LumosCore: Highly Scalable LLM Clusters with Optical Interconnect

Anonymous Author(s)*

A APPENDIX2

A.1 OCS Reconfiguration Scheduling

Given the Integer Decomposition Theory Lemma, we can find a OCS configuration for any logical topology matrix C. Given $G_k(E_i^h)$ which means the number of links between h-th spine in i-th pod and k-th OCS in h-th OCS group in one sub phsical topology, the sub-logical topology A and u which means the number of currently used link between the h-th spine of the i-th Pod and the h-th spine of the j-th Pod through the k-th OCS in the h-th OCS group, we want to calculate the corresping OCS reconfiguration $x^h_{i,j,k}$. Suppose the number of spine in each Pod is $H = K_{leaf}/\tau$, to expedite the program's execution speed, we can compute H instances of x^h in parallel.

Algorithm 1: FindConnectionAlgorithm

Input: A_h, the OCS set in $\{o_1, o_2, \ldots, o_{K_{spine}}\}$, currently used link u

Output: spine-OCS-spine connection x_h which is for h-th spine in each Pod

`/* Step1: Return Result when there is only one OCS in current OCS set */`

$s \leftarrow$ number of OCSes in the current graph;

if $s = 1$ **then**

  return c

`/* Step 2, Merge OCSes */`

Choose any $bipartition^2$ $\{G_1^{semi(g)}, G_2^{semi(g)}\}$ of $G^{semi(g)}$;

$u^{sub}_{ij\kappa h} = \sum_{\kappa \in G_p^{semi(g)}} u_{ij\kappa h}, \forall i \in I, j \in J, p \in \{1,2\}$;

$(x_h^{semi(g)})^* =$ The optimal soltuion of $OPT(c, u^{sub})$;

`/* Step 3, Decompose sub problems */`

$u^{sub(p)}_{ij\kappa h} = u^{sub}_{ij\kappa h} \forall i \in I, j \in J, p \in \{1,2\}$;

$c^{sub(p)}_{ij\kappa h} = x^{semi(g)})^*_{ij\kappa h} \forall i \in I, j \in J, p \in \{1,2\}$;

`/* Step3: Merge the result of subproblem */`

$(x_h^{semi(g)})^p = FindConnectionAlgorithm(c^{sub(p)}, K^p, u^{sub(p)}) \forall p \in \{1,2\}$;

return $x_h^{semi(g)}$;

A.1.1 Solve the problem using MCF when there are only two OCSes for h-th spine. When there are only two OCSes in each OCS group, the problem can be solved using a MCF algorithm. For the h-th OCS group, since we have $A_{ijh} = x^{semi(g)}_{ij0h} + x^{semi(g)}_{ij1h}$, the problem can be reduced into the following problem.

$$\text{Minimize} \sum_{i,j,h} ((u_{ij0h} - x^{semi(g)}_{ij0h})^+ + (u_{ij1h} + x^{semi(g)}_{ij0h} - A_{ijh})^+) \tag{1}$$

$$\text{s.t.} \sum_{j,h} x^{semi(g)}_{ij0h} + \sum_{j,h} x^{semi(g)}_{ij1h} = \sum_j A_{ijh} \tag{2}$$

$$\sum_{i,h} x^{semi(g)}_{ij0h} + \sum_{i,h} x^{semi(g)}_{ij1h} = \sum_i A_{ijh} \tag{3}$$

$$\sum_j x^{semi(g)}_{ij0h} \le \min(G_1(E_i^h), \sum_j A_{ijh}) \tag{4}$$

$$\sum_i x^{semi(g)}_{ij0h} \le \min(G_1(E_j^h), \sum_i A_{ijh}) \tag{5}$$

This problem is equivalent to the following MCF problem $OPT(c, u)$. Given the number of Pod N and the number of spines in each Pod H, build a MCF model with $N*H$ supplies $\{s_1, s_2, ..., s_{N*H}\}$ and $N*H$ supplies $\{d_1, d_2, ..., d_{N*H}\}$. Each supply node s_i has $G_0(E_i^h)$ unit of supply, and the demand node d_j has $G_0(E_i^h)$ unit of demand. For each pair of (s_i, d_j), consider the following function

$$f_{ij}(x) = (u_{ij0h} - x^{semi(g)}_{ij0h})^+ + (u_{ij1h} + x^{semi(g)}_{ij0h} - A_{ijh})^+, x^{semi(g)}_{ij0h} \in [0, A_{ijh}] \tag{6}$$

Assume it has q noncontinuous points $\{x_1^{semi(g)}, ..., x_q^{semi(g)}\}$ and define $x_0^{semi(g)} = 0$, $x_{q+1}^{semi(g)} = A_{ijh}$, then we add $q+1$ arcs from s_j to d_j. For the p-th arc, the cost is the slope of $[x_{p-1}^{semi(g)}, x_p^{semi(g)}]$ of f_{ij}. We can solve this problem Note that in polynomial time by the Goldberg-Tarjan algorithm [**?**].

A.1.2 Solve the problem using MCF when there are more than two OCSes. When there are more than two OCSes, we can merge some OCSes into a larger OCS, so the physical topology appears to have only two OCSes. For example, when multiple OCS exist within a cluster, we solve the problem by adopting a strategy of OCS merging and subproblem splitting. We verify the feasibility of this solution method in the Appendix A.1.3.

Step 1: Merge OCS For a set of OCS $\{o_0, o_1, \ldots, o_\kappa, \ldots\}$, we randomly divide the OCS into two groups, namely $Group_0 = \{o_0, o_2, o_4, ...\}$ and $Group_1 = \{o_1, o_3, o_5, ...\}$. We denote the connections from each spine to the 0-th group of OCS by $G_p(E_i^h)^{new} = \sum_\kappa G_p(E_i^h)$ if $o_\kappa \in Group_p$. By employing this method, the logical cluster consists of only two OCSs, and we can derive the spine-OCS-spine connection $(x_{ij\kappa h}^{semi(g)})^{sup}$ using the algorithm mentioned in §A.1.1. When there are only 2 OCSes this solution addresses the problem directly. In other cases, the problem needs to be decomposed into two sub-problems based on the grouping of OCSs for further resolution.

Step 2: Split subproblem In Step 1, we divided all OCS into two groups. In Step 2, we decomposed the original problem into two sub-problems based on the grouping of OCS. Specifically, we use the topology formed by the links connected to the OCS in the p-th Group as the topology for the p-th subproblem. Then we use the input logical topology matrix $A_h^p = \sum_{ij\kappa h}(x_{ij\kappa h}^{semi(g)})^{sup}$ if $o_\kappa \in Group_p$ to solve the p-th subproblem in the same way as mentioned in Step 1.

We iteratively solve the problem by employing the OCS merge-subproblem splitting approach, with the specific solution process detailed in Algorithm 1.

A.1.3 Proof the Feasibility When There are More Than 2 OCSes. To prove the feasibility of the MCF model in §A.1.1, we need to prove that the solution obtained by combining the solutions of the two sub-problems remains feasible. We set that

$$x_{ij0h}^{semi} = \frac{|O_1^{semi(g)}|}{|O_0^{semi(g)} \cup O_1^{semi(g)}|} A_{ijh} \tag{7}$$

By substituting Equation (7) into Equation (4) we have that:

$$\sum_j x_{ij0h}^{semi(g)} = \sum_j \frac{|O_1 semi(g)|}{|O_0^{semi(g)} \cup O_1^{semi(g)}|} A_{ijh}$$

$$\leq \frac{|O_1^{semi(g)}|}{|O_0^{semi(g)} \cup O_1^{semi(g)}|} \sum_{\kappa \in O_0^{semi(g)} \cup O_1^{semi(g)}} G_\kappa(E_i^h)$$

$$= \sum_{\kappa \in O_1^{semi(g)}} G_\kappa(E_i^h)$$

$$= G_1(E_i^h)$$

what is more, we have that

$$x_{ij0h}^{semi} = \frac{|O_1^{semi(g)}|}{|O_0^{semi(g)} \cup O_1^{semi(g)}|} A_{ijh}$$

$$\leq A_{ijh}$$

$$\sum_j x_{ij0h}^{semi(g)} \leq \min(G_1(E_i^h), \sum_j A_{ijh})$$

so the Equation (4) is satisfied.

By substituting Equation (7) into Equation (5) we have that:

$$\sum_i x_{ij0h}^{semi(g)} = \sum_i \frac{|O_1 semi(g)|}{|O_0^{semi(g)} \cup O_1^{semi(g)}|} A_{ijh}$$

$$\leq \frac{|O_1^{semi(g)}|}{|O_0^{semi(g)} \cup O_1^{semi(g)}|} \sum_{\kappa \in O_0^{semi(g)} \cup O_1^{semi(g)}} G_\kappa(E_j^h)$$

$$= \sum_{\kappa \in O_1^{semi(g)}} G_\kappa(E_j^h)$$

$$= G_1(E_j^h)$$

what is more, we have that

$$x_{ij0h}^{semi} = \frac{|O_1^{semi(g)}|}{|O_0^{semi(g)} \cup O_1^{semi(g)}|} A_{ijh}$$

$$\leq A_{ijh}$$

$$\sum_i x_{ij0h}^{semi(g)} \leq \min(G_1(E_j^h), \sum_i A_{ijh})$$

so the Equation (5) is satisfied.

Clearly $x_{ij0h}^{semi} = \frac{|O_1^{semi(g)}|}{|O_0^{semi(g)} \cup O_1^{semi(g)}|} A_{ijh}$ is a feasible solution. A famous property of MCF problems is that if there exists a real number feasible solution, then there must exist an integer feasible solution, so $OPT(c, u)$ is feasible.

A.2 How to Handle AlltoAll Traffic?

AlltoAll is a common communication operator in the training process of large models. In a Clos network, an AlltoAll communication involving $N \times P$ GPUs consists of $N \times P - 1$ phases. In the t-th phase, the i-th GPU sends data to the $(i+t)$ mod $(N \times P)$-th GPU. If inter-Pod AlltoAll communication exists in LumosCore, we need to allocate $N \times P \times (N \times P - 1)$ links for $N \times P$ GPUs, which may lead to severe flow contention issues. In this section, we discuss how to handle AlltoAll traffic in LumosCore.

It is noteworthy that AlltoAllv traffic is irregular and difficult to predict. Therefore, we recommend constructing a full-mesh network through OCS reconfiguration to handle AlltoAllv traffic. With the widespread application of technologies such as expert shadowing in the industry, load balancing among different experts during training can be achieved. Consequently, in LumosCore, our primary focus is on handling AlltoAll traffic.

Given the p_0-th Pod, as well as the n_0-th GPU and the n_1-th GPU in the Pod, to avoid potential flow contention, we need to ensure that any two GPUs belongs to the same Pod do not send flow to the same Pod during the same phase. To achieve this objective, we have revised the AlltoAll communication algorithm as follows:

Given the p_0-th Pod and the n_0-th within it, to avoid potential flow contention, we must ensure that no two GPUs in the same Pod send data to the same Pod during the same phase. To achieve this objective, we have revised the AlltoAll communication algorithm as follows:

$$\forall_{p_0,n_0} Op(p_0 * N + n_0, t) = (p_0 * (N+1) + N * n_0 + t - 1)\%(N * P) \tag{8}$$

Suppose there exists p_0, n_0, p_1, n_1, where $p_0 * N + n_0 \neq p_1 * N + n_1$, set if there exists flow contention between Pods, which means:

$$Op(G(p_0 * N + n_0), t) = Op(G(p_1 * N + n_1), t)$$

$$(p_0 * (N+1) + N * n_0 + t - 1)\%(N * P) = (p_0 * (N+1) + N * n_1 + t - 1)\%(N * P)$$

$$((N*(p_0+n_0)+p_0+t)\%(N*P) = ((N*(p_1+n_1)+p_1+t)\%(N*P)$$

Obviously $(N * (p_0 + n_0) + p_0 + t \neq N * (p_1 + n_1) + p_1 + t$, we might as well assume that $N * (p_1 + n_1) + p_1 + t > N * (p_0 + n_0) + p_0 + t$, then we have that

$$(p_0 + n_0) + \frac{p_0}{N} = (p_1 + n_1) + \frac{p_1}{N} - N$$

so we must have that $\frac{p_0}{N} = \frac{p_1}{N}$ and $p_0 = p_1$, so

$$n_0 = n_1 - N \tag{9}$$

but Equation 9 is not possible since $n_0 \leq N-1, n_1 \leq N-1$, so contention can be eliminated.