Transitioning Uber Michelangelo's Batch Prediction from Apache Spark to Ray

Baojun Liu

BEAM SUMMIT

September 4-5, 2024

Sunnyvale, CA. USA

Intro

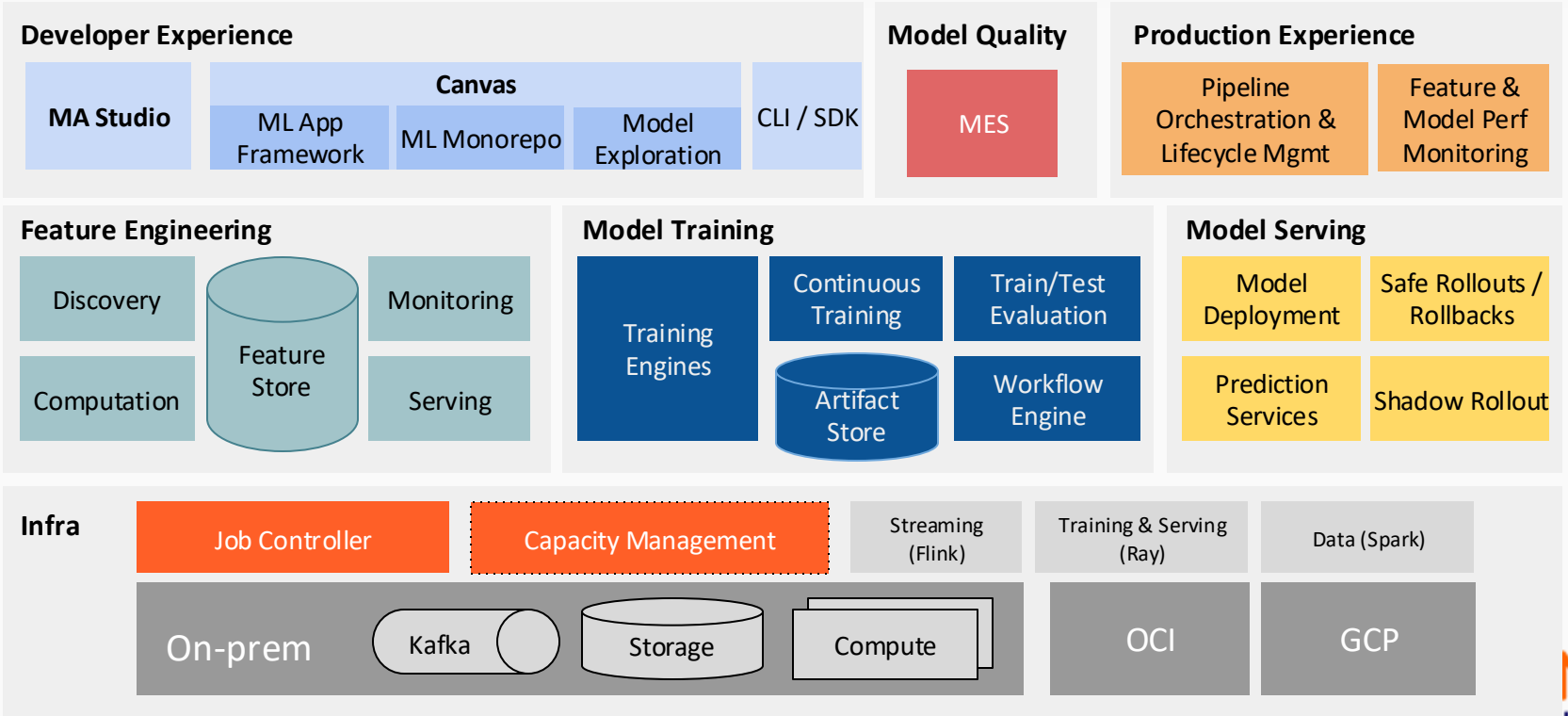
- Michelangelo @Uber
- Spark for Data Processing
- Ray for ML
- LLM Batch Prediction

BEAM SUMMIT

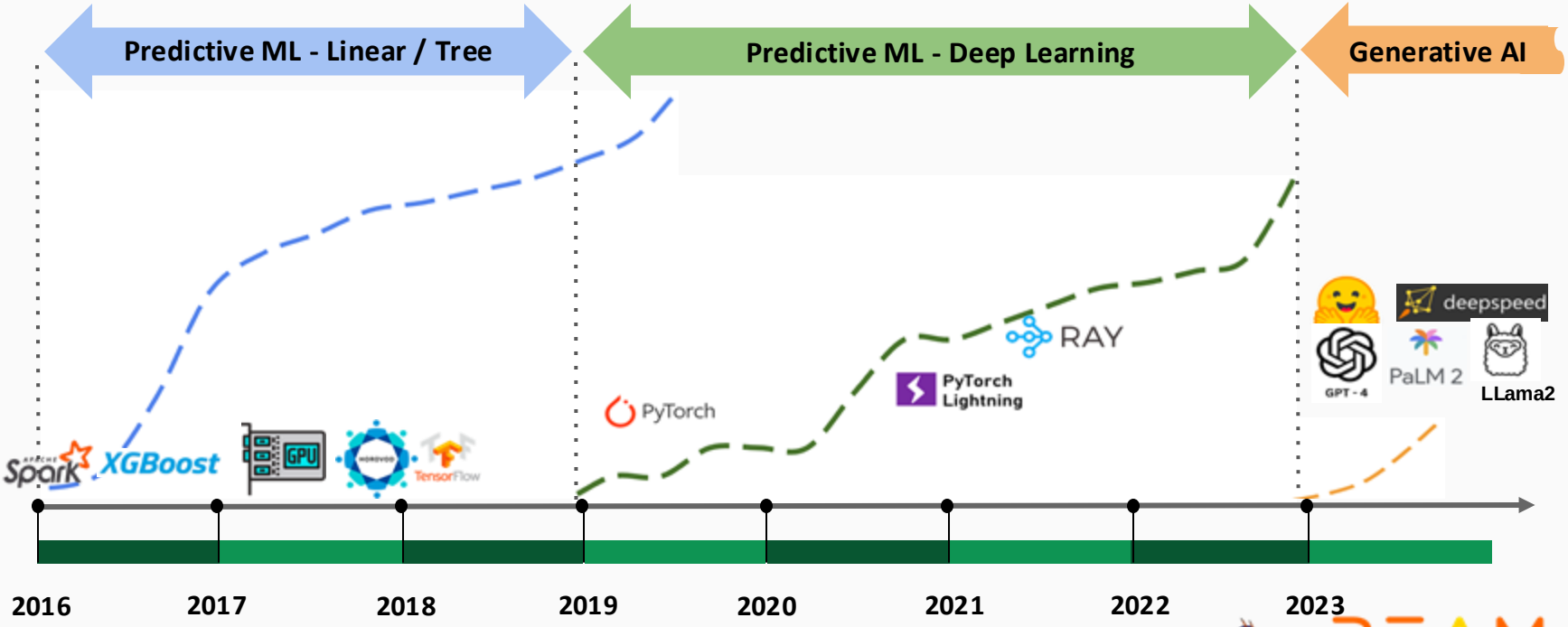
Michelangelo :
Uber’s Unified ML Platform

BEAM SUMMIT

Michelangelo Overview



ML/AI Evolution



AI/ML at Uber's Scale

20K

# of model trained / month

5.3K

Models in Production

16M

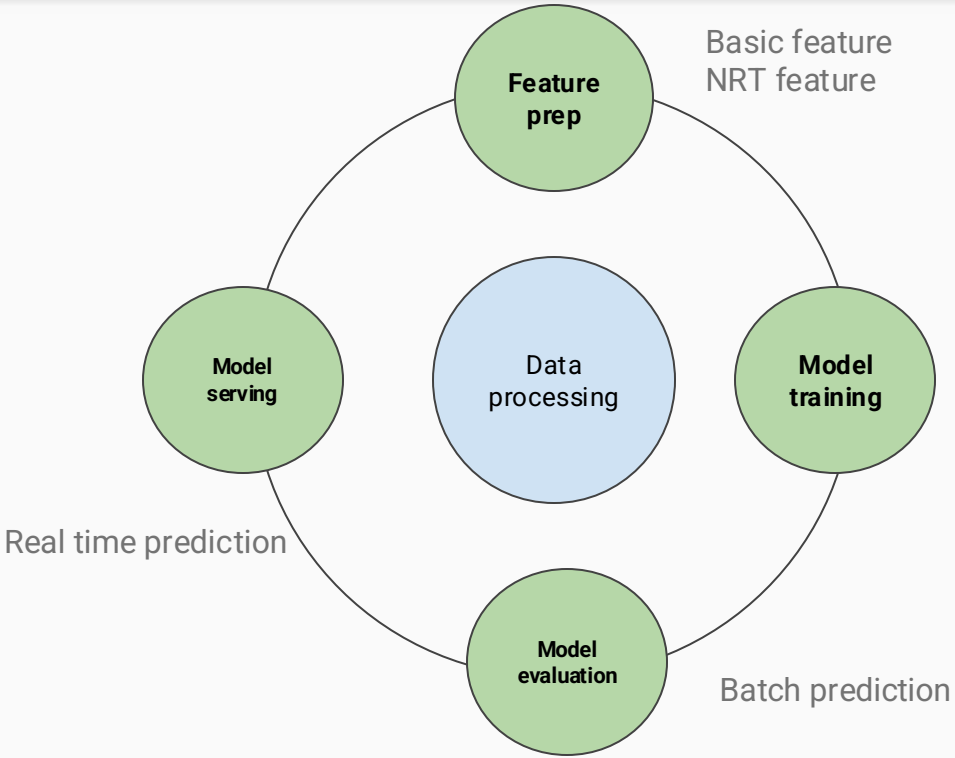
Peak Predictions Per Second

BEAM SUMMIT

Data Processing with Spark

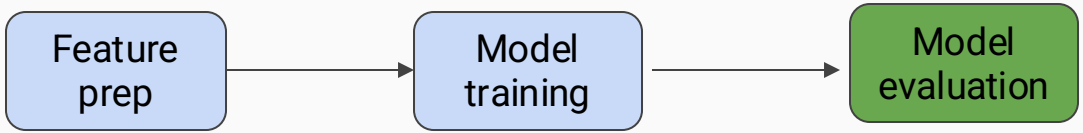
BEAM SUMMIT

Data Processing Cycle in ML

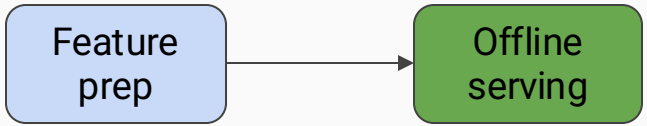


Batch Prediction

Model evaluation



Offline serving



1K

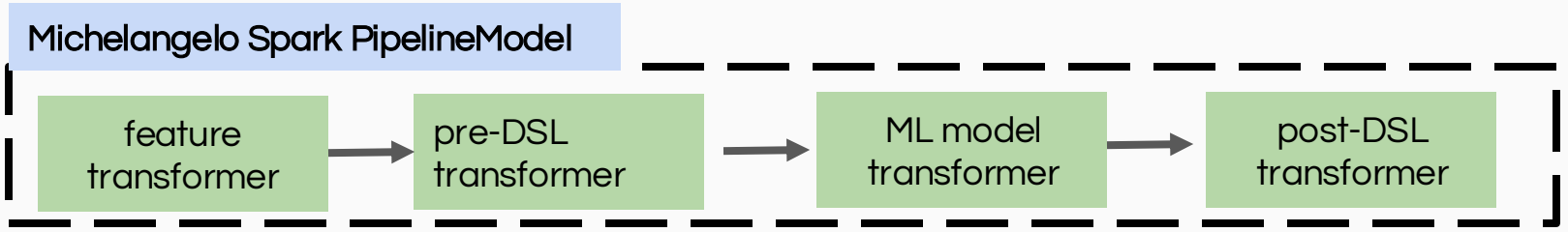
# of pipeline daily

Spark for data processing

- Scalability
  - Large scale data processing, distrutisted
- Speed
  - In memory speed
- Ease to use
  - High level APIs in Java, Scala, Python
- Unified Analytics
  - Batch processing, real time streaming

BEAM SUMMIT

Spark Pipeline Model




```
model.transform(input_data)

UDF for ML model prediction

DSL = domain specific language
```

Spark for ML

- Tabular data
- ~ 1M -10M size
- CPU

- Unstructured / numerical data
- ~ 100M -1G size
- GPU

Lack of GPU support and flexibility for ML

Ray for ML

Ray Intro

- Open source framework
  - parallel and distributed systems
- Flexible API, scalable, low latency
- Tailored for ML
  - Ray Tune, Ray RLib, Ray Serve
  - Hyperparameter tuning, training

BEAM SUMMIT

Spark vs Ray

	Spark	Ray
Design propose	Large scale data processing, big data	Distributed computing with flexibility and scalability
Programming model	High level declarative, defined transformation	Flexible programming, imperative, user defined function
ML/AI	Traditional batch ML, classification, regression	Fine grained control, hyper-parameter tuning, training
Real-time/streaming	Large amount/high throughput	Low latency, quick scale individual task

BEAM SUMMIT

Spark vs Ray (cont'd)

	Spark	Ray
Ecosystem	Mature, integrated with Hadoop, Hive, Hbase	New and growing, Python based ML Frameworks, PyTorch, TF, SKLearn
Easy to use	High level APIs, Java, Scala, R Limited flexibility	Flexible, low level API, deep understanding from user

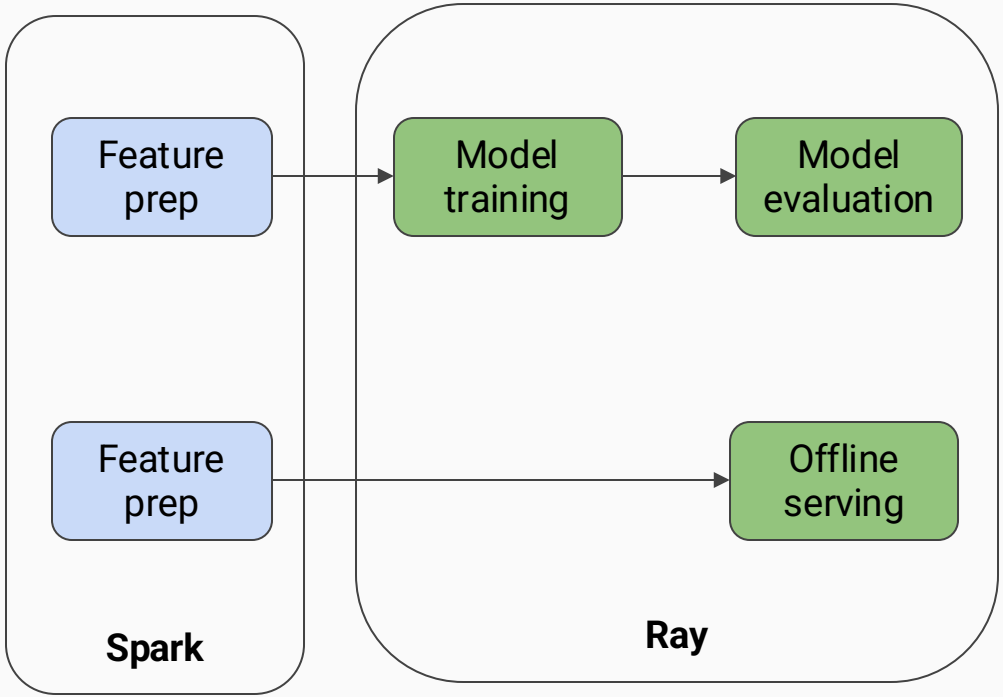
BEAM SUMMIT

Why Ray for ML

- ML focus
- Support modern ML case
- Fine grained control
- Scalability and efficiency
  - Efficient resource management
  - Great GPU support
  - Ray on Kubernetes

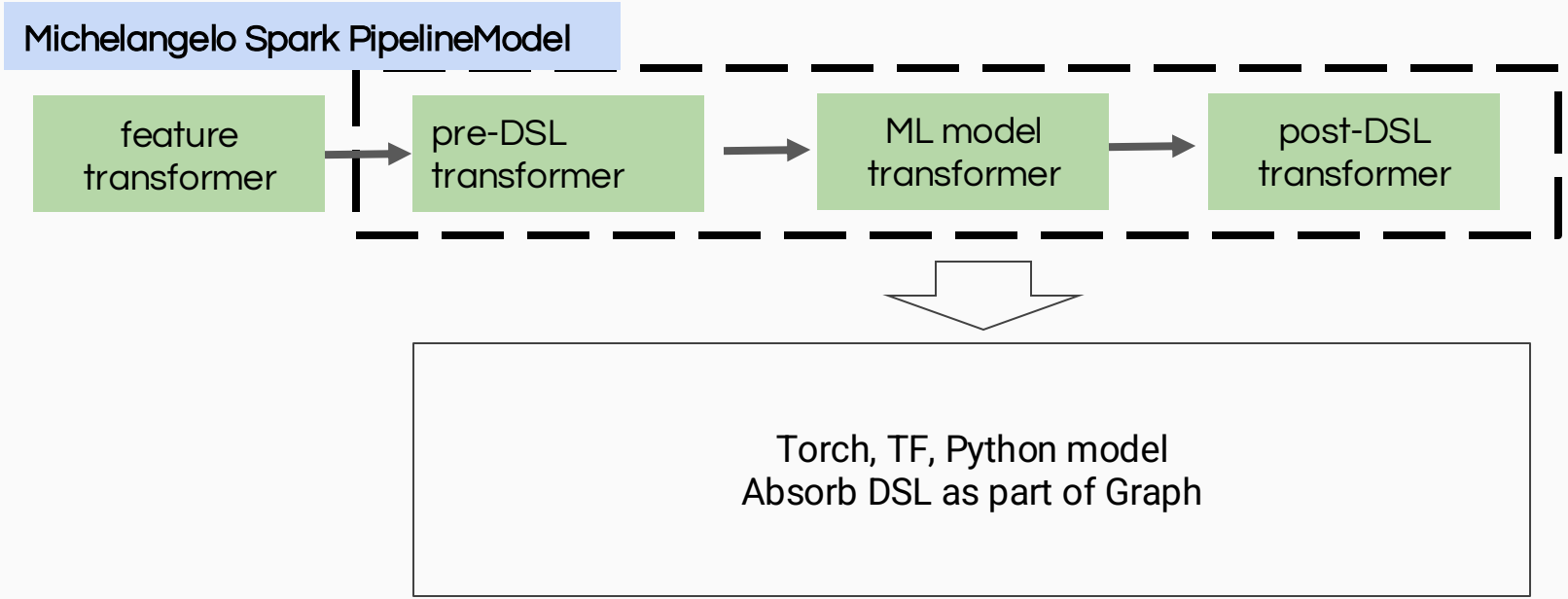
BEAM SUMMIT

Ray for batch Prediction

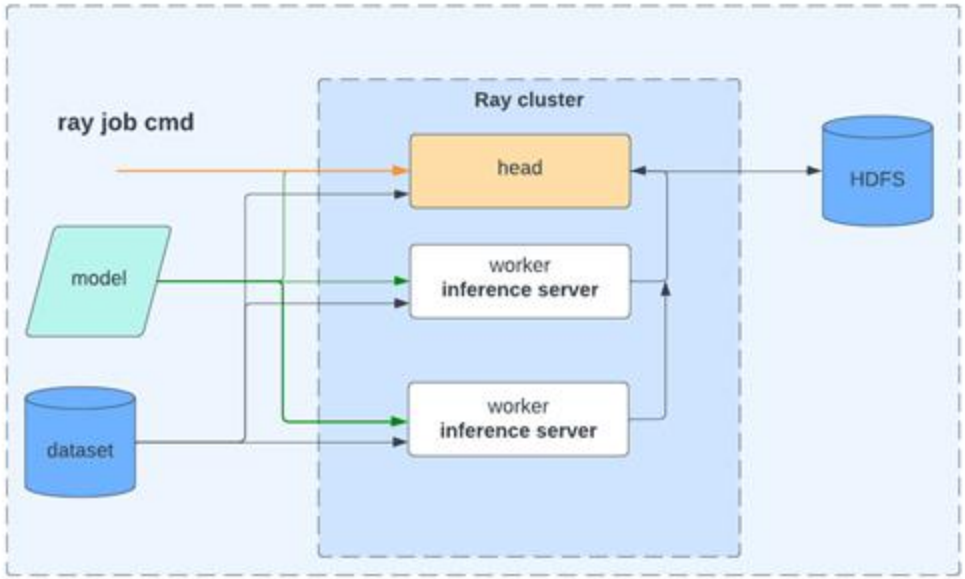


- ❑ Handle DSL transformers
- ❑ Online / offline consistency

Model modification



Ray for Batch Prediction



- Inference server
  - Same inference server as online for consistency
- GPU for prediction
  - Hundreds of GPUs
  - 100M rows

LLM batch prediction

LLM

- LLama 8b, 70b, 405b, Mixtral 8x7b
- Fine tuning for custom tasks
- LLM batch prediction
  - Open source / fine tuned model evaluation
  - Large scale batch prediction jobs

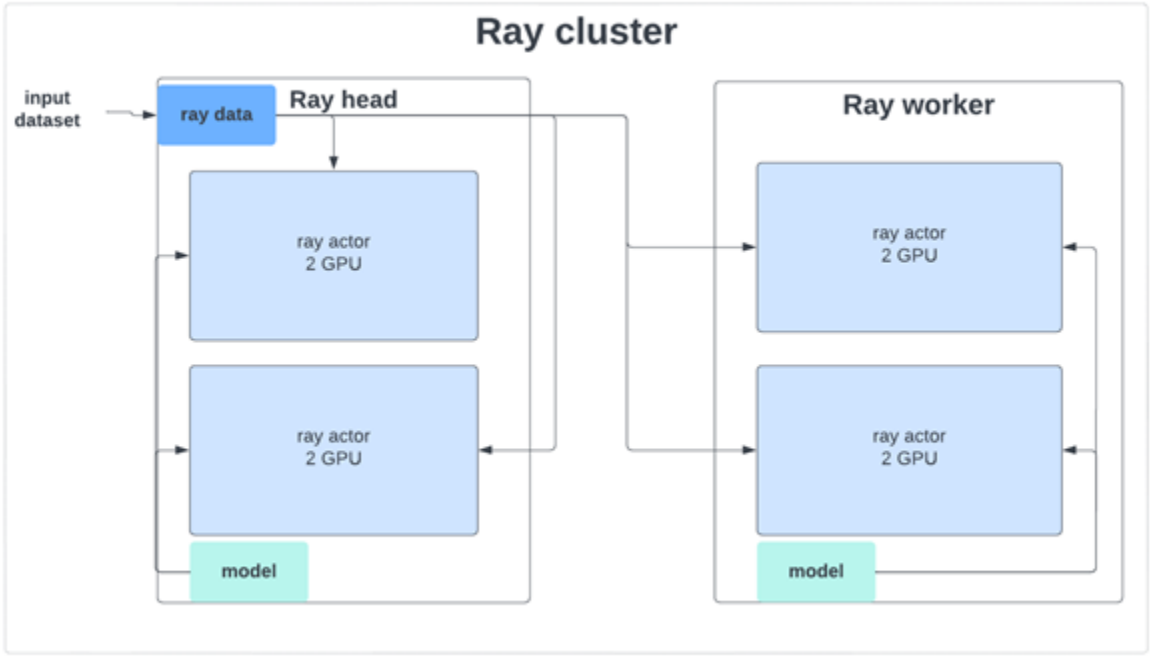
BEAM SUMMIT

LLM prediction is costly

- Large model size
  - 70b -> 140GB, 405b -> 810GB
  - Resource intensive, network bandwidth for downloading
- High latency
  - ~second vs ~ms (ML model)
  - High gpu hours for large jobs
    - 1M rows -> 500 GPU hrs
- Distributed with GPU
  - High end GPUs, H100
  - Multiple level GPU orchestration
    - Parallel node level
    - Tensor parallel in node

BEAM SUMMIT

Ray for LLM Batch Prediction



- Kubernetes for clusters
- Ray data for streaming execution
- Inference server
  - vLLM
  - Triton Server

Ray + vLLM

vLLM - fast and easy to use LLM serving


```
ray_dataset.map_batches(
  LLMPredictor,
  fn_constructor_kwargs={
    "model_path": model_path,
    "tensor_parallel_size": tensor_parallel_size,
    "temperature": temperature,
    "top_p": top_p,
    "max_tokens": max_tokens,
  },
  concurrency=concurrency,
  batch_size=batch_size,
  **resources_kwarg,
)
```

```
model = LLM(
    model=model_path,
    tensor_parallel_size=tensor_parallel_size,
)

output = model.generate(input)
```

BEAM SUMMIT

Summary

- Spark for feature prep
- Ray for ML
- Open source solutions make things easier

BEAM SUMMIT

Thank you!

Questions?

Software engineer @Uber
Michelangelo

Online/offline serving, AI infra, DL framework, distributed system

baojun@uber.com

https://www.linkedin.com/in/baojunliu/