and how they compare to databases.

The interesting part about dataframes is that they don't require you to define your data schema first. So, in a database, basically you have to specify the schema before you can start putting data into the database. For most databases, this is the case. In a dataframe, there is no requirement for the schema to be known up front. Actually, the system can go in and determine the types from the data itself.

Pandas does this, and this is a really useful feature for working with this dirty data that I was talking about before. So, you may not know the schema for your data before you ingest it. And what Pandas and other dataframes do is they basically determine the type of the data from the data itself.

Dataframes also have a lot of other properties, like they support matrix-like computations and they also support some spreadsheet-like operations as well, or spreadsheet formulae, if you will. And, really, they do a very wide variety of things that you would need to do when you're doing exploratory data analysis and, you know, data cleaning and data processing.

So, what we've done is we've basically taken this definition that we have. We've also actually extracted an algebra. So, you have relational algebra for databases and SQL-like systems. For

---

dataframes, we actually extracted a dataframe algebra, which allows us to kind of rewrite all of the APIs that you might use in Pandas.

And it drills them down and rewrites them into something that we can optimize, effectively. And we use this theoretical basis for building a system called Modin. And Modin started as a drop-in replacement for Pandas, trying to solve all these problems that we noticed with systems that scale really well, but they're really, really difficult for new data scientists to come on board.

And with Modin, we exposed the Pandas API, it's a drop-in replacement so you just replace the import statement and it effectively uses all of the cores of your machine, or it can work in a cluster, and you get that scalability without needing to change your code, even, from Pandas.

There are a lot of challenges involved in that, and Modin really is just one part of the end-to-end pipeline that we were talking about at the beginning. Modin is a part of the Intel AI Analytics Toolkit, and effectively what this toolkit does is it gives users scalability by optimizing existing libraries for Intel hardware. It provides drop-in replacements for scikit-learn, and it has optimizations for many machine learning algorithms included. There's also an Intel-optimized Python that's a part of the AI Analytics Toolkit. So, basically, this analytics toolkit is designed to improve the experience of the developer on Intel hardware.

As I mentioned, Modin is a part of this, and with all of the optimizations that we've done in Modin, we noticed that it would be really nice if we could not tie Modin necessarily to having to execute in any one way. And so, one thing that we've done is we've actually exposed this dataframe algebra for other libraries to be able to implement.

And one of the things that we've been working on in collaboration with OmniSci is actually bringing OmniSciDB as an execution layer underneath of Modin so that we can run extremely fast and JIT-compiled queries while still exposing the Pandas API.

Alex Baden: Yeah, so we've talked about OmniSci a couple of times. So, OmniSci is an independent company.

We have a partnership with Intel, and we work closely with Devin and a number of other engineers at Intel. And we produce an analytics platform. And essentially what that means is we take large quantities of data and we make it available to end users, whether that's data scientists, analysts, through a number of different products.

We have OmniSciDB, which is a GPU and CPU high-performance parallel database. We have OmniSci Render, which allows us to render results from the database directly as PNGs and serve them to the user. And we have OmniSci Immerse, which is a web client for the database and rendering engine that ties together everything and provides interactivity.

We also have a number of data science tools built in Python that we've collaborated with folks at Intel and Quansight Labs and a number of other open source communities. So, we're

---

very excited about this collaboration with Intel on this work with Modin as part of the Intel AI Analytics Toolkit. Most of that right now falls under OmniSciDB, which is the product that I work on, the team that I lead.

So, you know, databases are very complicated systems traditionally, and even with the rise of NoSQL databases, remain complicated. But the two sort of major components of a database are a system or a component that takes SQL and converts it to machine code, and a component that loads data and runs that machine code from the SQL query over that data.

And so OmniSciDB has both of those components: the SQL compiler and a high-performance parallel execution framework, which we'll talk about a little bit later. But, one of the issues that we've seen is systems like OmniSciDB are quite good at these data cleaning operations, and, in fact, a lot of these Pandas operations, right?

One of the predominant operations that you do in Pandas is a GroupBy which you can kind of think of as a MapReduce, right? Or at least a Map. And we're able to do that in parallel across a bunch of different devices. We've built the framework for that because GroupBy is very, very important for our visual analytics use cases.

The problem is that specifying these operations in SQL becomes very gnarly. It's one thing if you want to do some sort of mapping exercise or create a bar chart. It's another thing if you've got a GroupBy 15 columns or something like that, 15 individual attributes, and you want to compute other aggregates and, you know, other operations on top of that. And so, SQL is probably not a great language for expressing this.

But we were able, both because of the architecture of OmniSciDB and the architecture of Modin, we're able to strip the SQL layer off OmniSciDB and bring the Modin relational algebra / dataframe algebra directly into OmniSciDB at our relational algebra layer. This essentially eliminates the friction point between a high-performance OLAP SQL engine and the Python data science ecosystem, and it allows for very slick coupling between the two. So, you don't have to throw out all your tools. You don't have to write everything in SQL. You can basically keep doing what you're doing, but get much, much better performance on a very large number of workloads.

Devin Petersohn: So, how would you say that Modin and OmniSci compare to other things that exist out there?

Alex Baden: You know, before I answer, one thing to note is that Modin and OmniSci together don't shut out other frameworks, right? Just to give a couple of examples, OmniSci integrates very closely with other community projects, namely Apache Arrow.

It has many components, but we use Apache Arrow for their serialization format and to be able to take data in one process and move it to another process, which may have been a different language, may be on a remote server, may even be on the same server. Arrow exposes serialization formats that allow us to use interprocess communication APIs on CPUs and GPUs to basically zero copy move data between processes.

You know, OmniSciDB is also open source. There are a couple of enterprise components that are not available in the open source version, but query execution, SQL parsing, users, permissions, the Arrow APIs, those are all available in the open and, in fact, developed in the open. You can go to GitHub and look at our code.

And so, what's nice about that is you don't have to abandon your existing ecosystem to move to some new ecosystem. You can bring in OmniSci and Modin and gain access to those performance enhancements and new capabilities without having to rewrite all of your code, buy new hardware, various other things, move to a completely different workflow, right?

Devin Petersohn: So, Modin is actually designed to integrate with the rest of the Python ecosystem, and it is designed to kind of run on the hardware that you have rather than requiring you to buy a certain setup or spin up very expensive instances in the cloud. I'm really glad you brought that up because that idea of scaling data scientists where they're at is that the core of Modin's design.

Alex Baden: I think the point about the cloud is a great point because the cloud has numerous advantages, of course. Not discounting any of the advantages of the cloud, but there is a very interesting paper from a couple of years ago by a guy named Frank McSherry, where he talks about scalability at what cost, and he develops a metric called COST, which is essentially the penalty you pay when you go to scale out something, when you go distributed.

And we have a great example of this with OmniSciDB, and there's a tech blogger named Mark Litwintshik, who takes this New York city taxi rides dataset. It's a bunch of individual taxi rides in New York city. And by a bunch, I mean 1.1 billion over a period of a couple months, perhaps a year, maybe more. So, Mark has run this benchmark on a wide variety of databases / data analytics systems. Recently, Mark ran OmniSci on a 16-inch MacBook Pro. So, this is a MacBook Pro was an Intel I9 CPU, 64 gigs of RAM. And this is only using that Intel CPU on the MacBook Pro. And we essentially beat out every other distributed and single-node CPU solution on his benchmarks page with the exception of, interestingly enough, kdb+ running on Xeon Phi.

So, I think that's a good example of how the cloud can help you scale to numbers of users. The cloud can help you scale to extreme data sizes, and really that's starting to be petabytes and exabytes, not so much terabytes, right? But you do pay a cost. And it's important to keep that in mind, particularly if you are a smaller operation, you're resource constrained. There are ways of eliminating this cost with different tooling.

So, some people may be listening right now and thinking, "Oh, OmniSci, isn't that a GPU database?" In fact, we were formerly known as MapD, the massively parallel database. And, you know, we essentially made our name on being a GPU database, but it turns out that being a GPU database means a couple of things that, with a little bit of work, extend to other systems, right? Other GPU systems, CPUs, et cetera. And essentially there are roughly two

---

things that you need to be a GPU database. You need to be able to generate GPU machine code, and you need to be able to run queries in parallel.

Well, if you can generate CPU machine code, being able to run queries in parallel on modern CPU hardware, it gives you phenomenal performance. I mean, modern CPUs typically start at four cores. Most people have eight. With hyperthreading, now you're up to 16 threads. And with some of the new Xeons, hundreds of threads on a 1 or 2U server chassis, that's easy. You can go out and buy that off the shelf or get it from a cloud provider.

And so, you know, we leverage the LLVM compilation framework and smart parallel execution engine to, basically, take everything that runs well on GPU and allow us to run well on CPUs too. This also means that we're able to run on a wide variety of different GPUs, right? Because unlike CPUs, each GPU manufacturer has different instruction sets and different code generation is required, but we're able to generate code for a wide variety of platforms, which again, means that you don't need to buy new hardware to use this solution.

Devin Petersohn: I think that's a really good point, you know, not needing to buy new hardware to get better performance is really important in the current state of data science. What I've noticed is that data scientists, they typically don't care how the code executes or where it's executing. Typically, they just want the results and they want them fast, right?

The fundamental goal of data science is to extract value from data. And so, when you start pushing these requirements that you need to know whether you're on a GPU or CPU, you need to know, you know, how many nodes do you have, or are you in the cloud, or are you on some on-prem cluster. When you start pushing these requirements onto data scientists, all of a sudden, your data scientists are, you know, no longer doing data science, they're actually doing a whole lot of provisioning of resources and they're doing some orchestration themselves, even, in some cases.

And this Modin OmniSci collaboration really lets the user continue doing what they were before, even using the same notebook between different clusters because OmniSci works on the hardware that they have. And Modin does too, by extension.

Alex Baden: Yeah, and we should note that, you know, ideally, the more decoupling you have between software and hardware upgrade cycles the better, right?

So, if you're running now on hardware you have, and new hardware comes out, one of the nice features of OmniSci is we're able to take advantage of features and new hardware by extending our compilation framework. It doesn't require a wholesale rewrite. It doesn't require us to go out and make significant changes.

And so, this allows you to, basically, make your hardware decision independent of the software that you're running, to a much larger extent than other systems.

Episode 10: An Open Road to Swift Dataframe Scaling
Host: Nicole Huesman, Intel
Guests: Devin Petersohn, Intel; Alex Baden, OmniSci

We were particularly excited about the announcements that Intel made at Hot Chips earlier in August, at the Hot Chips conference.

So, there are a number of Intel-specific optimizations that we're pretty excited about in the works now, some available now and some coming soon. We'd probably need another podcast to talk about all of them, but I'll just touch each of them briefly.

One of the things that has been great is, and I'm speaking now as a developer using Intel hardware, is the Intel developer ecosystem. I mean, I think a lot of developers hopefully are already aware, but should be aware, Intel provides a large number of tools to help us offer development. And so, one tool that we've been using pretty heavily for the past year at least is the VTune Profiler. And that allows us to really maximize our performance on Intel hardware. And it's caught a lot of interesting things that were very difficult to see.

So you're able to do everything from looking at your CPU instructions and whether you're making the best use of the CPU instruction caches and instruction pipelines to memory bandwidth, memory accesses, all of that stuff. So, we've made significant use of VTune and we're actually able to plug VTune into our compilation framework and profile our Just-In-Time generated code, which is very cool.

On the hardware side, we're very excited about Intel Optane. You know, OmniSci has a storage component, so we're not necessarily an in-memory DB. But primarily our computation is done in memory because we're a parallel database executing across multiple threads. Memory accesses are very important to us. And so Optane allows us to bring more data into memory.

Now we have to manage that memory. It's not quite as simple as just make your memory pool bigger, but it's actually more powerful because of the persistence it gives you, right? And so, anybody who's worked with really big data sets knows it can be painful to load multiple terabytes into memory. At a certain point, you're just bound by physics. The persistence of Optane lessens that pain should you want to change server configuration, restart, et cetera, et cetera.

And then last, but certainly not least, Intel's GPU lineup is very exciting for us. Like I said earlier, our compilation framework allows us to generate code for CPUs and GPUs from any manufacturer. And some of the announcements at Hot Chips are very exciting and point to a very exciting future for this heterogeneous computing space.

Devin Petersohn: So for those of you listening, if you're interested in trying Modin on OmniSci, it was recently released as a part of the beta of AI Analytics Toolkit, which is a part of Intel's oneAPI. You can get access from the Intel website. So, I would suggest you just Google Intel AI Analytics Toolkit, and it will bring you to a page where you can request access.

Later this year, we will have the gold release, which should be happening in the fall or winter. We would love to hear your experience with this. And if you have a chance to, you

---

know, come raise an issue, you can go to github.com/modin-project/modin. We'd really love to hear your experience and how you're using it.

So feel free to come by and talk with us.

Nicole Huesman: As Devin just mentioned, the Intel AI Analytics Toolkit, available in beta now, is a great thing to check out, and we're excited for its gold release later this year. Alex, thanks so much for your insights. Such a great discussion.

Alex Baden: Thank you, Nicole. I appreciate being here and it's a very exciting space to be working in.

Nicole Huesman: And Devin, it's been great to have you on today's program.

Devin Petersohn: Thanks Nicole.

Alex Baden: For all of you listening, thanks so much for tuning in. Let's continue the conversation at oneapi.com. Until next time.