| Notation | Description |
|---|---|
| $H(x,\, y)$ | Heaviside Function $H(x)$, where $H(0) = y$ following [Con23] |
| $k$ | Number of experts selected during token routing, $k \in \mathbb{N}$ see *top_k* in [Sha+17; FZS22] |
| $X$ | 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 $\lvert N\rvert \cdot \lvert J_n\rvert = \lvert W\rvert > 0$ |
| $\varepsilon$ | Optimism factor, $\varepsilon \in \{0, \dots, \lvert W\rvert - 1\} \subset \mathbb{Z}^+$, $\forall w \in W\ \varepsilon_w = \varepsilon$ |
| $G^n_j$ | Worker with rank $j$ in node $n$ |
| $X^n_j$ | Set of experts hosted on $G^n_j$, $X^n_j \subseteq X$ |
| $\phi$ | Tensor mapping $G^n_j$ 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 $G^n_j$

**Data:** $Q$: a blocking task FIFO queue
**Require:** $X^n_j$, $S$, $\phi$, $\varepsilon$, $\lvert W\rvert$

**begin**
  $\xi \leftarrow \varepsilon$
  $G_\phi \leftarrow$ **GetWorkers**$(\phi,\, S)$
  $\mu \leftarrow \lvert G_\phi\rvert + H(\lvert X^n_j\rvert,\, 0) \cdot (\lvert W\rvert - 1)$ `// Upper bound for tasks`
  $flag \leftarrow$ **False**
  $W' \leftarrow S.W'$
  **if** $G^n_j \in G_\phi$ **then**
    $\mu \leftarrow \mu + 1$
    $\xi \leftarrow \xi + 1$
  **end if**
  `// Send token slices to corresponding workers`
  **broadcast**$(shouldProcess,\, \phi,\, G_\phi)$
  **while** $Q.timeout ==$ **False and** $\mu > 0$ **do**
    **if** $flag ==$ **False and** $\xi == 0$ **then**
      $Q$.**ActivateCountdown**()
      $flag \leftarrow$ **True**
    **end if**
    $t \leftarrow Q$.**take**()
    **if** $t.processed ==$ **True then**
      **PostProcessing**$(\phi,\, t)$
      $\xi \leftarrow \xi - 1$
      $\mu \leftarrow \mu - 1$
    **else if** $t.shouldProcess ==$ **True then**
      $\theta =$ **ExpertProcessing**$(t.\tau,\, X^n_j)$
      **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**
    **broadcast**$(processed,\, W')$
  **end if**
**end**

| 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 $i,\ j \in W$ |
| $\alpha_{ij}$ | $\alpha$ cost between $i$ and $j$, $\alpha \in \mathbb{R}^+$ See [Sol16] for an overview |
| $\beta_{ij}$ | $\beta$ cost between $i$ and $j$, $\beta \in \mathbb{R}^+$ |
| $G$ | Graph adjacency matrix where $G[i, j] = (\alpha_{ij},\ \beta_{ij})$ and $G = (V, E)$, $V = W$ |
| $m_j$ | Memory capacity available for experts on 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 $x$ [Nar+21] |
| $\mathcal{F}_j$ | *Actual* FLOPS of worker $j$ |
| $\mathcal{C}_j$ | Communication cost for worker $j$ |
| $T_\rho(n)$ | Time for all-reduce on $n$ processes, see [Rab04; TRG05] |

Table 2: Notation for *Lysi*

# Lysi

## Preliminaries

Let's 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 [Nar+21], $L$: num of layers, $B$: global batch, $i$: MoE layer frequency [Lie+24], $b$: mini-batch and $\omega$: effective world size $\omega \in (0, W]$. Also define $\eta$ as the frequency of communicating over an edge.

## Objective

We capture how MoE Expert Parallelism (EP) and DDP affect the end-to-end time of a single training iteration below. For inference, $T_\rho(|\mathcal{G}|) = 0$

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

Further, 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** (Algorithm 2), a novel *deterministic* graph partitioning algorithm reminiscent of Kruskal's classical MST algorithm [Wik24]. We do not yet have a tight bound, as precisely analyzing the complexity of evaluating $T(g)$ in Line 9 is non-trivial due to the communication cost $\mathcal{C}$. However, clearly $o(N^4)$ holds. We reiterate $T_g$ below, clarifying $\pi(g)$, and $\mathcal{C}$. We define $T_\rho$ per the worst case latency of [Rab04]

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

where

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

and $\forall a, b \in \mathcal{G}$, $\alpha^* = \max\{\alpha_{ab}\}$ and $\beta^* = \max\{\beta_{ab}\}$. Also $d_{ar}$ is the all-reduce buffer size.

## LysiGroup

Compared to Kruskal's, in Line 11, we only merge groups when the objective value is minimized.

**Algorithm 2:** *LysiGroup* Generates expert parallel groups

**Require:** $G = (V,\ E)$, $\mathcal{F}$, $f$, $\eta$, $d$, $d_{ar}$

**Result:** $\mathcal{G}$: Groups

1 **begin**
2   Compute weights and sort $E$ ascending
3   $\mathcal{G} \leftarrow \{V\},$ Initialize disjoint-set data structure with all nodes in $V$
4   **while** $E.size() > 0$ **do**
5     $(u, v, w) \leftarrow E.pop()$
6     $g_u \leftarrow \mathcal{G}.\text{FIND}(u)$
7     $g_v \leftarrow \mathcal{G}.\text{FIND}(v)$
8     **if** $g_u \neq g_v$ **then**
9       $T_{uv} \leftarrow \text{T}(g_u \cup g_v)$
10       **if** $[T(g_u) == \infty$ **or** $T_{uv} < T(g_u)]$ **and** $[T(g_v) == \infty$ **or** $T_{uv} < T(g_v)]$ **then**
11         $g_m \leftarrow \mathcal{G}.\text{UNION}(g_u, g_v)$
12       **end if**
13     **end if**
14   **end while**
15   **return** $\mathcal{G}$
16 **end**

## LysiAssign

Now, we present the final algorithm that allocates experts to workers within a single group $W_g \in \mathcal{G}$ generated by **LysiGroup**.

**Algorithm 3:** *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 order by FLOPS
  $B \leftarrow [\mathcal{X}]$, binary search structure ordered by compute cost of $x \in \mathcal{X}$
  **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** $\texttt{WrapAround}\ () ==$ **False then**
      $W'_g \setminus j$
    **end if**
  **end while**
  **return** $\mathcal{S}$
**end**

**Problem Statement**

Formally, LysiAssign solves an optimization problem with objective

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

subject to,

$$\text{per-worker memory constraint} \qquad \sum_{x \in \mathcal{X}} y_{xj} \geq m_j \qquad \forall j \in W_g \tag{6}$$

$$\text{allocating all experts} \qquad \sum_{g \in \mathcal{G}} \sum_{j \in W_g} y_{xj} = |\mathcal{X}| \qquad \forall j \in W_g, x \in \mathcal{X} \tag{7}$$

$$\text{allocating an expert to one worker only} \qquad \sum_{j \in W_g} y_{xj} = 1 \qquad \forall x \in \mathcal{X} \tag{8}$$

$$\text{binary decision variable} \qquad y_{xj} \in \{0, 1\} \qquad \forall j \in W_g, x \in \mathcal{X} \tag{9}$$

LysiAssign adopts a best-fit greedy approach with complexity $\mathcal{O}(|\mathcal{X}| \log |\mathcal{X}|)$.

# References


[Rab04] Rolf Rabenseifner. “Optimization of Collective Reduction Operations”. In: *Computational Science - ICCS 2004*. Ed. by Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and Jack Dongarra. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–9. ISBN: 978-3-540-24685-5. URL: https://link.springer.com/content/pdf/10.1007/978-3-540-24685-5_1.pdf.

[TRG05] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. “Optimization of Collective Communication Operations in MPICH”. In: *The International Journal of High Performance Computing Applications* 19.1 (2005), pp. 49–66. DOI: 10.1177/1094342005051521. eprint: https://doi.org/10.1177/1094342005051521. URL: https://doi.org/10.1177/1094342005051521.

[Sol16] Edgar Solomonik. *CS 598: Communication Cost Analysis of Algorithms Lecture 1: Course motivation and overview; collective communication*. 2016. URL: https://solomonik.cs.illinois.edu/teaching/cs598%5C_fall2016/lectures/lec1.pdf.

[Sha+17] Noam Shazeer et al. “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer”. In: *CoRR* abs/1701.06538 (2017). arXiv: 1701.06538. URL: http://arxiv.org/abs/1701.06538.

[Nar+21] Deepak Narayanan et al. “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. St. Louis, Missouri, Association for Computing Machinery, 2021. ISBN: 9781450384421. DOI: 10.1145/3458817.3476209. URL: https://doi.org/10.1145/3458817.3476209.

[FZS22] William Fedus, Barret Zoph, and Noam Shazeer. “Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity”. In: *Journal of Machine Learning Research* 23.120 (2022), pp. 1–39. URL: http://jmlr.org/papers/v23/21-0998.html.

[Con23] PyTorch Contributors. *Torch.heaviside*. 2023. URL: https://pytorch.org/docs/stable/generated/torch.heaviside.html.

[Lie+24] Opher Lieber et al. *Jamba: A Hybrid Transformer-Mamba Language Model*. 2024. arXiv: 2403.19887 [cs.CL].

[Wik24] Wikipedia contributors. *Kruskal’s algorithm — Wikipedia, The Free Encyclopedia*. [Online; accessed 6-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Kruskal%27s_algorithm&oldid=1221098161.