Supplementary Materials - Resource Allocation with Service Affinity in Large-Scale Cloud Environments

1 RASA problem is NP-hard

Proof. Given that the Bin-Packing problem is an NP-hard problem [2], we only need to prove that the Bin-Packing problem can be polynomially reduced to the RASA problem.

Let us consider an input $X = \{I, B, K\}$ for the Bin-Packing problem, where I is a set of items with $|I| = N$, and the i-th item requires resources of size x_i. The bin's size is B, and there are K bins.

We construct a new input X' for the RASA problem as follows:

1. We consider $N + 1$ services in X'. The first N services require resources $x_1, x_2, \ldots, x_N$, respectively, and each service has only one container. The $N + 1$-th service requires $B + 1$ resources and has K containers.

2. We have K machines, each with a total capacity of $2B + 1$.

3. The affinity graph $G =< V, E >$ consists of $N + 1$ vertices representing the $N + 1$ services in V. The edge set E contains N edges, specifically $\{(x_1, x_{N+1}), (x_2, x_{N+1}), \ldots, (x_N, x_{N+1})\}$, where each edge has a weight of 1.

We will now prove that the items in I can be packed into K bins with capacity B if and only if the gained affinity of the solution to the RASA problem with input X' equals 1.

First, if the items in I can be packed into K bins with capacity B, we can use this deployment to construct a feasible solution for the RASA problem with input X'. We deploy the first N services in X' using the same deployment as the items in I. Since each machine still has at least $B + 1$ free resources, we deploy each container of service $N + 1$ to a different machine. This solution is feasible, and the gained affinity is exactly 1.0.

Second, if the gained affinity of the solution to the RASA problem with input X' equals 1, then based on the structure of the affinity graph, all containers of the $N + 1$ services in X' must be deployed. Since the size of containers of service $N + 1$ is $B + 1$, and the total capacity of each machine is $2B + 1$, each machine must host exactly one container of service $N+1$. We can simply use the deployment for the first N services in the RASA problem as the deployment for the N items in the Bin-Packing problem with input X.

In summary, the Bin-Packing problem can be polynomially reduced to the RASA problem, and solving the RASA problem can produce a solution for the Bin-Packing problem. Thus, the RASA problem is NP-hard, since the Bin-Packing problem itself is NP-hard.

2 Gained Affinity implies Max Localized Traffic Under Traffic Load Balancing

Recalling the definition of *the gained affinity*:

Definition 1 (The Gained Affinity)**.** *Consider a machine $m \in \mathcal{M}$. The affinity between services s and s' is $w_{s,s'}$ and the number of containers that the two services scheduled on machine m is $x_{s,m}$ and $x_{s',m}$ respectively, then the gained affinity of services s and s' on machine m is*

$$a_{s,s',m} = w_{s,s'} \cdot \min\left\{\frac{x_{s,m}}{d_s}, \frac{x_{s',m}}{d_{s'}}\right\}. \tag{1}$$

The overall gained affinity is the sum of all $a_{s,s',m}$, $\forall\left(s,s'\right) \in E$ and $m \in \mathcal{M}$, where $\mathcal{M}$ is the set of all machines.

Some readers may calculate that if each container uniformly distributes its traffic to the containers of the target services, the affinity would be $w_{s,s'} \cdot \frac{x_{s,m}}{d_s} \cdot \frac{x_{s',m}}{d_{s'}}$. While this is a straightforward strategy that ensures traffic load balancing, it is possible to leverage a more advanced traffic distribution algorithm can achieve a higher affinity than this baseline value.

We will demonstrate that the above definition of gained affinity between two services implies the maximum localized traffic under load balancing. For services s and s', the total traffic between these two services is $w_{s,s'}$. Accordingly, when traffic is load balanced, a container of service s will send $\frac{w_{s,s'}}{d_s}$ traffic to each containers of service s', while a container service s' will receive $\frac{w_{s,s'}}{d_{s'}}$ traffic from each containers of service s. Considering a machine m, let x_s and $x_{s'}$ denote the number of containers placed for each service on machine m. Under load balancing, the traffic sent from containers of service s on machine m to service s' is $w_{s,s'} \cdot \frac{x_s}{d_s}$, and the traffic received by containers of service s' on machine m from service s is $w_{s,s'} \cdot \frac{x_{s'}}{d_{s'}}$. Therefore, the maximum traffic that can be localized within machine m, with service s sending traffic to service s', is given by $\min\left\{w_{s,s'} \cdot \frac{x_s}{d_s}, w_{s,s'} \cdot \frac{x_{s'}}{d_{s'}}\right\} = w_{s,s'} \cdot \min\left\{\frac{x_{s,m}}{d_s}, \frac{x_{s',m}}{d_{s'}}\right\}$. The above traffic distribution ratio can be achieved by designing a non-uniform distribution algorithm. This requires prioritizing traffic allocation to containers on the same machine, as much as possible, within the maximum traffic capacity that the target service on the same machine can handle.

3 Ommited Proof - Lemma 4.1

Recalling the assumption and the lemma in the manuscript:

Assumption 3.1. *The total affinity of the s^{th} service satisfies $T(s) \propto \frac{1}{s^\beta}$ for some constant $\beta > 1$, for all $s = 1, \cdots, N$.*

Lemma 1. *Under Assumption 3.1 with a power of $\beta > 1$, for any $\epsilon \in (0, 1]$, let $\gamma = (\beta - 1)(1 - \epsilon)$. Then the total affinity of the last $N - O\left(\ln^{1-\epsilon} N\right)$ services is bounded by $O\left(\frac{1}{\ln^\gamma N}\right)$.*

Proof. With Assumption 4.1, for some constants $\beta > 1$ and c, we have

$$T(s) = \frac{c}{s^\beta}$$

Then, let L be the total affinity of the last $N - O\left(\ln^{1-\epsilon} N\right)$, we have

$$\begin{aligned}
L \leq & \sum_{s=\ln^{1-\epsilon} N}^{N} T(s) \\
= & \sum_{s=\ln^{1-\epsilon} N}^{N} T\left(\ln^{1-\epsilon} N\right) \cdot \frac{\left(\ln^{1-\epsilon} N\right)^\beta}{s^\beta} \\
= & T\left(\ln^{1-\epsilon} N\right) \cdot \ln^{\beta \cdot (1-\epsilon)} N \cdot \sum_{s=\ln^{1-\epsilon} N}^{N} \frac{1}{s^\beta} \\
\leq & O(1) \cdot \int_{\ln^{1-\epsilon} N}^{N} \frac{1}{x^\beta} dx \\
\leq & O(1) \cdot \left\{ \left(\ln^{1-\epsilon} N\right)^{1-\beta} - N^{1-\beta} \right\} \\
\leq & O\left(\ln^{(1-\beta)(1-\epsilon)} N\right)
\end{aligned}$$

Let $\gamma = (\beta - 1)(1 - \epsilon)$, then $L = O\left(\frac{1}{\ln^\gamma N}\right)$. □

4 Ommited Formulations in Column Generation Algorithm

In our manuscript, the formal formulations of column generation algorithm is omitted. Here, we present the formal formulation of pattern generation and cutting stock formulation of RASA problem.

4.1 Pattern Generation Formulation

A pattern $p \in \mathbb{N}^N$ is a combination of service containers, which represents a feasible placement of containers on a machine (i.e., p_s is the number of containers that service s places on the machine). The formulation below generates one possible pattern of a machine m:

$$
\begin{aligned}
\text{max.} \quad & \sum_{s \in \mathcal{S}} \hat{\pi}_s \cdot p_s + \sum_{(s,s') \in E} a_{s,s'} && && (2) \\
\text{s.t.} \quad & w_{s,s'} \cdot \frac{p_s}{d_s} \leq a_{s,s'}, && \forall \left(s, s'\right) \in E && (3) \\
& w_{s,s'} \cdot \frac{p_{s'}}{d_{s'}} \leq a_{s,s'}, && \forall \left(s, s'\right) \in E && (4) \\
& \sum_{s \in \mathcal{S}} p_s \cdot R^S_{r,s} \leq R^M_{r,m}, && \forall r \in \mathcal{R} && (5) \\
& \sum_{s \in A_k} p_s \leq h_k, && \forall A_k \in \mathcal{A} && (6) \\
& 0 \leq p_s \leq b_{s,m} \cdot d_s, && \forall s \in \mathcal{S} && (7) \\
& p_s \in \mathbb{N}, && \forall s \in \mathcal{S}. && (8)
\end{aligned}
$$

Here, p_s's and $a_{s,s'}$'s are the decision variables, $\hat{\pi}_s$'s are the optimal solution of the dual variables for constraints in the cutting stock formulation.

4.2 Cutting Stock Formulation

Given a pattern set $\mathcal{P}_m$ of machine m, which consists of feasible patterns on machine m. Let $p^{(l)} = \left[p^{(l)}_1, p^{(l)}_2, \ldots, p^{(l)}_N\right] \in \mathcal{P}_m$ be the l^{th} pattern of machine m, and $\sigma_{m,l}$ be the gained affinity of the pattern $p^{(l)}$

The formal cutting stock formulation of RASA is

$$
\begin{aligned}
\text{max.} \quad & \sum_{m \in \mathcal{M}} \sum_{p^{(l)} \in \mathcal{P}_m} \sigma_{m,l} \cdot y_{m,l} && && (9) \\
\text{s.t.} \quad & \sum_{m \in \mathcal{M}} \sum_{p^{(l)} \in \mathcal{P}_m} p^{(l)}_s \cdot y_{m,l} \leq d_s, && \forall s \in \mathcal{S} && (10) \\
& \sum_{p^{(l)} \in \mathcal{P}_m} y_{m,l} \leq 1, && \forall m \in \mathcal{M}. && (11) \\
& y_{m,l} \in \{0, 1\}, && \forall m \in \mathcal{M} && (12)
\end{aligned}
$$

In our practical application, to speed up the computation, we relax the integer constraint of $y_{m,l}$ for all $m \in \mathcal{M}$, which corresponds to removing expression (12). This modification allows for faster computation of the cutting stock formulation, but it results in fractional solutions for container placements. To address this, we apply a rounding algorithm to the fractional solution y, although this may introduce some loss of optimality in the final integer solution.

4.3 Complexity Analysis

The conclusion is that the worst-case complexity of the column generation algorithm for RASA is $O(e^N)$, where N represents the input size. The maximum number of iterations for column generation is bounded by $O(e^N)$ due to the exponential number of patterns. In each iteration, the algorithm solves M linear programming problems (corresponding to expressions (2) to (8)) and one integer programming problem (corresponding to expressions (9) to (12)), where M is the number of machines. The time complexity for solving M linear programming problems is polynomial in N [3], while the complexity for solving one integer programming problem is $O(e^N)$ [1]. Therefore, the time complexity for each iteration is $O(e^N)$. Combining all these factors, the overall time complexity of the column generation algorithm is $O(e^N) \cdot O(e^N) = O(e^N)$. This represents the worst-case complexity analysis for the column generation algorithm.

However, in practice, when applying column generation algorithms, we typically impose a time limit that allows for only a fixed number of iterations, resulting in an approximate solution. Therefore, this complexity analysis has no reference value in practical usage.

5 More on Service Partitioning Evaluations.

In this section, we discuss the issues of the loss of optimality and the time overhead associated with service partitioning.

Among the four types of partitioning, only master-partition and equal-partition have the potential to compromise optimality. No-affinity-partition separates services that lack affinity with other services, thus not contributing any gained affinity to the solution. Compatibility-partition, on the other hand, separates services that cannot be deployed together due to their incompatibility. Hence, partitioning these services does not affect solution. Consequently, we solely focus on analyzing the effects of master-partition and equal-partition.

Fig. 1 illustrates the affinity loss due to master-partition and equal-partition. This loss represents the total affinity between different service sets caused by the two partitioning strategies. It serves as an upper bound on the potential loss in optimality caused by partitioning. Fig. 1 shows that the partitioning-induced loss remains relatively low, with a maximum loss of up to 12%.

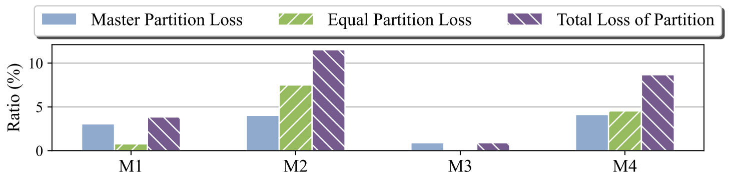

Figure 1: The loss of total affinity of partition algorithm.

Fig. 2 illustrates the time overhead of the partitioning algorithm, where the proportion of time spent on partitioning relative to the total time of our entire algorithm is used as the metric. It can be observed that, except for cluster $M1$, the partitioning algorithm accounts for approximately 8% of the total algorithm time. However, for the other three clusters, the partitioning algorithm takes less than 3% of the total algorithm time. These values are considered negligible.

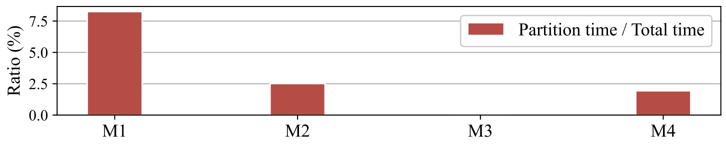

Figure 2: The time overhead of partition algorithm.

References

[1] Amitabh Basu et al. "Complexity of branch-and-bound and cutting planes in mixed-integer optimization - II". In: *Comb.* 42.6 (2022), pp. 971–996.

[2] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[3] Narendra Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Comb.* 4.4 (1984), pp. 373–396.