THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

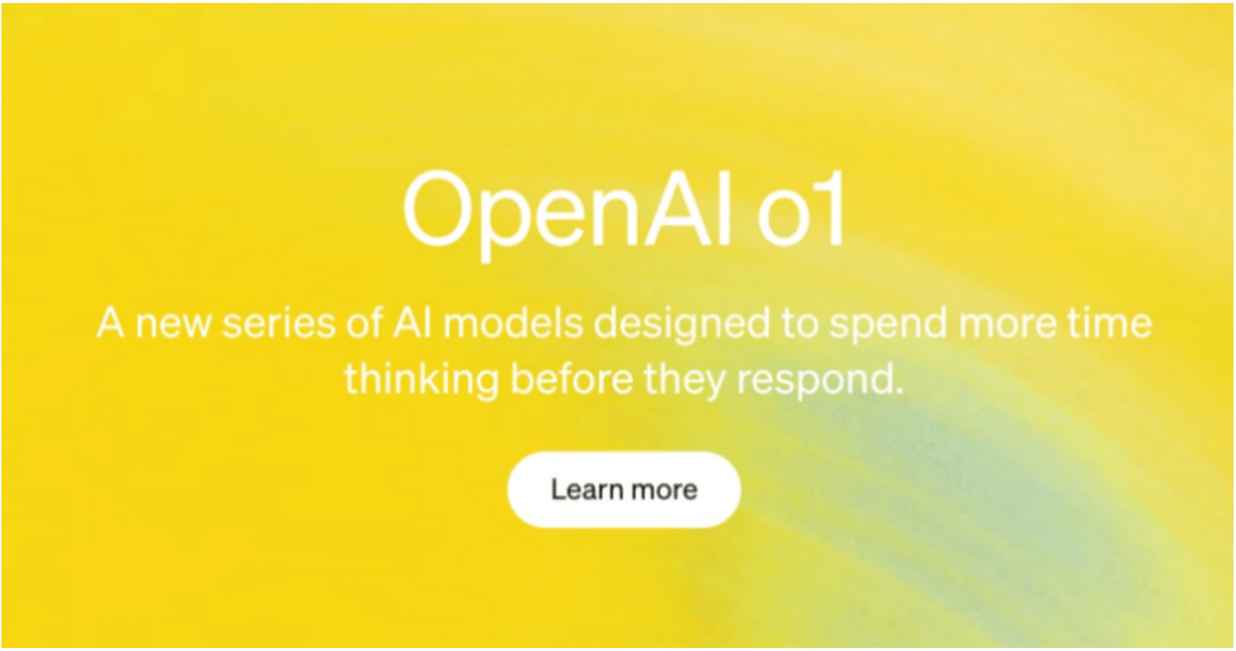
RELAXED SYSTEM LAB

Accommodating LLM Service over Heterogeneous Computational Resources

Binhang Yuan

19.10.24

Amazing Progress of ML/AI



The challenge of Today:

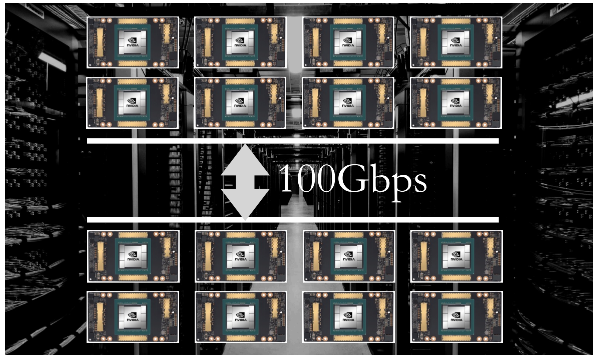
(Million $)

Building ML Applications at SOTA scale is expensive!

Further scaling is facing non-linear bottlenecks.

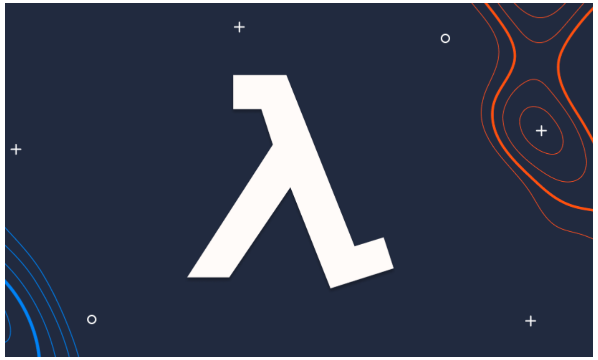
Communication Bottlenecks across Infrastructure

Ccommunication becomes slower, open up more choices (and some can be cheaper);

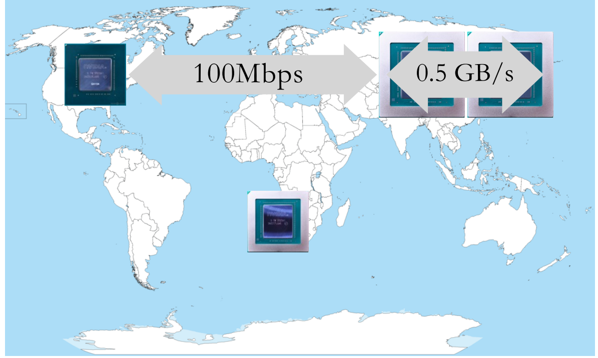


Data Center

(Multi-cloud) Spot Instances



Serverless Environment



Decentralized Network

The more we can optimize communications, the more choices we have when building our infrastructure.

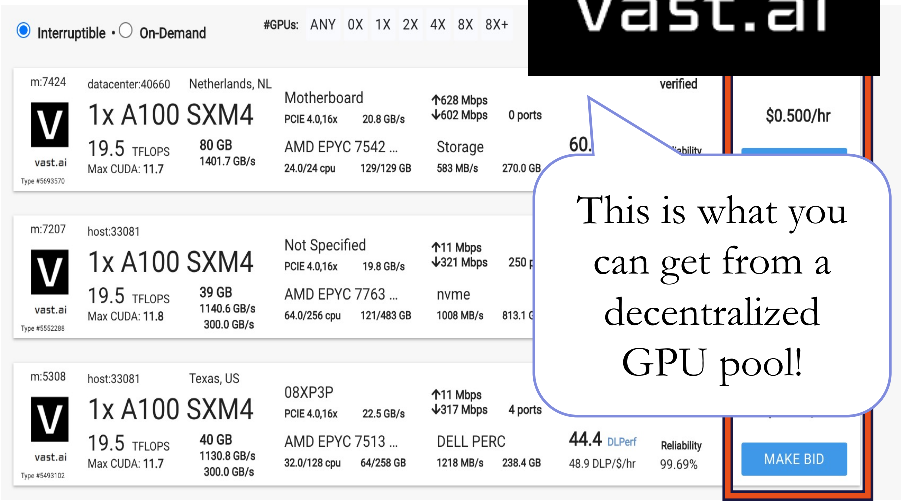
From Cloud to Decentralized Compute Resource

Instance Size	vCPUs	Instance Memory (GiB)	GPU – A100	GPU memory	Network Bandwidth (Gbps)	GPUDir RDM		…rage (GB)	…Bandwidth (Gbps)	…demand Price/hr
p4d.24xlarge	96	1152	8	320 GB HBM2	400 ENA and EFA	Yes	60 NV			
p4de.24xlarge (preview)	96	1152	8	640 GB HBM2e	400 ENA and EFA	Yes	60 NV			

aws

This is $4.09/hour for an A100 GPU.

vast.ai



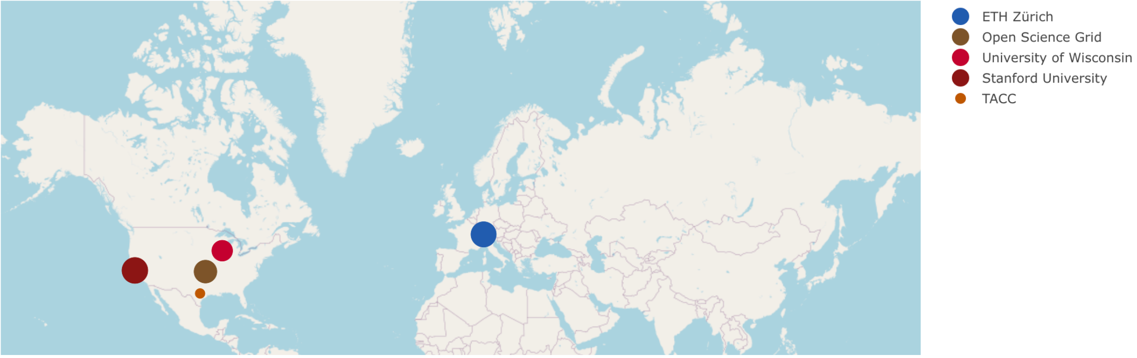
This is what you can get from a decentralized GPU pool!

Available TFlops
71,509 TFlops

Available GPUs
536

Total TFlops
124,428 TFlops

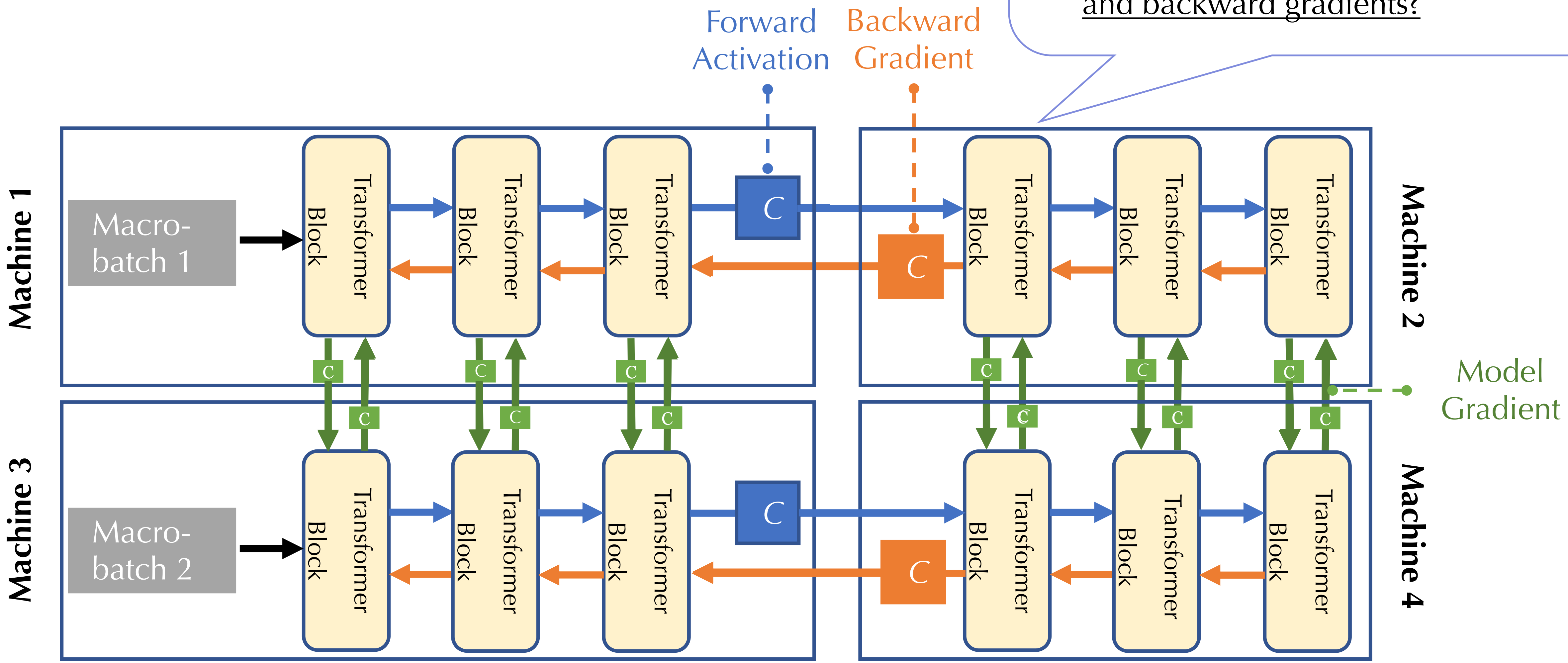
Status Global View



Accommodate LLM training through heterogeneous network.

Pipeline Parallelism

1. How to schedule the communication to accommodate the decentralized connections?
2. How to compress forward activations and backward gradients?



Decentralized Training of Foundation Models

- <u>Decentralized training of FM</u>: the network is 100× slower, but the pre-training throughput is only 1.7~3.5× slower!
- <u>Decentralized fine-tuning of FM</u>: ***AQ-SGD*** communication-efficient pipeline training with activation compression.

Decentralized Training of Foundation Models in Heterogeneous Environments

Binhang Yuan†*, Yongjun He†*, Jared Quincy Davis‡, Tianyi Zhang‡, Tri Dao‡, Beidi Chen‡, Percy Liang‡, Christopher Re‡, Ce Zhang†

†ETH Zürich, Switzerland ‡Stanford University, USA
{binhang.yuan, yongjun.he, ce.zhang}@inf.ethz.ch
{tz58, jaredq, beidic, trid, pliang, chrismre}@stanford.edu

Abstract

Training foundation models, such as GPT-3 and PaLM, can be extremely expensive, often involving tens of thousands of GPUs running continuously for months. These models are typically trained in specialized clusters featuring fast, homogeneous interconnects and using carefully designed software systems that support both data parallelism and model/pipeline parallelism. Such dedicated clusters can be costly and difficult to obtain. *Can we instead leverage the much greater amount of decentralized, heterogeneous, and lower-bandwidth interconnected compute?* Previous works examining the heterogeneous, decentralized setting focus on relatively small models that can be trained in a purely data parallel manner. State-of-the-art schemes for model parallel foundation model training, such as Megatron, only consider the homogeneous data center setting. In this paper, we present the first study of training large foundation models with model parallelism in a decentralized regime over a heterogeneous network. Our key technical contribution is a scheduling algorithm that allocates different computational "tasklets" in the training of foundation models to a group of decentralized GPU devices connected by a slow heterogeneous network. We provide a formal cost model and further propose an efficient evolutionary algorithm to find the optimal allocation strategy. We conduct extensive experiments that represent different scenarios for learning over geo-distributed devices simulated using real-world network measurements. In the most extreme case, across 8 different cities spanning 3 continents, our approach is 4.8× faster than prior state-of-the-art training systems (Megatron).

Code Availability: `https://github.com/DS3Lab/DT-FM`

1 Introduction

Recent years have witnessed the rapid development of deep learning models, particularly foundation models (FMs) [1] such as GPT-3 [2] and PaLM [3]. Along with these rapid advancements, however, comes computational challenges in training these models: the training of these FMs can be very expensive — a single GPT3-175B training run takes 3.6K Petaflops-days [2]— this amounts to $4M on today's AWS on demand instances, even assuming 50% device utilization (V100 GPUs peak at 125 TeraFLOPS)! Even the smaller scale language models, e.g., GPT3-XL (1.3 billion parameters), on which this paper evaluates, require 64 Tesla V100 GPUs to run for one week, costing $32K on AWS. As a result, speeding up training and decreasing the cost of FMs have been active research areas. Due to their vast number of model parameters, state-of-the-art systems (e.g., Megatron[4], Deepspeed[5], Fairscale[6]) leverage multiple forms of parallelism [4, 7, 8, 9, 10, 11]. However, their design is only tailored to *fast*, *homogeneous* data center networks.

* Equal contribution.

1

[NeurIPS 2022-(a)]

Fine-tuning Language Models over Slow Networks using Activation Compression with Guarantees

Jue Wang†*, Binhang Yuan†*, Luka Rimanic†*, Yongjun He†, Tri Dao‡, Beidi Chen‡, Christopher Re‡, Ce Zhang†

†ETH Zürich, Switzerland ‡Stanford University, USA
{jue.wang, binhang.yuan, luka.rimanic, yongjun.he, ce.zhang}@inf.ethz.ch
{beidic, trid, chrismre}@stanford.edu

Abstract

Communication compression is a crucial technique for modern distributed learning systems to alleviate their communication bottlenecks over slower networks. Despite recent intensive studies of gradient compression for data parallel-style training, compressing the *activations* for models trained with pipeline parallelism is still an open problem. In this paper, we propose `AC-SGD`, a novel activation compression algorithm for communication-efficient pipeline parallelism training over slow networks. Different from previous efforts in activation compression, instead of compressing activation values directly, `AC-SGD` compresses the *changes of the activations*. This allows us to show, to the best of our knowledge for the first time, that one can still achieve $O(1/\sqrt{T})$ convergence rate for non-convex objectives under activation compression, without making assumptions on gradient unbiasedness that do not hold for deep learning models with non-linear activation functions. We then show that `AC-SGD` can be optimized and implemented efficiently, without additional end-to-end runtime overhead. We evaluated `AC-SGD` to fine-tune language models with up to 1.5 billion parameters, compressing activations to 2-4 bits. `AC-SGD` provides up to $4.3\times$ end-to-end speed-up in slower networks, without sacrificing model quality. Moreover, we also show that `AC-SGD` can be combined with state-of-the-art gradient compression algorithms to enable "end-to-end communication compression": *All communications between machines, including model gradients, forward activations, and backward gradients are compressed into lower precision*. This provides up to $4.9\times$ end-to-end speed-up, without sacrificing model quality.

Code Availability: `https://github.com/DS3Lab/AC-SGD`

1 Introduction

Recent efforts in improving communication efficiency for distributed learning have significantly decreased the dependency of training deep learning models on fast data center networks — the *gradient* can be compressed to lower precision or sparsified [1, 2, 3, 4], which speeds up training over low bandwidth networks, whereas the *communication topology* can be decentralized [5, 6, 7, 8, 9, 10], which speeds up training over high latency networks. Indeed, today's state-of-the-art training systems, such as Pytorch [11, 12], Horovod [13], Bagua [14], and BytePS [15], already support many of these communication-efficient training paradigms.

However, with the rise of large foundation models [16] (e.g., BERT [17], GPT-3 [18], and CLIP[19]), improving communication efficiency via compression becomes more challenging. Current training systems for foundation models such as Megatron [20], Deepspeed [21], and Fairscale [22], allocate different layers of the model onto multiple devices and need to communicate — *in addition to* the gradients on the models — the

* Equal contribution.

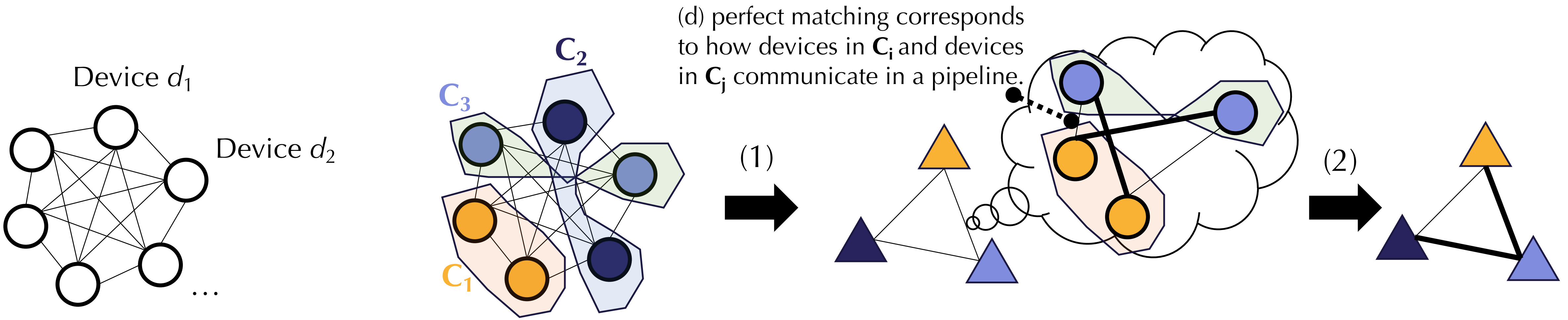
1

[NeurIPS 2022-(b)]

Accommodate Communication in a Decentralized network

A bi-level scheduling algorithm based on an extended balanced graph partition to estimate the communication cost:

- Data parallel communication cost: nodes handling the same stage need to exchange gradients;
- Pipeline parallel communication cost: nodes handling nearby stages for the same micro-batch need to communicate activation in the forward propagation and gradients of the activation in the backward propagation.



(a) Communication Topology Graph **G** over N devices

(b) Each partition $\mathbf{C_i}$ deals with one stage, running data parallel within each partition

(c) Coarsened graph $\widehat{\boldsymbol{G}}$ decoding the cost of pipeline parallel

(e) Open-loop-traveling-salesman provides a pipeline structure

AQ-SGD

$$\min_{x \in \mathbb{R}^d} f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} F(b(a(\xi, x^{(a)}), x^{(b)}))$$

Direct quantization only works to some degree.

Converge better

Model change smaller

Activation change smaller

Activation Diff Smaller

Quantization Error of Diff smaller

...

$a(-, x^{(a)})$ → $F \cdot b(-, x^{(b)})$

diff across epochs should diminish

Quantize Diff

...

$a(-, x_+^{(a)})$ → $F \cdot b(-, x_+^{(b)})$

...

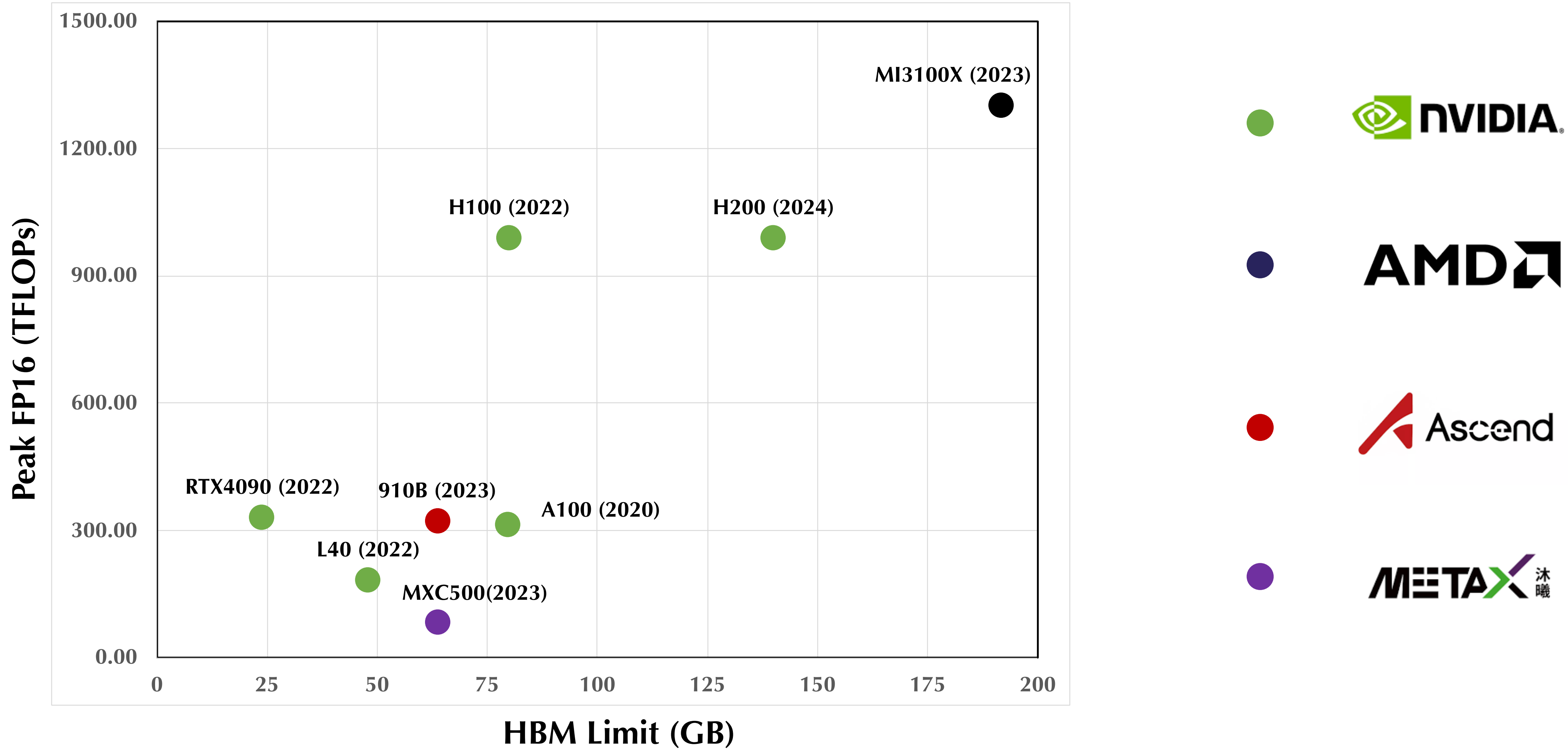
- **(A1: Lipschitz assumptions)** We assume that ∇f, $\nabla(f \circ b)$ and a are L_f, $L_{f \circ b}$-, and ℓ_a-Lipschitz, respectively, recalling that a function g is L_g-Lipschitz if
$$\|g(x) - g(y)\| \le L_g \|x - y\|, \quad \forall x, \forall y.$$
Furthermore, we assume that a and $f \circ b$ have gradients bounded by C_a and $C_{f \circ b}$, respectively, i.e. $\|\nabla a(x)\| \le C_a$, and $\|\nabla(f \circ b)(x)\| \le C_{f \circ b}$.
- **(A2: SGD assumptions)** We assume that the stochastic gradient g_ξ is unbiased, i.e. $\mathbb{E}_\xi[g_\xi(x)] = \nabla f(x)$, for all x, and with bounded variance, i.e. $\mathbb{E}_\xi \|g_\xi(x) - \nabla f(x)\|^2 \le \sigma^2$, for all x.

Theorem 3.1. *Suppose that Assumptions A1, A2 hold, and consider an unbiased quantization function $Q(x)$ which satisfies that there exists $c_Q < \sqrt{1/2}$ such that $\mathbb{E}\|x - Q(x)\| \le c_Q \|x\|$, for all x.*[1] *Let $\gamma = \frac{1}{3(C + 3L_f)\sqrt{T}}$ be the learning rate, where*
$$C = \frac{4 c_Q \ell_a (1 + C_a) L_{f \circ b} N}{\sqrt{1 - 2c_Q^2}}.$$
Then after performing T updates one has
$$\frac{1}{T} \sum_{t \in [T]} \mathbb{E}\|\nabla f(x_t)\|^2 \lesssim \frac{(C + L_f)(f(x_1) - f^*)}{\sqrt{T}} + \frac{\sigma^2 + (c_Q C_a C_{f \circ b})^2}{\sqrt{T}}. \tag{3.1}$$

[NeurIPS 2022-(b)]

Accommodate LLM training through heterogeneous hardware.

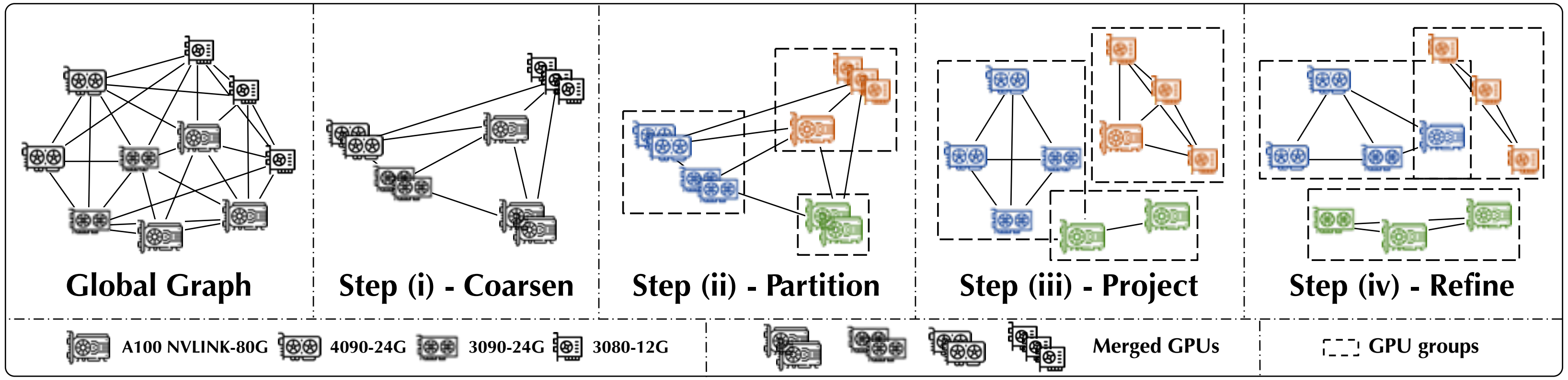
AI Chips Heterogeneous Computation Capacity

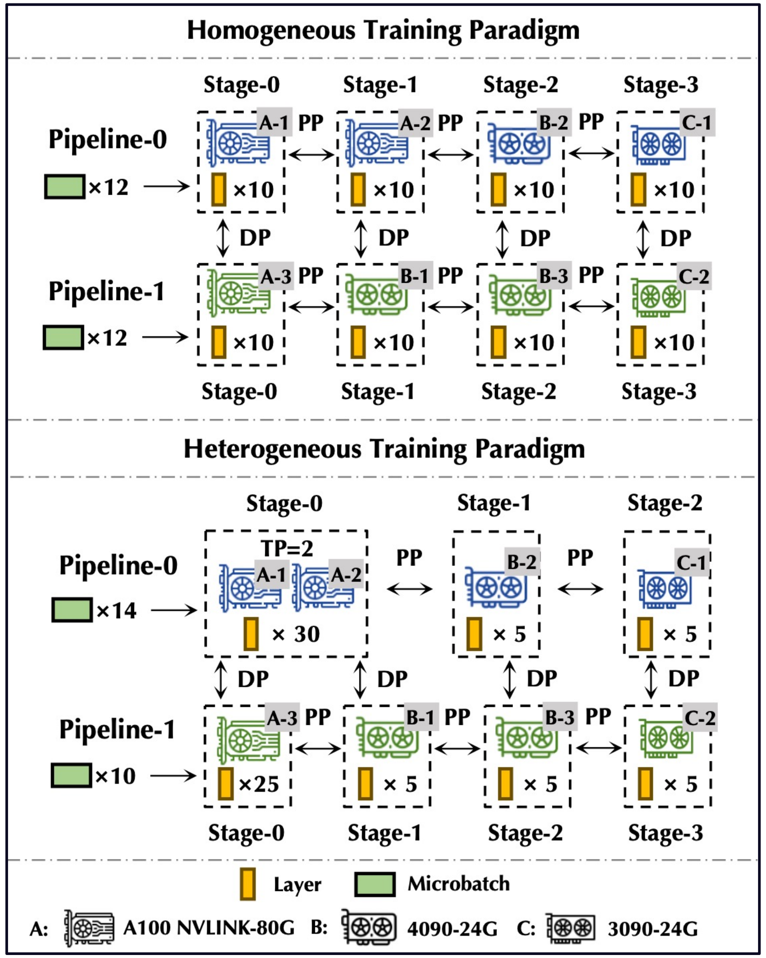


FlashFlex

Accommodating Large Language Model Training over Heterogeneous Environment

- A heterogeneous LLM training system that supports:
  - Data parallelism;
  - Pipeline parallelism;
  - Tensor model parallelism;
- A scheduling algorithm:
  - Two-phase graph partition algorithm





FlashFlex

Accommodating Large Language Model Training over Heterogeneous Environment

- A heterogeneous LLM training system that supports:
  - Data parallelism;
  - Pipeline parallelism;
  - Tensor model parallelism;
- A scheduling algorithm:
  - Two-phase graph partition algorithm

https://github.com/Relaxed-System-Lab/FlashFlex

FlashFlex: Accommodating Large Language Model Training over Heterogeneous Environment

Ran Yan*†, Youhe Jiang*†, Wangcheng Tao†, Xiaonan Nie‡, Bin Cui‡, Binhang Yuan†

†The Hong Kong University of Science and Technology ‡Peking University

{ryanaf,wangcheng.tao}@connect.ust.hk,{xiaonan.nie,bin.cui}@pku.edu.cn,{cseyouhej,biyuan}@ust.hk

arXiv:2409.01143v1 [cs.DC] 2 Sep 2024

ABSTRACT

Training large language model (LLM) is a computationally intensive task, which is typically conducted in data centers with homogeneous high-performance GPUs. This paper explores an alternative approach by deploying the training computation across heterogeneous GPUs to enable better flexibility and efficiency for heterogeneous resource utilization. To achieve this goal, we propose a novel system, FlashFlex, that can flexibly support an asymmetric partition of the parallel training computations across the scope of data-, pipeline-, and tensor model parallelism. We further formalize the allocation of asymmetric partitioned training computations over a set of heterogeneous GPUs as a constrained optimization problem and propose an efficient solution based on a hierarchical graph partitioning algorithm. Our approach can adaptively allocate asymmetric training computations across GPUs, fully leveraging the available computational power. We conduct extensive empirical studies to evaluate the performance of FlashFlex, where we find that when training LLMs at different scales (from 7B to 30B), FlashFlex can achieve comparable training MFU when running over a set of heterogeneous GPUs compared with the state of the art training systems running over a set of homogeneous high-performance GPUs with the same amount of total peak FLOPS. The achieved smallest gaps in MFU are 11.61% and 0.30%, depending on whether the homogeneous setting is equipped with and without RDMA. Our implementation is available at https://github.com/Relaxed-System-Lab/FlashFlex.

1 INTRODUCTION

Over the past few years, large language models (LLM) have demonstrated impressive performance and sparked a new wave of exciting AI applications [4]. However, training these LLMs, such as GPT [35], Claude [3], Gemini [40], Llama [8, 45], Mixtral [16], Yi [54], Falcon [12] etc., can be extremely computation-intensive, often involving thousands of GPUs running continuously for months. The high cost of deploying such training tasks in a cluster with homogeneous GPUs has become an obvious obstacle limiting the evolution of LLMs. In this paper, we explore an alternative approach by *distributing the parallel training computations across heterogeneous GPUs*, to enable greater flexibility in heterogeneous resource utilization and further democratize the LLM training service.

Distributing parallel training computations across heterogeneous GPUs is a natural option to democratize LLM training. In the current exciting era of generative AI, chip vendors typically release new generations of AI chips every 24 months. For instance, Nvidia introduced the Turing architecture in 2018 [31], Ampere in 2020 [32], Hopper in 2022 [33], and Blackwell is scheduled for Q4, 2024 [34]. On the other hand, one particular version of an AI chip often remains in use by cloud service platforms, technology companies, or research institutions for a much longer period. For example, K80 GPUs with Tesla architecture, released in 2006 [30], are still available on AWS as p2 instances [2]. This observation highlights the important opportunity to explore effective ways to maximize the efficiency of such widely available yet heterogeneous hardware to facilitate more cost-effective and accessible LLM training services.

On the other hand, deploying the large-scale training computation for LLM over a set of heterogeneous GPUs with different technique specs would be a challenging task regarding training learning system design and implementation. To effectively distribute the training computation over thousands of GPUs, the state-of-the-art training systems, like Megatron [29] and DeepSpeed [38] usually supports: (i) tensor model parallelism [26, 29]; (ii) pipeline parallelism [11, 27, 28, 52]; and (iii) data parallelism (with potentially sharded implementations of parameters, gradients, and optimizer states across multiple devices, also known as fully sharded data parallelism) [17, 38, 39, 41]. However, these systems typically only support homogeneous configurations, which require the entire training cluster to operate under a fully symmetric setup – This means that all tensor model parallel groups must have the same degree of parallelism, and the same applies to pipeline parallel groups as well as data or optimizer parallel groups. Such implementation assumes all the GPUs take the *same* amount of computation load, which significantly limits the system efficiency when deploying the training computation over GPUs with *different* computation capability (measured by the peak FLOPS), *different* device memory (i.e., HBM) capacity, and *different* network bandwidth for each pair of GPUs (inter-node and intra-node).

Concretely, there are two fundamental challenges stemming from the *heterogeneity*:

- **Different GPU computation capability.** In heterogeneous environments, GPUs can vary significantly in terms of computation capability (i.e., FLOPS) and memory capacity. This disparity poses a challenge in distributing the computation across all available resources. If not properly managed, the most capable GPUs can be underutilized, while less powerful GPUs can become bottlenecks, leading to inefficiencies and increased training time. Partitioning the computation to match the capabilities of each GPU is essential to fully utilize the available hardware.
- **Different GPU-GPU network bandwidth.** The heterogeneous nature of connections between GPUs, ranging from high-speed

Equal contributions are indicated by *.
Correspond to Binhang Yuan (biyuan@ust.hk).

[Preprint: arxiv.2409.01143]

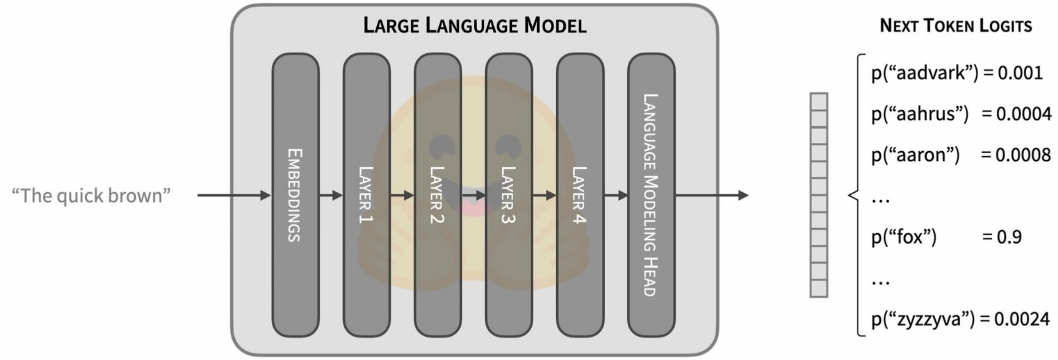
LLM service is NOT all about training.

"90% of the machine learning demand in the cloud is for inference."

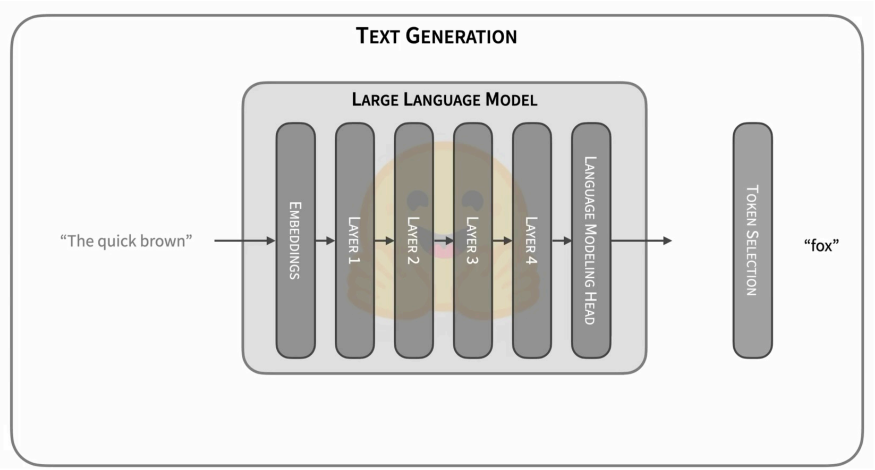
-- AWS Report

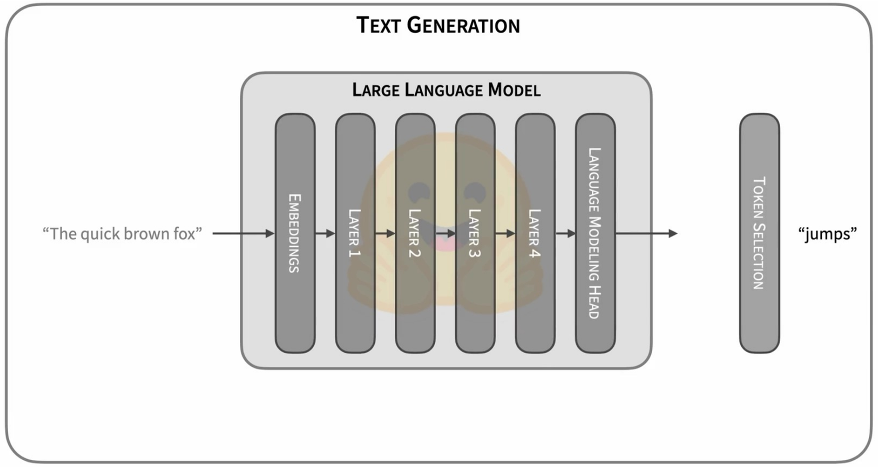
Autoregressive Generation

Prefill phase: the model takes a prompt sequence as input and engages in the generation of a key-value cache (KV cache) for each Transformer layer.



Decode phase: for each decode step, the model updates the KV cache and reuses the KV to compute the output.





The quick brown => fox

Decode step 1

The quick brown fox => jumps

Decode step 2

HexGen: Accommodating LLM Inference over Heterogeneity

- HexGen: schedule the generative inference under the colocating paradigm;
- HexGen-2: schedule the generative inference under the disaggregated paradigm;

HEXGEN: Generative Inference of Large Language Model over Heterogeneous Environment

Youhe Jiang [*1] Ran Yan [*1] Xiaozhe Yao [*2] Yang Zhou [3] Beidi Chen [3] Binhang Yuan [1]

Abstract

Serving generative inference of the large language model is a crucial component of contemporary AI applications. This paper focuses on deploying such services in a heterogeneous and cross-datacenter setting to mitigate the substantial inference costs typically associated with a single centralized datacenter. Towards this end, we propose HEXGEN, a *flexible distributed inference engine* that uniquely supports the asymmetric partition of generative inference computations over both tensor model parallelism and pipeline parallelism and allows for effective deployment across diverse GPUs interconnected by a fully heterogeneous network. We further propose a *sophisticated scheduling algorithm* grounded in constrained optimization that can adaptively assign asymmetric inference computation across the GPUs to fulfill inference requests while maintaining acceptable latency levels. We conduct an extensive evaluation to verify the efficiency of HEXGEN by serving the state-of-the-art LLAMA-2 (70B) model. The results suggest that HEXGEN can choose to achieve up to 2.3× lower latency deadlines or tolerate up to 4× more request rates compared with the homogeneous baseline given the same budget. Our implementation is available at `https://github.com/Relaxed-System-Lab/HexGen`.

1. Introduction

Large language models are distinguished by the vast scale of parameters being trained over a substantial pre-train corpus. Such extensive training enables them to be remarkably adaptable across a broad spectrum of downstream tasks (Bommasani et al., 2021). In fact, large language models such as GPT-4 (Bubeck et al., 2023), Llama2-70B (Touvron et al., 2023), and Falcon-180B (Institute, 2023) have essentially revolutionized the way AI systems are developed and deployed, which have nourished a large number of advanced applications. In such an ecosystem, serving the generative inference requests for large language models presents a critical challenge — given the unprecedented model scale, unlike classic machine learning models, parallel inference strategies have to be leveraged to accommodate the high computational and memory demands while ensuring low-latency generative inference outcomes.

The state-of-the-art inference service of the large language model is usually hosted in a single centralized data center with homogeneous high-performance GPUs, which can be very expensive in terms of the cloud service fee. The high cost of such deployment potentially limits the democratization of this great technique. Alternatively, the deployment of the large language model inference over a heterogeneous cross-datacenter environment can be a promising direction to reduce the inference cost, which has not been fully explored. The heterogeneous environment for foundation model inference service can encompass a wide range of options, including more affordable cloud services (such as spot instances (Thorpe et al., 2023; Athlur et al., 2022) and serverless computing (Guo et al., 2022)) to even fully decentralized platforms (Yuan et al., 2022; Borzunov et al., 2023) that leverage a diverse set of GPUs contributed by volunteers in an extreme setting.

However, deploying large language model inference across a heterogeneous environment presents some unique challenges. Unlike traditional machine learning models, large language model inference consists of two different phases: a prompt phase that handles a sequence of input tokens at once and a decoding phase where output tokens are generated step-by-step. Additionally, large language models require the adoption of specialized *parallel inference strategies* to effectively distribute the intensive computations across multiple GPUs. The two most commonly employed approaches are *tensor model parallelism* and *pipeline parallelism*. In

[*]Equal contribution [1]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China [2]Department of Computer Science, ETH Zurich, Zürich, Switzerland [3]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania. Correspondence to: Binhang Yuan <biyuan@ust.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[ICML 2024]

HEXGEN-2: DISAGGREGATED GENERATIVE INFERENCE OF LLMS IN HETEROGENEOUS ENVIRONMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Disaggregating the prefill and decoding phases represents an effective new paradigm for generative inference of large language models (LLM), which eliminates prefill-decoding interference and optimizes resource allocation. However, it is still an open problem about how to deploy the disaggregated inference paradigm across a group of heterogeneous GPUs, which can be an economical alternative to deployment over homogeneous high-performance GPUs. Towards this end, we introduce HEXGEN-2, a distributed system for efficient and economical LLM serving on heterogeneous GPUs following the disaggregated paradigm. Built on top of HEXGEN, the core component of HEXGEN-2 is a *scheduling algorithm* that formalizes the allocation of disaggregated LLM inference computations and communications over heterogeneous GPUs and network connections as a constraint optimization problem. We leverage the *graph partitioning* and *max-flow* algorithms to co-optimize resource allocation, parallel strategies for distinct inference phases, and the efficiency of inter-phase key-value (KV) cache communications. We conduct extensive experiments to evaluate HEXGEN-2, i.e., on OPT (30B) and LLAMA-2 (70B) models in various real-world settings, the results reveal that HEXGEN-2 delivers up to a 2.0× and on average a 1.3× improvement in serving throughput, reduces the average inference latency by 1.5× compared with state-of-the-art systems given the same price budget, and achieves comparable inference performance with a 30% lower price budget.

1 INTRODUCTION

Large Language Models (LLMs), such as OPT Zhang et al. (2022), LLAMA Touvron et al. (2023), GPT OpenAI (2024), GEMINI Reid et al. (2024), CLAUDE Anthropic (2024) and MIXTRAL Jiang et al. (2024a) have shown exceptional performance across various advanced applications. However, deploying the generative inference service for such LLMs can be costly, typically requiring a substantial number of homogeneous, high-performance GPUs to meet the service demands, such as first token latency and generation throughput. In this paper, we explore an alternative solution that *deploys the most advanced disaggregated generative inference paradigm over a set of heterogeneous GPUs to provide an efficient and economical LLM service.*

Disaggregated inference is currently the most *efficient* framework for serving the generative inference requests of LLMs Zhong et al. (2024); Patel et al. (2024). By splitting the prefill phase (compute-bounded) and decoding phase (HBM IO-bounded) across different GPUs, the disaggregation significantly reduces interference between different requests and enables more flexible parallel configurations for the two phases. When compared with colocating the prefill and decoding computations, the disaggregated approach optimizes resource usage and enhances the scalability and efficiency of the LLM inference service. Recent efforts Jiang et al. (2024b); Griggs et al. (2024); Zhao et al. (2024); Miao et al. (2024) have shown that serving LLMs with heterogeneous GPUs can be a *economical* alternative to deploying over homogeneous high-performance GPUs. Heterogeneous deployments offer significant opportunities to reduce inference service costs by leveraging the wide availability of diverse GPU types across commercial and private computing platforms. Note that Nvidia typically releases new GPU generations every 24 months, e.g., Turing in 2018, Ampere in 2020, Hopper in 2022, and Blackwell scheduled for Q4 2024; but one particular version of GPU general remains in use for a much longer period[1].

[1]For example, Tesla K80 GPUs, released in 2006, are still available on AWS as `p2` instances

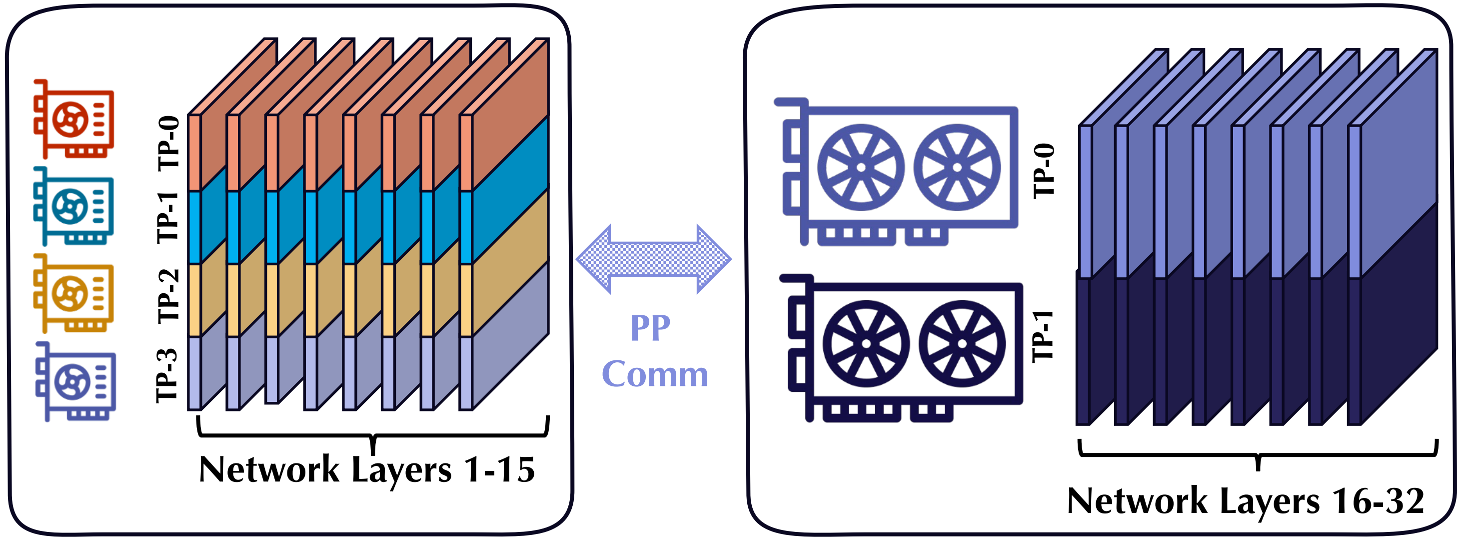
[Under Review]

HexGen

Generative Inference of Large Language Model over Heterogeneous Environment

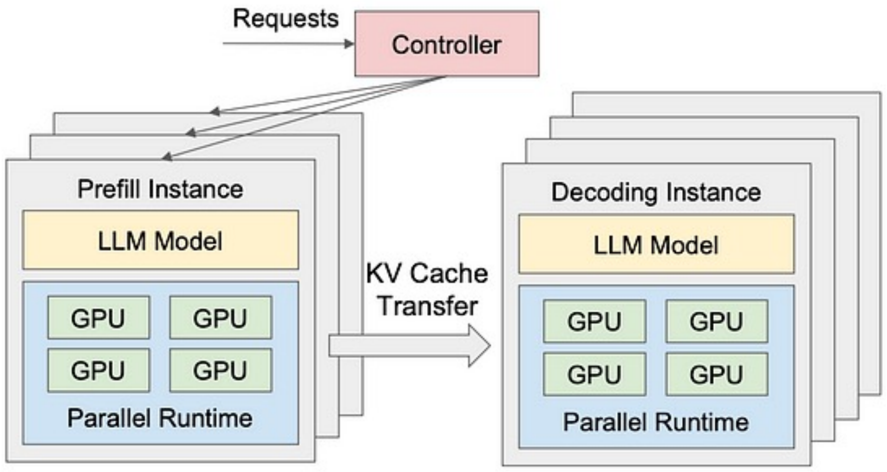
- An implementation that accommodates tensor model parallelism and pipeline parallelism.
- A scheduling algorithm that optimizes pipeline partitions and parallel strategies over heterogeneous GPUs.
  - Optimizing the layout of a pipeline through dynamic programming;
  - Solve the global scheduling through a genetic algorithm.

https://github.com/Relaxed-System-Lab/HexGen



Disaggregated Inference

- **Key ideas:**
  - Prefill computation on some GPUs;
  - Decoding computation on some other GPUs;
  - Prefill and decoding instances can have different parallel configurations;
  - Dynamic configuration of the prefill / decoding ratio;
  - Overhead: KV-cache communication.
- **Frameworks:**
  - DistServe, Splitwise.

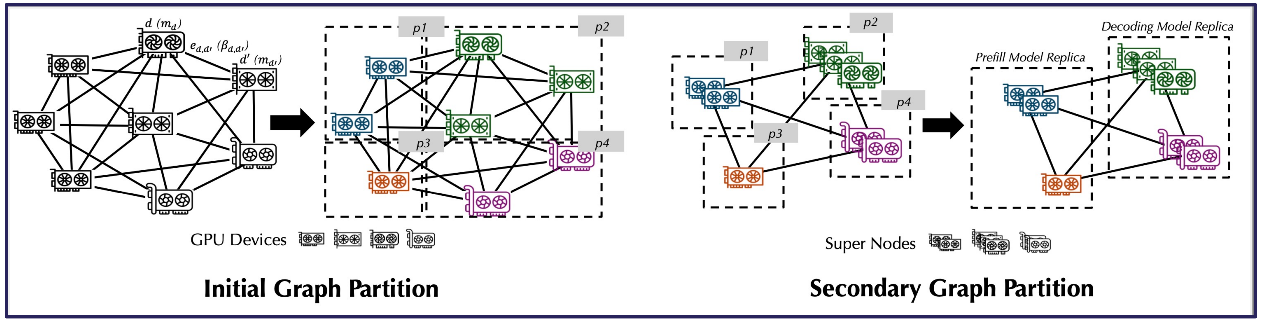


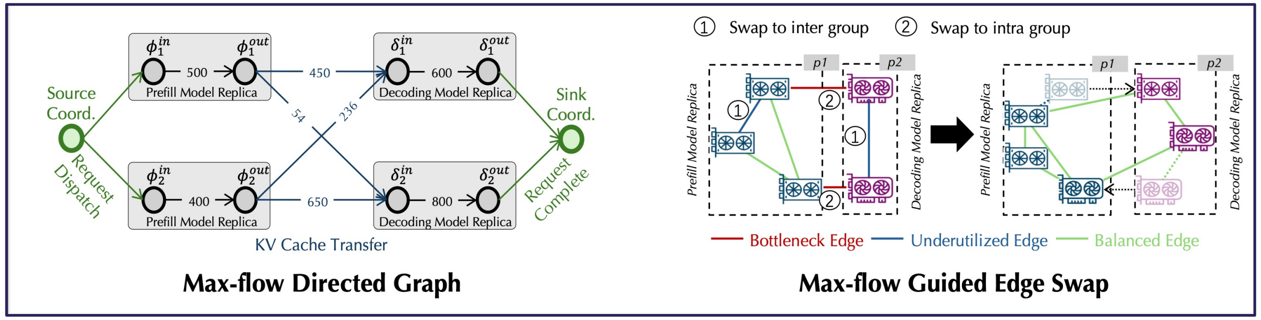
https://github.com/LLMServe/DistServe

HexGen-2

Disaggregated Generative Inference of LLMs in Heterogeneous Environment

- Scheduling for the disaggregated framework:
    - **Graph-partition**: partition the set of heterogeneous GPUs into multiple model serving groups, where each group could serve a prefill or a decoding phase;
    - **Max-flow**: find the current optimal parallel strategies for prefill and decoding model replicas and generate the optimal KV cache communication strategy among them;
    - **Iterative refinement**: we iteratively repeat the two-phase algorithm to find the optimal model placement strategy.





Summary

- ***Accommodate LLM training over heterogeneity***:
  - To accommodate heterogeneous networks, we do efficient system scheduling and algorithm design;
  - To accommodate heterogeneous computation, we do necessary system extension and efficient system scheduling.
- ***Accommodate LLM inference over heterogeneity***:
  - We support co-locating generative inference;
  - We also support disaggregated generative inference.

Personal page:
https://binhangyuan.github.io/site/

Thank you!