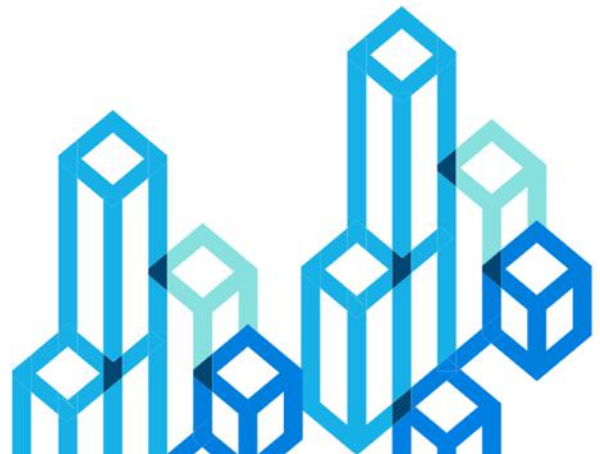
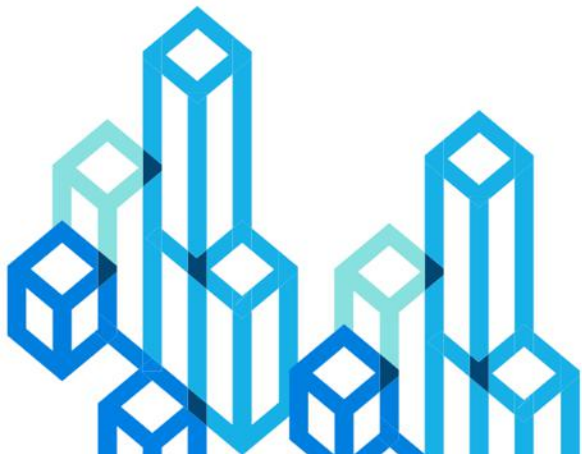


万亿参数多模态模型高效训练的架构实践

张杰
阿里云-机器学习平台PAI
高级技术专家





多模态模型M6简介



模型训练发展趋势和挑战



分布式框架Whale介绍



Whale分布式实践示例



多模态模型M6简介



模型训练发展趋势和挑战



分布式框架Whale介绍

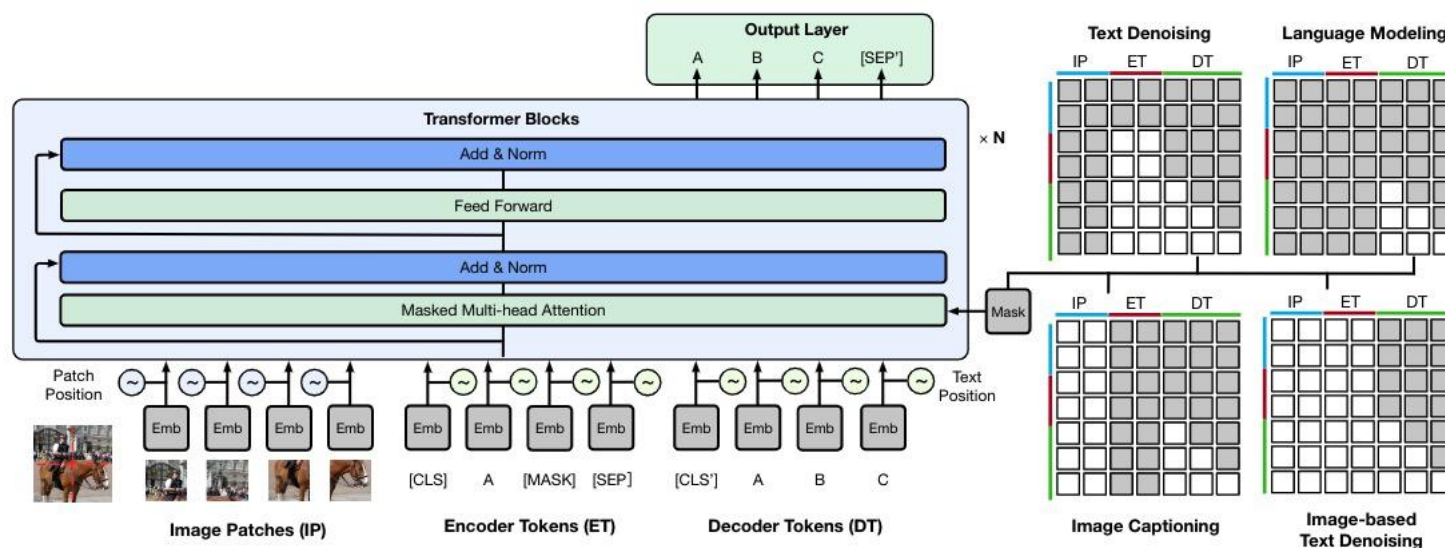


Whale分布式实践示例

M6(MultiModality-to-MultiModality Multitask Mega-transformer)

特点:

- 多模态：图/文 → 图/文，利用掩码设计支持多种类型的预训练任务，如下图所示；
- 模型结构：N层Transformer Blocks；
- 训练数据：20+ TB多模态数据、400+ GB纯文本；
- 参数量：M6-10B(百亿)，M6-100B(千亿)，M6-1000B(万亿)；



图到文生成



M6-base: 经典的西装领设计，修饰颈部线条，凸显女性的干练气质，宽松的版型，不挑身材，穿着舒适自在，优质的面料，亲肤透气，上身挺括有型。

M6-10B: 纯色西装领外套，经典西装领，精致的走线工艺，腰间腰带扣收腰，立体显瘦，袖口荷叶边拼接设计，抬手臂间，带出妩媚温柔的女人味。衣身面料凹凸有致，丰富衣服层次感。

M6-100B: 一款简约不失优雅气质的风衣，采用经典翻领设计，完美修饰脸型。腰间系带装饰，可根据自身需求调节松紧度，穿着舒适方便。两侧大口袋点缀，丰富层次感很实用。

图到文生成



M6-base: 北欧风格以简洁著称于世，并影响到后来的极简主义、简约主义、后现代等风格。在20世纪风起云涌的工业设计浪潮中，北欧风格的简洁被推到极致。

M6-10B: 忙碌了一天回到家中，希望身心彻底放松，一款柔软舒适的抱枕能够轻松满足您的愿望，给您温暖贴心呵护。

M6-100B: 北欧风格家具，以简洁时尚的造型出现于世人面前，可轻松为居室营造舒适惬意的休闲空间氛围，让你感受更为舒适温馨的家居生活体验。

文本到图像生成（女士九分破洞牛仔裤）



文本QA（知乎式问答）

问题

如何增加一个人的自信？



生成回答

谢邀。首先，你要知道自信是什么。自信是一个人对自我的肯定，也就是说，你的自信来源于你的内在，而不是外在。其次，你要明白，自信的本质是什么？自信的来源是你对自我的肯定。所以，你需要做的就是让自己变得更好，让自己的内在更强大，这样你才能获得自信。

zero-shot分类

问题

这是什么品牌的手机?



生成回答

苹果



多模态模型M6简介



模型训练发展趋势和挑战



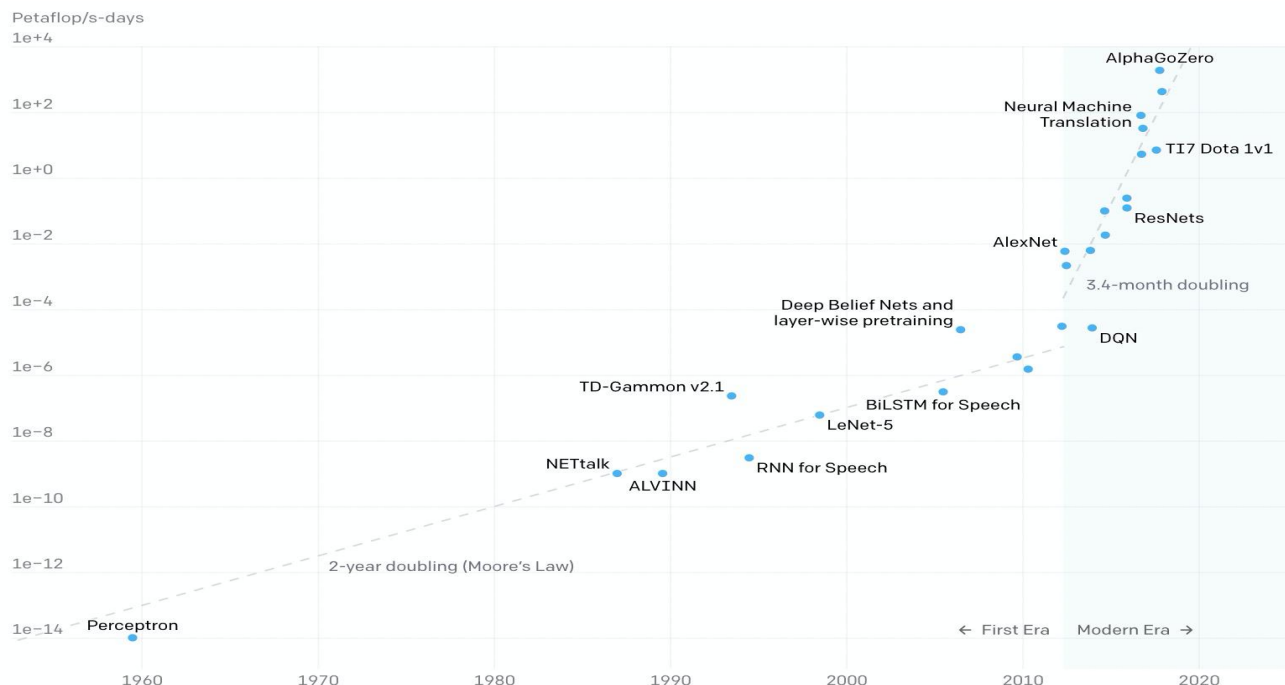
分布式框架Whale介绍



Whale分布式实践示例

深度学习模型发展的两个阶段

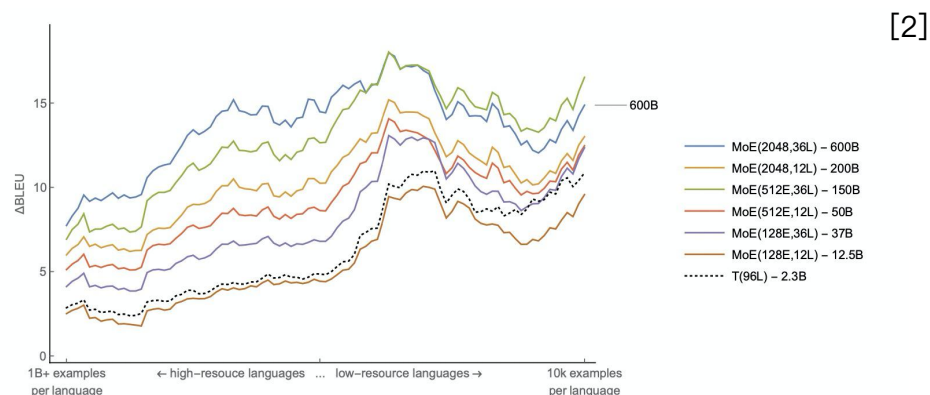
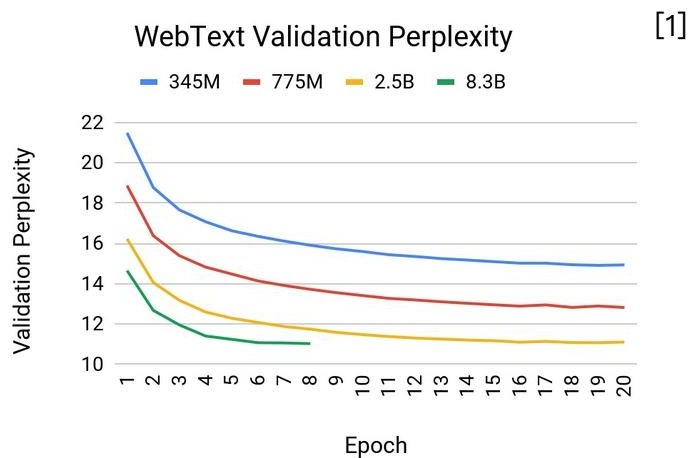
- 2012年以前：模型算力需求每2年翻一倍
- 2012年以后：模型算力需求每3.4个月翻一倍



[1]

模型参数规模和模型效果

- Bert 模型参数规模越大，模型困惑度越低；
- MoE Transformer 模型参数规模越大，翻译质量越高；

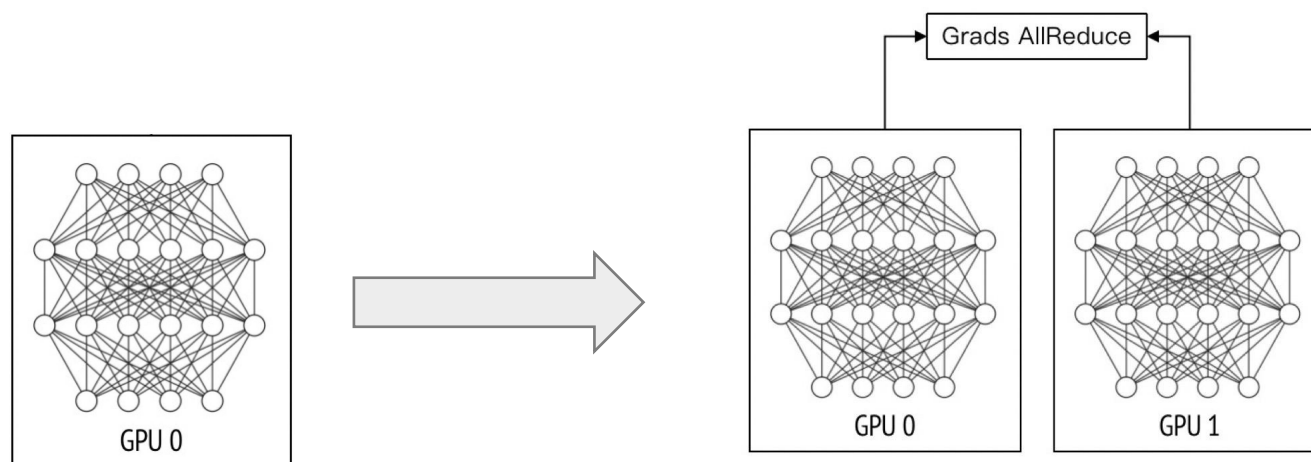


[1] <https://developer.nvidia.com/blog/training-bert-with-gpus/>

[2] GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding

训练方式的变迁

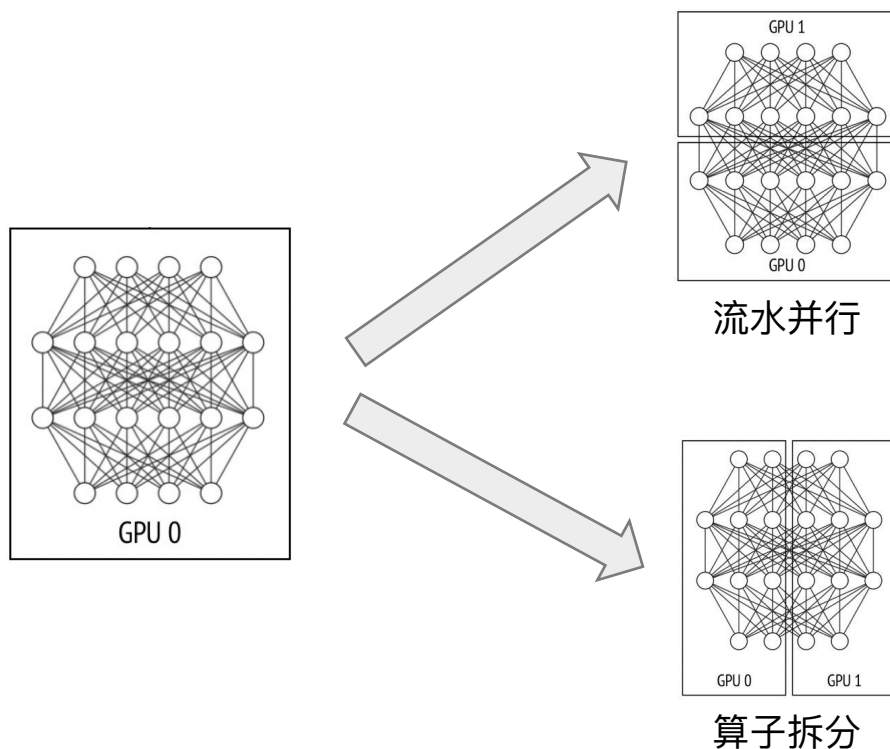
- 模型参数规模越来越大，训练数据越来越多；
- 单GPU训练速度不能满足需求；
- 采用数据并行来加快训练速度；



数据并行加快训练速度

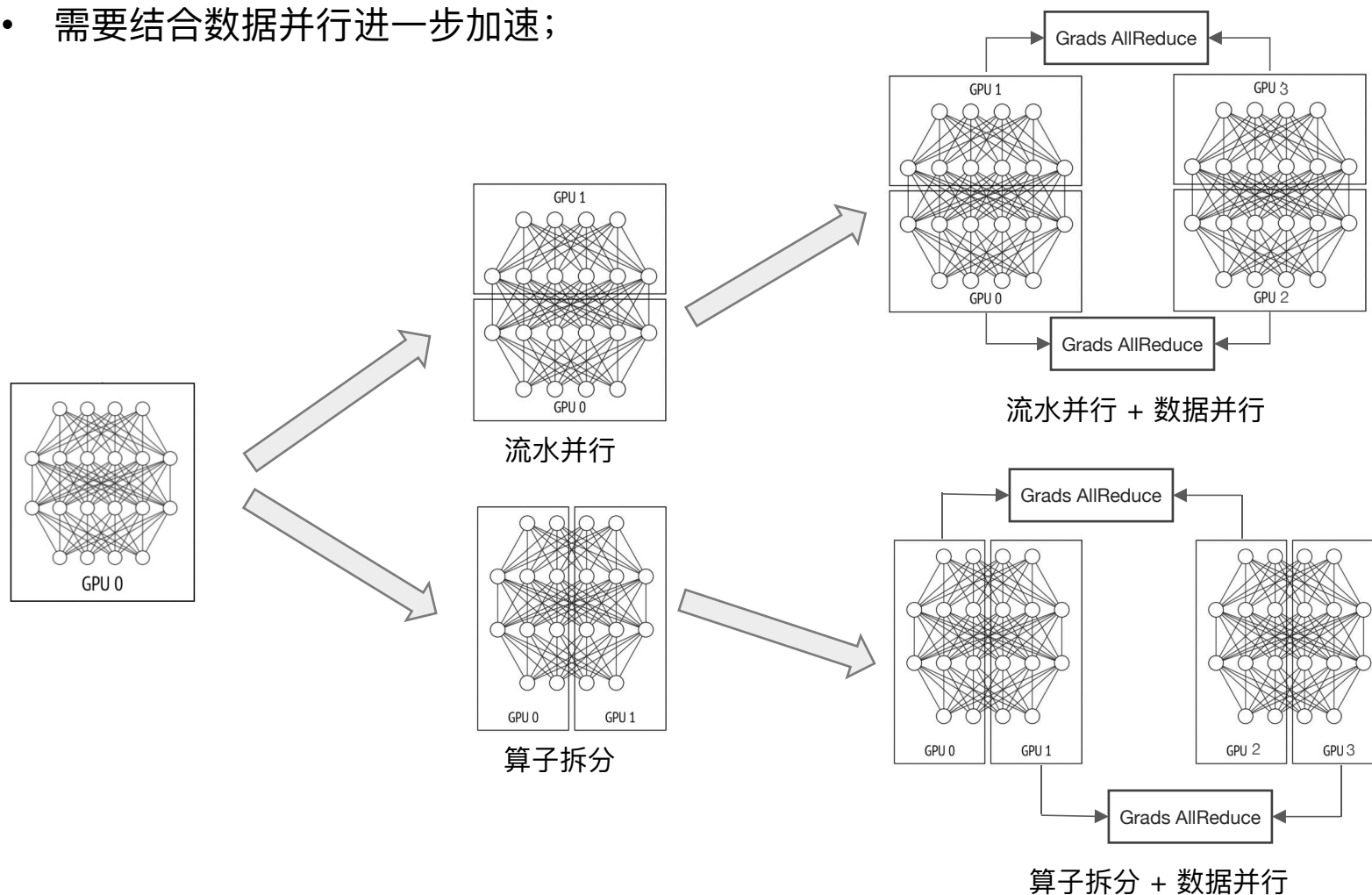
训练方式的变迁

- 随着模型参数规模进一步扩大，单GPU显存已经不能存放模型副本；
- 采用模型并行的策略来进行纵向扩展；
- 模型并行又包括流水并行和算子拆分；



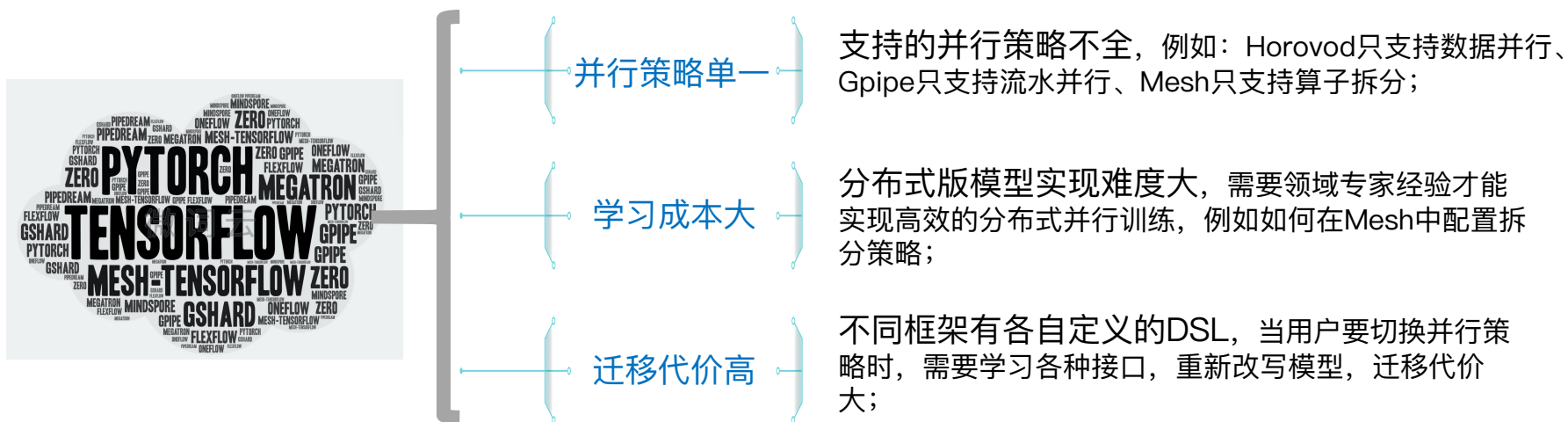
训练方式的变迁

- 单一模型并行训练速度并不能满足业务需求；
- 需要结合数据并行进一步加速；



分布式训练框架

- 当前使用最广的模型训练框架是Tensorflow和PyTorch。有不少基于Tensorflow和PyTorch开发的分布式训练框架，如：Horovod、Mesh Tensorflow等；
- 还有很多其他训练框架如：MindSpore、OneFlow、PaddlePaddle、MXNet等；
- 这些框架支持各种并行策略，但还有不少挑战：





多模态模型M6简介



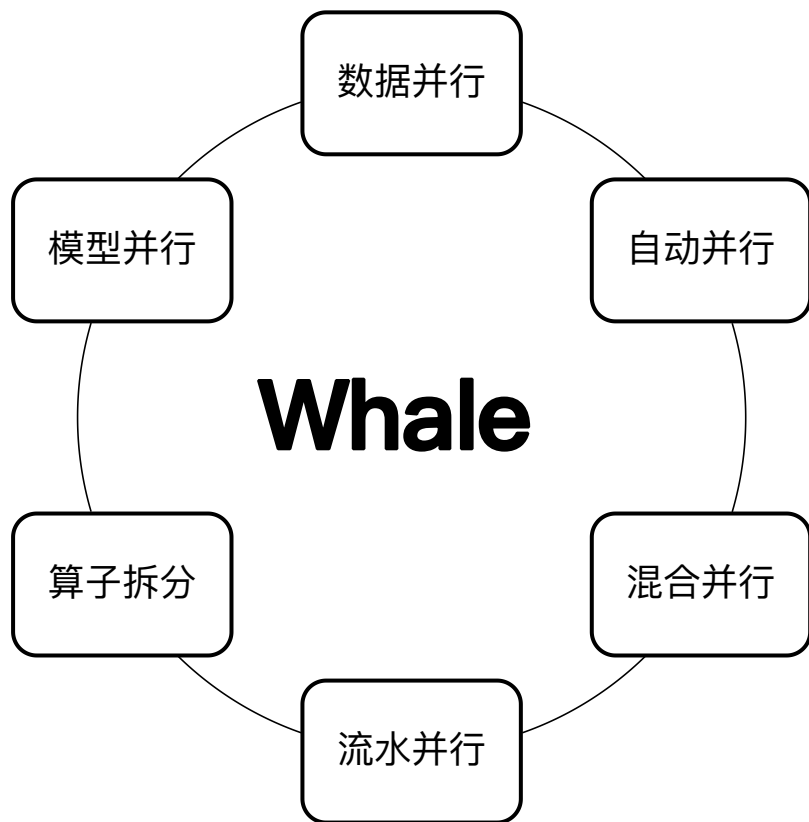
模型训练发展趋势和挑战



分布式框架Whale介绍

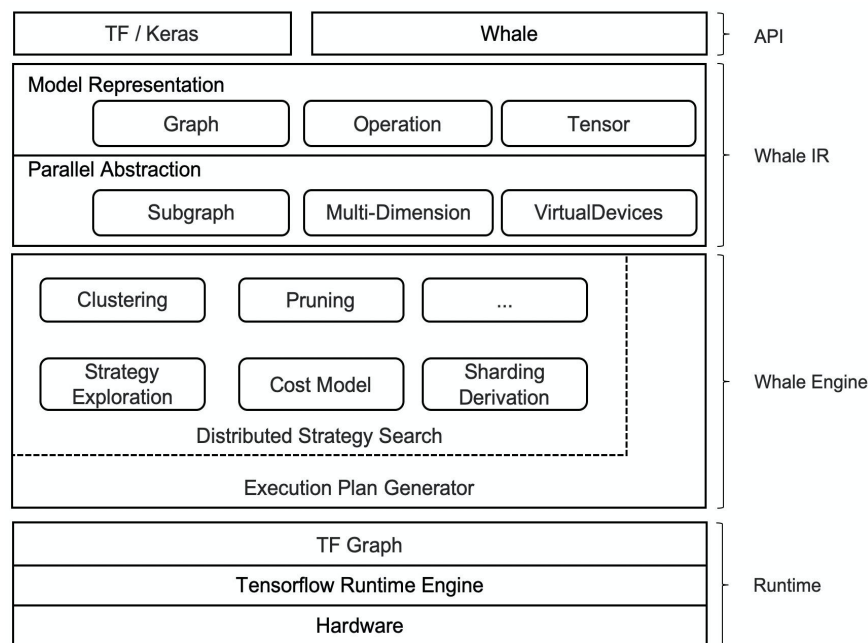


Whale分布式实践示例



特点:

- **多种并行策略统一**: 将不同并行化策略进行统一抽象、封装，在一套分布式训练框架中支持多种并行策略；
- **接口灵活易用**: 基于Tensorflow设计一套分布式并行接口，完全兼容Tensorflow。用户仅仅只需添加几行API调用就可以实现丰富的分布式并行策略；
- **分布式性能更优**: 结合模型结构和网络拓扑进行调度和通信优化，提供高效的分布式训练；
- **自动并行化**: 自动探索最优的并行化策略；

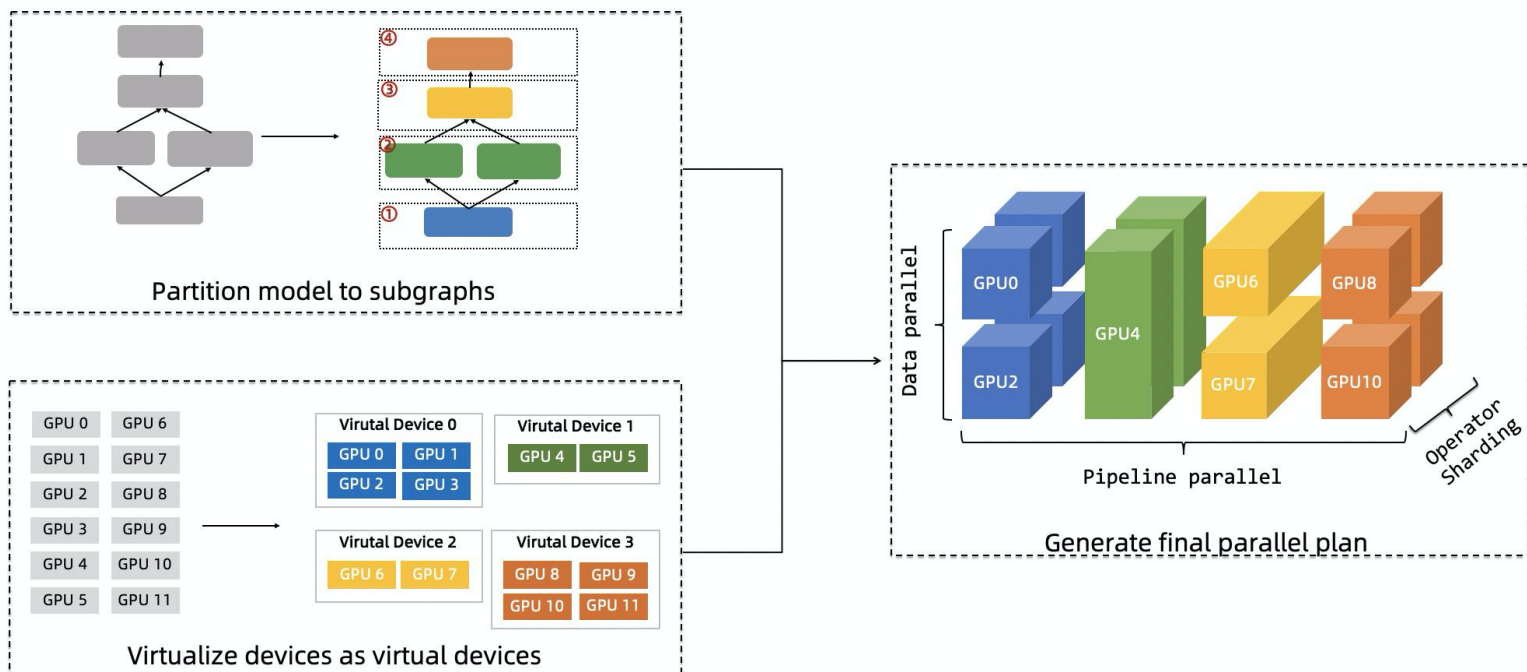


Whale框架:

- **API:** 提供简洁易用接口, 让用户组合使用各种混合并行策略;
- **Whale IR:** 将并行策略转成内部表达, 通过 subgraph、Multi-Dimension、VirtualDevices 抽象来表达各种并行策略;
- **Whale Engine:** 基于Whale IR, 通过图编辑工具来构建分布式执行图;
- **Runtime:** 将分布式执行图转成TF Graph, 再调用TF 的Runtime来执行;

分布式训练流程


- 子图划分和策略配置：通过Whale API来划分子图，并为每个子图配置各种并行策略；
- 虚拟资源划分：按并行策略为每个子图分配devices；
- 生成分布式执行图：基于并行策略和资源，使用图编辑工具来编辑执行图（图拷贝、拆分、插入通信节点等），生成最终的分布式执行图；
- 分布式训练：调用TF的runtime来分布式执行图；



2组APIs配置混合并行策略


cluster

虚拟资源划分

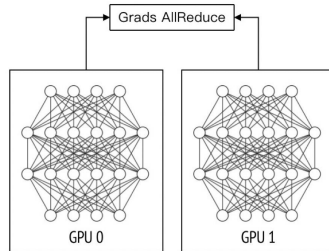
- 
- Row Layout
 - Column Layout
 - Average Layout
 - Specified Layout
 - Auto Layout

scope

基础并行策略

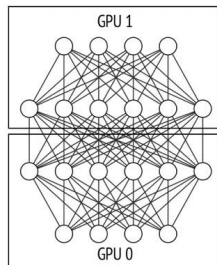
- 
- replica (数据并行)
 - stage (模型并行)
 - split (算子分片)
 - pipeline (流水并行)
 - auto-parallel (自行并行)

示例:



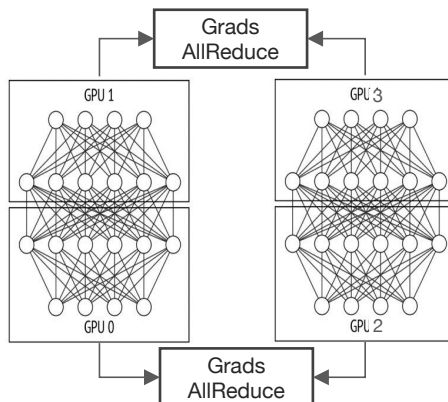
数据并行

```
with wh.cluster():  
  with wh.replica():  
    out = Model()
```



流水并行

```
with wh.cluster():  
  with wh.pipeline():  
    with wh.stage():  
      out = ModelPart1()  
    with wh.stage():  
      out = ModelPart2(out)
```



流水并行+数据并行

```
with wh.cluster():  
  with wh.replica():  
    with wh.pipeline():  
      out = ModelPart1()  
    with wh.stage():  
      out = ModelPart2(out)
```

更多示例：

```
with wh.cluster():  
    with wh.split():  
        out = Model()
```

算子拆分

```
with wh.cluster():  
    with wh.replica():  
        out = ModelPart1()  
    with wh.split():  
        out = ModelPart2(out)
```

组合（算子拆分、数据并行）

```
with wh.cluster():  
    with wh.replica():  
        with wh.split():  
            out = Model()
```

嵌套（算子拆分、数据并行）

```
wh.auto_parallel()  
  
out = Model()
```

自动并行



多模态模型M6简介



模型训练发展趋势和挑战



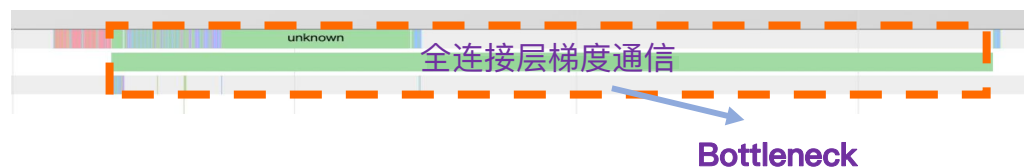
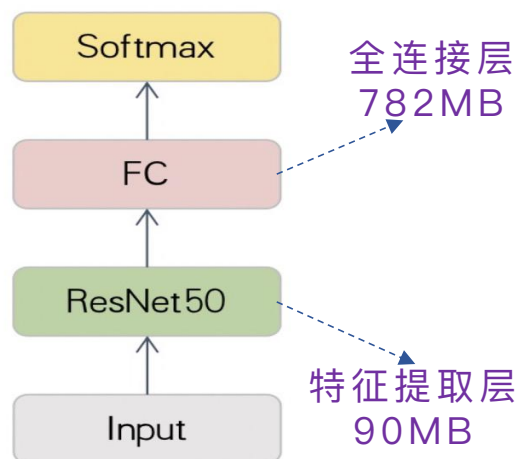
分布式框架Whale介绍



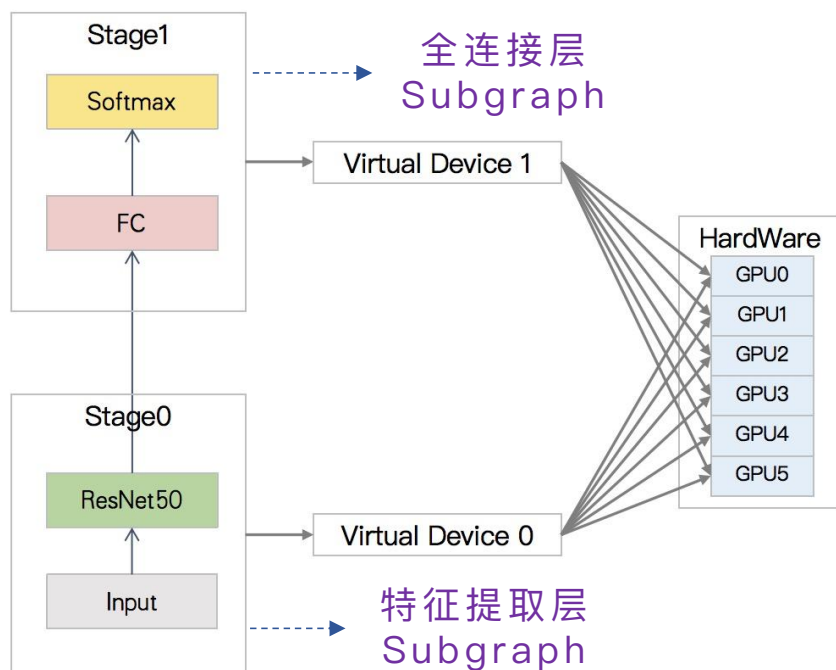
Whale分布式实践示例

100,000分类模型：

- 特征提取层参数90MB，全连接层参数782MB；
- 全连接层参数太大，数据并行训练时梯度通信成为最大瓶颈；



模型数据并行训练timeline



混合并行训练加速：

- **混合并行策略：** 全连接层参数太大，采用算子拆分避免梯度通信，特征提取层参数不大，采用数据并行提高并发度；
- **划分Subgraph：** 将模型划分2个subgraph，特征提取层配置数据并行策略，全连接层配置算子拆分策略；
- **配置Virtual Devices：** 将为每个子图分配一个Virtual Device，完成Virtual Device到Physical Device的映射；

Whale混合并行表达：

- 数据并行和算子拆分组合的并行策略
- 5行代码完成模型分布式改写

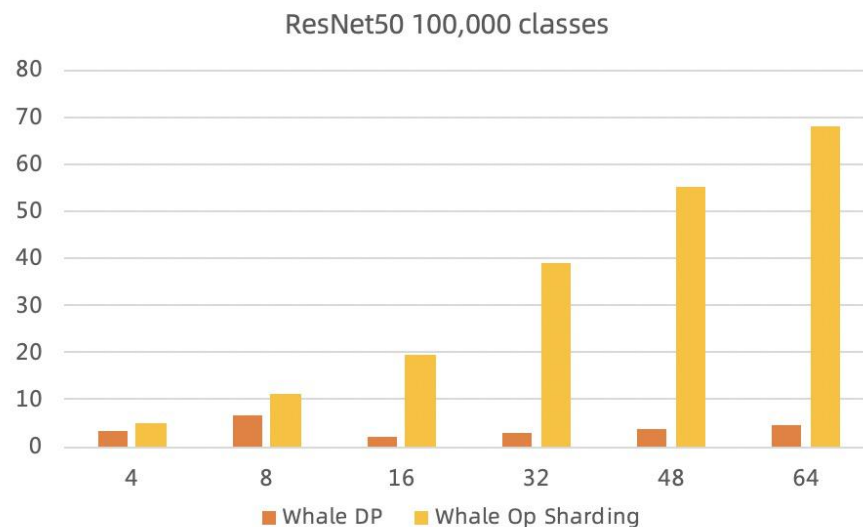
性能：

- 64卡混合并行性能 = **14.8** * 数据并行

扩展性：

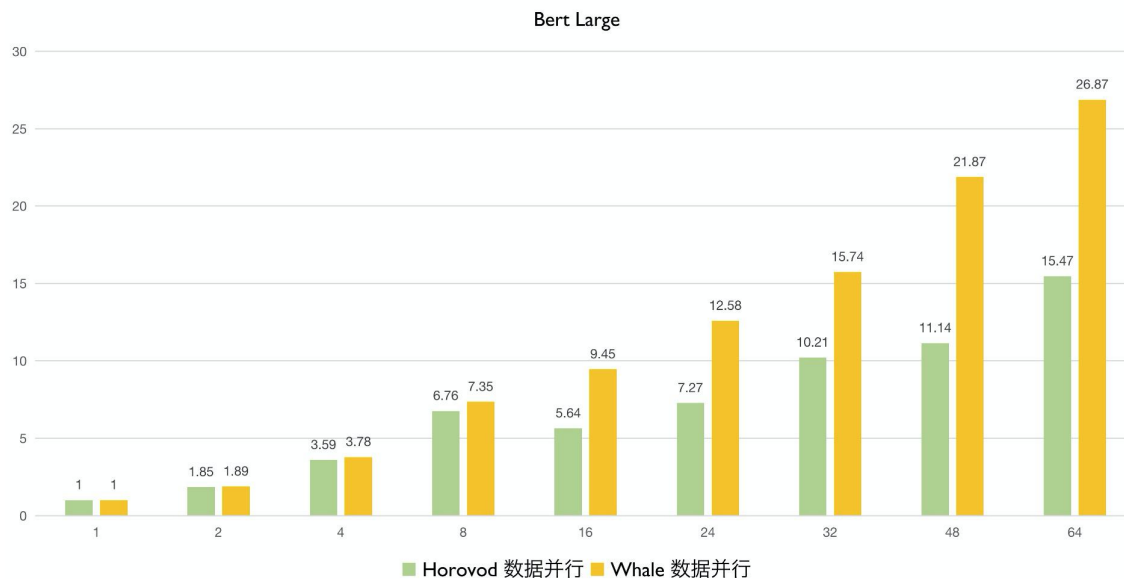
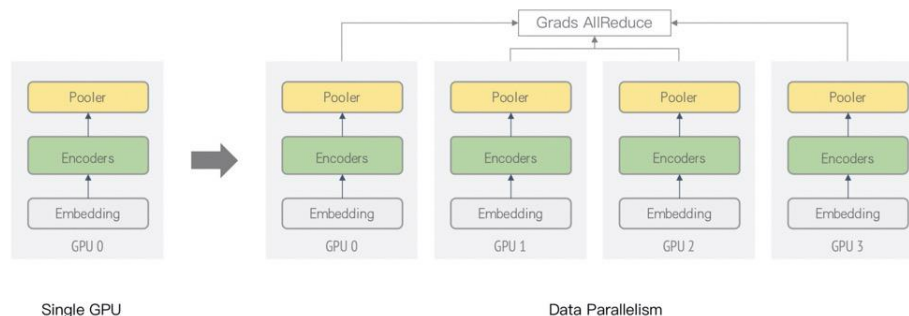
- 只需增加GPU，无需代码修改，可直接扩展更大规模（如1亿分类）

```
import whale as wh
cluster = wh.cluster(layout = {"all"})
with cluster:
    with wh.replica():
        features = ResNet50(inputs)
    with wh.split():
        logits = FC(features)
        predictions = Softmax(logits)
```



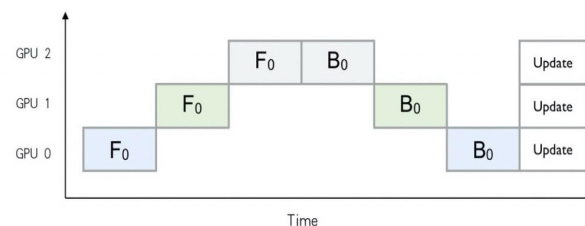
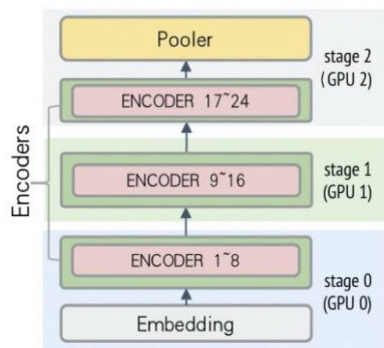
数据并行性能

- 64卡Whale的加速比是Horovod的**1.74**倍；
- 但整体加速比不够理想，主要是梯度通信占时间长，Bert large模型参数有1.2~1.3GB大小；

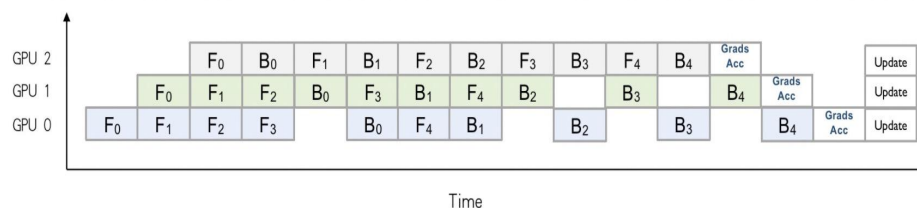


流水并行训练加速：

- Bert large有24层encoder layer，模型有大量重复结构，适合进行模型并行；
- 但模型并行的数据依赖造成device空闲，资源利用率低；
- 结合流水来提高资源利用率；

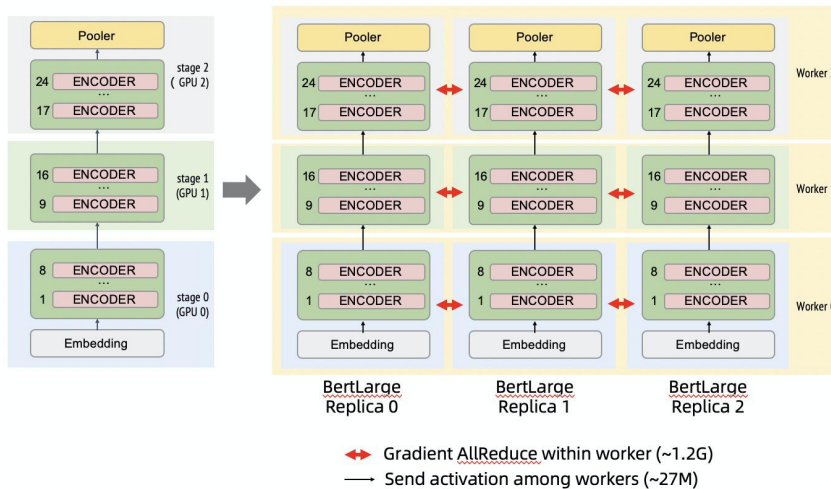


纯模型并行



流水并行

流水并行结合数据并行进行分布式扩展



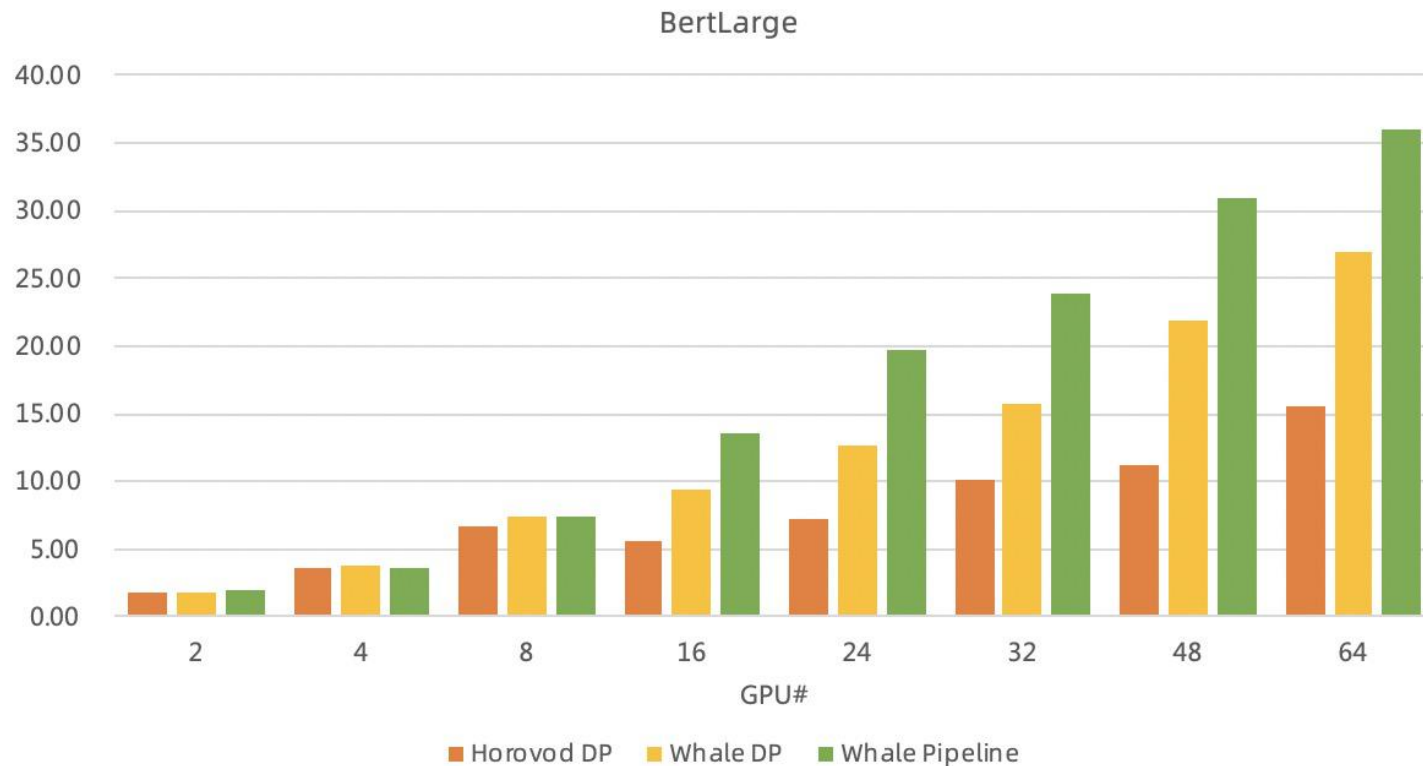
Pipeline + Data Parallelism并行策略

```
import whale as wh
with wh.cluster():
    with wh.replica():
        with wh.pipeline():
            with wh.stage():
                output = embedding(inputs)
                output = encoder_layer_0_8(output)

            with wh.stage():
                output = encoder_layer_8_16(output)

            with wh.stage():
                output = encoder_layer_16_24(output)
                output = pooler(output)
```

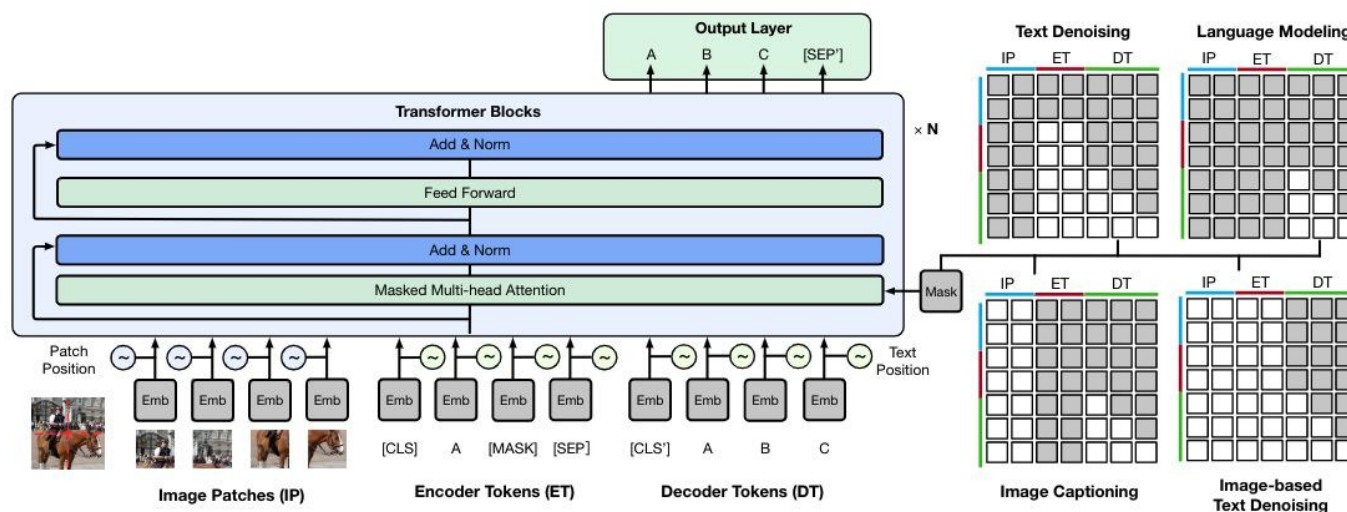
Pipeline + Data Parallelism 训练代码



- 64 V100 GPUs, 机间网络带宽 35Gb
- Whale Pipeline = **1.34** * Whale DP
= **2.32** * Horovod DP

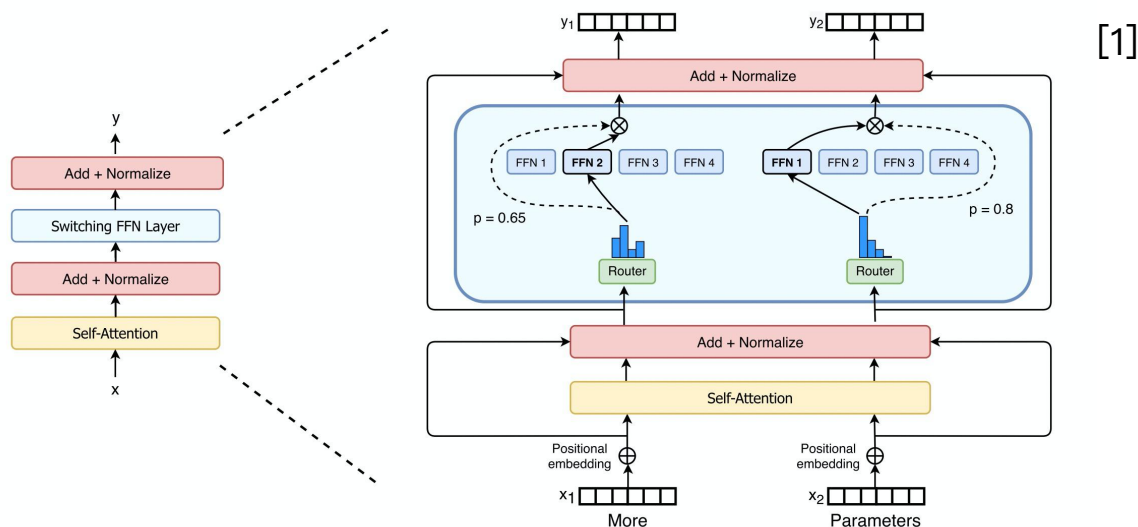
万亿规模模型预训练挑战

- 训练难：
 - 如何实现复杂的并行策略；
 - 如何给用户提供简洁、易用的接口；
 - 万亿规模模型对计算效率、通信效率都带来很大挑战，如何解决；
- 成本高：
 - 模型参数4TB、梯度4TB，加上optimizer states和active tensor，显存需求巨大；
 - 业界训练同等规模模型需要的资源：英伟达 3072 A100、谷歌 2048 TPU；
 - 如何降本增效，使用更少的资源，更快的训练收敛；



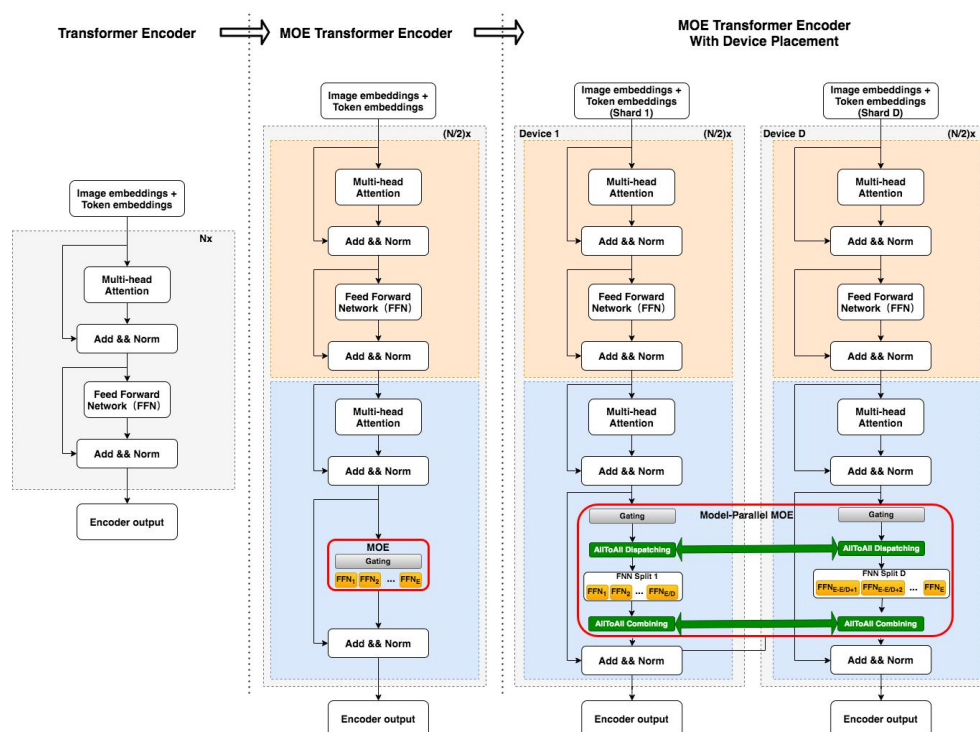
MoE结构进行计算加速

- MoE(Mixture-of-Experts) , 主要特点是稀疏激活:
 - 多专家(experts)策略: 创建多个专家(experts), 如图多个FFN(feed forward)层;
 - 稀疏路由: 对每一个输入, 只选择top k 个experts进行训练, k常用取值有1、2;
- 用MoE layer替换原FFN layer, 降低模型算力需求;



分布式训练策略

- Whale中实现了MoE(Mixture-of-Experts) layer, 并支持专家并行, 将experts拆分到多个Devices上, 降低单个Device的显存和算力需求;
- 数据并行有利于提升训练的并发度, 因此采用数据并行+专家并行组合的混合并行策略:
MoE layer采用专家并行, 其他layer采用数据并行;



分布式训练优化

使用MoE layer和专家并行训练万亿规模模型需要大量的资源。Google训练1.6万亿参数的Switch Transformer模型，需要2048 TPU v3。Whale中实现大量优化技术来降本增效：

- 显存优化
 - Auto Gradient Checkpoint, 自动选择最优checkpoint节点，节约activation的显存；
 - Group-wise Apply, 优化Optimizer Apply阶段的显存；
 - CPU Offload技术，优化Optimizer status和Weight的显存；
 - 通信池化，控制通信的数据块大小和并发，节约通信的显存；
- 计算、通信加速：
 - 采用数据并行+模型并行(experts拆分)的混合并行策略，降低算力需求；
 - 采用分组融合通信、半精度通信、拓扑感知的All2All通信算子等技术来提高通信效率；
 - 结合混合精度、编译优化等技术提高训练效率；

简洁易用接口

- 混合并行策略，增加几行annotation：
- Whale自动进行DP+EP的并行策略训练：

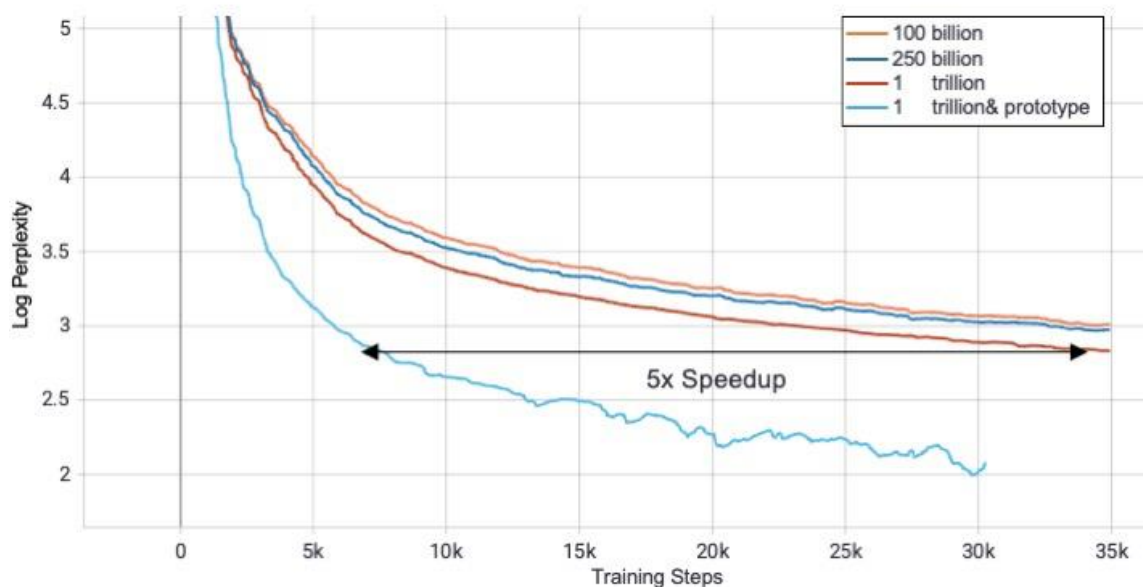
```
1 import whale as wh
2 wh.set_default_scope(wh.replica)
3
4 gates = softmax ( einsum ("GSM , ME -> GSE", inputs , wg ))
5 combine_weights , dispatch_mask = Top2Gating ( gates )
6 dispatched_expert_inputs = einsum ( "GSEC , GSM -> EGCM", dispatch_mask , reshaped_inputs )
7
8 with wh.split():
9     h = einsum ("EGCM , EMH -> EGCH", dispatched_expert_inputs , wi )
10    h = relu ( h )
11    expert_outputs = einsum ("EGCH , EHM -> GECM", h , wo )
12
13 outputs = einsum ("GSEC , GECM -> GSM", combine_weights, expert_outputs)
```

默认采用数据并行策略

MoE部分采用算子拆分策略

效果

- **480** GPU(V100)上完成万亿规模模型预训练；
- **3天**内模型训练收敛；
- 相比于英伟达(3072 A100)、谷歌（2048 TPU v3）训练同等规模模型资源大幅减少、性价比大幅提升；





多模态模型M6简介



模型训练发展趋势和挑战



分布式框架Whale介绍



Whale分布式实践示例

Thanks!