

# Troubleshooting Beam/Dataflow ML pipelines related Common Issues

Rajkumar Gupta



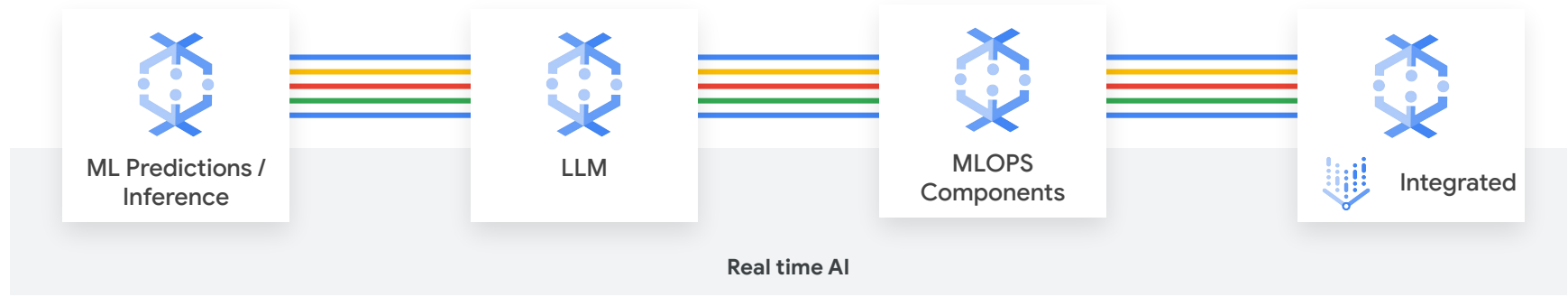
BEAM  
SUMMIT

September 4-5, 2024  
Sunnyvale, CA. USA

# Agenda

- **Introduction to Beam/Dataflow ML**
- **Core RunInference related issues**
  - Utilization: Why do I keep getting small batch sizes?
  - Utilization: Why am I not getting sufficient utilization out of my GPU?
  - Hangs/Exceptions: Why do I keep getting OOM exceptions?
  - Hangs/Exceptions: Why is an element stuck in my pipeline?
- **Non RunInference related issues**
  - Dependency related errors
  - Container Image related errors
  - Other issues


# Beam/Dataflow ML - Serving - where most of the workloads/issues come from today






### ML predictions

- Local models
- Remote models
- GPU Choice
- GPU Efficiency
- Model Ensembles




### LLM

- MLTransform Embedding generation
- Transport to VectorDB
- Integrate with OSS LLM




### ML Frameworks & Platforms

- Vertex AI
- Huggingface
- Tensorflow
- PyTorch
- scikit-learn, XGBoost
- others...



### Continuous data preparation

- Vertex AI Pipelines
- Kubeflow Pipelines
- TFX



### Insights activation

Downstream integrations with 30+ Sources / Sinks:

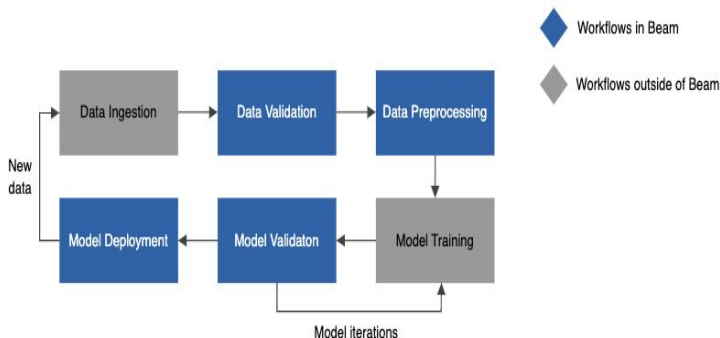
- BigQuery
- BigTable
- Pub/Sub
- Apigee & others

# RunInference Overview

- Serving (or model inference) - taking a pretrained model and using it to repeatedly produce inferences on incoming data
  - Example - Spotify's [podcast summarization](#)
- RunInference - Beam/Dataflow's mechanism for serving models
  - Recommended way for serving models

## AI/ML workloads

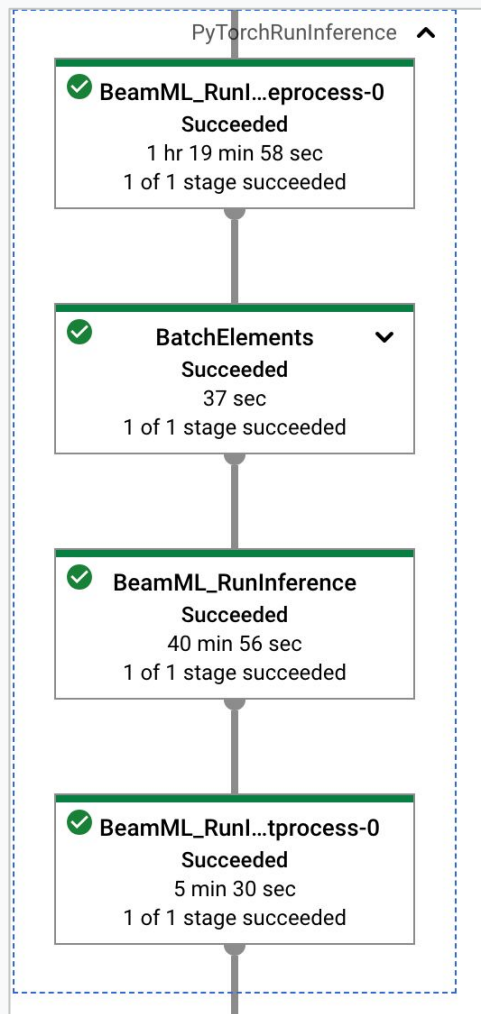
You can use Apache Beam for data validation, data preprocessing, model validation, and model deployment and inference.



```
1 my_model_handler = pytorch_inference.PytorchModelHandlerTensor(  
2     state_dict_path='gs://bigdatapivot/models/model_file.pth',  
3     model_class=model_class,  
4     model_params={'pretrained':False, 'pretrained_backbone' : False})  
5  
6 my_keyed_model_handler = beam.ml.inference.base.KeyedModelHandler(my_model_handler)  
7  
8 with beam.Pipeline() as p:  
9     (p  
10      | beam.io.fileio.MatchFiles('gs://bigdatapivot/images/bicycle.jpg')  
11      | beam.io.fileio.ReadMatches()  
12      | beam.Map(preprocess_image)  
13      | beam.ml.inference.RunInference(my_keyed_model_handler)  
14      | beam.Map(print)  
15     )
```

# RunInference Lifecycle

- `load_model` - at setup time
- `BatchElements` - batches data for more efficient inference
- `run_inference` - runs against batched data with loaded model
- `with_preprocess_fn` and `with_postprocess_fn` - optional pre/postprocessing functions (user Map functions)



## Utilization: Why do I keep getting small batch sizes?

- Screenshot shows example Job with healthy batching
  - Unhealthy, all batching counters will be at or close to 1, even if `min_batch_size` is set
- Almost always same cause - in general streaming pipeline uses non-stateful (default) batching which batches at the bundle level
  - Streaming == small bundles
- Recommendation: Switch to stateful batching implementation using `max_batch_duration_secs`

The screenshot displays a job execution interface for a job named 'beam-ml-start...'. The interface is divided into two main sections: 'JOB GRAPH' and 'JOB INFO'.

**JOB GRAPH:** Shows a vertical sequence of five stages, all of which are 'Succeeded':

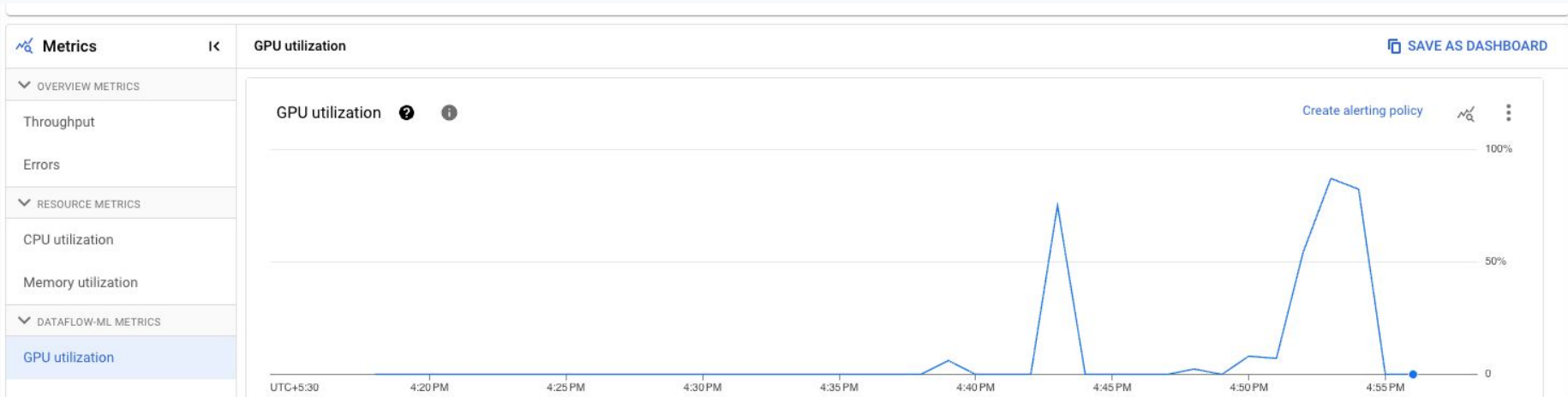
- ReadImageNames:** Succeeded, 22 sec, 2 of 2 stages succeeded.
- FilterEmptyLines:** Succeeded, 0 sec, 1 of 1 stage succeeded.
- ReadImageData:** Succeeded, 2 hr 11 min 59 sec, 1 of 1 stage succeeded.
- PreprocessImages:** Succeeded, 11 min 22 sec, 1 of 1 stage succeeded.
- RunInference:** Succeeded, 59 min, 1 of 1 stage succeeded.

**JOB INFO:** A table showing various counters and their values. The 'batch\_size' counters are highlighted in blue:

Counter name	Value
batch_size_COUNT	3,735
batch_size_MAX	100
batch_size_MEAN	13
batch_size_MIN	1
msec_per_batch_COUNT	3,735
msec_per_batch_MAX	17,983
msec_per_batch_MEAN	945
msec_per_batch_MIN	28
inference_batch_latency_micro_secs_COUNT	3,735
inference_batch_latency_micro_secs_MAX	17,974,418
inference_batch_latency_micro_secs_MEAN	941,173
inference_batch_latency_micro_secs_MIN	28,399
inference_request_batch_byte_size_COUNT	3,735
inference_request_batch_byte_size_MAX	8,815
inference_request_batch_byte_size_MEAN	1,193
inference_request_batch_byte_size_MIN	102
inference_request_batch_size_COUNT	3,735
inference_request_batch_size_MAX	100
inference_request_batch_size_MEAN	13
inference_request_batch_size_MIN	1
load_model_latency_milli_secs_COUNT	44
load_model_latency_milli_secs_MAX	2,380
load_model_latency_milli_secs_MEAN	1,981
load_model_latency_milli_secs_MIN	757

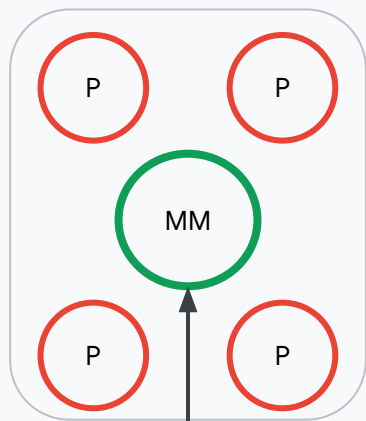
# Utilization: Why am I not getting sufficient utilization out of my GPU?

- Can either be based on Dataflow metrics or expected throughput numbers
- 2 main causes:
  - Source isn't pumping in enough data
  - Not enough models to feed GPU



# Memory Sharing in RunInference

- Option 1: Model per worker process, shared among threads (**default**)
  - Pros: Usually best performance (especially if CPU only)
  - Cons: OOMs
  - Recommendations to increase performance: Try [NVIDIA MPS](#) GPUs
- Option 2: Single shared copy (**large\_model=True OR share\_model\_across\_processes() returns True**)
  - Pros: Fewer OOMs
  - Cons: Single threaded for inference stage
  - Recommendation: Try loading more than one models (option 3)
- Option 3: N models loaded (**model\_copies=N OR model\_copies() returns True**)
  - Pros: Good in between option
  - Cons: Added in Beam SDK v2.56.0
  - Recommendation: Try tuning/increasing the number of models loaded



Python process



## Hangs/Exceptions - Why do I keep getting OOM exceptions?

- Memory pressure is the most common issue when serving models, especially LLMs
- Even small LLMs consume O(10s) of GBs of memory
- Increased by default setting (loading model per process)
- Options:
  - First: load fewer copies of the model (starting with 1)
  - Next: increase size of machine or accelerator
  - Last: Try decreasing batch sizes

Workload	A100	L4	T4
Model fine tuning	Recommended		
Large model inference	Recommended	Recommended	
Medium model inference		Recommended	Recommended
Small model inference		Recommended	Recommended

# Hangs/Exceptions - Why is an element stuck in my pipeline?

- Symptoms
  - High backlog, eventually low throughput
  - Long running user operation log under warnings
- If no long running operation or OOM, likely not an inference issue (look at the source I/O)
- If it is coming from RunInference, typically is user code
  - Recommend adding some thread safety, in user-code (ideal since it maintains parallelism) or by using `large_model=True`

```

File "/usr/local/lib/python3.10/dist-packages/apache_beam/runners/worker/sdk_worker.py", line 663, in
process_bundle
    bundle_processor.process_bundle(instruction_id))

File "/usr/local/lib/python3.10/dist-packages/apache_beam/runners/worker/bundle_processor.py", line 1056, in
process_bundle
    input_op_by_transform_id[element.transform_id].process_encoded(

File "/usr/local/lib/python3.10/dist-packages/apache_beam/runners/worker/bundle_processor.py", line 237, in
process_encoded
    self.output(decoded_value)

File "/usr/local/lib/python3.10/dist-packages/apache_beam/ml/inference/base.py", line 1423, in process
    return self._run_inference(batch, inference_args)

File "/usr/local/lib/python3.10/dist-packages/apache_beam/ml/inference/base.py", line 1391, in
_run_inference
    result_generator = self._model_handler.run_inference(
File "/app/src/text_embeddings_v2/beam/transforms/inference.py", line 123, in run_inference
    all_predictions = model.predict(batch)

File "/app/src/text_embeddings_v2/inference.py", line 114, in predict
    embeddings = self.generate_embeddings(self._processor(texts))

File "/app/src/text_embeddings_v2/inference.py", line 104, in generate_embeddings
    embeddings = self._model(**inputs)

```

## Dependency or Container Image related errors

### Common Issues:

- Job is not starting
- Worker is taking long time to startup

### Troubleshooting Tips & Best Practices:

- Use latest versions and also pin them so that pip doesn't have to guess
- Always use latest version of Beam SDK
- Review dependencies closely
- Add more logging at each step
- Use Prebuilding or Custom Containers to avoid installation related slowdown
- Check if Image is corrupt or not properly built
- Don't use 'latest' tag in container image path, instead use specific tags

## I'm stuck, what else can I try?

- Limit the number of variables
  - Use reshuffles before/after inference step
  - Use larger machines and share a single model
  - Upgrade to most recent Beam and make sure dependencies are up to date (especially ML dependencies)
- Consult other resources
  - [Dataflow ML Docs](#)
  - Reach out to Beam Community or Dataflow Support

# Thank you!

Questions?

[rajcumargupta@google.com](mailto:rajcumargupta@google.com)



**BEAM**  
SUMMIT