Aristos: Communication, Computation and Memory Optimality for Mixture-of-Experts

Osayamen Jonathan Aimuyo [1]

Abstract

We propose *Aristos*, a complete, automated system for multi-criteria optimization of the Distributed Mixture-of-Experts (DMoE) architecture across computation and communication latency subject to memory constraints. The centerpiece of Aristos is a fused DMoE kernel using kernel-initiated, asynchronous communication to maximize the overlap of communication and expert computation. Compared to the state-of-the-art Megatron-DeepSpeed, which uses ≥ 10 kernels and synchronous NCCL all-to-all for DMoE, Aristos is projected to have much less kernel scheduling overhead and uses NVSHMEM for inter-GPU communication. Second, Aristos introduces two novel *deterministic* locally optimal algorithms, *LysiGroup* and *LysiAssign* for *heterogeneous*, topology-aware expert parallelism and load-balanced expert allocation, respectively, while satisfying stringent memory constraints. Compared to naive parallelism heuristics from existing work, Aristos yields **1.49X** better training throughput and end-to-end iteration latency.

1. Introduction

Larger Transformer models achieve significantly better performance (Brown et al., 2020; Kaplan et al., 2020), hence the appeal to scale their size continuously. However, this scale demands exorbitant compute resources, requiring a distributed setting as memory requirements exceed the capacity of any single accelerator (GPU, TPU, FPGA, etc.) (Smith et al., 2022) This interplay begs the question: *how do we achieve efficiency when scaling large transformer models?*

[1]Department of Computer Science, Cornell University, Ithaca, NY, USA. Correspondence to: Osayamen Jonathan Aimuyo <oja7@cornell.edu>.

Copyright 2024 by the author.

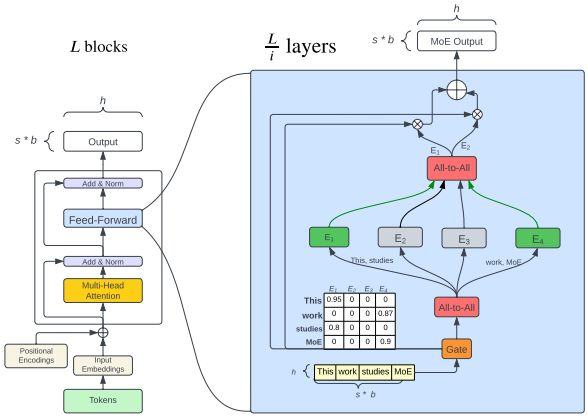


Figure 1. Distributed MoE Layer as described in (Shazeer et al., 2017; Lepikhin et al., 2020). Note the Feed-Forward Network (FFN) in the transformer blocks (left) is substituted $\frac{L}{i}$ times for the MoE layer, comprising FFNs termed *experts*. This MoE layer depicts a simple example where $k = 1$, E_j, $j \in \{1, 2, 3, 4\}$ denote experts, s is sequence length, b is minibatch and h is the embedding dimension. The shown matrix exemplifies the logits of the gate network.

1.1. Background

Existing work shows the Mixture-of-Experts (MoE) architecture (Jacobs et al., 1991) is a productive answer to this question. In current implementations (Shazeer et al., 2017; Sanseviero et al., 2023), outlined in Figure 1, the MoE architecture entails substituting the Feed-Forward Network (FFN) of a transformer encoder or decoder with n independently trained FFNs termed *experts* and a gating network, collectively comprising an MoE layer (Lepikhin et al., 2020; Du et al., 2022). The key insight is computational efficiency as only a subset of the model's parameters are activated during inference while training converges faster due to increased parameters (Fedus et al., 2021; SnowflakeAI, 2024a). For example, the recently released Mixtral 8x22B, a multilingual and coding-capable MoE model, has inference cost of 39B parameters instead of its actual 141B parameters (MistralAI, 2024). Another example comes from the DeepSpeed team of Microsoft who showed 5x less training time for GPT-3

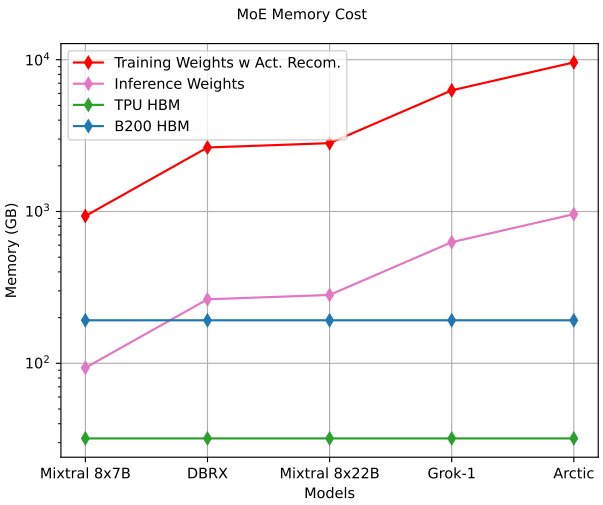


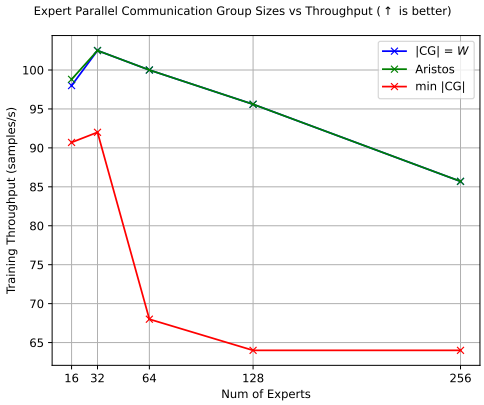
Figure 2. Training and Inference Memory Estimates for state-of-the-art MoE models. **Act. Recom.** denotes activation recomputation. We compare the memory cost to the capacity of a single accelerator, excluding optimizations such as ZeRO (Wang et al., 2023). Note that *none* of these models is trainable on any single accelerator. However, aggregating the memory of 16 B200 GPUs will allow for training the first three MoE models. We calculate memory cost using the rule of thumb described in (Smith et al., 2022)

MoE 1.3B and 7.3X faster inference for a trillion-parameter MoE model (Rajbhandari et al., 2022). Due to these efficiency benefits, it is no surprise that the MoE architecture has gained traction among enterprise Large Language Models (LLMs), namely Gemini 1.5 by Google (GeminiTeam & Google, 2024), Arctic by Snowflake (SnowflakeAI, 2024a), Grok-1 by x.ai (xAI, 2024), and DBRX by DataBricks (MosaicResearch, 2024).

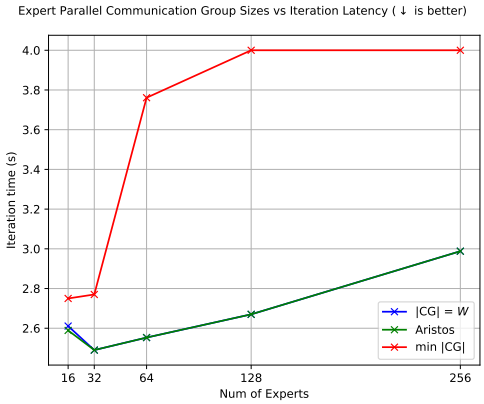
1.2. Challenges

Challenge 1 However, within the distributed setting, judiciously extracting the efficiency of MoE models is no easy feat (SnowflakeAI, 2024b; MosaicResearch, 2024). In particular, it requires an optimal *expert parallelism* strategy that deftly navigates the tradeoff of inter-worker communication, intra-worker computation and memory capacity. Observe from Figure 2 that memory is still a stringent constraint even for the bleeding-edge B200 with 192 GB HBM3 memory capacity (NVIDIA, 2024a). Further, we reify these tradeoffs in Figure 3, where naively minimizing all-to-all *communication group* sizes leads to consistently suboptimal performance with the worst at **1.49X** degradation of training throughput and latency.

This result motivates the necessity of a *topology-aware* solution revisiting prior work which recommended configuring communication groups to singular node sizes (Narayanan et al., 2021) or to span the minimum of process world or number of experts (Rajbhandari et al., 2022; Fedus et al., 2021) Indeed, determining a global optimum expert parallel strategy is a hard problem as it reduces to solving graph partitioning (largely uniform) and bin-packing with fixed bins, both of which are NP-Complete (Andreev & Räcke, 2004; Garey & Johnson, 1990).



(a) GPT-3 350M MoE training throughput on 4x4 A100 80 GB GPUs



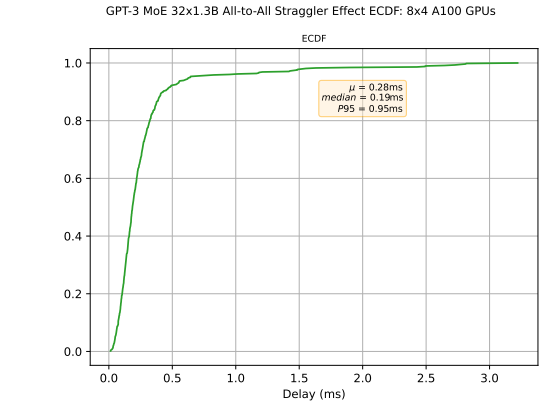
(b) GPT-3 350M MoE training iteration latency on 4x4 A100 80 GB GPUs

Figure 3. Effect of Expert parallel communication groups sizes $|CG|$ on training throughput and latency on a supercomputer. CG translates to all-to-all communication size and $\min |CG|$ denotes the smallest possible group of GPUs that can support E experts without an OOM error. Note that the worst-case GPU-GPU inter-node bandwidth of the supercomputer testbed is 25 GB/s, intra-node bandwidth is at NVLink 3.0 speed.

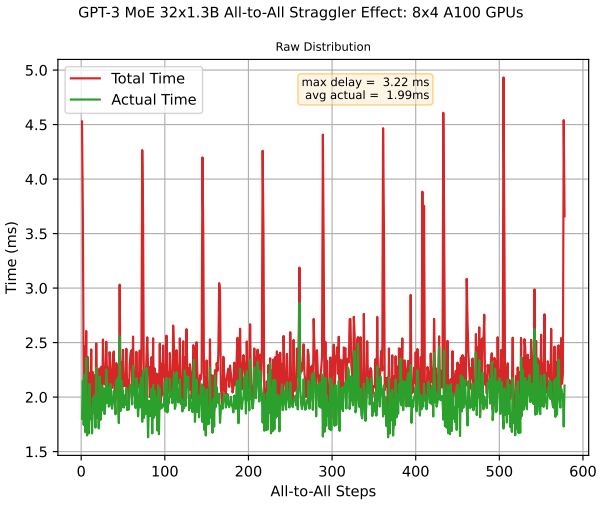
Challenge 2: In addition, all-to-all [1] as implemented by NCCL currently [2] is a *synchronous* collective operation among all participating GPUs. In this setting, disparities in processing speeds among workers induce a strag-

[1] See PyTorch's implementation here

[2] See NCCL's code here



(a) ECDF



(b) Raw Distribution

Figure 4. Straggler effect of all-to-all, when $|CG| = 32$ for 8X4 A100 GPUs. **Actual Time** t_a denotes the fastest kernel eecution time across parallel workers, conversely **Total Time** $t = Delay + t_a$, where $Delay$ is the maximum across workers

gler effect detrimental to the collective operation's performance. Specifically, as shown in Figure 4, for distributed training of a 1.3B MoE model across 32 A100 GPUs, we see P95 communication performance degradation of **1.47X** when compared to the mean actual kernel time from Figure 4b. This reduction is rather tame due to the already high-communication latency at 3 ms per all-to-all kernel. On the other hand, the performance loss is more severe in a single node of 8 V100 GPUs with higher bandwidth, where we observe p50 performance reduction of **3.37X**. In line with prior work (Bell et al., 2006; Chen et al., 2023) from the HPC community, we argue that obviating the synchronization overhead using one-sided communication primitives would mitigate the straggler effect and thus its ensuing communication performance loss.

2. Method

In addressing these two challenges, we present [3] *Aristos*[4], a three-pronged novel system for determining a topology-aware expert parallel strategy optimized across computation latency and communication costs while subject to stringent memory constraints. In the following three subsections, we outline the algorithms for each component of Aristos.

2.1. Preliminaries

Firstly, define $\gamma(\omega) \geq 1$ as the frequency, due to Distributed Data Parallelism (DDP), at which the MoE layer is executed in a single iteration.

$$\gamma(\omega) = \frac{(2+r) \cdot L \cdot B}{i \cdot \omega \cdot b} \tag{1}$$

Note that r: activation re-computation amount (Narayanan et al., 2021), L: num of layers, B: global batch, i: MoE layer frequency (Lieber et al., 2024), b: mini-batch and ω: effective world size $\omega \in [0, W]$. Also define η as the frequency of communicating over an edge. Next, we capture how MoE Expert Parallelism (EP) and DDP affect the end-to-end time of a single training iteration below. For inference, $T_\rho(|\mathcal{G}|) = 0$.

Table 1. Notation for *Lysi*

Notation	Description
W	Set of multi-node, distributed workers, $\|W\| > 0$
g	Subset of workers, $g \subseteq G$, $g = (V_g, E_g)$
$\mathcal{G}$	Set of all g, note $1 \leq \|\mathcal{G}\| \leq \|V\|$
$i,\ j$	Workers $i,\ j \in W$
α_{ij}	α cost between i and j, $\alpha \in \mathbb{R}^+$
β_{ij}	β cost between i and j, $\beta \in \mathbb{R}^+$
G	Graph adjacency matrix where $G[i,j] = (\alpha_{ij},\ \beta_{ij})$ and $G = (V, E)$, $V = W$
m_j	Expert memory capacity of worker j
$\mathcal{X}$	Set of all experts, for **valid** groups $g^* \in \mathcal{G}$, $\sum_{j \in V_{g^*}} m_j \geq \|\mathcal{X}\|$
$\pi(g)$	**Expert** compute cost of group g
f_x	Floating-point operations for computing x
$\mathcal{F}_j$	*Actual* FLOPS of worker j
$\mathcal{C}_j$	Communication cost for worker j
$T_\rho(n)$	Time for data parallel all-reduce on n processes

$$\forall g \in \mathcal{G} \quad \min \ T(g),$$
$$\text{where } T(g) = \gamma(\omega) \left(\pi(g) + \max_{i \in V_g} \mathcal{C}_j \right) + T_\rho(|\mathcal{G}|) \tag{2}$$

[3] Code available here

[4] Translation of optimal in Greek

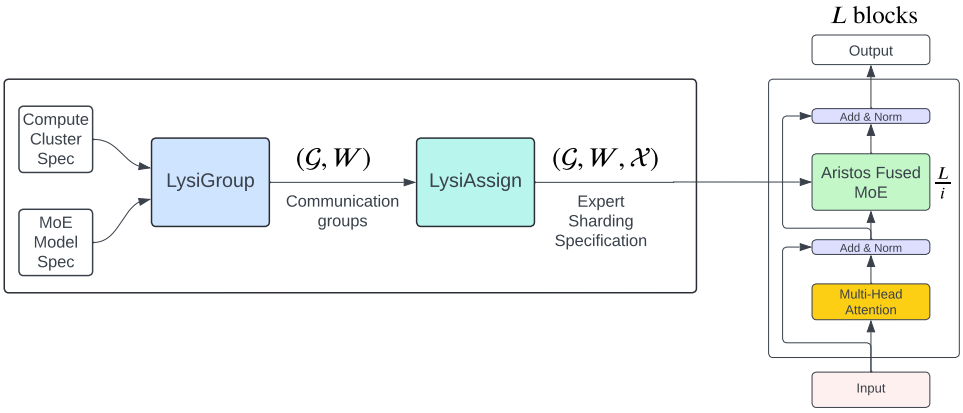


Figure 5. System architecture of Aristos. **LysiGroup** and **LysiAssign** are subcomponents of Aristos. LysiGroup receives two inputs: specifications about the MoE model and the *heterogeneous* compute cluster, notably the $\alpha - \beta$ adjacency matrix, device memory capacities and computational capabilities in FLOPs. With these inputs, LysiGroup optimizes a multi-criteria objective function capturing computation latency, communication costs, subject to memory constraints and generates an output tensor mapping workers to communication-optimal groups. For each group, LysiAssign evenly load balances assigns across heterogenous workers, subject to memory constraints. The resulting tensor is used as input to AristosFused.

We reiterate T_g below, clarifying $\pi(g)$, and $\mathcal{C}$. We define T_ρ per the worst case latency of (Rabenseifner, 2004)

$$T(g) = \begin{cases} T'(g) & \sum_{j \in V_g} m_j \geq |\mathcal{X}| \\ \infty & \text{otherwise} \end{cases} \tag{3}$$

where $T(g) \in \mathbb{R}^+$ and

$$T'(g) = \gamma(\omega) \left(\frac{\sum_{x \in \mathcal{X}} f_x}{\sum_{j \in g} \mathcal{F}_j} + \max_{i \in V_g} \sum_{(i,j) \in E_g} \eta(\alpha_{ij} + \frac{d}{|V_g|} \beta_{ij}) \right) + 2(|\mathcal{G}| - 1) \left(\alpha^* + d_{ar} \frac{\beta^*}{|\mathcal{G}|} \right) \tag{4}$$

2.2. LysiGroup

Further, like prior work (Rajbhandari et al., 2022) and for simplicity, we assume, only these parallelism types and not tensor parallelism for the MoE layers *only*. Conversely, the parallelism strategy of non-MoE layers is of no concern to this work and thus does not affect any of its results.

We solve **Equation** 2 using **LysiGroup**, a novel *deterministic* graph partitioning algorithm reminiscent of Kruskal's classical MST algorithm (Kruskal, 1956). Compared to Kruskal's, in Line 11, we only merge groups when the objective value is minimized.

Proposition 1. *Define p as a policy that determines whether $g_u \cup g_v$ optimizes $T(g)$ for any g_u and g_v. Further let $T_{uv} = T(g_u \cup g_v)$, $T_u = T(g_u)$ and $T_v = T(g_v)$. $p = T_{uv}\left(\frac{1}{T_u} + \frac{1}{T_v}\right) \leq 2$ is a max-min fair policy that optimizes $T(g)$.*

Algorithm 1: *LysiGroup*

Require: $G = (V,\ E)$, $\mathcal{F}$, f, η, d, d_{ar}
Result: $\mathcal{G}$: Groups

```
1  begin
2      Sort E ascending
3      G ← {V}, Initialize disjoint-set with V
4      while E.size() > 0 do
5          (u, v, w) ← E.pop()
6          g_u ← G.FIND(u)
7          g_v ← G.FIND(v)
8          if g_u ≠ g_v then
9              T_uv ← T(g_u ∪ g_v)
10             if [T_uv (1/T(g_u) + 1/T(g_v)) ≤ 2] then
11                 g_m ← G.UNION(g_u, g_v)
12             end if
13         end if
14     end while
15     return G
16 end
```

Proof. We need to show T_{uv} yields a net-positive outcome for the new group $g_{uv} = g_u \cup g_v$. Firstly, define $r_u = \frac{T_u - T_{uv}}{T_u}$ as the perceived outcome for g_u and likewise for g_v. Note that a selfish policy would disagree on a merge if $T_u < 0$ or $T_v < 0$. Proceeding, T_{uv} optimizes $T(g)$ when

$r_u + r_v \geq 0$, which occurs if

$$\frac{T_u - T_{uv}}{T_u} + \frac{T_v - T_{uv}}{T_v} \geq 0$$
$$\Rightarrow \frac{T_u - T_{uv}}{T_u} \geq \frac{T_{uv} - T_v}{T_v}$$
$$\Rightarrow 1 - \frac{T_{uv}}{T_u} \geq \frac{T_{uv}}{T_v} - 1$$
$$\Rightarrow 2 \geq \frac{T_{uv}}{T_u} + \frac{T_{uv}}{T_v}$$
$$\Rightarrow 2 \geq T_{uv}\left(\frac{1}{T_u} + \frac{1}{T_v}\right)$$
$$\Rightarrow T_{uv}\left(\frac{1}{T_u} + \frac{1}{T_v}\right) \leq 2$$

thus completing the proof. □

In addition, **LysiGroup** runs in $\mathcal{O}(|E|logV)$, where E denotes the set of edges in the topology and V the set of machines or GPUs. Note $|E| = \frac{V(V-1)}{2}$ since most physical network topologies are complete graphs.

2.3. LysiAssign

LysiAssign is a best-fit approximation algorithm that solves the bin packing problem defined by the following.

$$\min \quad \frac{\sum_{x \in \mathcal{X}} y_{xj}}{\mathcal{F}_j} \qquad \forall j \in W_g \tag{5}$$

subject to

$$\sum_{x \in \mathcal{X}} y_{xj} \geq m_j \qquad \forall j \in W_g \tag{6}$$
$$\sum_{g \in \mathcal{G}} \sum_{j \in W_g} y_{xj} = |\mathcal{X}| \qquad \forall j \in W_g, x \in \mathcal{X} \tag{7}$$
$$\sum_{j \in W_g} y_{xj} = 1 \qquad \forall x \in \mathcal{X} \tag{8}$$
$$y_{xj} \in \{0, 1\} \qquad \forall j \in W_g, x \in \mathcal{X} \tag{9}$$

y_{xj} denotes a mapping of expert $x \in \mathcal{X}$ to worker $j \in W$. Equation 6 captures the per-worker memory constraint while Equation 7 ensures the allocation of all experts. Equation 8 enforces that an expert is mapped to one worker only. A worker can host many experts, but the converse is invalid for our setting. LysiAssign adopts a best-fit greedy approach with complexity $\mathcal{O}(|\mathcal{X}| \log |\mathcal{X}|)$, where $|\mathcal{X}| \geq W$

2.4. Aristos Fused MoE

With $\mathcal{S}$ generated by LysiAssign, Aristos executes Algorithm 3. The key insight of this algorithm is that we decouple the all-to-all into one-sided remote send and receive operations, thus obviating synchronization overhead. We anticipate implementing this procedure in CUDA using NVSHMEM (NVIDIA, 2024c) which supports kernel-initiated communication routines.

Algorithm 2: *LysiAssign* allocates experts $\mathcal{X}$ to workers W_g while optimizing computation and satisfying memory constraints m

Require: W_g, $\mathcal{X}$, m, $\mathcal{F}$, f
Result: $\mathcal{S}$: Assignment
begin
  $W'_g = \texttt{Sort}(W_g)$, descending by FLOPS
  $B \leftarrow [\mathcal{X}]$, binary search structure sorted by f_x
  while $B.size() > 0$ **do**
    $j \leftarrow W'_g.\texttt{RoundRobinNext}()$
    $budget \leftarrow \left\lceil \frac{\mathcal{F}_j \cdot \sum_{x \in \mathcal{X}} f_x}{\sum_{i \in W'_g} \mathcal{F}_i} \right\rceil$
    while $budget > 0$ **and** $m_j > 0$ **and** $\mathcal{X}.size() > 0$ **do**
      $x \leftarrow \texttt{BestFit}(budget, B)$
      $\mathcal{S}[j].\texttt{Append}(x)$
      $B \setminus x$
      $budget \leftarrow budget - f_x$
      $m_j \leftarrow m_j - 1$
    end while
    if $m_j == 0$ **or** `Wrap` $() ==$ **False then**
      $W'_g \setminus j$
    end if
  end while
  return $\mathcal{S}$
end

Table 2. Aristos Fused Notation

Notation	Description
$H(x,\ y)$	Heaviside Function $H(x)$, where $H(0) = y$ following (Contributors, 2023)
k	Number of experts selected during token routing, $k \in \mathbb{N}$ see *top_k* in (Shazeer et al., 2017; Fedus et al., 2021)
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 $\|N\| \cdot \|J_n\| = \|W\| > 0$
ε	Optimism factor, $\varepsilon \in \{0, \ldots,\ \|W\| - 1\} \subset \mathbb{Z}^+, \forall w \in W\ \varepsilon_w = \varepsilon$
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 $X' \subseteq X$, $\phi \in \mathbb{R}^{\|T\| \times k}$
S	Sharding specification tensor mapping X to $W' \subseteq W$

3. Evaluation

We implement LysiAssign and LysiGroup in ≈ 600 lines of Python and also include a Jupyter Notebook demonstrating its capabilities. In addition, we evaluated both components on the Perlmutter and compared against naive expert parallel heuristics, namely single-node and world-size communication groups due to Narayan et al. (Narayanan et al., 2021)

Algorithm 3: *Aristos* executed by each G_j^n

Data: Q: a blocking task FIFO queue
Require: X_j^n, S, ϕ, ε, $|W|$

begin
  $\xi \leftarrow \varepsilon$
  $G_\phi \leftarrow \textbf{GetWorkers}(\phi,\ S)$
  $\mu \leftarrow |G_\phi| + H(|X_j^n|,\ 0) \cdot (|W| - 1)$ $flag \leftarrow$ **False**
  $W' \leftarrow S.W'$
  if $G_j^n \in G_\phi$ **then**
    $\mu \leftarrow \mu + 1$
    $\xi \leftarrow \xi + 1$
  end if
  $\textbf{broadcast}(shouldProcess,\ \phi,\ G_\phi)$
  while $Q.timeout ==$ **False and** $\mu > 0$ **do**
    if $flag ==$ **False and** $\xi == 0$ **then**
      $Q.\textbf{ActivateCountdown}()$
      $flag \leftarrow$ **True**
    end if
    $t \leftarrow Q.\textbf{take}()$
    if $t.processed ==$ **True then**
      $\textbf{PostProcessing}(\phi,\ t)$
      $\xi \leftarrow \xi - 1$
      $\mu \leftarrow \mu - 1$
    else if $t.shouldProcess ==$ **True then**
      $\theta = \textbf{ExpertProcessing}(t.\tau,\ X_j^n)$
      $\textbf{Send}^\dagger(processed,\ \theta,\ t.workers)$
      $W' \leftarrow W' - t.workers$
      $\mu \leftarrow \mu - 1$
  end while
  if $\epsilon > 0$ **and** $Q.timeout ==$ **True then**
    $\textbf{broadcast}(processed,\ W')$
  end if
end

and Shazeer et al. (Fedus et al., 2021) and Rajbhandari et al. (Rajbhandari et al., 2022), respectively.

3.1. Perlmutter

Our hardware testbed is the Perlmutter, ranked 14 in the top500 supercomputer list [5]. The cluster comprises 1792 GPU nodes, with each node having 4 A100 GPUs interconnected by NVLink 3.0. Each GPU has its own NIC, supporting RDMA communication bandwidth of 25 GB/s to any other inter-node GPU. Further details on the topology are available in (NERSC, 2024).

[5]Referenced here

3.2. Experiment Plan

We perform all training experiments using the GPT-3 MoE model from our forked [6] repository of Megatron-Deepspeed. Using NCCL-Tests and NVSHMEM benchmarks, we micro-evaluate the cluster to get the $\alpha - \beta$ adjacency matrix. Further, we run memory assessments to determine the expert memory capacity m_j for a single A100 40 GB or 80 GB GPU. The former is 16 experts and the latter is 32 experts for the GPT-3 350 model. We feed these inputs into LysiGroup and LysiAssign and use the sharding specification $\mathcal{S}$ to perform training on 4x4 A100 80 GB GPUs We profile training for at least 50 iterations and at most 80 iterations and report our results in Figure 3.

3.3. Results

Observe that Aristos consistently yields the best latency and throughput compared to both naive baselines. Importantly, for $|\mathcal{X}| \geq 64$, executing the model becomes compute-bound hence the huge spike in latency for the $\min |CG|$ heuristic. At 128 experts, Aristos yields a best case iteration latency improvement of **1.498X** and throughput increase of **1.493X** over the naive single-node communication grouping heuristic from existing work (Narayanan et al., 2021).

4. Conclusion & Discussion

We conclude by firstly highlighting that Aristos deviates from the norm of using Linear Program (LP) solvers (Tarnawski et al., 2020; Zheng et al., 2022) but instead develops novel deterministic approximation algorithms that run well in practice. Was this departure necessary? We answer resoundingly in the affirmative because linear programs are unpredictable in runtime and even feasibility (Tarnawski et al., 2020). Further, *tractable* programs have computational complexity exponential in the number of constraints and quadratic with respect to the number of variables (Papadimitriou, 1981). So, even though Aristos provides no guarantee of finding the globally optimal solution, it compensates with its simplicity and low computation footprint. For example, it is possible, even encouraged, for Aristos to integrate with a traffic-engineering paradigm, wherein periodically the algorithm is executed to better reflect the state of the cluster's hardware. We finalize by reiterating that future work of Aristos includes development of the fused CUDA kernel using bleeding-edge libraries like CUBLASdx (NVIDIA, 2024b), which supports in-kernel GEMMs and holistically evaluating with cutting-edge work, namely Tutel (Hwang et al., 2023), SmartMoE (Zhai et al., 2023), and HetuMoE (Nie et al., 2022).

[6]Repository available here

References

Andreev, K. and Räcke, H. Balanced graph partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, pp. 120–124, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138407. doi: 10.1145/1007912.1007931. URL https://doi.org/10.1145/1007912.1007931.

Bell, C., Bonachea, D., Nishtala, R., and Yelick, K. Optimizing bandwidth limited problems using one-sided communication and overlap. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, pp. 10 pp.–, 2006. doi: 10.1109/IPDPS.2006.1639320.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Chen, Y., Brock, B., Porumbescu, S., Buluc, A., Yelick, K., and Owens, J. Atos: A task-parallel gpu scheduler for graph analytics. In *Proceedings of the 51st International Conference on Parallel Processing*, ICPP '22, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450397339. doi: 10.1145/3545008.3545056. URL https://doi.org/10.1145/3545008.3545056.

Contributors, P. Torch.heaviside, 2023. URL https://pytorch.org/docs/stable/generated/torch.heaviside.html.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., Zoph, B., Fedus, L., Bosma, M. P., Zhou, Z., Wang, T., Wang, E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K., Duke, T., Dixon, L., Zhang, K., Le, Q., Wu, Y., Chen, Z., and Cui, C. GLaM: Efficient scaling of language models with mixture-of-experts. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021. URL https://arxiv.org/abs/2101.03961.

Garey, M. R. and Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990. ISBN 0716710455.

GeminiTeam and Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

Hwang, C., Cui, W., Xiong, Y., Yang, Z., Liu, Z., Hu, H., Wang, Z., Salas, R., Jose, J., Ram, P., Chau, J., Cheng, P., Yang, F., Yang, M., and Xiong, Y. Tutel: Adaptive mixture-of-experts at scale, 2023.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 03 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Kruskal, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. ISSN 00029939, 10886826. URL http://www.jstor.org/stable/2033241.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. *CoRR*, abs/2006.16668, 2020. URL https://arxiv.org/abs/2006.16668.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, S., Belinkov, Y., Shalev-Shwartz, S., Abend, O., Alon, R., Asida, T., Bergman, A., Glozman, R., Gokhman, M., Manevich, A., Ratner, N., Rozen, N., Shwartz, E., Zusman, M., and Shoham, Y. Jamba: A hybrid transformer-mamba language model, 2024.

MistralAI. Cheaper, better, faster, stronger, 2024. URL https://mistral.ai/news/mixtral-8x22b/.

MosaicResearch. Introducing dbrx: A new state-of-the-art open llm, 2024. URL https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm.

Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., Phanishayee, A., and Zaharia, M. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3476209. URL https://doi.org/10.1145/3458817.3476209.

NERSC. Perlmutter architecture, 2024. URL https://docs.nersc.gov/systems/perlmutter/architecture/.

Nie, X., Zhao, P., Miao, X., Zhao, T., and Cui, B. Hetumoe: An efficient trillion-scale mixture-of-expert distributed training system, 2022.

NVIDIA. Nvidia blackwell architecture technical brief, 2024a. URL https://resources.nvidia.com/en-us-blackwell-architecture/blackwell-architecture-technical-brief.

NVIDIA. Nvidia cublasdx, 2024b. URL https://docs.nvidia.com/cuda/cublasdx/.

NVIDIA. Nvshmem, 2024c. URL https://developer.nvidia.com/nvshmem.

Papadimitriou, C. H. On the complexity of integer programming. *J. ACM*, 28(4):765–768, oct 1981. ISSN 0004-5411. doi: 10.1145/322276.322287. URL https://doi.org/10.1145/322276.322287.

Rabenseifner, R. Optimization of collective reduction operations. In Bubak, M., van Albada, G. D., Sloot, P. M. A., and Dongarra, J. (eds.), *Computational Science - ICCS 2004*, pp. 1–9, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24685-5. URL https://link.springer.com/content/pdf/10.1007/978-3-540-24685-5_1.pdf.

Rajbhandari, S., Li, C., Yao, Z., Zhang, M., Aminabadi, R. Y., Awan, A. A., Rasley, J., and He, Y. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18332–18346. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

Sanseviero, O., Tunstall, L., Schmid, P., Mangrulkar, S., Belkada, Y., and Cuenca, P. Mixture of experts explained, 2023. URL https://huggingface.co/blog/moe.

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

Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., Liu, Z., Prabhumoye, S., Zerveas, G., Korthikanti, V., Zheng, E., Child, R., Aminabadi, R. Y., Bernauer, J., Song, X., Shoeybi, M., He, Y., Houston, M., Tiwary, S., and Catanzaro, B. Using deepspeed and megatron to train megatron-turing NLG 530b, A large-scale generative language model. *CoRR*, abs/2201.11990, 2022. URL https://arxiv.org/abs/2201.11990.

SnowflakeAI. Snowflake arctic: The best llm for enterprise ai – efficiently intelligent, truly open, 2024a. URL https://www.snowflake.com/blog/arctic-open-efficient-foundation-language-models-snowflake/.

SnowflakeAI. Snowflake arctic cookbook, 2024b. URL https://www.snowflake.com/en/data-cloud/arctic/cookbook/.

Tarnawski, J., Phanishayee, A., Devanur, N., Mahajan, D., and Paravecino, F. N. Efficient algorithms for device placement of dnn graph operators. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Wang, G., Qin, H., Jacobs, S. A., Holmes, C., Rajbhandari, S., Ruwase, O., Yan, F., Yang, L., and He, Y. Zero++: Extremely efficient collective communication for giant model training, 2023.

xAI. Open release of grok-1, 2024. URL https://x.ai/blog/grok-os.

Zhai, M., He, J., Ma, Z., Zong, Z., Zhang, R., and Zhai, J. SmartMoE: Efficiently training Sparsely-Activated models through combining offline and online parallelization. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pp. 961–975, Boston, MA, July 2023. USENIX Association. ISBN 978-1-939133-35-9. URL https://www.usenix.org/conference/atc23/presentation/zhai.

Zheng, L., Li, Z., Zhang, H., Zhuang, Y., Chen, Z., Huang, Y., Wang, Y., Xu, Y., Zhuo, D., Xing, E. P., Gonzalez, J. E., and Stoica, I. Alpa: Automating inter- and Intra-Operator parallelism for distributed deep learning. In *16th USENIX Symposium on Operating Systems Design and*

Implementation (OSDI 22), pp. 559–578, Carlsbad, CA, July 2022. USENIX Association. ISBN 978-1-939133-28-1. URL https://www.usenix.org/conference/osdi22/presentation/zheng-lianmin.