Getting Started with Remote ML Inference in Beam Java

Ganesh Sivakumar

Agenda

1. Why LLMs/ML in Apache Beam
2. Python RunInference
3. Introducing RemoteInference for Java
4. How to use it
5. How to write your own model handler
6. References

Why LLMs/ML in Apache Beam?

- **LLM-as-a-transform:** treat a model like any other PTransform in your pipeline.
- Enrich, classify, summarize, reason on your data within the pipeline.
- Beam foundation

Python Run Inference transform

```
from apache_beam.ml.inference.base import RunInference
with pipeline as p:
   predictions = ( p |  'Read' >> beam.ReadFromSource('a_source')
                     | 'RunInference' >> RunInference(<model_handler>)
```

But ML inference in Java pipelines ?

Introducing RemoteInference transform for Beam Java SDK

A PTransform for making remote inference calls to external machine learning services.

RemoteInference provides a framework for integrating remote ML model inference into Apache Beam pipelines and handles the communication between pipelines and external inference APIs.

Usage

1. Add the Remote Inference dependencies from maven(artfact: beam-sdks-java-ml-inference-remote)

2. Choose your model provider and add their dependency (Eg: OpenAI provider: beam-sdks-java-ml-inference-openai)

3. Understand the inputs and output types of the model defined by the module (Eg: OpenAIModelInput, OpenAIModelResponse) these classes tell the transform, what’s the input to the model and what does model respond in (Eg: String / JSON /numbers)

4. Define your required model parameters. Like, API key, model name and instruction Prompt

5. Run Inference 🚀

```
RemoteInference.<ModelInputType, ModelOutputType>invoke()
               .handler(handler.class)
               .withParameters(params)
```

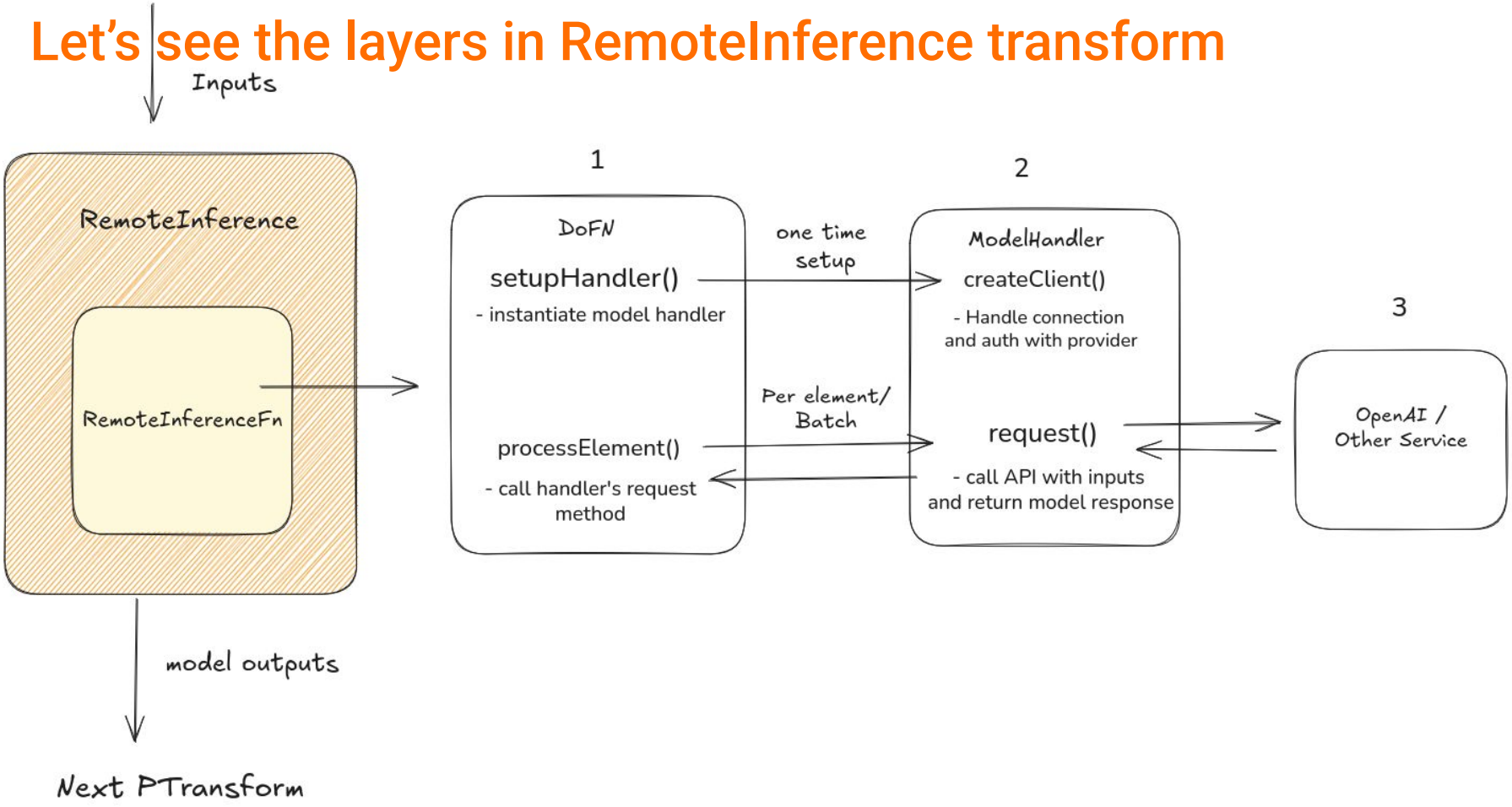
Example:

```
// Create model parameters
OpenAIModelParameters params = OpenAIModelParameters.builder()
    .apiKey("your-api-key")
    .modelName("gpt-4")
    .instructionPrompt("Analyse sentiment as positive or negative")
    .build();

// Apply remote inference transform
PCollection<OpenAIModelInput> inputs = pipeline.apply(Create.of(
    OpenAIModelInput.create("An excellent B2B SaaS solution that streamlines bu
siness processes efficiently."),
    OpenAIModelInput.create("Really impressed with the innovative features!")
));

PCollection<Iterable<PredictionResult<OpenAIModelInput, OpenAIModelResponse>>>
results =
    inputs.apply(
        RemoteInference.<OpenAIModelInput, OpenAIModelResponse>invoke()
            .handler(OpenAIModelHandler.class)
            .withParameters(params)
    );
```

How does RemoteInference transform work?

Typical PTransform is like an Onion – Multiple Layers

Let's see the layers in RemoteInference transform



How to integrate your custom model services

Java remote module provides the foundation for LLM inference in Beam Java pipelines so that you can bring in your own model service and Run Inference on top of base abstractions. It mostly comes down to defining the model inputs outputs and tell the RemoteInference Transfrom how it should communicate with your model service.

1. Define your model service's configuration

```
import org.apache.beam.sdk.ml.inference.remote.BaseInput;
import org.apache.beam.sdk.ml.inference.remote.BaseResponse;
import org.apache.beam.sdk.ml.inference.remote.BaseModelParameters;

public class YourModelInput implements BaseInput {
      // input parameters
}

public class YourModelResponse implements BaseResponse {
      // model response parameters
}

public class YourModelParameters implements BaseModelParameters {
   // model parameters like API key, model name, prompt or other parameters
}
```

2. Write your model handler

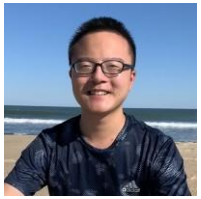
- Implement the BaseModelHandler contract/interface
- Collect required parameters and initialize the model client in createClient()
- Set the client object transient
- request() method receives a batch of model inputs and it needs to be processed by the model client and return and response as PredictionResult pairs.

Java

```java
import org.apache.beam.sdk.ml.inference.remote.BaseModelHandler;
import org.apache.beam.sdk.ml.inference.remote.PredictionResult;


public class YourModelHandler
    implements BaseModelHandler<YourParameters, YourModelInput, YourModelResponse> {

  private transient ModelClient client;
  private ModelParameters parameters;


/** * Initializes the Model client with the provided parameters, authenticated client using
the API key from the parameters. */
  @Override
  public void createClient(OpenAIModelParameters parameters) {

    this.modelParameters = parameters;
    this.client = initializeClient()
  }

/** * @param input the list of inputs to process * @return an iterable of model results and
input pairs */
  @Override
  public Iterable<PredictionResult<OpenAIModelInput, OpenAIModelResponse>> request(
      List<OpenAIModelInput> input) {

    return client.responses()
  }
```

Reference Links

1. Java remote inference module - https://github.com/apache/beam/tree/master/sdks/java/ml/inference/remote
2. OpenAI modelhandler - https://github.com/apache/beam/tree/master/sdks/java/ml/inference/openai



Thanks to Jack McCluskey (jrmccluskey) for iterating on the transform design and reviewing my PR and Yi Hu(Abacn) for reviewing Java module part of PR.

Thank you!

Feel free to reach out :)

Ganesh Sivakumar

Email: ganesh.gsivakumar@gmail.com

LinkedIn: www.linkedin.com/in/ganesh-sivakumar

Github: https://github.com/Ganeshsivakumar