51CTO WOT

World Of Tech 2024

WOT全球技术创新大会

2024

智启新纪
慧创万物

51CTO WOT

Colossal-AI:
LLM训练和加速的新技术与挑战

潞晨科技CTO
卞正达

51CTO WOT

Colossal-AI

https://github.com/hpcaitech/ColossalAI

微信群

尤洋 创始人 & 董事长

- 伯克利博士
- 新加坡国立大学-校长青年教授
- IEEE杰出新人奖
- 2020全球HPC领域引用量最高的博士

Berkeley UNIVERSITY OF CALIFORNIA NUS National University of Singapore IEEE

James Demmel 首席战略官

- 伯克利杰出教授
- 美国科学院院士、美国工程院院士
- 美国艺术与科学院院士
- IEEE/ACM Fellow

Berkeley UNIVERSITY OF CALIFORNIA NATIONAL ACADEMY OF SCIENCES IEEE

Colossal-AI:
LLM训练和加速的新技术与挑战

Why Colossal-AI?
大模特时代的挑战与机遇

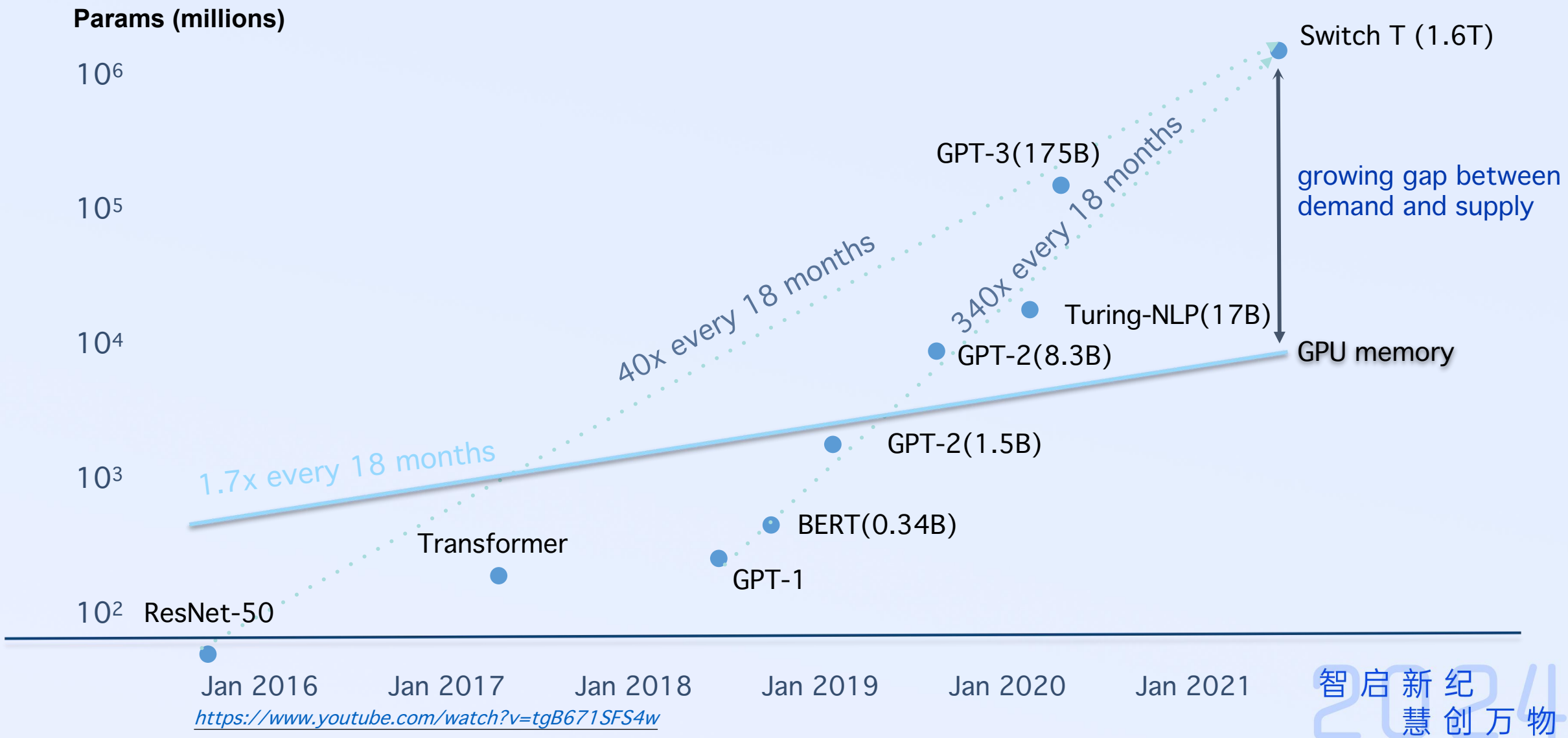
How Colossal-AI Works?
底层系统和基础设施

What Has Colossal-AI Achieved?
Benchmark & 社区影响

Why Colossal-AI?

大模特时代的挑战与机遇

AI大模型发展现状



https://www.youtube.com/watch?v=tgB671SFS4w

AI大模型的挑战

- 大模型的部署门槛过高

Year	Model	#Param	Mem cost	Best GPU
2018	BERT	340M	8 GB	V100 32GB
2023	LLAMA2	70B	1200+ GB	H100 80GB

AI大模型的挑战

- 大模型的部署成本过高

PaLM：300 years by 1 NV A100 GPUs，
$9.2M+

Training

- GPT-2 (2019): “COVID-19 is a high capacity LED-emitter.”
- GPT-J (2021): “COVID-19 is a novel coronavirus.”

Need to re-train on new data repeatedly

Inference Fine-tuning

- A cluster of GPUs is required simply to load & make predictions
- GPT-3: 2400+ GB; NV A100 GPU: 80 GB

Single GPU server is out-of-memory

Deployment

A company needs 70 people building their internal tools for AI: $20M per year (impossible for startups)

Expensive Infrastructure and Systems

Colossal-AI = 高性能 + 高效率 + 低成本

Hardware → Colossal-AI → Framework

- CPU
- GPU
- TPU
- FPGA

- PyTorch
- Keras
- Hugging Face
- Lightning AI

- Maximize computational efficiency
- Minimize system running time
- Minimize communication
- Minimize code refactoring
- Dynamic adaptive scaling
- Reduce memory footprint

Colossal-AI：零代码LLM开发

- 近似PyTorch语义的接口设计
- 几乎零代码侵入
- 全面的分布式模块封装
- 超低学习成本

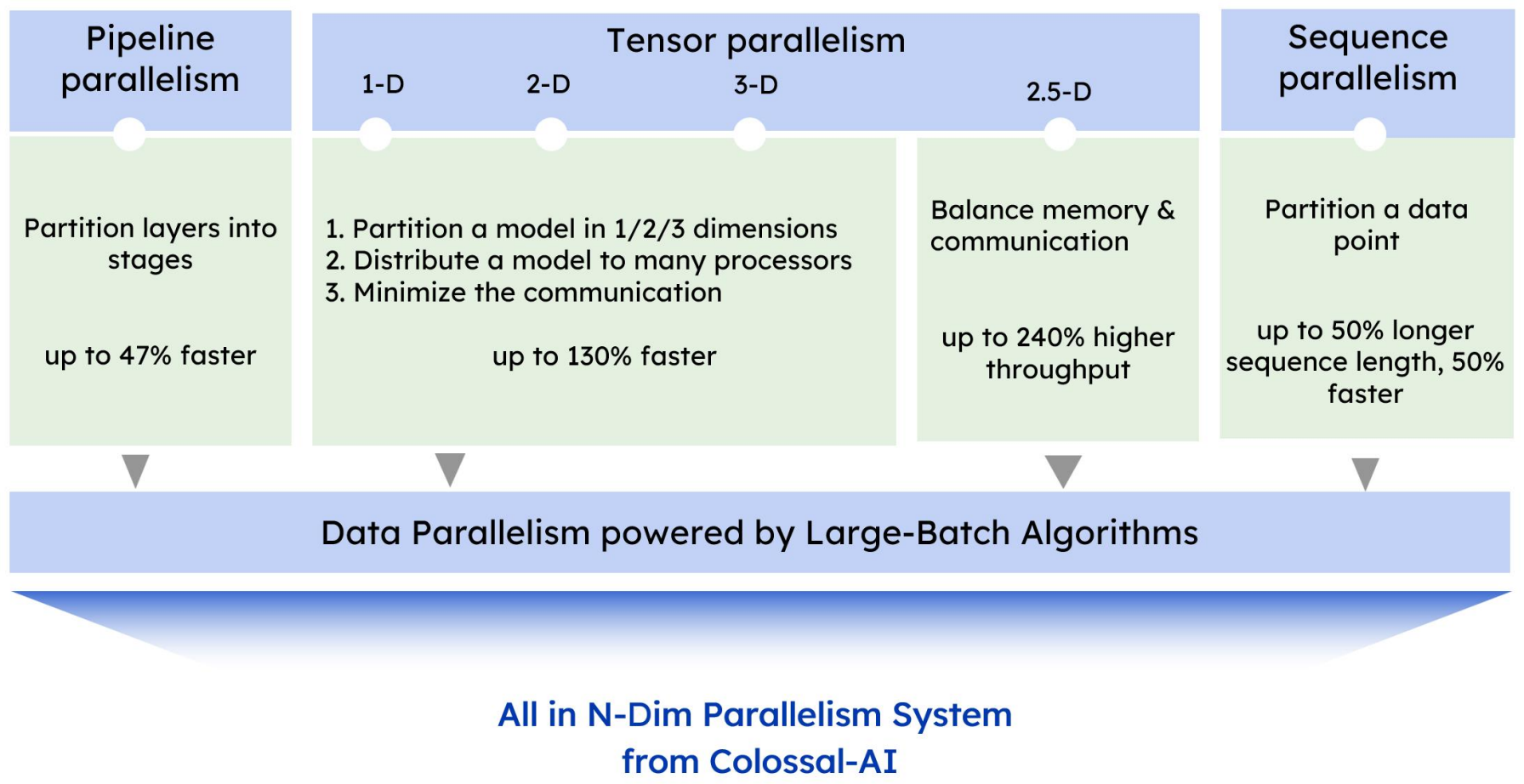
```python
from colossalai.booster import Booster, GeminiPlugin
from colossalai.lazy import LazyInitContext

# setup distributed strategy for booster
plugin = GeminiPlugin(placement_policy='auto', ...)
booster = Booster(plugin=plugin)

# initialize model in a lazy fashion
with LazyInitContext():
    model = LlamaModel()
# initialize optimizer
optimizer = AdamW(model.parameters(), ...)

# process model & optimizer according to
# the distributed strategy
model, optimizer = booster.boost(model, optimizer)

# train the model
...
```

How Colossal-AI Works?

底层系统和基础设施

N 维并行系统

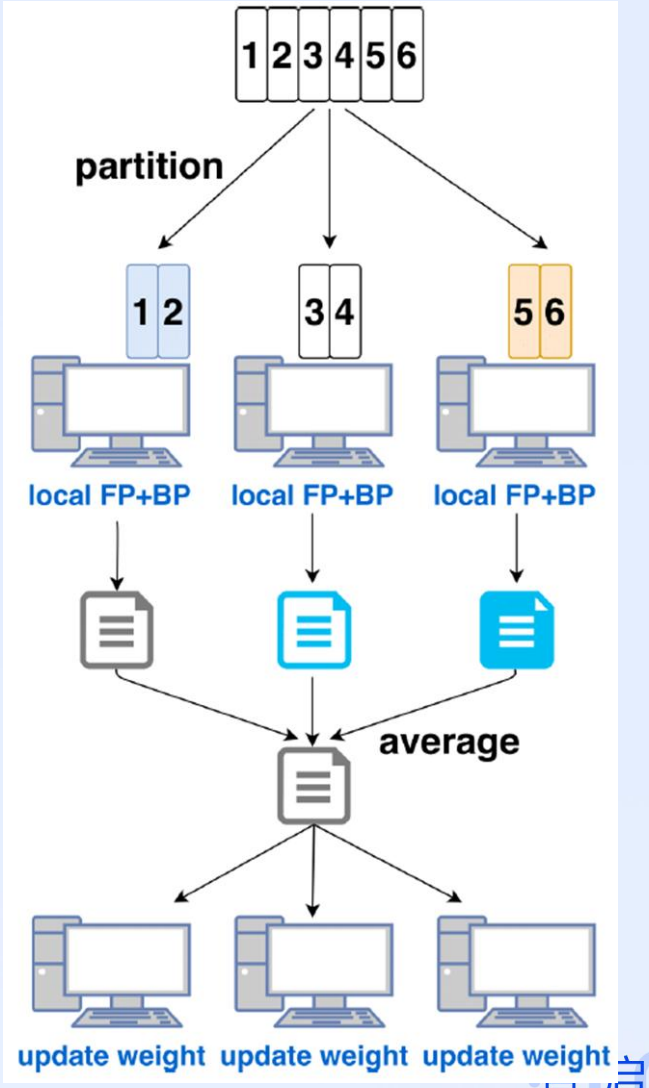
Scaling AI models to any scale



数据并行

Batch Size	Epochs	Iterations
512	100	250,000
1024	100	125,000
2048	100	62,500
4096	100	31,250
8192	100	15,625
...	...	...
1,280,000	100	100

- 单卡：每步处理512个样本，共2500万步
- 2500卡：每步处理128万个样本，仅需1万步



LARS/LAMB: 大规模可扩展优化器

Layer-wise adaptive learning rate:

within layer l and iteration t

$g_t = \frac{1}{B}\sum_{i=1}^{B}\nabla f(x_t^i, w_{t-1}^l)$ /* compute the gradients */

$m_t^l = \beta_1 m_{t-1}^l + (1-\beta_1) g_t^l$ /* compute the first moment */

$v_t^l = \beta_2 v_{t-1}^l + (1-\beta_2) g_t^l \odot g_t^l$ /* compute the second moment */

$\hat{m}_t^l = m_t^l/(1-\beta_1^t)$ /* bias correction for first moment */

$\hat{v}_t^l = v_t^l/(1-\beta_2^t)$ /* bias correction for second moment */

$r_t^l = 1.0$ /* initialize the trust ratio */

$r_1 = \phi(\|w_{t-1}^l\|)$ /* compute the norm of the weights */

$r_2 = \|\frac{\hat{m}_t^l}{\sqrt{\hat{v}_t^l+\epsilon}} + \lambda w_{t-1}^l\|$ /* element-wise weight decay */

if $r_2 > 0$, then $r_t^l = r_1/r_2$ /* compute the trust ratio */

$w_t^l = w_{t-1}^l - \eta \times r_t^l \times (\frac{\hat{m}_t^l}{\sqrt{\hat{v}_t^l+\epsilon}} + \lambda w_{t-1}^l)$ /* update the weights */

Notations

B: batch size; η: learning rate; w_t: weights; λ: weight decay; ϕ: scaling function;
x_t: data samples; f: loss function; g_t: gradients β_1/β_2: coefficient of first/second moment;

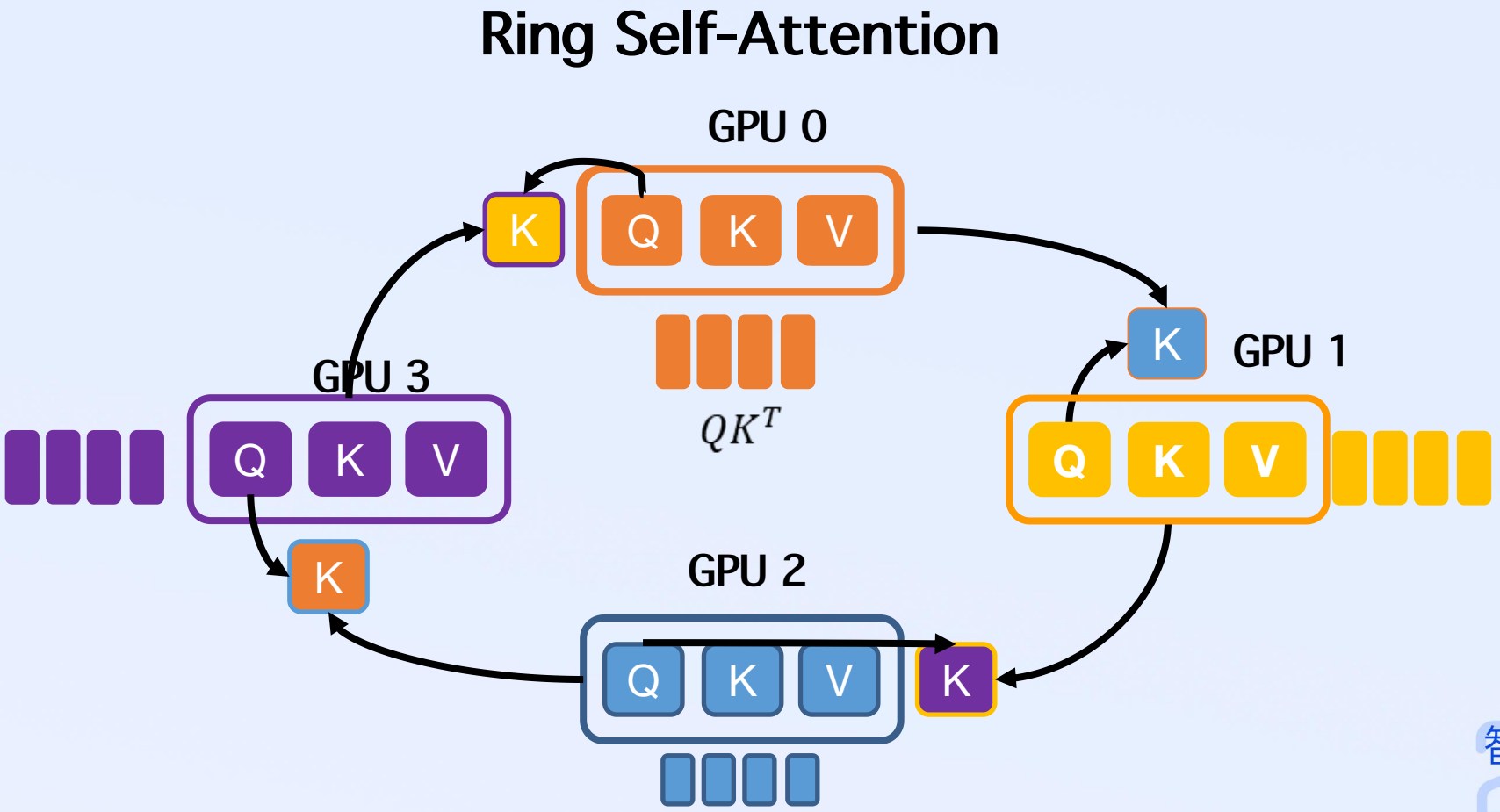
adaptive element-wise updating + layer-wise correction

element-wise weight decay is more accurate (this preserves more information)

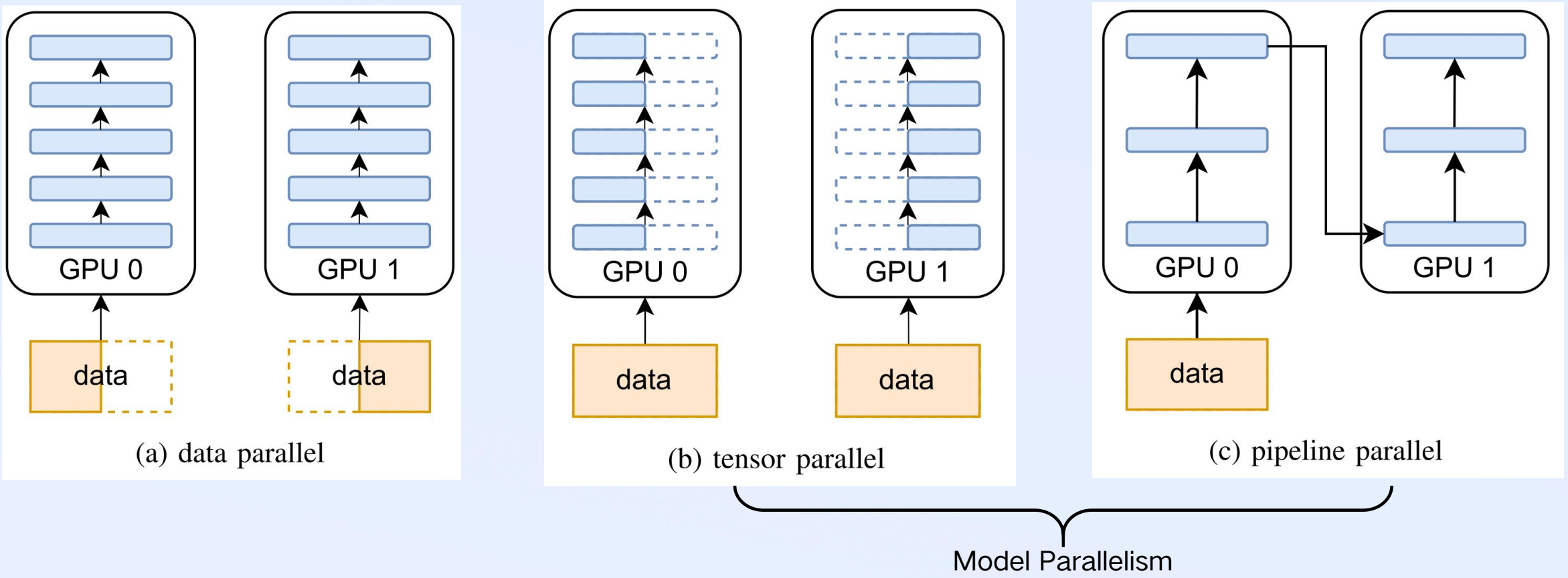
source: NUS CS5260

序列并行：支持无限长序列

- 对于超长context length，处理单个序列需要的显存开销就可能会造成OOM



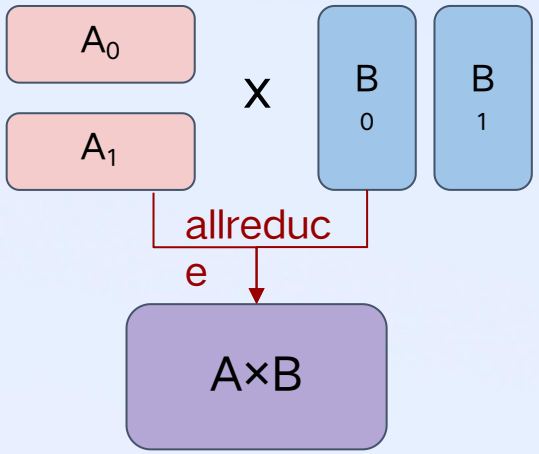
模型并行


GPU 0
GPU 1
data
data
(a) data parallel
GPU 0
GPU 1
data
data
(b) tensor parallel
GPU 0
GPU 1
data
(c) pipeline parallel
Model Parallelism

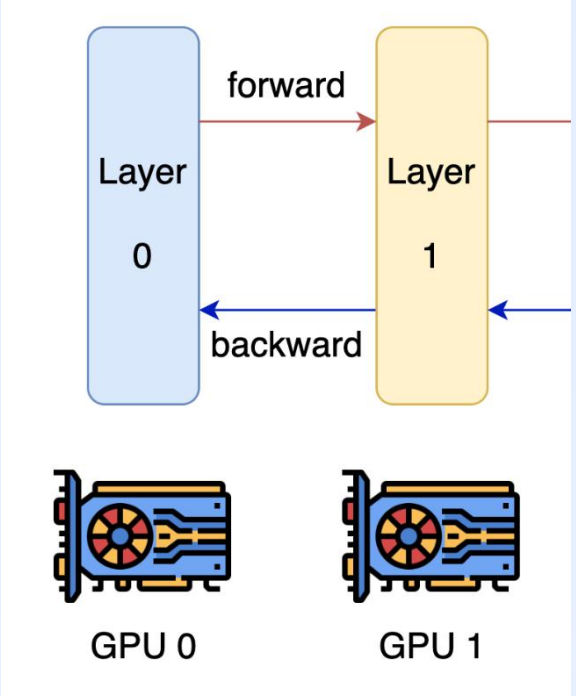
- 常见并行模式

模型并行

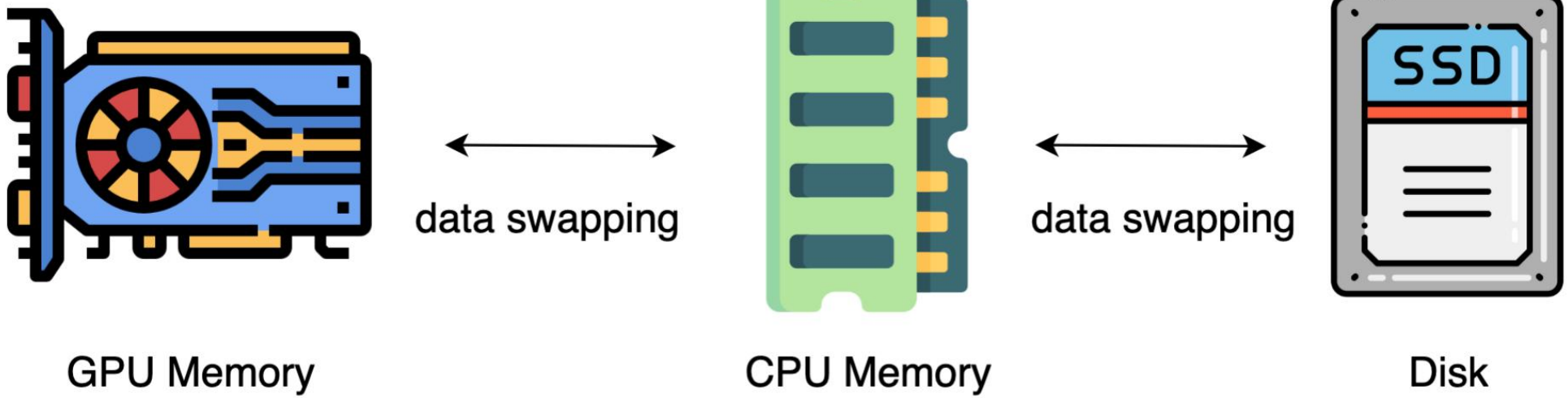
- 张量并行
  - 几乎每个参数都被均摊到多卡上
  - 每张卡上的计算&存储开销基本相同
  - 每一层的计算结果都需要通过通信进行同步



- 流水线并行
  - 模型不同的层被均摊到多卡上
  - 计算&存储开销不尽相同，存在bubble
  - 不同卡之间的计算结果才通过通信进行同步



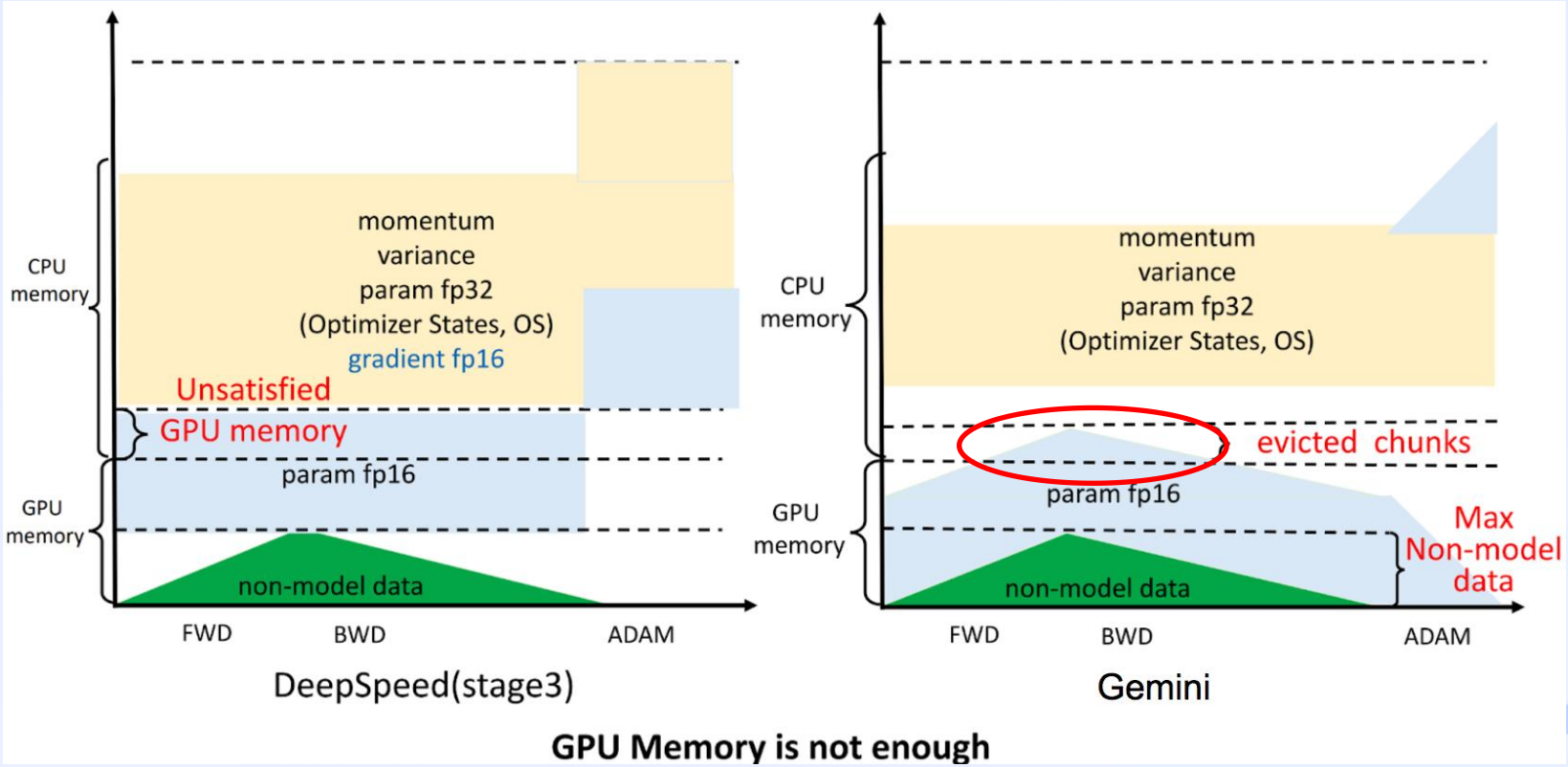
异构存储系统



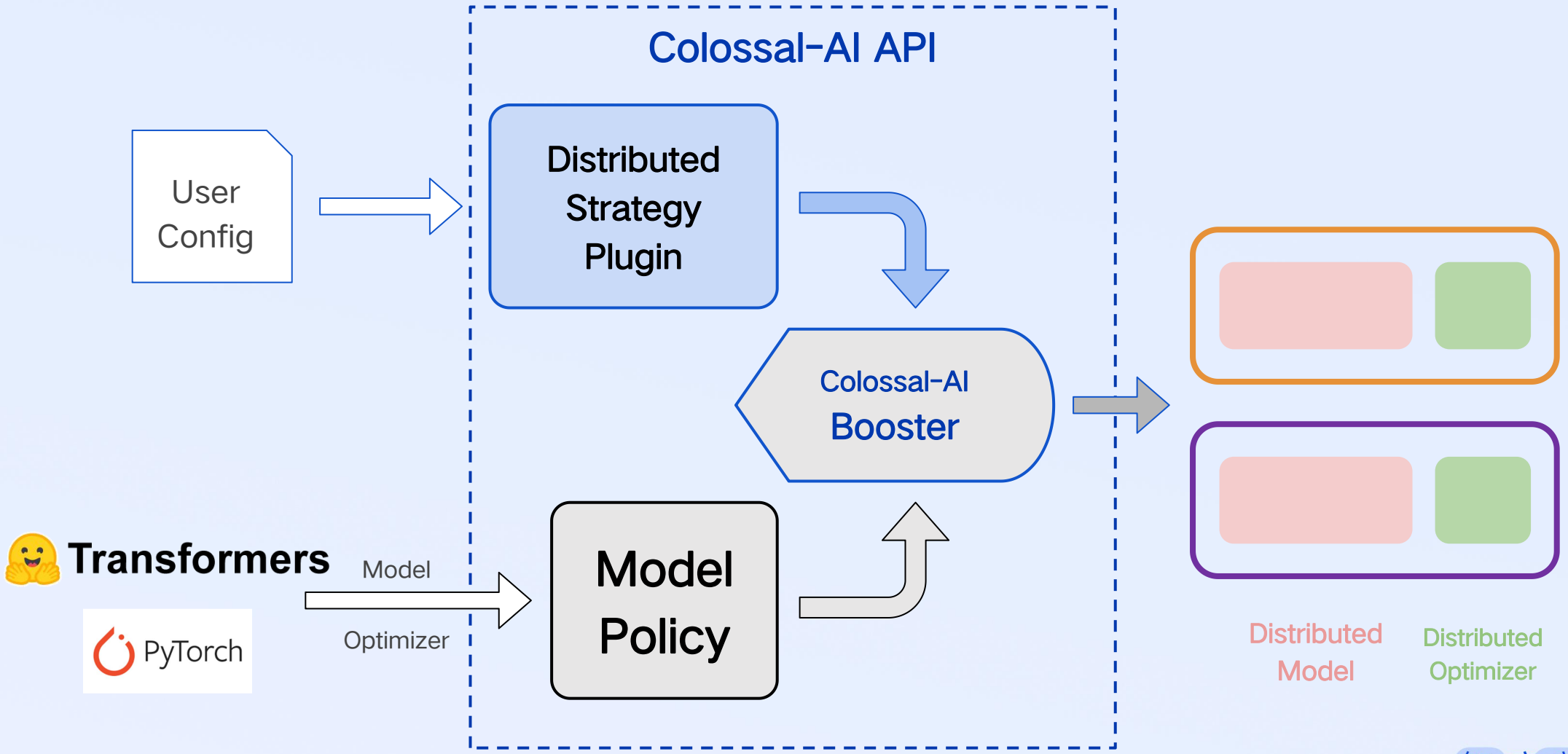
异构存储系统

Gemini offload:

- 自适应的offload粒度来减少不必要的数据移动



Colossal-AI：易用性 + 通用性 + 扩展性



What has Colossal-AI Achieved?

Benchmark & 社区影响

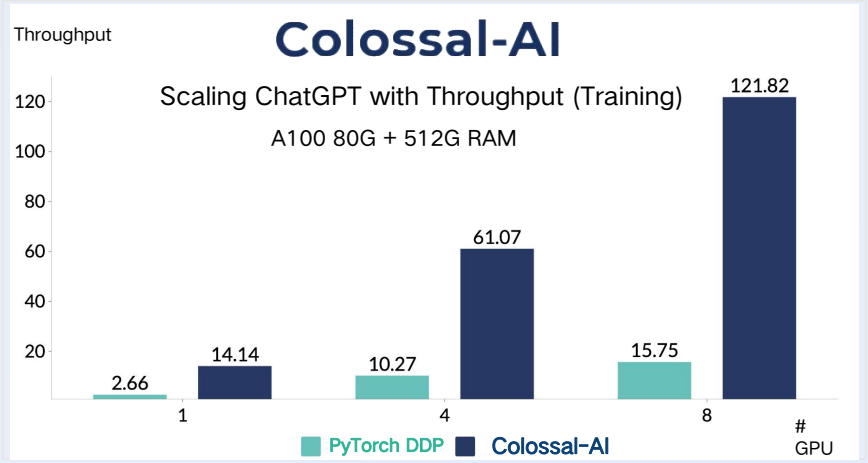
Benchmark结果

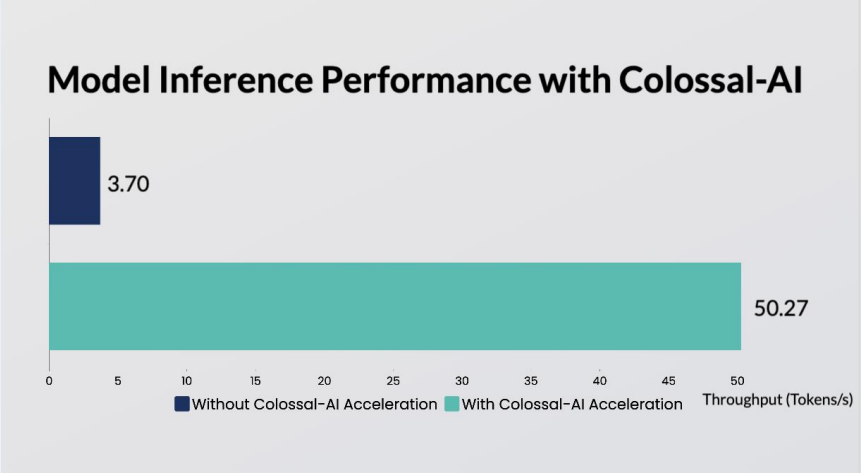
- 第一个在大规模集群上训练LLAMA达到如下SOTA性能的开源产品：
  - 185 HW TFLOPs per GPU (TFLOPs)
  - GPU利用率(59.3%)

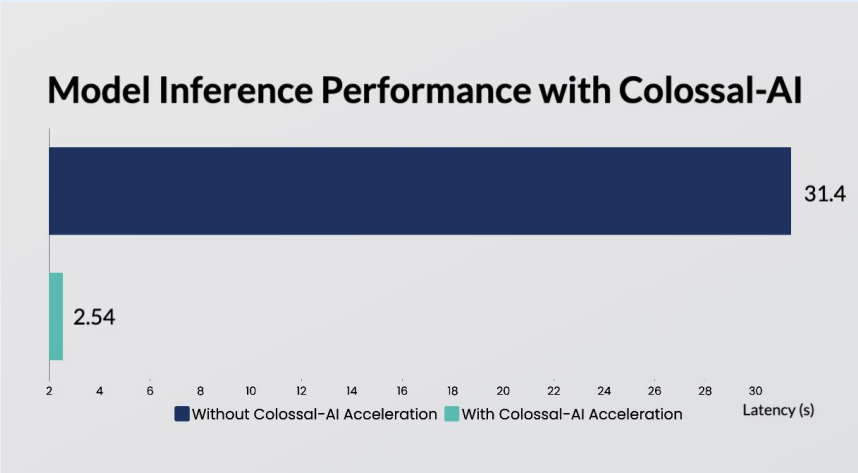
Strategy	Model	#A100	TFLOPS
DeepSpeed	BLOOM-176B	384	156
Pytorch FSDP	GPT-175B	128	159
Sagemaker	Falcon-40B	256	166
JAX + Alpa	OPT-175B	1024	179
Colossal-AI	LLaMA-65B	>700	**185**

Benchmark结果

- ChatGPT: 微调7.73倍加速，推理13.6倍加速







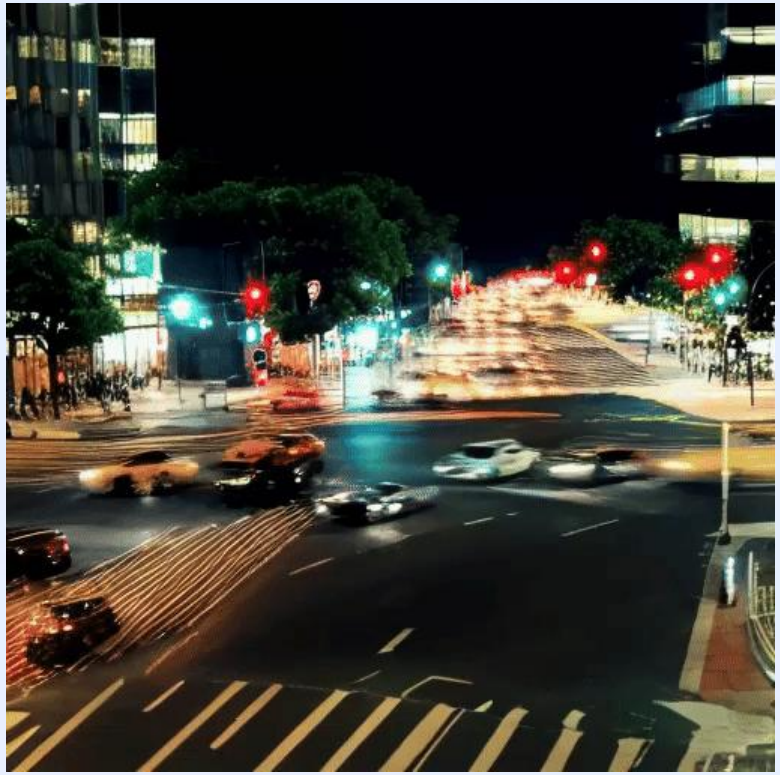
Colossal-AI应用场景

Model	Backbone	Tokens Consumed	MMLU	CMMLU	AGIEval	GAOKAO	CEval
Baichuan2-7B-Base	From scratch	2.6T	46.97	57.67	45.76	52.6	54
Baichuan2-13B-Base	From scratch	2.6T	54.84	62.62	52.08	58.25	58.1
ChatcLM2-6B	From scratch	1.4T	44.74	49.4	46.36	45.49	51.7
InternLM-7B	From scratch	1.6T	46.7	52	44.77	61.64	52.8
Qwen-7B	From scratch	2.2T	54.29	56.03	52.47	56.42	59.6
Llama-2-7B	From scratch	2.0T	44.47	32.97	32.6	25.46	-
Linly-AI/Chinese-LLaMA-2-7B-hf	Llama-2-7B	1.0T	37.43	29.92	32	27.57	-
TigerResearch/tigerbot-7b-base	Llama-2-7B	0.3T	43.73	42.04	37.64	30.61	-
FlagAlpha/Atom-7B	Llama-2-7B	0.1T	49.96	41.1	39.83	33	-
IDEA-CCNL/ziva-LLaMA-13B-v1.1	Llama-13B	0.11T	50.25	40.99	40.04	30.54	-
Colossal-LLaMA-2-7b-base	Llama-2-7B	0.0085T	53.06	49.89	51.48	58.82	50.2
Colossal-LLaMA-2-13b-base	Llama-2-13B	0.025T	56.42	61.8	54.69	69.53	60.3

- 8.5B token数据、15小时、数千元的训练成本
- 与开源社区同7B规模预训练SOTA模型媲美
- 克服英文的灾难性遗忘
(MMLU: 44.47 -> 53.06)
- 中文能力极大提升
(CMMLU: 32.97->49.89)

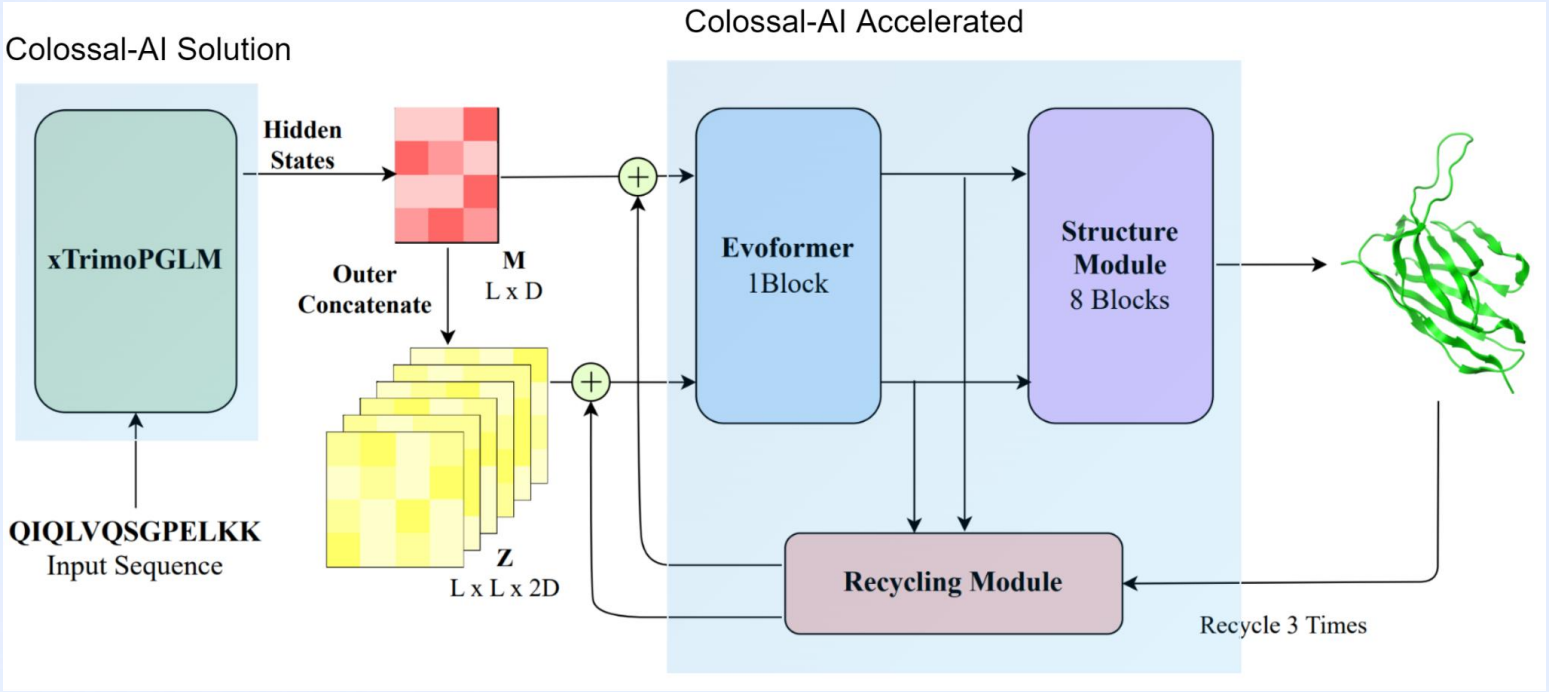
GitHub https://github.com/hpcaitech/ColossalAI

Colossal-AI应用场景



Colossal-AI团队**发布全球首个类Sora架构视频生成模型**「Open-Sora」
开源模型仅使用百卡H100训练，可生成16s左右720P的优质视频

Colossal-AI应用场景



xTrimoMultimer is a cooperation project by BioMap and HPC-AI TECH which provides a **high-performance implementation of AlphaFold and AlphaFold Multimer** with the following characteristics.

1. Fast kernel performance on GPU platform.
2. Supporting Various Parallelism including Dynamic Axial Parallelism(DAP) by FastFold in multi-GPU environment for both AlphaFold monomer and multimer.
3. Support long sequence training(To be supported in the future) and inference in both monomer and multimer.

Colossal-AI: 世界顶级人工智能训练性能基准 MLPerf 的官方推荐

MLPerf

The Global Standard in AI Training Benchmarks

MLCommons GitHub dependents

顶级人工智能生态系统

- InternLM训练所用的InternEvo轻量级训练框架

Training Performance

InternLM deeply integrates Flash-Attention, Apex and other high-performance model operators to improve training efficiency. By building the Hybrid Zero technique, it achieves efficient overlap of computation and communication, significantly reducing cross-node communication traffic during training. InternLM supports expanding the 7B model from 8 GPUs to 1024 GPUs, with an acceleration efficiency of up to 90% at the thousand-GPU scale, a training throughput of over 180 TFLOPS, and an average of over 3600 tokens per GPU per second. The following table shows InternLM's scalability test data at different configurations:

GPU Number	8	16	32	64	128	256	512	1024
TGS	4078	3939	3919	3944	3928	3920	3835	3625
TFLOPS	193	191	188	188	187	185	186	184

Acknowledgements

InternLM codebase is an open-source project contributed by Shanghai AI Laboratory and researchers from different universities and companies. We would like to thank all the contributors for their support in adding new features to the project and the users for providing valuable feedback. We hope that this toolkit and benchmark can provide the community with flexible and efficient code tools for fine-tuning InternLM and developing their own models, thus continuously contributing to the open-source community. Special thanks to the two open-source projects, flash-attention and ColossalAI.

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

商汤 sensetime

顶级人工智能生态系统

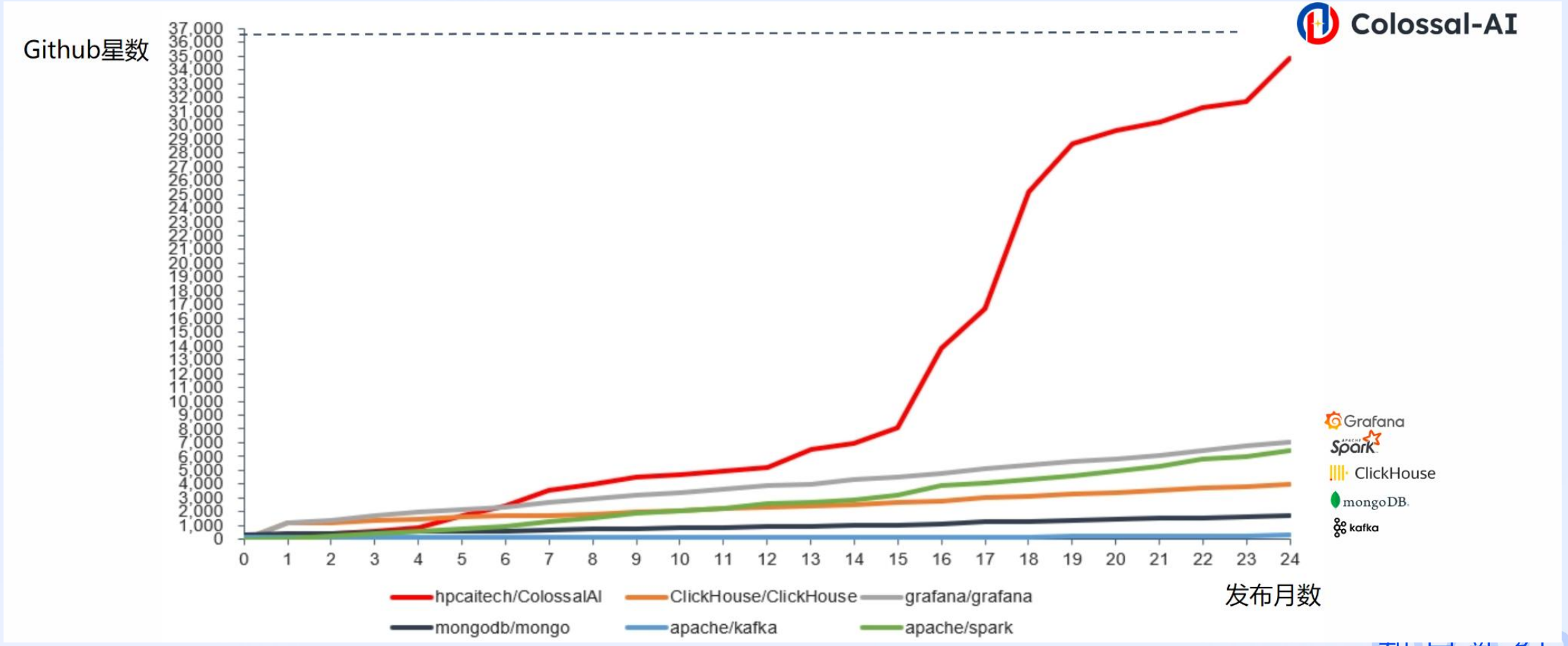
- 字节跳动的MegaScale分布式框架

Failures and stragglers are the norm rather than the exception for LLM training. At such a scale, the consequences of failures and stragglers are devastating. Failures are very expensive, and it is critical to reduce the recovery time, given the large scale. A straggler not only affects its own work, but slows down the entire job involving tens of thousands of GPUs.

In this paper, we present the design, implementation and engineering experience of MegaScale, a production system for training LLMs at scale. MegaScale enables us to scale LLM training to more than 10,000 GPUs. We are able to harness the power of the massive number of GPUs to train LLMs with high training efficiency and stability. In building and operating MegaScale, we apply two systems principles: algorithm-system co-design and in-depth observability.

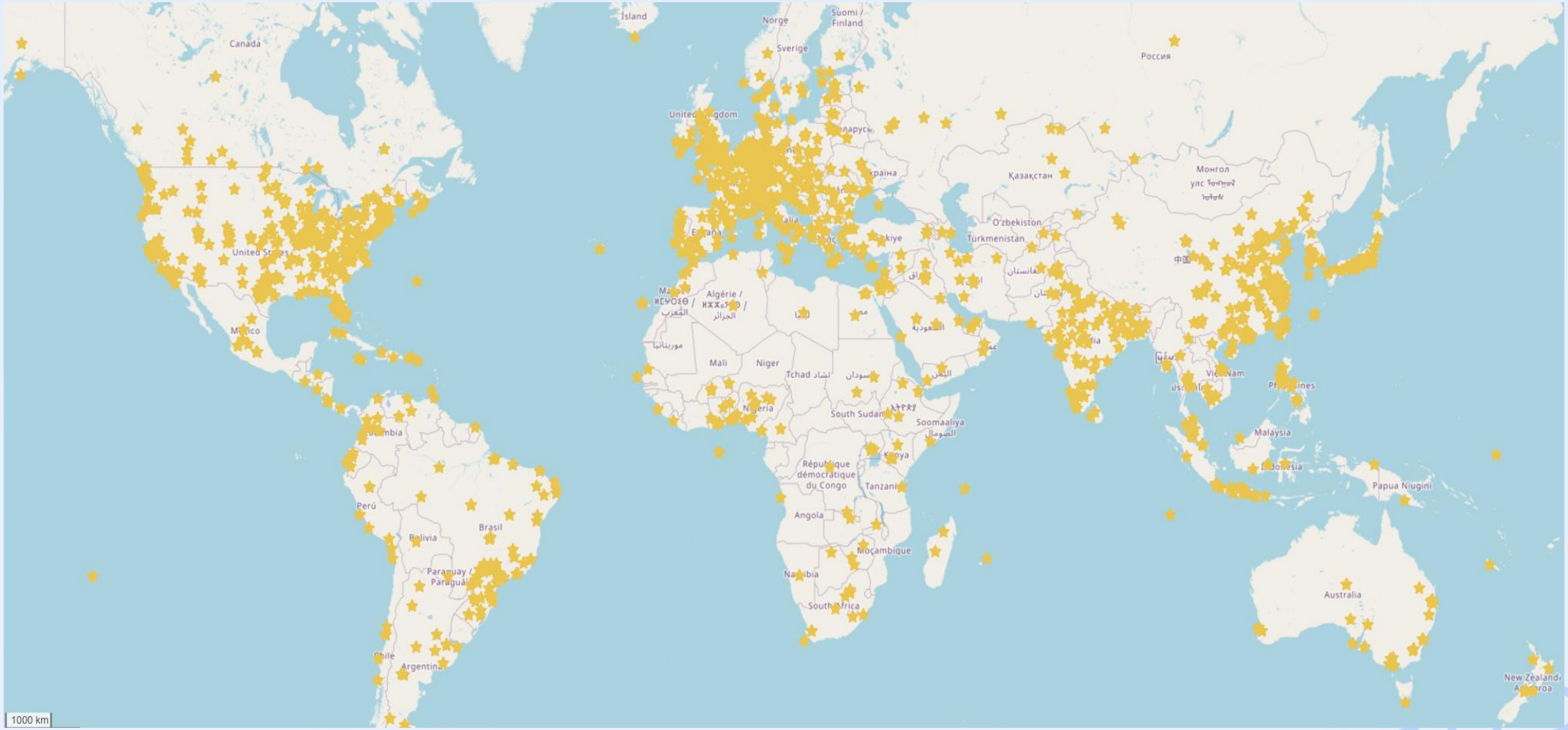
MegaScale is a specialized system tailored for LLM training. Algorithm-system co-design is a key principle to maximize performance for specialized systems, which has been applied widely in computer systems. We apply this principle to MegaScale in the context of LLM training with a full-stack approach that spans all important system components. We make several modifications and incorporate effective optimization techniques to the model architecture, including parallel transformer block [5], sliding window attention [8] and LAMB optimizer [9]. We leverage mixed parallelism strategies that combine data parallelism, pipeline parallelism, tensor parallelism, and sequence parallelism. Importantly, we design custom techniques based on the pattern of each parallelism strategy to maximize the overlapping between communication and computation. We apply prefetching and tree-based loading to optimize the data pipeline. We leverage non-blocking asynchronous operations and eliminate global barriers for large-scale collective communication group initialization. We design a custom network topology, reduce ECMP hash conflicts, customize congestion control, and tune retransmit timeout parameters for high network performance.

快速增长的开源社区



Colossal-AI is public and available at https://github.com/hpcaitech/ColossalAI

Colossal-AI 用户遍布全球



学术认可

NEURAL INFORMATION PROCESSING SYSTEMS

SC22 Dallas, TX hpc accelerates.

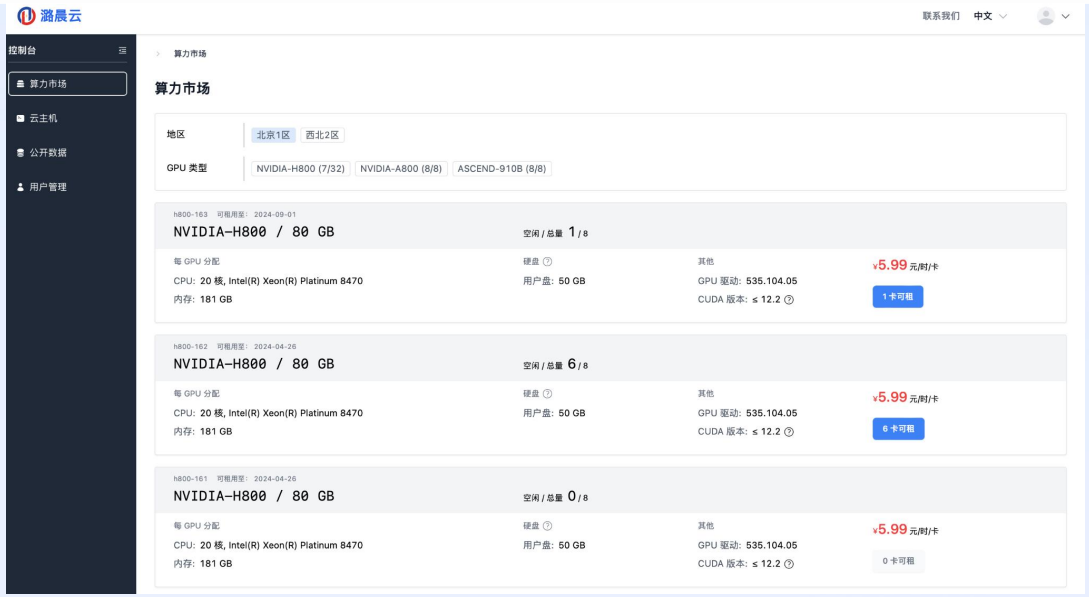
AAAI Association for the Advancement of Artificial Intelligence

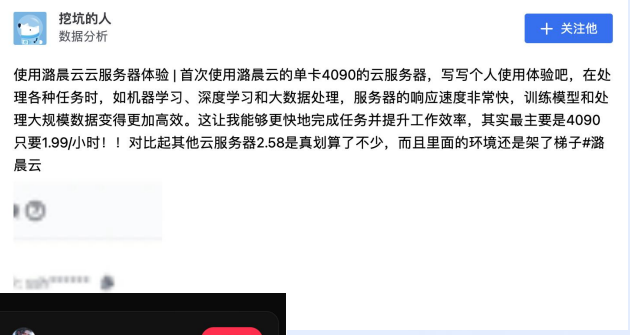
JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

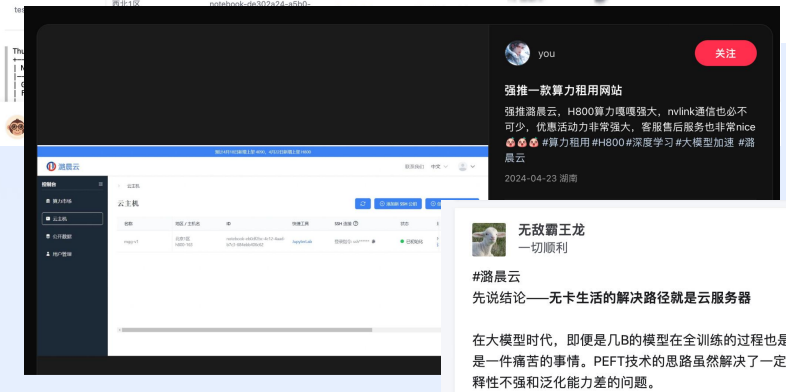
ISC High Performance The HPC Event.

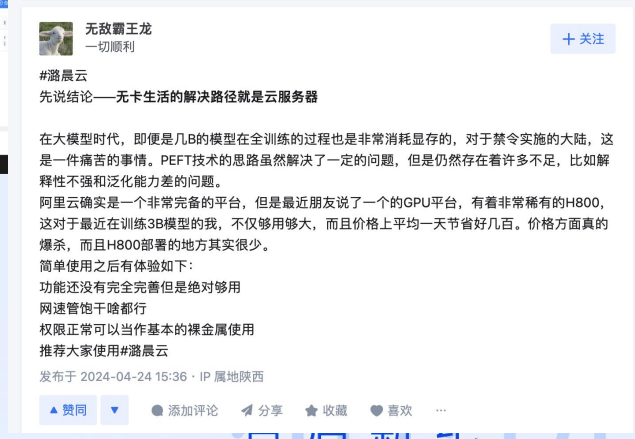
NVIDIA GTC GPU TECHNOLOGY CONFERENCE

潞晨云：稳定、便宜的GPU租赁









51CTO WOT

谢 谢 观 看

THANKS

智启新纪
慧创万物
2024