KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

THE LINUX FOUNDATION

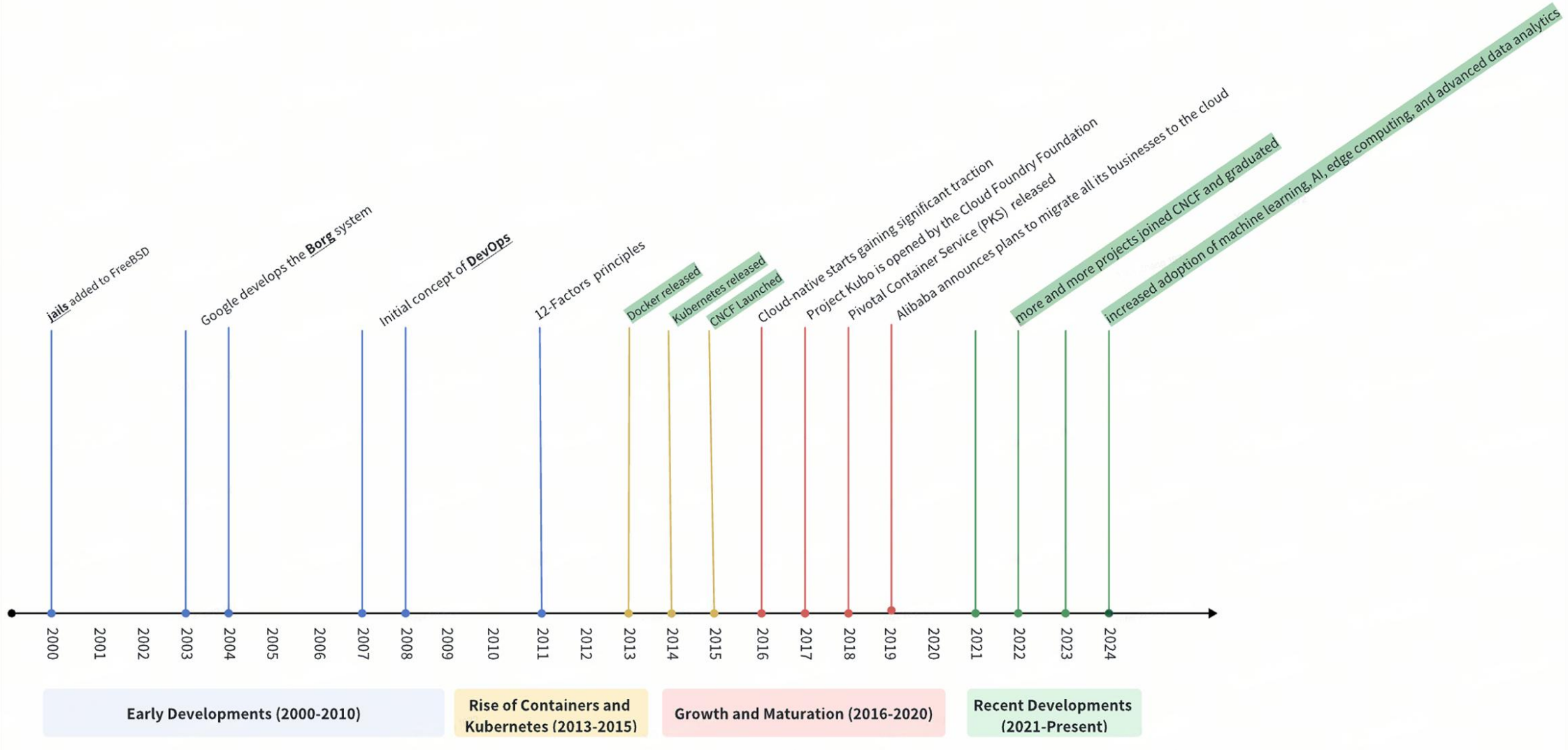
China 2024

AI 推理性能加速: 方法 / 工具 / 部署工作流

张翼飞、钱磊

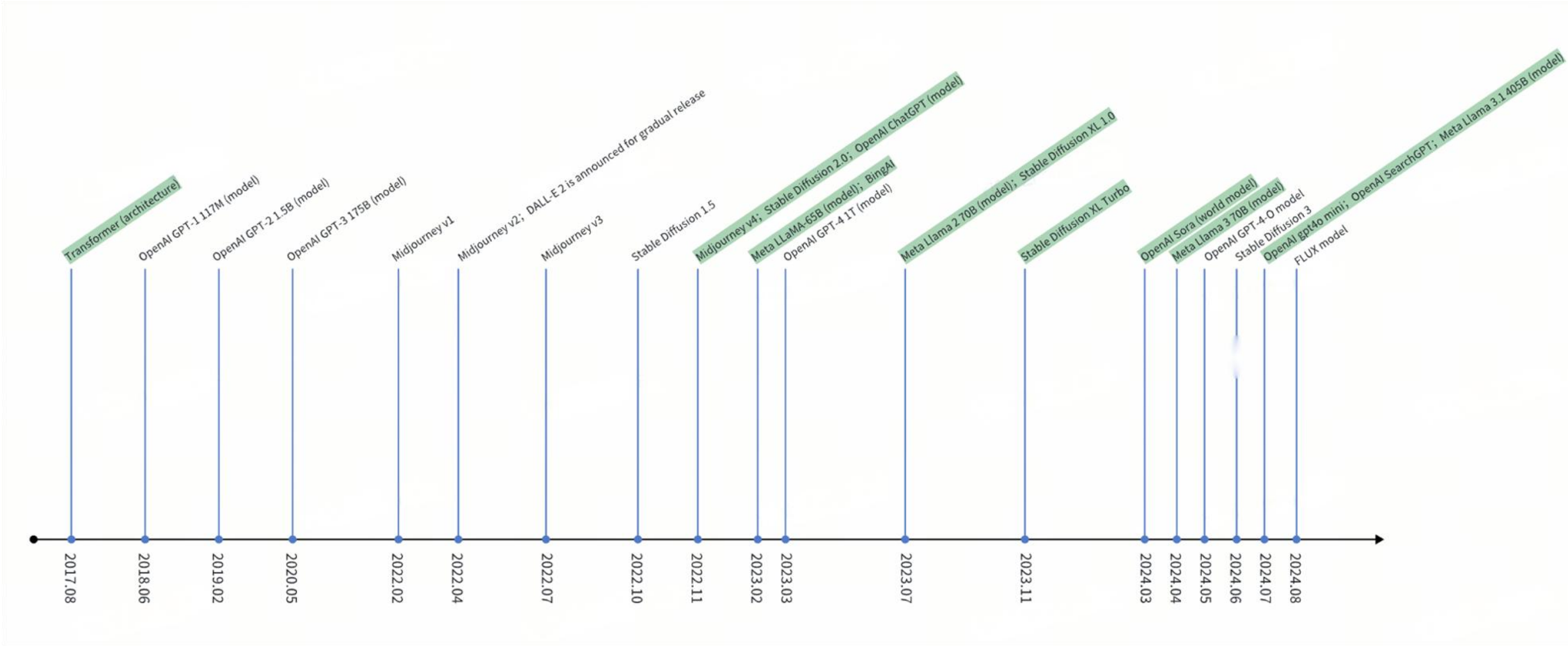
字节跳动云原生开发工程师

AI + 云原生技术趋势



https://www.hivenet.com/post/cloud-computing-history
https://www.redapt.com/blog/cloud-native-applications-a-brief-history

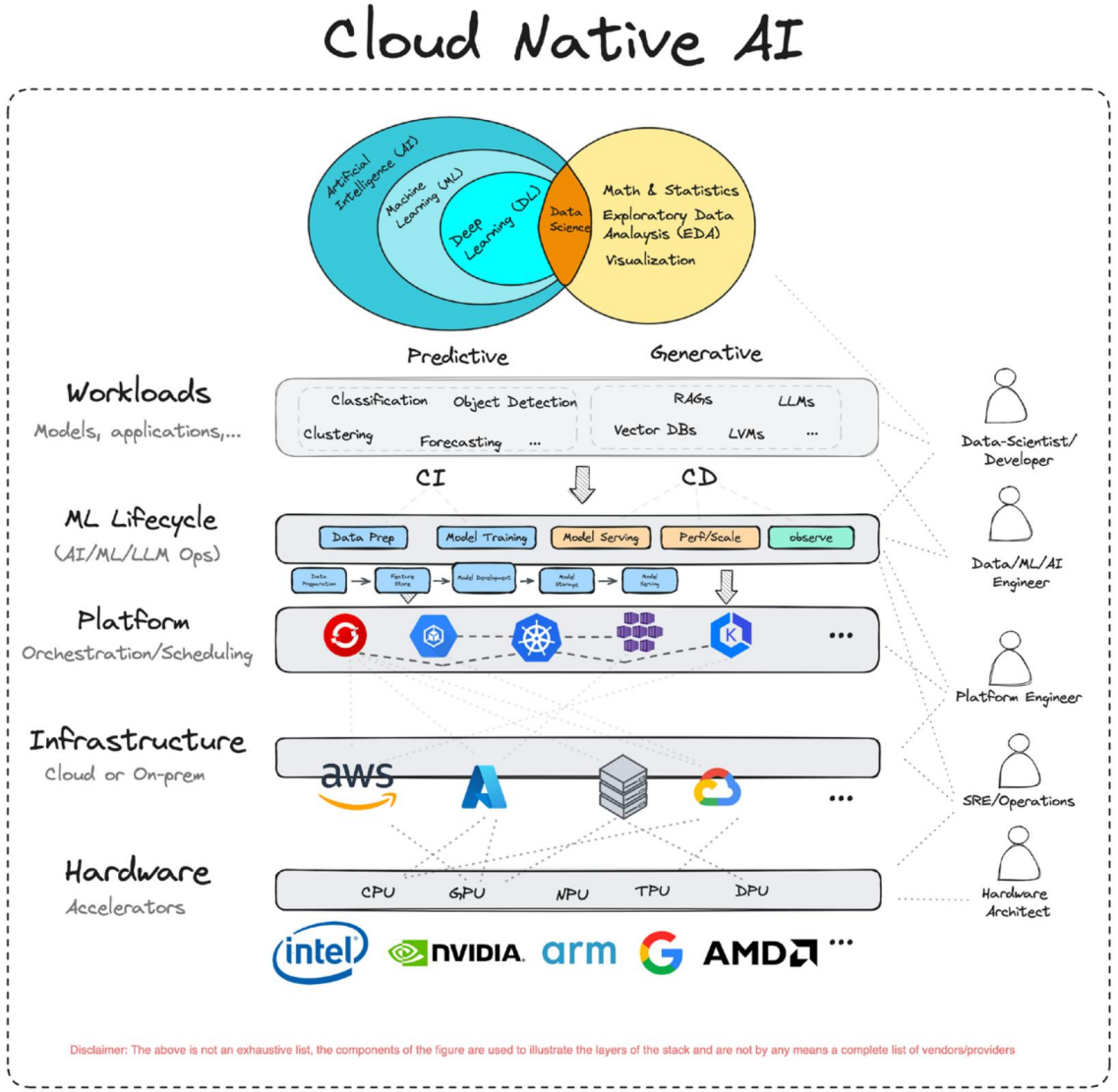
AI + 云原生技术趋势



https://lifearchitect.ai/timeline/
https://nhlocal.github.io/AiTimeline/

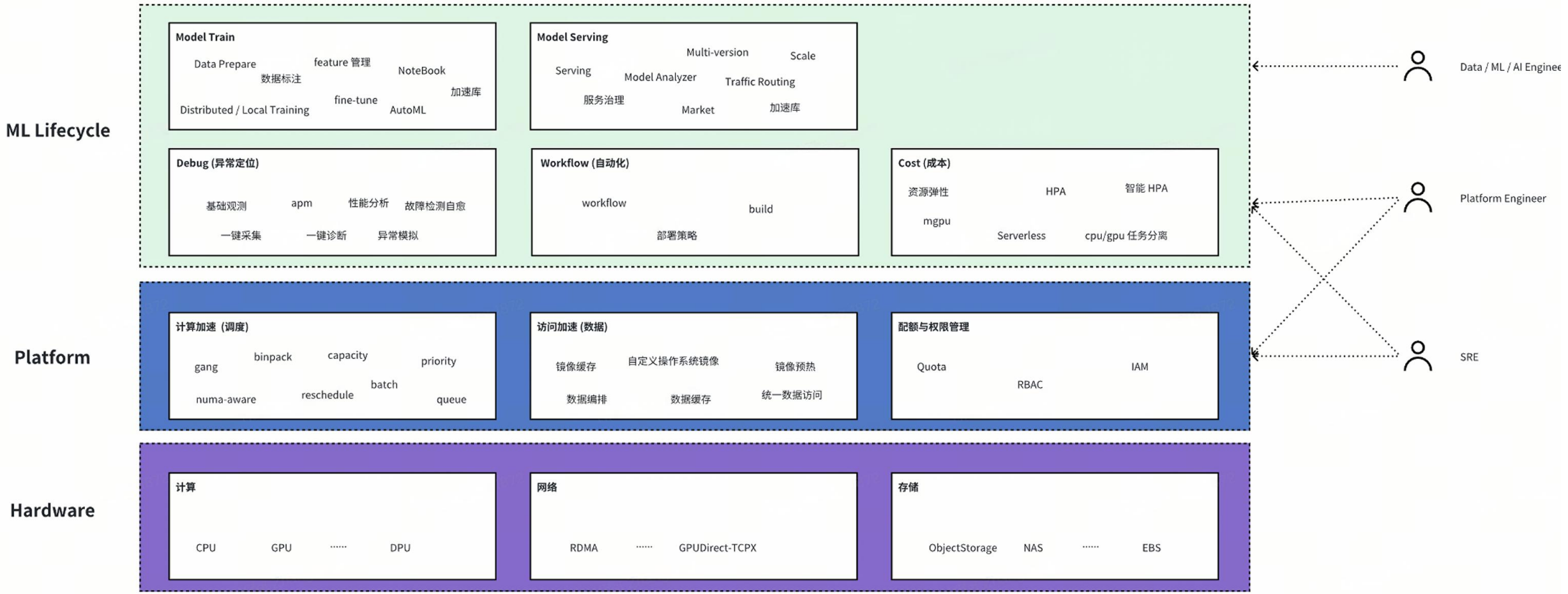
AI + 云原生技术趋势

2024.03

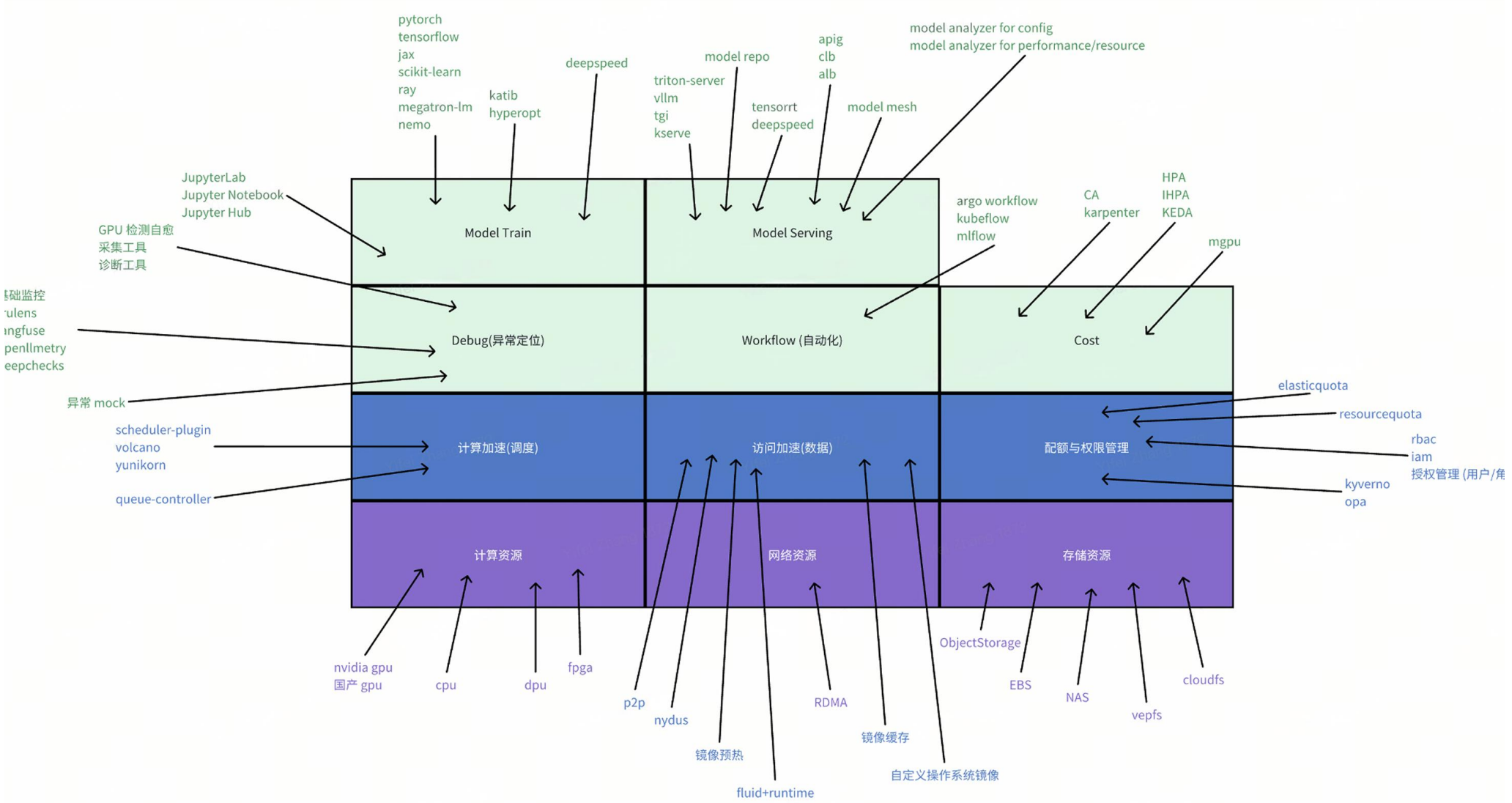


https://www.cncf.io/reports/cloud-native-artificial-intelligence-whitepaper/

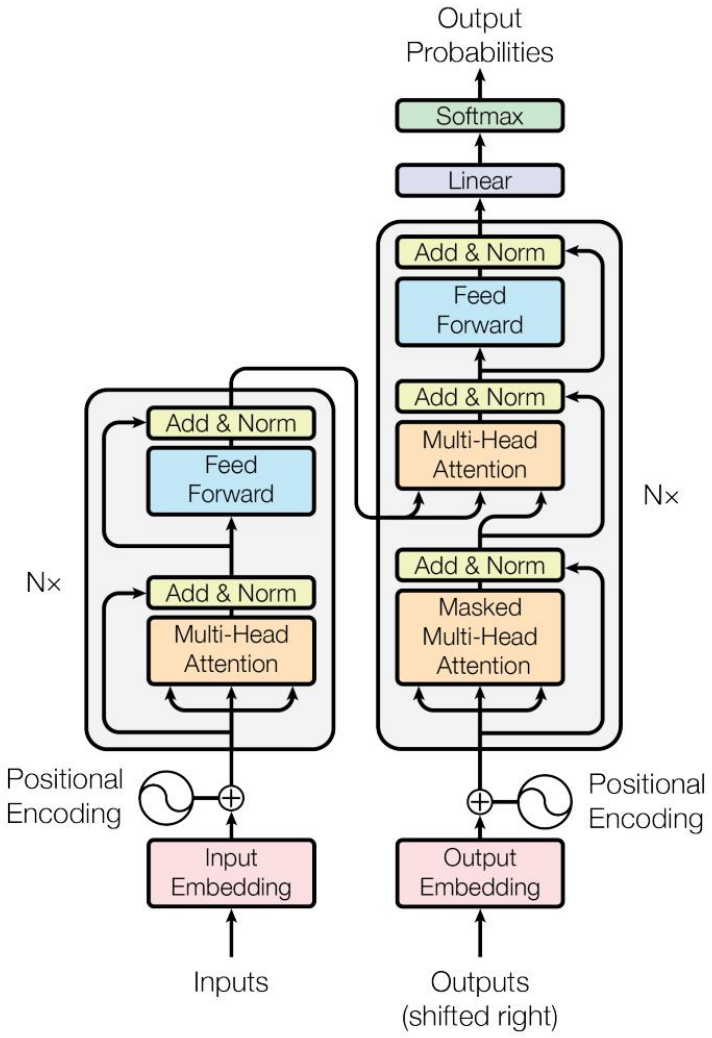
AI + 云原生技术趋势

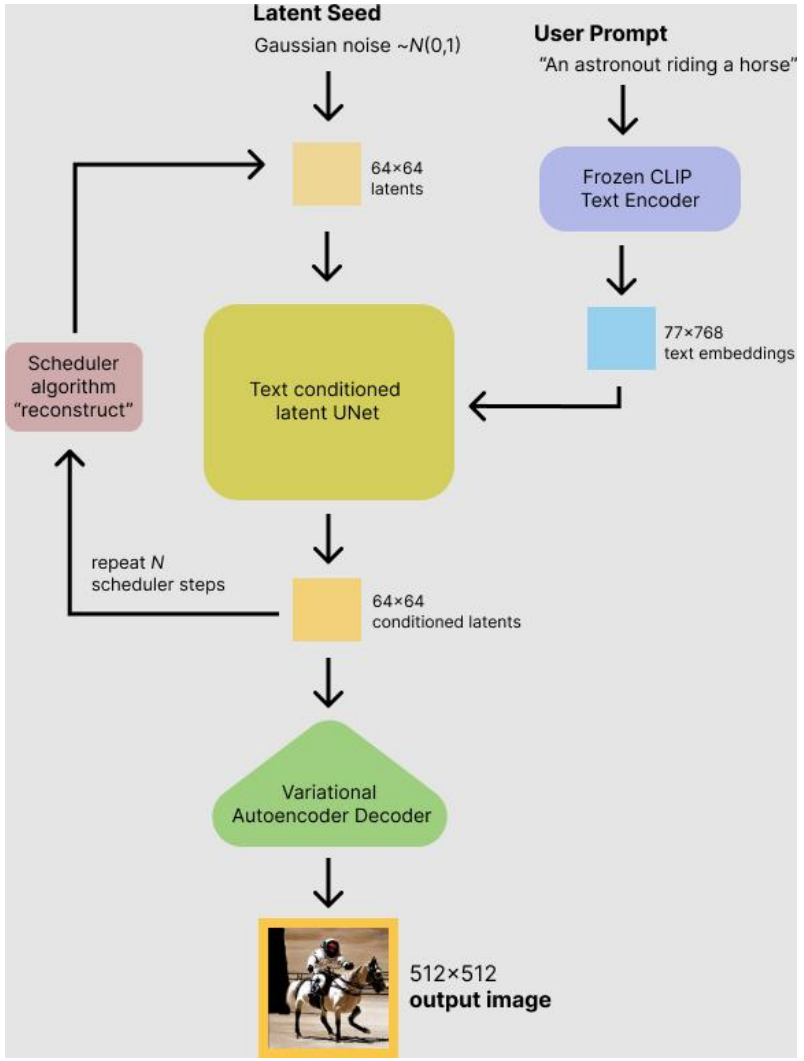

ML Lifecycle
Model Train
Data Prepare
feature 管理
NoteBook
数据标注
加速库
Distributed / Local Training
fine-tune
AutoML
Model Serving
Multi-version
Scale
Serving
Model Analyzer
Traffic Routing
服务治理
Market
加速库
Debug (异常定位)
基础观测
apm
性能分析
故障检测自愈
一键采集
一键诊断
异常模拟
Workflow (自动化)
workflow
build
部署策略
Cost (成本)
资源弹性
HPA
智能 HPA
mgpu
Serverless
cpu/gpu 任务分离
Platform
计算加速 (调度)
gang
binpack
capacity
priority
numa-aware
reschedule
batch
queue
访问加速 (数据)
镜像缓存
自定义操作系统镜像
镜像预热
数据编排
数据缓存
统一数据访问
配额与权限管理
Quota
IAM
RBAC
Hardware
计算
CPU
GPU
·····
DPU
网络
RDMA
·····
GPUDirect-TCPX
存储
ObjectStorage
NAS
·····
EBS
Data / ML / AI Enginee
Platform Engineer
SRE

AI + 云原生技术趋势


pytorch
tensorflow
jax
scikit-learn
ray
megatron-lm
nemo
katib
hyperopt
deepspeed
triton-server
vllm
tgi
kserve
model repo
tensorrt
deepspeed
apig
clb
alb
model mesh
model analyzer for config
model analyzer for performance/resource
JupyterLab
Jupyter Notebook
Jupyter Hub
GPU 检测自愈
采集工具
诊断工具
Model Train
Model Serving
argo workflow
kubeflow
mlflow
CA
karpenter
HPA
IHPA
KEDA
mgpu
基础监控
ulens
ngfuse
penllmetry
eepchecks
Debug(异常定位)
Workflow (自动化)
Cost
异常 mock
elasticquota
resourcequota
rbac
iam
授权管理 (用户/角
scheduler-plugin
volcano
yunikorn
queue-controller
计算加速(调度)
访问加速(数据)
配额与权限管理
kyverno
opa
计算资源
网络资源
存储资源
nvidia gpu
国产 gpu
cpu
dpu
fpga
p2p
nydus
镜像预热
fluid+runtime
RDMA
镜像缓存
自定义操作系统镜像
ObjectStorage
EBS
NAS
vepfs
cloudfs

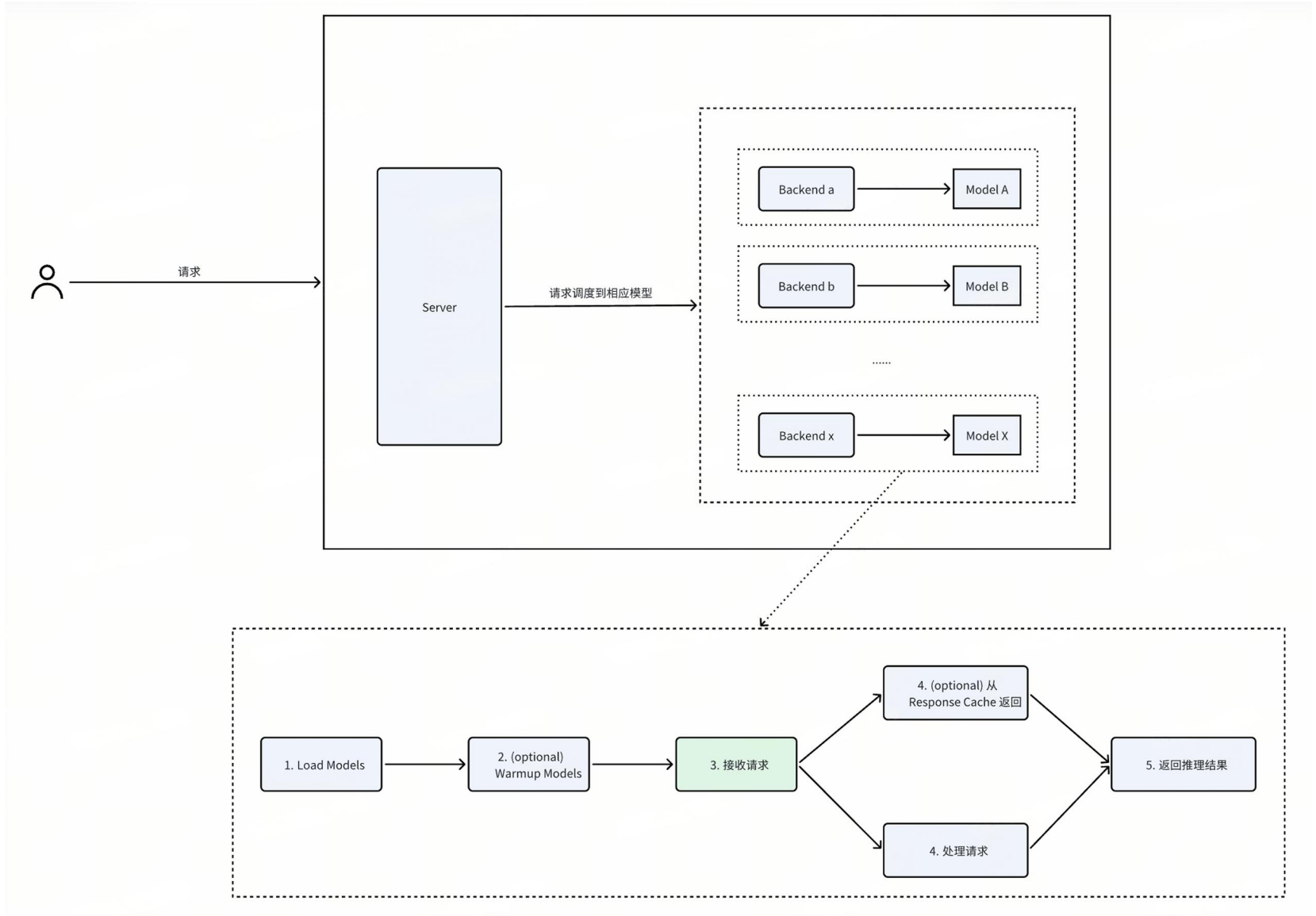
AI 推理





https://arxiv.org/abs/1706.03762
https://huggingface.co/blog/stable_diffusion

AI 推理


请求
Server
请求调度到相应模型
Backend a
Model A
Backend b
Model B
.....
Backend x
Model X
1. Load Models
2. (optional) Warmup Models
3. 接收请求
4. (optional) 从 Response Cache 返回
4. 处理请求
5. 返回推理结果

AI 推理

Methods	Description
Batching	Batching is one of the most effective ways to improve throughput for models that support it.
TensorRT / TensorRT-LLM Optimization	Converting models to TensorRT / TensortRT-LLM format can significantly improve performance.
Model Instances	Creating multiple instances of the same model allows concurrent execution and increases throughput, especially for real-time inferencing.
Response Caching	Enabling response caching at the server or model level can reduce latency by storing and retrieving responses for repeated requests.
Model Warmup	Avoid slow initial inference requests due to deferred initialization of models.
Quantization	Quantization techniques focus on representing data with less information while also trying to not lose too much accuracy. This often means converting a data type to represent the same information with fewer bits. Lower precision can also speed up inference because it takes less time to perform calculations with fewer bits.
Torch compile	`torch.compile` is a feature introduced in PyTorch 2.0 designed to optimize PyTorch code by JIT-compiling it into highly efficient kernels. This process aims to enhance the performance of PyTorch models with minimal code changes.
Distributed KV Cache	A distributed key-value (KV) cache in AI inference is a system designed to store and manage intermediate computational results across multiple nodes or servers to optimize the performance and efficiency of AI models, particularly large language models (LLMs). This technique is crucial for handling the high computational demands and large memory requirements of modern AI models.
Storage Access Acceleration	Optimize the performance of remotely pulling the model.

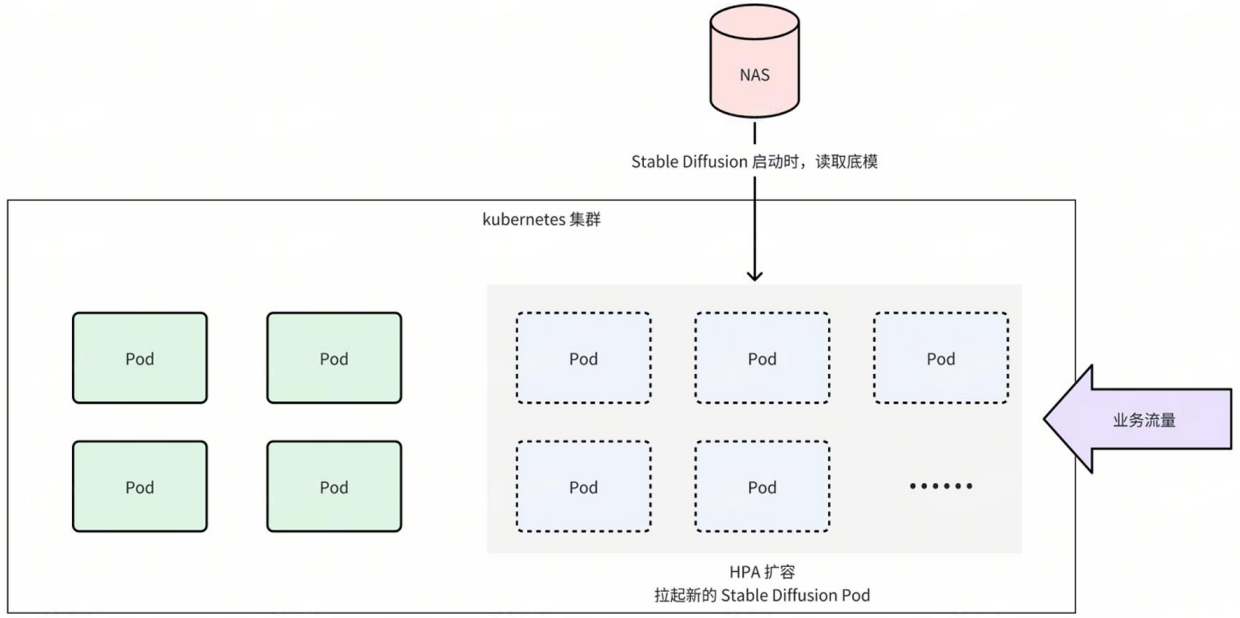
存储访问加速 --- 客户场景 & 问题

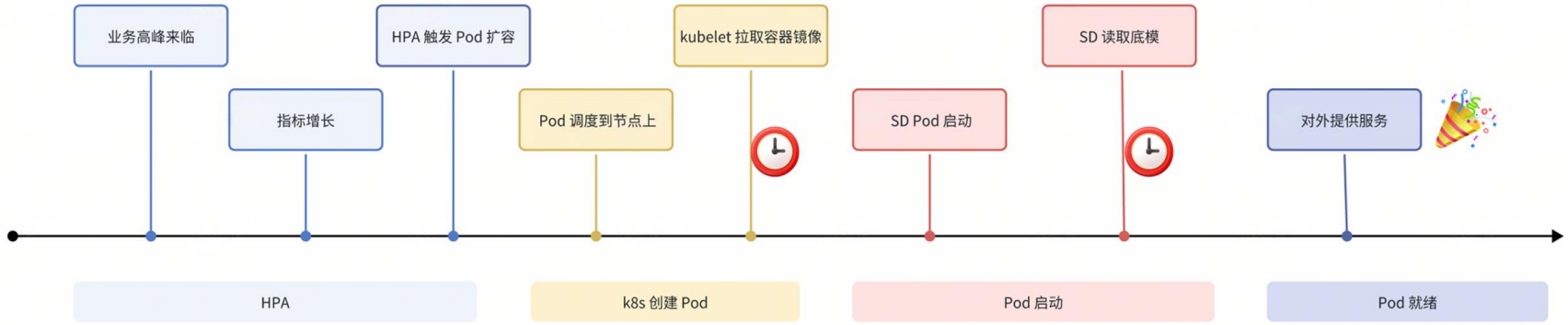
客户场景：
Serverless Pod 启动 Stable Diffusion 服务

问题：
弹性速度慢

耗时点：
- 拉取容器镜像
- Stable Diffusion 启动时读取底模

解法：
- 镜像缓存
- Fluid + Alluxio

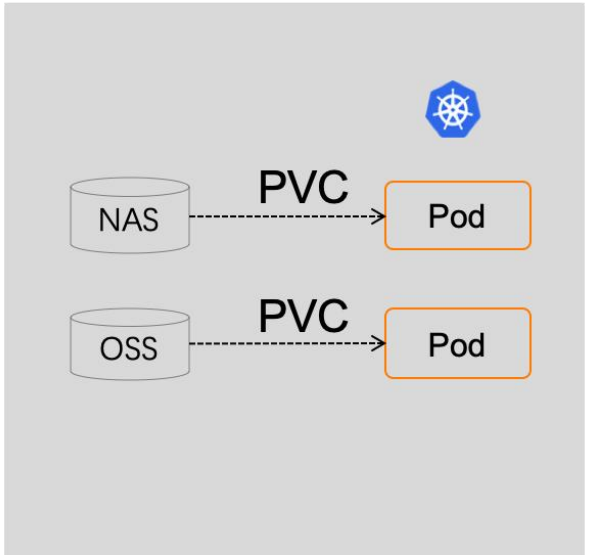




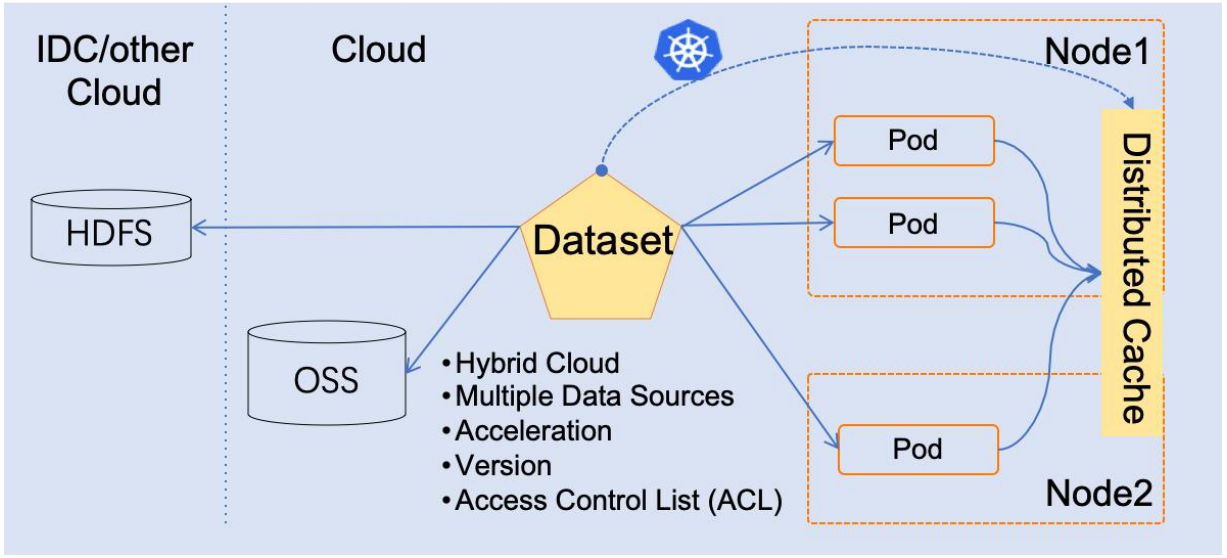
存储访问加速 --- Fluid 介绍

- Kubernetes-native
- 管理数据集（编排、加速）
- 支持多种存储实现
- CNCF sandbox 开源项目

The perspective of CSI

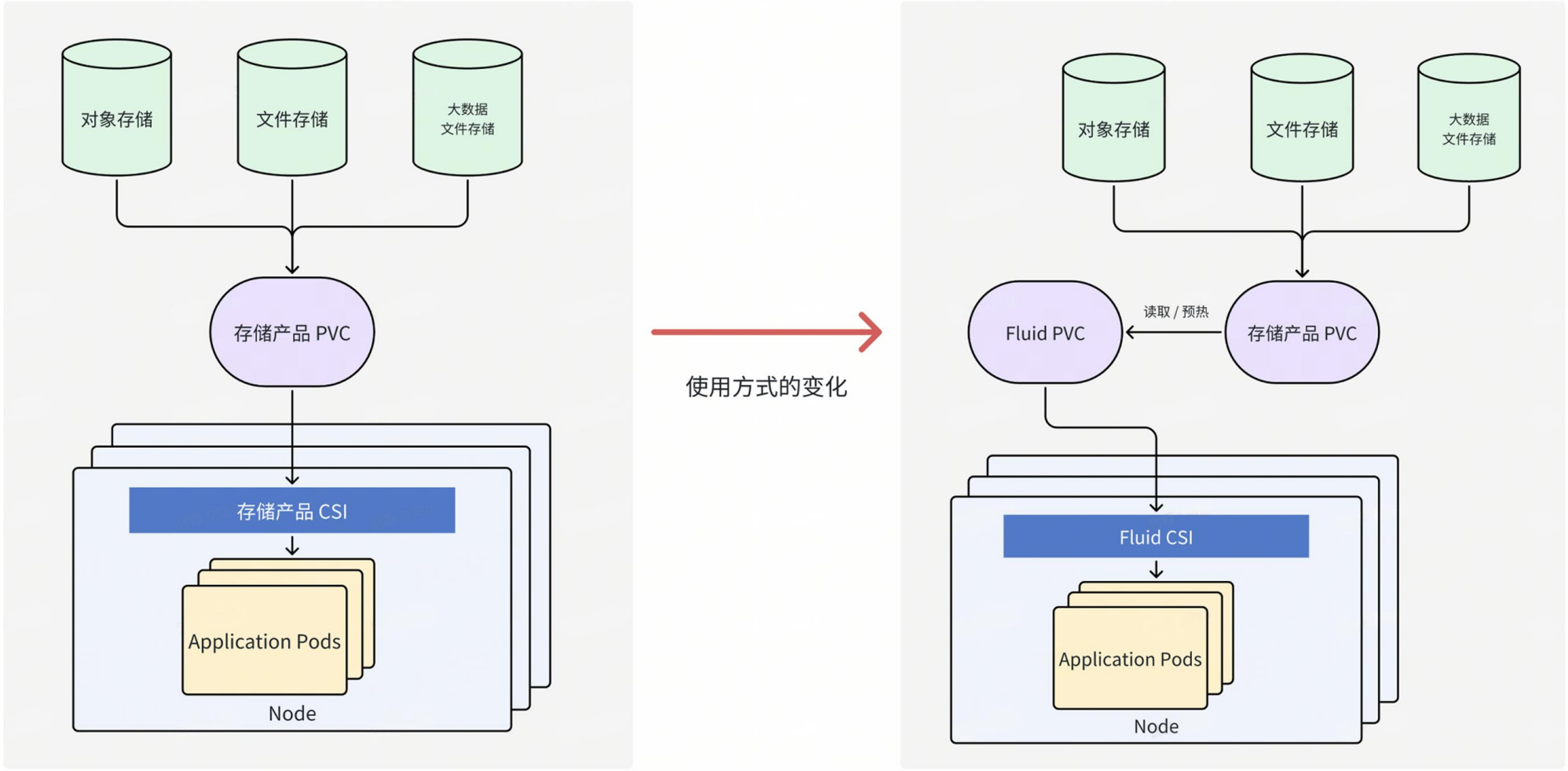


The data usage perspective of Fluid



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

存储访问加速 --- 使用方式

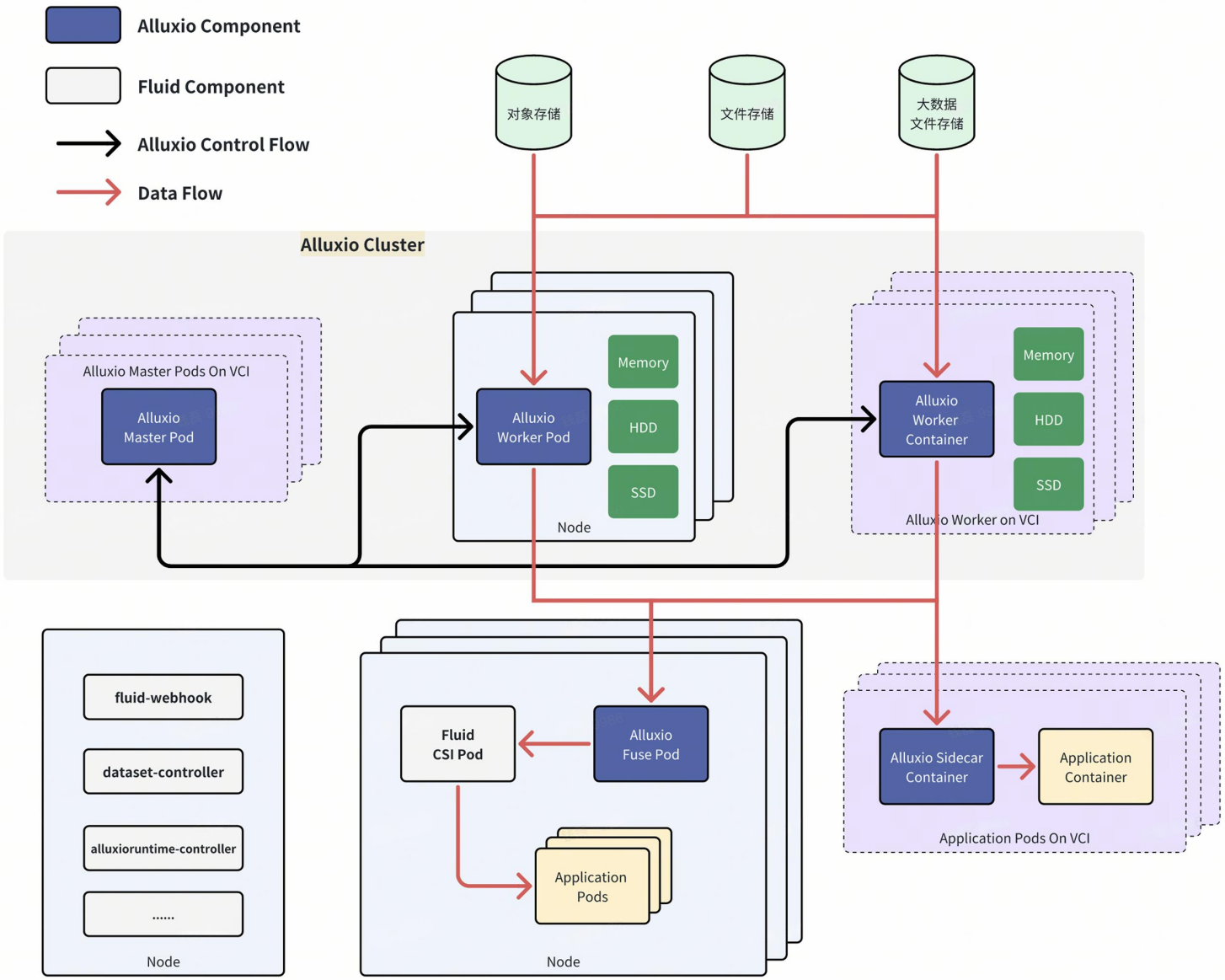

对象存储
文件存储
大数据
文件存储
存储产品 PVC
存储产品 CSI
Application Pods
Node
使用方式的变化
对象存储
文件存储
大数据
文件存储
Fluid PVC
读取 / 预热
存储产品 PVC
Fluid CSI
Application Pods
Node

存储访问加速 --- 整体部署

Fluid 组件：
Node / Serverless Pod 部署皆可

Alluxio Master Pod：
Serverless Pod 部署，避免 Node 抖动的影响

Alluxio Worker Pod：
Node 方式部署，复用已有的 Node，提升利用率
弹性能力，忙时扩容提高性能，闲时缩容省成本



存储访问加速 --- 性能 & 效果

首次文生图请求，所有 **Pod** 的完成时间大幅提升

场景	P90 耗时 （相比直连 NAS 的提升）
直连 NAS	-
3 个 Alluxio Worker Pod，提前预热	**284%**
6 个 Alluxio Worker Pod，提前预热	**480%**
6 个 Alluxio Worker Pod，不提前预热	**305%**

What's next:

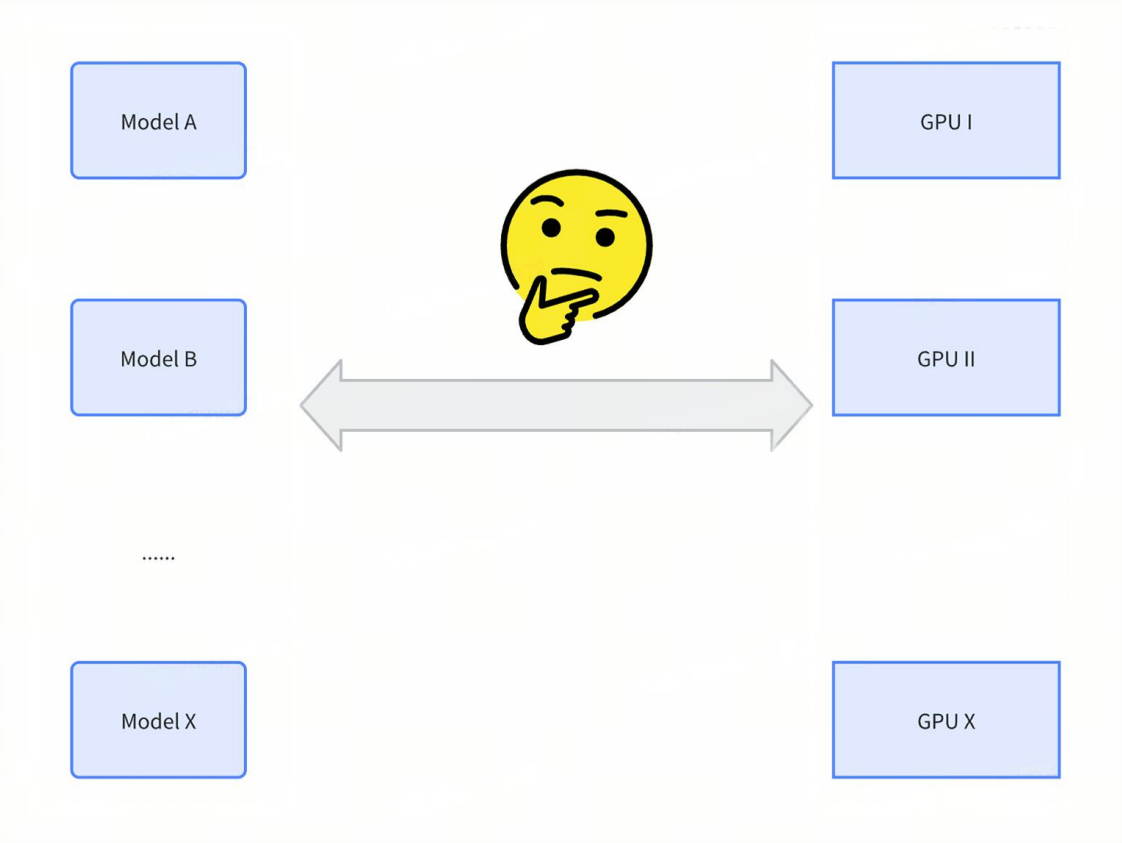
- 更快、更稳定的 runtime
- 集成 veTurboIO，开源、高性能读写模型文件的 Python 库
- 产品化，更易用
- 更大规模的验证

性能评测工具

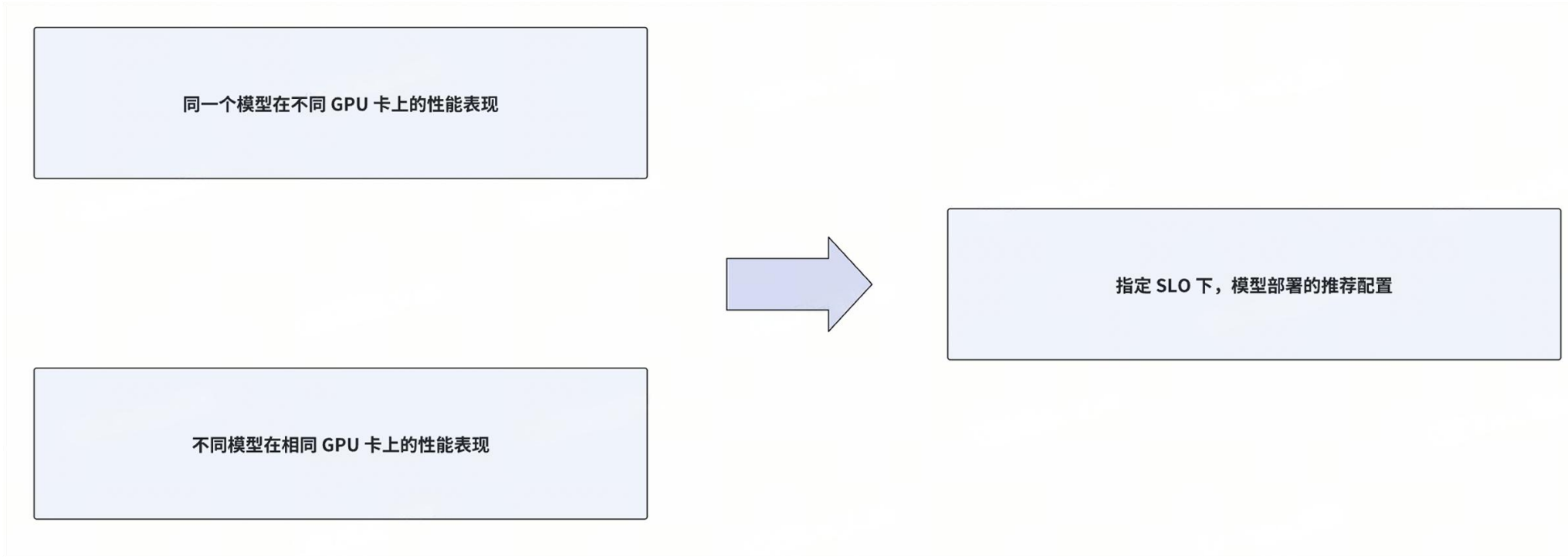
客户场景：
为不同的模型选择合适的卡推理

问题：
模型的性能如何？
不同的卡适合运行哪种模型？
如何为模型选择最佳的部署配置？

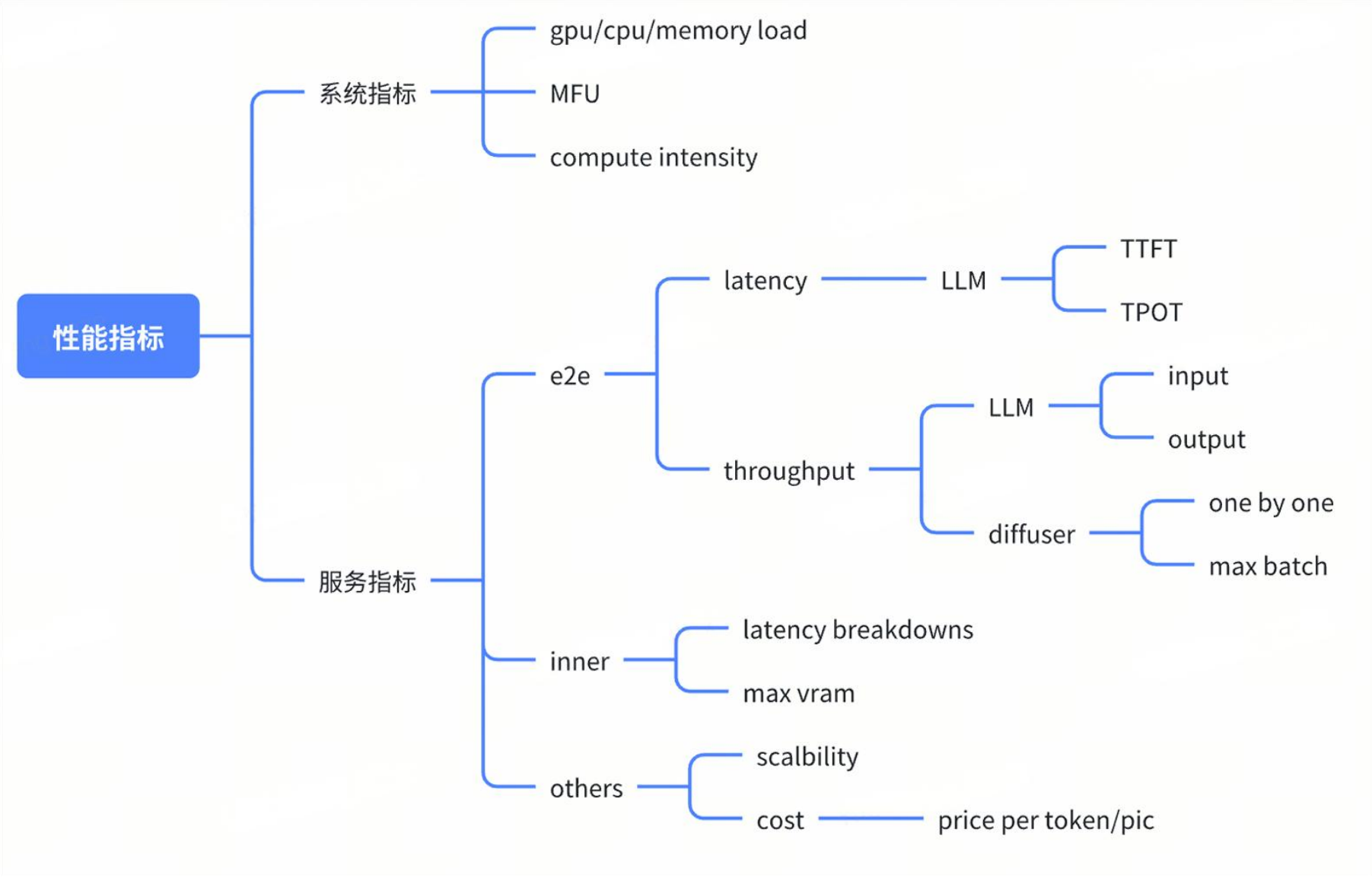
解法：
- 统一性能评测工具
- 部署配置推荐



性能评测工具

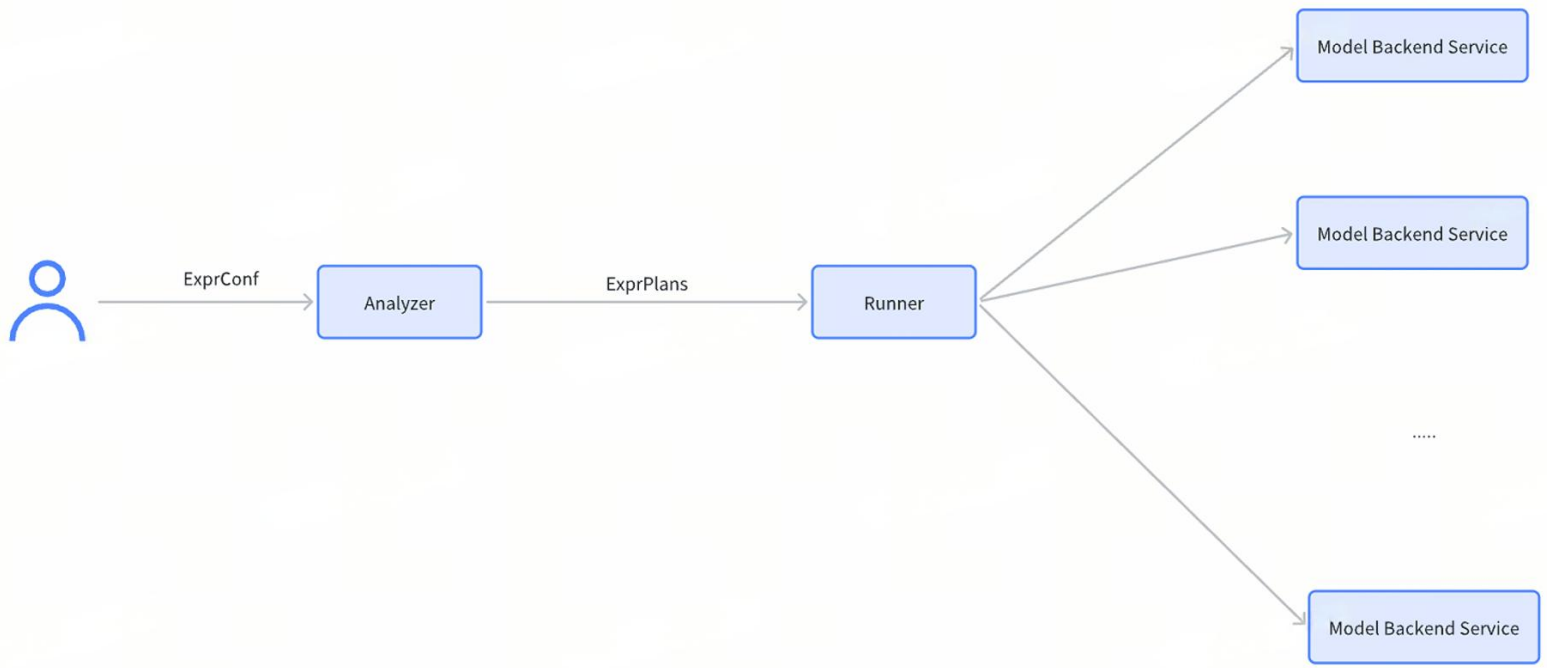

同一个模型在不同 GPU 卡上的性能表现
不同模型在相同 GPU 卡上的性能表现
指定 SLO 下，模型部署的推荐配置

性能评测工具


性能指标
系统指标
gpu/cpu/memory load
MFU
compute intensity
服务指标
e2e
latency
LLM
TTFT
TPOT
throughput
LLM
input
output
diffuser
one by one
max batch
inner
latency breakdowns
max vram
others
scalbility
cost
price per token/pic

性能评测工具


```yaml
objectives:
 - MaxThroughput

backends:
 - name: VLLMOpenAIOnline
   models:
   - llama3
   task: TextGeneration
   environment_parameters:
     - name: DeviceIds
       value: [all]
   parameters:
   - name: TensorParallelSize
     mutate_policy: Fixed
     value: [1,2,4,8]
   - name: max_num_seqs
     mutate_policy: Fixed
     value: [128, 256, 512]
 - name: HuggingfaceTgi
   models:
   - llama3
   task: TextGeneration
   parameters:
   - name: TensorParallelSize
     mutate_policy: Fixed
     value: [1,2,4,8]

scenarios:
 - total_requests: 300
   warmup_requests: 30
   request_profile:
     distribution: Poisson
     arrival_rate: 15
     parameters:
     - name: MaxPromptToken
       mutate_policy: Fixed
       value: [100,1000]
     - name: PromptDistribution
       value: [Gamma]
     - name: GammaShape
       value: [1]
     - name: GammaScale
       value: [200]
```

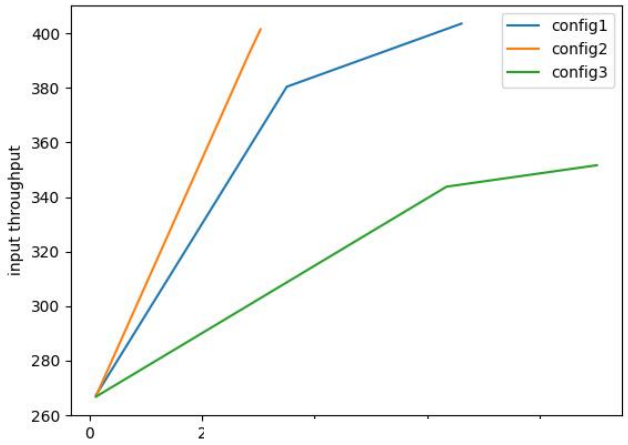


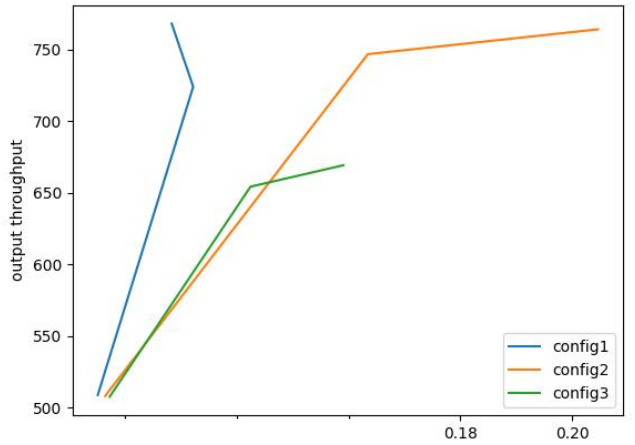
性能评测工具

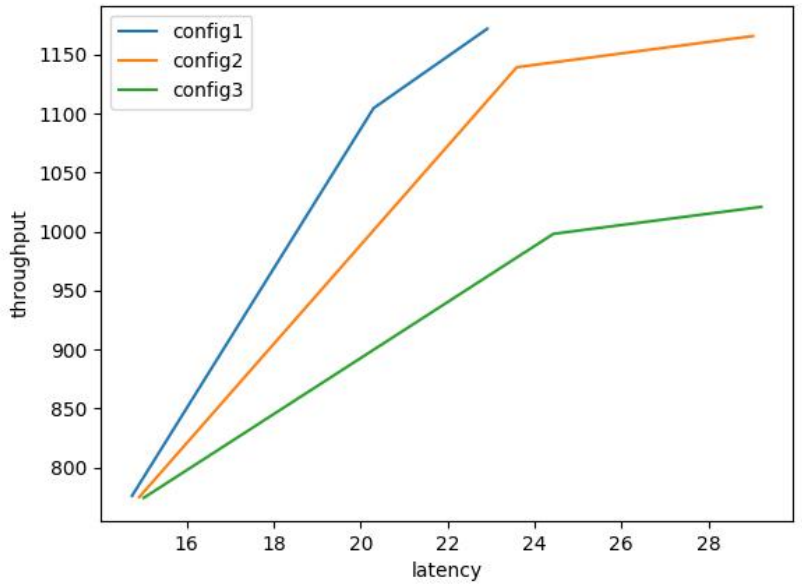

```
------
Result for ExprPlan 1
------
TTFT
count: 500
mean: 1.825434488832 s
stddev: 0.14498870880203713
p99: 1.968870144 s
p95: 1.968870144 s
p90: 1.968870144 s
p50: 1.95446656 s

TPOT
count: 500
mean: 0.09616399742588977 s
stddev: 0.00122213186684615
p99: 0.09747481725648295 s
p95: 0.09747417327874015 s
p90: 0.09747354984146982 s
p50: 0.09690578746456693 s

JCT
count: 500
mean: 14.03826216192 s
stddev: 0.29742351646087456
p99: 14.348171935573333 s
p95: 14.348090150400001 s
p90: 14.348010973866666 s
p50: 14.261501568 s

tokenize time
count: 500
mean: 0.000138018038 s
stddev: 6.355090792551172e-05
p99: 0.0002505621500000000 7 s
p95: 0.0002294281499999999 4 s
p90: 0.0002130584333333333 2 s
p50: 0.000131658 s

input_throughput
value: 495.2658436064178 token/s

output_throughput
value: 1139.9343303895107 token/s
```

```
------
Result for ExprPlan 2
------
TTFT
count: 500
mean: 1.381456792064 s
stddev: 0.7493186622894569
p99: 2.1823961053866676 s
p95: 2.148490496 s
p90: 2.148490496 s
p50: 1.598062592 s

TPOT
count: 500
mean: 0.17353241527836222 s
stddev: 0.05295802498117623
p99: 0.30187345284367456 s
p95: 0.28689303594330706 s
p90: 0.2686527295832021 s
p50: 0.15085138544881893 s

JCT
count: 500
mean: 23.420073532416 s
stddev: 6.842699711382243
p99: 39.93599110314667 s
p95: 38.0334781568 s
p90: 35.71695924906667 s
p50: 19.246099328 s

tokenize time
count: 500
mean: 0.00013838738 s
stddev: 6.276053674548014e-05
p99: 0.0002643714900000000 43 s
p95: 0.0002275032 s
p90: 0.0002124118333333333 2 s
p50: 0.0001306035 s

input_throughput
value: 457.0821161414881 token/s

output_throughput
value: 1052.048314502454 token/s
```

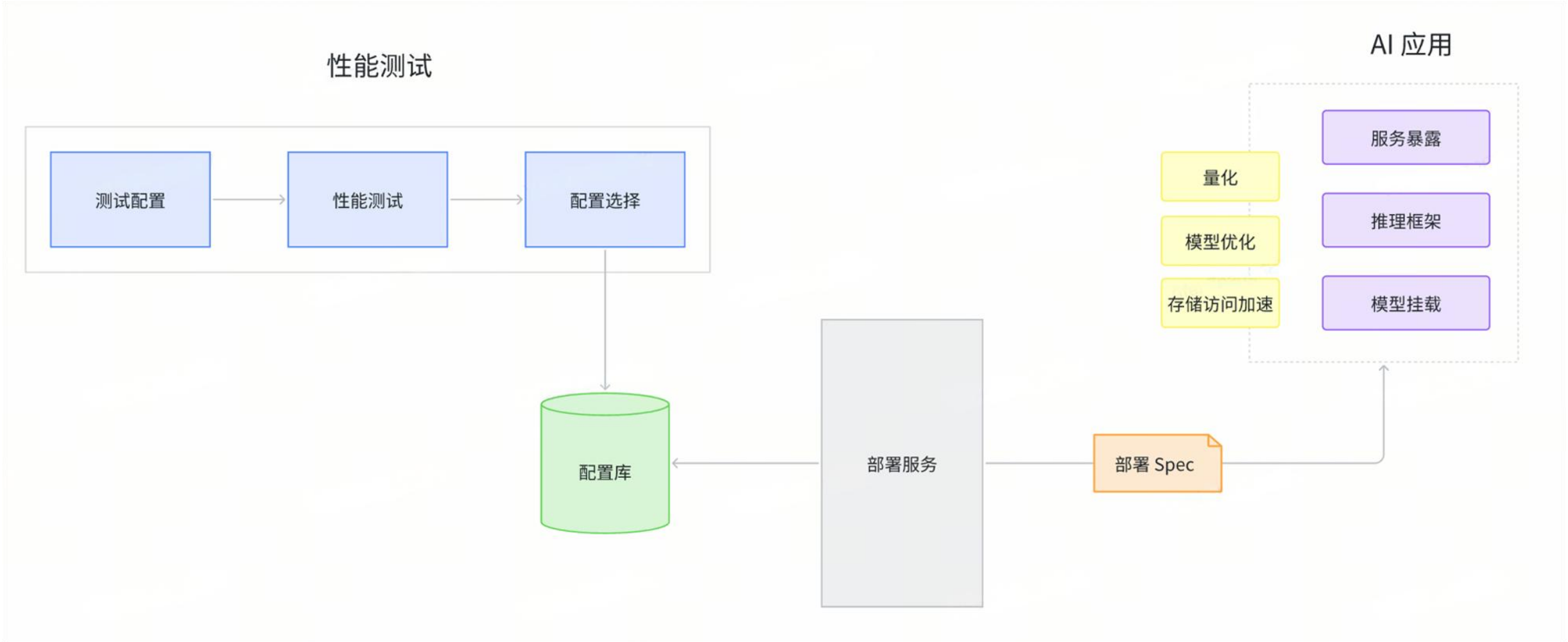
```
------
Result for ExprPlan 3
------
TTFT
count: 500
mean: 0.942095332352 s
stddev: 0.8856858033330711
p99: 2.204497408 s
p95: 2.204497408 s
p90: 2.204497408 s
p50: 0.283148288 s

TPOT
count: 500
mean: 0.17581645832869292 s
stddev: 0.05930062742010439
p99: 0.3337383068690814 s
p95: 0.30376179885354326 s
p90: 0.2549980297070866 s
p50: 0.15912877152755905 s

JCT
count: 500
mean: 23.270785540096 s
stddev: 7.637014045701006
p99: 43.68257130837334 s
p95: 39.875554790399995 s
p90: 33.6825561088 s
p50: 21.42635904 s

tokenize time
count: 500
mean: 0.000138566612 s
stddev: 6.550952676029779e-05
p99: 0.0002682519900000000 1 s
p95: 0.00022738425 s
p90: 0.0002115468333333333 3 s
p50: 0.0001299945 s

input_throughput
value: 352.14318735801106 token/s

output_throughput
value: 810.5144210211 token/s
```







部署工作流


性能测试
测试配置
性能测试
配置选择
AI 应用
量化
模型优化
存储访问加速
服务暴露
推理框架
模型挂载
配置库
部署服务
部署 Spec

谢谢!

欢迎加入火山引擎云原生 AI 套件用户群

AI 火山引擎云原生 AI ...
ByteDance

扫描群二维码，立刻加入该群

该二维码永久有效