
KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024

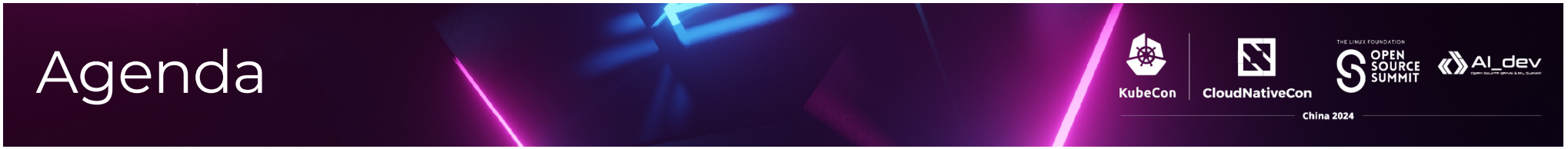

KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Empower Large Language Models (LLMs) in Production With Cloud Native AI Technologies
Lize Cai, Senior Software Engineer, SAP
Yang Che, Senior Engineer, Alibaba Cloud

About us

Lize Cai
Senior Software Engineer in SAP

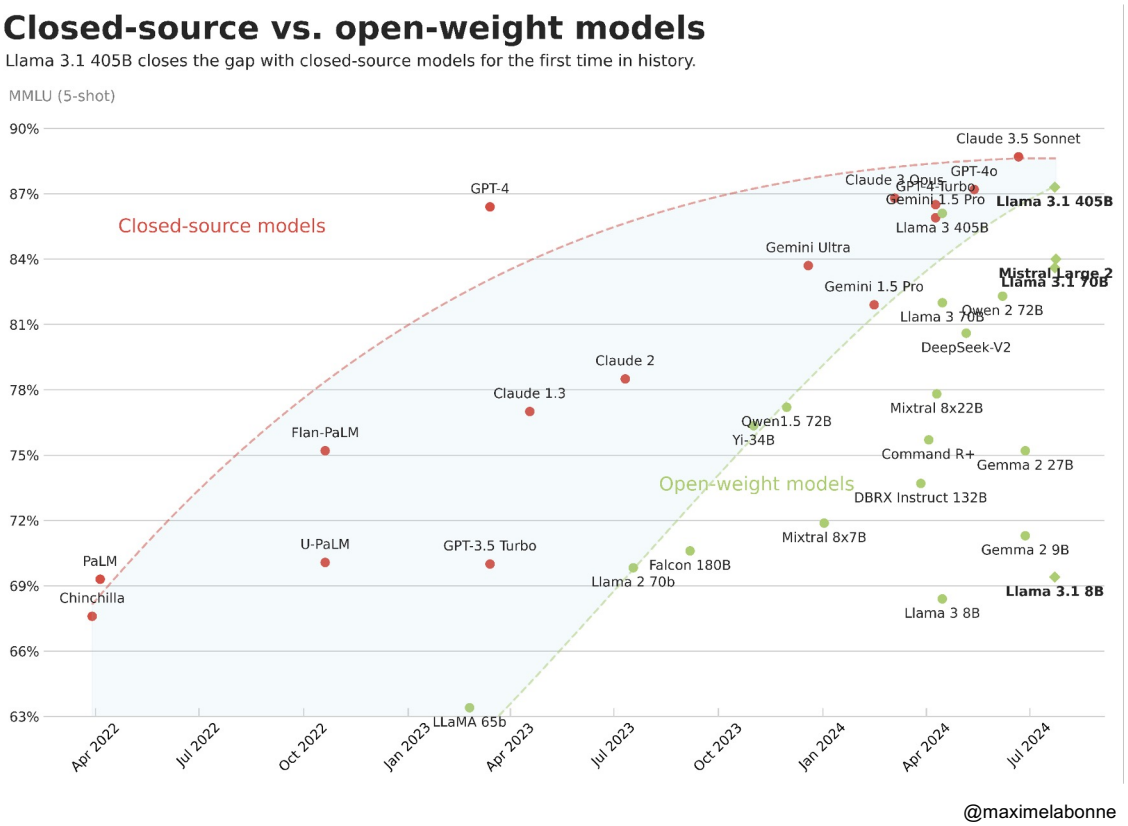
Yang Che
Senior Engineer in Alibaba Cloud

Agenda

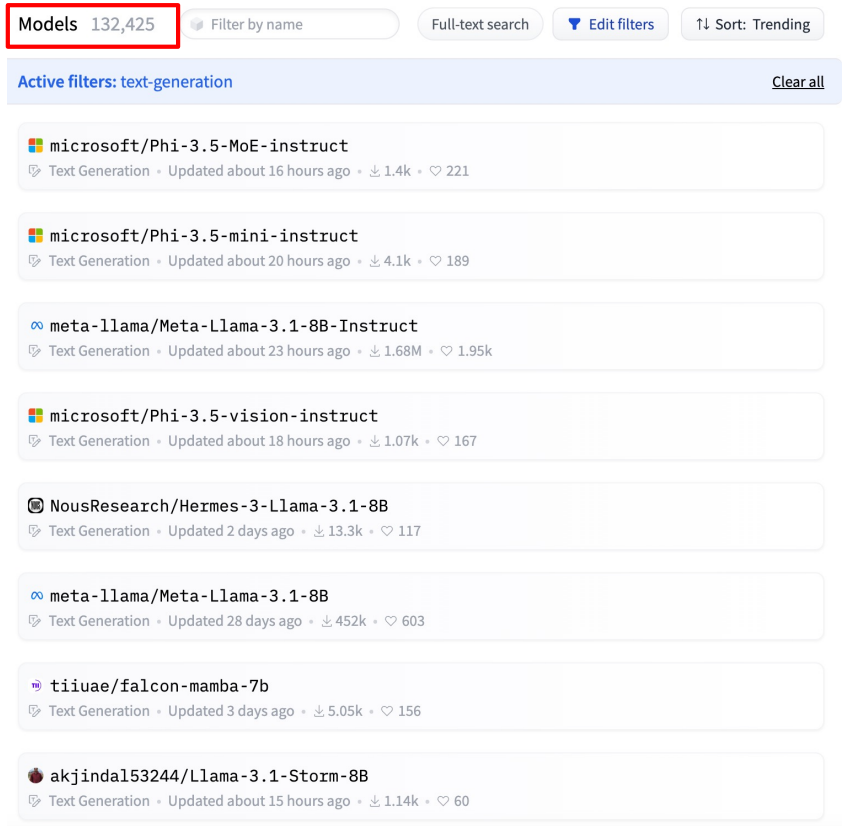


- Introduction
- LLM Challenges in Production
- Manages LLM lifecycle in the K8s way - KServe
- Accelerates LLM scaling from data perspective - Fluid
- Demo
- Future Works
- QA

Introduction

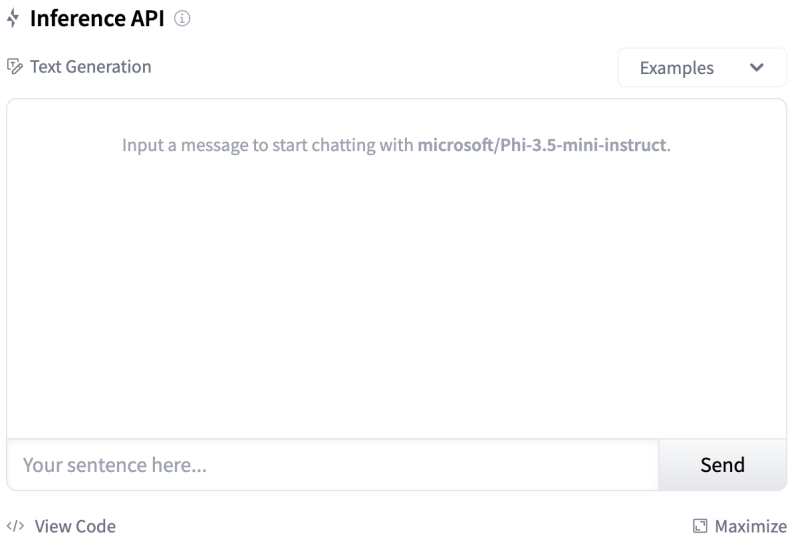


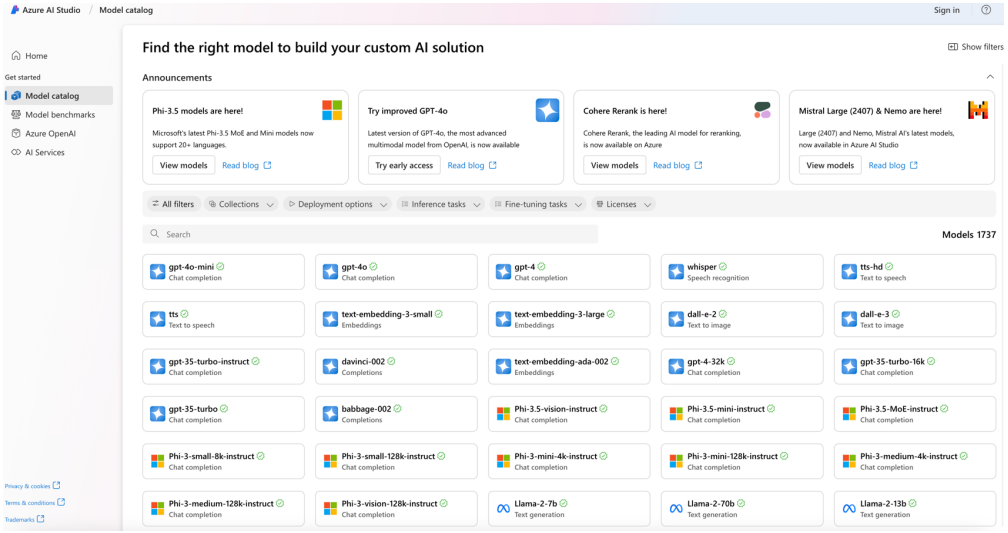
@maximelabonne



Introduction

It is a common use case to provide a playground to try out different models …

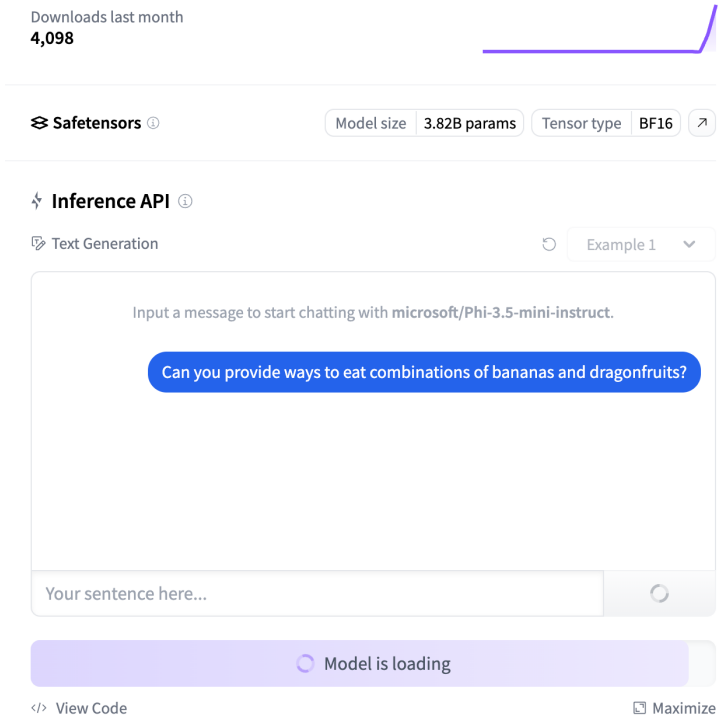


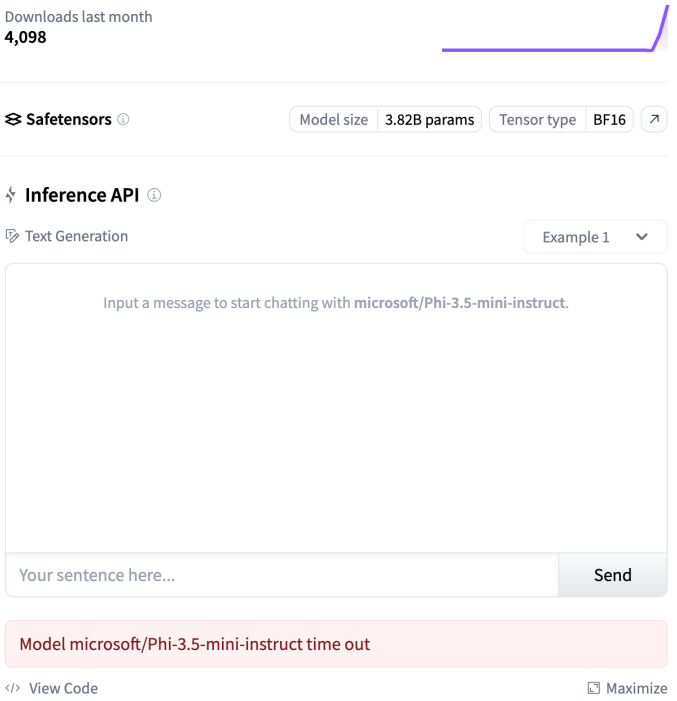


Introduction

KubeCon CloudNativeCon China 2024 | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

But it is not so easy…





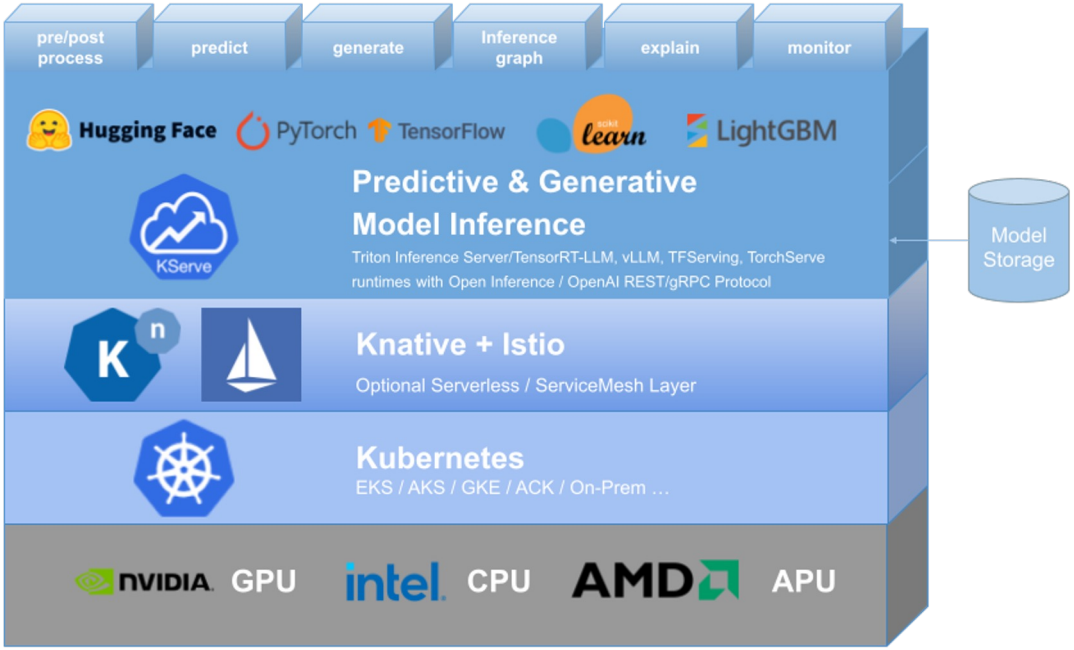
LLM Challenges in Production

- **New requirements on serving LLM**
  - New inference APIs like text generation, embeddings.
  - Streaming response is required for real-time user experience.
- **Variety of models and runtimes**
  - TGI, vLLM, TRT-LLM etc.
  - Llama, Mistral, Phi, Qwen etc.
- **LLM services from cloud providers**
  - Different providers have their own spec (api and token calculation) which leading to a poor user experience and increased maintenance efforts.
- **High computing cost**
  - the need for expensive hardware, high energy consumption, and associated infrastructure expenses.
- **Data privacy**
  - Model and request data can be sensitive and private for inference.


Manages LLM lifecycle in the K8s way - KServe

What is KServe?

Highly scalable and **standards**-based **cloud-native model inference platform** on **Kubernetes** for trusted AI that encapsulates the complexity of deploying AI models to production.



What is KServe?

Core Inference

- Transformer/Predictor
- Serving Runtimes
- Custom Runtime SDK
- Open Inference Protocol
- Serverless Autoscaling
- Cloud/PVC Storage

Advanced Inference

- ModelMesh for Multi-Model Serving
- Inference Graph
- Payload Logging
- Request Batching
- Canary Rollout

Model Explanability & Monitoring

- Text, Image, Tabular Explainer
- Bias Detector
- Adversarial Detector
- Outlier Detector
- Drift Detector

What is KServe?

Serving runtime support matrix

Serving Runtime/ Model Format	scikit-learn	xgboost	lightgbm	TensorFlow	PyTorch	TorchScript	ONNX	MLFlow	Custom	HuggingFace
MLServer (open)	✓	✓	✓					✓	✓	
Triton (open)				✓		✓	✓			
TorchServe (v1, open)					✓	✓				✓
KServe Runtime (v1, open)	✓	✓	✓						✓	
TFServing (v1)				✓						
HuggingFace Server (v1,v2,openAI)										✓
HuggingFace vLLM Server (v2, openAI)										✓

KServe on LLM

Inference Service and Serving Runtime for LLM

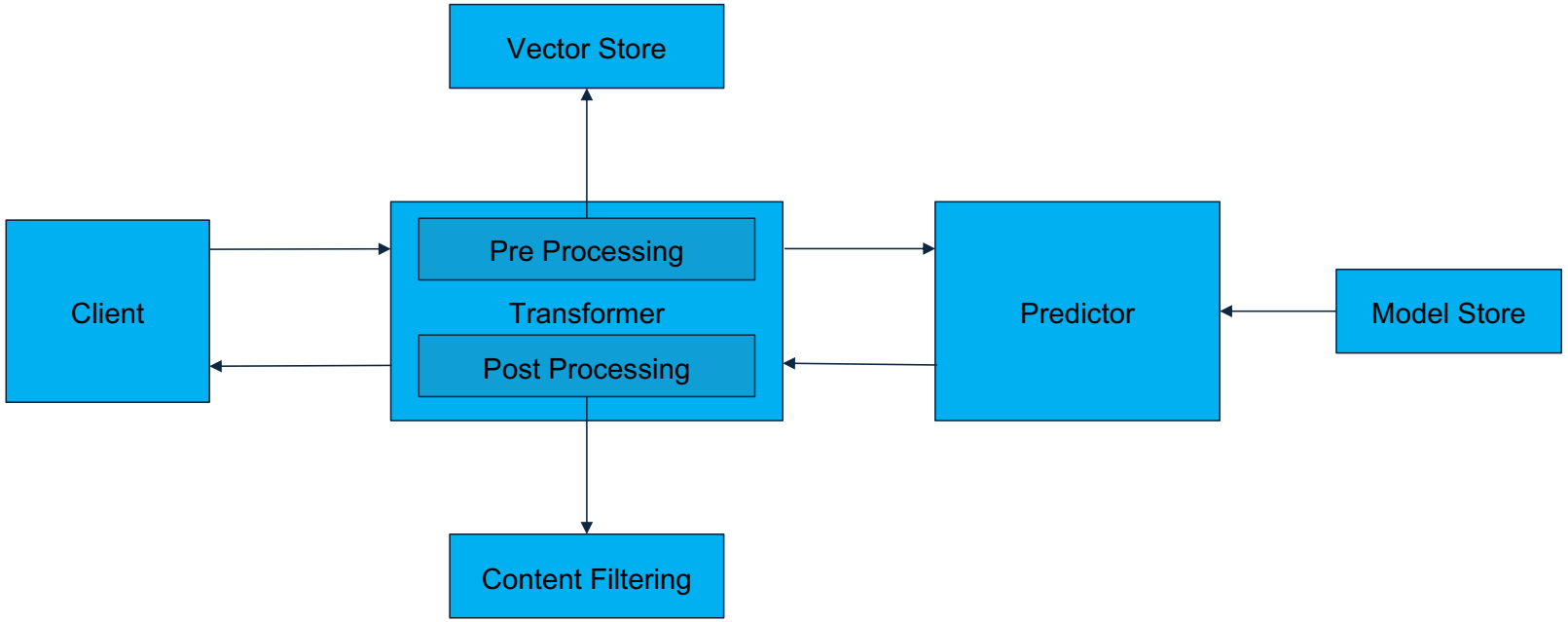
KServe User

```yaml
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: huggingface-llama3
spec:
  predictor:
    model:
      modelFormat:
        name: huggingface
      args:
        - --model_id=meta-llama/meta-llama-3-8b-instruct
      resources:
        limits:
          cpu: "6"
          memory: 24Gi
          nvidia.com/gpu: "1"
        requests:
          cpu: "6"
          memory: 24Gi
          nvidia.com/gpu: "1"
```

KServe Admin

```yaml
apiVersion: serving.kserve.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: kserve-huggingfaceserver
spec:
  annotations:
    prometheus.kserve.io/port: '8080'
    prometheus.kserve.io/path: "/metrics"
  supportedModelFormats:
    - name: huggingface
      version: "1"
      autoSelect: true
      priority: 1
  protocolVersions:
    - v1
    - v2
  containers:
    - name: kserve-container
      image: "kserve/huggingfaceserver:latest"
      args:
        - --model_name={{ .Name }}
      resources:
        requests:
          cpu: "1"
          memory: 2Gi
        limits:
          cpu: "1"
          memory: 2Gi
```

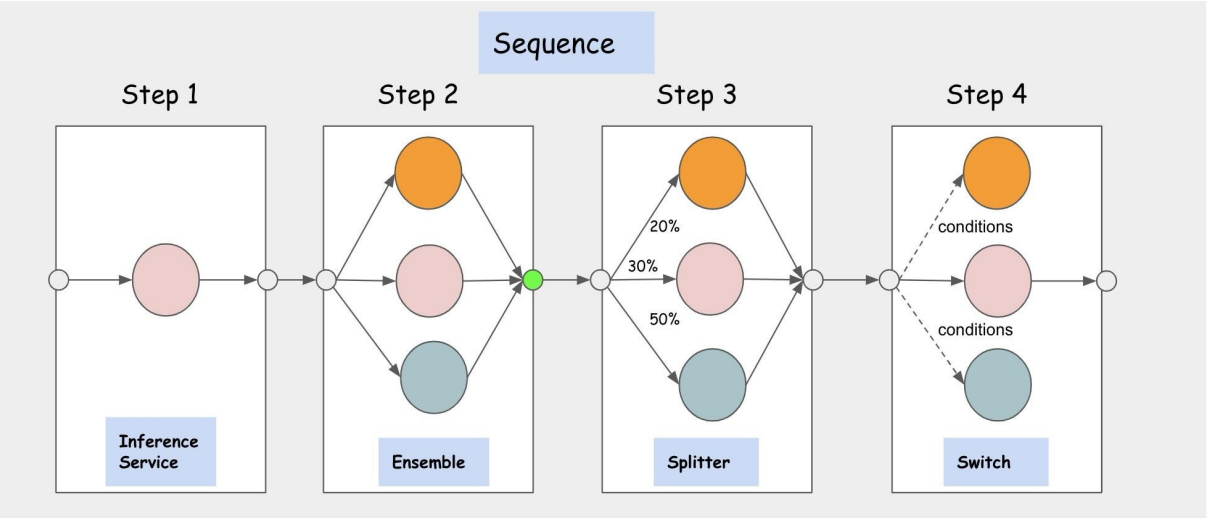
Support of common LLM Use cases



KServe on LLM

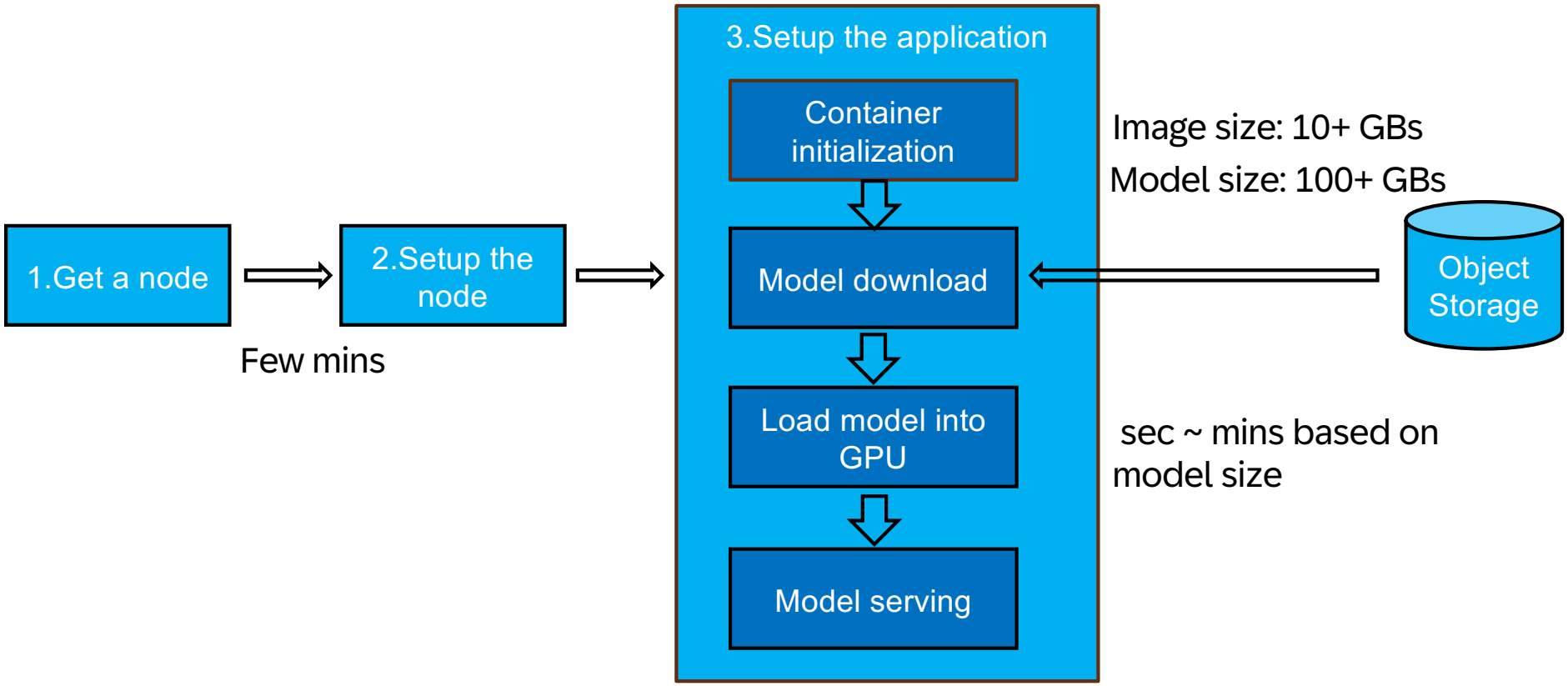
Inference Graph:

- Inference Graph is built for more complex **multi-stage inference pipelines**.
- Inference Graph is deployed in a **declarative way and highly scalable**.
- Inference Graph supports **Sequence, Switch, Ensemble and Splitter** nodes.
- Inference Graph is **highly composable**. It is made up with a list of routing nodes and each node consists of a set of routing steps which can be either route to an InferecenService or another node.

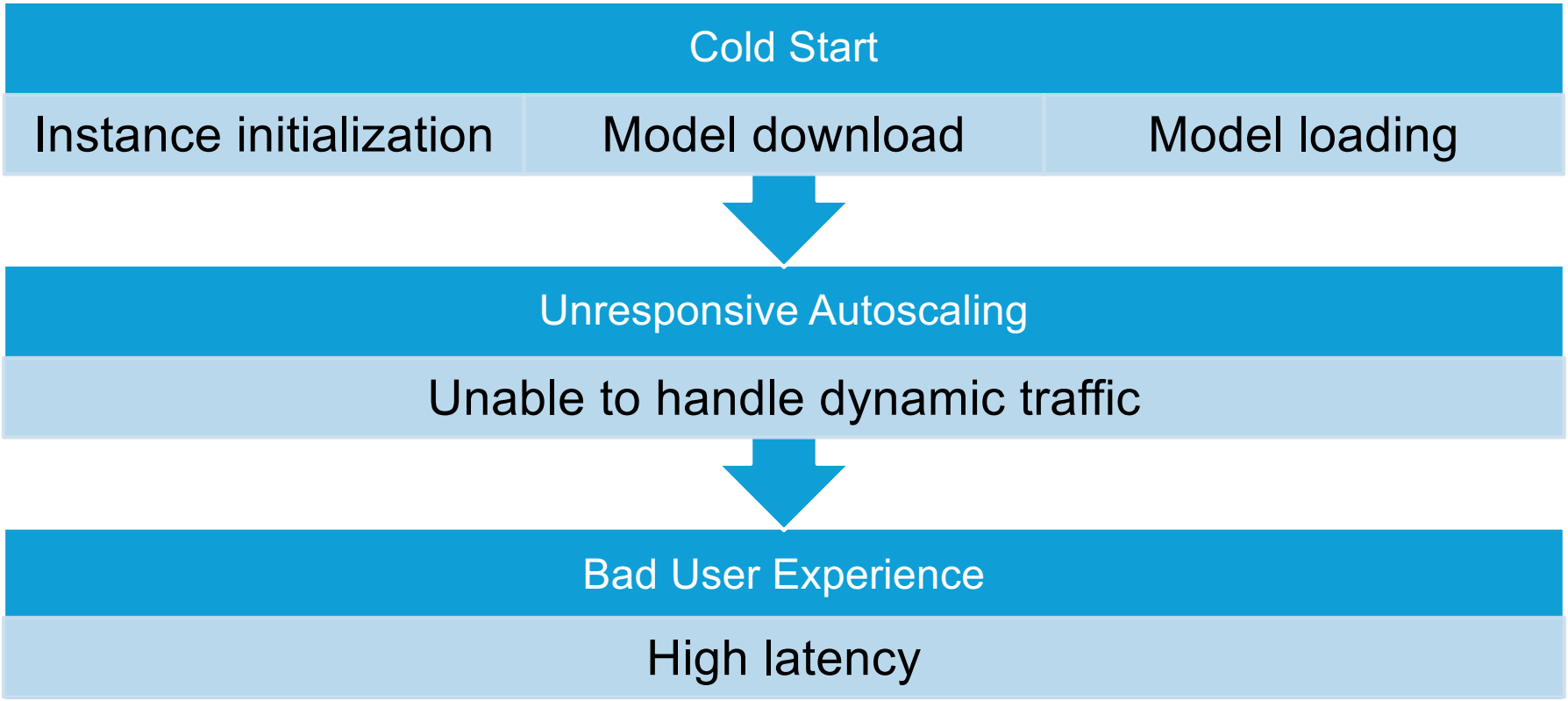



```yaml
apiVersion: "serving.kserve.io/v1alpha1"
kind: "InferenceGraph"
metadata:
  name: "dog-breed-pipeline"
spec:
  nodes:
    root:
      routerType: Sequence
      steps:
      - serviceName: cat-dog-classifier
        name: cat_dog_classifier # step name
      - serviceName: dog-breed-classifier
        name: dog_breed_classifier
        data: $request
        condition: "[@this].#(predictions.0==\"dog\")"
```

Process of Deploying LLM



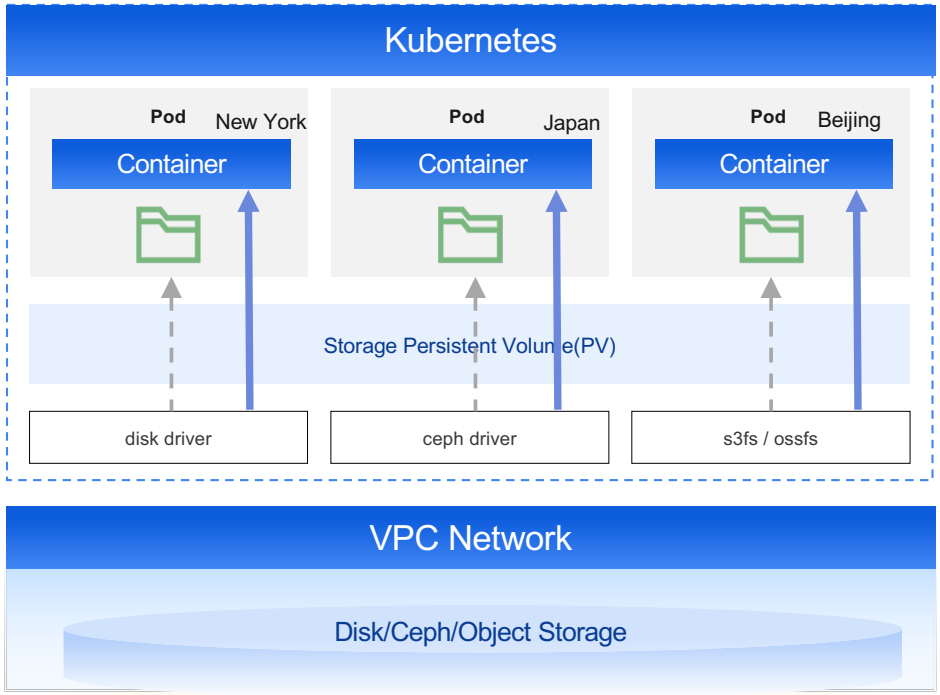
Challenges of Autoscaling in LLM



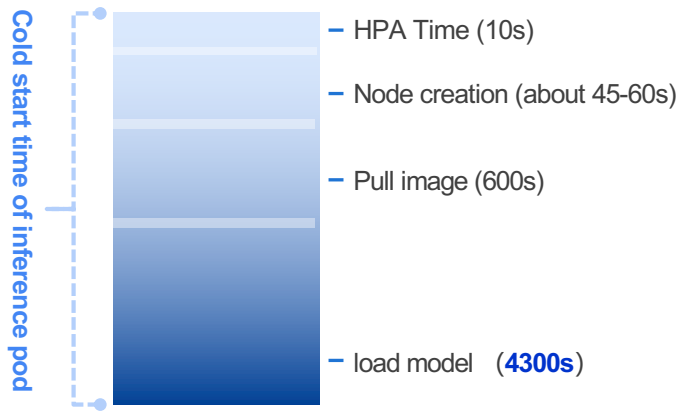

Accelerates LLM scaling from data perspective - Fluid

Pains of LLM inference services in Kubernetes

Compute-storage separation causes high data access latency and limited bandwidth in large AI models, exacerbating cost, performance, and efficiency issues.

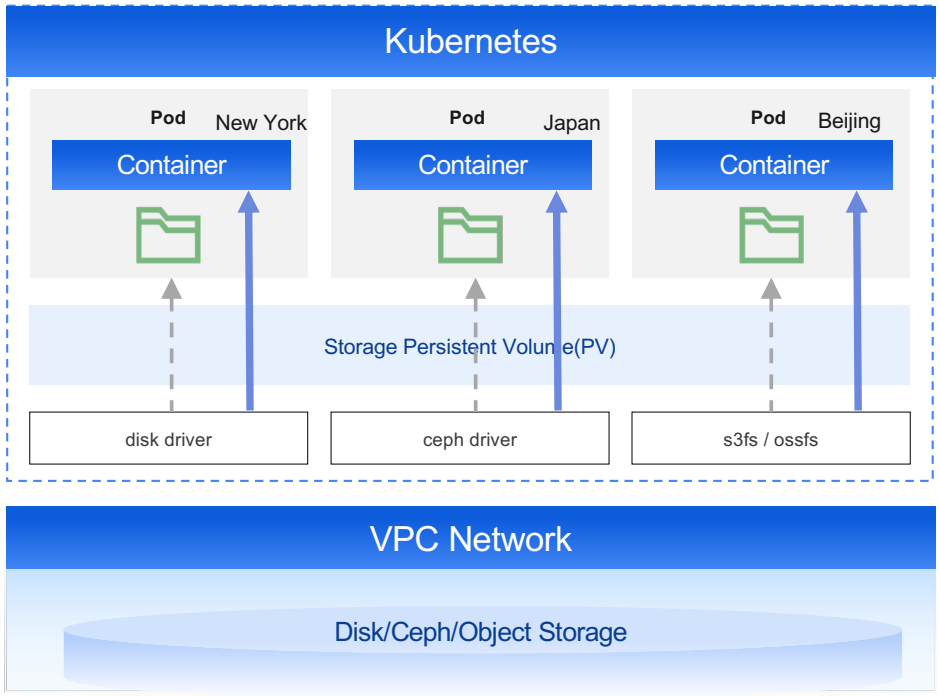


A Pod starts the Bloom-175B model (FP16, approximately 340 GiB), it takes about 4970 seconds, with model loading taking 4300 seconds (85%).



Pains of LLM inference services in Kubernetes

Cache can’t always speed up data access due to engineering challenges in model inference.



Cache Scheduling:

- How to configure affinity scheduling?

User Experience:

- How to use data in cache?
- How to integrate with applications?

Configuration:

- How to distribute cached data?
- What media to use for storage?

Resource Cost :

- Extra compute and storage needs for cache. How to cut costs?
- Cross-region traffic costs

Operations and Management:

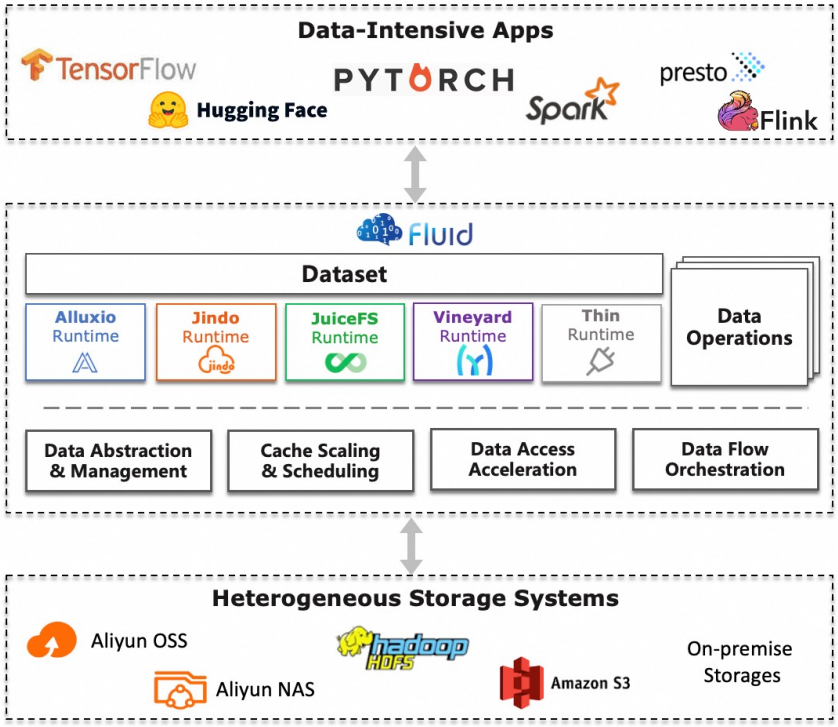
- How to manage cache system lifecycle?
- Operations observability?
- Cross-region model synchronization

Performance Tuning:

How to optimize cache parameters for specific scenarios?

Fluid

Fluid: Data and Task Orchestrator in Kubernetes



Joint launched by Nanjing University, Alibaba Cloud and Alluxio

https://github.com/fluid-cloudnative/fluid

- **Standardized:** K8s Native APIs for data access and distributed cache management.
- **Extensible**: Runtime plugins for different distributed cache and storage backends.
- **Elasticity**: Scale out and in the distributed cache on demand.
- **Performance:** Accelerate data access via elastic distributed cache
- **Automation**: Operation for Data like. *prefetching processing*, *migration* and *cache scaling*
- **Orchestration**: Data and task co-aware scheduling

Fluid Optimization for LLMs

KubeCon CloudNativeCon China 2024 | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

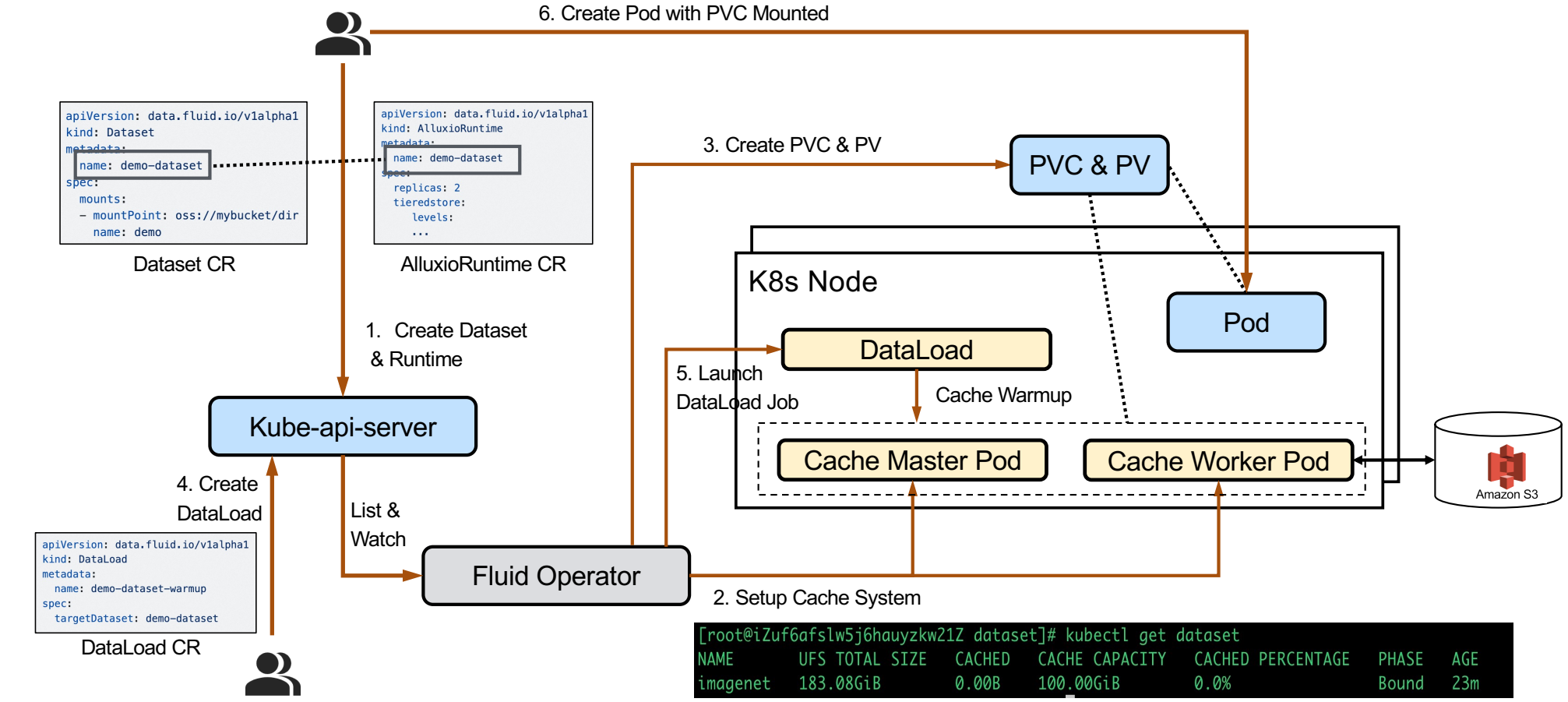
Characteristics and Current Issues in LLMs:

- Distributed caching are *complex* and *vary greatly* across environments
- How to balancing Performance and Cost
- Cross-region/zone data access affects performance
- Data Operations are complex and time-consuming.

Capabilities Provided by Fluid:

- Out-of-the-box acceleration capabilities.
- Elastic Compute-Side Distributed Cache
- Affinity Scheduling for Data and Workloads
- Data Flow for Automated Data Management and Consumption Processes

Out-of-the-Box Distributed Cache



Distributed Cache can Run Anywhere

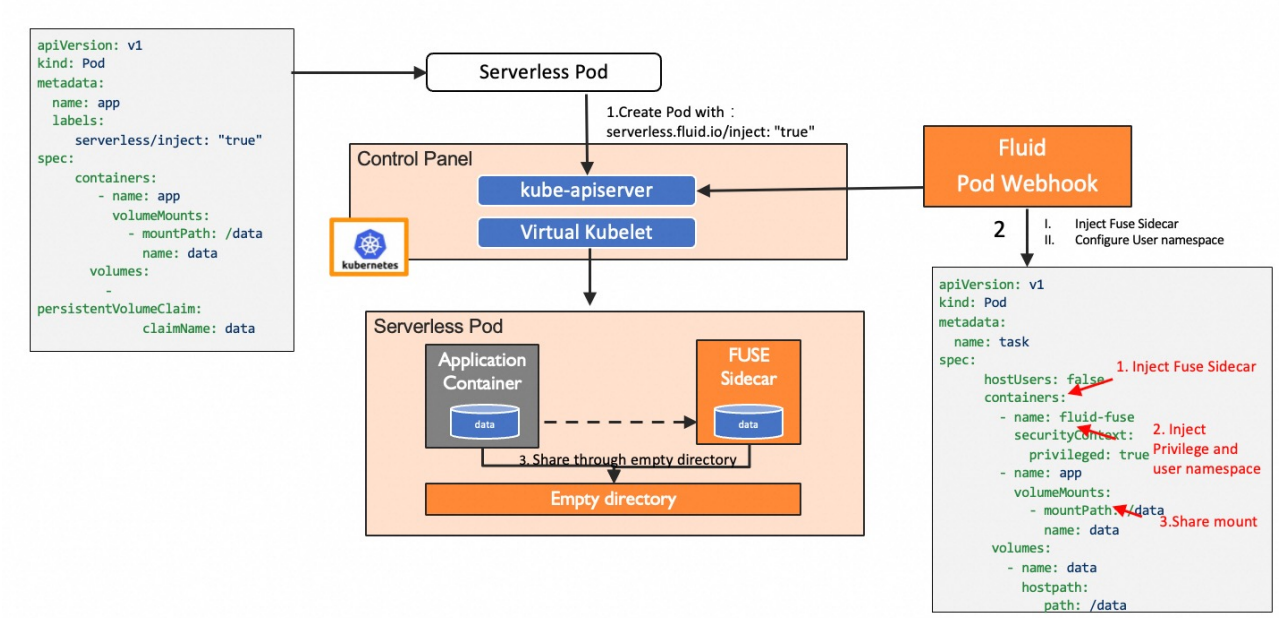
KubeCon CloudNativeCon China 2024 | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

Problem:

LLMs often run in serverless environments cannot use third-party distributed caches due to *extensibility* and *security* limitations.

Fluid:

- ✓ Inject a sidecar to replace PVC with a fuse sidecar
- ✓ control the container startup sequence
- ✓ Also stop sidecar when main container exit



Elasticity of Distributed Cache is important for LLMs

KubeCon CloudNativeCon China 2024 | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

Problem:

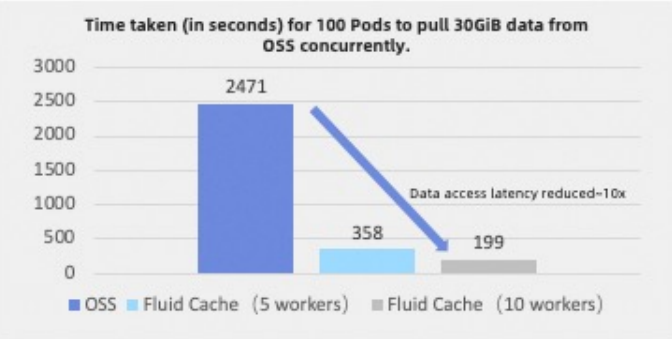
Limited storage-side bandwidth causes ‘cold start’ issues in large language model inference.

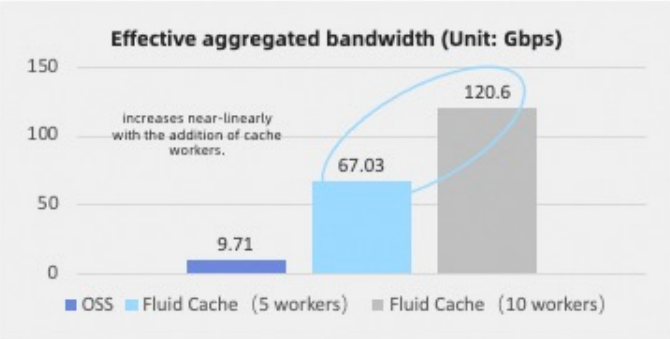
Example:

If 10Gbps OSS Bucket bandwidth is equally split among 100 instances, with 0.1Gbps each, sequential read of a 30GB file could take 2400s.

Fluid:

Using a scalable distributed cache optimizes bandwidth, which directly relates to the size and number of cache nodes.





Elasticity of Distributed Cache is important for LLMs

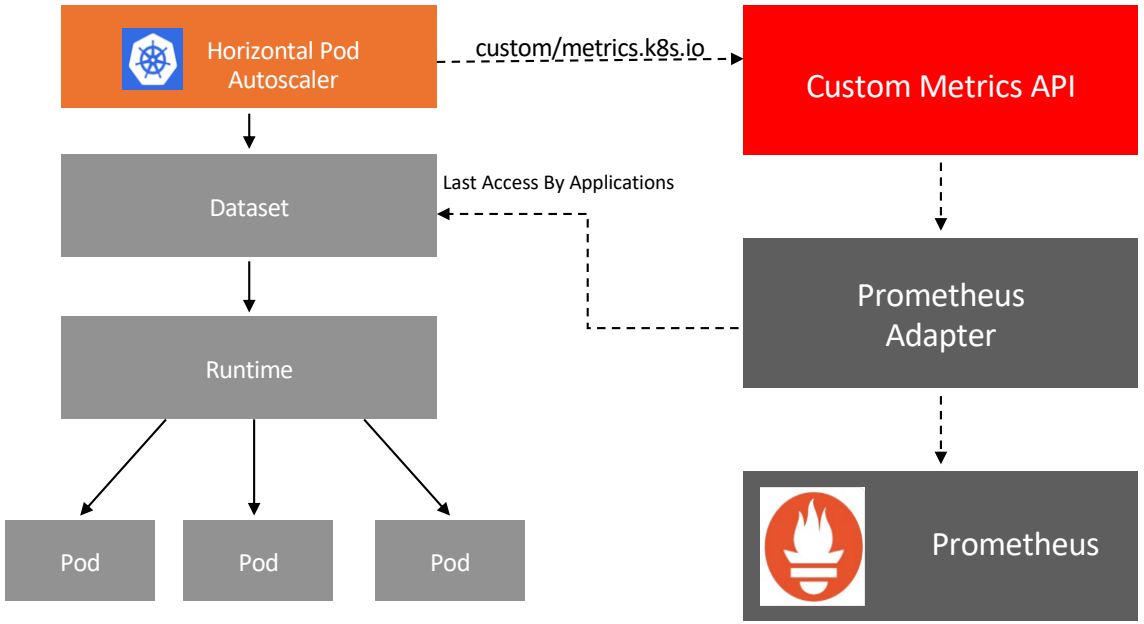
KubeCon CloudNativeCon China 2024 | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

Cost Strategies Across Different cases:

- ✓ One-time intensive loading for LLM models without repetition
- ✓ Frequent loading for image-text models, requiring hot caches
- ✓ Temporary cache expansion needed during business peaks

Fluid:

- ✓ Fine-tuned control over data cache lifecycle, with elastic expansion and contraction based on business needs




```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
 name: imagenet-hpa
spec:
 scaleTargetRef:
  apiVersion: data.fluid.io/v1alpha1
  kind: JindoRuntime
  name: imagenet
 minReplicas: 2
 metrics:
  - type: Object
    object:
      metric:
        name: capacity_used_rate
      describedObject:
        apiVersion: data.fluid.io/v1alpha1
        kind: Dataset
        name: imagenet
      target:
        type: Value
        value: ”90"
```

Hierarchical Cache aware scheduling

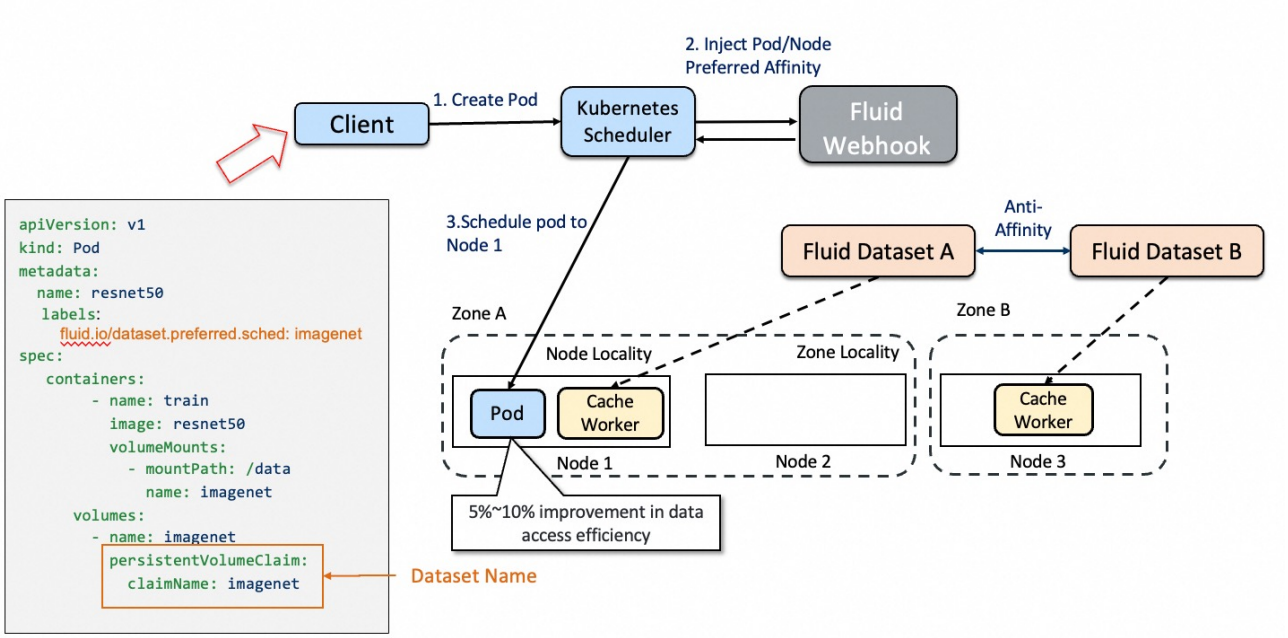
KubeCon CloudNativeCon China 2024 | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

Problem:

LLMs often run in serverless environments cannot use third-party distributed caches due to *extensibility* and *security* limitations.

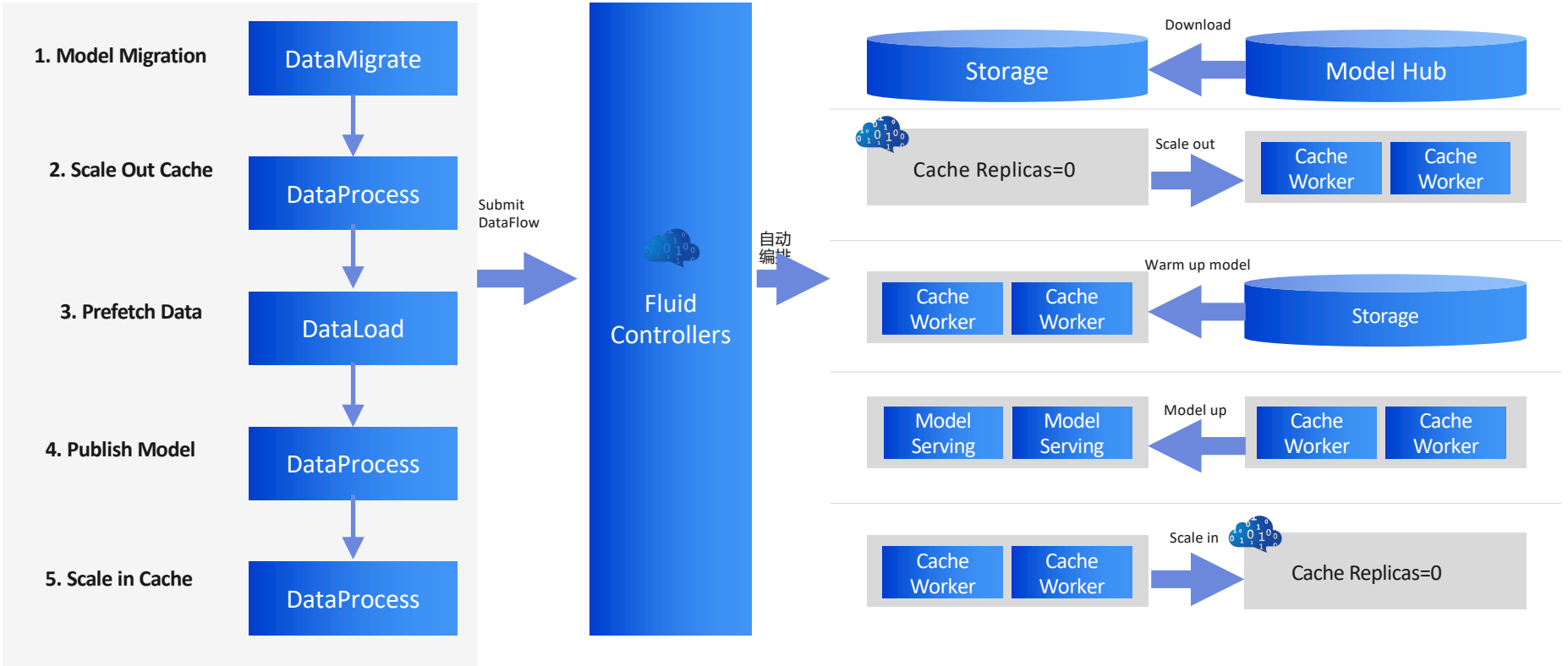
Fluid:

- ✓ Inject a sidecar to replace PVC with a fuse sidecar
- ✓ control the container startup sequence
- ✓ Also stop sidecar when main container exit



Dataflow

Use Fluid for Automated Operations and Maintenance to Balance Performance and Cost in LLMs Daily Work




Demo

Demo

- Cache cluster by Fluid
- Cluster serving runtime by KServe
- Inference service by KServe

- Environment*:
  - m5n.xlarge for cache cluster
  - g5.8xlarge for inference service

- Goals:
  - Fluid integration with KServe
  - OpenAI compatible API in KServe

* The performance might be affected by the environment and setup.

Demo

- Patch min replicas to 2
- Environment*:
  - g5.8xlarge for inference service
- Goals:
  - Scaling feature of Inference Service
  - Speed up scaling by distributed cache

* The performance might be affected by the environment and setup.

Future Works

- LLM Serving Runtimes: TGI, TRT-LLM etc
- LLM RAG Pipeline Orchestration
- GenAI Task APIs
- LLM Gateway
- Enhance Resilience of distributed cache


KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Thank you & QA
Lize Cai
Yang Che