推理系统系列

什么是推理系统

ZOMI

Talk Overview

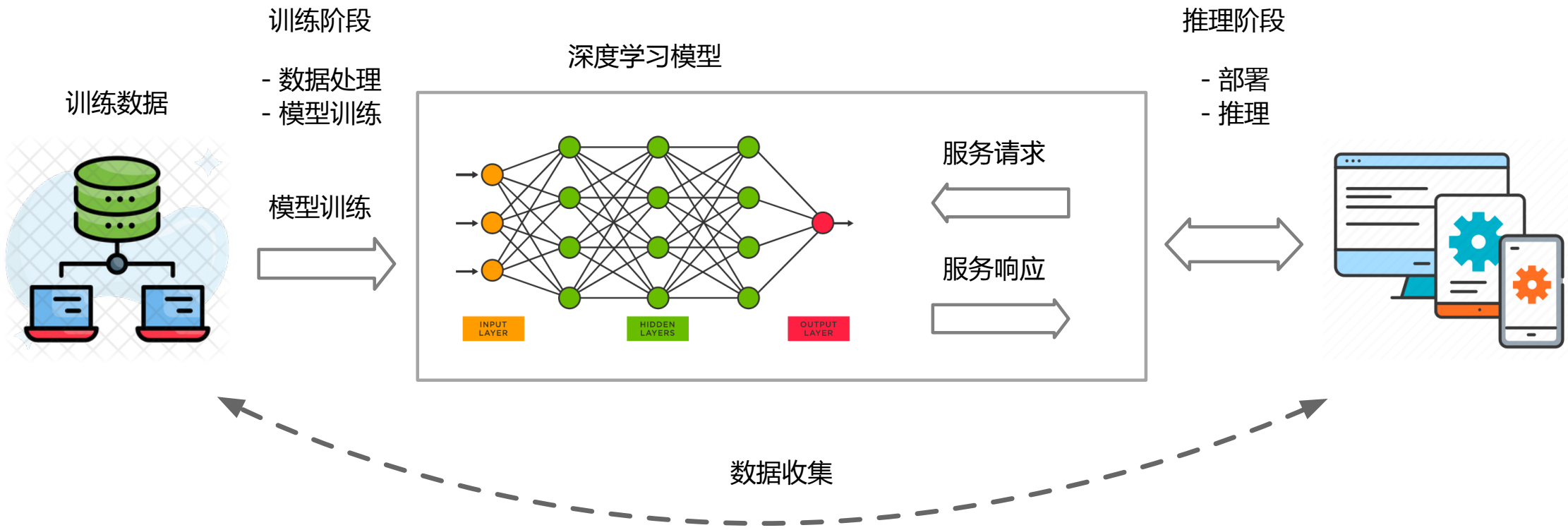
1. **推理系统介绍**
   - 推理系统与推理引擎区别
   - 推理工作流程
   - 推理系统介绍
   - 推理引擎介绍
2. **模型小型化**
   - NAS神经网络搜索
   - CNN小型化结构
   - Transform小型化结构
3. **离线优化压缩**
   - 低比特量化
   - 二值化网络
   - 模型模型剪枝
   - 模型模型蒸馏
4. **部署和运行优化**
   - 图转换优化（算子融合/重排/替换）
   - 并发执行与内存分配
   - 动态batch与bin Packing

Talk Overview

I. 推理系统与推理引擎

- Training and Inference – 训练和推理服务的区别
- What is inference system - 什么是推理系统
- Optimization objectives and constraints - 推理系统优化目标与约束
- Difference bet inference system and engine - 推理系统与推理引擎

深度学习模型的生命周期

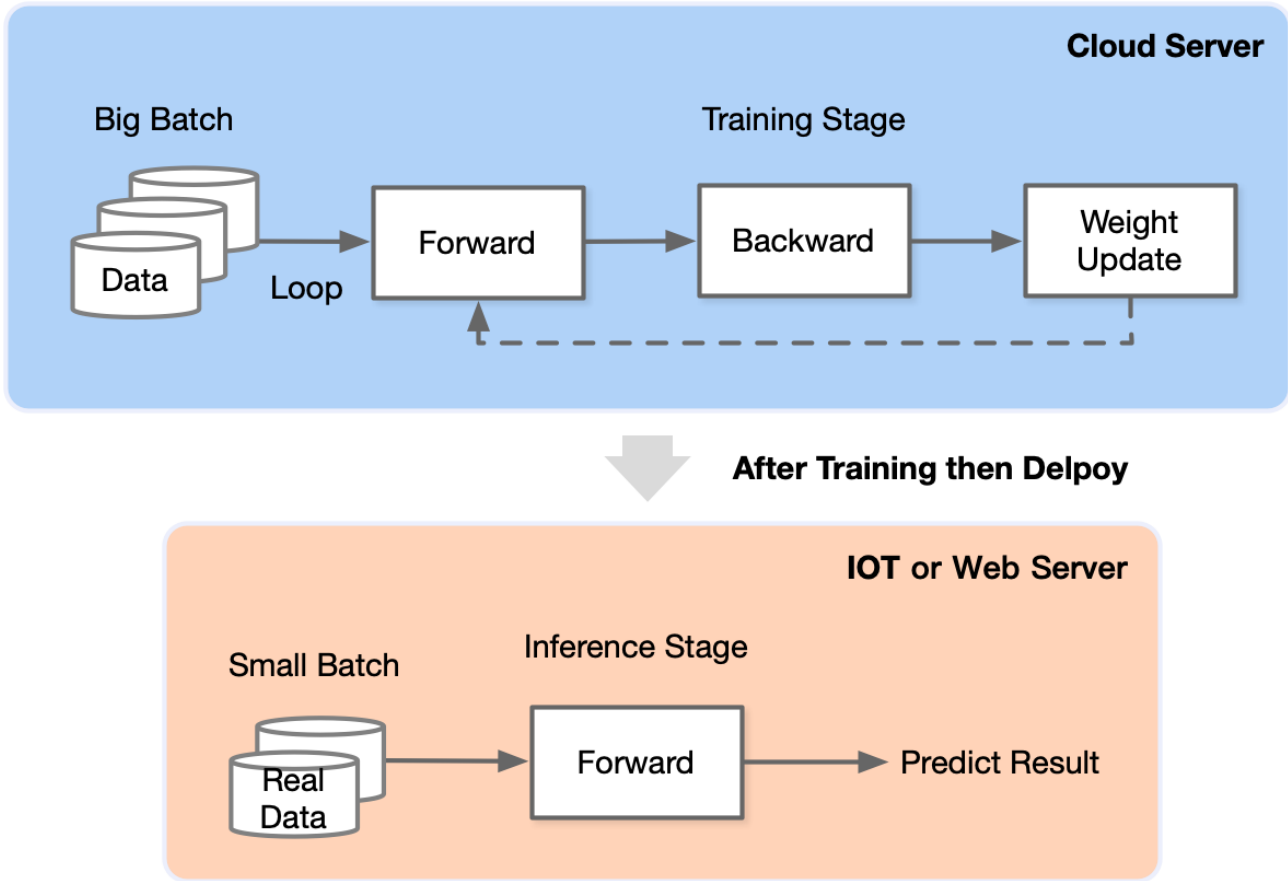


深度学习模型的生命周期

- **训练任务**：数据中心中更像是传统的批处理任务，需要执行数小时，数天才能完成，其一般配置较大的批尺寸追求较大的吞吐，将模型训练达到指定的准确度或错误率。
- **推理任务**：执行 7 X 24 的服务，其常常受到响应延迟的约束，配置的批尺寸更小，模型已经稳定一般不再被训练。

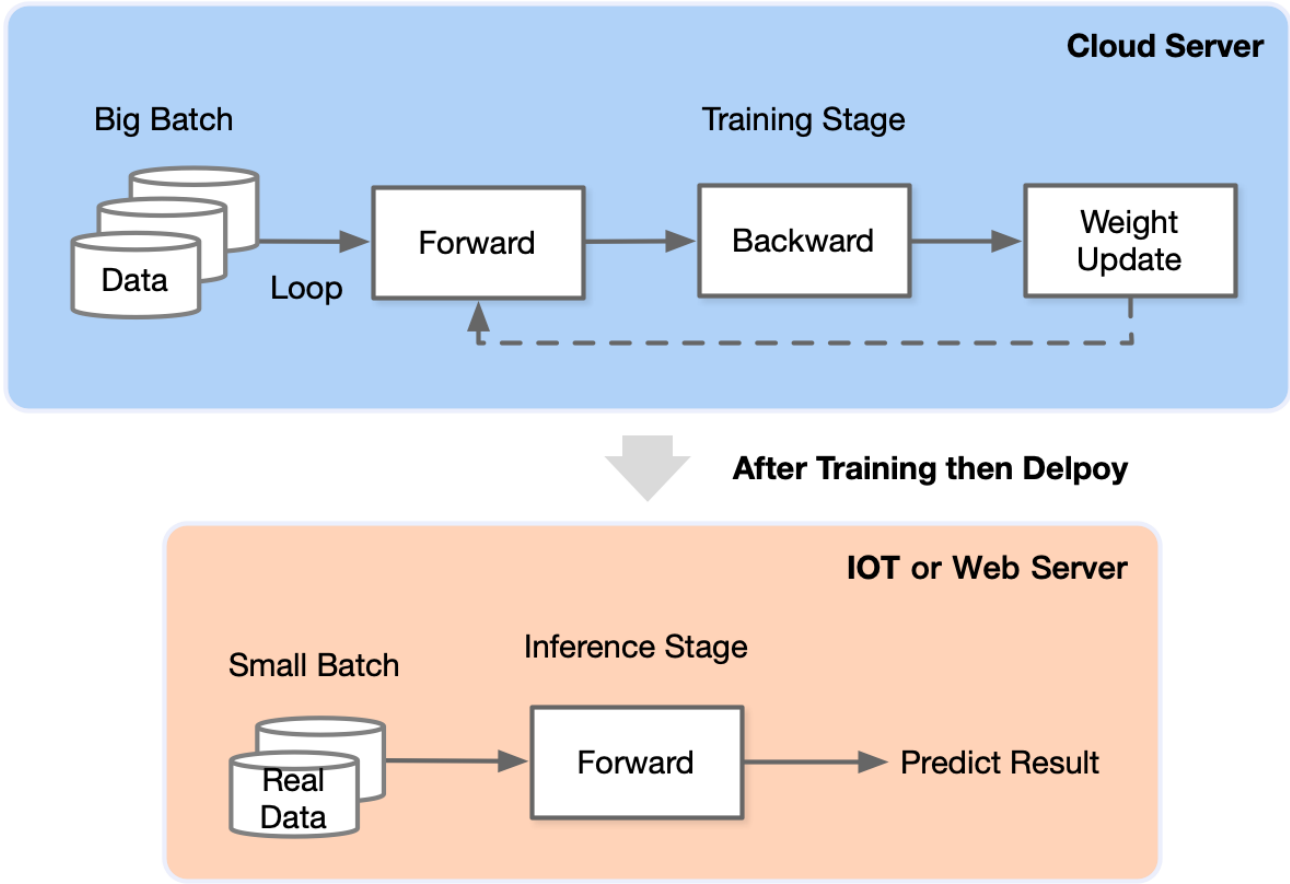
推理相比训练的新特点与挑战

- **训练过程**通过设计合适 AI 模型结构以及损失函数和优化算法，将数据集以 mini-batch 反复进行前向计算并计算损失，反向计算梯度利用优化函数来更新模型，使得损失函数最小。训练过程最重要是梯度计算和反向传播。
- **推理**在训练好的模型结构和参数基础上，一次前向传播得到模型输出过程。相对于训练，推理不涉及梯度和损失优化。最终目标是将训练好的模型部署生产环境中。



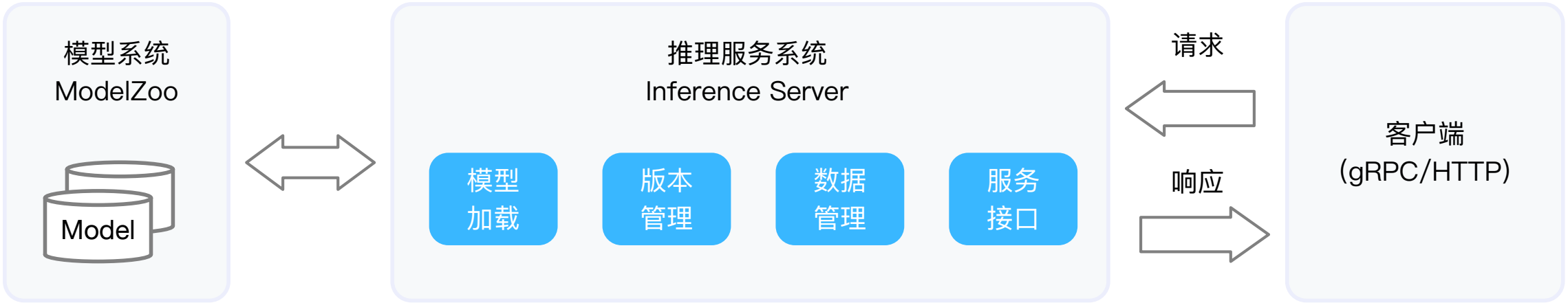
推理相比训练的新特点与挑战

1. 模型被部署为长期运行的服务
2. 推理有更苛刻的资源约束
3. 推理不需要反向传播梯度下降
4. 部署的设备型号更加多样



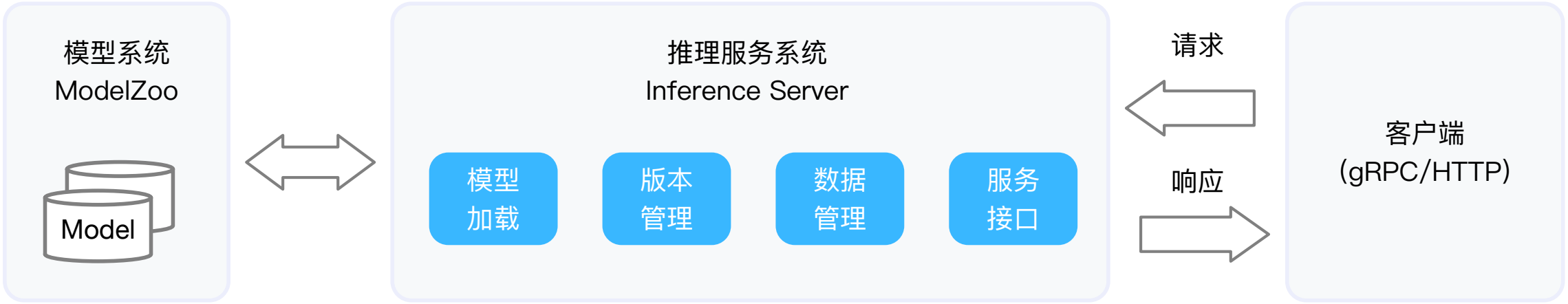
推理系统

- 模型训练后会保存在文件系统中，随着训练处的模型效果不断提升，可能会产生新版本的模型，并存储在文件系统中并由一定的模型版本管理协议进行管理。之后模型会通过服务系统部署上线，推理系统首先会加载模型到内存，同时会对模型进行一定的版本管理，支持新版本上线和旧版本回滚，对输入数据进行批尺寸（Batch Size）动态优化，并提供服务接口（例如，HTTP，gRPC等），供客户端调用。用户不断向推理服务系统发起请求并接受响应。除了被用户直接访问，推理系统也可以作为一个微服务，被数据中心中其他微服务所调用，完成整个请求处理中一个环节的功能与职责。



推理系统

- 推理系统中，以数据中心的服务端推理系统为主，兼顾边缘侧移动端推理的场景，但是这些策略本身大部分是数据中心与边缘侧都适用。



Question?

- 深度学习推理系统设计需要考虑多目标和约束
- 推理系统相比传统服务系统有哪些新的挑战？
- 云和端的服务系统有何不同的侧重和挑战？

推理系统
优化目标与约束

在线推荐系统的服务需求

例如某在线新闻APP公司希望部署内容个性化推荐服务并期望该服务能满足以下需求：

- ✓ 低延迟
  - 互联网上推荐文章延迟（ <100毫秒 ）
- ✓ 高吞吐
  - 突发新闻驱动的暴增人群的吞吐量需求
- ✓ 扩展性
  - 扩展到不断增长的庞大的用户群体
- ✓ 准确度
  - 随着新闻和读者兴趣的变化提供准确的预测

在线推荐系统的服务需求

- 除了应用场景的需求，推理系统也需要应对不同模型训练出的框架和多样性的推理硬件所产生的部署环境多样性，部署优化和维护困难且容易出错的挑战：

设计推理系统的优化目标

- **低延迟(Latency)：**满足服务等级协议的延迟
- **吞吐量(Throughputs)：**暴增负载的吞吐量需求
- **高效率(Efficiency)**：高效率，低功耗使用GPU, CPU
- **灵活性(Flexibility)**：支持多种框架, 提供构建不同应用的灵活性
- **扩展性(Scalability)**：扩展支持不断增长的用户或设备

灵活性 Flexibility

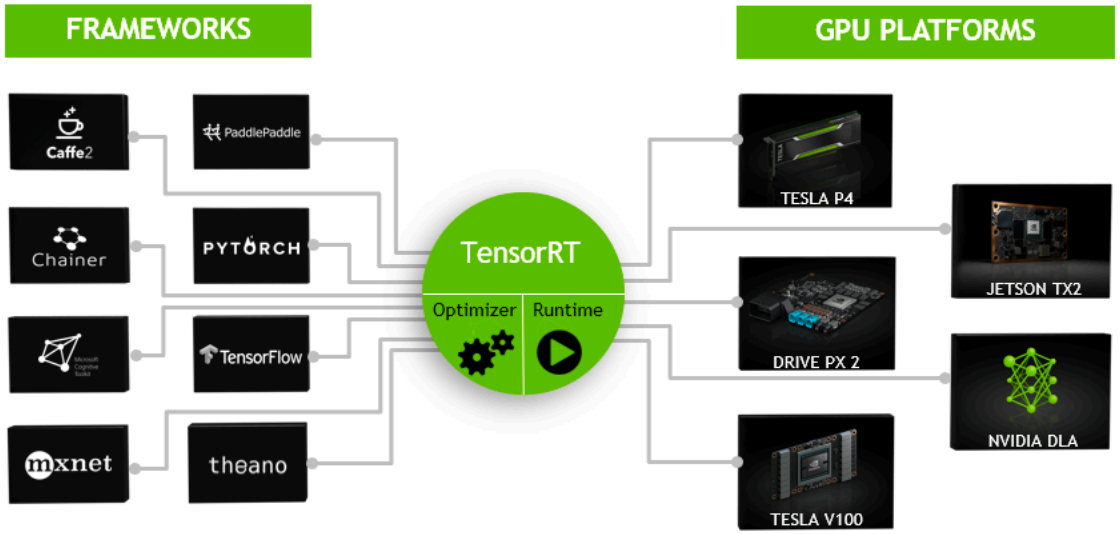
AI 服务的部署，优化和维护困难且容易出错

框架多样：

- 大多数框架都是为训练设计和优化
- 开发人员需要将必要的软件组件拼凑在一起
- 跨多个不断发展的框架集成和推理需求

硬件多样：

- 多种部署硬件的支持



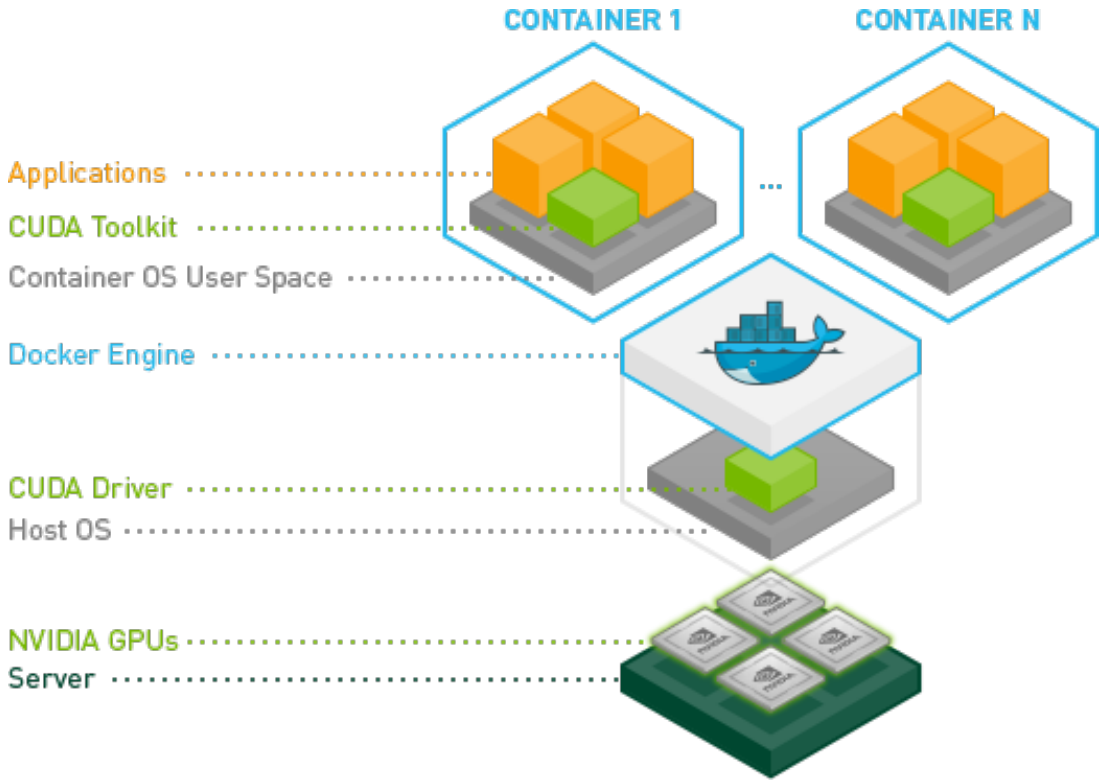
Flexibility 灵活性

服务系统需要灵活性：

- 支持加载不同 AI 框架的模型
- AI 框架推陈出新和版本不断迭代
- 与不同语言接口和不同逻辑的应用结合

解决方法：

- 深度学习模型开放协议：跨框架模型转换
- 接口抽象：提供不同框架的通用抽象
- 容器：运行时环境依赖与资源隔离
- RPC：跨语言，跨进程通信



Latency 延迟

推理延迟：

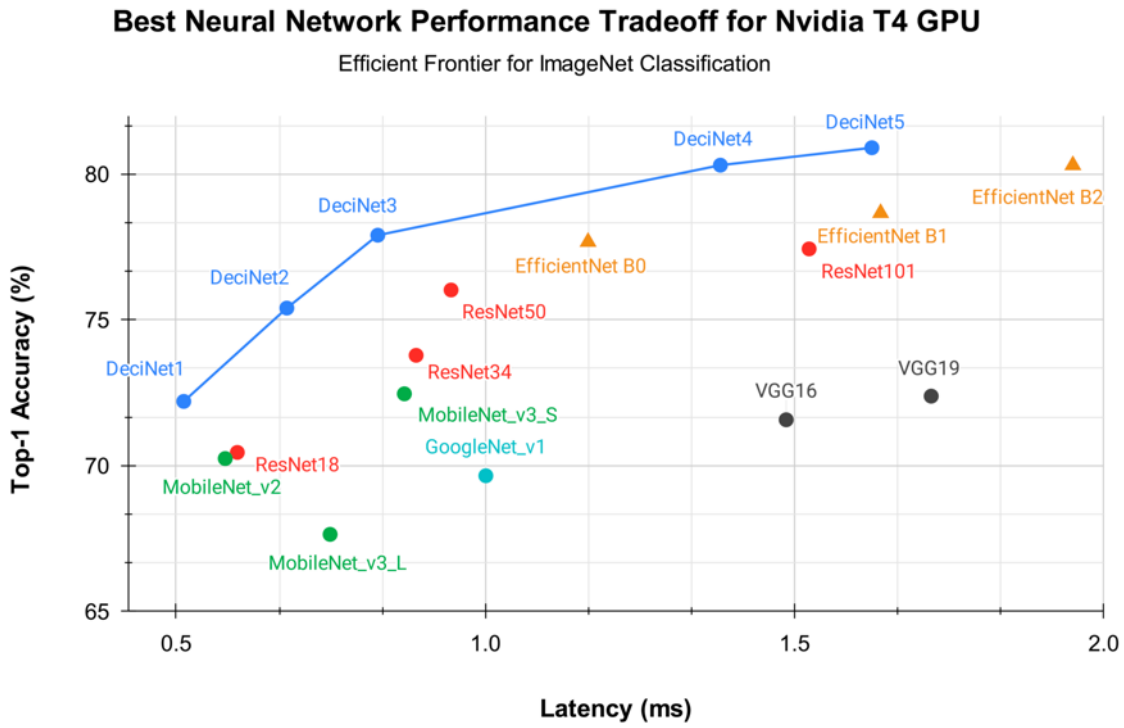
- 延迟是用户给出查询后呈现推理结果消耗的时间
- 必须既快速同时满足有限的尾部延迟 Tail Latency

需要低延迟：

- 服务水平协议(SLA)：Sub-second 级别延迟

低延迟挑战：

- 交互式APP 低延迟需求与训练 AI 框架目标不一致
- 大模型更准确，但浮点运算量更大
- Sub-second 级别延迟约束制数据 Batch Size
- 模型融合容易引起长尾延迟 Long Tail Latency



Throughputs 吞吐量

需要高吞吐的目的：

- 突发的请求数量暴增
- 不断扩展的用户和设备

达到高吞吐的策略：

- 充分利用AI芯片能力1）批处理请求；2）指令级运算
- 支持动态 Shape 和自适应批尺寸 Batch Size
- 多模型装箱使用加速器
- 容器扩展副本部署

Efficiency 效率

需要高效的原因：

- 内存、ALU数量等资源约束
- 移动端有极高的功耗约束
- 云端有预算的约束

高效率策略：

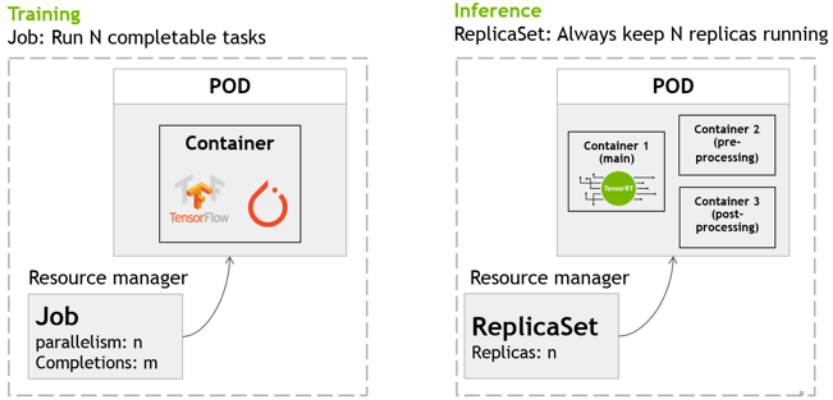
- 模型压缩
- 高效使用 AI 推理芯片
- 装箱（bin-packing）使用加速器

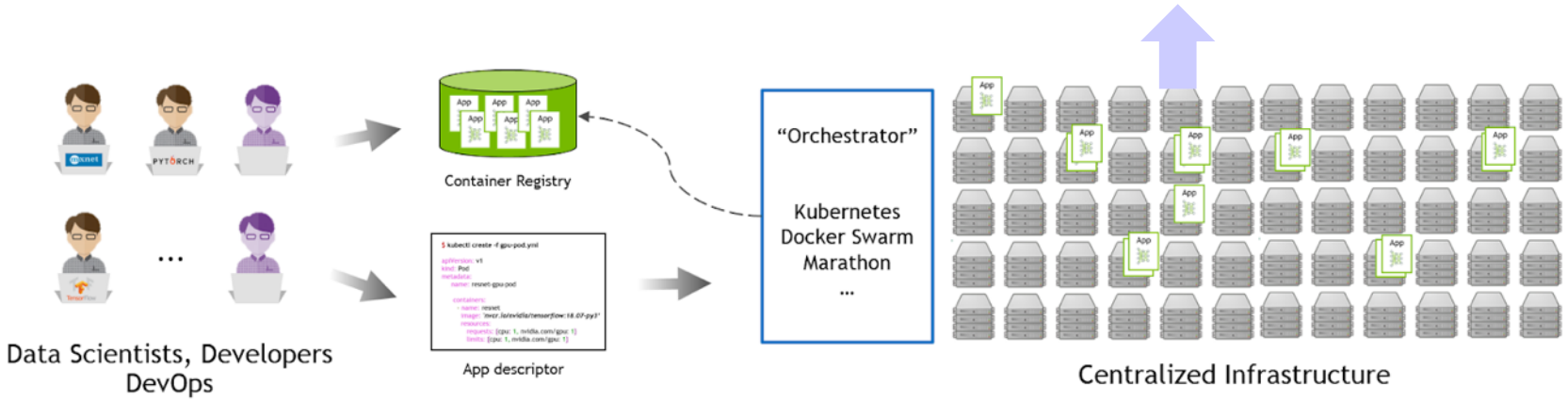
Scalability 扩展性

扩展性原因：

- 应对用户与请求的增长
- 提升推理系统吞吐量

随着请求负载增加自动部署更多的解决方案 ，进而才可以应对更大负载，提供更高的推理吞吐和让推理系统更加可靠。





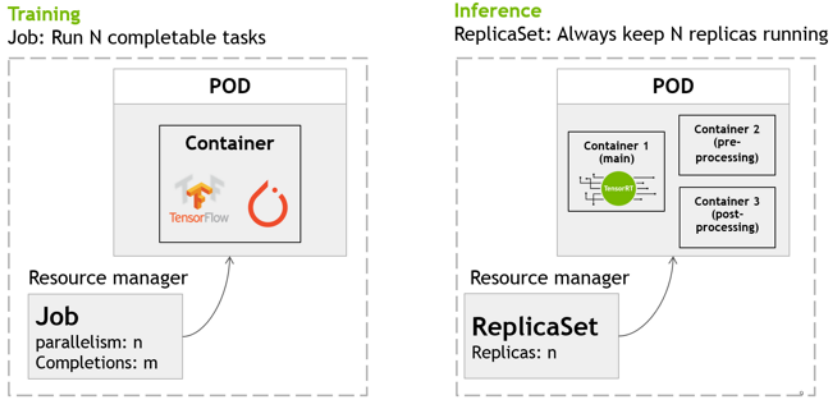
https://devblogs.nvidia.com/kubernetes-ai-hyperparameter-search-experiments/

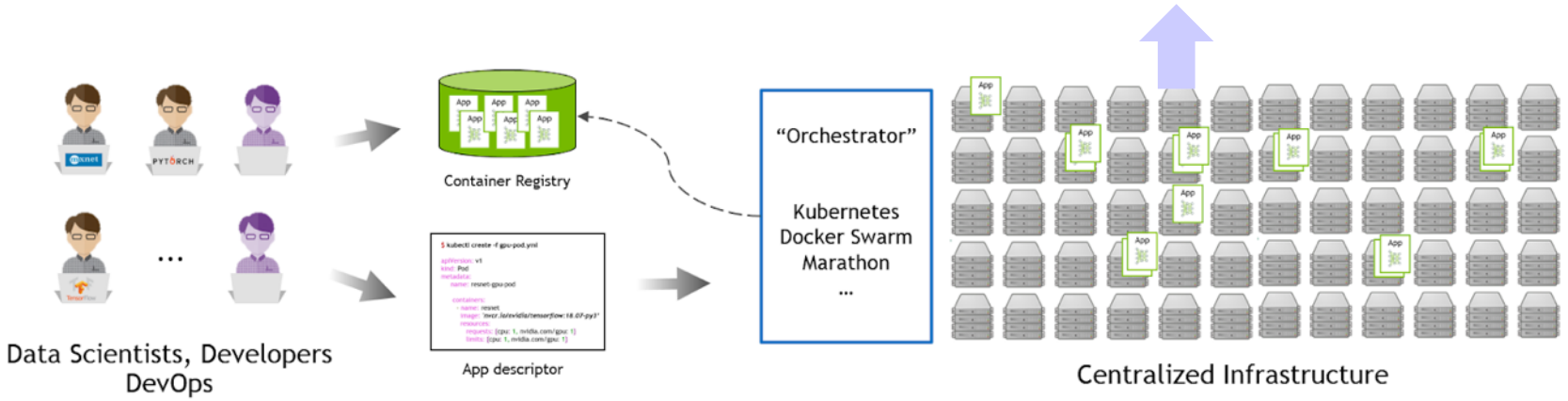
Scalability 扩展性

扩展性原因：

- 应对用户与请求的增长
- 提升推理系统吞吐量

通过底层 Kubernetes 部署平台，用户可以通过配置方便地自动部署多个推理服务的副本，并通过部署前端负载均衡服务，达到高扩展性提升吞吐量。更多副本也使得推理服务有了更高的可靠性。

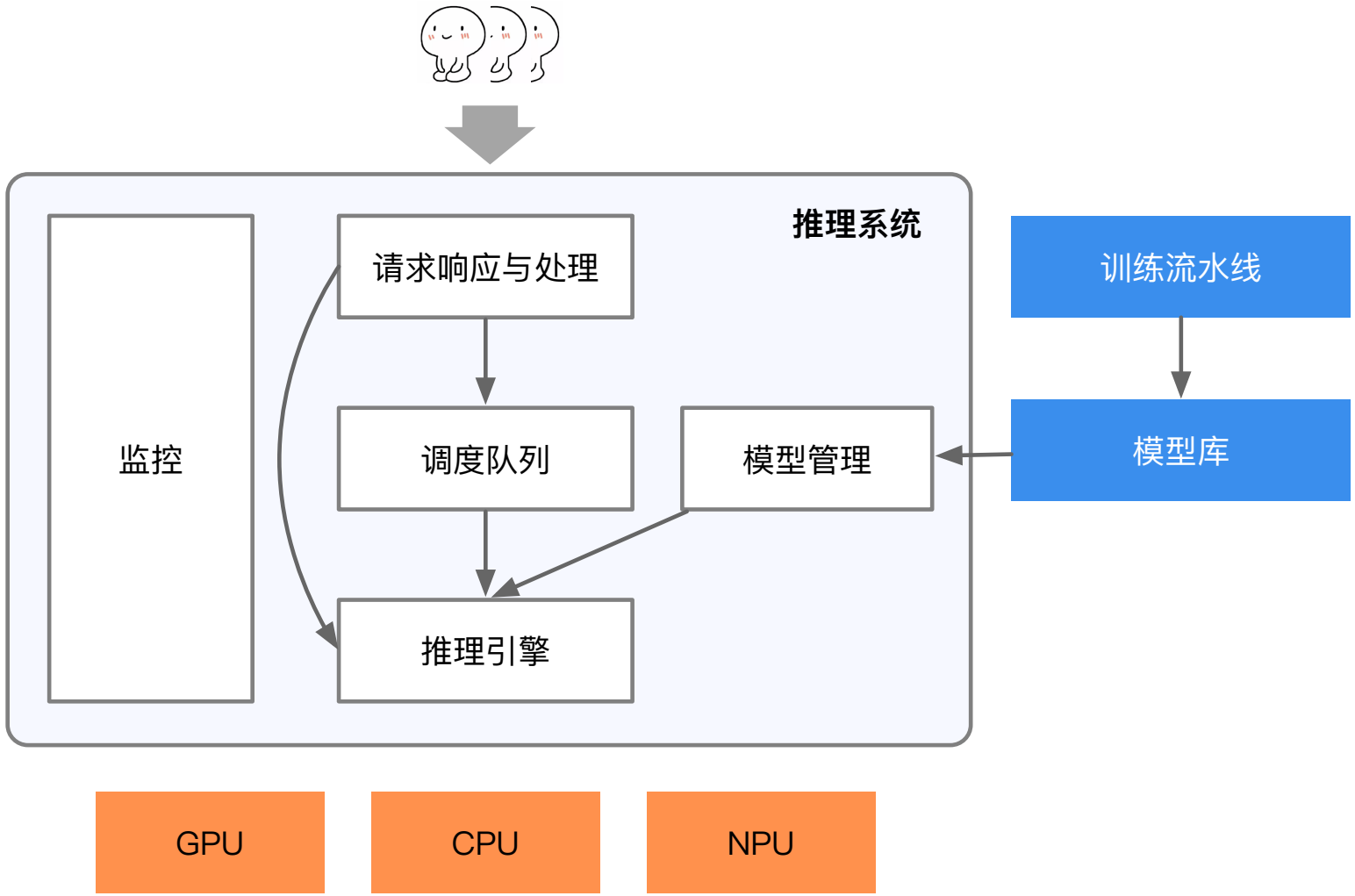




https://devblogs.nvidia.com/kubernetes-ai-hyperparameter-search-experiments/

推理系统 vs 推理引擎

推理系统 vs 推理引擎

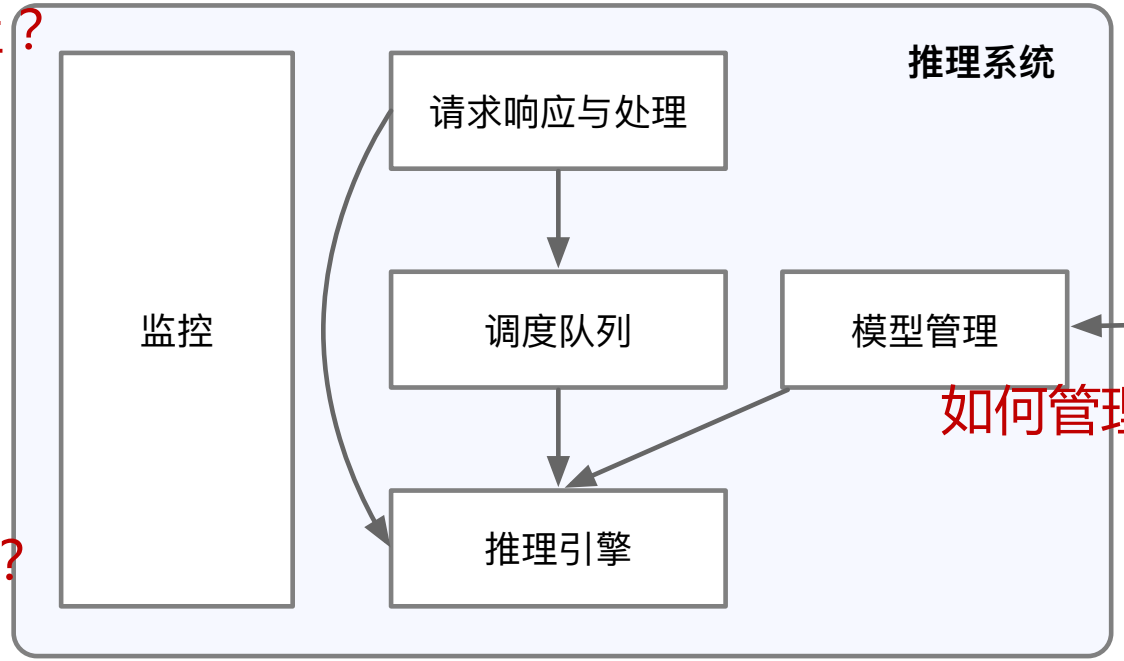


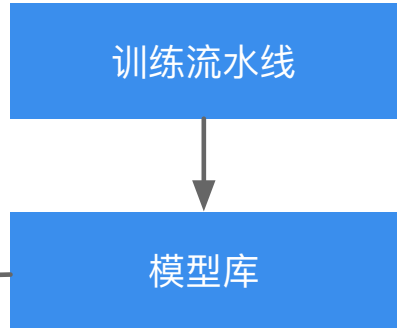
推理系统 vs 推理引擎

如何实现低延迟高吞吐？

如何分配调度请求？

如何管理AI生命周期？

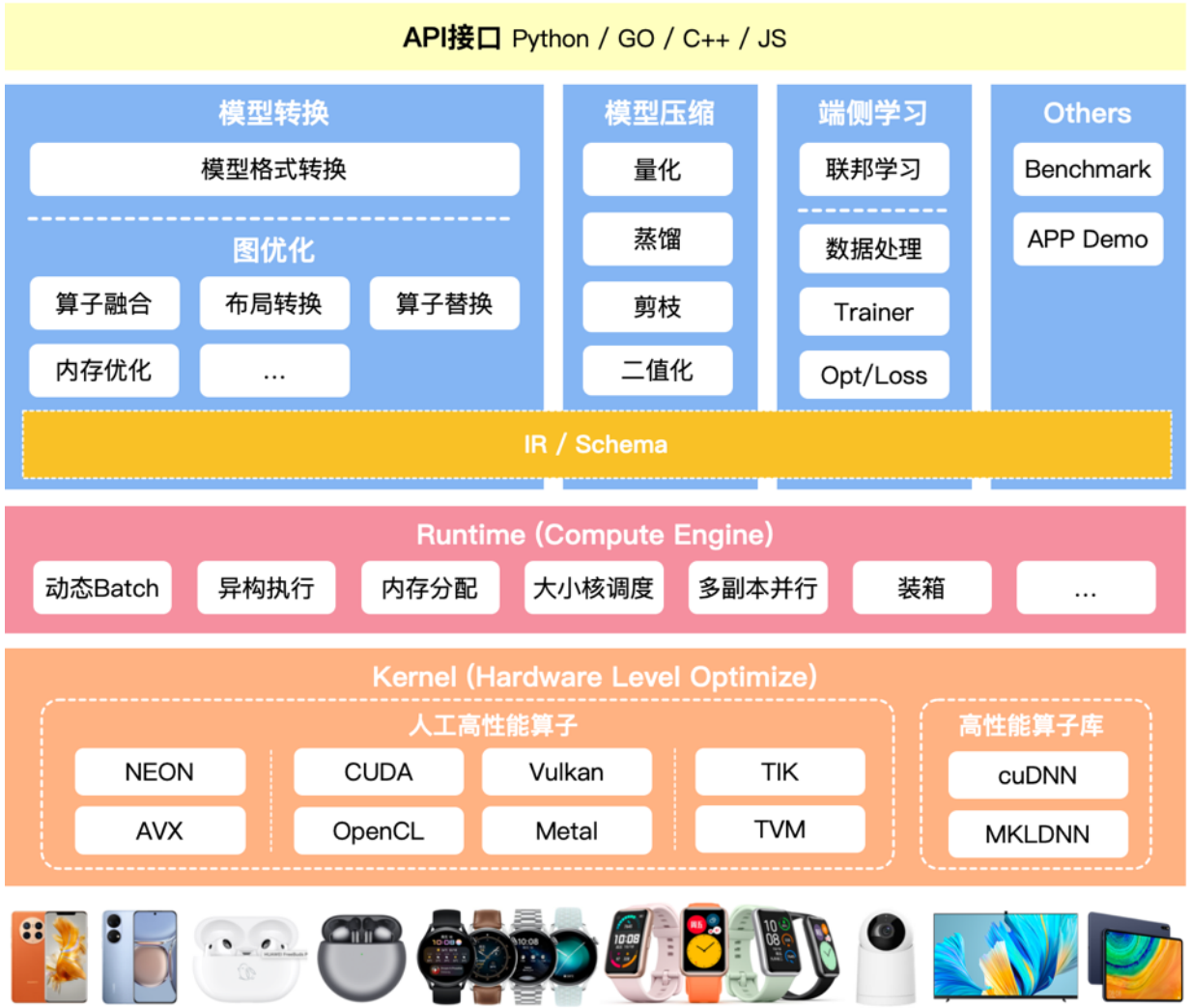




如何管理模型？

推理芯片如何加速？

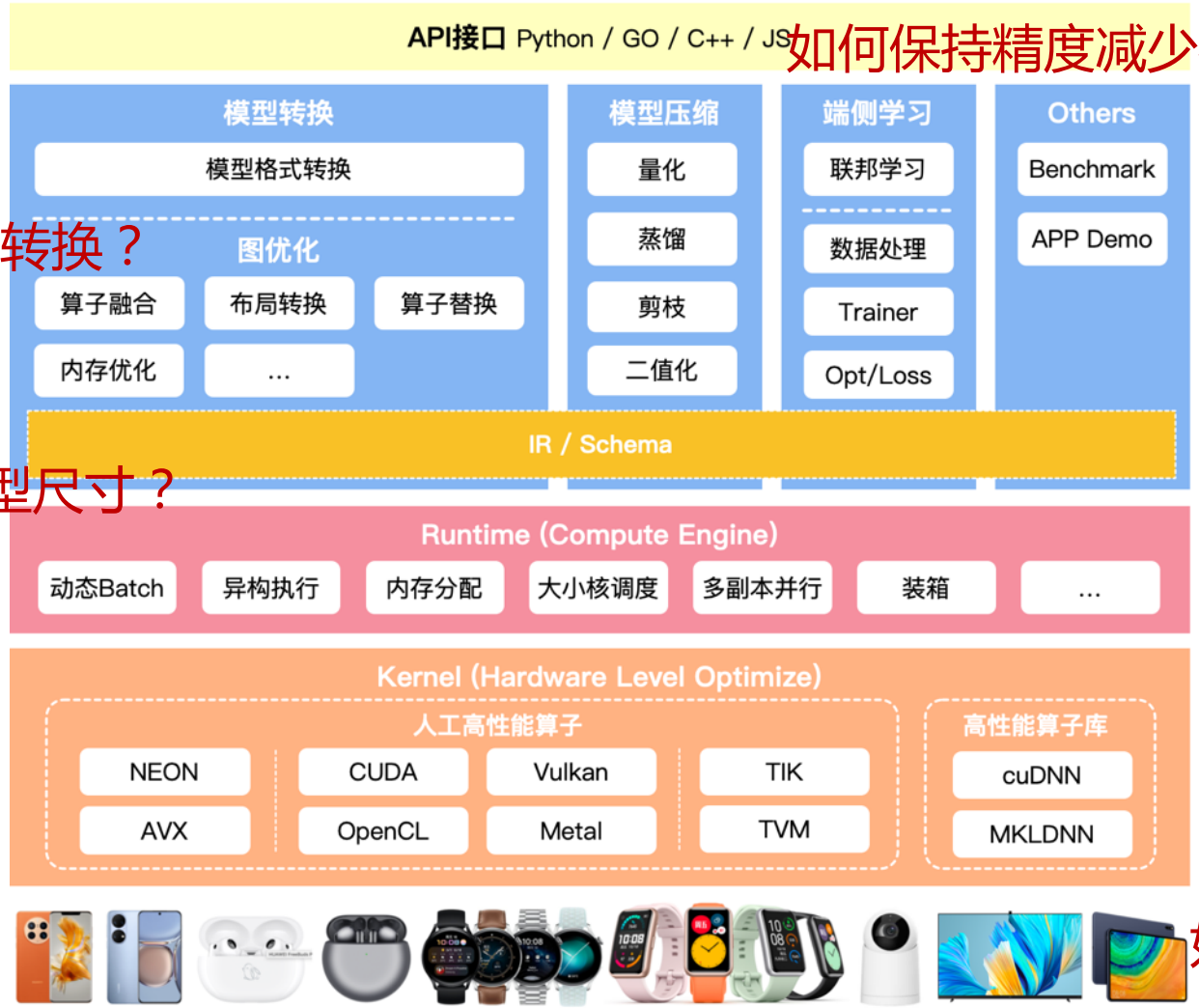
推理系统 vs 推理引擎



推理系统 vs 推理引擎

如何对不同 AI 框架结果转换？

如何保持精度下减少模型尺寸？



如何保持精度减少模型尺寸？

如何加快调度与执行？

如何提高算子性能？

如何利用边缘设备算力？

参考文献

1. Deep Learning Inference in Facebook Data Centers: Characterization, Performance Optimizations and Hardware Implications
2. Clipper: A Low-Latency Online Prediction Serving System
3. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform
4. TensorFlow-Serving: Flexible, High-Performance ML Serving
5. Optimal Aggregation Policy for Reducing Tail Latency of Web Search
6. A Survey of Model Compression and Acceleration for Deep Neural Networks
7. CSE 599W: System for ML - Model Serving
8. https://developer.nvidia.com/deep-learning-performance-training-inference
9. DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING
10. Learning both Weights and Connections for Efficient Neural Networks
11. DEEP LEARNING DEPLOYMENT WITH NVIDIA TENSORRT
12. Halide: A Language and Compiler for Optimizing Parallelism,Locality, and Recomputation in Image Processing Pipelines
13. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning
14. 8-bit Inference with TensorRT
15. https://github.com/microsoft/AI-System