Aristos: Pipelining One-sided Communication in Distributed Mixture of Experts

Osayamen J Aimuyo
Cornell University
oja7@cornell.edu

ABSTRACT

We propose *Aristos* [1] a communication-optimal, distributed algorithm that uses *asynchronous communication* interleaved with computation for specifically tackling the communication overhead of Distributed Mixture-of-Experts (DMoE) transformer models. DMoE as implemented today entails frequent synchronous *All-to-All* communication operations that scale linearly *per step* with a model's number of layers. Thus, we first seek clarification on how *All-to-All* bottlenecks DMoE and whether the interconnect or communication algorithm is at fault.

We investigate more than **21k** *All-to-All* CUDA kernels and empirically confirm that their runtimes exhibit a significant long-tail distribution in both multi-node and single-node settings, respectively. We argue that this phenomenon is a shortcoming of the *global barrier* necessitated by the synchronous implementation of *All-to-All* in state-of-the-art collective libraries. We use these empirical insights to motivate Aristos, which obviates the global barrier, instead favoring asynchronous communication yielding native support for pipelining. Aristos also exposes tunable hyperparameters that navigate the tradeoff of faster performance and reduced token dropping.

1. INTRODUCTION

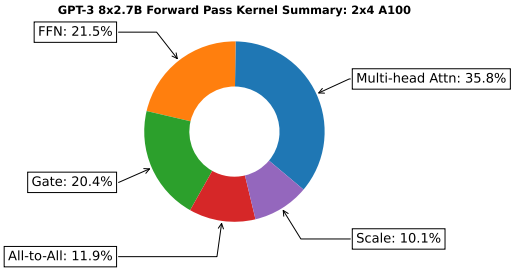
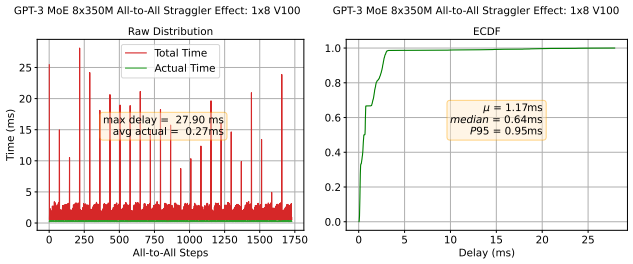


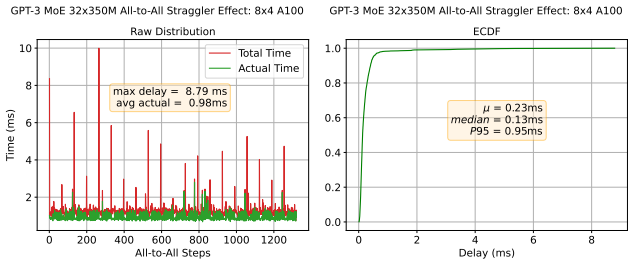
Figure 1: DMoE Training Step CUDA Kernel Distribution on a supercomputer. Noice the *All-to-All* overhead fraction.

In this work [2], we focus on the overhead (Figure 1) of synchronous *All-to-All*, within Distributed MoE [3] and make the following contributions. Firstly, we empirically demonstrate that the synchronous constraint of *All-to-All* as implemented by NCCL degrades performance by up to 100X due to the open *straggler effect* problem [1]. Next, we introduce Aristos, a distributed algorithm that obviates this synchronous barrier, in favor of one-sided communication primitives and maximizes concurrent overlap of communication and computation specifically for the MoE layer. Also, we provide a proof sketch verifying the liveness property of Aristos. Further, we sketch an optimization for topology-aware expert parallelism aiming at minimizing communication and computation costs, while satisfying memory constraints.

[1]Translation of **optimal** in Greek
[2]Notebook for all plots and traces



(a) NDv2 (b) NDv2 ECDF



(c) 8x4 GPUs (d) 8x4 GPUs ECDF

Figure 2: Straggler Effect in DMoE Training. **Actual Time** t_a denotes time for the slowest worker to complete the kernel, while **Total Time** t includes the time the fastest worker had to *wait*. Note $Delay = t - t_a$ and μ denotes mean.

2. EXPERIMENTAL METHODS

To isolate communication bottlenecks, we only consider Distributed Data Parallelism (DDP) and expert parallelism [3]. Improving the data collection of related work [4], we use NVIDIA Nsight Systems to profile over 21k *All-to-All*

CUDA kernels spanning training runs[3] of a GPT-3 350M MoE model on Azure NDv2 and 32 GPUs across 8 Perlmutter nodes. Note we performed all experiments on *dedicated* hardware allocations without sharing.

Figure 2 makes it clear that the interconnect is not the bottleneck, but rather the synchronization delay due to the implicit barrier of synchronous *All-to-All*. We clarify that the multi-node results should be much *worse* on commodity clusters that lack the high-speed interconnect or RDMA stack of Perlmutter's Slingshot-11 [2].

3. ALGORITHM

Notation	Description
$H(x,\, y)$	Heaviside Function $H(x)$, where $H(0) = y$
d	Data parallel degree, $d \in \mathbb{N}$
m	model parallel degree, $m \in \mathbb{N}$
k	Number of experts selected during token routing, $k \in \mathbb{N}$ see *top_k* in [5]
X	Set of all experts
N	Set of all nodes, $N \subset \mathbb{Z}^+$
J_n	Set of all ranks in node n, $J_n \subset \mathbb{Z}^+$
W	World of workers, note $\lvert N\rvert \cdot \lvert J_n\rvert = \lvert W\rvert = m \cdot d$
ε	Optimism factor, $\varepsilon \in \{0, \ldots, \lvert W\rvert - 1\} \subset \mathbb{Z}^+$, $\forall w \in W\ \varepsilon_w = \varepsilon$
G_j^n	Worker with rank j in node n
X_j^n	Set of experts hosted on G_j^n, $X_j^n \subseteq X$
ϕ	Tensor mapping G_j^n tokens T to experts $X' \subseteq X$, $\phi \in \mathbb{R}^{\lvert T\rvert \times k}$
S	Sharding specification tensor mapping X to $W' \subseteq W$

Table 1: Notation

Algorithm 1: Aristos; executed by each G_j^n

```
   Data: Q: a blocking FIFO queue of tasks
   Require: X_j^n, S, φ, ε, |W|
 1 begin
 2     ξ ← ε
 3     G_φ ← GetWorkers(φ, S)
 4     μ ← |G_φ| + H(|X_j^n|, 0) · (|W| − 1)
 5     flag ← False
 6     W' ← S.W'
 7     if G_j^n ∈ G_φ then
 8         μ ← μ + 1
 9         ξ ← ξ + 1
10     end if
11     broadcast(shouldProcess, φ, G_φ)
12     while Q.timeout == False and μ > 0 do
13         if flag == False and ξ == 0 then
14             Q.ActivateCountdown()
15             flag ← True
16         end if
17         t ← Q.take()
18         if t.processed == True then
19             PostProcessing(φ, t)
20             ξ ← ξ − 1
21             μ ← μ − 1
22         else if t.shouldProcess == True then
23             θ = ExpertProcessing(t, X_j^n)
24             Send(processed, θ, t.workers)
25             W' ← W' − t.workers
26             μ ← μ − 1
27     end while
28     if ε > 0 and Q.timeout == True then
29         broadcast(processed, W')
30     end if
31 end
```

In addition to pipelining, Aristos enables early-stopping via *timeouts*, wherein a fast worker can discontinue the collective communication and proceed to their next independent task, albeit without updated logits for their tokens and dropping the tokens of other participating workers. Aristos exposes a hyperparameter ε to navigate this tradeoff between performance and token dropping. We defer a complete overview and later optimizations of Aristos to the full version of the paper, but briefly present the following.

THEOREM 3.1. *Assume all processes $w \in W$ are correct with no failures, links are reliable and messages arrive in the order they are sent. If $\forall w \in W$, $\varepsilon_w = \varepsilon$, then Aristos satisfies liveness.*

Define τ_i as the timeout time units for a process p_i. Also, define a *passive process* as one that has used their timeout and thus terminated the algorithm. We start with an example and follow with a proof sketch. Consider two processes p_1, p_2 where p_1 sets $\varepsilon_1 = 0$ and p_2 sets $\varepsilon_2 = 1$. Consider the case where p_1, after executing line 14, times out and exits the loop, while p_2 blocks in line 17. Note that p_2 will block indefinitely. This would not be the case if $\varepsilon_1 = \varepsilon_2 = 0$ (both eventually terminate) or $\varepsilon_1 = \varepsilon_2 = 1$ The latter prevents blocking as well because line 29 forces the fastest of p_1 or p_2 to unblock the other process which they did not receive from during their execution window.

Proof Sketch. Observe that if $\varepsilon_i = 0$, then any blocked process p_i will eventually time out after τ_i and terminate the algorithm. If $\varepsilon_i > 0$ then process p_j who p_i awaits, will eventually unblock p_i after executing either lines 24 or 29.

4. TOPOLOGY-AWARE PARALLELISM

Note that we foresee deriving S by solving an optimization problem that automatically shards experts across workers distributed per some topology denoted as a graph G. Specifically, define $G = (V, E)$ as the cluster topology, where V denotes devices and E communication links. Equation 1 captures the core objective, namely finding $G^* = \{g = (V_g,\ E_g) \mid g \subseteq G\}$ the set of optimal expert parallel groups or the *topology-aware sharding specification.*

$$\min \max_{g \in G^*} \kappa \left(\pi(g) + \max_{i \in V_g} \mathcal{C}_i \right) + T_\rho(|G^*|) \tag{1}$$

subject to,

$$\sum_{j \in V_g} m_j \geq |\mathcal{X}| \qquad \forall g \in G^* \tag{2}$$

where, $\mathcal{X}$ is the set of all experts and m_j is expert memory capacity for device j. Also, κ identifies the frequency of MoE computation, $\pi(g)$ is the compute cost of g, C_i denotes the communication cost of device i and $T_\rho(|G^*|)$ is the cost of inter-group all-reduce on MoE parameters due to data parallelism.

5. ACKNOWLEDGMENTS

We thank Dr. Rachee Singh for supervising this work. We acknowledge using supercomputing resources under award DDR-ERCAP0027296 of the National Energy Research Scientific Computing Center (NERSC), a Department of Energy Office of Science User Facility supported under contract no DE-AC02-05CH11231. We thank Dr. Guila Guidi for introducing the author to NERSC Perlmutter and Dr. Erik Palmer of LBNL for clarifying Perlmutter's topology.

[3]All hyperparameters and training code are available here

6. REFERENCES

[1] A. Harlap, H. Cui, W. Dai, J. Wei, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing. Addressing the straggler problem for iterative convergent parallel ml. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, SoCC '16, page 98–111, New York, NY, USA, 2016. Association for Computing Machinery.

[2] K. S. Khorassani, C.-C. Chen, B. Ramesh, A. Shafi, H. Subramoni, and D. K. Panda. High performance mpi over the slingshot interconnect. *Journal of Computer Science and Technology*, 38(1):128–145, Feb 2023.

[3] D. Lepikhin et al. Gshard: Scaling giant models with conditional computation and automatic sharding. *CoRR*, abs/2006.16668, 2020.

[4] J. Li, Y. Jiang, Y. Zhu, C. Wang, and H. Xu. Accelerating distributed MoE training and inference with lina. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pages 945–959, Boston, MA, July 2023. USENIX Association.

[5] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv*, abs/1701.06538, 2017.