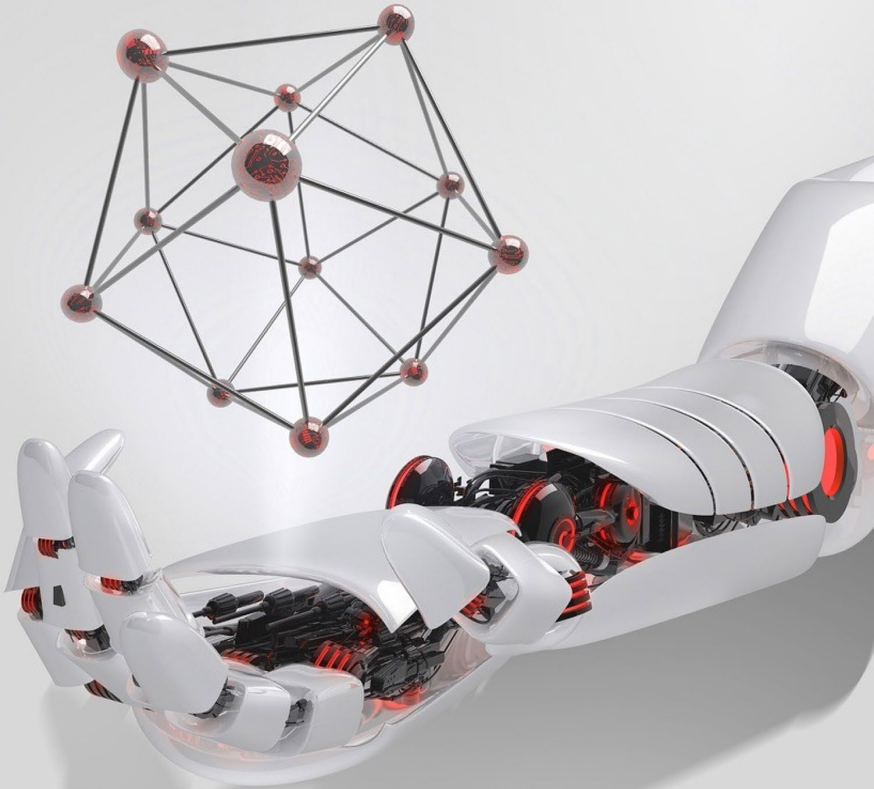
Ray @ eBay: Pioneering the Next Gen AI Platform

演讲人：Yucai Yu

eBay AI平台架构师



目录

eBay AI Strategy

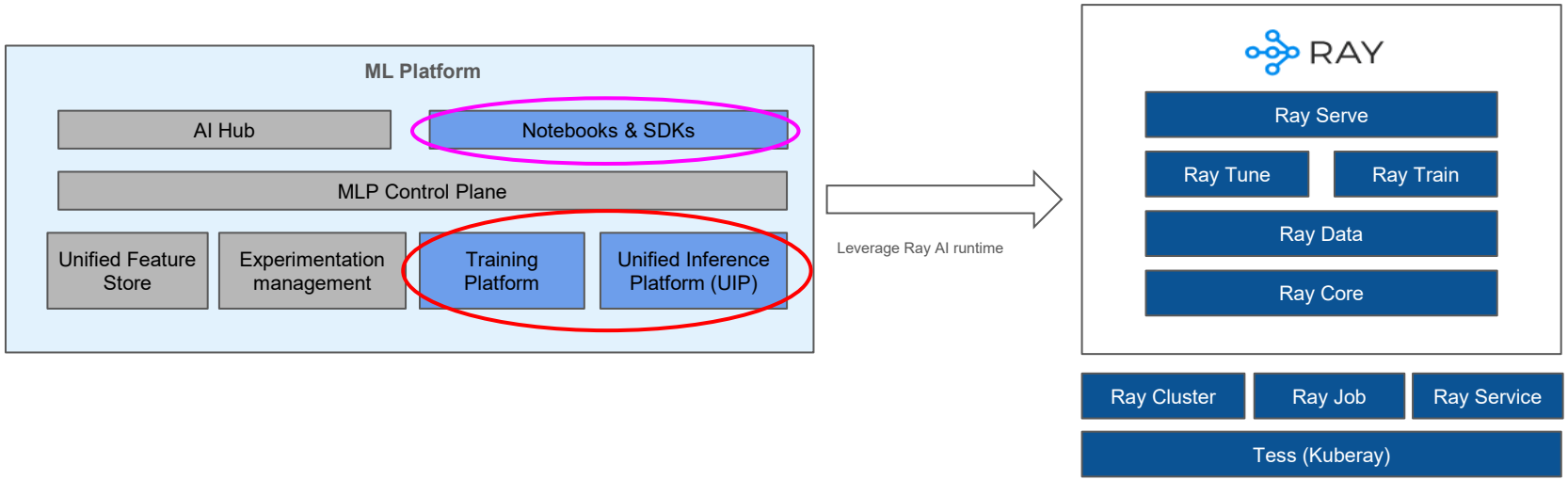
https://x.com/SquawkStreet/status/1781008021230846219

We believe eBay is best positioned to capture upside from gen AI in '24, to the extent its seller-focused features drive listing velocity and quality.

- Morgan Stanley Analyst (2024/04/18)

eBay's AIP 2.0 Upgrade

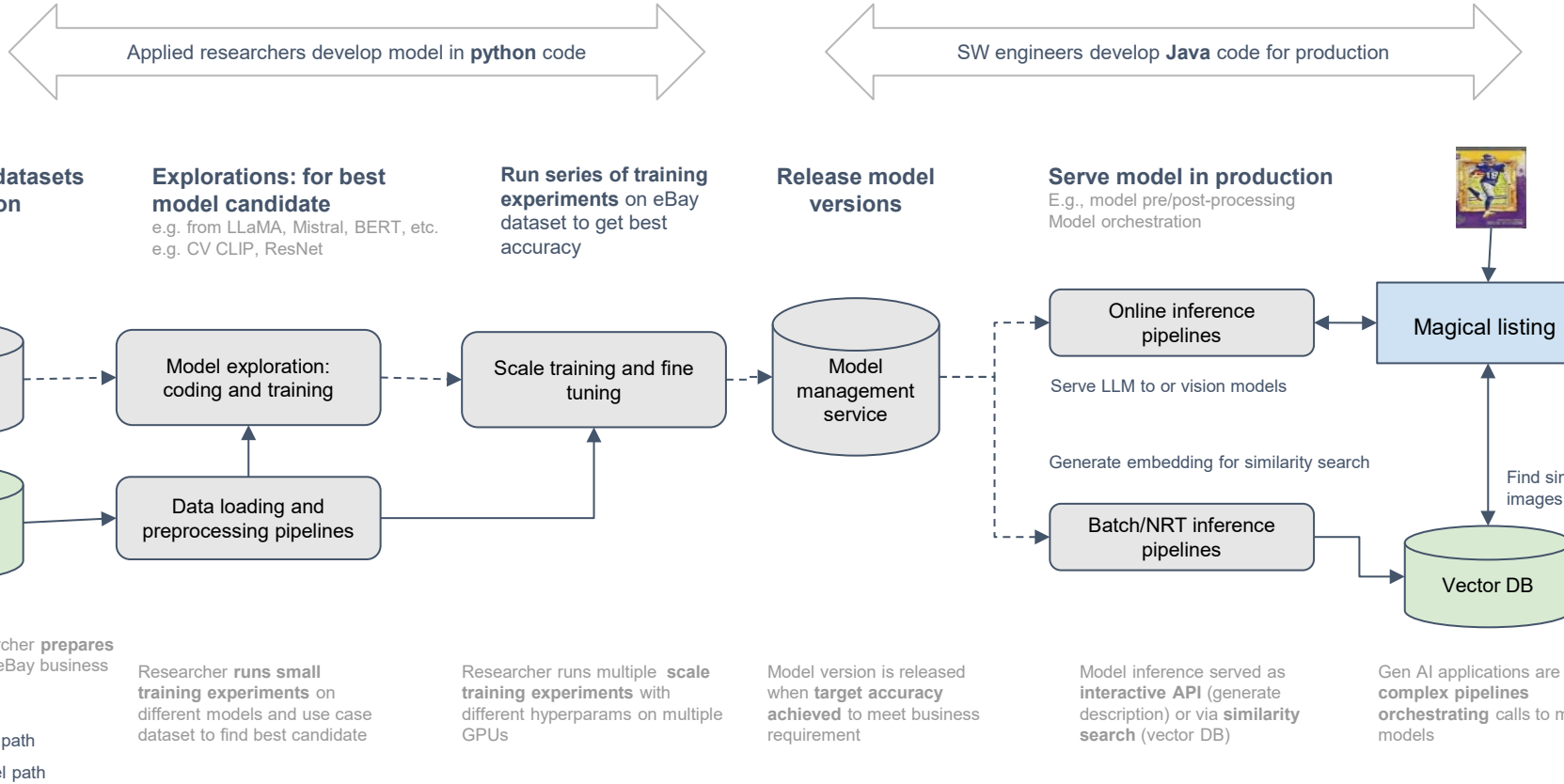
Generative AI revolution caused a step jump in large and complex models, increased GPU requirements. New use cases are rapidly increasing across eBay. Our infrastructure must quickly respond.



目录

- Background: eBay AI 2.0 Initiative
- Problems in Model Dev & Deployment
- eBay AIP 2.0 Powered by Ray
- Future Plans

Model Development & Deployment



Problems in Model Dev & Deploy

1 **~50% solution development time wasted** in back/forth porting python to Java

Applied researchers develop model in **python** code

SW engineers develop **Java** code for production

Training datasets preparation

Explorations: for best model candidate
e.g. from LLaMA, Mistral, BERT, etc.
e.g. CV CLIP, ResNet

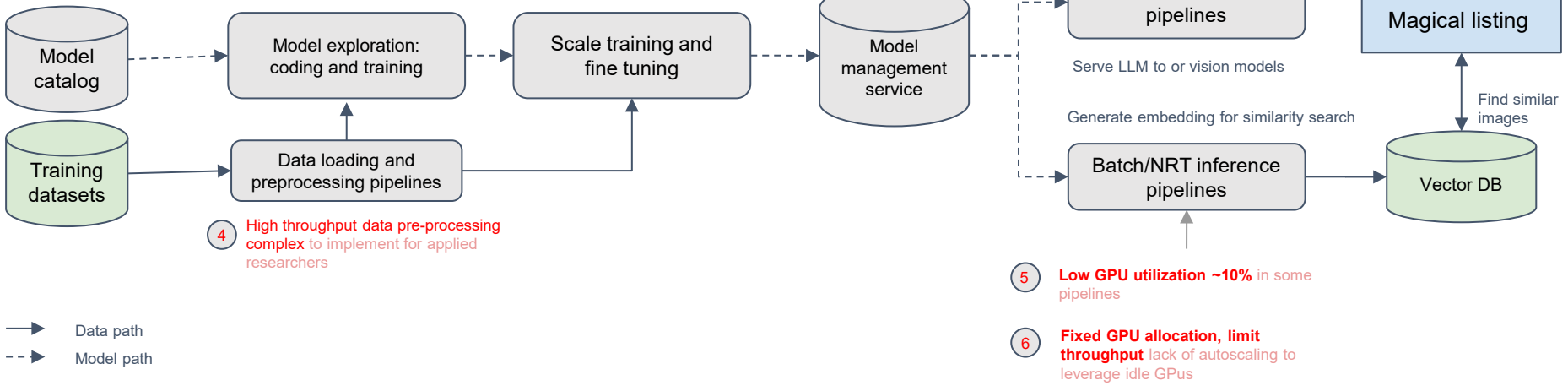
Run series of training experiments on eBay dataset to get best accuracy

Release model versions

Serve model in production
E.g., model pre/post-processing
Model orchestration

2 **Notebook dev env has very limit GPUs not sufficient for large model** development and debug

3 **A few weeks to rewrite pre/post processing and orchestration logic**



4 High throughput data pre-processing **complex** to implement for applied researchers

5 **Low GPU utilization ~10%** in some pipelines

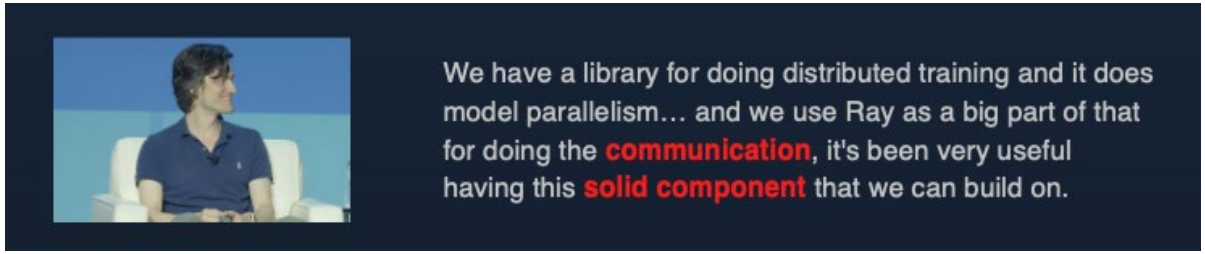
6 **Fixed GPU allocation, limit throughput** lack of autoscaling to leverage idle GPus

Data path
Model path

eBay AIP 2.0 Powered By Ray

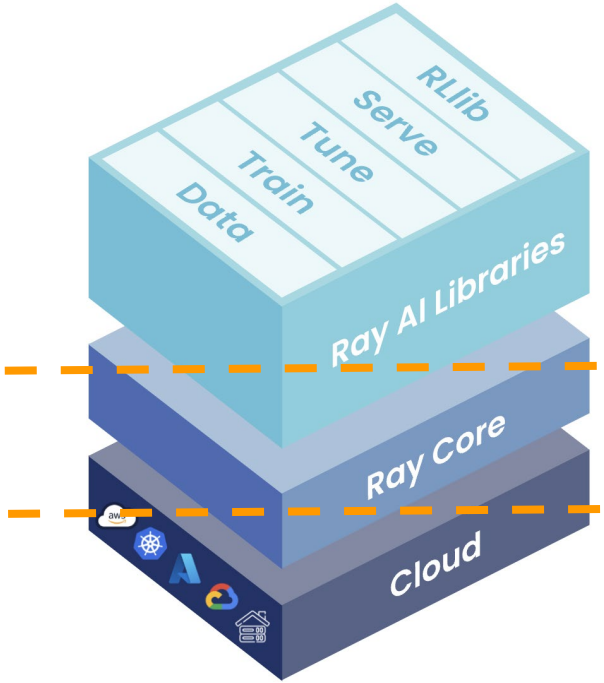
- Ray Introduction
- High Level Architecture
- Production Scenarios
  - Notebook for Research & Experiment
  - Batch Inference
  - Near Real Time Inference

Ray.io Introduction



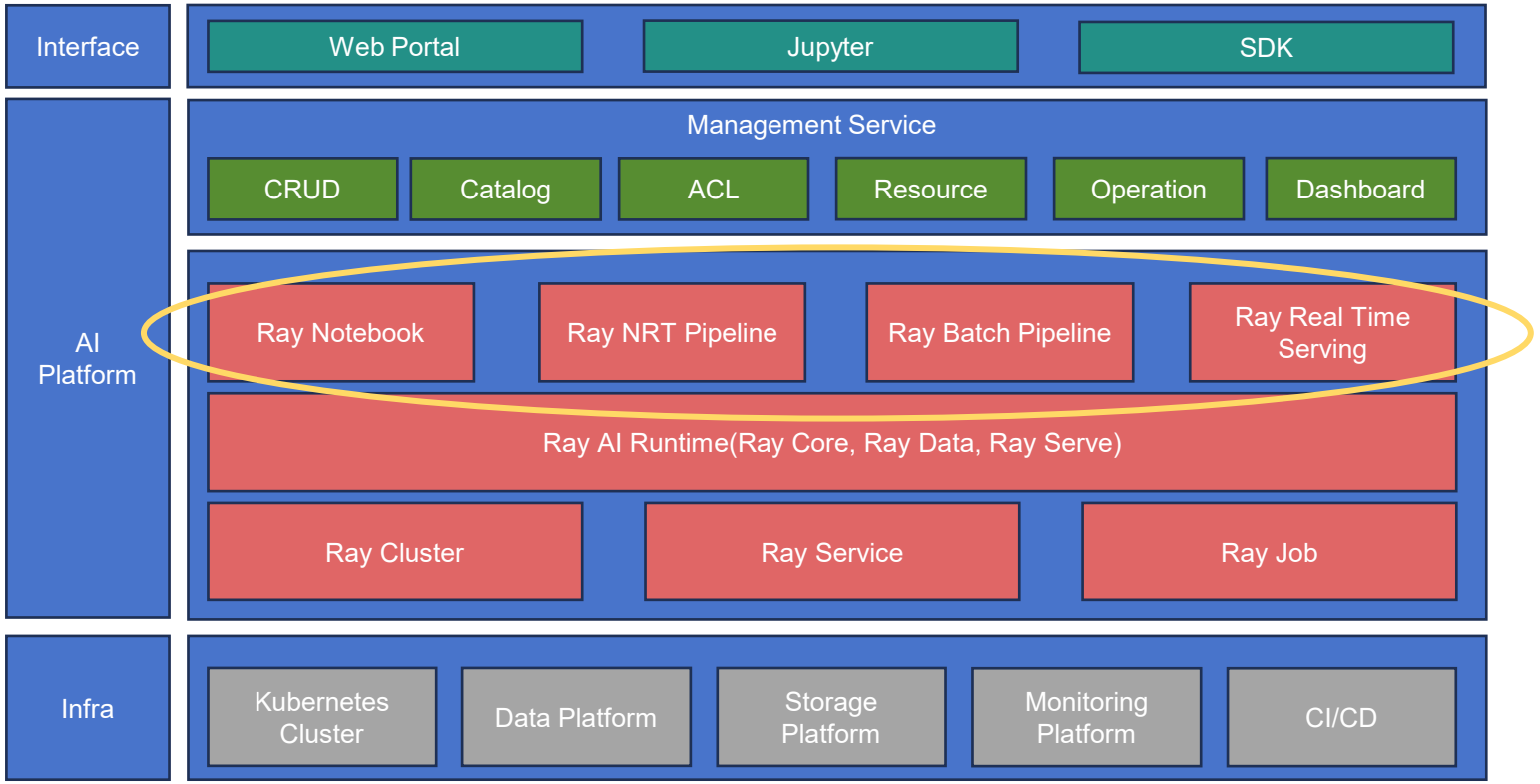
OpenAI, Co-founder John Schulman

Ray.io Introduction



- Provides AI runtime libraries with abstractions to simplify ML code
- Ray Data for high throughput streaming/batch for data loading in distributed ML training
- Distributed workflows to orchestrate across pipeline of model endpoints in python.
- Managing Cloud-Native Applications with the Kuberay operator.

High Level Architecture

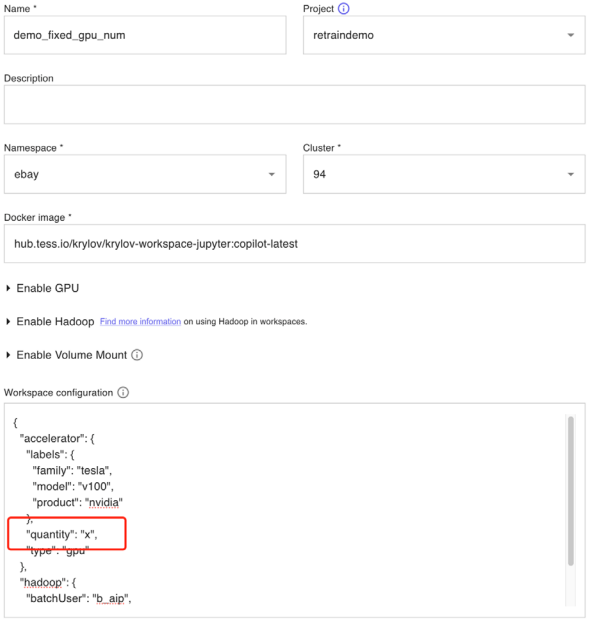


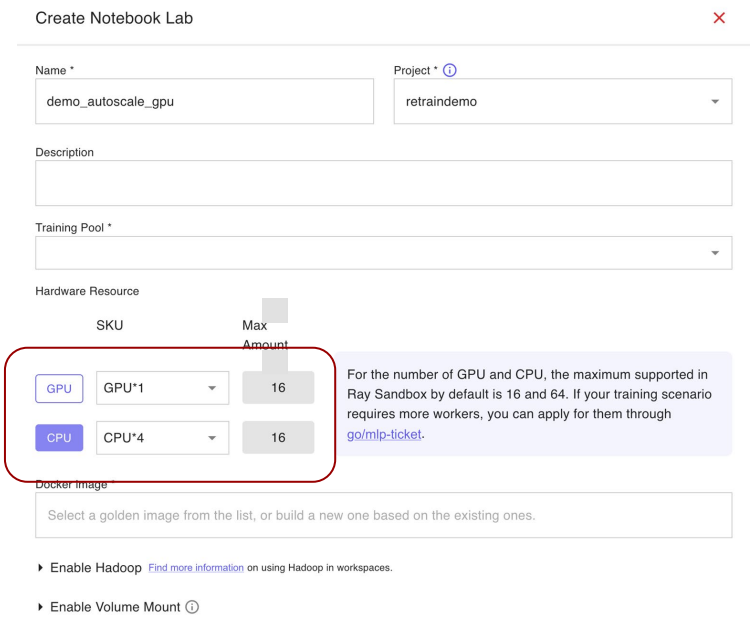
eBay AIP 2.0 Powered By Ray

- Ray Introduction
- High Level Architecture
- Production Scenarios
  - Notebook for Research & Experiment
  - Batch Inference
  - Near Real Time Inference

Notebook for Research & Experiment

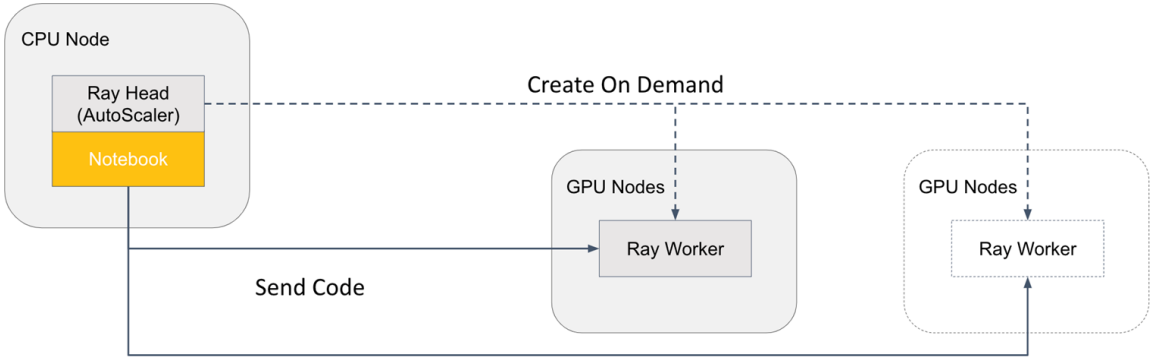
- Old notebook dev env has very limit GPUs, not sufficient for large model development and debug





Ray Notebook

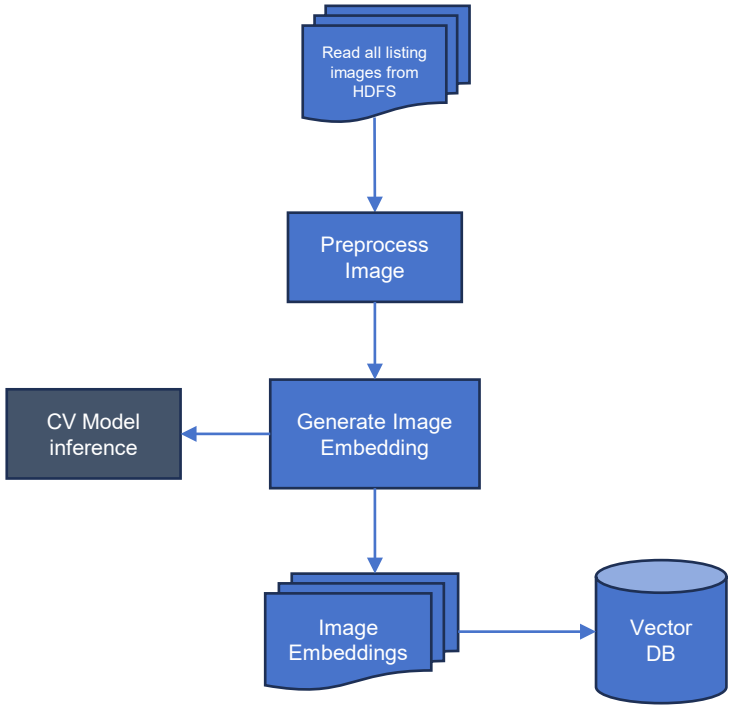
- **Autoscale** Ray cluster based on resource demand, enhancing GPU availability & utilization efficiency.
- Define **standardized GPU SKU** to reduce GPU resource fragmentation.



Standard GPU Sku: fix GPU to Memory ratio

	SKU	Max Amount
GPU	GPU*1	
CPU	CPU*4	

Batch Inference



Low GPU utilization ~10% in some pipelines

- Ray data **stream execution** to reduce IO
- **Big batch size** with Triton
- **Fraction GPU** with Ray

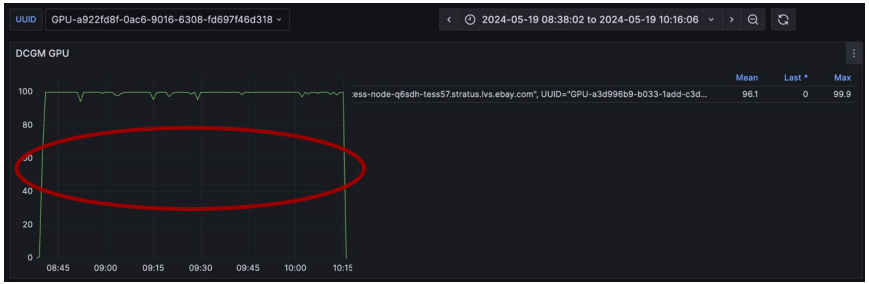
Fixed GPU allocation, limit throughput lack of autoscaling to leverage idle GPUs

- Standardized **GPU SKU**
- Ray **auto scaling**
- K8S **preemption** scheduling

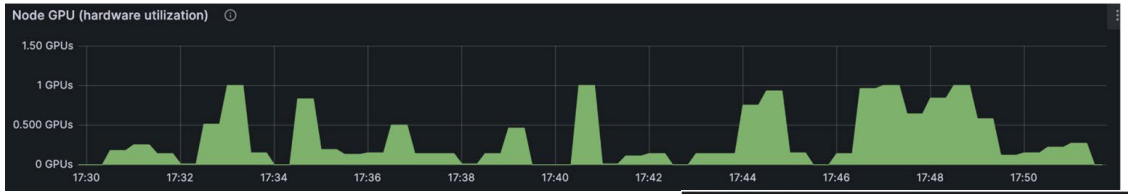
Maximize GPU Utilization

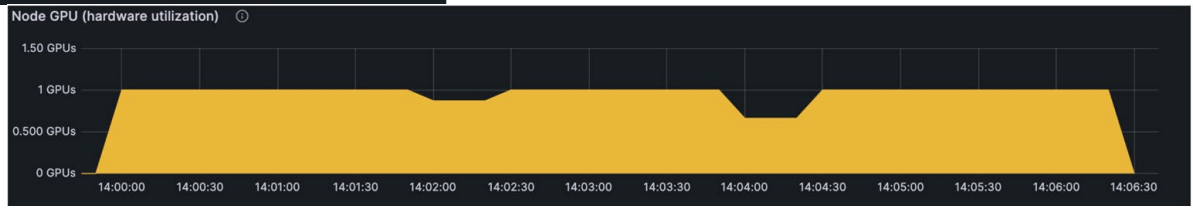
```
dataset = dataset.map_batches(ImageDownloader,
                              batch_size=200,
                              concurrency=(1, 1000), num_cpus=1,
                              resources={"node:worker_group:cpu-group": 0.0001})
dataset = dataset.map(ClipEmbedding_preprocess,
                      resources={"node:worker_group:cpu-group": 0.0001})
dataset = dataset.map_batches(
  ClipModel,
  concurrency=(1, 10),
  num_gpus=0.1,
  batch_size=32,
)
```

4.x Throughput & GPU Utilization (V100)



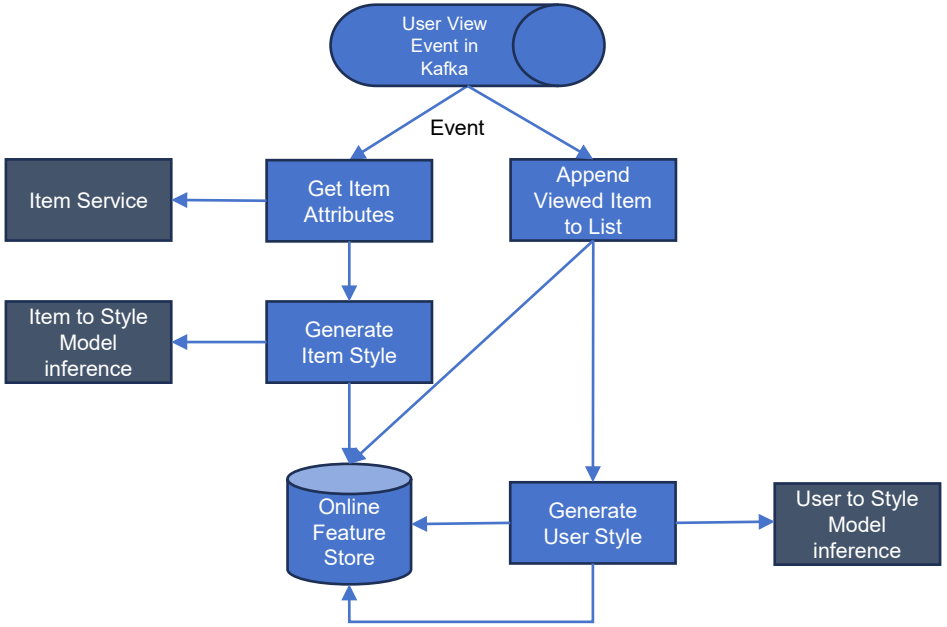
Risk SPM inference GPU utilization improved from 25% to 75%, resulting in an approximately 264% enhancement in end-to-end performance.





Near Real Time Inference

- User viewed items =>
- User style classification



User Style Generation NRT Pipeline

NRT Inference: Low Dev Velocity

Model Development

- Pre/Post processing logic with Python
- Heterogeneous compute: CPU/GPU task split

Data Pipeline Development

- Rewrite pre/post processing with Java/DSL
- Orchestrate pipeline with Java / DSL & optimize DAG

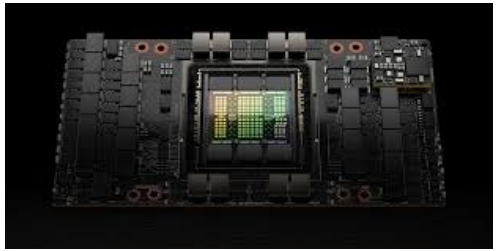
Unified Ray Pythonic API

- Dev both model and production pipeline in Python
- Easy to handle both GPU and CPU tasks
- Easy to orchestrate pipeline and optimize DAG

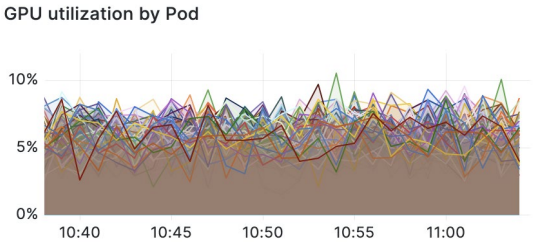
```
@serve.deployment(
    name="NsfwModel",
    …
    ray_actor_options={"num_cpus": x, "num_gpus": y}
)
Class NSFWModel:
    …


@serve.deployment(
    name="ImageDownload",
    …
    ray_actor_options={"num_cpus": x, "num_gpus": 0}
)
Class ImageDownload:
    …
```

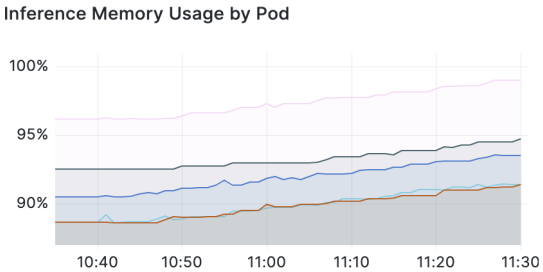
```
with ray.init(...):

  ingestion = ImageDownload.bind()

  # deploy model deployment
  model = NSFWModel.bind()

  # deploy solution deployment
  solution_deployment = ImageProfile.bind(ingestion,
model)
  serve.run(solution_deployment, name="solution")
```

NRT Inference: Low GPU Utilization

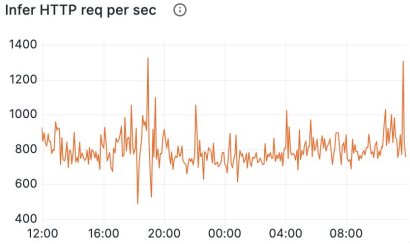


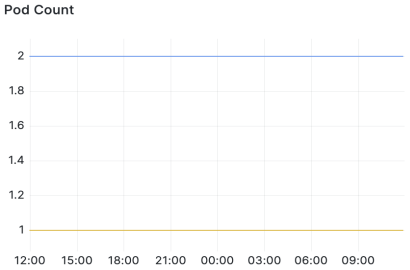
Expensive but in High Demand





Bottleneck on CPU/Memory

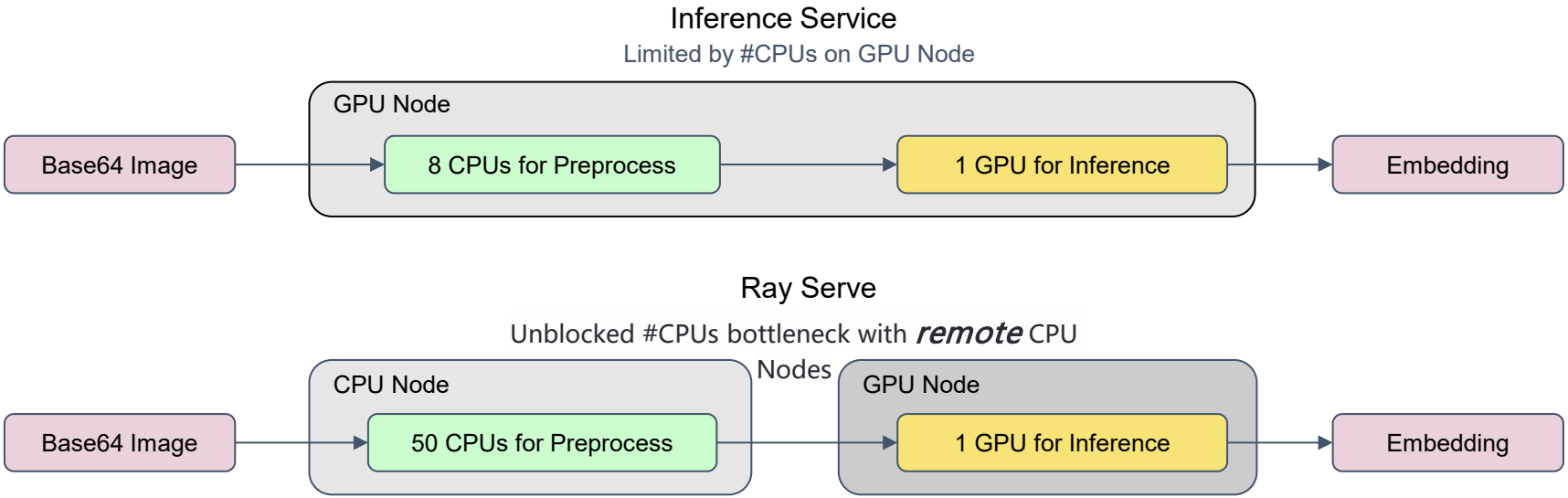




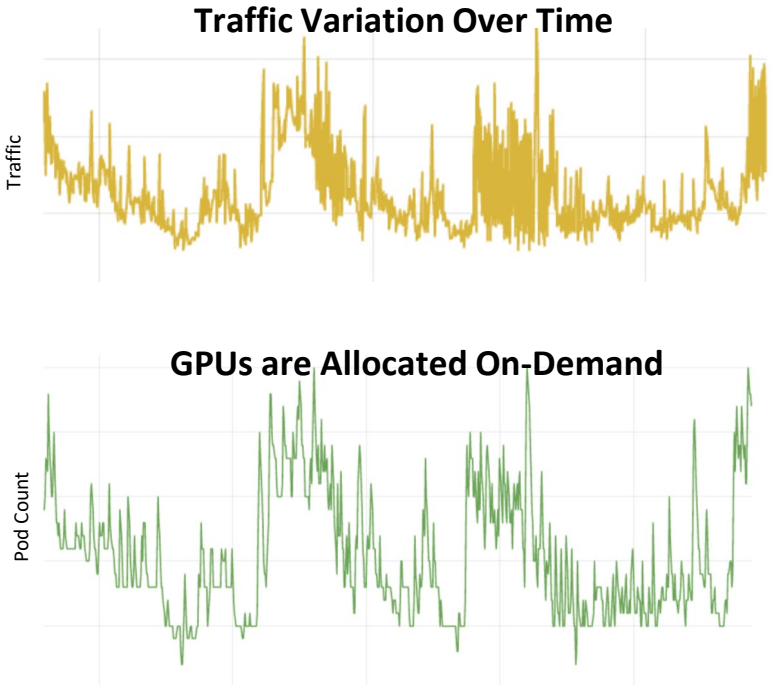
Peak Traffic based Allocation

CPU Bottleneck

- CPU and GPU do not match well, solved by remote CPUs with Ray



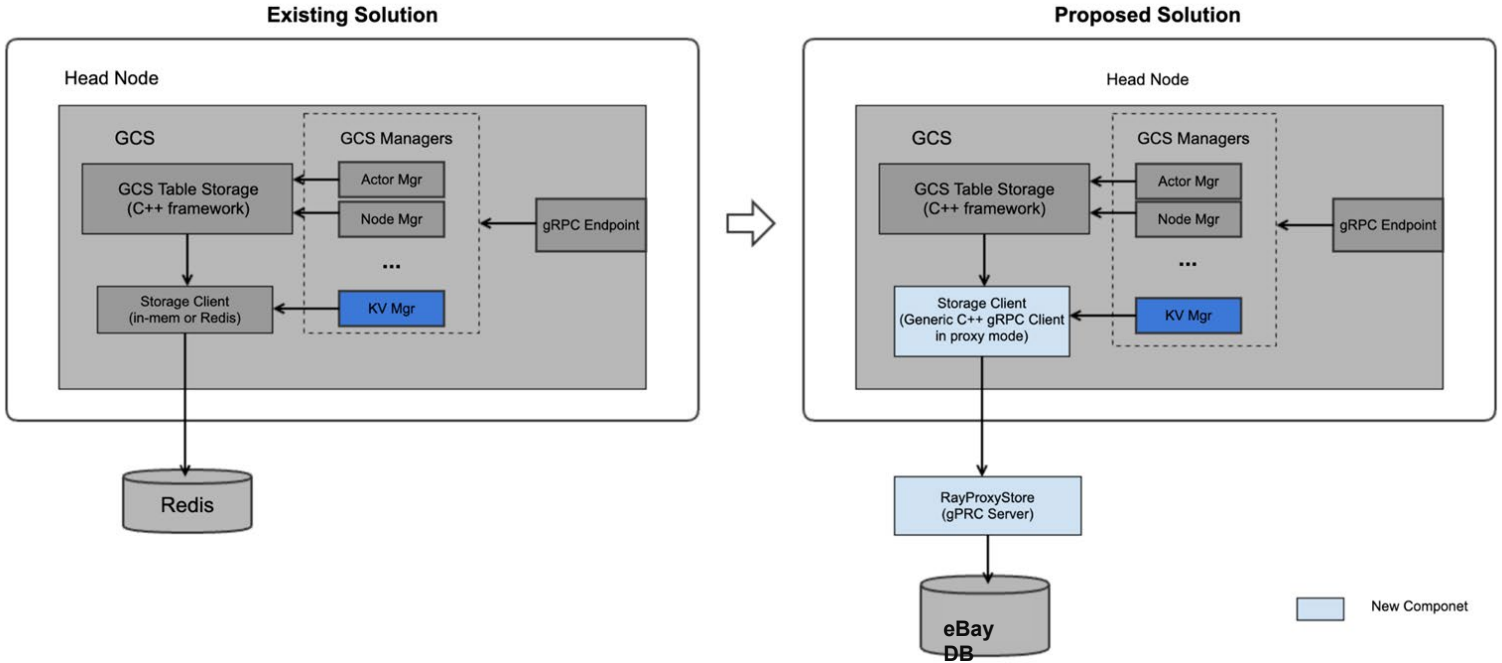
Increase GPU Utilization



7x Throughput & 4x GPU Utilization (V100)

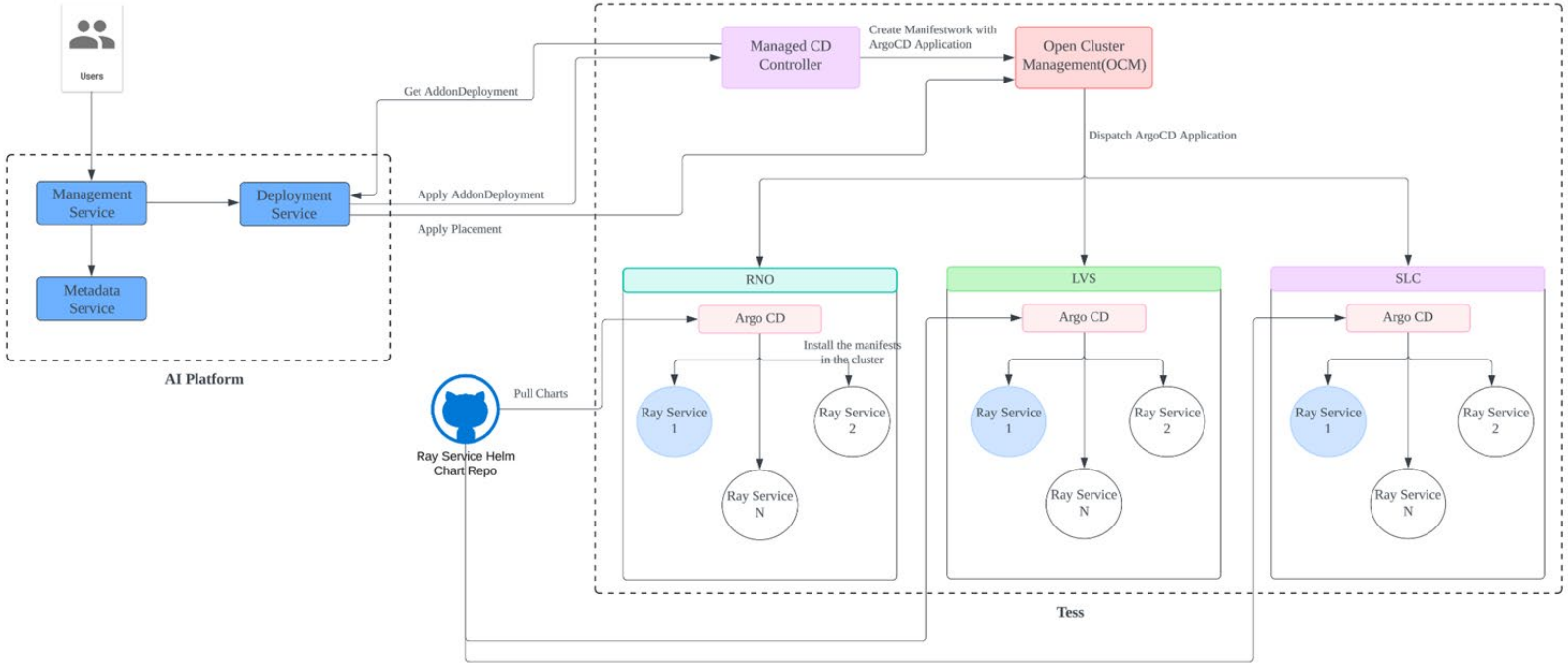
Ray GCS Fault Tolerance

- eBay does not maintain Redis, replacing it with eBay DB



Ray Federation Deployment

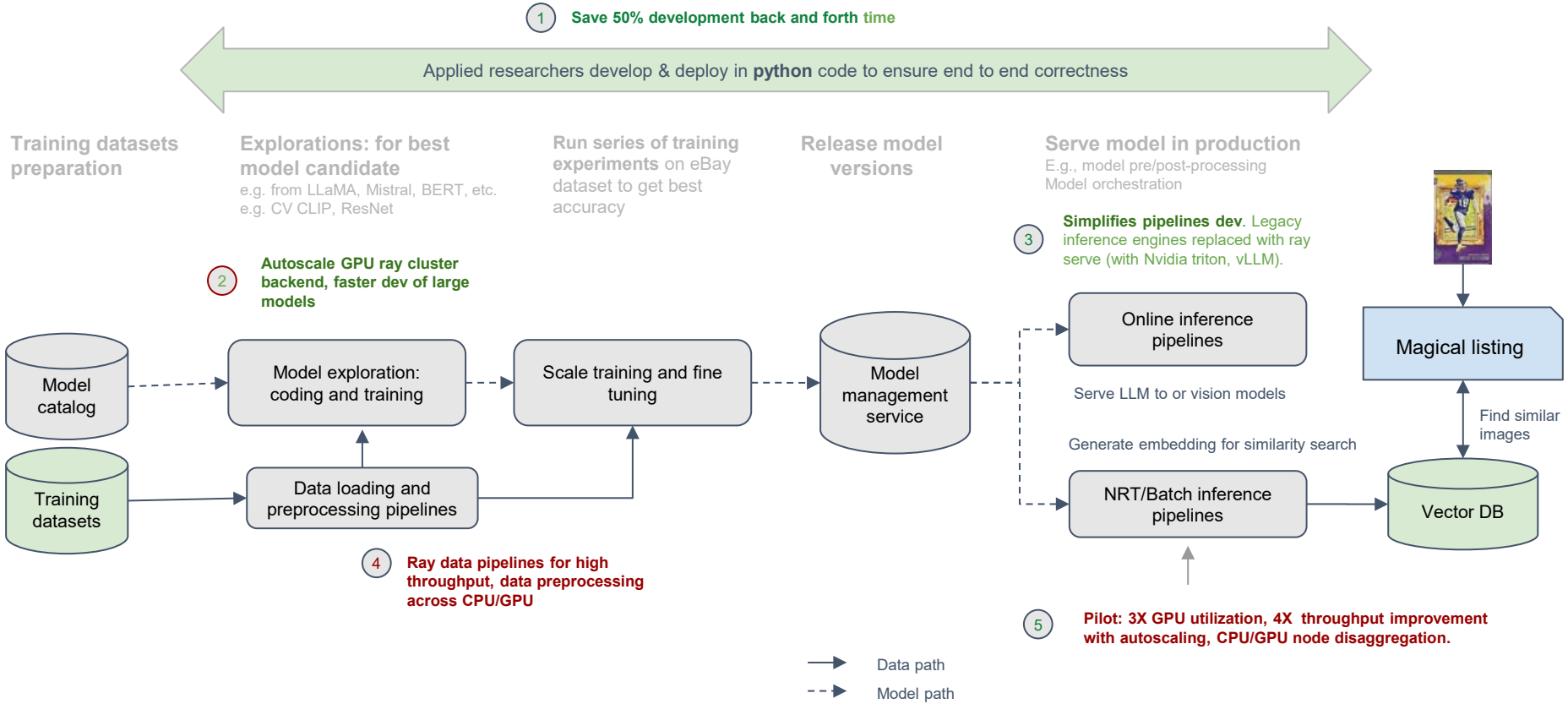
- Multi data center for NRT Inference/Online Serving HA



Important Fixes

- Ray & DeepSpeed ZeRO-3 integration bug fix (Merged)
  - https://github.com/huggingface/accelerate/pull/2578
- Ray & Java reference memory leak bug fix (Merged)
  - https://github.com/ray-project/ray/pull/45729
- Ray & Java Cross-Lang Service bug fix (In Review)
  - https://github.com/ray-project/ray/pull/46770
  - https://github.com/ray-project/ray/pull/46771
- More to Come ….

eBay AIP 2.0 Powered By Ray



Summary

Pillar	Current Pain Points	Value of Ray
Velocity & TTM	• Model Dev - Engineers get engaged to rewrite pre/post processing logic (Python) for production (**a few weeks per model refresh**) • ML Solution Dev - Huge back & forth communications & engineer engagement efforts (**~50% solution develop time**) due to code re-writing and solution fine tuning needed for production	• Unified frameworks for research, testing & production • Code portability and deployment flexibility • Minimized PD engagements and communications efforts for ML solution integration
GPU Utilizations	• Some low GPU utilization ML workloads are bottleneck on CPU **(GPU utilization ~10%)** • Splitting CPU/GPU workloads need **heavy dev works**	• Huge improvements on GPU utilizations by separating CPU/GPU computations • Fraction GPU
GPU Availability	• Each research env(krylov notebook) can only get **very limited GPUs.** • Job size is **static** during runtime	• On-demand resource allocation • Elastic training • Ray auto-scaling

Future Plan

- eBay Ray Cluster HA Solution
- Online Serving Based on Ray
- Log History Server Solution
- eBay Security Integration
- LLM Serving Support

Thank You !