A survey on

Power-Efficient Virtual Machine Replication in Data Centers

and

Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

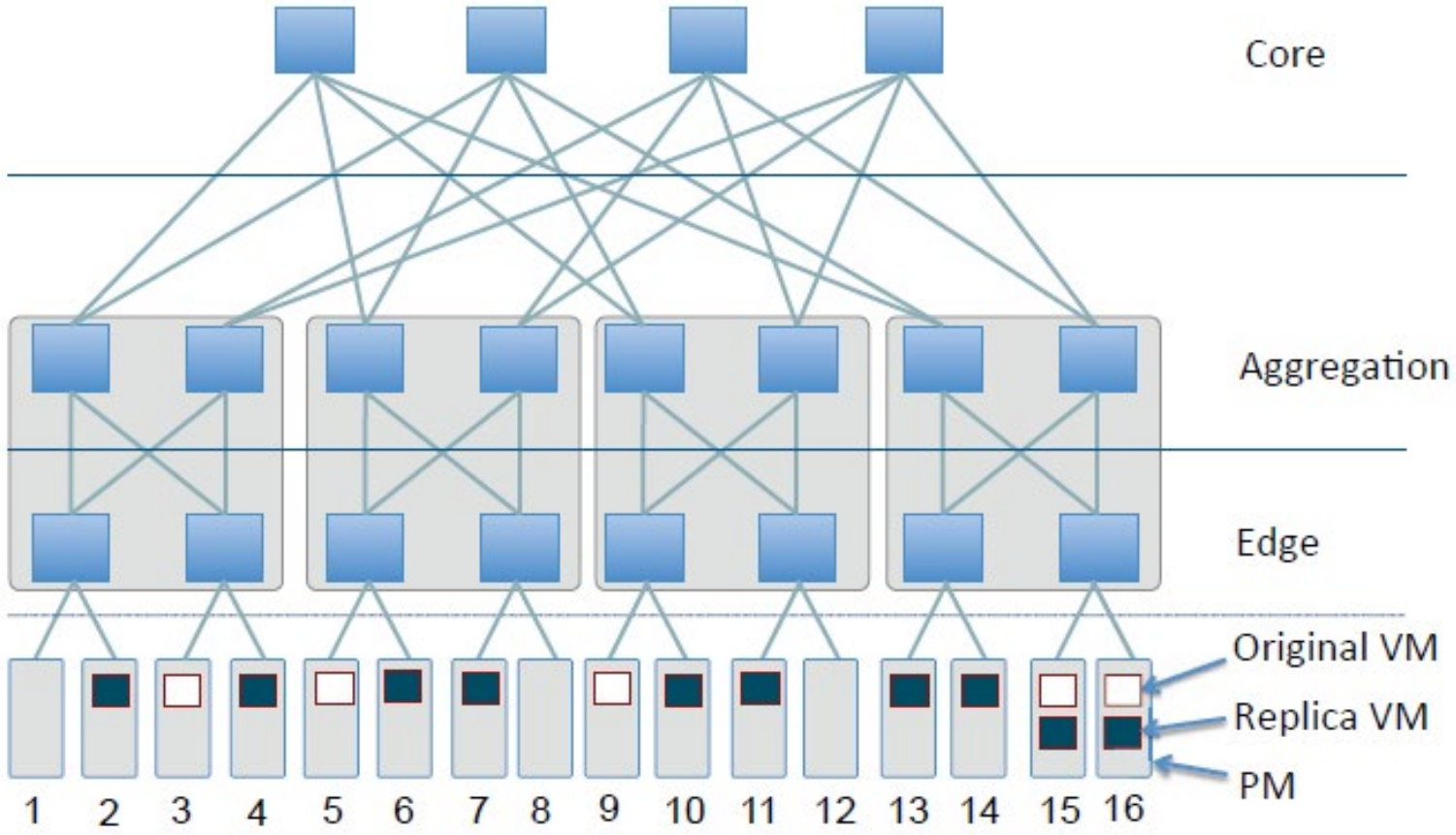
papers

PRESENTED BY: PAYMAN KHANI

Overview

- Quick study of Power-Efficient Virtual Machine Replication in Data Centers - MCF
- Study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization - OPVMP
- Reviewing their differences
- Possible ways to combine them

Quick study of Power-Efficient Virtual Machine Replication in Data Centers - MCF



Quick study of Power-Efficient Virtual Machine Replication in Data Centers

- Studies the **virtual machine replication problem** (VMR) in data centers, with the goal of **minimizing the total power consumption** in this process.
- To **guarantee that each VM is available in the event of server failure**, it replicates R copies of each VM and place them into different physical machines (PMs) in the data centers, where R depends upon the server failure probability.

Problem Model and formulation:

Let r_e, r_a, and r_c denote the power consumption in the edge, aggregate, or core switches by relaying one VM. Let $c_{i,j}$ denote the minimum power consumption of transmitting one VM from one PM $i \in V_s$ to another PM $j \in V_s$, and assume along the way there are x edge switches, y aggregate switches, and z core switches. Therefore, $c_{i,j} = x * r_e + y * r_a + z * r_c$.

We set $r_e = 10$, $r_a = 5$, and $r_c = 1$

Quick study of Power-Efficient Virtual Machine Replication in Data Centers

Problem Model and formulation (continued):

- We calculate Network Power cost to transmit the VM copies to different PMs (VMs replica placement cost) from their original VMs. VMR is to find a replication function r to:

minimize the *total replication cost*, which is the sum of the minimum power consumption of transmitting all the $R-1$ replica copies (excluding the original copy) of all the p VMs:

$$\sum_{j=1}^{p}\sum_{k=2}^{R} s_j \times c_{S(j),r(j,k)},$$

under the storage capacity constraint of each PM:

$$|\{1 \le j \le p | r(j,k) = i\}| \le m_i^e, \forall i \in V_s, 1 \le k \le R,$$

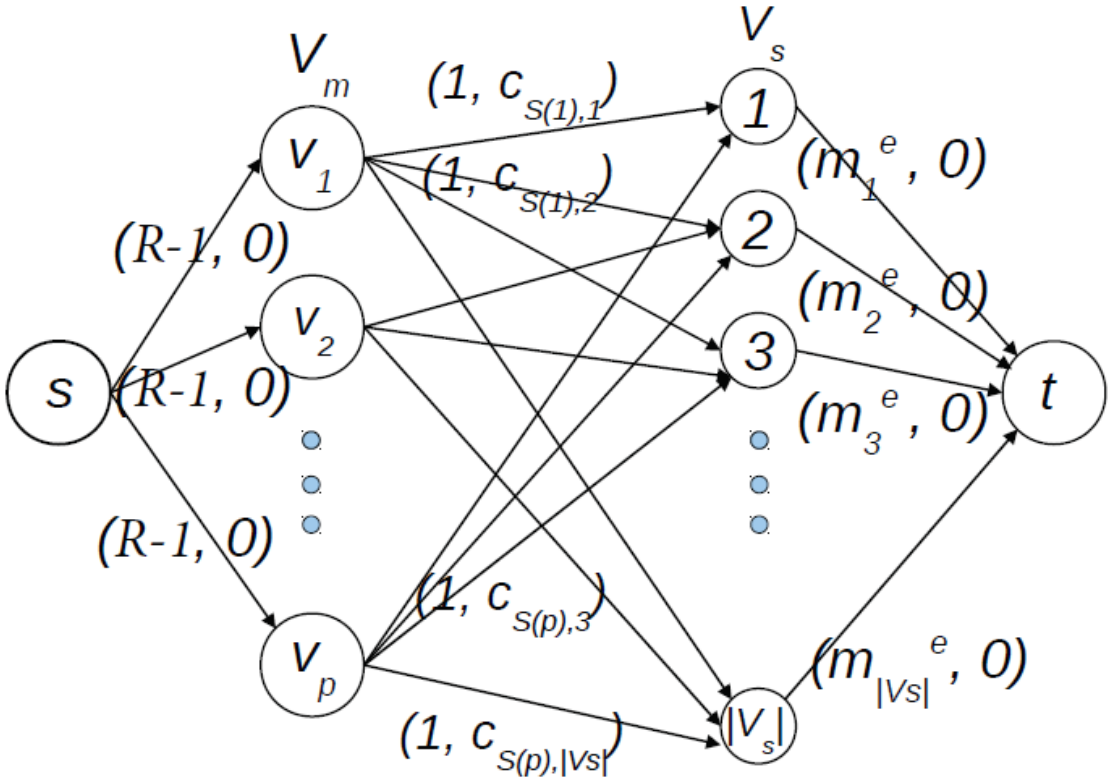
and the replication constraint of VMs:

$$r(j,k) \ne r(j,k'), 1 \le j \le p, k \ne k'.$$

Quick study of Power-Efficient Virtual Machine Replication in Data Centers

Algorithmic Solutions of VMR (**Minimum Cost Flow (MCF) Solution**):

- We show that VMR is equivalent to the minimum cost flow problem, which can be solved efficiently and optimally.
- Using MCF we are to find the minimum energy cost sending one VM replica from its source PM S(i) to its destination PM j.



Quick study of Power-Efficient Virtual Machine Replication in Data Centers

Server Consolidation Algorithm:

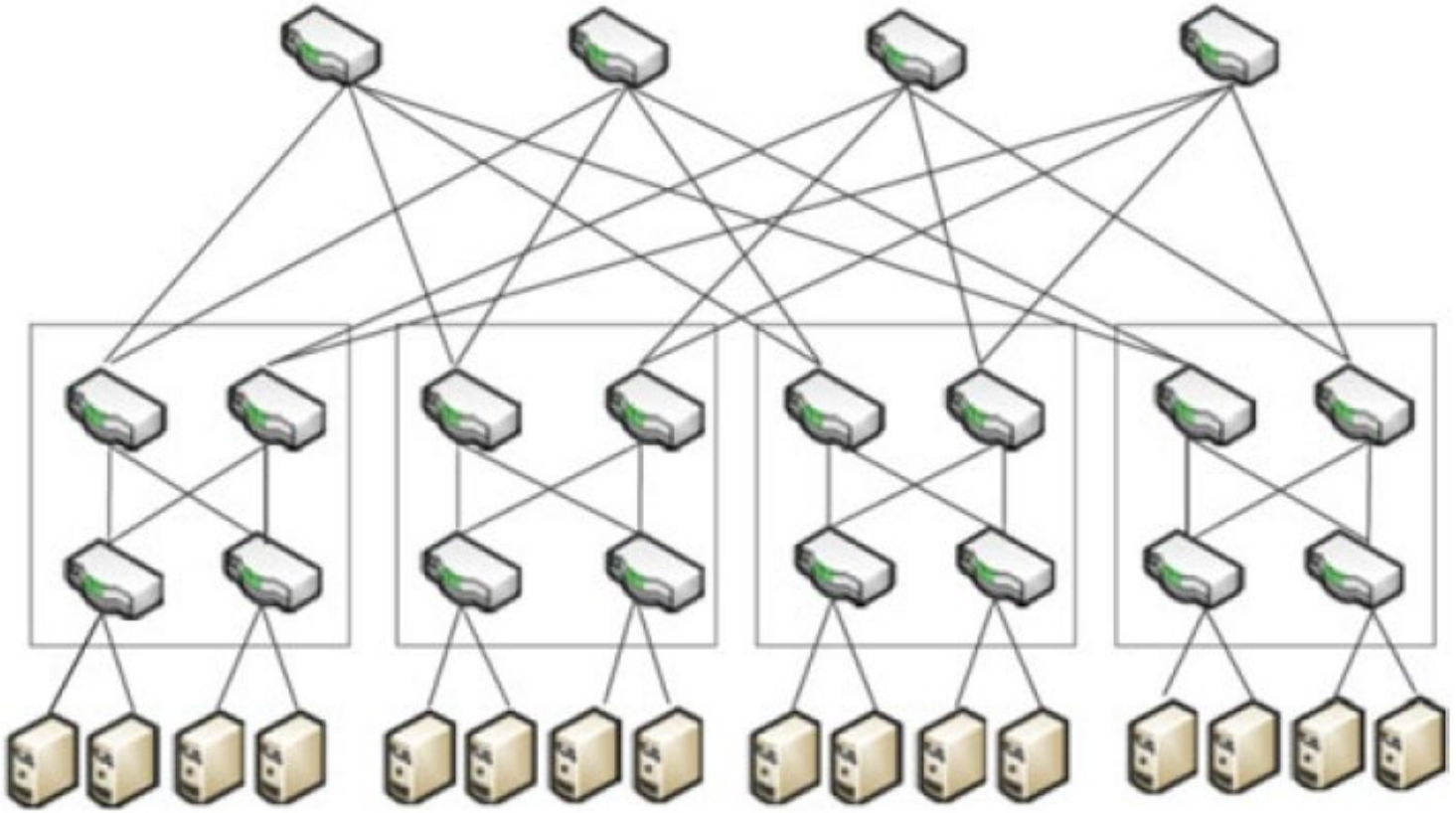
- We further reduce the power consumption in the data center by consolidating PMs that store VM replicas and turning off inactive ones.

- It first finds all the CPMs that has **only one replica VM**, and finds the TPMs for this replica. If at least one such TPM exists, it moves this replica VM to any one of them and turns off this CPM and marks it IPM. We assume that once a PM is identified as TPM, it can no longer be turned off.
- Next, it finds all the CPMs that **has two replica** VMs, tries to find their TPMs and turns off these CPMs. This takes place until all the CPMs are checked.

Quick study of Power-Efficient Virtual Machine Replication in Data Centers

- We generate data centers of different sizes: k = 8, a small data center with 128 PMs; and k = 16, a large data center with 1028 PMs. (where k is the number of ports of each switch)
- The original VMs are randomly generated on the PMs.
- The size of each VM (and its replica copies) is 1 unit.
- The storage capacity of each PM is 30, which means each PM can store maximum 30 VMs.
- Performance Comparison:
  - Performance comparison by varying p, number of original VMs
  - Performance comparison by varying R, number of replica copies of each VMs
- Performance Comparison after server consolidation:
  - Server consolidation by varying p, number of initial VMs.
  - Server consolidation by varying R, number of replica copies of each VM.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization – OPVMP



Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

Preliminaries:

- Reliability is an important aspect of Quality of Service (QoS). With many virtual machines (VMs) running in a cloud datacenter, it is difficult to ensure all the VMs always perform satisfactorily.
- Many solutions have been proposed to address service reliability issues. Here are are four basic reliability enhancement techniques. :
  - Fault removal
  - Fault prevention
  - Fault forecasting
  - Fault tolerance
- The first three, attempt to identify and remove faults that occur in the system with the goal of preventing impact-making faults. (This goal is unrealistic for a complex computing system)
- Last one ,the fault tolerance techniques, tries to ensure service continuity when failure occurs.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

- Many fault tolerance mechanisms have been proposed, below are the two famous onse:

- ***Checkpointing*** : a common fault tolerance mechanism. Periodically saves the execution state of a running task (e.g., as a VM image file), and enables the task to be resumed from the latest saved state after failure occurs (resources and time-consuming.)
- ***Replication*** : another common fault tolerance mechanism, which exploits redundant deployment of computing resources e.g., VMs.
  - **k-fault tolerance** is a specific type of replication-based fault tolerance mechanism and supports a configuration-based fault-tolerance measurement of a server-based service.
  - A k-fault-tolerant service must be configured with *k additional servers* such that the *minimum server configuration for the service can still be satisfied when k hosting servers fail simultaneously*.

- Example: deploying a specific service on only one server makes the service 0-fault-tolerant (because the service becomes unavailable when the only hosting server fails), regardless the number of redundant VMs that may have been deployed on the same server and the feasibility of restoring the affected service via server reboot or replacement.

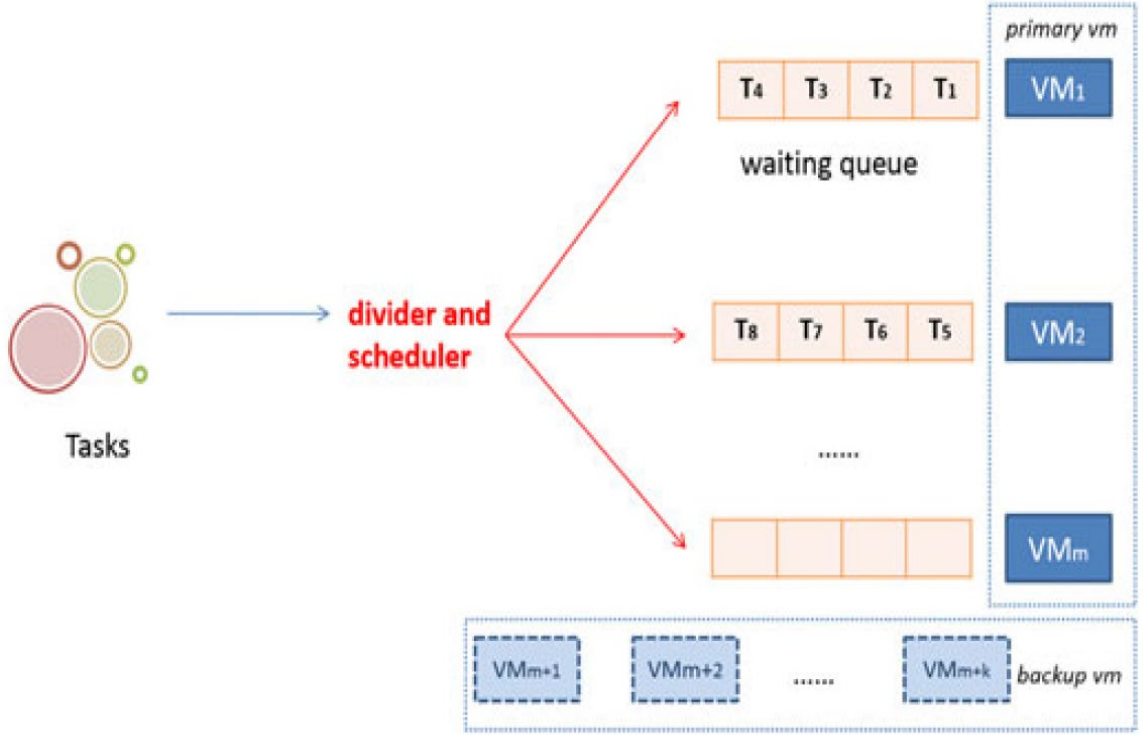
Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

- Re-assigning an incomplete task from one failed primary VM to a backup VM (replica) needs to be done via one of the following ways:
  - From the central storage servers (database) : when a VM failure event is caused by hardware problems and a huge amount of data to be retrieved and processed. This is a time-consuming and network-resource-consuming process.
  - From the server of failed VM: when a VM failure event is caused by software problems and the server which hosts the failed VM may still be running and be able to let the data accessible from within the VMs running on other servers, e.g., the backup VMs that are in proximity to the failed primary VM.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

Task processing model:

- Upon receiving the service requests, the divider partitions the large scale task into smaller sub-tasks.
- Based on the scheduling algorithm, each task to one of the service-providing VMs. Each VM has a task waiting queue. Since a VM may fail due to a software or hardware fault, an assigned task may not be completed as scheduled.
- Besides the **m** number of primary VMs, there are **k** backup VMs **for each service**.
- All the primary and backup VMs are placed on different host servers.
- Failures of one of the primary VMs result in it being mapped to a backup VM, and the tasks in its waiting queue are reassigned to the backup VM.



Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

Backup VMs locations:

1) If the primary and the backup VMs are in the same subnet (connected to the same edge switch), the transfer only consumes the edge-level network resource.
2) However, if the primary and the backup VMs are in the same pod, the transfer will consume both the edge level and the aggregation-level network resources.
3) If the primary and the backup VMs are not even in the same pod, the transfer will consume the edge level, the aggregation-level and core (root) level network resources.

Thus, appropriate VM placement could save considerable amount of time and network resources in failure recovery mode.

- The best solution would be to place all the primary and backup VMs on host servers in the same subnet. However, this may not be possible:
  - If some of the host servers in the datacenter have already been allocated to other tasks and have insufficient free computing resources.
  - A subnet may not even contain a sufficient number of available host servers.

Therefore, it may be necessary to place the (m+k) VMs in different subnets

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

- Aiming at reducing the lost time (enhancing cloud service reliability) and the network resource consumption when the k-fault-tolerance requirement must be satisfied,

this paper proposes a novel redundant VM placement approach to enhancing the reliability of cloud services, which is named:

Optimal redundant virtual machine placement (OPVMP)

Notations

Symbol	Meaning
PM_i	The physical machine or host server in the data center, $i = 1, 2, ...$
VM_j	The virtual machines in the data center, $j = 1, 2, ...$
pod_x	The pods in the data center, $x = 1, 2, ...$
T_y	The task submitted by users, $y = 1, 2, ...$
$subnet_l$	The subnet in the data center, $l = 1, 2, ...$
max_subnet	The number of subnets which contain available host servers
S	A service
$VM_P(S)$	Return the primary VMs of S
$VM_B(S)$	Return the backup VMs of S
$VM_F(S)$	Return current failed VMs of S
$PM(S)$	Return all the servers on which service providing VMs of S locate
$Succ$	The next element after current element
$size$	Return the element number of a list
$DSize$	Return the size of the data stored in a server
a	The number of available PM in a subnet
m	The number of primary VM of S
k	The number of backup VM of S
min	Return the min value of the inputs
$length$	The linkage length
$copy$	Copy all data from a vector to another vector

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

Problem formulation:

$$minUP(S) and UD(S)$$

subject to:

$$UP(S) = \sum_i \sum_j (DSize(pkt_i)) * w_{ij}$$

$$UD(S) = \sum_i \sum_j delay_{ij}$$

- UP(S) denotes the total network consumption for service S.

w_{ij} is 1 if pkt_i transfers through $link_j$. where pkt_i denotes a network packet.

- UD(S) denotes the total data transfer delay for service S.

$delay_{ij}$ denotes the transfer delay of pkt_i transferring through $link_j$.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

Proposed approach:

- The formulated problem essentially involves **finding (k+m) host servers** followed by **placing (k+m) VMs on those host servers**. Since there are a huge number of host servers in a cloud datacenter, the possible number of solutions is exponentially large. It is consequently necessary to identify a subset of good host servers from which to obtain the best solutions. The procedure that was used to select (m+k) good host servers is provided in phase 1 (host server selection) and the algorithm used for placing (m+k) VMs on those host servers is presented in phase 2 (optimal VM placement). Given the information about the failed and the backup VMs, a recovery strategy decision algorithm calculates the optimal matching strategy. The proposed recovery strategy decision algorithm will be discussed in phase 3.
- So the proposed approach is a three-step process with one algorithm for each of the steps, namely :

(1) Host server selection,

(2) Optimal VM placement

(3) Recovery strategy decision.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

1) Host server selection:

- It follow s **"Just enough rule "** for **each service**: we select the pod or subnet with just enough resource, and leave the residual capacities for future use. If none of the subnets have a sufficient number of available host servers, the VMs must be distributed to several subnets or even several pods. In this case, those pods with a greater number of available host servers will be considered first to avoid traffic between pods in the recovery stage (sorting, searching)
- Ex. two subnets, (m+k+20) available host servers, and (m+k+1) host servers, this approach selects second option.

2) Optimal VM placement:

- Placing (m+k) VMs on the (m+k) host servers requires the number of backup and primary VMs in each subnet to be determined.
- A heuristic algorithm is used to solve this problem. Two heuristic conditions are adopted to narrow the searching space:

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

- **First heuristic condition:**
  - If there are even number of available servers **in a subnet**, the number of backup vm in the subnet(Ki) should be less or equal to the number of primary vm in the subnet (Mi). (Ki=<Mi)
  - If there are odd number of available servers in a subnet, the number of backup vm in the subnet should be less or equal to one plus the number of primary vm in the subnet.(Ki=< (1+Mi))

So we need to have enough number of backup for the primary VMs.

- Ex: Suppose there is a subnet that contains fewer primary VMs than backup VMs. As the total number of primary VMs is larger than or equal to the number of backup VMs, there is at least one subnet in which the number of backup VMs is smaller than the number of primary VMs. Now, a backup VM in the first subnet and a primary VM in the second subnet exchange position with each other.
  Compared to the first strategy, one more failed VM in the second subnet does not require to be mapped to a backup VM in different subnets when k number of VMs fail at the same time (as we just gave it a backup VM). The new placement strategy will consume less aggregation layer network resource.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

➢ **Second heuristic condition:**

- The subnet that contains more available host servers should be allocated more backup VMs. (the difference between the number of primary VMs and backup VMs of it should be smaller). Otherwise, there is a larger chance that the data transfer would consume more network resources. (less available k for the failed Ms)

  - The algorithm is recursive in nature. In each recursion, the algorithm determines the number of backup VMs that are placed in the current subnet, and the rest backup VMs are placed in the following subnets.
  - When the number of rest backup VM equals 0 or the last subnet has reached, the resource consumption of the current placement strategy is computed and compared with the current optimal one.
  - If the resource consumption is smaller than that of the current optimal strategy, the current strategy is considered optimal.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

3) Recovery strategy decision

- When one or more VMs fail, a recovery strategy has to be decided upon, and each failed VM has to be mapped to a backup VM (**it selects which backup VM needs to be matched here**). All tasks in the waiting queue of the failed VM are rescheduled to its mapping backup VM, and the data to be processed have to be retrieved again to the backup VM.
- The recovery strategy should minimize the total network resource consumption.

- In this case it is possible to formulate the recovery strategy decision problem as a minimum weight matching in bipartite graphs.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

Given a complete bipartite graph $G=(V,E)$ with bipartition(V_F,V_B), where V is the set of all failed VMs and backup VMs, V_F is the set of all failed VMs, V_B is the set of all backup VMs, and E is the set of shortest paths connecting nodes for each pair of VMs from different partitions. A matching set M is a subset of E. Suppose w denotes the weight function, and then it is necessary to find a matching of minimum weights where the weight of matching M is given by:

$$w(M) = \sum_{e \in M} (w(e))$$

In other words, the recovery problem can be formulated as the following:

$$min \sum_{(V_F,V_B)} (w(v_f, v_b) * x(v_f, v_b))$$

$$w(v_f, v_b) = DSize(v_f) * length(e(v_f, v_b))$$

$x(v_f,v_b) = 1$ if the edge (v_f,v_b) belongs to the matching; otherwise, $x(v_f,v_b) = 0$. $DSize$ in (19) returns the data size that can be retrieved from the host server on which v_f is placed. Therefore, $w(v_f,v_b)$ denotes the transfer cost.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

- For each backup VM, it tries to find a corresponding failed VM in the same subnet for it if there is one. Otherwise, VMs in different subnets would have to be mapped.

Step 1: For each backup VM, all failed VMs in the same subnet that have not been matched are sorted according to their data size. The backup VM is mapped to the failed VM with the largest data size.

Step 2: If there still remain failed VMs that have not been matched to a backup VM, all unmatched failed VMs in the same pod are sorted according to data size. The backup VM is mapped to the failed VM with the largest data size in the pod.

Step 3: If there still remain failed VMs that have not been matched to a backup VM, the backup VMs are randomly matched to the failed VMs. In addition, the data are re-fetched from the storage server.

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

Experimental Setup:

- 32-port fat-tree data center network is constructed. There are 16 host servers in each subnet.
- Each of these host servers can host 4 VMs at most.
- The capacity of the root-layer link and aggregation-layer link is set as 10 Gbps, and the capacity of the edge-layer link is set as 1 Gbps.
- All the methods were evaluated using the following performance metrics:

PRoot: The total size of network packet that has been transferred by the root layer switches. *PRoot* can be calculated as follows:

$$PRoot = \sum_i w_r \times size(pkt_i), \quad (21)$$

where w_r denotes the frequency with which packet pkt_i has been transferred by the root switches.

$$PAgg = \sum_i w_a \times size(pkt_i),$$

$$PEdge = \sum_i w_e \times size(pkt_i),$$

$$PTotal = PRoot + PAgg + PEdge,$$

$$TDelay = \sum_i delay(pkt_i),$$

Quick study of Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization

Performance Evaluations:

- The experiment involved 20 services,
- Each of which involved 50 primary VMs and 40 backup VMs.
- 200 VM failure events were triggered.
- 400 data processing tasks were generated.
- The data size of each task is 300 MB
- The task size is set as 10 minutes. The task arrival rate of each service is 200 per hour.

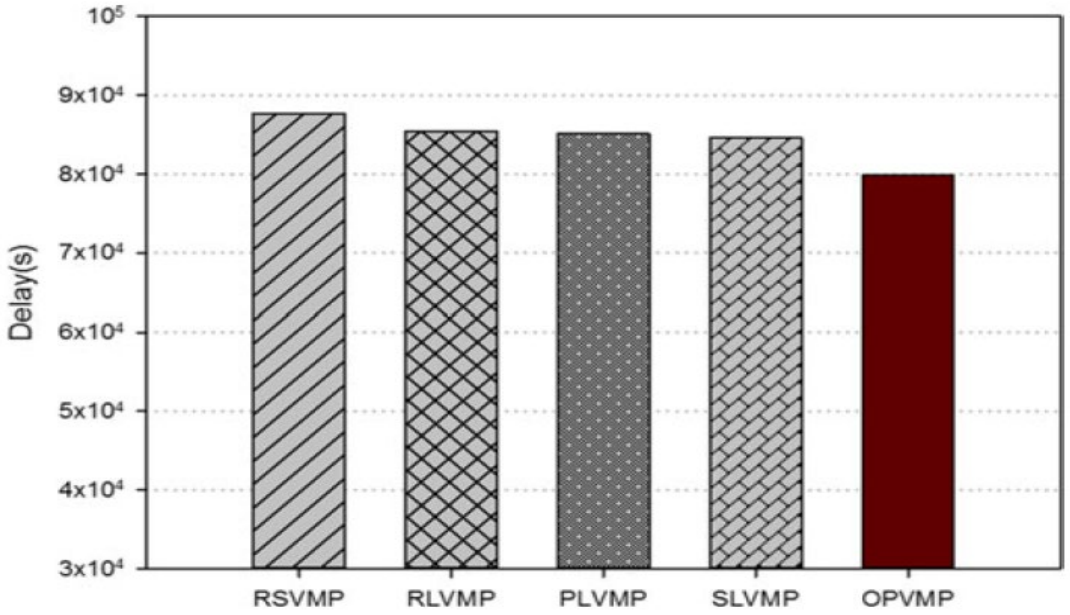


Fig. 3. The performance of total data transfer delay (s).

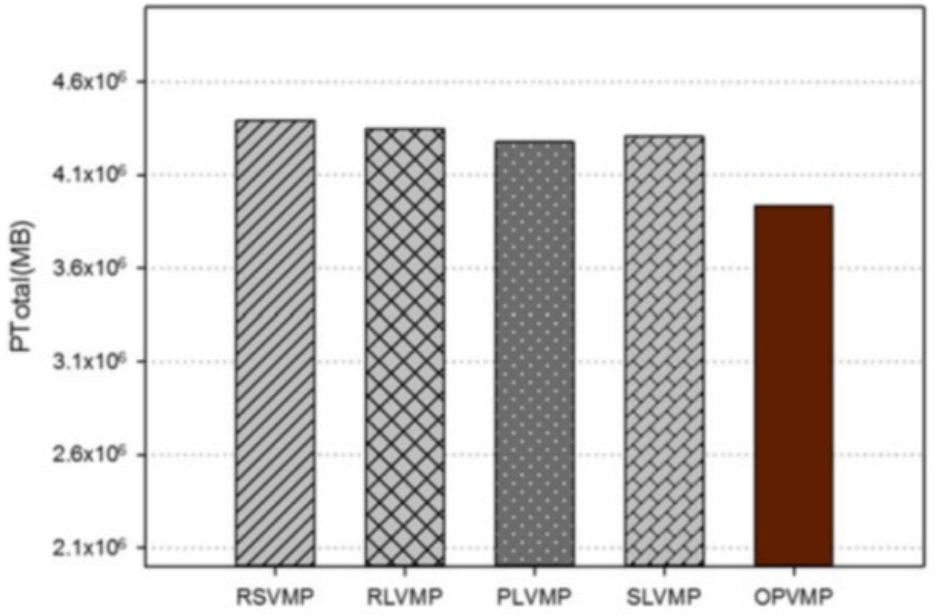


Fig. 7. The performance of total network resource consumption (MB).

Reviewing their differences

Reviewing their differences

- We had same number of PM and capacity on all subnets and pods vs in their paper not clear
- We didn’t talk about job/service/data size vs they do
- We had different number of copies for each VM (3 to 11) vs they have k additional servers (backups) such that the minimum server configuration for the service can still be satisfied when k hosting servers fail simultaneously.
- We had 8 an16 ports switch vs they have 32 ports.
- We had each PM can store maximum 30 VMs vs they have each host servers can host 4 VMs at most.
- We had 1, 5, 10 as switches power consumptions vs they set 1 and 10 as switch capacities (Gbps) to calculate transfer cost or network resource consumption. Eventually they evaluate the performance of total network resource consumption as the size of packets processed (MB)

Possible ways to combine them

Possible ways to combine them

- We may be able to use their first phase (Host server selection) to place our original VMs instead of placing them randomly.
  However to do so, we need to define different services and subnet/pods (they can have different number of servers and each server can have different capacity)

- Inserting file/packet size to our formulation:

$$\sum_{j=1}^{p}\sum_{k=2}^{R} s_j \times c_{S(j),r(j,k)}, \quad c_{i,j} = x * r_e + y * r_a + z * r_c.$$

$$UP(S) = \sum_i \sum_j (DSize(pkt_i)) * w_{ij}$$

$$w(v_f, v_b) = DSize(v_f) * length(e(v_f, v_b)),$$

$$PEdge = \sum_i w_e \times size(pkt_i),$$