Code Together Podcast Episode #33

Title: Solving for Productivity in Data Science with Modin and AI Optimizations

Transcript

Radhika (00:04):

Welcome to Code Together, a discussion series exploring the possibilities of cross-architecture development with those who live it. I'm your host, Radhika Sarin.

In recent years, artificial intelligence, machine learning and data science have seen an exceptional growth and acceptance. The rapid advancement in these fields have allowed data scientists to take advantage of these technologies to arrive at meaningful insights. However, it's not been an easy task to optimize machine learning infrastructures to allow data scientists to focus on their core expertise. Today, we will discuss how certain tools and hardware optimizations are not only saving time, but also enabling data scientists to be more productive.

We have with us, Devin Petersohn, co-founder and CTO of Ponder. Welcome Devin.

Devin Petersohn (01:04):

Thank you.

Radhika (01:05):

We also have Areg Melik-Adamyan, Principal Engineer and Engineering Manager at Intel. It's great to have you back, Areg.

Areg Melik-Adamyan (01:15):

Thank you for inviting me.

Radhika (01:16):

So, let's get started. Devin and Areg, can you talk about some of the challenges that data scientists face and how are some of the tools alleviating that?

Areg Melik-Adamyan (01:27):

Okay, Radhika, this is a really good question. In my job, we are focused on generic data science and we are targeting everything starting from the data acquisition, data ingress, data pro processing cleaning, preparing data for the further machine learning processing, like running classical machine learning algorithms, regressions, k-means, and the further step will be deep learning algorithms. And we found that the way the data scientists look at these problems are completely different than the data engineers, these problems. So, data scientists are used to their APIs and methods that they like. For example, in data processing world, the factor standard is... handles API. They use Pandas Python library, to inverse data, to do data manipulation, prep, processing, and cleaning. So, you can move data further to the libraries like you Scikit-learn, XGBoost, etc. And one of the problems there is that these type of libraries and the Pandas API is very convenient for usage, but it is single threaded. Cannot run in a performative way, especially if you're doing complex optimizations and that's where solutions like Modin kick in. And I will go over to Devin to talk about Modin.

Devin Petersohn (02:58):

Thank you. So Modin is a drop-in replacement for Pandas, and it kind of comes from this idea that there are things that data scientists should care about And there are things that data scientists shouldn't care about. Data scientists are fundamentally just trying to extract value from data. So they don't need to worry about things like, how data is laid out, how data is partitioned, how to best schedule jobs or anything like that, Right? None of that is really directly tied to this goal of extracting value from data. And so a lot of existing tools that work on the large scale, they're not very usable. They kind of just require you to learn a lot more about distributing computing to understand how the data's laid out. And it's really a big overhead just to kind of get started with these tools. And so we have this silo in a lot of organizations where there are different people who work at the small scale than that work at the large scale.

Devin Petersohn (03:59):

And the way that I like to frame it usually is that there are tools that are usable and there are tools that are scalable. The tools that are usable are like Pandas, right? Everybody's using them to kind of iterate quickly on their data and to quickly gain insights from their data. On the other hand, we have tools that are scalable and those are tools like big data tools, big data infrastructure tools. They really give you a lot of scale and performance. You can add more machines, you can throw more money at the problem and you can solve bigger problems if you throw more money at it. This is a big disconnect though, because the people who are used to working on tools that are on this smaller scales, can't use these tools that are at the larger scale. So Modin's goal is to basically break down that barrier.

Devin Petersohn (04:48):

It's to say, data scientists know enough to be able to do analytics on their data sets, no matter how large. We can just replace the import statement with Modin. Instead of importing Pandas, you would import Modin.Pandas, and then Modin will just take care of all of the scaling, all of the scalability problems, questions like this. I mean, we know that engineers can really tune these systems and get good performance. The problem is that machines are generally good at choosing intuitive defaults. So what we've done in Modin is abstracted away a lot of these complex decisions that you have to make about where to put the data, how to lay out the data and that sort of thing, and really made a system that's both usable and scalable.

Areg Melik-Adamyan (05:30):

Yeah. I think you touched very important point at that three aspects. One is the APIs the data scientists are using and used to, the second part is about execution. And execution always connected with data aspect, so where is your data, how it is laid out? Do I need to do partitioning, charting, etc. And data engineers and the systems that are scalable are explicitly leaking those notions to the upper level, to the data scientist level and requiring them to consciously make decision about distributed processing, which they often don't have expertise to do. And the second part is about the execution. Okay, you will have the API call. How it's going to be executed? Do I need to do it multi-threaded, multi-processing, distributed? If yes, how to organize that. And well-designed system should not leak those instructions up to the upper level.

Areg Melik-Adamyan (06:31):

Though it should provide ways, if you know what you're doing, to instruct the system how to do that. And one of the advantages of Modin is this well-designed, layered architecture where on top of the system, you have user-facing APIs and Modin out-of-the-box supports, Pandas API, 100% functional and about, if I'm not mistaken, 90% performant. Also you can have plugged in SQL font so you can use SQL interchangeably. And there's a possibility to plug in domain specific language, to leverage all the layered architecture. And this information hiding, and layered API is giving possibility for system engineers to create systems that are using the same APIs, but can run on completely different types of hardware, both locally and in a distributed manner, and even in the cloud. Okay. So I think it'll be good to have an overview in Modin architecture. In general, the layers and information exchange and how this layer architecture allows users to do performant processing of data. Devin, do you want to take the lead here?

Devin Petersohn (07:57):

Yeah, sure. I'll jump in here. So Areg, you already mentioned a bit about the API layer and how we can actually just treat the API as a different layer. The layered architecture does give us a lot of power, so the Pandas API, a SQL API... We even have been toying around with an experimental spreadsheet interface. All of these things are just different ways of expressing what you want to do. And so the view that we've taken for Modin is that fundamentally, the tool should not get in the way of what you want to do.

Devin Petersohn (08:32):

It should just be that you're expressing computation, right? Or you're expressing what you want to do. And then the computation is just handled for you. Speaking of APIs, one thing that I've seeing people say about Pandas is that the Panda's API is not scalable. And I think that there's a big distinction between Pandas not being scalable, which is true, and the Pandas API not being scalable. The API is massive, of course. And it's very, very difficult to kind of go through and individually parallelize all of these things. The layered architecture of Modin actually gives us a lot of power to reduce the surface that we need to parallelize. And I'll talk a little bit more about that, but an API is just a way of expressing what you want to... It can't be scalable or not scalable. It's...

Areg Melik-Adamyan (09:16):

Yeah, scalable API is kind of, I don't know, oxymoron to say. Execution can be scalable, or data processing can be scalable. The API is just an API.

Devin Petersohn (09:25):

Right, right. I think that it's kind of a mistake to tie the Pandas API to the Pandas execution and say that Pandas API cannot be scalable because Pandas isn't scalable. Pandas is just not optimized and Pandas itself... Actually, there are a lot of leaky details about how things are implemented, but what we've done in Modin is done a lot of things like separating the logical order from the physical order. And a bunch of different concepts that you have in databases that when you try to apply them to data frames, it takes a little bit of finesse to make it work, but you get so much more benefit whenever you're not leaking these details, you get so much more benefit whenever you parallelize

things. So there's a lot of things that we can do once we decouple the API from the execution like we have in Modin.

Devin Petersohn (10:14):

And so the internal layers... The kind of narrow waist of Modin is the data frame algebra, which was developed at Berkeley as a part of my PhD thesis, effectively. So what we can do is take the over 600 operators in Pandas and express those in around 16 operators. And the 16 operators are very powerful, very expressive, but they also give us the ability to express things in an optimal way. We can do things like suggest type hints, for example, or assume an output order because we've done a lot of the work that it takes to kind of identify this narrow waist algebra. And the algebra is actually what we can implement in a scalable way. Then all it is just a matter of basically translating the Pandas API into this algebra. And once you have a scalable implementation of the algebra, you have a scalable Pandas effectively because of that narrow waist.

Areg Melik-Adamyan (11:11):

Yeah, and as we know in a world design system, which layer provides a different abstraction from the layers above and below, which is nicely done in Modin, yes. Modin is not ideal, but this layer way is done very nicely. And that's what we're leveraging. If you follow a single operation as it moves up and down to the layers, obstructions change with each method. And that's what makes Modin powerful. It's deep. One of my favorite examples is Unix file functions, manipulation functions. There are only five of it, but each function is very deep, because Linux or Unix is comprised of very well-defined obstruction layers.

Areg Melik-Adamyan (11:48):

And Modin following the same tradition, providing different obstructions, different layer. Like if you want to do... get a frame read, then each layer is dealing with its own level of obstructions. From the user perspective, it's just a Pandas call, but they would know that it needs to go and find the file. It needs to do some assumptions about file. How many workers needs to be involved? Are we doing in a distributed way or just in a multi-processing way? Perform the parallel read from the source, fill out some metadata that can be used in the further operations, and this is the power that Modin is providing to be able to optimize on the needed level or the needed task.

Devin Petersohn (12:37):

Right, so this doesn't actually just limit us to being able to use Ray and Dask, which are commonly associated with Modin running on. We can also leverage existing high performance database systems.

Areg Melik-Adamyan (12:51):

Or data processing systems in general.

Devin Petersohn (12:53):

Yes. I mean, as a long as you can implement the algebra or some subset of the algebra, you can have a Pandas API on top of it. And so we have of course OmniSci, which has been under development for quite a while now, actually.

Areg Melik-Adamyan (13:07):

Yeah. It's more than a year, right?

Devin Petersohn (13:08):

Yeah. It's quite robust I think at this point. OmniSci is a very performant database that can run things on CPU or GPU. And so we have the ability to be heterogeneous on these engines. Modin isn't tied to really any specific execution model or anything like that. We have just a lot of power by having this narrow waist of the data frame algebra.

Areg Melik-Adamyan (13:32):

Yeah, on this data, this layered architecture with the data from algebra, allows to use OmniSci in different modes. On the layer that is responsible for the data from algebra, we generate a query plan. Then it goes to the OmniSci DB and because OmniSci DB is able to do heterogeneous execution, boom. Now we can run in our CPU or GPU, or both on CPU and GPU, which is quite unique in the world of data frames.

Devin Petersohn (14:03):

Yeah. So one of my thesis topics was actually around data frames. And when we talk about data frames, everybody's got kind of their own idea of what a data frame is. When I'm talking about a data frame, typically I'm talking about the kind of our and Pandas data frame world where... I think our team at Berkeley was the first to define a data frame data model and defining this and being able to express it in math, having the data model and having the algebra. This actually gives us a lot of future really interesting work because we can start to optimize things once we can express them in math and the mathematical foundation of Modin, it's there. Not only does it give us this ability to go in and optimize things, but it also gives us the ability to run computation on heterogeneous systems or execution.

Areg Melik-Adamyan (14:56):

Yeah, I think for the listeners who are worried about it from algebra, I recommend them to go with your papers, but the analogy is quite good. The same way as databases... We're using relational algebra is a foundation of the further work cause there's again, very well-layered architecture on the top. You have database manipulation language, which usually is a SQL, then which is being translated into the relational algebra and that's the point of entry to the real databases and real execution.

Areg Melik-Adamyan (15:25):

The same is here. Data from algebra provides you the entry point where based on the additional information, about the execution part, about the system you are running on, you can slice and dice

and optimize the way needed to fulfill some service level objective. You want to do it fast. You want to do it minimally involving course in the certain time, etc., etc., all those optimizations are now possible. Because you have this very powerful and explicit intermediate language called data frame algebra. And I propose to switch gears a little bit and talk about some limitations of Python that we're encountering in Modin, and in general the advancement and trends that we see, and preparing to implement in Modin.

Devin Petersohn (16:23):

Sounds great.

Areg Melik-Adamyan (16:24):

Okay. So let me ask you a quite controversial question like why Python? Why do people in data science use Python? I have my answer, but I would love to hear yours.

Devin Petersohn (16:38):

Yeah, it's a great question. Some people have said that the reason Python is so popular is actually because of Pandas. Now, Python is not a strongly typed language. When you're doing data science that can actually be powerful because sometimes you get data in that's not clean, you get data in that's not, fully adhering to your desired schema and you have to do something with it. Before data frames it wasn't really that easy to load something in that had multiple different data types within a column, for example. Python as a language can do things like throw run time type errors. You can throw type errors at run time and Pandas will try to infer the type of a column at run time. These are extremely powerful when it comes to trying to clean data and trying to analyze data that isn't necessarily well-typed or well-schemed.

Devin Petersohn (17:39):

And so I think at least, this is somehow contributing to the major rise in Python, the major rise in Pandas, because these are concepts that we've worked on formalizing in terms of the type inference rules for example, and the rules for how to maintain order. All these components of the data frame are things that we've worked on formalizing, and so we have a lot of challenges in trying to scale some of these behaviors that aren't present in other systems. There aren't a lot of systems that let you have really loosely typed columns, and that don't really adhere to any specific schema, but still allow you to do things like joins when your schema is not well-defined. Pandas is... It allows you to do that.

Areg Melik-Adamyan (18:24):

Yeah, it's really good point. Python is a language without bones. So it's ideally created for adopting DSLs, domain specific languages, because Pandas is a domain specific language, TensorFlow is a domain specific language. All these popular frameworks are domain specific languages and Python, it's a ideal candidate, ideal language to adopt to the requirements of DSL. It'll be creative if Python out-of-the-box was distributed concurrent programming language. Then lots of problems that we're now dealing by creating these layered architectures and hiding execution from the top level, would have been gone.

Areg Melik-Adamyan (19:05):

But we have what we have and the problems with the concurrency where we're alone in the Python world. And because of that, systems like Dask, Ray, couple of others occur who are trying to fill the gap of this lack of concurrency and lack of distribution. So what do you think is going to happen in the future in terms of the execution engine in Modin? And why am I interested? Because as you know if Intel has Intel Distribution of Modin, which is part of Intel® oneAPI AI Analytics Toolkit, and to have the best optimizations for the Intel platform, we are constantly monitoring and optimizing Modin both in upstream and Intel® Distribution for Python for these various executional engines. So what's your take on this?

Devin Petersohn (19:59):

Yeah. So you asked about the future of Modin in terms of execution. One of the observations that I've had in the context of Ponder, Ponder is the company that's behind Modin. One of the things that I've noticed with a lot of organizations, is that these folks who do data science don't often get to choose their infrastructure. And so what we are doing is basically working on building Pandas on everything. Where it'll basically run on your hardware, software, on your deployments, wherever you run your data pipelines, Modin will basically allow you to run on those. And so we have proven good performance with systems that perform SQL queries. We have good performance on things like Dask and Ray. So what we're really trying to do is enable data scientists to just run Pandas on their company's chosen platform. And we've seen a lot of value in some of the companies that we're working with, who want to migrate from one system to another.

Devin Petersohn (21:08):

What they're doing is they're using Modin... we're integrating Modin with the first system. The migration is happening at some point in the future and whenever that happens, we'll have all the testing in place, and everything will just be ready. We'll just turn a switch within Modin and then Modin will now be using system B. And it didn't take thousands of human hours to sit at the computer and go and type around trying to figure out how to debug all these corner cases in the new system, right? Because Modin knows how to translate these Pandas scripts down into both system A and system B. And so what we are trying to build and what we're trying to unlock is the productivity that you would gain from not having to do all of this extra boiler plate work, all of this extra translating. I mean, do we really need people whose job it is to translate Pandas to Spark?

Devin Petersohn (22:03):

That seems like a total waste of time to me. So what we really want to do is just elevate the level at which people think about things. Work in Pandas, work in SQL, work in whatever language you like, Modin will do the work of translating it down into the data frame algebra, which is extremely expressive as we've been mentioning. And then that data frame algebra will be implemented and translated to multiple other systems. So the Pandas on everything vision, isn't just Pandas, of course, it's really multiple frontends to multiple backends. But that's been the vision from the beginning, is this idea that data scientists can just sit down at a computer and immediately be productive.

Areg Melik-Adamyan (22:46):

Yeah, you touched really very important points. And one of the aspects that we are constantly working is standardization. And both Ponder and of course Intel are part of the Data APIs Standardization consortium. Intel is a founding member of the Data APIs Standardization Consortium. And currently this consortium is focused on standardizing two aspects of data science: standard array APIs and array exchange protocols, and data frame APIs and data frame exchange protocols. Array APIs and exchange is a little bit further, more advanced. The first version that's out is already adopted by most popular libraries like NumPy, PyTorch, MXNet, scikit-learn. So this... APIs standardization allows one of the first aspects of breaking from this jail that Devin was mentioning when dealing with the arrays. And the next step is data frame APIs, data frame exchange protocols that are coming, data frame exchange protocol RFC is published, and APIs will follow, and this will allow to use new standard APIs in various systems, and the vision that Modin has is becoming even easier.

Areg Melik-Adamyan (24:10):

Now you don't need to do end-to-end translation of various APIs to the Modin execution, but you can actually focus on a single standard API if the standard API is adopted into our various systems. I think often we see interest from Pandas, Modin, Vaex, and cuDF. So I think that these systems will be the first systems to adopt the standardized data from APIs, which will make first of all, data scientists' life much easier, but of course it'll allow us to focus more on the optimizations instead of doing a lot more translation and covering corner cases.

Radhika (24:53):

Devin, could you provide some resources for developers to learn more?

Devin Petersohn (24:57):

So Modin is completely opensource. The work we're doing at Ponder is opensource. You can go to Ponder.io and you can find all the links there, but Modin is on GitHub at github.com/modin project/modin. That's M-O-D-I-N. You can also PIP install Modin. We actually recently surpassed two and a-half-million installs since the project's inception, so it's growing and the community is constantly growing. There are lots of other tools and utilities on the Ponder company website and also the GitHub. You can also just email me directly at devin@ponder.io. That's D-E-V-I-N at ponder.io, or you can reach out over LinkedIn and connect with me. I'm always happy to chat about the things that we're doing and help folks get introduced into the Modin ecosystem.

Radhika (25:47):

Areg, do you have any additional resources or tools that our listeners can use from the Intel site?

Areg Melik-Adamyan (25:55):

Intel is a partner for the open sourcing so with all the optimizations and the work related to the performant backend, attaching and optimizing is also done in opensource. And you can see in the GitHub all the names, Intel folks that are working on this and optimizing and moving the community

forward. The product Intel is bringing to the market is called Intel® oneAPI AI Analytics Toolkit. It's based on the open-source versions of Modin, optimized by Intel® Extension for Scikit-learn*, Intel® Optimization for PyTorch*, and Intel® Optimization for TensorFlow*.

Radhika (26:33):

Perfect. Well, we're almost out of time. I wanted to thank both of you. Devin, Areg, it has been such an exciting discussion and it's been a pleasure to have you both. I would love to thank all of our listeners for joining us today. Let's continue the conversation on oneapi.com.

PMR- Attached
Transcript – attached
Audio file – attached

Code Together Podcast Episode #33
Title: *Solving for Productivity in Data Science with Modin and AI Optimizations*

Abstract

The rapid advancement in machine learning and data science fields have aided data scientists in arriving at meaningful insights. However, it's not been an easy task to optimize machine learning infrastructures to allow data scientists to focus on their core expertise. Today, we will discuss how certain tools and hardware optimizations are not only saving time, but also enabling data scientists to be more productive.

Learn more
RESOURCES | share resources + links

- Ponder.io
- Intel® oneAPI AI Analytics Toolkit
- Intel® Distribution for Python
- Intel® Distribution of Modin*
- Intel® Extension for Scikit-learn*
- Intel® Optimization for PyTorch*
- Intel® Optimization for TensorFlow*
- https://medium.com/intel-analytics-software/data-science-at-scale-with-modin-5319175e6b9a
- https://towardsdatascience.com/the-modin-view-of-scaling-pandas-825215533122

Guests

- **Devin Petersohn,** Cofounder and CTO of Ponder
- **Areg Melik-Adamyan,** Principal Engineer and Engineering Manager at Intel

Code Together Podcast Episode #33

Title: Solving for Productivity in Data Science with Modin and AI Optimizations

Keywords

- Modin
- oneAPI
- AI Artificial intelligence
- Machine learning
- Data science
- Data scientist
- Pandas
- PyTorch
- XGBoost
- OmniSci
- Python
- Opensource
- API
- Heterogeneous
- SQL
- domain specific languages

Hashtags:

- #AI #ML #DL
- #oneAPI
- #Modin
- #Pandas
- #PyTorch
- #XGBoost
- #Python
- #Opensource
- #API
- #HeterogeneousComputing
- #SQL
- #DSL
- #OmniSci

Social handles to ~potentially tag/engage:
https://twitter.com/ponderdata @ponderdata
https://twitter.com/modin_project @modin_project
@IntelAI @IntelDevTools
https://twitter.com/aregm @aregm
https://www.linkedin.com/in/devinpetersohn/
https://www.linkedin.com/in/aregm/

Code Together Podcast Episode #33

Title: Solving for Productivity in Data Science with Modin and AI Optimizations

Promotion timing – Aiming for April 7th, 2022, release.