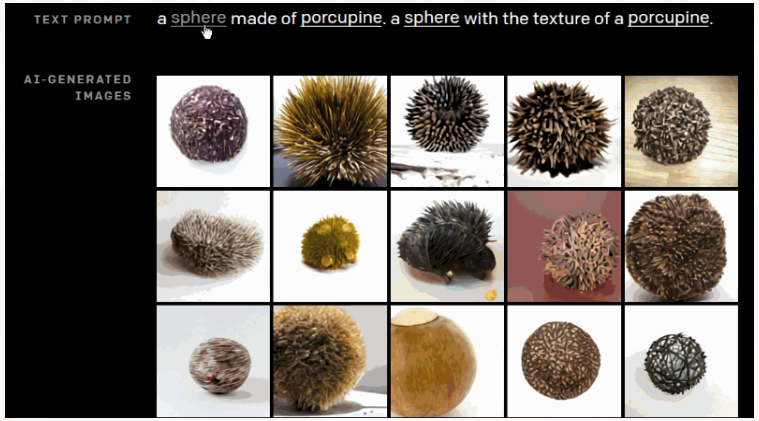
支持大模型的分布式机器学习系统性能优化

北京大学 计算机学院
崔斌

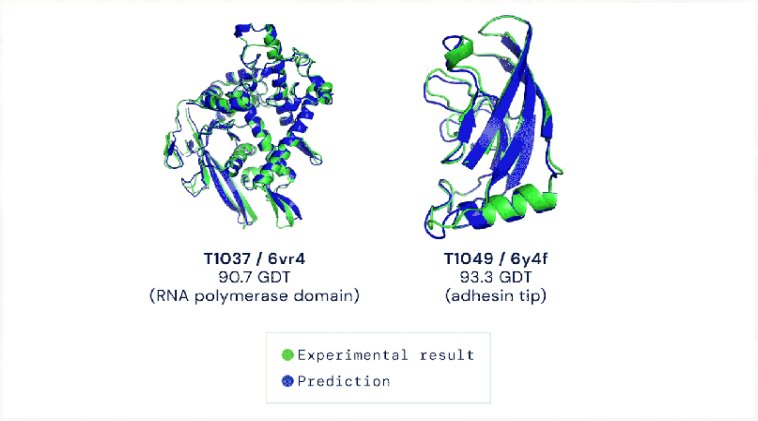
2023.11

深度学习

- 以深度学习为代表的人工智能技术飞速发展，取得了广泛的应用



DALL-E
Art Generator



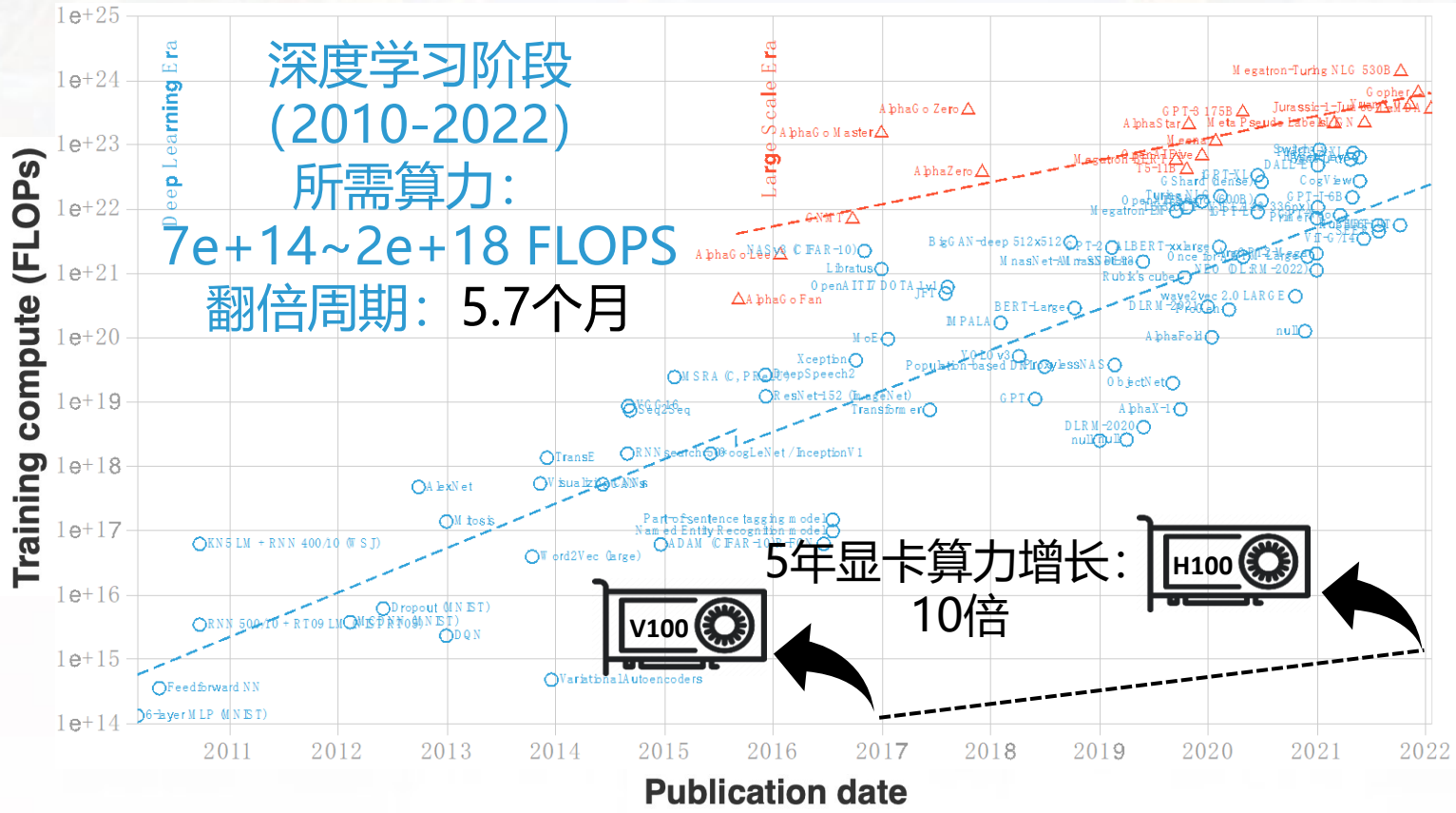
AlphaFold2
Protein Prediction

ChatGPT
Chatbot Copilot

深度学习发展带来的挑战

- 深度学习训练算力需求增长远超**摩尔定律**，单一硬件设备难以满足计算需求

机器学习训练所需算力发展趋势随时间变化情况

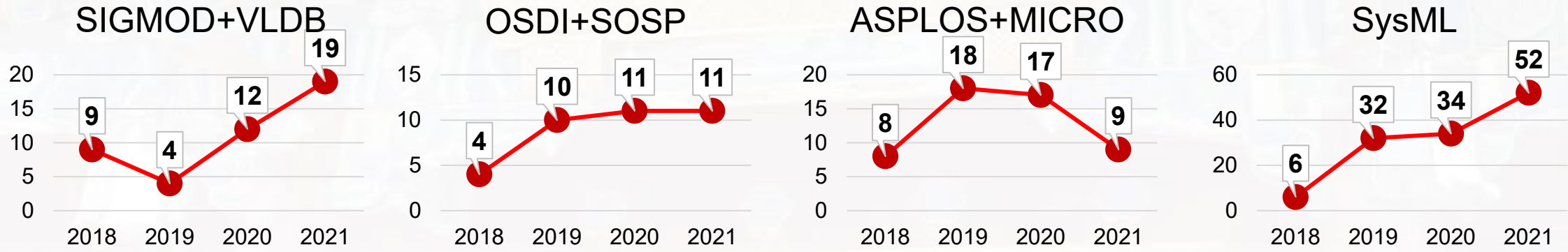


分布式训练势在必行！

Sevilla et al., Compute Trends Across Three Eras of Machine Learning. 2022.

背景与意义

- 日益增长的模型和数据规模对现有系统带来了严峻的挑战，分布式深度学习系统变得越来越重要。
- 分布式机器学习/深度学习系统相关研究在人工智能、数据管理、操作系统、体系架构等相关国际顶级学术会议上热度呈逐年波动上升趋势。



计算机科学顶级会议历年接受论文中AI系统相关论文数

统计数据来源：https://github.com/Hsword/Awesome-Machine-Learning-System-Papers

- 分布式深度学习系统的设计既需要熟悉深度学习模型，更需要对数据管理、分布式系统、计算机网络以及高性能计算等有深刻的认识。
- 分布式深度学习系统领域的发展有助于研究者挖掘大数据中隐藏的知识，探索更大规模的AI模型。

报告内容

What

什么是大规模分布式训练?

Why

为什么要做大规模分布式深度学习框架?

How

怎么做大规模分布式训练?

Hetu 河图

分布式深度学习系统

github.com/PKU-DAIR/Hetu

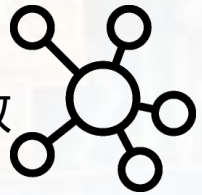
什么是大规模分布式深度学习训练?

学习目标

$$\min_x \left[F(x) := \frac{1}{|\xi|} \sum_i f(x; \xi_i) \right]$$

海量训练数据 ξ

超大模型参数 x

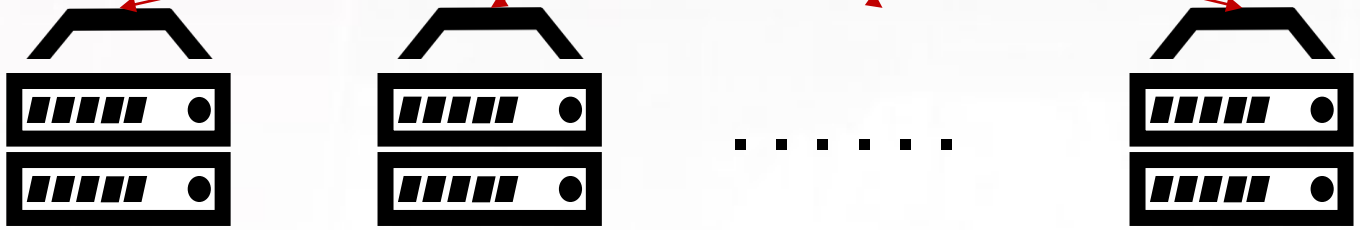


学习算法：分布式SGD

迭代更新公式(第k轮): $x_{k+1} = x_k - \gamma \left[\frac{1}{N} \sum_{i=1}^{N} g_k(x_k; \xi^i) \right]$

$g_k(x_k; \xi^i)$: 模型x_k关于数据ξ^i的梯度

γ: 学习率



数据并行

- 在数据并行模式下，每个设备上部署有完整的模型，对模型的权重、结构不进行切分，而只对训练样本切分为多个mini-batch，每个设备负责一个mini-batch的训练。

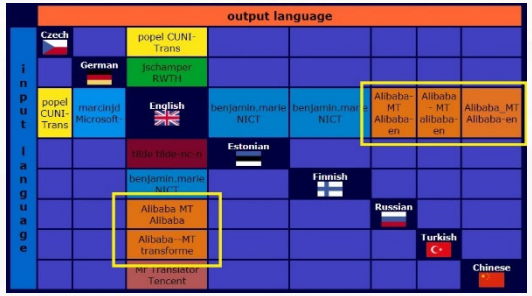
模型并行

- 模型并行从计算图的切分角度，可以分为以下几种：
  - 1、按模型的layer层切分到不同设备，即层间并行，也称为流水并行。
  - 2、将计算图中的层内的参数切分到不同设备，即层内并行，也称为张量并行。

大规模分布式深度学习训练

- “大”数据?
  - 有Label数据—GB级别

ImageNet
图像分类



Workshop on Machine Translation
机器翻译

Open Graph Benchmark
图学习

COCO
目标检测

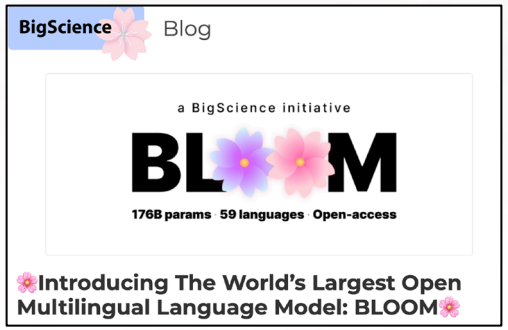
Criteo
点击率预测

大规模分布式深度学习训练

- “大”数据？
  - 无Label数据—TB级别

PanGu 40TB训练数据

WuDao 3TB文本+90TB图文+181GB对话

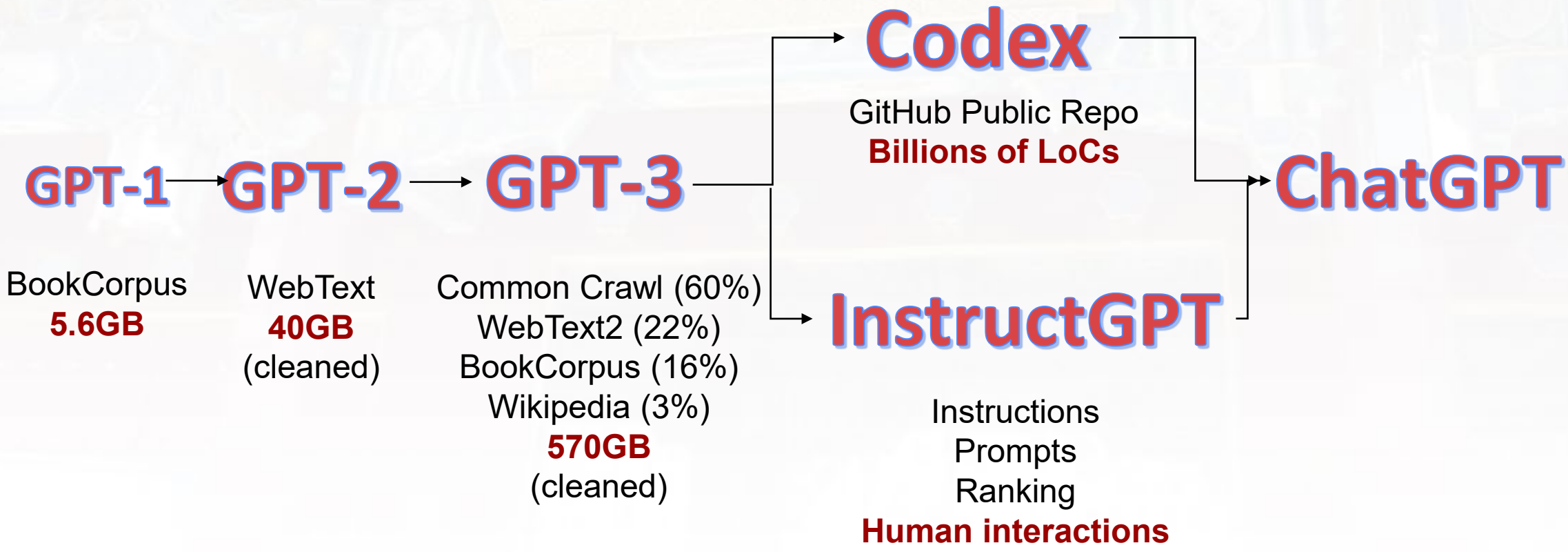


BigScience BLOOM 1.5 TB文本

DeepMind Gopher 10.5 TB文本

大规模分布式深度学习训练

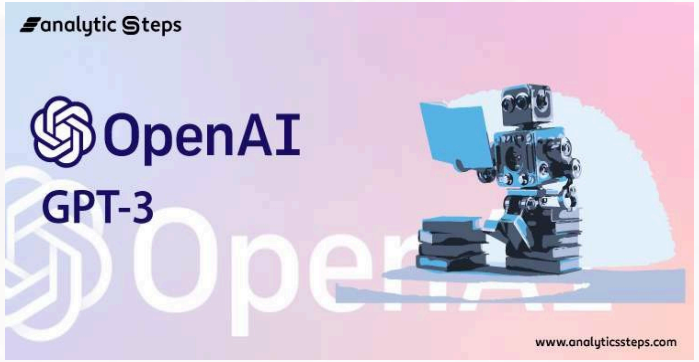
- “大”数据?
  - 混合人工数据 — Human-in-the-Loop



大规模分布式深度学习训练

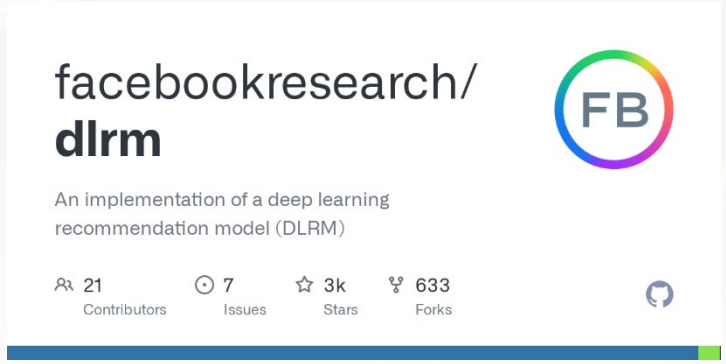
- “大”模型？
  - 超过单个GPU显存容量限制：40GB -> 100亿参数
  - 目前的大模型参数量最高已经达到万亿级别

稠密—Transformer模型



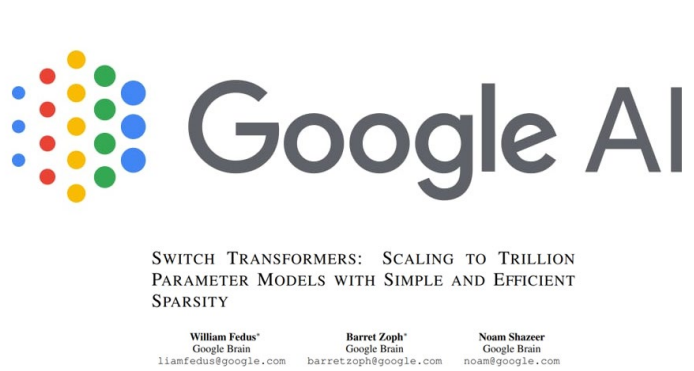
OpenAI GPT-3（ChatGPT）
(175 Billion params)

稀疏—Embedding模型



Facebook DLRM
(12 Trillion params)

稀疏—MoE模型



Google Switch Transformer
(1.6 Trillion params)

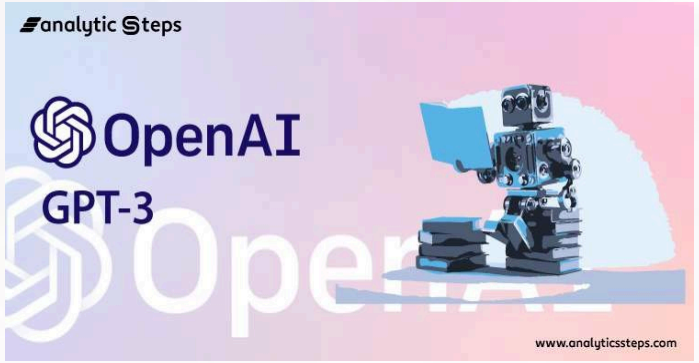
大规模分布式深度学习训练

- “大”模型？
  - 超过单个GPU显存容量限制：40GB -> 100亿参数
  - 目前的大模型参数量最高已经达到万亿级别

以GPT-3为例

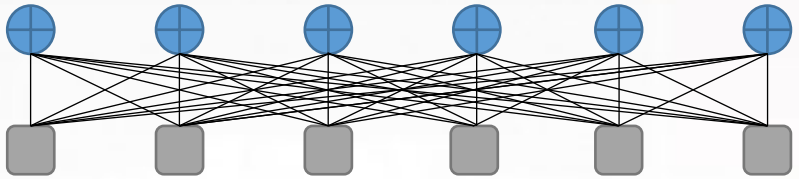
$n_{layers} = 96,$
$d_{model} = 12288,$
$n_{heads} = 96,$
$d_{FFN} = 4 * d_{model},$
参数量约为:
$(12288 * 128 * 4 * 96$ 注意力层
$+$
$2*4*12288*12288)$ 全联接层
$* 96 \approx 175\ B$

稠密—Transformer模型



OpenAI GPT-3
(175 Billion params)

- 类型1: 稠密—Transformer模型
  - 参数量：数亿至千亿规模，全联接层和注意力层
  - 计算量：全联接层和注意力层
  - 内存消耗量：全联接层和注意力层
  - 例：Google T5、OpenAI GPT-3、华为 PanGu



自注意力连接模型结构

大规模分布式深度学习训练

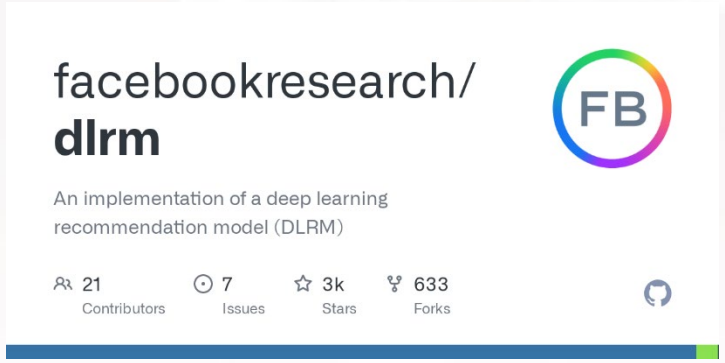
- “大”模型？
  - 超过单个GPU显存容量限制：40GB -> 100亿参数
  - 目前的大模型参数量最高已经达到万亿级别

以FB DLRM为例

总计10个embedding table，总参数1.2万亿，其中最大单表参数量约为:

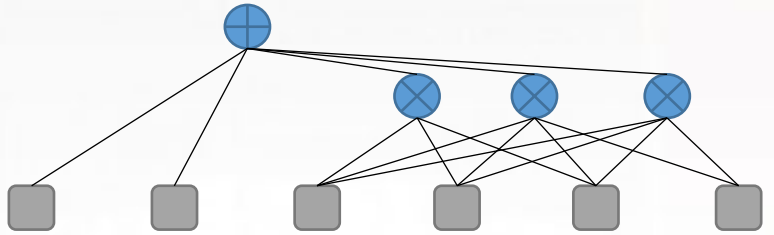
$$10\,B\ *256 \approx 2.6\,T$$

稀疏—Embedding模型



Facebook DLRM
(12 Trillion params)

- 类型2: 稀疏—Embedding模型
  - 参数量：可达万亿规模，高维Embedding层
  - 计算量：稀疏Embedding部分以及后续NN计算
  - 内存消耗量：Embedding层
  - 例：Facebook DLRM、快手 Kraken、百度 HierPS



Embedding

Wide & Deep Embedding模型结构

大规模分布式深度学习训练

- “大”模型？
  - 超过单个GPU显存容量限制：40GB -> 100亿参数
  - 目前的大模型参数量最高已经达到万亿级别

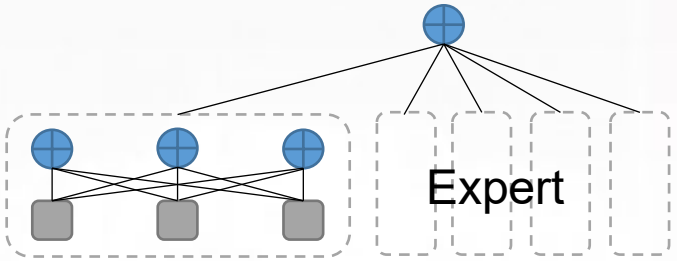
以WuDao为例

$n_{expert} = 480,$
$n_{layers} = 18,$
$d_{model} = 4096,$
$d_{FFN} = 24576,$
参数量约为:
$(4096 * 4096 * 4$ 注意力层
$+$
$2*4096*24576*480)$ Expert 全联接层
$* 18 \approx 1621\ B$

稀疏—MoE模型

Google Switch Transformer
(1.6 Trillion params)

- 类型3: 稀疏—MoE模型
  - 参数量：可达万亿规模，全体Expert模型
  - 计算量：稀疏Expert模型
  - 内存消耗量：Expert模型
  - 例：谷歌Switch Transformer、智源WuDao、阿里M6、**（GPT4, 1.8T，16个1.1B Experts？）**

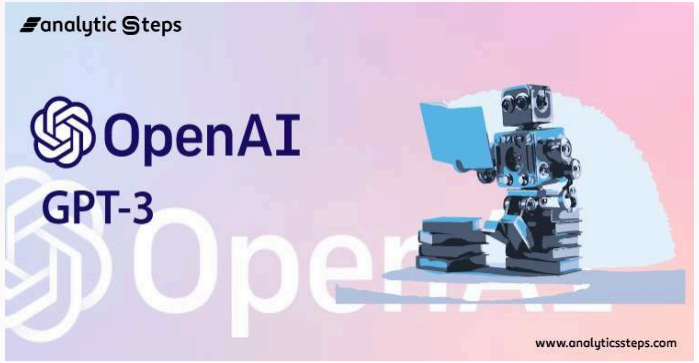


Mixture of Experts 模型结构

大规模分布式深度学习训练

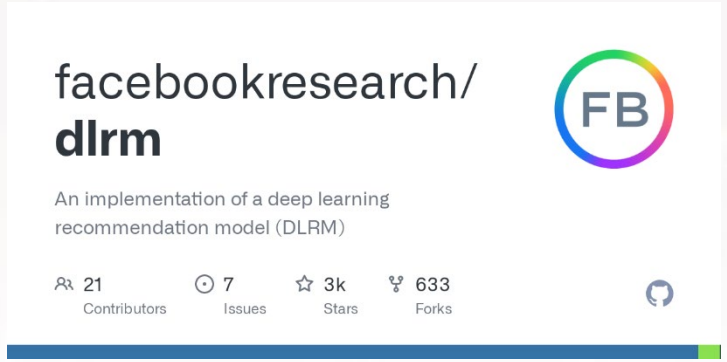
- “大”模型？
  - 超过单个GPU显存容量限制：40GB -> 100亿参数
  - 目前的大模型参数量最高已经达到万亿级别

稠密—Transformer模型



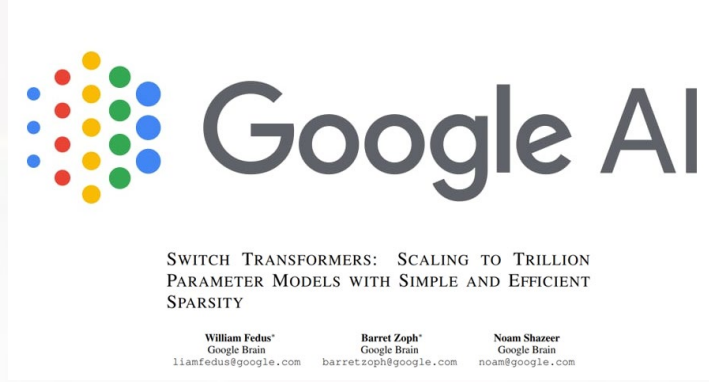
OpenAI GPT-3
(175 Billion params)

稀疏—Embedding模型



Facebook DLRM
(12 Trillion params)

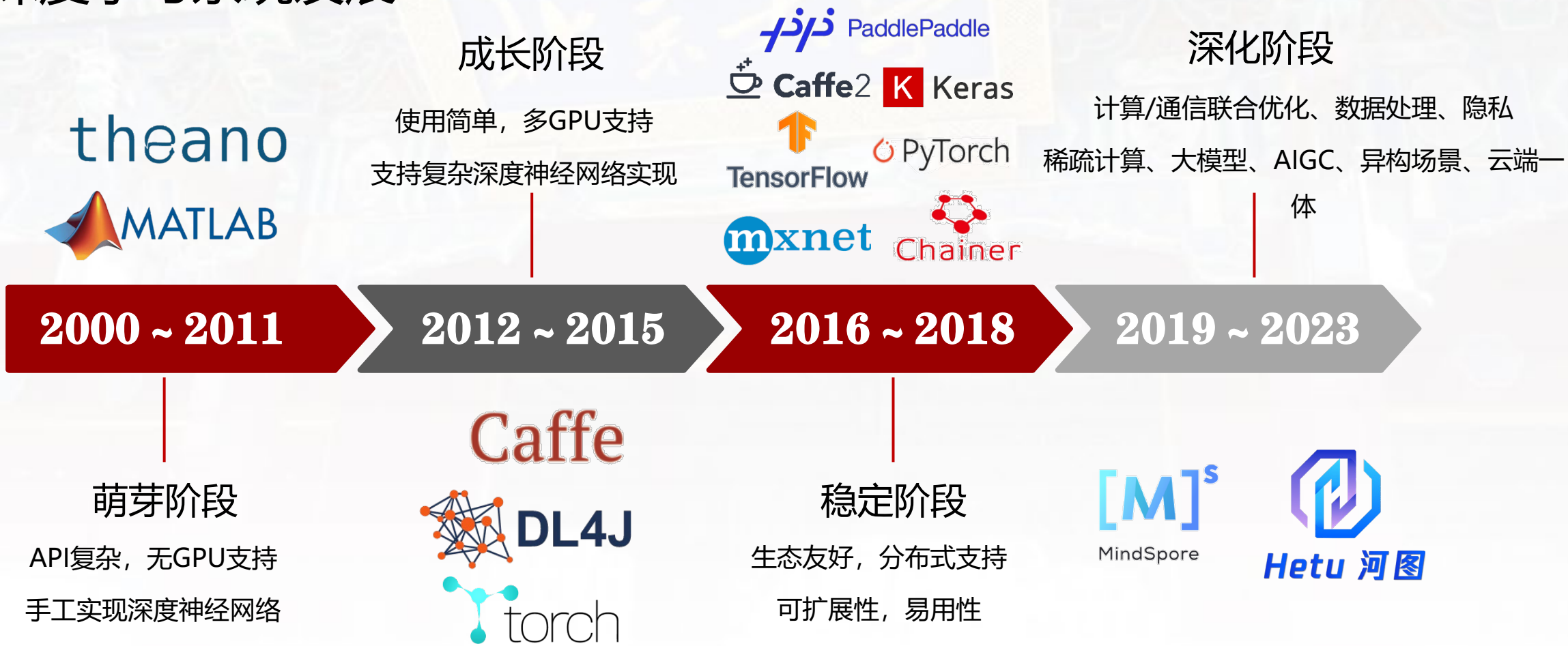
稀疏—MoE模型



Google Switch Transformer
(1.6 Trillion params)

为什么要做大规模分布式深度学习框架?

深度学习系统发展



分布式拓展能力逐渐成为DL框架的核心竞争力，但仍存在多方面的挑战

为什么要做大规模分布式深度学习框架?

- 现有分布式深度学习系统存在的问题:

系统功能性

支持的通信架构、并行策略、一致性协议受限

系统易用性

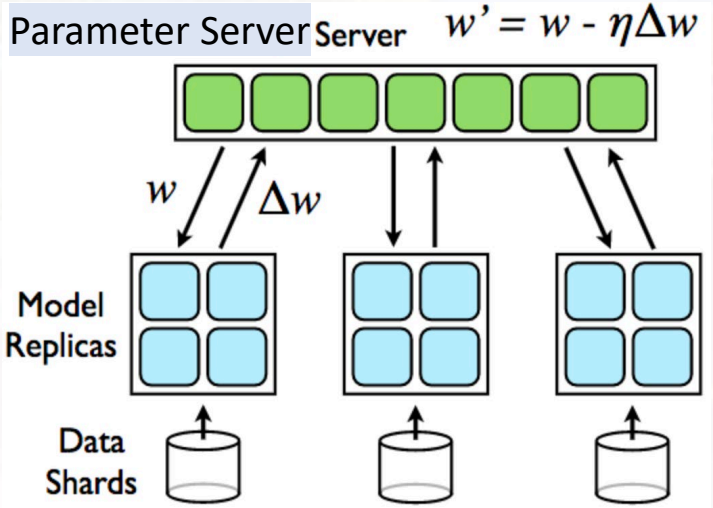
分布式执行部署复杂，学习成本高

系统复杂性

计算与通信耦合程度高，不利于扩展和优化

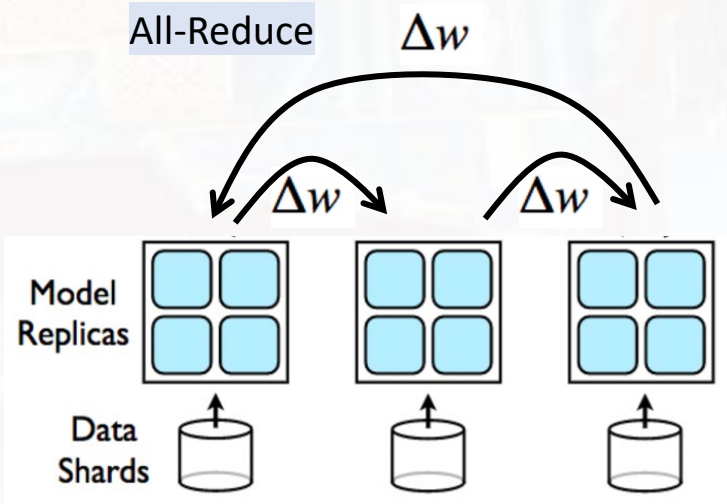
为什么要做大规模分布式深度学习框架?

- 系统功能性
  - 以通信架构为例：支持架构较为单一



特点： 中心化存储
常见应用场景： IO密集型任务（数据大，模型大）
- 点击率预估、词向量学习等

系统： TensorFlow、MXNet



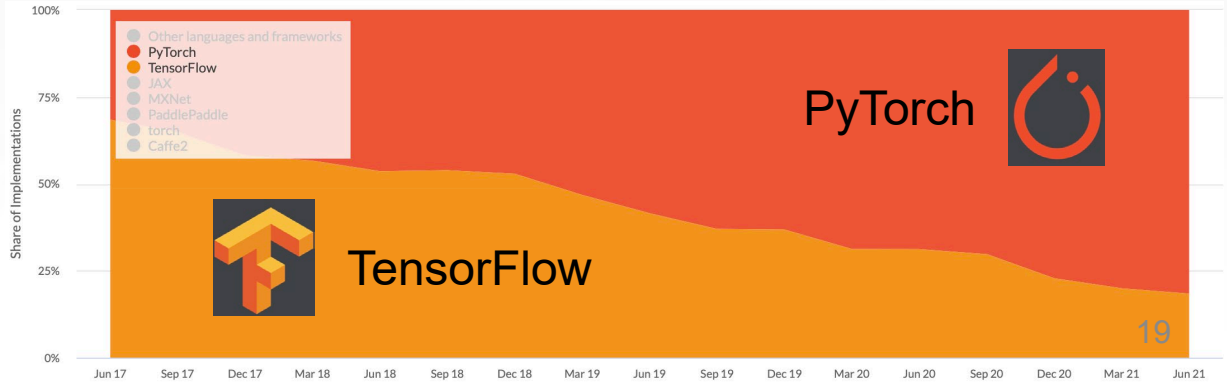
特点： 去中心化存储
常见应用场景： 计算密集型任务
- 图像分类、语音识别、机器翻译等

系统： PyTorch、Caffe2

为什么要做大规模分布式深度学习框架?

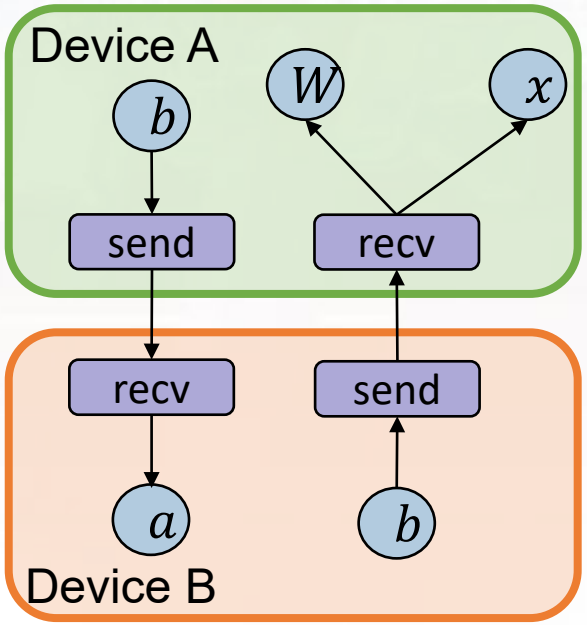
- 系统功能性
  - 以并行策略（模型并行）为例：支持的并行策略受限
    - 参数并行：Parallax、HugeCTR
    - 算子并行：Mesh-TensorFlow、FlexFlow、Megatron-LM
    - 流水并行：Gpipe、Pipedream、DAPPLE
    - 优化器并行：DeepSpeed & ZERO
    - 子图并行：FastMoE、GShard

2021年发表论文中DL框架使用趋势

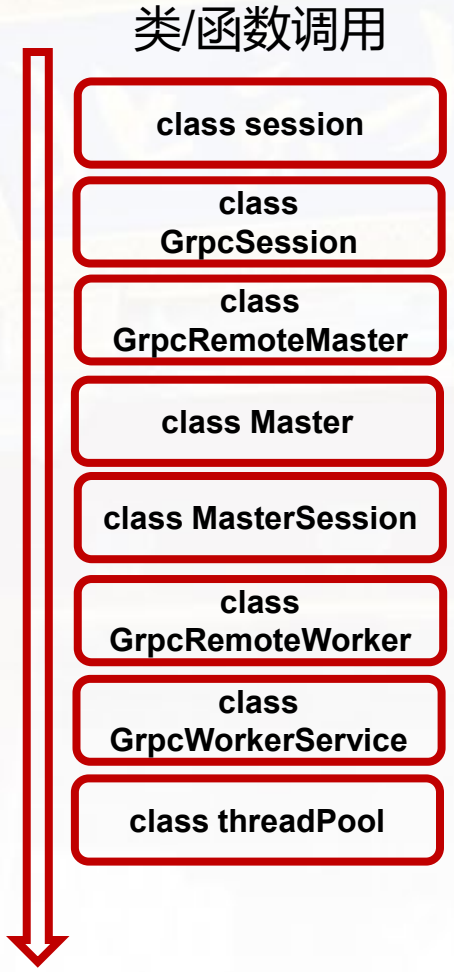


为什么要做大规模分布式深度学习框架?

- 系统复杂性
  - 代码调用栈
  - 系统间差异
  - 统一抽象&非统一实现



计算图抽象



TensorFlow分布式会话函数调用分析

为什么要做大规模分布式深度学习框架?

- 系统易用性：分布式训练代码复杂

```
1  import tensorflow as tf
2  N = num_samples
3  x, y = get_input_data(N)
4  logits = build_vgg16_model()
5  loss = loss_function(logits, y)
6  opt = SGD_optimizer(lr)
7  train_op = opt.minimize(loss)
8  sess = tf.Session()
9  for i = 1 -> num_epochs do
10   return_val = sess.run(train_op)
```

TF 单卡训练

```
1  import tensorflow as tf
2  N = num_samples
3  total_op = []
4  total_grads = []
5  for i = 1 -> num_gpus do
6    x,y = get_input_data(N/num_gpus)
7    logits = build_vgg16_model()
8    loss = loss_function(logits, y)
9    opt = SGD_optimizer(lr)
10   grads = opt.compute_gradients(loss)
11   total_grads.append(grads)
12 grads, vars = split_grads_and_vars(total_grads)
13 avg_grads = []
14 for grad in grads do
15   avg_grads.append(aggregate_grads(grad)/num_gpus)
16 total_grads = merge_grads_and_vars(avg_grads, vars)
17 for i = 1 -> num_gpus do
18   total_op.append(opt.apply_gradients(total_grads))
19 sess = tf.Session()
20 for i = 1 -> num_epochs do
21   return_val = sess.run(total_op)
```

TF 多卡训练

```
1  import tensorflow as tf
2  import parallax
3  import parallax_config
4  N = num_samples
5  sess_config = tf.ConfigProto()
6  config = parallax_config.build_config()
7  config.sess_config = sess_config
8  single_gpu_graph = tf.Graph()
9  with single_gpu_graph.as_default():
10   x,y = get_input_data(N/num_gpus)
11   logits = build_vgg16_model()
12   loss = loss_function(logits, y)
13   opt = SGD_optimizer(lr)
14   train_op = opt.minimize(loss)
   # e.g. resource_info_file = IP_Address:0,1,2,3
15 parallax_sess, _, _, num_replicas_per_worker = \
       parallax.parallel_run(single_gpu_graph, \
                             resource_info_file, config)
16 for i = 1 -> num_epochs do
17   return_val = parallax_sess.run(train_op)
```

Parallax 多卡训练

data preparation
model construction
additional user effort
model execution

为什么要做大规模分布式深度学习框架?

- 系统易用性：高效部署需要人工干预

- 现有半自动并行方案：
  - 策略式：TF.distribute.Strategy、PyTorch Distributed、Parallax、AutoDist
  - 标注式：Mesh-TensorFlow、GShard、MindSpore、OneFlow、Whale

策略式

- tf.ParameterServerStrategy
- torch.nn.parallel.DistributedDataParallel
- autodist.strategy.PSLoadBalancing

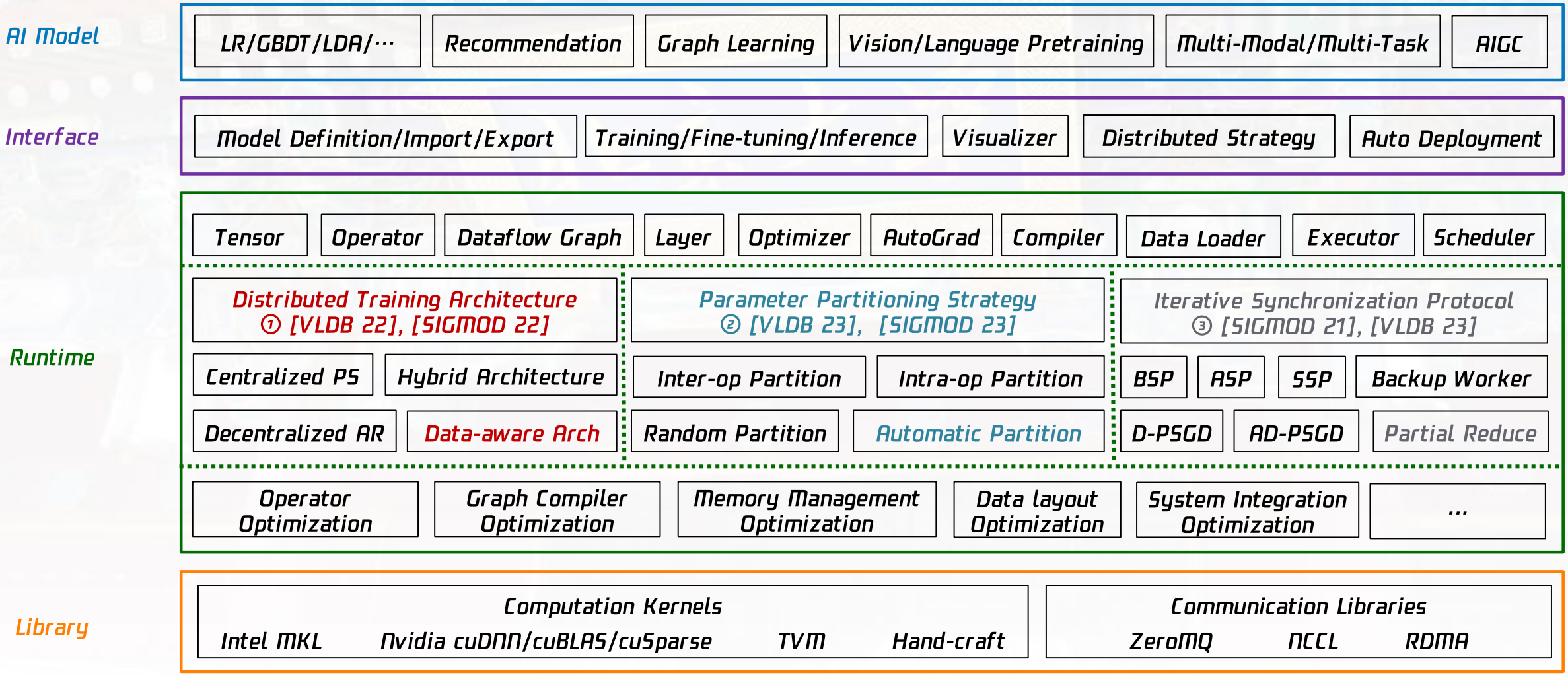
标注式

- GShard: split/replicate/shard
- Mindspore: shard/pipeline_stage
- Whale: split/replica/stage

怎么做大规模分布式深度学习框架?

Why Hetu?

- 系统功能性
  - 支持所有主流通信架构、并行模式、同步协议以及常见优化方案
- 系统复杂性
  - 支持统一分布式计算图中间表达，编译后适配多种通信算子
- 系统易用性
  - 支持半自动以及自动并行模式，硬件自适应感知最优分布式部署方案



Hetu https://github.com/PKU-DAIR/Hetu

AI Model

LR/GBDT/LDA/… | Recommendation | Graph Learning | Vision/Language Pretraining | Multi-Modal/Multi-Task | AIGC

Interface

Model Definition/Import/Export | Training/Fine-tuning/Inference | Visualizer | Distributed Strategy | Auto Deployment

Runtime

Tensor | Operator | Dataflow Graph | Layer | Optimizer | AutoGrad | Compiler | Data Loader | Executor | Scheduler

Distributed Training Architecture
① [VLDB 22], [SIGMOD 22]

Parameter Partitioning Strategy
② [VLDB 23], [SIGMOD 23]

Iterative Synchronization Protocol
③ [SIGMOD 21], [VLDB 23]

Centralized | Worker

Decentraliz | Reduce

Operat Optimiza | …

Library

Intel MKL | …

亮点一：基于数据/通信模式感知的分布式训练架构

VLDB 2022 — HET
通用Embedding模型

ACM SIGMOD PODS 2022 Philadelphia, PA, USA — HET-GMP
广告推荐模型

ICDE 2022 Kuala Lumpur | Malaysia — HET-KE
知识图谱

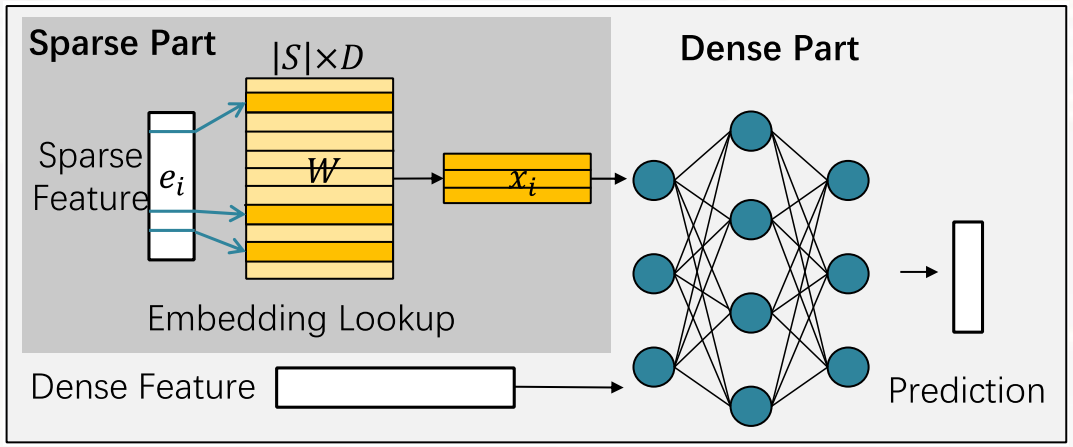
VLDB 2022 — CELU-VFL
纵向联邦学习

Hetu https://github.com/PKU-DAIR/Hetu

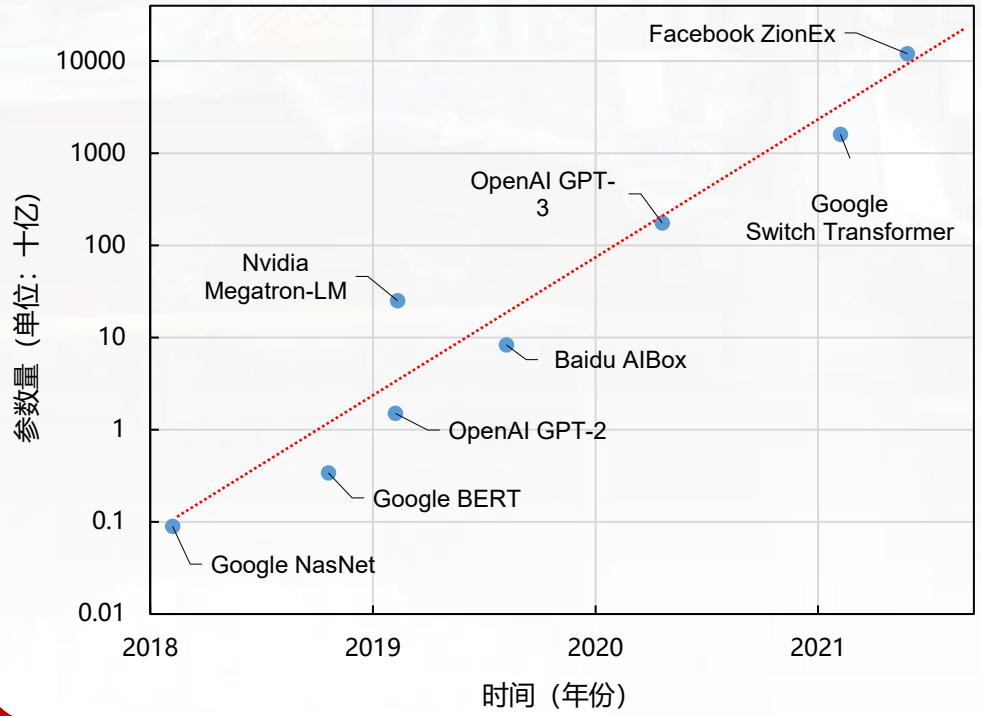
亮点一：基于数据/通信模式感知的分布式训练架构

- 高维大规模嵌入（Embedding）模型

嵌入模型结构



Facebook DLRM模型规模



- 典型嵌入模型：
  - 深度学习推荐模型 (DLRM)
  - 图神经网络 (GNN)

亮点一：基于数据/通信模式感知的分布式训练架构

■ 大规模嵌入模型训练框架

Parallax, EuroSys19
SNU

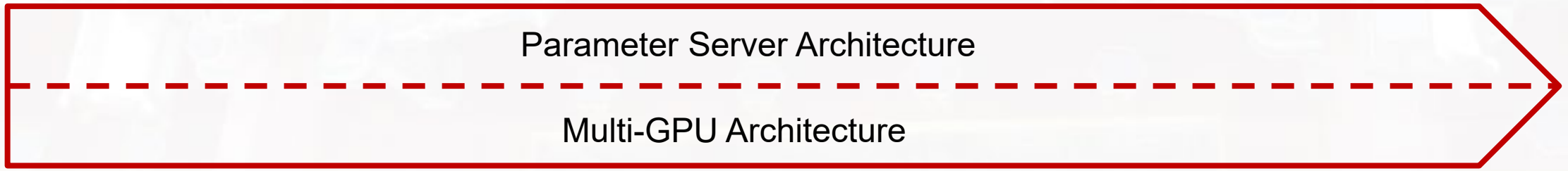
HierPS, MLSys19
Baidu

Kraken, SC20
Kuaishou

ZionEX, ISCA22
Facebook

HET, VLDB22
PKU & Tencent

Persia, KDD22
ETH & Kuaishou



AIBox, CIKM19
Baidu

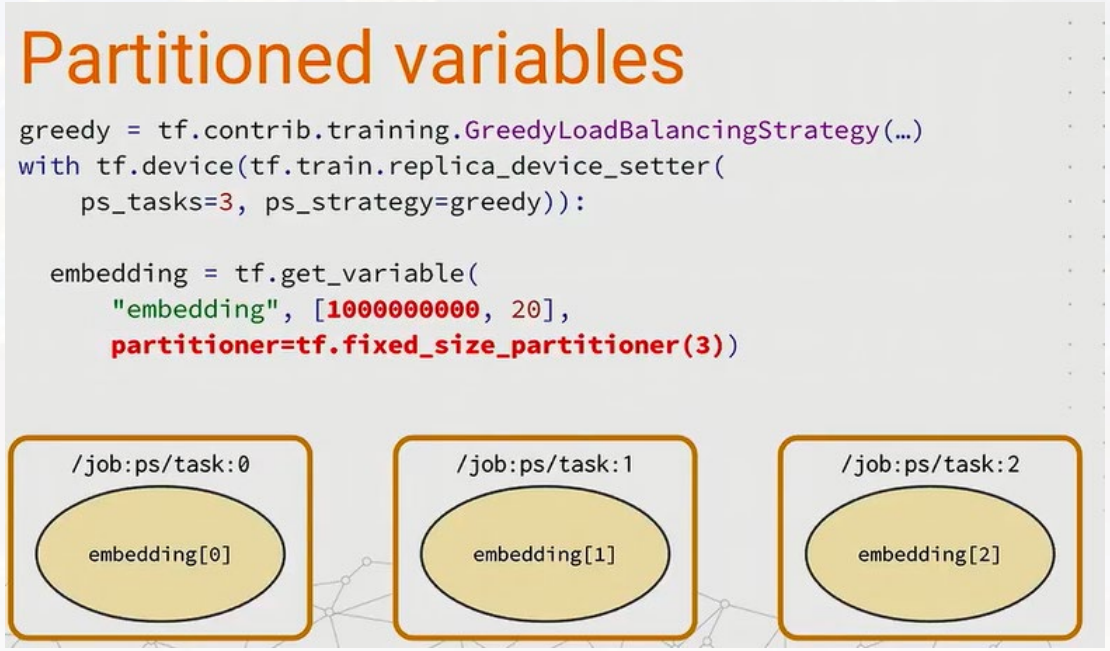
HugeCTR, IRS20
Nvidia

ScaleFreeCTR, SIGIR21
Huawei

HET-GMP, SIGMOD22
PKU

HET: 基于参数缓存的大规模稀疏嵌入模型训练框架

- 基于参数服务器的嵌入模型训练：
  - 模型参数中的嵌入表被参数服务器切割开，分布式存储
  - 计算节点通过稀疏推送/稀疏拉取操作访问服务器上的嵌入表
  - 代表系统：TensorFlow、Parallax



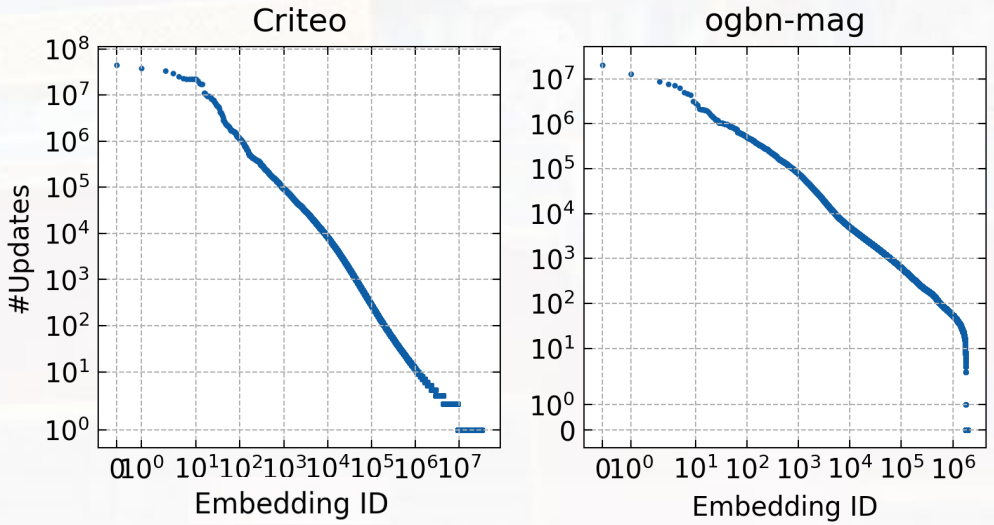
TensorFlow中的Partitioned Embedding Table

- 挑战：高维大规模嵌入模型的分布式训练面临严重的**通信瓶颈**
  - 例：10亿参数规模的DLRM模型在实验环境下70%时间都用于分布式通信

Miao, Xupeng, et al. HET: Scaling out Huge Embedding Model Training via Cache-enabled Distributed Framework. **VLDB 2022** *(Best Scalable Data Science Paper Award)*

HET: 基于参数缓存的大规模稀疏嵌入模型训练框架

- 如何解决上述挑战?
  - 细粒度参数缓存
    - 嵌入表更新具有稀疏性
    - 嵌入向量访问频率通常遵循幂律分布
    - 减少网络通信
  - 参数异步控制
    - 允许读取版本稍旧的嵌入向量
    - 允许稍晚写回嵌入向量的梯度
    - 保证收敛质量
- 如何实现?

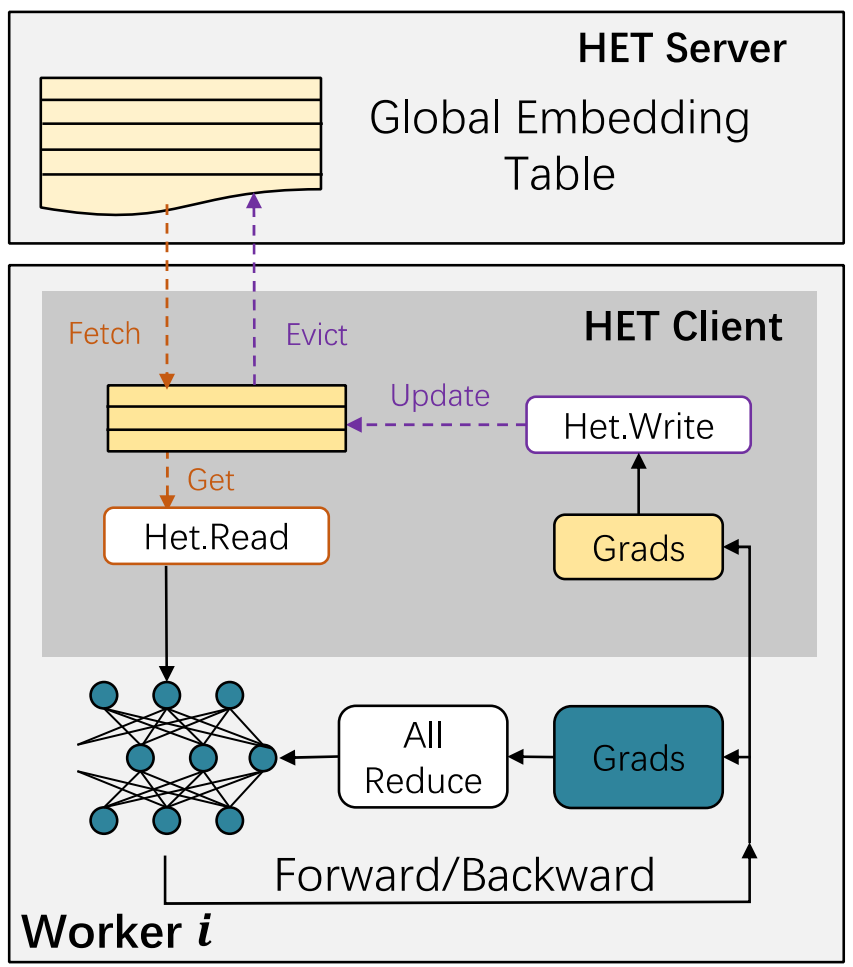


真实数据集Embedding访问分布情况

Miao, Xupeng, et al. HET: Scaling out Huge Embedding Model Training via Cache-enabled Distributed Framework. **VLDB 2022** *(Best Scalable Data Science Paper Award)*

基于模型并行的大规模嵌入模型训练框架

- 嵌入缓存管理机制
- 例： Het.Read(keys)
  - 读取缓存时检查嵌入向量版本
  - 保证缓存版本与服务器上全局版本相差不超过设定的阈值
  - 检查通过的嵌入向量可直接对缓存访问（Get）而无需访问服务器（Fetch）



基于模型并行的大规模嵌入模型训练框架

- 理论证明，在有限版本差控制内，基于嵌入缓存的方法可以达到和标准训练方法接近的模型收敛质量

THEOREM 1 (GLOBAL CONVERGENCE RATE). *Consider an arbitrary objective function f, under the per-key-clock-bounded consistency model and Assumptions 1 and 2. Given the success parameter $\epsilon > 0$, a constant learning rate value*

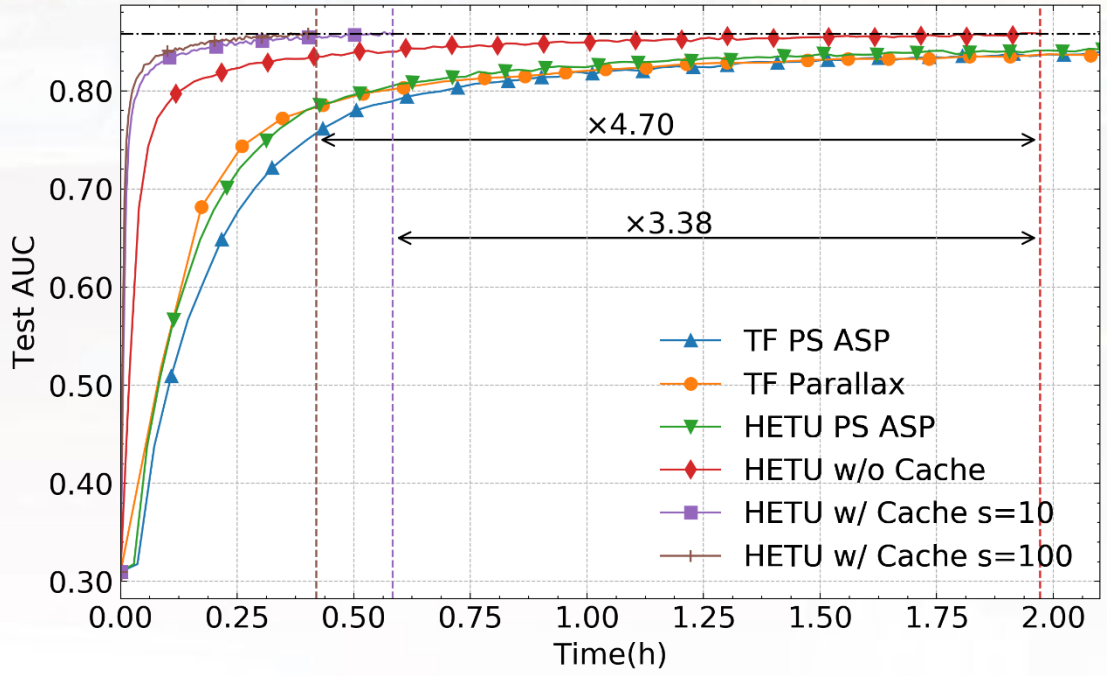
$$\eta \leq \min\left(\frac{\sqrt{\epsilon}}{4\sqrt{3}sLMB}, \frac{\sqrt{\epsilon}}{4\sqrt{L}sMB}, \frac{\epsilon}{12M^2B^2L}\right)$$

and $T = \Theta\left(\frac{f(x(0))-f_{inf}}{\epsilon\eta}\right)$ *iterations, for worker j, we are guaranteed to reach some iterate* $\mathbf{x}^j(t^\star)$ *with* $1 \leq t \leq T$ *such that*

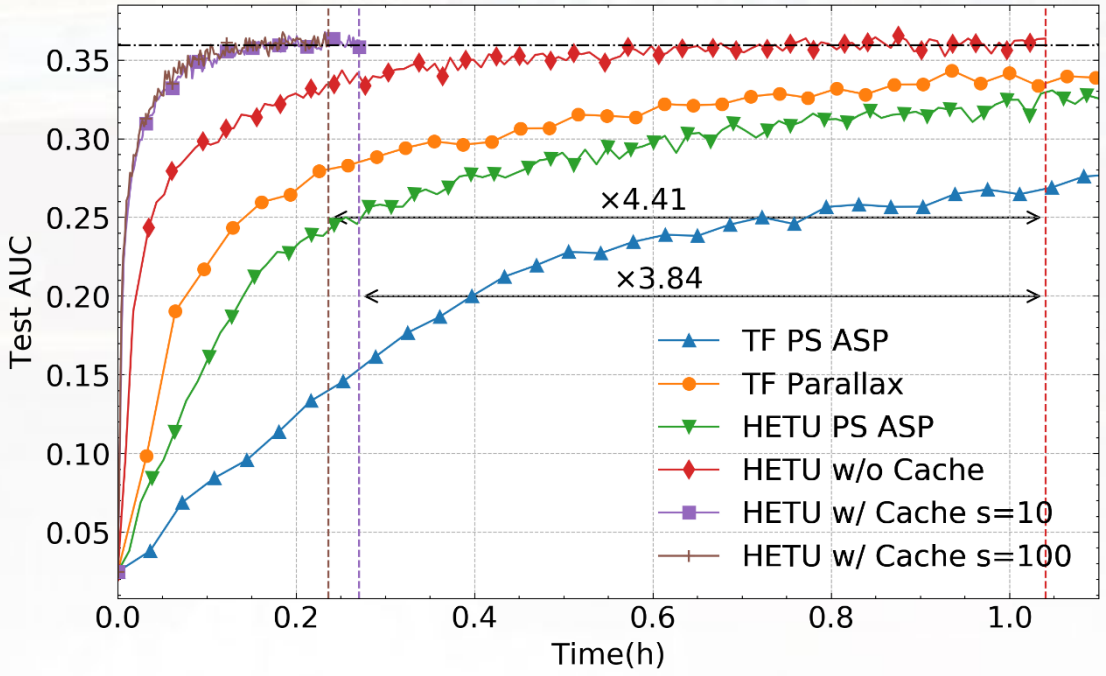
$$\mathbb{E}\|\nabla f(\mathbf{x}^j(t^\star))\|^2 \leq \epsilon.$$

基于模型并行的大规模嵌入模型训练框架

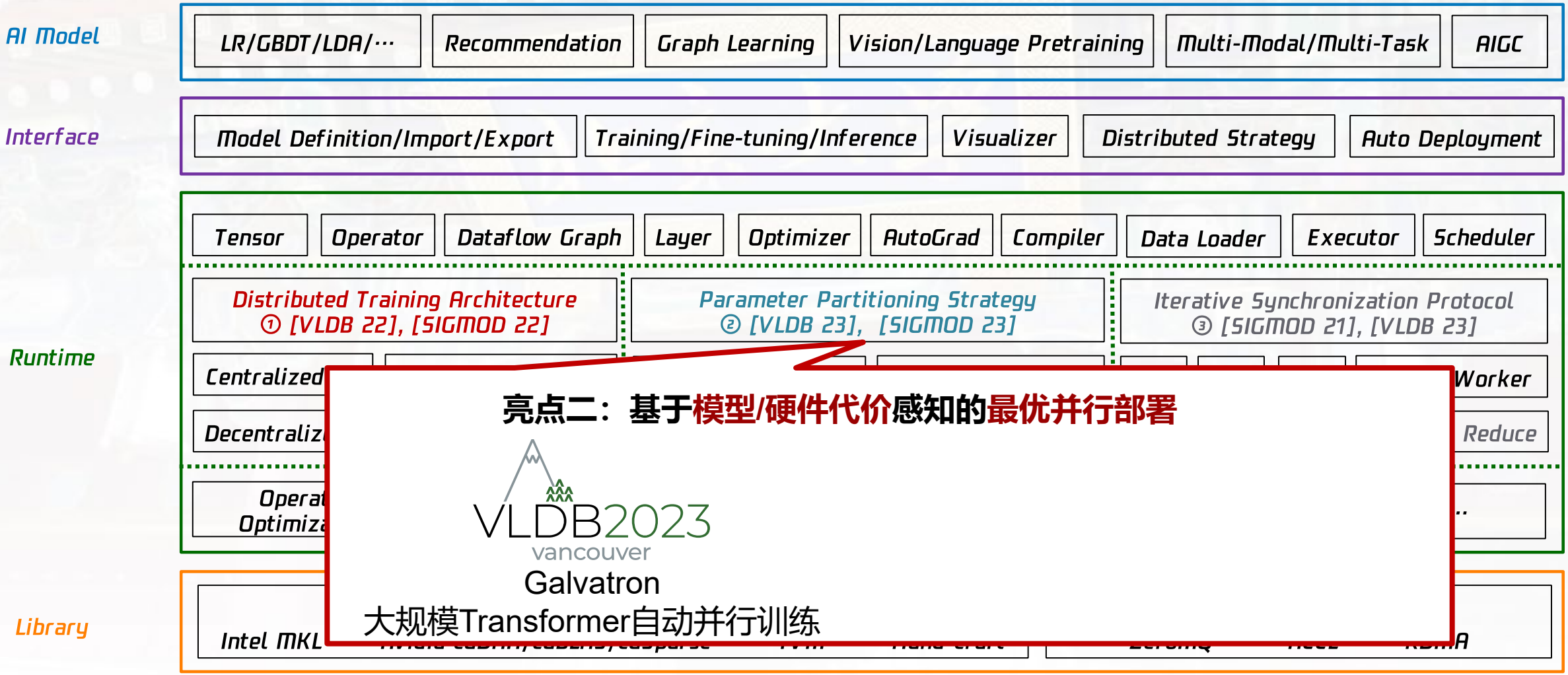
- 实验结果：相比于SOTA方法，可以减少88%的Embedding通信，实现20.7倍的训练加速。



Wide & Deep on Criteo



GNN on OGBN-MAG



亮点二：基于模型/硬件代价感知的最优并行部署

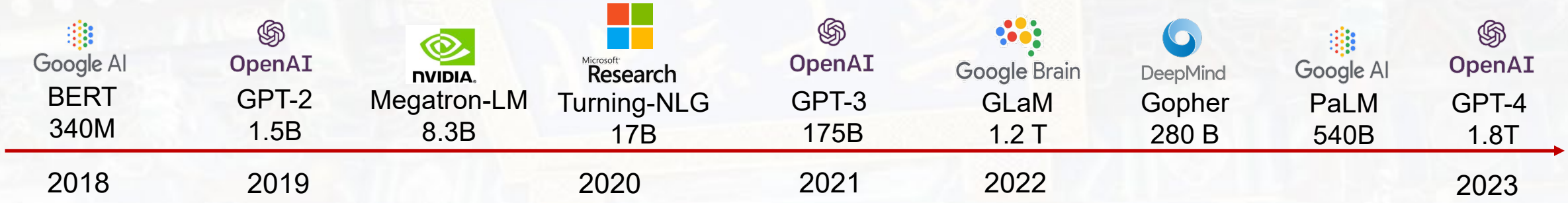
VLDB2023 vancouver

Galvatron

大规模Transformer自动并行训练

Hetu https://github.com/PKU-DAIR/Hetu

Galvatron: 大规模Transformer模型自动并行高效分布式训练



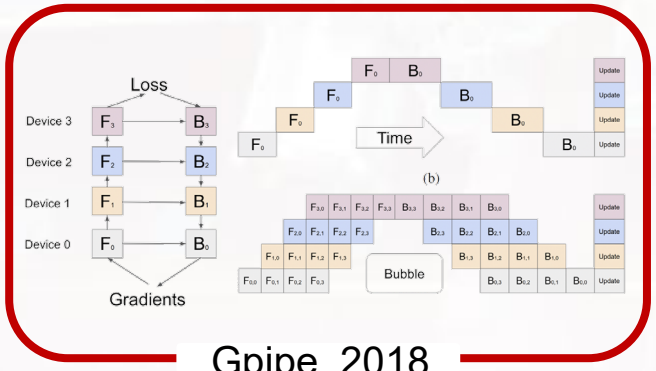
"Larger Transformers, higher model performance!"

—— **Scaling Laws for Large Models**

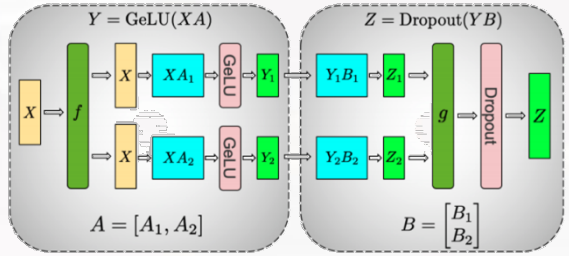
Miao et al. Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism. **VLDB 2023**

Galvatron: 大规模Transformer模型自动并行高效分布式训练

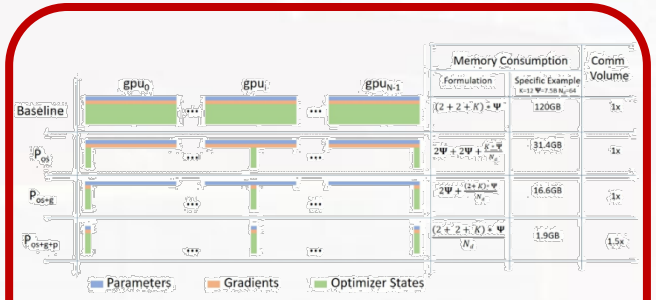
2018	2019		2020	2021	2022			2023
Google AI BERT 340M	OpenAI GPT-2 1.5B	NVIDIA Megatron-LM 8.3B	Microsoft Research Turning-NLG 17B	OpenAI GPT-3 175B	Google Brain GLaM 1.2 T	DeepMind Gopher 280 B	Google AI PaLM 540B	OpenAI GPT-4 1.8T



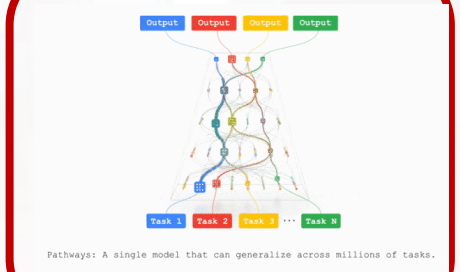
Gpipe, 2018



Megatron, 2019



DeepSpeed, 2020



Pathways, 2022

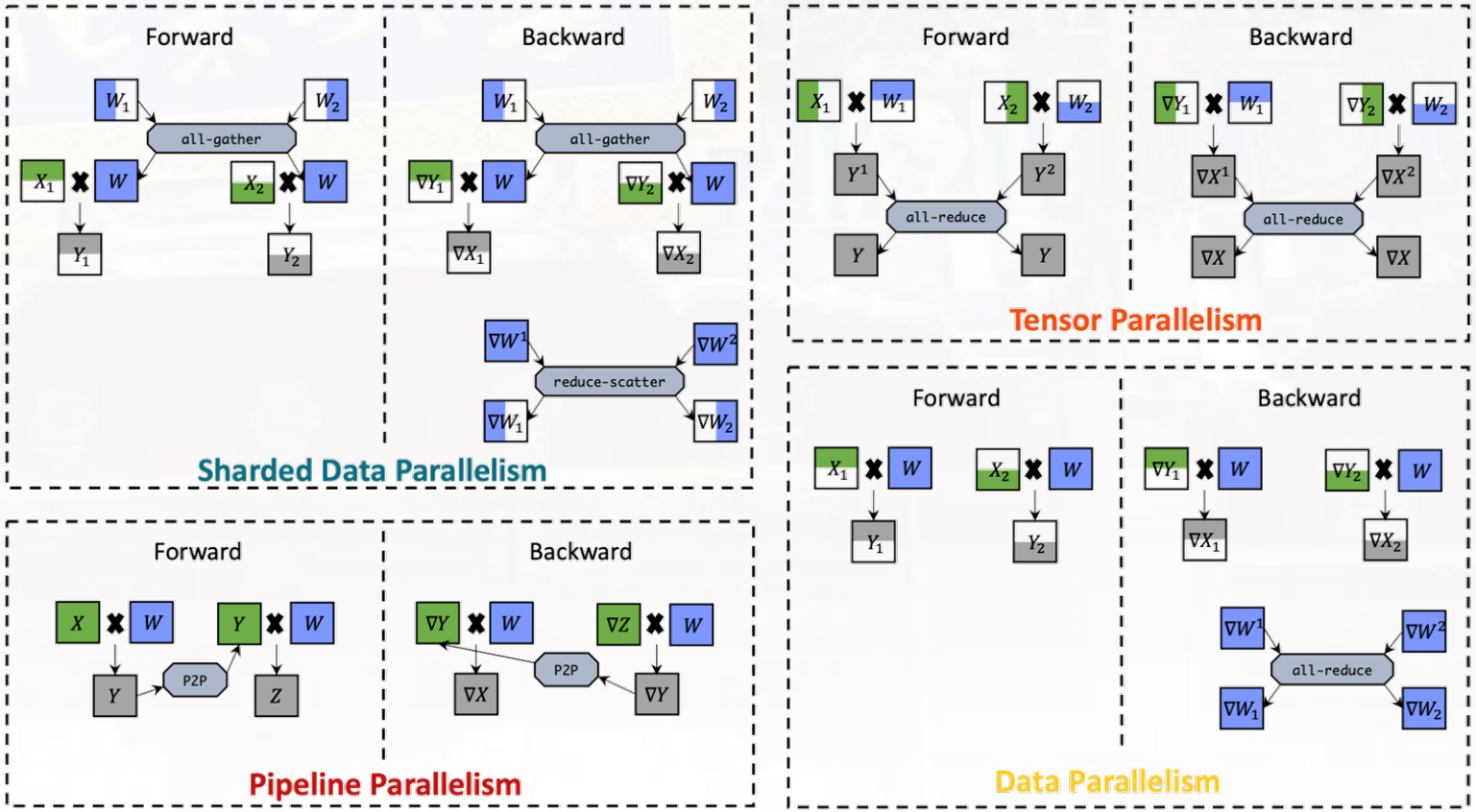
The development of distributed DL systems are the foundations of large models!

Miao et al. Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism. **VLDB 2023**

Galvatron: 大规模Transformer模型自动并行高效分布式训练

- 现有工具:
  - NVIDIA Megatron-LM
  - Microsoft DeepSpeed
  - Meta FairSeq
- 缺乏自动化部署能力
  - 模型配置 (e.g., hidden size)
  - 模型结构 (e.g., layer)
  - 硬件算力 (e.g., V100, A100)
  - 显存空间 (e.g., 8GB, 32GB)
  - 通信带宽 (e.g., PCIe, NVLink)

依赖专家经验+人工调试
探索最优部署方案，费事费力!



Miao et al. Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism.
VLDB 2023

Galvatron: 大规模Transformer模型自动并行高效分布式训练

- PyTorch原生支持，一键实现大模型高效自动并行

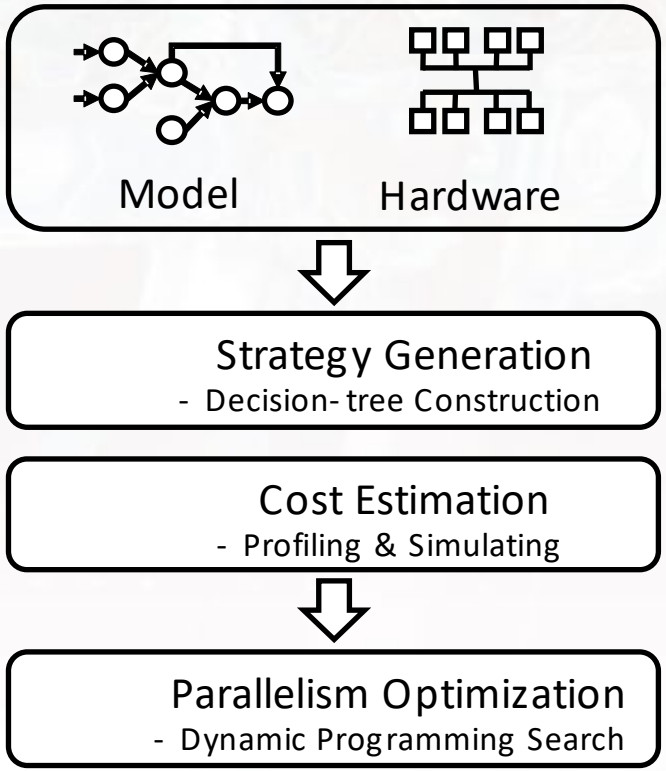
Parallelism	Data		Mode		Automatic
	DP	SDP	TP	PP	
PyTorch DDP [VLDB'22]	√	×	×	×	×
DeepSpeed ZeRO-3 [SC'20]	√	√	×	×	
Megatron [arXiv'19]	√	×	√	×	
PyTorch Gpipe [NeurIPS'19]	×	×	×	√	
DeepSpeed 3D [arXiv'22]	√	×	√	√	
FlexFlow [MLSys'19]	√	×	√	×	√
Tofu [EuroSys'19]	√	×	√	×	
PipeDream [SOSP'19]	√	×	×	√	
DAPPLE [PPoPP'21]	√	×	×	√	
Unity [OSDI'22]	√	×	√	√	
Alpa [OSDI'22]	√	×	√	√	
Galvatron [VLDB'23]	√	√	√	√	

- 多样性:
  - 支持并行维度最多
  - 支持多种复杂模型
  - 支持不同硬件环境
- 复杂性:
  - 搜索空间剪枝
- 实用性:
  - 代价精准建模

Miao et al. Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism.
VLDB 2023

Galvatron: 大规模Transformer模型自动并行高效分布式训练

- PyTorch原生支持，一键实现大模型高效自动并行



Galvatron User Script

- with Runtime

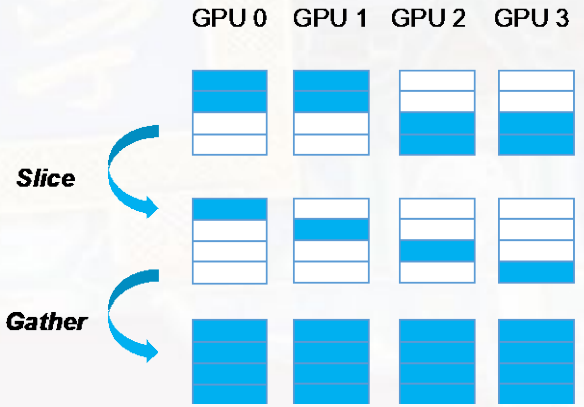
```
  import torch
+ from hetu import galvatron
  ......
+ opt_parallel_configs= galvatron.get_configs(args)
  # Distributed Model Definition
  model = myModel(model_config)
+ model = galvatron.construct_parallelmodel(
+     model,
+     model_config,
+     training_args,
+     opt_parallel_configs,
+ )
  # Optimizer Definition
  optim = Optimizer(model.parameters(), lr=lr)
  # Model Training
  for e in range(epochs):
      for batch in dataloader:
          model.zero_grad()
          outputs = model(batch[...inputs...])
          loss = loss_fn(outputs, batch[...label...])
          loss.backward()
          optim.step()
```

- 多样性:
  - 支持并行维度最多
  - 支持多种复杂模型
  - 支持不同硬件环境
- 复杂性:
  - 搜索空间剪枝
- 实用性:
  - 代价精准建模

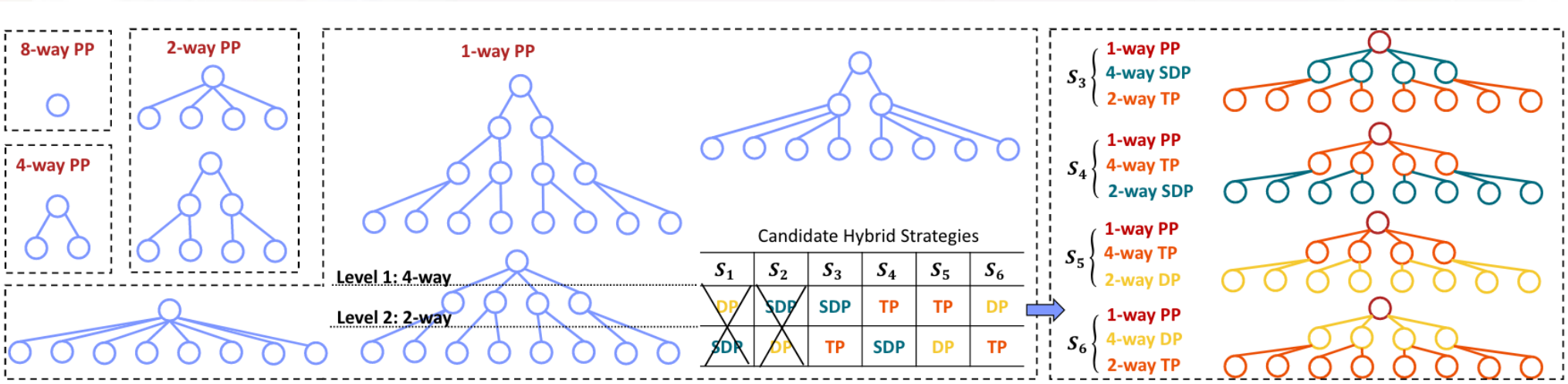
Miao et al. Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism.
VLDB 2023

Galvatron: 大规模Transformer模型自动并行高效分布式训练

- 核心技术
  - 基于决策树的搜索空间分解
  - 基于动态规划的并行优化算法
  - 基于混合建模的执行代价估计方法



跨策略间
分布式张量
状态自动转换



Miao et al. Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism.
VLDB 2023

Galvatron: 大规模Transformer模型自动并行高效分布式训练

- 模型：BERT、T5、ViT、Swin Transformer
- 相比于现有系统吞吐性能提升200%～338%
- 相比于受限并行策略空间性能提升28%～55%

Table 3: Comparison with 16 GPUs.

Memory	Strategy	BERT-Huge-32	BERT-Huge-48	ViT-Huge-32	ViT-Huge-48
8G	PyTorch DDP (DP)	OOM	OOM	OOM	OOM
	Megatron (TP)	OOM			
	PyTorch GPipe (PP)	13.79 (16)			
	FSDP/ZeRO-3 (SDP)	8.95 (16)			
	DeepSpeed 3D	**15.24** (16)			
	Galvatron (DP+TP)	OOM	OOM	54.43 (64)	24.56 (32)
	Galvatron (DP+PP)	13.91 (16)	5.88 (8)	68.56 (128)	35.02 (72)
	Galvatron (ours)	**15.24** (16)	**8.43 (16)**	**76.74** (128)	**38.32** (88)
16G	PyTorch DDP (DP)	12.14 (16)	OOM	88.06 (128)	OOM
	Megatron (TP)	6.12 (16)	4.23 (16)	17.11 (64)	11.26 (48)
	PyTorch GPipe (PP)	23.29 (40)	12.92 (24)	69.72 (320)	50.23 (208)
	FSDP/ZeRO-3 (SDP)	30.37 (64)	11.74 (32)	123.95 (320)	61.49 (224)
	DeepSpeed 3D	23.92 (48)	13.03 (24)	91.56 (256)	53.81 (152)
	Galvatron (DP+TP)	23.01 (32)	10.50 (16)	99.22 (160)	49.82 (96)
	Galvatron (DP+PP)	23.73 (40)	13.12 (40)	115.88 (224)	61.38 (208)
	Galvatron (ours)	**32.67** (64)	**14.74** (40)	**131.15** (320)	**72.74** (208)

Table 4: Comparison with 64 GPUs.

Memory	Strategy	BERT-xHuge	ViT-xHuge
16G	PyTorch DDP (DP)	OOM	OOM
		0.68 (3)	1.94 (12)
		9.74 (16)	61.95 (96)
		OOM	OOM
		8.44 (16)	64.91 (96)
	Galvatron (DP+TP)	1.73 (4)	5.07 (2)
	Galvatron (DP+PP)	9.74 (16)	64.83 (104)
	Galvatron (ours)	**13.77** (24)	**68.35 (136)**
32G	PyTorch DDP (DP)	OOM	OOM
	Megatron (TP)	0.77 (7)	2.11 (28)
	PyTorch GPipe (PP)	21.38 (48)	94.84 (288)
	FSDP/ZeRO-3 (SDP)	OOM	OOM
	DeepSpeed 3D	21.28 (40)	91.19 (256)
	Galvatron (DP+TP)	1.73 (4)	5.51 (68)
	Galvatron (DP+PP)	23.64 (48)	110.98 (232)
	Galvatron (ours)	**27.49** (64)	**114.55 (328)**

Miao et al. Galvatron: Efficient Transformer Training over Multiple GPUs Using Automatic Parallelism.
VLDB 2023

AI Model

LR/GBDT/LDA/… | Recommendation | Graph Learning | Vision/Language Pretraining | Multi-Modal/Multi-Task | AIGC

Interface

Model Definition/Import/Export | Training/Fine-tuning/Inference | Visualizer | Distributed Strategy | Auto Deployment

Runtime

Tensor | Operator | Dataflow Graph | Layer | Optimizer | AutoGrad | Compiler | Data Loader | Executor | Scheduler

Distributed Training Architecture
① [VLDB 22], [SIGMOD 22]

Parameter Partitioning Strategy
② [VLDB 23], [SIGMOD 23]

Iterative Synchronization Protocol
③ [SIGMOD 21], [VLDB 23]

Centralized … Worker

Decentraliz… Reduce

Operat… Optimiza… …

Library

Intel MKL … A

亮点二：基于模型/硬件代价感知的最优并行部署

VLDB2023 vancouver

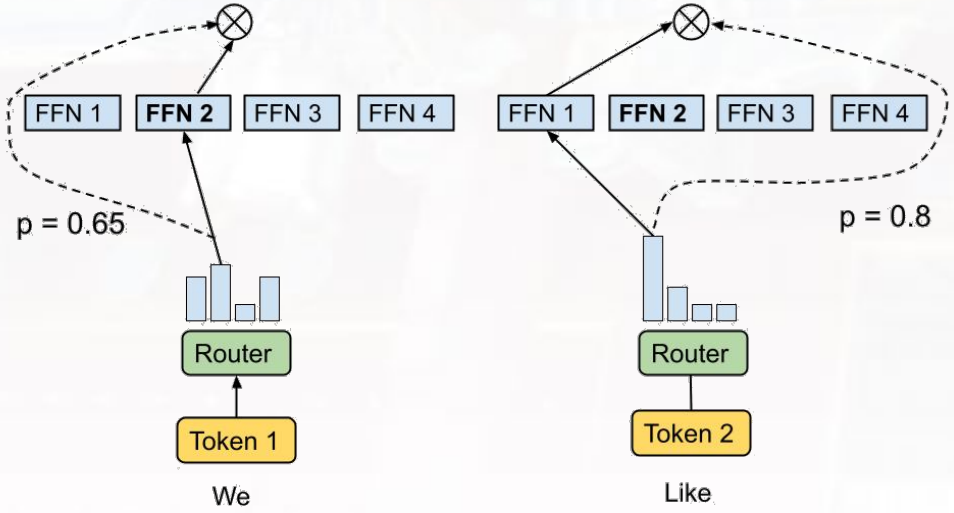
Galvatron
大规模Transformer自动并行训练

FlexMoE
多专家混合模型动态负载均衡

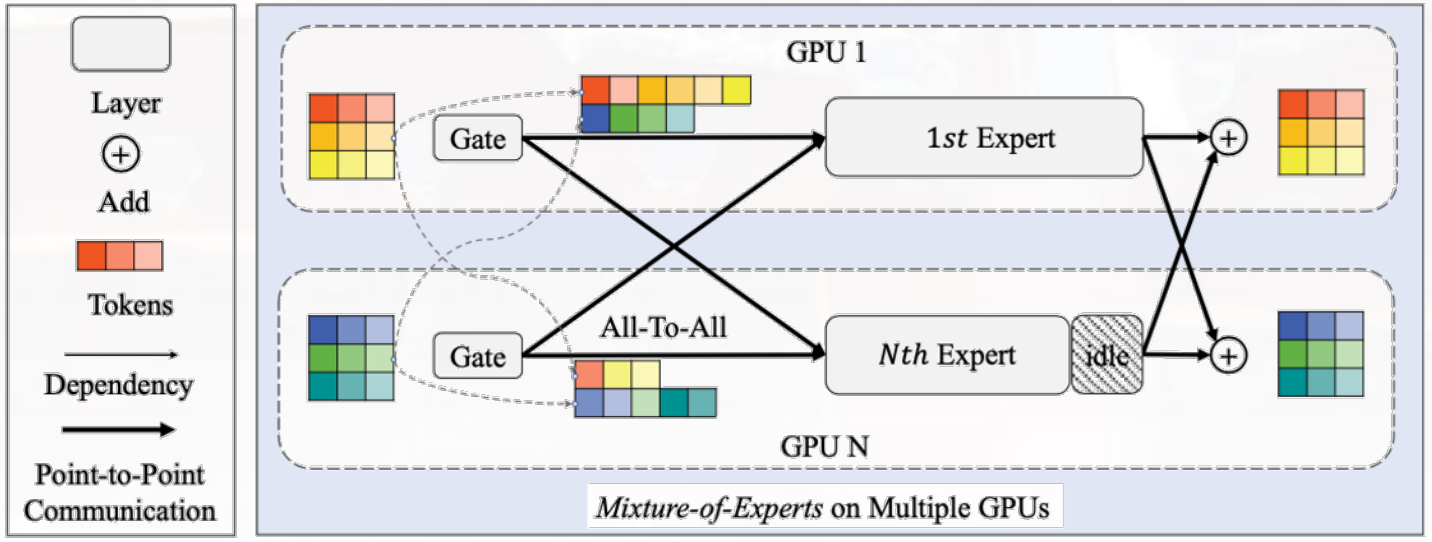
Hetu https://github.com/PKU-DAIR/Hetu

FlexMoE: 基于动态专家并行的大规模MoE模型分布式训练

- MoE模型分布式训练方案：专家并行
  - 专家网络均匀分配到多个GPU（专家并行），非专家层复制在每个GPU上（数据并行）
  - 数据通过GPU间AlltoAll通信发送给门控网络制定的专家网络



MoE模型结构示意图

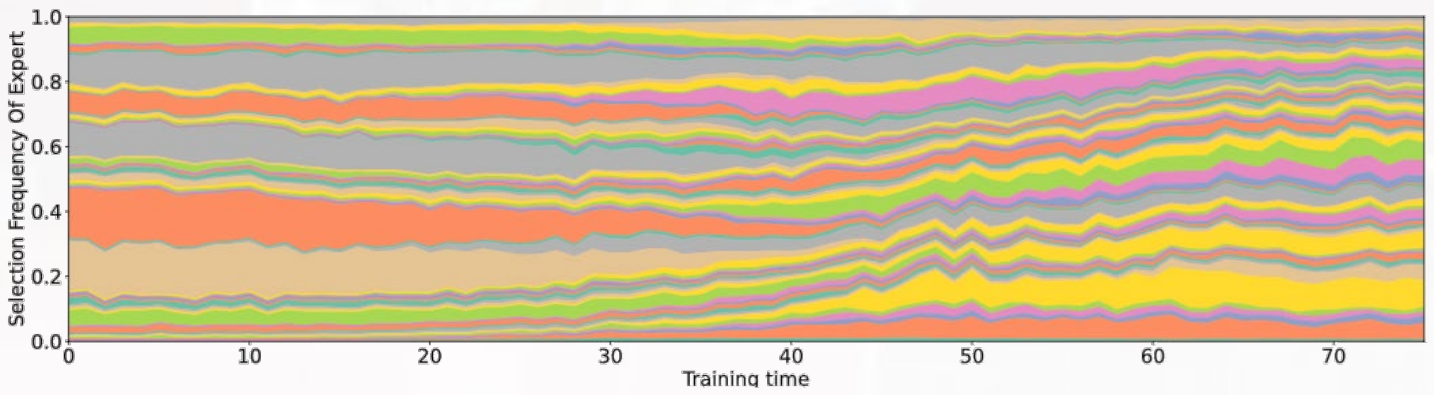


专家并行示意图

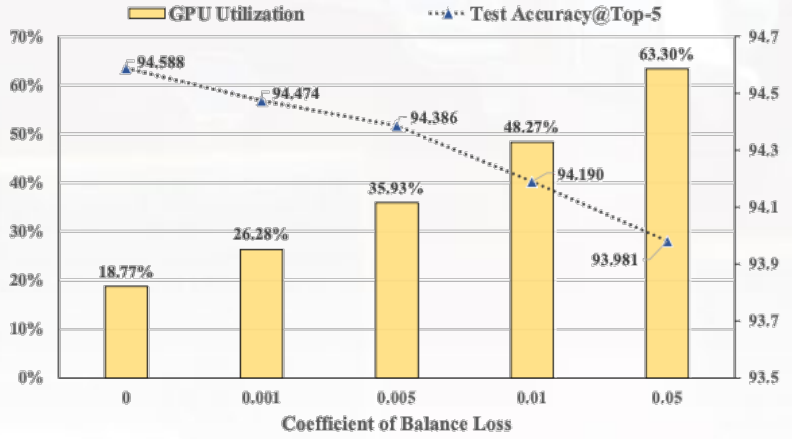
Nie et al. FlexMoE: Scaling Large-scale Sparse Pre-trained Model Training via Dynamic Device Placement. SIGMOD 2023

FlexMoE: 基于动态专家并行的大规模MoE模型分布式训练

- 由于专家间负载不均衡且动态变化（左图），MoE模型的专家并行训练面临严重的**掉队者问题**
- 现有系统通过修改MoE模型定义来提升吞吐。代表系统：DeepSpeed
  - 方式：1. 增加Balance Loss优化负载的不均衡程度；2. 限制每个专家处理数据的上限
  - 结果：模型越均衡，系统吞吐越高，但模型精度越低（右图）



训练过程中，不同专家接收到的负载比例。
注：阴影部分代表专家的负载



不同均衡程度负载下模型精度和GPU利用率

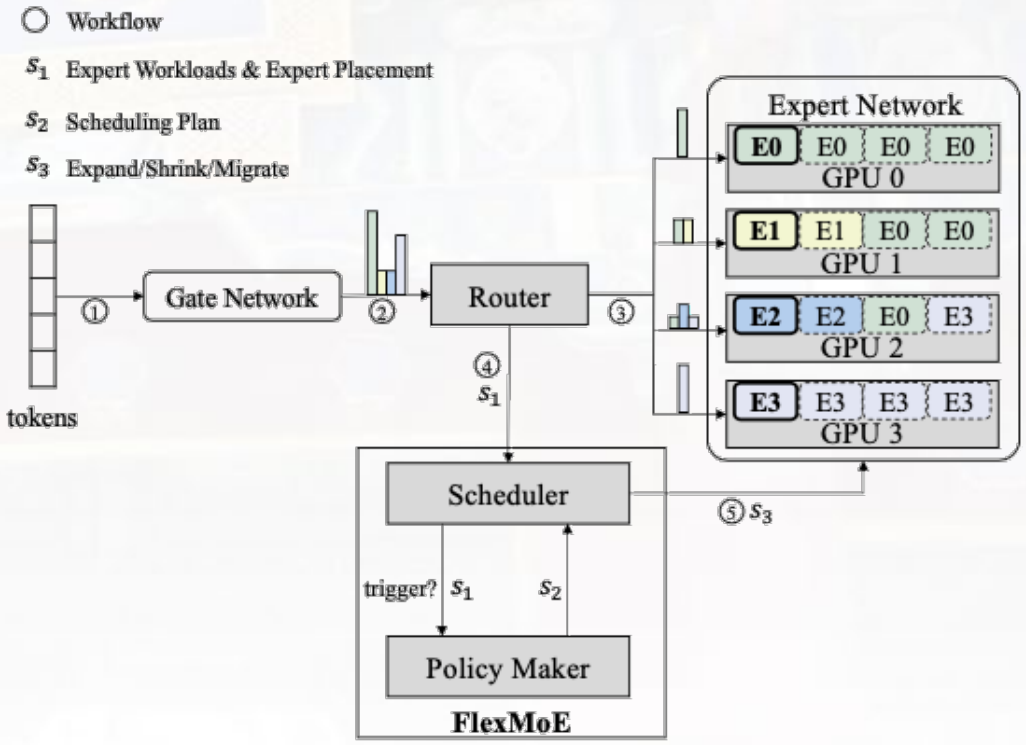
Nie et al. FlexMoE: Scaling Large-scale Sparse Pre-trained Model Training via Dynamic Device Placement. SIGMOD 2023

FlexMoE: 基于动态专家并行的大规模MoE模型分布式训练

- 挑战：如何在尽可能保持模型精度的情况下最大化系统吞吐?

- 核心思路
  - 专家负载分布动态变化，连续且缓慢
  - 在训练过程中自适应调整专家网络摆放方案

- 系统设计
  - 实时监控负载并决策是否调整专家摆放
  - 基于vExpert的细粒度专家并行
  - 自适应调整专家摆放方案

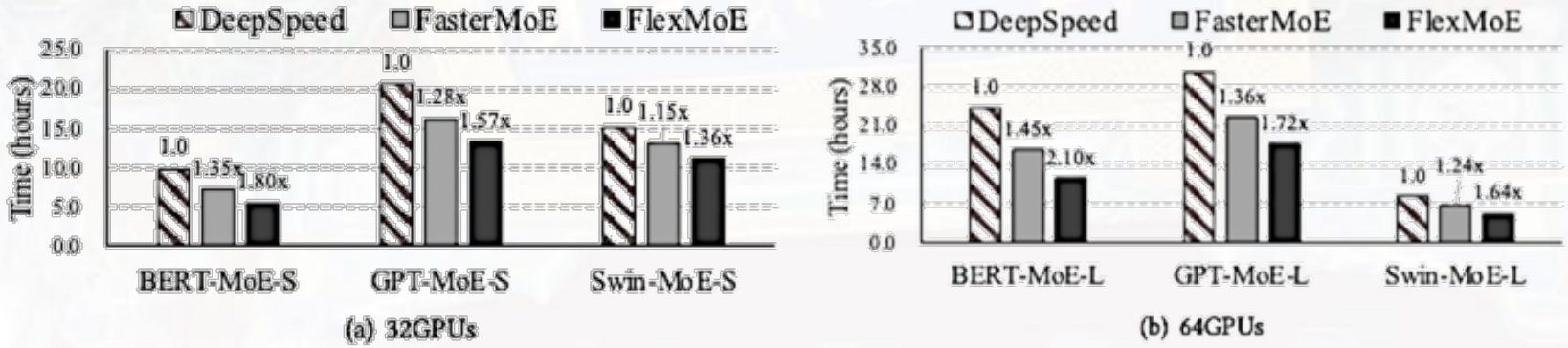


系统流程示意图

Nie et al. FlexMoE: Scaling Large-scale Sparse Pre-trained Model Training via Dynamic Device Placement. SIGMOD 2023

FlexMoE: 基于动态专家并行的大规模MoE模型分布式训练

- 实验结果：相比于SOTA方法，可以实现1.18-2.10倍的加速。



收敛到相同精度所需的时间

	Metric	Masked Language Modeling		Language Modeling		Metric	Image Classification	
		BERT-MoE-S	BERT-MoE-L	GPT-MoE-S	GPT-MoE-L		Swin-MoE-S	Swin-MoE-L
DeepSpeed	PPL ↓	3.53	3.31	12.2	10.71	acc@1 ↑	77.316	77.022
						acc@5 ↑	93.838	93.642
FlexMoE	PPL ↓	3.14	3.07	11.72	10.47	acc@1 ↑	77.754	77.109
						acc@5 ↑	94.042	93.663

模型收敛精度的对比
修改模型结构 V.S. 修改系统实现

Nie et al. FlexMoE: Scaling Large-scale Sparse Pre-trained Model Training via Dynamic Device Placement. SIGMOD 2023

AI Model

LR/GBDT/LDA/… | Recommendation | Graph Learning | Vision/Language Pretraining | Multi-Modal/Multi-Task | AIGC

Interface

Model Definition/Import/Export | Training/Fine-tuning/Inference | Visualizer | Distributed Strategy | Auto Deployment

Runtime

Tensor | Operator | Dataflow Graph | Layer | Optimizer | AutoGrad | Compiler | Data Loader | Executor | Scheduler

Distributed Training Architecture
① [VLDB 22], [SIGMOD 22]

Parameter Partitioning Strategy
② [VLDB 23], [SIGMOD 23]

Iterative Synchronization Protocol
③ [SIGMOD 21], [VLDB 23]

Centralized | Worker

Decentraliz | Reduce

Operat
Optimiza

Library

Intel MKL

亮点三：基于同构/异构环境感知的迭代同步协议

Partial Reduce
数据并行下的局部规约方法

VLDB2023
vancouver

SDPipe
数据+流水并行的半去中心化同步协议

Hetu https://github.com/PKU-DAIR/Hetu

AI Model

LR/GBDT/LDA/… | Recommendation | Graph Learning | Vision/Language Pretraining | Multi-Modal/Multi-Task | AIGC

Interface

Model Definition/Import/Export | Training/Fine-tuning/Inference | Visualizer | Distributed Strategy | Auto Deployment

Runtime

Tensor | Scheduler

Distribu… ① [V… | …rotocol [23]

Centralize… | …Worker

Decentrali… | …al Reduce

Operator Optimization | Graph Compiler Optimization | Memory Management Optimization | Data layout Optimization | System Integration Optimization | …

亮点四：基于混合/多级存储感知的张量存储管理

ICDE 2022 Kuala Lumpur | Malaysia

TSplit
CPU内存+GPU显存管理优化

腾讯太极机器学习平台

AngelPTM
SSD/CPU/GPU多级存储联合优化

Library

Computation Kernels: Intel MKL | Nvidia cuDNN/cuBLAS/cuSparse | TVM | Hand-craft

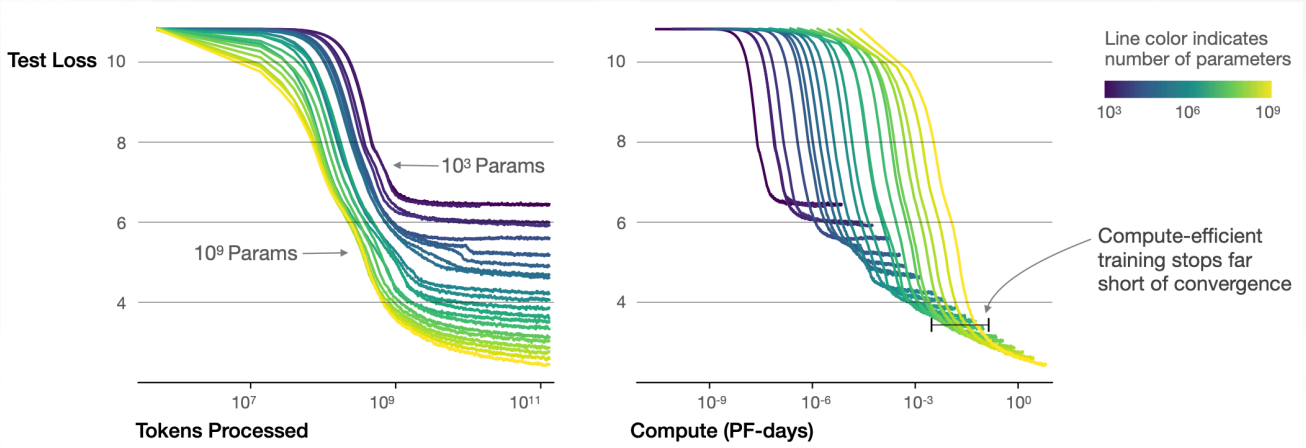
Communication Libraries: ZeroMQ | NCCL | RDMA

Hetu https://github.com/PKU-DAIR/Hetu

AngelPTM: 基于多级存储的企业级预训练大模型训练框架

- OpenAI提出了语言模型的缩放法则：
  - 更大的模型在相同数据量下能够实现更好的模型质量。
  - 更大的模型达到相同的模型质量需要更少的计算量。
  - 模型质量 ∝ 模型参数量 × 数据量 × 计算量

⟹ 模型规模越来越大



语言模型的缩放法则[1]

[1] Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models[J]. arXiv 2022. https://arxiv.org/pdf/2001.08361.pdf

AngelPTM: 基于多级存储的企业级预训练大模型训练框架

Tencent

- 工业生产中大模型训练相关任务：
  - 预训练(Pre-Training)：从大量的训练数据集中提取基础知识
    - 在 **数百-数千** 个GPU上训练，持续 **数周**，占总任务数的 **10%**。
    - 痛点：在大规模集群上长时间训练，任务故障中断的频率较高。
  - 微调(Fine-Tuning): 将模型与特定的下游任务进行对齐
    - 在 **数十-数百** 个GPU上进行训练，持续 **数小时**，占总任务数的 **90%**。
    - 痛点：由于微调的数据量较小，任务计算量小，而训练所需GPU数量多，导致GPU利用率低。

Nie et al. Angel-PTM: A Scalable and Economical Large-scale Pre-training System in Tencent..
VLDB 2023

AngelPTM: 基于多级存储的企业级预训练大模型训练框架

- 企业级预训练大模型训练框架的设计准则：
  - 易用性：用户只需修改几行代码即可使用
  - 扩展性：可简单扩展至更大模型和更大集群
  - 高效性：最大化GPU计算资源的使用，提高计算效率
  - 经济性：无需庞大的GPU资源，也可训练和微调大模型
  - 容错性：在机器故障导致可用GPU减少时，可迅速恢复训练

Nie et al. Angel-PTM: A Scalable and Economical Large-scale Pre-training System in Tencent..
VLDB 2023

AngelPTM: 基于多级存储的企业级预训练大模型训练框架

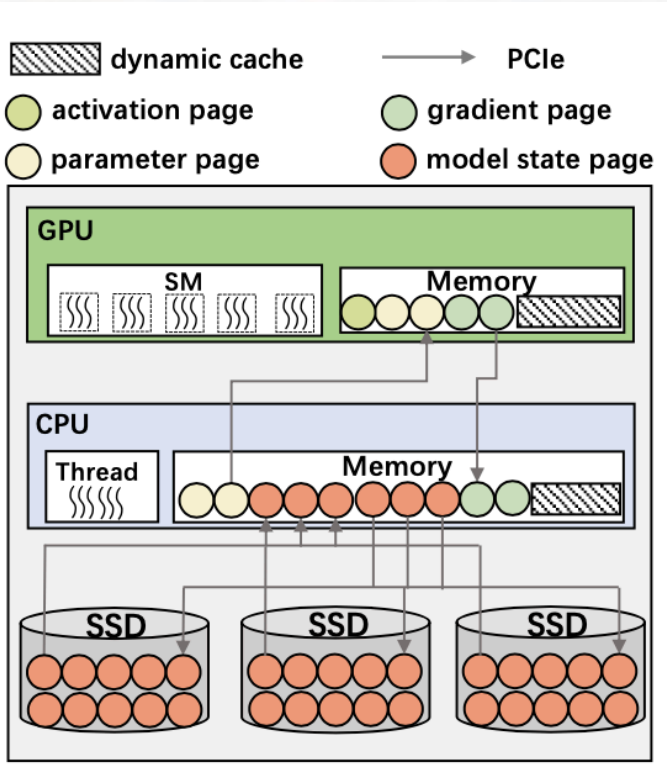
- 开源训练框架的解决方案:

	DeepSpeed 数据并行，DP + ZeRO + Offload	**Megatron-LM** 混合并行，DP + TP + PP
易用性	✓ 仅需修改几行代码	✗ 1. 需要修改大量的模型代码， 2.需要大量手动搜索最优的并行策略
扩展性	✓	✓
高效性	✗ 1. 当集群扩大到数千卡， 数据并行的集合通信效率降低 2.需要引入Offload支撑大模型，调度开销大	✓ 利用模型并行来缓解集合通信的衰减
经济性	✓ 利用CPU内存来容纳部分模型	✗ 需要足够的GPU来容纳模型， 如训练GPT3-175B需要128 A100-40GB
容错性	✓ 仅需修改checkpoint读取逻辑即可	✗ 需要重新搜索最优的并行策略

Nie et al. Angel-PTM: A Scalable and Economical Large-scale Pre-training System in Tencent. **VLDB 2023**

AngelPTM

- 为了满足以上的设计准则，AngelPTM的设计方案如下：
  - 利用 机内张量并行 + 机间数据并行的方案满足 易用性 扩展性 容错性
  - 利用 GPU显存、CPU内存和SSD存储共同支撑大模型训练 经济性
  - 提出以下优化方案来满足高效性： 高效性
    1. 细粒度的内存管理
       以Page为单位进行内存管理和操作，减少内存碎片数量
    2. 基于张量生存期的统一视角调度器
       利用迭代训练的特性进行按需调度，提升调度效率
    3. 无锁的参数更新方式
       通过引入异步的更新方案，消除SSD IO速度慢对训练的影响



系统架构图

Nie et al. Angel-PTM: A Scalable and Economical Large-scale Pre-training System in Tencent..
VLDB 2023

AngelPTM

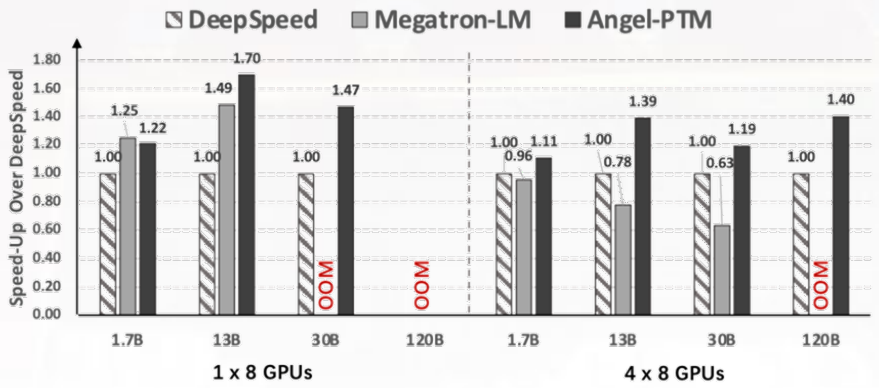
- 实验结果：相比SOTA方法，
  - 最大可训练的模型规模提高1.14倍
  - 模型训练吞吐提升88.9%
  - 随着GPU数量的增多，取得了线性的加速比
  - 通过引入Lock-Free的参数更新方式，在无模型质量损失的情况下，消除了SSD的IO瓶颈。

System	#Params	#GPUs	Samples/s	Valid Loss↓
AngelPTM	1T	64	37.26	1.124
	10T	576	317.82	**0.853**
+ *Lock-Free*	10T	576	**942.31**	0.861

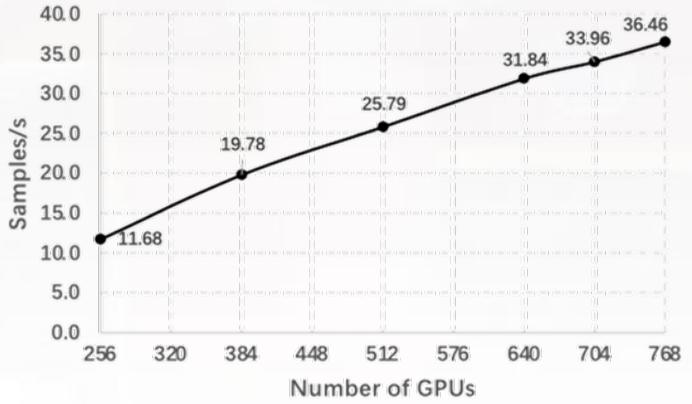
引入SSD训练后T5-MoE的训练性能

Model	System	#Params	#Batch	GPU Mem	Samples/s
GPT	DeepSpeed	28B	1	18	0.404
		28B	36	40	7.61
	AngelPTM	28B	38	39	**10.99**
		55B	1	33	0.464
		55B	10	40	3.34
T5	DeepSpeed	27B	1	20	0.317
		27B	32	39	7.31
	AngelPTM	27B	50	40	**14.38**
		58B	1	38	0.432
		58B	4	40	3.37

单机可训练的最大模型



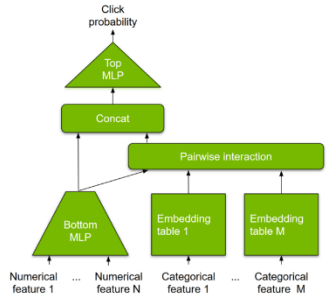
不同规模下的吞吐对比



GPT3-175B的扩展性实验

Nie et al. Angel-PTM: A Scalable and Economical Large-scale Pre-training System in Tencent..
VLDB 2023

亮点一：基于数据/通信模式感知的分布式训练架构



业界领先的稀疏大模型解决方案

HET v.s. *TensorFlow*
HET-GMP v.s. *NVIDIA HugeCTR*

亮点二：基于模型/硬件代价感知的最优并行部署

维度最多的自动模型并行调优

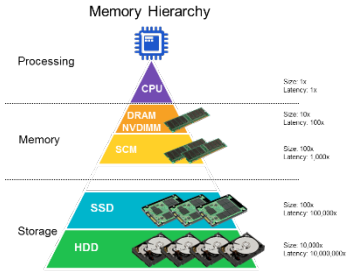
Galvatron v.s. *Megatron/DeepSpeed/...*
FlexMoE v.s. *DeepSpeed-MoE/FasterMoE/...*

Distributed Training
① *[VLDB 22], [SIGMOD 22]*

Parameter Partitioning
② *[VLDB 23], [SIGMOD 23]*

Iterative Synchronization Protocol
③ *[VLDB 23]*

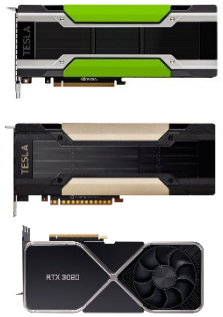
亮点四：基于混合/多级存储感知的张量存储管理



经济高效的低门槛大模型部署

TSplit v.s. *Checkpointing/Swapping/...*
AngelPTM v.s. *DeepSpeed-Zero*

亮点三：基于同构/异构环境感知的迭代同步协议



灵活多样的异构设备协同训练

Partial Reduce v.s. *All Reduce*
SDPipe v.s. *PipeDream/HetPipe/...*

Hetu https://github.com/PKU-DAIR/Hetu

系统开源影响力

- 机器之心「AI中国」2021年度开源事件TOP10
  - 唯一入选高校单位
- Pop SOTA！List for AI Developers 2021
  - 年度最受AI开发者认可的价值工作名录
- 2021 CCF 大数据与计算智能大赛 （BDCI）
  - 总决赛唯一最高奖 - 综合特等奖
  - “基于BERT的大模型容量挑战赛”单赛题冠军
- 第四届中国软件开源创新大赛
  - 开源项目创新赛（命题组）一等奖

入选榜单的10个年度开源事件为（按时间排序）

2021.04.26 MindSpore盘古α，华为

2021.05.20 PaddlePaddle 2.1版本，百度

2021.06.22 AliceMind，阿里巴巴达摩院

2021.07.15 NPU ISA，ONIA

唯一入选高校 **2021.07.18 Hetu，北京大学**

2021.08.28 DeePMD-kit v2.0.0，深势科技

2021.10.26 源1.0，浪潮

2021.10.29 PatrickStar，腾讯

2021.11.09 openEuler，华为

2021.12.03 ERNIE-M，百度

https://github.com/PKU-DAIR/Hetu

总结与展望

- 河图已经完成了多种主流通信架构、并行模式、同步协议以及相关优化方案的设计与实现。
- 河图提供了一套统一分布式计算图中间表达，实现了基于标注的半自动并行以及初步的自动并行分布式部署方案。
- 河图目前已经在大规模稀疏嵌入模型、稠密预训练模型、稀疏-稠密MoE模型等场景的计算、通信、存储等方面进行了前沿的创新和尝试。
- 未来将会继续探索面向新兴AI计算负载、更为复杂多样的深度学习系统设计与优化技术。

Hetu 河图

谢谢，请批评指正！

分布式深度学习系统 Hetu

https://github.com/PKU-DAIR/hetu

https://toscode.gitee.com/pku-dair/Hetu

自动化机器学习系统 Mindware https://github.com/PKU-DAIR/mindware

高效通用的黑盒优化系统 OpenBox https://github.com/PKU-DAIR/open-box

可扩展的图神经网络系统 SGL https://github.com/PKU-DAIR/SGL