Efficient Scheduling of Distributed DNN Workloads

Lyra: Elastic Scheduling for Deep Learning Clusters

Jiamin Li*, Hong Xu†, Yibo Zhu‡, Zherui Liu‡, Chuanxiong Guo‡, Cong Wang*,
*City University of Hong Kong, †The Chinese University of Hong Kong, ‡ByteDance Inc.

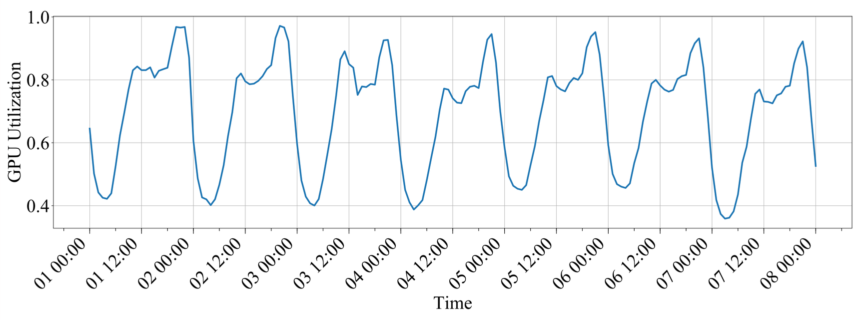
ByteDance 字节跳动 · CityU 香港城市大學 City University of Hong Kong · 香港中文大學 The Chinese University of Hong Kong

Introduction

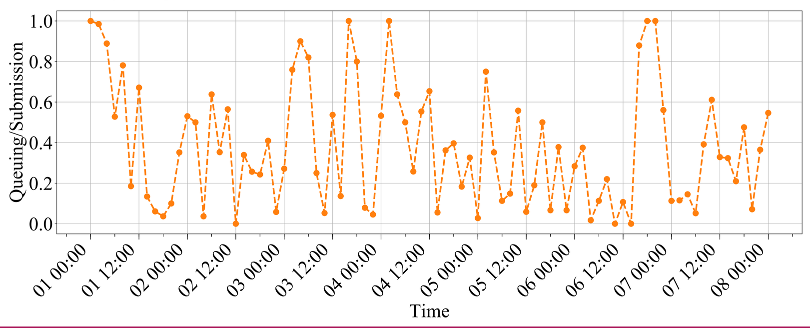
[Separate management of training and inference cluster.]

Inference requires less computation and GPU memory than training, therefore using weaker GPUs like Nvidia T4, with a fraction of the resources of the training GPUs, such as Nvidia V100 and A100.

Diurnal pattern: ~40% cluster utilisation



Long queuing: ~10,000s (95%ile)

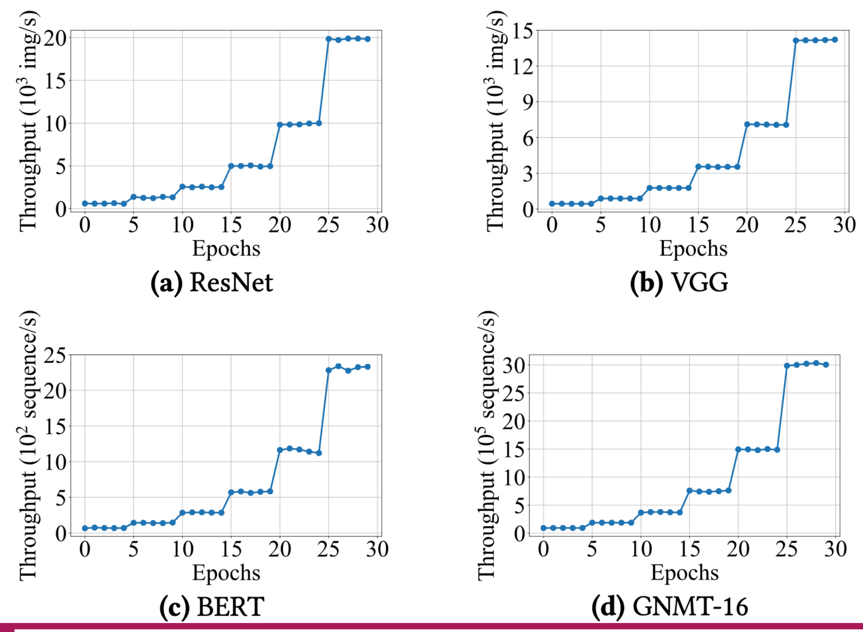


[Elastic scaling of distributed training jobs.]

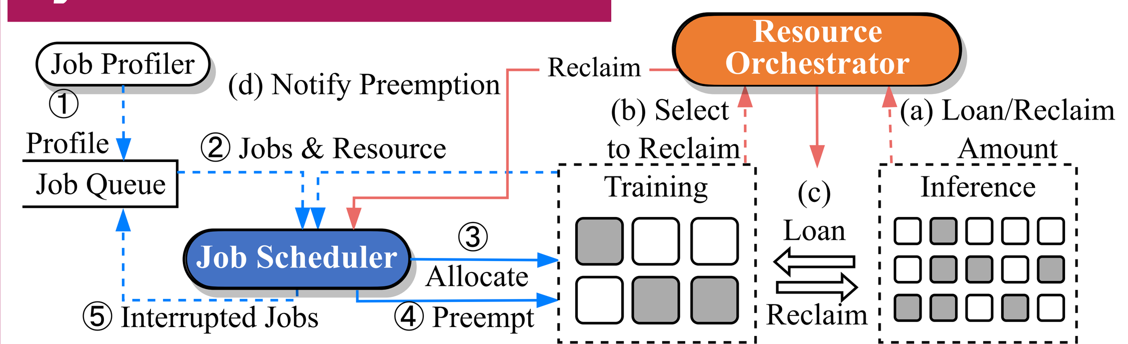
Jobs can take a <u>variable</u> number of workers according to resource availability. One can even adjust the number of workers on-the-fly when the job is running.

[Limited elasticity] Some model families enjoy a linear scaling efficiency within a range.

~5% of all jobs (account for 36% of training cluster resources)

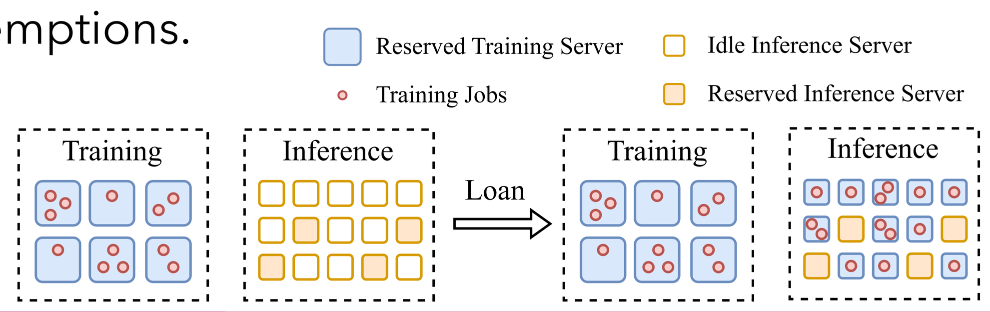


System Architecture



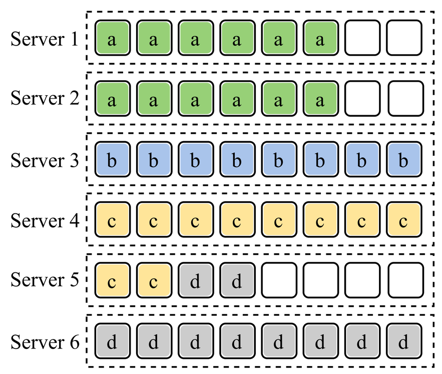
[Capacity Loaning] Exploit the unused inference resources to run training jobs temporarily, i.e. loaning inference capacity for training.

[Elastic Scaling] Training jobs can dynamically scale out to use more GPUs to accelerate training and scale in to free some servers without high-overhead preemptions.



Capacity Loaning

Which on-loan servers should be reclaimed to minimise preemptions?



For a 2-server reclaim request…

Naively selecting servers leads to unnecessary preemption.

Job placement is usually messy and servers have inter-dependency when co-hosting a job.

[Knapsack with dependent item values]

A new value definition: sum of job's server fraction

a. Greedily selects the lowest-value server
b. Preempt jobs & reclaim the server
c. Update the values

Job	Co-hosted by # of servers	Per Server Value
a	2	0.5
b	1	1
c	2	0.5
d	2	0.5

Elastic Scaling

<u>Shortest-job-first is not always optimal.</u>

Cluster Capacity: 8

Job	w^{min}	w^{max}	Min. running time
A	2	6	50
B	2	6	20

Sol.	Initial Allocation		JCT		Average JCT
	A	B	A	B	
1	6	2	50	53.33	51.67
2	2	6	63.33	20	41.67
3	4	4	60	30	45

Job	w^{min}	w^{max}	Min. running time
A	2	**3**	100
B	2	6	20

Sol.	Initial Allocation		JCT		Average JCT
	A	B	A	B	
1	3	5	100	24	62
2	2	6	106.67	20	63.33

Elastic job = Base (first-class citizen) + Flexible Demand

[Two-phase resource allocation]

a. Prioritise Base Demand SJF to minimise queuing time,
b. Allocate the remaining resources to fulfill the Flexible Demand to minimise running time.

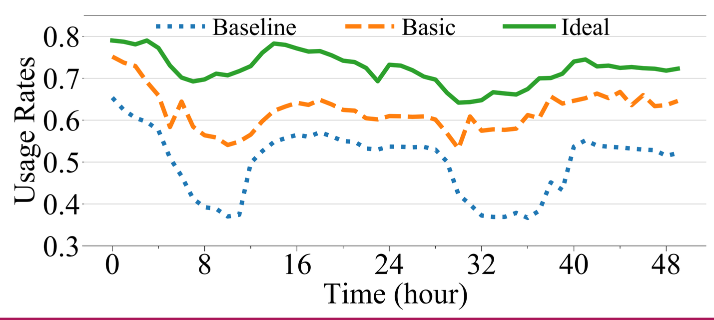
Group	Item	Weight	Value
A	1	2	50
B	1	1	20
	2	2	30
	3	3	36
	4	4	40

Multiple-choice Knapsack problem

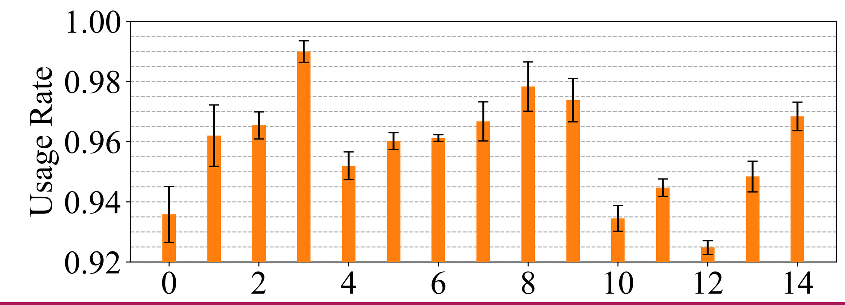
Evaluation

Avg. Queuing time:
1.52x -> 1.67x -> 2.66x
Avg. JCT:
1.48x -> 1.59x -> 1.87x

#	Scenario	Scheme	Queuing Time (s) Mean	Median	95%ile	JCT (s) Mean	Median	95%ile	GPU Usage Training	Overall [1]	Preemption Ratio [2]
1	—	Baseline [3]	3072	55	8357	16610	791	82933	0.72	0.52	0
2	Basic	Lyra	2010	26	3358	11236	568	56477	0.86	0.65	12.24%
3	Advanced		1835	24	3238	10434	525	56553	0.86	0.68	7.35%
5	Ideal		1157	22	3204	8891	422	41146	0.93	0.72	5.72%



Utilisation of on-loan servers: ~93%



Accelerating Distributed MoE Training and Inference with Lina

Jiamin Li*, Yimin Jiang‡, Yibo Zhu, Cong Wang*, Hong Xu†
*City University of Hong Kong, ‡ByteDance Inc., †The Chinese University of Hong Kong

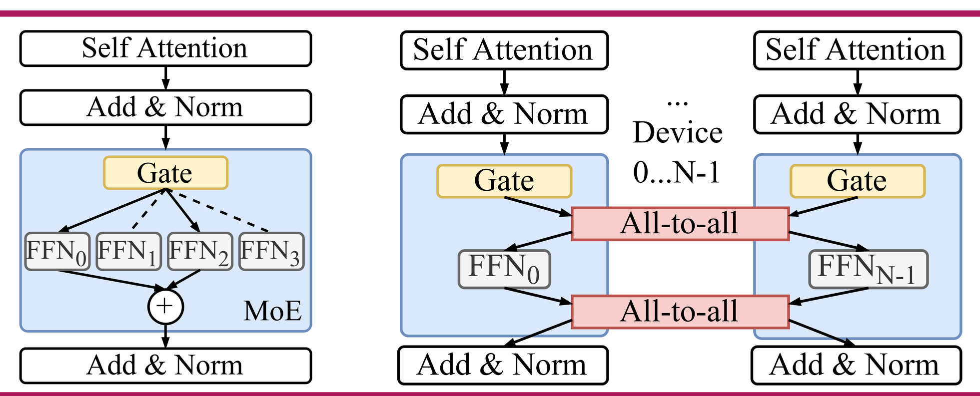
CityU 香港城市大學 City University of Hong Kong · 香港中文大學 The Chinese University of Hong Kong

Introduction

Mixture-of-Experts (MoE): a popular way to curb the computation cost of deep learning models.

[MoE in language models] MoE layer replaces the FFN layer in Transformer. It consists of multiple FFNs as experts, and a gating network. The gating network dispatches the token to a small number of experts (top-1, top-2).

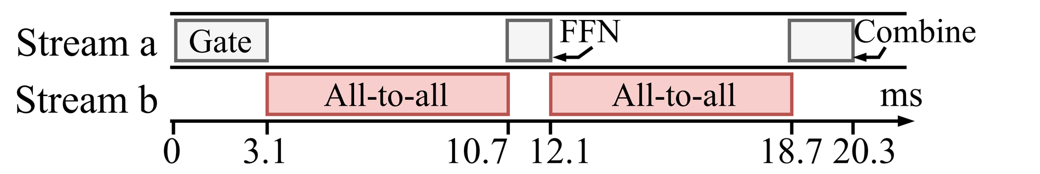
[Distributed MoE] Data parallelism and expert parallelism are applied. It allocates one unique GPU for each expert and use <u>all-to-all</u> to exchange tokens.



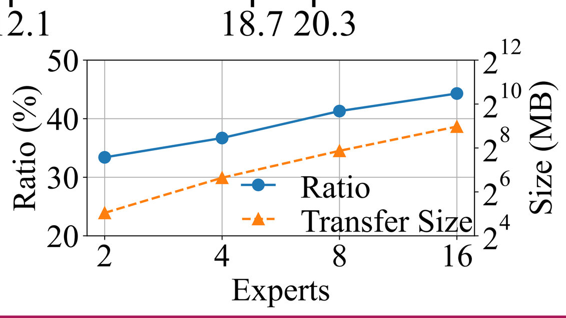
Motivation

All-to-all is the bottleneck in distributed MoE. But why?

[Synchronous all-to-all with large data transfer]

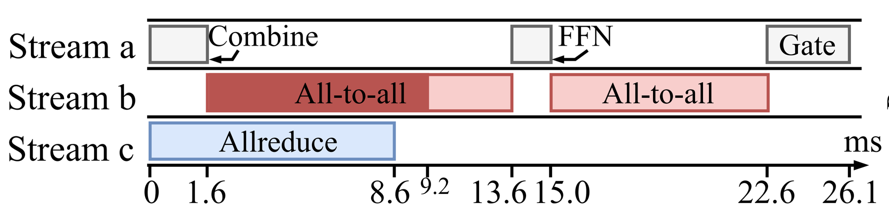


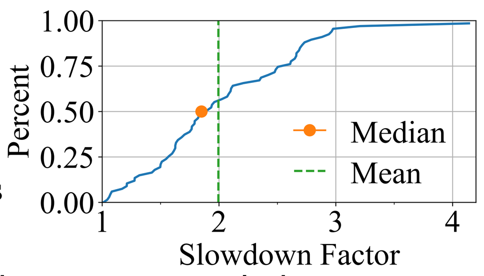
74.9% of the running time of one MoE layer.



Training

[Prolonged all-to-all with allreduce]

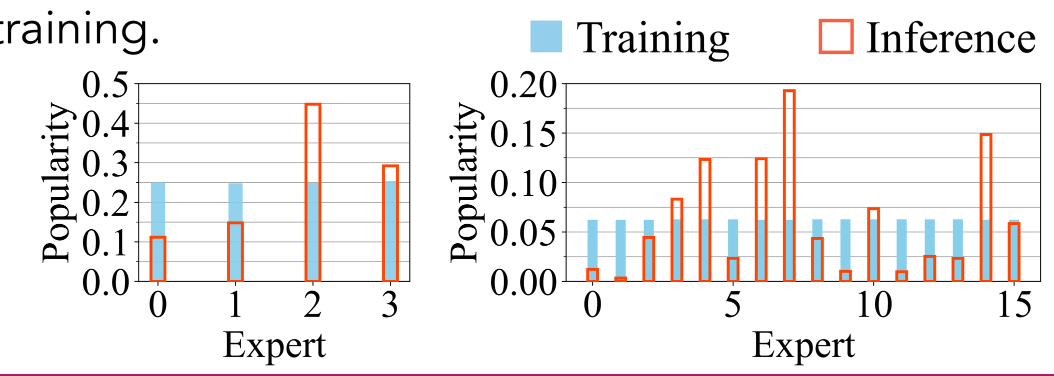




In the backward pass, all-to-all and allreduce control their own process group and overlap, they contend for the network bandwidth and their completion times are severely prolonged.

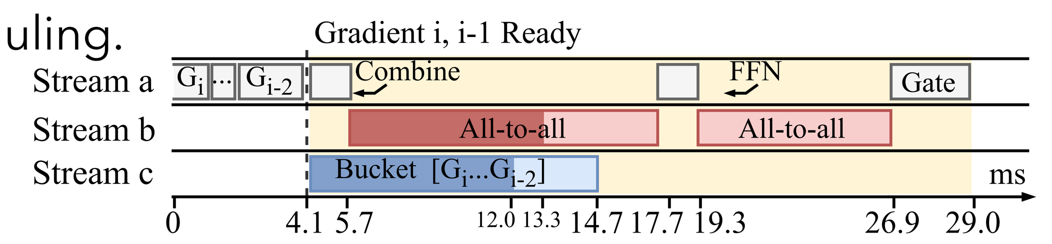
Inference

[Skewed expert popularity] Token-to-expert distribution in inference is purely workload-driven and highly skewed in sharp contrast to training.

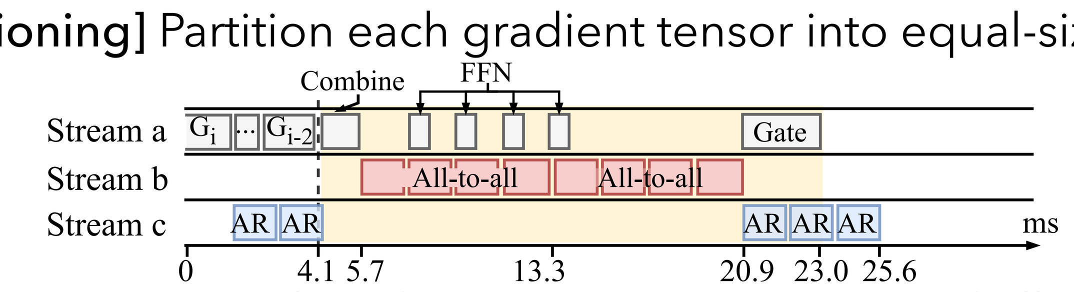


Design

Lina <u>prioritizes all-to-all</u> and avoids concurrent execution with allreduce with priority scheduling.

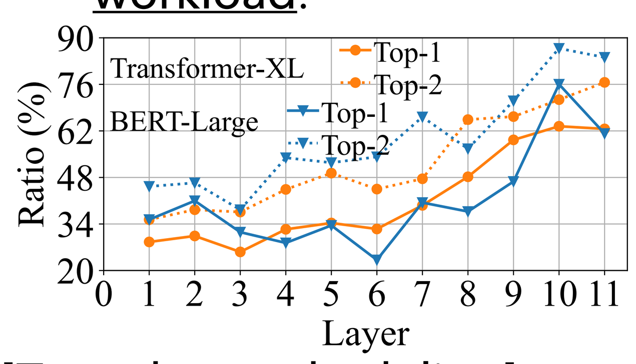


[Tensor partitioning] Partition each gradient tensor into equal-sized small chunks.



[Pipelining micro-ops] Pipeline the expert computation and all-to-all micro-ops, because the FFN computation is in token granularity.

Lina replicates popular experts on proportionally more devices to <u>balance the workload</u>.



How to know the expert popularity a prior?

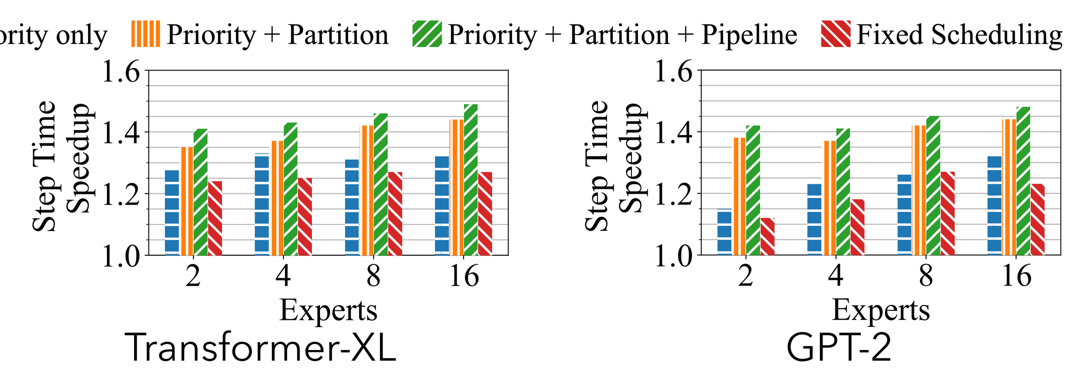
[Pattern in expert selection] Tokens that have selected the same expert in layer i tend to select the <u>same</u> expert again in layer i + 1.

[Two-phase scheduling]

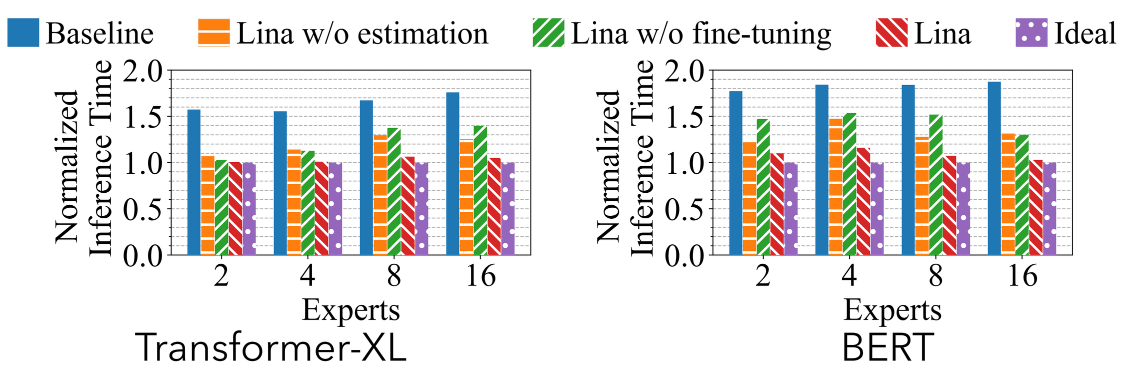
a. Resource scheduling based on estimated popularity
   - Estimate with patterns profiled during training
b. Low-overhead fine-tuning on actual routing decision

Evaluation

Reduce the training step time by 1.73x.



Reduce the 95%ile inference time by ~1.63x



[1] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, De- hao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling gi- ant models with conditional computation and automatic sharding. arXiv preprint arXiv:2006.16668, 2020.
[2] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arXiv preprint arXiv:2101.03961, 2021.
[3] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Min- jia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. arXiv preprint arXiv:2201.05596, 2022.

jiaminli.icy@gmail.com