Estimating Allocatable Neural Accelerators

Business Requirement

With the coming of the LLM age, cloud providers have started providing ML training infrastructure as a service. Huawei Cloud enables the customers to run their ML training jobs on servers equipped with Ascend NPU accelerators of its own production. The servers are divided into clusters, and incoming training jobs are sent to one of the clusters based on hardware requirements and user preferences. In what follows we limit ourselves to homogeneous server clusters.

Due to workload imbalance and sub-optimal placement decisions, it often happens that in some clusters the jobs are stuck in the queue, violating the waiting time SLA, while in the other clusters the NPUs mostly stay idle. The problem of estimating allocatable neural accelerators involves, given a historical snapshot of idle servers and SLA-violating tasks, determining the upper bound on the utilization of the idle NPUs obtainable by SLA-compliant execution of some of the violating tasks.

Problem Statement

Input and Constraints

- There are *K* clusters with N_k nodes on each cluster, *k*, and each node has 8 cards but not all cards are available all the time (see below)
- We are given a set of tasks. Each task is a tuple (r_j, p_j, c_j, n_j), where r_j denotes the release time, p_j denotes the processing time, c_j denotes the number of NPU cards allocated by the task on one node, and n_j denotes the number of nodes (servers) used by the task.
- If a task, *j*, is allowed for execution, it must start executing at time $t \in [r_j, r_j + w]$, where *w* is the maximum allowed waiting time, typically 5 or 10 minutes, and executed without interruption for p_j seconds. Not all tasks have to be executed, and there is no penalty for not executing the tasks.
- During the execution c_j cards are used on each of the n_j different nodes selected by the scheduler.

- While multiple tasks can share a node, NPU cannot be shared. All nodes used by a task must belong to the same cluster.
- The input includes a description of resources available in each cluster at every timeslot. The resources of a cluster are not static: nodes are added and removed over time. The description is a sequence of historical events (t, Δ) meaning that at time t either Δ empty nodes are added (if Δ > 0) or −Δ empty nodes must be removed from the cluster (if Δ < 0). If node n is added at time l_n and removed at time r_n , all tasks executed at that node must start at time $\geq l_n$ and finish at time $\leq r_n$. Due to some technical limitations, it is not specified which of the previously added nodes are removed; any mapping respecting the given events is possible, and the algorithm for computing upper bounds must optimize over all valid mappings.

Objective Function

- We are interested in finding upper bounds on Maximum Utilization. Utilization is computed as follows. Firstly we compute total NPU capacity *C* over all clusters. If a node *n* is included to the resource pool from time l_n to time r_n, its capacity is $8(r_n - l_n)$, and the total capacity is the sum of individual node capacities. Then we compute total NPU usage *U* by taking the sum of $p_j c_j n_j$ over all executed tasks on all clusters. Finally, utilization is computed as U/C 100%. We consider two scenarios:
- Scenario 1. All tasks use full nodes, that is, $\forall j,\ c_j = 8$.
- Scenario 2. Some tasks can use less than 8 cards, but it is guaranteed that is always a power of two.

Input Dataset

The resource and task income traces are very specific and shall be prepared, anonymized and provided to the contestants. For each trace we additionally provide the utilization of the best scheduling algorithm on that trace. The absolute gap between the upper bound and the provided result, averaged over all traces, will be used for comparing solutions.

The granularity are such as the number of nodes per cluster, N_k, is measured by thousands, while the number of clusters, *K*, is about 10

Call for Solutions

Within the scope of the challenge, we seek solutions that find the bound with average gap as small as possible and work in reasonable time (< 5 hours).

As for the deliverables, we expect a description of the algorithm and code with publicly available dependencies (so, please avoid commercial solvers).