EPISODE 1163

[INTRODUCTION]

[00:00:00] JM: Newer machine learning tooling is often focused on streamlining the workflows and developer experience. One such tool is BentoML. BentoML is a workflow that allows data scientists and developers to ship models more effectively.

Chaoyu Yang is the creator of BentoML and he joins the show to talk about why he created Bento and the engineering behind the project.

[INTERVIEW]

[00:00:29] JM: Chao, welcome to the show.

[00:00:31] CY: Thanks, Jeff. Thanks for having me.

[00:00:33] JM: Simple question. When I deploy a machine learning model, what's the difference between deploying that and deploying a microservice?

[00:00:41] CY: In our opinion it should be the same from the point of view of a DevOps. I guess a lot of the model serving tools in this space are trying to treat machine learning workloads differently. You will see tools like TensorFlow Serving. It asks data scientists to package their entire model into a standard format. And then the way they deploy it is essentially uploading that model to a directory and then the TensorFlow server will pick up the model and expose to an API. But in our opinion, that's not the right way to do it. We think DevOps should be able to deploy machine workload exactly the same way as they deploy and manage other microservices.

[00:01:21] JM: And so what are the main frictions that prevent that from becoming both being a seamless, as easy as the other?

[00:01:28] CY: I think the main challenge here is that when you're talking about deploying machine learning models, it's a process that involves multiple teams. Typically you have data

scientists training and producing the machine learning model. And then it usually takes another engineering team to come in and look at the code, the Jupyter Notebook and the produced model files. They will try to refine the code, build API around the model and make sure it has a monitoring, login, tracing, everything that's making it production ready. And then this will be handed over to DevOps for deploying, operating in production environment.

And what's so different is that data scientists actually wants to get feedback from those production workflows and they want to be able to repeat this entire iteration loop fast and being able to make changes to their models and to the serving logic very quickly and get feedback again. So that's quite a different pattern than how people used to deploy other microservices.

[00:02:31] JM: Tell me about what you're building with BentoML.

[00:02:35] CY: Definitely. BentoML is an open source framework for model serving. As I just said, when we look at the problem, it's really the friction that comes in between data scientists and DevOps where you always require another engineering team to come in and help the data scientists move their workload to production. What BentoML is doing, that is we are providing an abstraction for data scientists to easily create prediction services in a way that on one end it captures all the data scientists needs. It integrates well with all the machine learning frameworks and workbench products, like experimentation platforms where data scientists will be training and producing machine learning models and allow them to easily patch and wrap those models along with all the dependencies and Python code into a prediction service that's ready to be deployed into production.

And on the other end, we provide tools. Allow DevOps and data engineers to easily apply these models and make it accessible by other applications or backend services. So for example, the same model packaged by data scientists can be deployed as a Docker container and exposing the model through a high-performance API endpoint. Similarly, you can deploy the exact same model to a serverless platform like AWS Lambda or Knative on Kubernetes cluster. You can also apply the same model to batch offline surfing running the model in an Airflow job or in your CI/CD environment against test dataset or on distributed dataset on a Spark Cluster.

So regardless where you're deploying this model, the data scientist does not need to be aware of that. They're just using this high-level API provided by BentoML and describe what that input and output of my prediction service should look like. What are the machine learning models I need to access in this prediction service? And how do I actually pre-process the input data, making it ready for my model to write inference? And how do I return the expected data format to my upstream service that's consuming the predictions.

[00:04:47] JM: So if I'm a developer, how am I getting started with BentoML?

[00:04:52] CY: Great question. So BentoML is an open source project. You can BentoML on GitHub. And there we have a number of getting started guide and documentations. We also have a gallery repository that contains example projects built with pretty much all of the popular machine learning frameworks that you can follow along and create prediction service to serve your machine learning model.

The typical experience basically looks like this; BentoML comes in after your machine learning model training pipeline. So where you write the code to train the model, to do evaluation, all those parts remains the same. After you produce a model, you can use BentoML to basically write a prediction service class and specify which are the machine learning model I need access to and define the inference API endpoint that will be basically invoking the model. And all that are defined in Python.

So through this prediction service class, data scientists can easily persist it to a standard format. We call it a Bento. And then we provide another set of API that allow data scientists to load back this prediction service and invoke the Python API to run it against test dataset or expose it as an API server. So this Bento format is what we call a standard format for model serving. It captures all the dependencies, everything you need to reproduce the exact same prediction service and write in the consistent way across all the different serving and inferencing scenarios I described earlier, like online serving, serverless serving and batch offline serving.

[00:06:34] JM: So tell me a little bit more about why this is advantageous for me as a developer.

[00:06:40] CY: Yes, definitely. So without BentoML, you really need to package your model and manage that whole process kind of manually. What we've been seeing is that teams spend years building internal infrastructure to support their data scientist team. What they've built is usually a one-off solution that supports, for example, deploying an extrivous model into an EC2 instance. But now your data scientist might start asking for support for other ML frameworks. They are adding other dependencies to their model. They are pulling in other code repository in their company and use that in their serving logic. It just becomes really hard for them to keep up with all those demands from the data scientist's side.

And similarly on the infrastructure and deployment side, assuming you're deploying your model into an API endpoint. But before you deploy it, you always want to have some kind of CI/CD setup for your API endpoint. You want to test the behavior of that prediction service. And that, you will need some kind of wrapper around your model to allow it to run against the batch dataset. And all of that will require custom software to be built around the specific type of machine learning framework and a set of dependencies. That just drastically slows down the entire process for model deployment.

As I described earlier, data scientists wants to be able to iterate fast. They want to be able to get their model to production very quickly in a repeatable way. And that's the biggest challenge here. We are seeing a really bold surface of integration point on both the data scientist the machine learning toolkit as well as the DevOps and infrastructure side. And BentoML as a standard format in between really helps unifies that entire border lifecycle of model deployment.

[00:08:40] JM: Could you talk more about the common difficulties of a data scientist productionizing a machine learning model?

[00:08:46] CY: Yes, definitely. I think one of the most common things we are seeing is that data scientists are not necessarily familiar with all the best practice and the common tools people are using for creating production services. For example, you will need to expose, instrument your code in a way that DevOps can easily set up a monitoring dashboard to understand how this service is running. You want to define a way to log all your prediction requests and potentially feedback requests so that you will be able to consume that log data later on in your analytic pipeline or in your development workflow as well as like tracing or containerizing your API

server. So a lot of those things are really not what data scientists are trained to do. So they are relying on external teams to help them build that. And that really slows down their iteration cycle of getting models to production.

[00:09:50] JM: So BentoML works to streamline the difficult problems involved in that model deployment process.

[00:09:58] CY: Exactly.

[00:10:00] JM: So say more about the format of a model that gets deployed using BentoML.

[00:10:07] CY: So the Bento format, essentially, it contains the serialized machine learning model. It also contains all the dependencies and all the python code you have around this model that does pre-processing and post-processing. In BentoML, we actually do a number of really user-friendly features such as we automatically figure out all the Python dependencies, the packages being used in your training environment. And we are able to actually pinch down all the versions of those libraries and save them into a configure file in this standard format.

It will also contain all the Python code that's being used in your prediction service code including other local Python code being imported in a way that people can easily reproduce the exact same environment using this standard format. In this format, it also contains files that's making it easy for DevOps to use and apply this Bento file. It contains a Docker file that makes it easy to containerize an API server that hosts this model. It also contains a Python package configure file that people can easily turn this model into a pipeline package that can be used and installed in your Python backend application.

[00:11:29] JM: And how does BentoML make model serving more effective?

[00:11:34] CY: Do you mean on the performance side or more on the workflow and user experience side?

[00:11:40] JM: I meant on the performance side, but maybe you could go through the user experience side first also.

[00:11:46] CY: Definitely. I really think the biggest challenge is on the border workflow and life cycle management side, and that's what most people are using BentoML for. There's lack of standard for managing all the artifacts being produced by data scientists and how did you manage all those in the production environment and make sure everything still runs in a consistent way? But what we see BentoML's true value is in that transition from data scientist workflow to a deployment and DevOps workflow. One common concern is does data scientists know how to actually build a high-performance API endpoint, for example? How do you make sure the performance and the throughput and latency, etc., matches your application's requirements? So there's one technique we built into BentoML, is called adaptive micro-batching. And that can easily bring up the overall throughput of your API server to, in some of our tasks, 100x over a traditional Python web server hosting the machine learning model.

What it does is essentially once the API server receives a prediction request, instead of putting those requests into a model individually one-by-one, it actually groups those requests into smaller batches and run a batch of input data at a time. The reason for doing that is just because most of the machine learning frameworks and the numerical analysis libraries under the hood are taking advantage of CPU instructions or GPU accelerations to process a batch of data at a time.

Traditionally, when people are building model server and handling one request at a time, it's essentially doing the big for loop and processing those items without utilizing all the optimizations already built into your machine learning toolkits. So micro-batching really helps solve that issue. But the problem with batching is that as the server receives a request from client, it needs to figure out how long it has to wait for the next request to come to make a batch. So you're basically thinking about trading your latency for throughput. And for some workloads, it may not make sense for every request to wait for another second or two seconds for other requests to come in.

So what we've built into BentoML is an adaptive layer that actually runs the lightweight regression model. It can predict when the next request will come as well as how much time it will take for a certain batch size to process by your model. And that just helps the data scientist to get a really good enough performance without spending a lot of time tuning all the different

parameters around batching. And in fact, you will need to do that for different type of hardware you're using for deploying your model. And with BentoML, data scientists just don't have to do that, and we take care of all that.

[00:14:51] JM: So there's a wide variety of different machine learning frameworks to cater to if you're trying to build generalized machine learning infrastructure. How do you cater to all the different frameworks?

[00:15:03] CY: That's a great question. So when we designed BentoML initially, we are trying to make a really generic API for people to integrate with a new machine learning framework. It essentially defines how a machine learning model is being serialized to a file format and similarly how to load it back in a serving runtime. And by doing that, I think the advantage of open source really comes in.

The BentoML team really just builds integration for some of the most popular, the big ML frameworks like PyTorch, TensorFlow, Psyckit Learn, and we are seeing the community helping us to contribute support for a wide range of other machine learning frameworks and new exciting frameworks. Like we got contribution from Hacking Phase two weeks ago from the community, two months back we even got contribution from Apple, adding support for CoreML support. So yeah, having the community and a community of contributors helping us to build those integration really allow us to support more frameworks in the ML space as we're seeing so many new framework and exciting tools that's coming out all the time.

[00:16:18] JM: How does the workflow of a machine learning team change when they adopt BentoML?

[00:16:25] CY: Great question. Without BentoML, we are seeing teams that basically building a storage layer that stores all the machine learning models being produced by data scientists or the training pipelines they produced. But then they will have a separate repository managing all the serving code that will be loading in the machine learning model and exposing the model through an API endpoint.

One of the challenges with that workflow is basically the prediction – The pre-processing code are usually bound to certain version of the model. Data scientists, as they are developing a model, they sometimes make are making changes to the pre-processing code. And you really want to actually version the model together with that pre-processing code so that the prediction actually makes sense. But without managing the code and producing model together, we are seeing people run into all kind of weird issues after a model is being deployed to production. So that's the first issue BentoML helps solve. We actually version the model together with your serving logic and all the code associated with that serving logic so that people can have the exact same serving and pre-processing logic when serving the specific version of the model.

The second change is that after you have all the necessary code and dependencies and models versioned together, BentoML actually provides a centralized repository for managing all the models that's been produced by your machine learning team. So through this way you can have a much more sophisticated workflow that involves multiple teams in your organization. So on one hand you have a team of data scientists that's trending and producing new models and they set up training pipelines that are producing new models over time. And every one of those training pipelines will basically be producing a new Bento file and they can basically push that Bento file to this central repository where others in your organization can come in. Machine learning engineers can set up a CI/CD pipeline against this repository and launch a testing job whenever there's a new model created.

Your web developer or mobile developer can easily pull down the new model created by you data scientist. Launch an API server locally for debugging and testing their integration with your prediction service. Your product manager can come in, check out all the models being created and allow it to add comment and review in that process and approve the model before it gets deployed to production. And most importantly, DevOps can come in and actually customize how a model is being deployed to their own infrastructure and being able to actually customize and write code to change how the deployment workflow look like. And then data scientists will then be able to utilize those scripts created by DevOps and easily apply and update their machine learning models running in production.

[00:19:38] JM: Do you have an example of like a case study of some team that has used BentoML to great advantage?

[00:19:45] CY: Yes, definitely. So one of our users is Line. One of the most popular messaging app from Korea. The Line team has been building an internal machine learning platform for years, and they were previously the maintainer of a project called Clipper. That's one of the first projects that introduces micro-batching to model serving to. And BentoML, micro-batching is greatly inspired by Clipper. So when they look at BentoML, besides micro-batching, they really see the value of how BentoML can change their overall workflow. How BentoML can give their data scientist interface for them to patch the model, and then in a consistent way the model can be deployed to a number of downstream serving infrastructures. So they now actually have multiple data scientists teamed across multiple countries using BentoML for serving. And one of their largest serving use cases now serving over 200 million predictions per day.

We are also seeing contributions from the Line team, helping us to build some of the missing features they really want. And that's of course one of the largest deployment we're seeing. At the same time, we're also seeing a lot of smaller teams who's just started doing machine learning model serving and deployment. And from their perspective, besides the model packaging and model management, another feature that's really valuable for them is BentoML allows a really easy way to get your model to running in the cloud. So we provide something called deployment operator. That's basically a way for DevOps to customize how a model is being deployed to your own infrastructure. And we provide a few pre-built deployment operators for some of the cloud services, like AWS Lambda, AWS EC2, SageMaker or AzureML. So that data scientists really just need to run one simple command or just click one button and they will get API endpoint running in the cloud without any DevOps involved. So that really gives the power to data scientists in the smaller team where they lack the appropriate DevOps resource, but really wants to get their model to run in production and give access to their app developers.

[00:22:08] JM: Tell me more about the ongoing operational challenges that a machine learning engineer would have that are alleviated by BentoML.

[00:22:17] CY: So one of the ongoing challenge for BentoML adapters is how do they actually manage the deployment workflow when they deploy to a Kubernetes cluster. So one of the most common use case we are seeing is that a team will have a DevOps team maintaining their own Kubernetes cluster or OpenShift cluster where they will have all their other backend services

that needs access to this model deployed there. They have all the feature data that will need access by this prediction service available in that cluster. And as they deploy those models to the Kubernetes cluster, they will need ways to manage this workload to do a blue-green deployment, to run A-B testing and experiments and then being able to easily set up a monitoring dashboard for data scientists, being able to collect the prediction logs for their data scientists.

So we're seeing a lot of users rebuilding some of the integrations around that kubernetes deployment workflow. So that's what's coming next in BentoML. We think we are going to provide an opinion way of how a prediction service can be deployed to Kubernetes as well as a number of, I guess, nicer user experience features for data scientists. So they can have a much easier access to understanding how their models are performing in production and access to the prediction logs and reuse those logs in their development environment.

For example, they can access all the prediction logs. Run that in their analytic pipeline and understand how those predictions are affecting their business metrics. Or as they're producing, they're training a new machine learning model they can rerun the prediction logs with this new model and understand how their behavior compared against the previous model in production.

[00:24:12] JM: What are the frictions in model deployment and management that are still outstanding?

[00:24:18] CY: I think for model management, there are really two types of model management tools out there. One is for people to use in the model development phase. Say, your data scientists are launching 20 different experiments with different parameters and then trying to figure out which one has the best performance and select the model. So you need something to manage all those experimentations and all the models produced by those experimentations. And we are seeing a large number of tools that are doing that and call themselves model management. But on the other hand, after you've set up a training pipeline, now you're producing and training a new model daily with a new dataset. You need another two to manage models that's more designed towards deploying those models. Managing the models that are ready for deploy in production and serving production traffic. So we really don't see any two other than BentoML out there that's doing model management to design for deployment and

serving. So that's why we don't want to reinvent the build. So we build this model management layer around BentoML.

The challenges around deployment, I think it comes back to there are so many different types of ways for people to run the machine learning model. And for each type of model serving scenario, like online API serving, serverless serving, batch serving, streaming serving. For each type of serving use case you will also have a large number of different cloud platforms or open source platform you can choose. And similarly, for a team to build that type of integration themselves is really hard. But as an open source framework, and we design more generic APIs for people to adapt, we really enable the community to build integrations and deployment operators to support that wide range of different platforms.

[00:26:17] JM: So it sounds like the real struggle is in the API surface and just the fact that there's so many different integrations that need to be written.

[00:26:25] CY: Exactly. Our team actually spent a lot of time designing that API and iterated with end users over time. I think the project has been around a little bit over a year, but at least this first six months or eight months we've been making big changes around the API multiple times.

[00:26:47] JM: So you're not just an open source project. You're a company. What does your company do?

[00:26:51] CY: So we started the open source project as a company and we are planning to build more enterprise features around this model deployment and model management layer. So similar to many other open core companies, we think the success of the open source project is definitely tied to the success of the company. We want to have as much people adopting BentoML open source as possible. And we are basically giving out all the core features around model packaging and model serving, high-performance API serving and batch serving, etc., into this open source framework, the core of the library. But there are also features around the model management, deployment layer that's more linear towards enterprise-grade features such as single sign on, account management, access control, auditing logs. So those would be the additional features that we are selling to big enterprise customers.

From a developer's point of view, I think that make a lot of sense as an open source model just because most of the enterprise features are not exactly fun to work with. You probably don't get too much contribution from the open source community. And it takes a lot of time and effort to actually build and maintain that set of features and makes perfect sense for us to offer as an enterprise-grade feature. And as we grow, the business grow, we'll have more resources deployed to the open source side, which benefits the entire community.

[00:28:26] JM: What have been the struggles of starting a company in this space?

[00:28:29] CY: I think the first step struggle is we originally tried to build a proprietary version of the model serving and deployment tool that we are building today. And it's really hard to sell such a product in such a noisy space. We started a company about two years ago, and when we show a prototype of our model serving product to data scientists, people love it. But when we show that to their bosses, their VP of eng or head of machine learning, the first question they will ask is, "Why would I choose you over, say, AWS SageMaker or Google AI platform?" And it's a hard question to answer, because we know even if we are in their position, we will not trust a startup of just a few people over the cloud provider where they have much stronger engineering resources.
But I think that entire situation has changed ever since we pivoted to building open source. We have a strong community around the project as we grow. And now we are seeing enterprise customers, big companies, switching from Google Ai platform and AWS SageMaker to doing serving with BentoML, and we think that's just something we would never be able to achieve with a pure high-priority software.

I think another struggle is just how crowded and how much other products are in this space and being able to really articulate the benefit of BentoML solution in comparison with all the other different solutions and open source projects in this space. It's hard. It's very hard.

[00:30:14] JM: What's on your roadmap for BentoML?

[00:30:17] CY: So we actually share a more comprehensive list of the roadmap on our GitHub discussions. So that will have all the detailed timeline, including the features we are building. But at a high level, we think the first thing we want to do is make it even easier for data

scientists to build prediction services. So adding features around how to debug a prediction service as they're building it. Adding more documentation and examples around different ways and the common challenges people will run into when creating a prediction service.

One of our users says that BentoML feels like Ruby on Rails for machine learning, and that's a really inspiring quote that we think traditionally people think data scientists are not supposed to be building prediction services and deploying them to production. But in our opinion, there's not good tools for them to do that. And we think BentoML is in a really good position to make that possible for data scientists.

The second big item on our roadmap is deeper integration around the Kubernetes ecosystem. Most of our larger enterprise users are deploying BentoML to Kubernetes today. And as I mentioned earlier, there's a lack of deeper support around the user experience, around managing the deployment and experiment workflow on Kubernetes for data scientists.

[00:31:46] JM: Okay. Imagine it's five years into the future. How does the machine learning tooling space look different?

[00:31:52] CY: Great question. The way machine learning teams are managing the workflow for serving and deployment today really feels just like uploading PHP files through FTP to an Apache server to host your web application probably 10-15 years ago. And we think all that will be changed five years from now.

What we're seeing today is that DevOps and engineers are treating machine learning workloads so differently. As we talked about at the beginning of the show, people treat ML workflow completely differently than other microservices. And they are using patterns such as uploading the model directly to an API server as it's running in production or rebuilding the pre-processing code after a model is trained and have no way to version that code and tied it back to the right version of the model. And there are just way too many ways you can introduce issues and bugs into that workflow. It almost feels like what software engineer and deployment looks like before Git or before Docker, and everyone is trying to figure out the right way to deploy a software. And we think that will drastically change in five years from now for the machine learning world as more and more people started putting models into production. More and more people start

understanding why they need to do CI/CD? Need to run tests against your model before deploying? What are the best practices for getting model into production and monitoring those models? So that's really BentoML's vision. Giving people this essential building block to get to that ideal workflow.

[00:33:48] JM: Well, that sounds like a good place to close off. Chaoyu, thank you for coming on the show. It's been great talking.

[00:33:52] CY: Thank you, Jeff. Thanks for having me.

[END]