Let's Stay Together: Towards Traffic Aware Virtual Machine Placement in Data Centers

Manar Alqarni
Proff. Tang

Topics covered

- ◇ INTRODUCTION
- ◇ HOMOGENEOUS CASE
  - -Algorithm 1 Recursive-based Placement RBP (m, c, n, r)
  - -Algorithm 2 construction(m, c, n, r)
- ◇ HETEROGENEOUS CASE

-Algorithm 3 Sorting-based Placement SBP (m, c, n, R)

- ◇ Host-Network Joint Optimization.
- ◇ Future works.

INTRODUCTION

◇A good placement will lead to better resource utilization and less cost.

◇A slot is used to represent one basic resource unit, each slot can host one VM. This process is well known as virtual machine placement (VMP).

◇The result shows that servers/physical machines (PMs) consume near 45% overall cost, and network occupies about 15%.

PM-cost and N-cost

- PM-cost: is proportional to the number of running PMs
- N-cost: is mainly determined by inter- PM traffic.

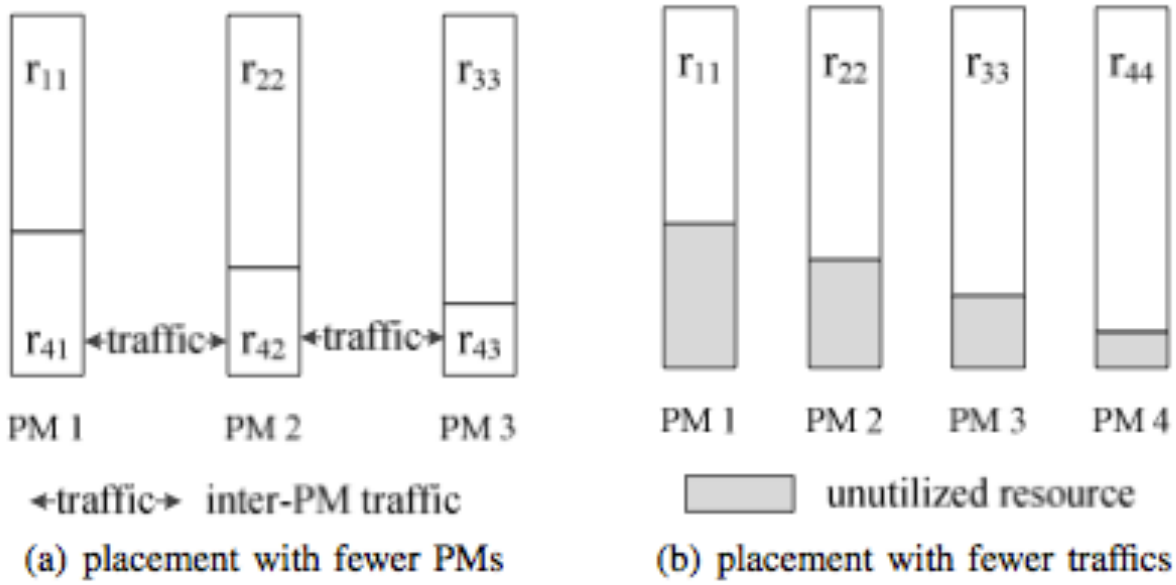


Fig. 1. Two placements for the same requests (r_1, r_2, r_3, r_4). r_{ij} indicates the VMs placed on PM j of request r_i. (a) Fewer PMs are occupied, but there exists more inter-PM traffic. (b) More PMs, but no inter-PM traffic.

Problem Description

- They used a slot to represent one resource unit (CPU/ memory/disk), and each slot can host one VM.
- They also considered the scenario where a cloud data center consists of uniform PMs, and tenants submit their resource demands, in terms of the number of slots (VMs), to the cloud.
- The cloud allocates the required resource units to tenants.

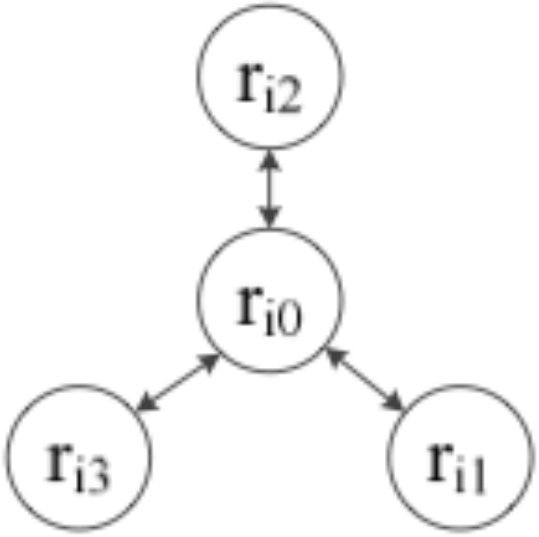
Problem Description

- For each request, it is preferable to place all the required VMs on the same PM (*perfect placement*), since there may be communication between VMs.

Cost Function

- **Centralized Model Cost Function (CCF):** $\phi_i^{(1)} = K_i.$

For each request, N-cost equals the number of pieces.

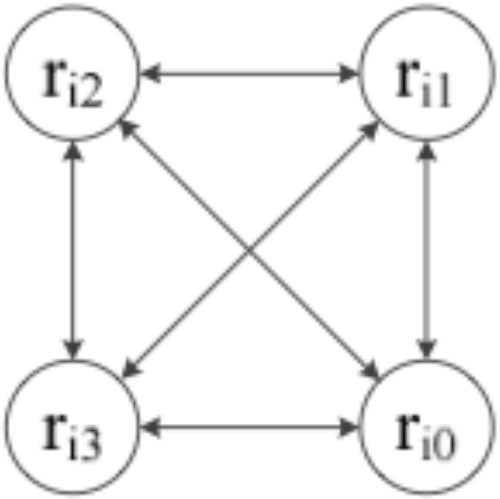


(a) Centralized Model

Cost Function

- **Distributed Model Cost Function (DCF):** $\phi_i^{(2)} = K_i^2.$

There exists a traffic link between every two pieces, which means that every two team members communicate with each other



(b) Distributed Model

Cost Function

- **Enhanced Distributed Model Cost Function (E-DCF):**

This function shares the same communication model with DCF, but the size of each piece is taken into account. In this function, the granularity of inter-PM link is VM-to- VM communication, while the previous two can be treated as “piece”-to-“piece” communication. For consistency, we let

$$\phi_i^{(3)} = 1, \text{ when } K_i = 1.$$

Homogeneous case and Heterogeneous case

- In homogeneous case, the tenants request the same amount of VMs,
- While the required number of VMs is different in the Heterogeneous case.
- **Homogeneous case:**

-Algorithm 1 Recursive-based Placement RBP (m, c, n, r)

-Algorithm 2 construction(m, c, n, r)

- **Heterogeneous case:**

-Algorithm 3 Sorting-based Placement SBP (m, c, n, R)

Algorithm 1 Recursive-based Placement RBP

- The basic idea is to achieve as many perfect placements as possible, then to split the unplaced requests into pieces.
- The process can be executed recursively. Obviously, if the number of requests is sufficiently small, say $n \leq \alpha \cdot m$ ($\alpha = c/r$), then there exists some perfect placement of VMs achieving zero additional N-cost.

Algorithm 1 Recursive-based Placement RBP

Algorithm 1 Recursive-based Placement $RBP(m, c, n, r)$

Input: m: number of PMs; c: capacity of PMs; n: number of requests; r: number of required VMs.

1: $\alpha \leftarrow c/r$, $u \leftarrow c \mod r$;
2: **if** $\alpha \cdot m \geq n$ **then**
3: perfect placement;
4: **else**
5: the first $\alpha \cdot m$ requests are placed perfectly;
6: $\beta \leftarrow r/u$, $v \leftarrow r \mod u$;
7: **for** $j = 0 \rightarrow \beta \cdot (n - \alpha \cdot m) - 1$ **do**
8: $r_{ij} = u$, $(i = \alpha \cdot m + (j \mod \beta))$;
9: $RBP(m - \beta \cdot (n - \alpha \cdot m), u, n - \alpha \cdot m, v)$;

Algorithm 1 Recursive-based Placement RBP

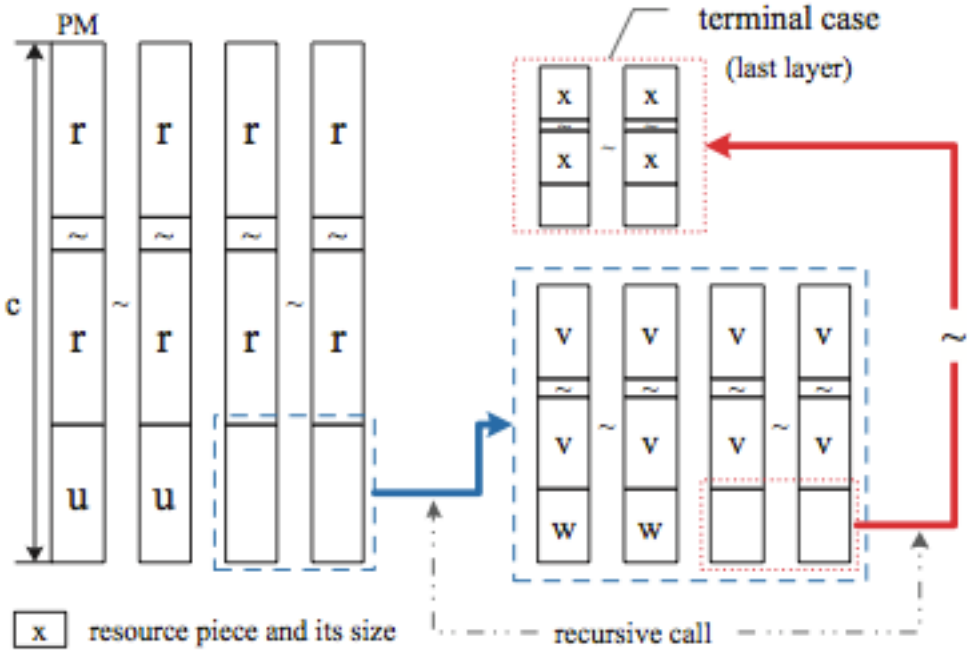


Fig. 3. Solution Structure. For each PM, the capacity is c, and each user requires r VMs. Layer 0 is shown in the left part, excluding the blue dashed rectangle. The upper red dashed rectangle is the last layer of the recursion, also the terminal case.

Algorithm 2 construction

- Algorithm 2 is based on the previous algorithm.

- For each request that has more than 2 pieces, we first find its TPC r_{ijK_i}, where we use j_K ($1 \le \kappa \le K_i$) to indicate the index of the PM that contains the κ_{th} piece of r_i. To decrease the number of pieces of r_i, we do swap(r_{ij_k} , $s_{j_{k-1}}$), wheres is the piece with size equal to r on PMj $_{k-1}$

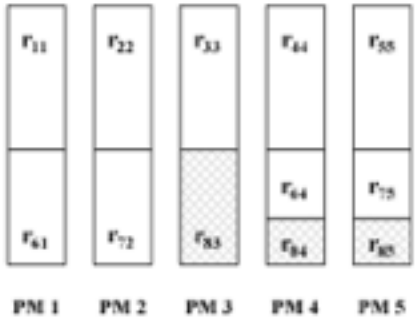
Algorithm 2 construction

Algorithm 2 $construction(m, c, n, r)$

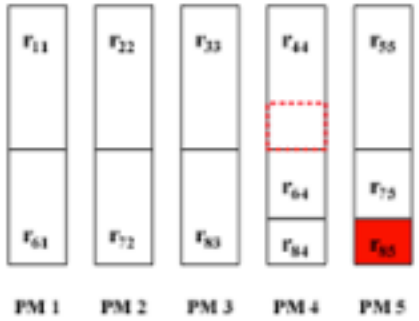
Input: m: number of PMs; c: capacity of PMs; n: number of requests; r: number of required VMs.

1: $RBP(m, c, n, r)$;
2: **for** $i = 0 \rightarrow n - 1$ **do**
3: **if** $K_i > 2$ **then**
4: **for** $\kappa = K_i \rightarrow 2$ **do**
5: $swap(r_{ij_\kappa}, s_{j_{\kappa-1}})$;

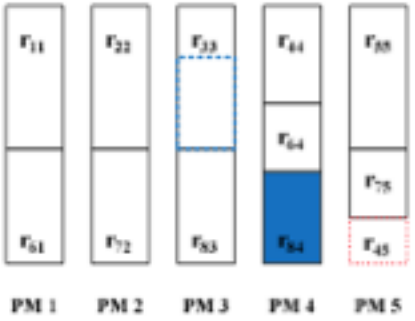
Algorithm 2 construction



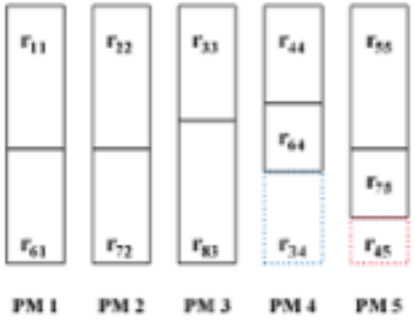
(a) Placement given by Algorithm 1. There are 2 layers, and $K_8 = 3$.



(b) The TPC of r_8 is located (red rectangle), and s_4 (red dashed rectangle) is selected.



(c) Do $swap(r_{85}, s_4)$, then we have $K_3 = 2, K_8 = 2$. s_3 (blue dashed rectangle) is selected.



(d) Do $swap(r_{84}, s_3)$. We achieve the final optimal placement.

Fig. 4. An example of homogeneous case, where $m = 5, c = 8, n = 8, r = 5$. Placement in (a) is optimal for the CCF, but not for DCF, since $K_8 = 3$. After two *swap* operations, we achieve the final optimal placement for DCF.

Algorithm 3 Sorting-based Placement SBP

- The number of required VMs of tenants are different.
- The basic idea is to place the requests with more required VMs first, since the requests with more VMs may lead to higher costs if they are split.
- We first order the requests in descending order of the number of required VMs.
- The process is that placing each request in first-fit manner if there exist PMs with sufficient slots to host the required VMs.
- Otherwise, selecting the PM with the most available slots to host as many required VMs as possible, the part (piece) that cannot be placed on the current PM is reinserted to the remaining unplaced requests set, while preserving the descending order.

Algorithm 3 Sorting-based Placement SBP

Algorithm 3 Sorting-based Placement $SBP(m, c, n, R)$

Input: m: number of PMs; c: capacity of PMs; n: number of requests; R: the set of requests

1: $sort(R)$;
2: **while** R is not empty **do**
3: **if** the first item (r_0) in R can be placed perfectly **then**
4: place r_0 in first fit manner;
5: **else**
6: split r_0 to two *pieces*, such that one *piece* can be placed the PM with the most available *slots* completely; then, insert another *piece* to the remaining set R, while preserving the order;
7: remove r_0 from R;

Algorithm 3 Sorting-based Placement SBP

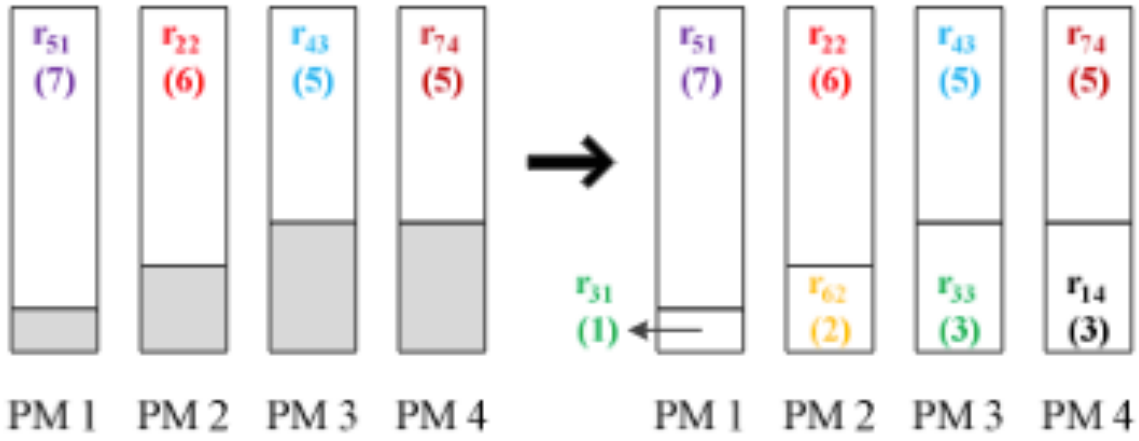


Fig. 5. An example of SBP algorithm. The input is: $r_1 = 3, r_2 = 6, r_3 = 4, r_4 = 5, r_5 = 7, r_6 = 2, r_7 = 5$. Each request is indicated with different colors. For each notation in the blank, the upper symbol r_{ij} refers to the piece placed on PM j of request r_i, the lower number in the rectangle is the value of r_{ij}, i.e. the size of this piece.

Host-Network Joint Optimization.

Host-Network Joint Optimization

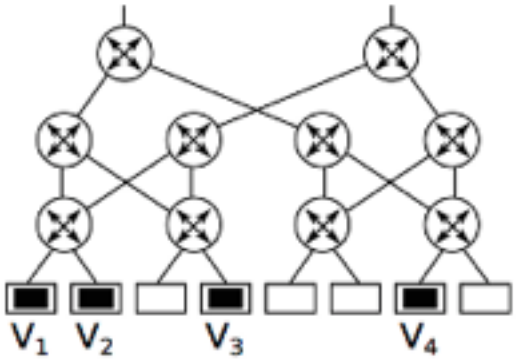
- Optimization for energy efficiency can be:

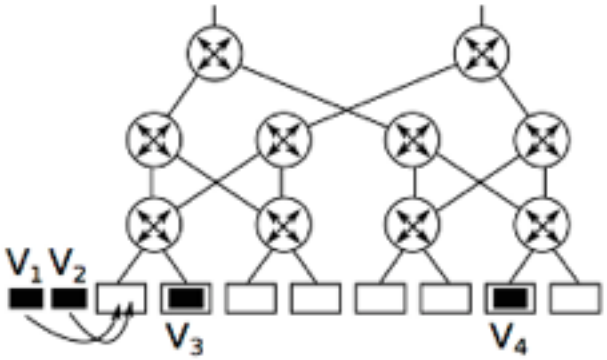
1. A host-side optimization aims to move VMs to a smaller set of servers,.
2. a network-side optimization tries to identify a smaller set of devices that are sufficient to handle current traffic flow.

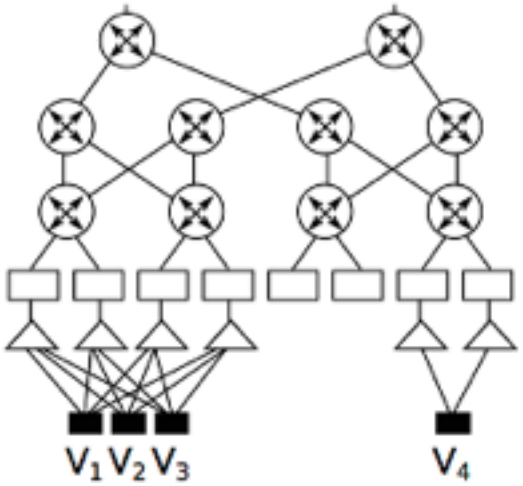
Host-Network Joint Optimization

- The key idea of the host-network joint optimization is based on the observation that the way a VM chooses which server to migrate to is similar to the way a server chooses which switch to send traffic to.
- When planning the routing paths, we can extract VMs from their current hosts and add to those VMs links to the servers that they may migrate to.
- Later, if a server is on an optimized path that connects a VM, the VM can migrate to that server.

Host-Network Joint Optimization







Summary

- INTRODUCTION
- HOMOGENEOUS CASE
  - -Algorithm 1 Recursive-based Placement RBP (m, c, n, r)
  - -Algorithm 2 construction(m, c, n, r)
  - -Algorithm 3 Sorting-based Placement SBP (m, c, n, R)
- Host-Network Joint Optimization.
- Future works.