ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2021.0350

REPLICATION AND MIGRATION COST MINIMIZATION OF CLOUD DATA CENTER

T. Arunambika and P. Senthil Vadivu
Department of Computer Science, Bharathiar University, India

Abstract

Cloud Storage Providers (CSPs) provide geography features of an area data stores offering numerous classes for storage accompanied by various costs. One significant problem faced by cloud consumers is how to utilize these storage classes to deliver a device with the time-varying workload on its objects at a minimal price. This price includes the price of resident (for example, storage, put and get prices) and the possible migration price (for example, network price). This paper proposes the Replication and Migration Cost Minimization (RMCM) algorithm in Green Cloud Computing to tackle this issue. This algorithm is using VM's direct migration method, which could decrease data centre prices and power usage by combining virtual resources. To decrease the price of data-placement for devices accompanied by varying workloads over time, developers must make optimal use of the price variation between storage and network services across multiple CSPs. This paper proposes an optimal cost-effective technique (OCET) for copying and migrating data into cloud data centres accompanied by numerous storage classes to attain this aim. This work goal to achieve price reduction in the load assign procedures in multiple data centre environments where virtual machines allocated to a provided data center taking into account energy price differences and the availability of local renewable power generation. The simulation outcomes demonstrate that the RMCM and OCET algorithm could reduce a replication and migration cost and decrease the power consumption of data centre in green cloud computing efficiently.

Keywords:

Replication Cost, Migration Cost, Cost Minimization, Virtual Machine Migration, Cloud Data Centre

1. INTRODUCTION

Google Cloud Storage (GCS), Amazon S3 with Microsoft Azure like most important Cloud Service Providers (CSP) provide file, block, and blob, and so on for storage accompanied by various costs. All CSP offers application programming interface (API) instructions to store recover and erase information via services of the network that inflicts out- and in-network price in an appliance. On most important CSPs, price of in-network is without charge, when out- network price is charge and might be dissimilar for CSPs. Furthermore, information transmitting across data centres (DCs) of a provider in various areas might price on a low fee (hereafter, it described decreased out- network price). This paper aspires on enhancing price which comprises of resident price with probable network price (for example, migration price)

The price of storage data managing is affected by the predictable amount of work of an object. Here is a challenging statistical technique amid the object load of work and the nature of an object, as experimental in an online social network (OSN) [1] with delay responsive multimedia substance used using mobiles [2] [3]. An object may be a tweet, a photo, a video, or still incorporation of these objects which distribute the same write with reading usage rate pattern. An object task load decided through how frequently it is written (rate of Put access) with reading (rate of Get access).

The rate of Get access for the object upload to OSNs is frequently too high in an initial life of an object, along with such an object supposed to be in hot-spot status and read-intensive. On the contrary, like time passes, an object rate of Get access is decreased, in addition to it goes to the cold-spot position wherever it believed like storage serious. A related fashion occurs for the object Put workload; namely, the rate of Put access reduces like time developments. Therefore, OSNs use multiple networks than storage in an initial life of an object, with like time passes; they employ storage over the network. So, (i) among the specified time changeable workloads in objects, with (ii) storage class provided through various CSPs accompanied by various costs, obtaining a not expensive network with storage resource at the suitable time of an object life span acts an essential character in the price enhancement of a data organization across CSPs.

Cloud consumers are necessary to respond two queries: (i) which class of storage from which provider must host an object (for example, placing), (ii) when an object must perhaps migrate from the storage class to one more possessed through the same or various providers. Lately, numerous analysis gets the benefit of cost variations of various possessions in inter- and intra-CSPs, where price could optimize through dealings off storage vs calculate [4], cache vs storage [5] [6], and price enhancement of information dispersal across CSPs [7] [8].

None of these analyses examined a tradeoff among storage with network price to enhance the price of migration with replication information across numerous providers. Also, these methods a lot depend on task load forecast. It is not forever possible with might guide to incorrect outcomes, particularly in the subsequent cases: (i) when forecasting approaches used to forecast long-term task load in the upcoming (for example, one year), (ii) for establishing industries which contain restricted or no the past of requiring information, in addition to (iii) when task loads are extraordinarily changeable and non-stationary.

This paper proposes the Replication and Migration Cost Minimization (RMCM) algorithm reduce the costs and power usage of a data centre by combining all virtual resources. Furthermore, this paper presented an Optimal Cost-Effective Technique (OCET) performed by cloud service providers mainly focuses on minimizing the cost to maintain all physical machines. The cost minimization typically achieved by reducing electricity consumption. A proposed approach involves dynamically halting physical machines and virtual machine migration. Conversely, for cost optimization executed by cloud consumers is to choose the correct cost-efficient CSP selection.

The remaining section of the work prepared as follows: the related work about replication and migration cost minimization reviewed in Section 2. The process of Replication and Migration Cost Minimization (RMCM) algorithm described in Section 3.

The process of Optimal Cost-Effective Technique (OCET) algorithm explained in Section 4. The outcomes of experiments are conversed in Section 5, after that Section 6, which completes the paper.

2. RELATED WORK

Fahrenheit et al. [9] suggested an approach of dynamic incorporation of virtual resources using an ant colony system (ACS), also known as the ACS-VMC approach. This algorithm turned the virtual resource incorporation issue within a multipurpose enhancement issue. Objective purposes of the multipurpose enhancement issue include reducing power usage, reducing the quantity of VM displacements, and keep away from SLA violations. Furthermore, by the PM load, the authors separated the PMs within four groups: P_{under}, P^{over}, P_{over}, P_{normal}.

Sohrabi et al. [10] presented the Bayesian Migration Heuristic (BMH) system. Furthermore, the BMH system is a heuristic resource incorporation system using Bayesian network. The BMH system developed the Migrant VM suite using estimates of Bayesian. Their outcomes demonstrated that BMH could successfully decrease the power usage of a data centre.

Wen et al. [11] suggested a Virtual Resource Dynamic Integration (VRDI) technique for reducing energy consumption in cloud computing. This method proposed minimum migration policy for VM selection and genetic algorithm for VM placement and migration. They concluded that the VRDI method reduces energy consumption efficiency in cloud computing.

Jiayin Li et al. [12] presented an adaptive resource allocation mechanism for the cloud technique and controllable functions to adapt resource allocation using updated real work execution. Fixed task planning for resource allocation uses the adaptive list scheduling (ALS) and adaptive Min-Min scheduling (AMMS) algorithms for task planning. An online adaptation process utilized often to re-evaluate sustainable resource allocation and a prior event. At each reassessment process, planners recalculate the power usage of their accurately submitted works.

Yazir et al. [13] present twice, in which the earliest scattered architecture modified. Resource management separated into autonomous tasks, all of which is expert by autonomous Node Agents (NAs) in a rotation of three processes: (1) VM Placement, VM by finding the suitable PM, which is capable of executing provided VM; (2) the total resources that the hosted VM monitors; (3) If local shelter is impossible, one VM will transfer to another PM should be relocated to, and hired during VM selection [20].

Sahar et al. [14] presented a genetic algorithm for assigning works efficiently to VMs, which allot resources using existing resources and power usage of all VM. Experimental outcomes demonstrate that the presented algorithm provides the best outcomes than the better-fit reduction and initial fit reduction algorithms. This technique should be enhanced with other techniques, besides consider the QoS parameters in cloud surroundings.

Quan et al. [15] presented an efficient power-proficient resource allocation mechanism named T-Alloc. The T-Alloc mechanism is focused on conventional data centres and is a multi-core processor with a single-core processor. Furthermore, along with the VM load, the T-Alloc mechanism adjusts the processor number energetically to decrease power usage.

Anton et al. [16] proposed the modified best Fit decreasing (MBFD) mechanism for placement of VM. Initially, the MBFD algorithm organized VMs in declining order of CPU usage. Furthermore, the MBFD mechanism moved VMs by merging the power usage of PMs and QoS of VMs.

Bobroff et al. [17] utilized a gateway to CPU usage. Once the CPU utility reaches the threshold, the resource incorporation plan induced. The method demonstrated a 30%reduction in data centre resource usage.

3. REPLICATION AND MIGRATION COST MINIMIZATION ALGORITHM

This algorithm classified into three sub-algorithms. The first sub-algorithm is RMCM-PM Selection algorithm. This algorithm discovers a group of PMs, which must be incorporated, using the load patterns and the equivalent thresholds of PMs. The second sub-algorithm is RMCM-VM Selection algorithm. This algorithm discovers a set of VMs that organized on the chosen PMs using the pattern of load and the connection among the PM load and the VM load. The last sub-algorithm is RMCM-VM Placement and Migration Algorithm. For all VM in a chosen set of VM, this algorithm discovers the novel PM, which could fulfil its resource necessities.

A Replication and Migration Cost Minimization algorithm minimizes the power usage of a data centre using reduces direct relocation of Virtual Machines and increases the number of closing PMs. Direct migration could move a VM from one PM to a new not affecting a standard service. Based on the VM direct relocation, this algorithm integrates virtual resources with shut the inactive PMs to decrease power utilization. Algorithm 1 explains the proposed RUFES algorithm.

Algorithm 1: Replication and Migration Cost Minimization (RMCM) algorithm

Input: Datacenter (*DC*), Virtual Machine Monitor (*VMM*), PM Set (pm_{List}), VM Set (vm_{List})

Output: Energy saving based on resource integration

Step 1: VMM collect data from the DC

$$pm_{List} = \{PM_1, PM_2, \dots, PM_n\}$$

PM_1 has Physical Machine 1's CPU and Memory Utilization.

Here,

$$PM_1 = \{PM_1^{cpu}, PM_1^{mem}\}, \text{ Similarly } PM_2, PM_3, \ldots$$

Step 2: CPU and Memory Utilization may be different in each PM.

a. To ensure the accuracy of the RMCM, first normalize it using Normalization.

b. Normalization means CPU and Memory values could be transmitted to a data with no physical dimension within [0, 1]

Step 3: *RMCM-PM Selection Algorithm*: Along with the PM's resource usage and PM's predefined thresholds associated with it, a group of PMs should be integrated resources to reduce power usage

Step 4: *RMCM-VM Selection Algorithm*: Along with the resource usage of PMs in the prior phase, two migration schemas are regarded. In the initial schema, each VMs organized on the PM must be moved. In the second schema, only a portion of the VMs need to be moved.

Step 5: *RMCM-VM Placement and Migration Algorithm*: Along with the outcomes of the RMCM-VM selection algorithm, for the VM that require to be moved, the RUF based VM placement algorithm selects another PM to migrate the VMs.

Step 6: The PM which has inferior threshold resource usage ($\leq Lower_i$) could be shut down.

First, the virtual machine monitor (VMM) element utilized to detect and gather information from a data centre (Step 1). This data contains PM Set. Furthermore, this set has each PMs CPU and Memory Utilization values. Finally, these values have different dimensions. So, first, normalize it into dimensionless data in the interval 0 to 1 (Step 2). After normalization, this algorithm applies RMCM-PM Selection (Step 3) which explained in section 1. It selects suitable PMs, which has the resource utilization less than a threshold value or greater than the threshold value. Followed by, this algorithm applies RMCM-VM Selection (Step 4) which explained in section 2. It selects suitable VMs from selected PMs Set. Furthermore, this algorithm applies RMCM-VM placement and Migration (Step 5), which explained in section 3. After VM migration which PM has less resource utilization, it should power off to reduce power usage and cost in a datacenter.

3.1 RMCM-PM SELECTION ALGORITHMS

Along with the resource usage of PMs with the equivalent predefined thresholds of a PMs, this algorithm generates a group of PMs which must be incorporated resources to decrease power usage. Algorithm 2 presents an RMCM-PM Selection algorithm. For PM_i, the load model is indicated based on $U_i = \{U_i^{cpu}, U_i^{mem}\}$. This algorithm denotes the low threshold of resource usage based on $Lower_i = \{Lower_i^{cpu}, Lower_i^{mem}\}$ and a higher threshold of resource usage based on $Upper_i = \{Upper_i^{cpu}, Upper_i^{mem}\}$. If the connection between resource usage and an inferior threshold of the PM satisfy,

$$(U_i^{cpu} < Lower_i^{cpu})\ \&\&\ (U_i^{mem} < Lower_i^{mem}) \tag{1}$$

It is essential to combine the virtual resource of a PM. This algorithm suggests every VMs organized on a PM to a new PM should be migrated, with a PM should group to an inactive or power-off condition. This algorithm denotes a combination executed in this situation based on allList. Furthermore, when a PM resource usage is near to a full load, it might disturb the VM performance. If a connection among resource usage and an upper threshold of a PM satisfy,

$$(U_i^{cpu} \geq Upper_i^{cpu})\ \&\&\ (U_i^{mem} \geq Upper_i^{mem}) \tag{2}$$

Algorithm 2: RMCM-PM Selection

Input: PM List (pm_{List}), Lower Threshold Resource Utilization ($Lower_i$), Upper Threshold Resource Utilization ($Upper_i$)

Output: all_{List}, $part_{List}$ (PM Set to be integrated), $otherPM_{List}$

```
$all_{List}=\{\}$, $part_{List}=\{\}$, $otherPM_{List}=\{\}$;
For each $PM_i$ in $pm_{List}$ do
  {
    Case 1: $PM_i$ Load Pattern $\rightarrow U_i = \{U_i^{cpu}, U_i^{mem}\}$
    Lower Threshold Resource Utilization $\rightarrow Lower_i = \{Lower_i^{cpu}, Lower_i^{mem}\}$
    IF $U_i^{cpu} < Lower_i^{cpu}$) && ($U_i^{mem} < Lower_i^{mem}$)
      {
        // $PM_i$ Selected
        // move around entire VMs positioned on a $PM_i$ to a new PM
        // $PM_i$ should be put to the inactive or power off status
        $all_{List} = all_{List} \cup PM_i$
        }
    Case 2: $PM_i$ Load Pattern $\rightarrow U_i = \{U_i^{cpu}, U_i^{mem}\}$
    Upper Threshold Resource Utilization $\rightarrow Upper_i = \{Upper_i^{cpu}, Upper_i^{mem}\}$
    IF ($U_i^{cpu} \geq Upper_i^{cpu}$) && ($U_i^{mem} \geq Upper_i^{mem}$)
    (Uicpu ≥ Uppericpu) andand (Uimem ≥ Upperimem)
      {
      // $PM_i$ Selected
      // migrate only a part of the VMs positioned on the $PM_i$ to a new PM
      // (If memory is filled to capacity, it would disturb the QoS of a VMs positioned on a $PM_i$)
      // (It means, should be migrate overloaded VMs positioned on a $PM_i$ to a new PM)
      $part_{List} = part_{List} \cup PM_i$
      }
    Else if $PM_i$ is not satisfied in both cases, it should be added in $otherPM_{List}$.
    {
    $otherPM_{List} = otherPM_{List} \cup PM_i$
    }
    }
End FOR
Return $all_{List}$ and $part_{List}$ to RMCM-VM Selection Algorithm
```

It is essential to incorporate a virtual resource of a PM. Because of one of the U_i^{cpu} or U_i^{mem} is upper than a higher threshold, it means memory-filled; it would disturb a QoS of a VMs positioned in a PM. This algorithm suggests merely the fraction of a VMs to new PMs should be migrated in a data centre to decrease a resource usage of a PM. This algorithm denotes the incorporation executed in this situation based on $part_{List}$. If PM is not satisfied in both cases, it should add in $otherPM_{List}$.

3.2 RMCM-VM SELECTION ALGORITHM

Along with a result obtained by a previous algorithm, two migration situations considered. Followed by, in an initial situation, entire VMs positioned in a PM must be moved. In the second situation, merely a fraction of VMs must migrate. Algorithm 3 explains the RMCM-VM Selection. For an all_{List} case, all of VMs require to move out of a PM. A PM requires adjusting on an inactive condition. It all VMs are added in $selectedVM_{List}$ (Step 5). For a $part_{List}$ situation, it is essential to choose a VMs set to move. This algorithm first calculates the

Euclidean distance among the PM's load pattern with VM's load pattern.

Euclidean distance (dij) = 1/√((Uicpu-Ujcpu)2) + 1/√ ((Uimem-Ujmem)2) (3)

Keep in mind that over large distances, the PM's effect dominates. Step 16-30 shows how to select the VMs which consume more resources. Through merely transmitting VMs that use numerous resources, the RUFES-VM chosen algorithm could efficiently decrease the quantity of VMs to migrate.

Algorithm 3: RMCM-VM Selection

Input: all_{List}, $part_{List}$, Upper Threshold Resource Utilization ($Upper_i$)

Output: $selectedVM_{List}$ (VM Set to be migrated)

```
Upper Threshold Resource Utilization → $Upper_i = \{Upper_i^{cpu}, Upper_i^{mem}\}$
$selectedVM_{List}=\{\}$;
For each $PM_j$ in $all_{List}$ do
  $X$ = get All VM's List deployed in $PM_j$
  $selectedVM_{List} = selectedVM_{List} \cup X$
End For
For each $PM_j$ in $part_{List}$ do
  $PM_j$ Load Pattern $\rightarrow U_j = \{U_j^{cpu}, U_j^{mem}\}$
  $X$ = get All VM's List deployed in $PM_j$
  $D=\{\}$;
  For each $VM_i$ in $X$ do
    $VM_i$ Load Pattern $\rightarrow U_i = \{U_i^{cpu}, U_i^{mem}\}$
    Find Euclidean distance
      $d_{ij} = \frac{1}{\sqrt{\left(U_i^{cpu} - U_j^{cpu}\right)^2}} + \frac{1}{\sqrt{\left(U_i^{mem} - U_j^{mem}\right)^2}}$
    $D = D \cup d_{ij}$
  End For
  $newVM_{List}$ = Sort $D$ using ascending order
    For each $VM_i$ in $newVM_{List}$ do
      Remove $VM_i$ from $PM_j$
      $selectedVM_{List} = selectedVM_{List} \cup VM_i$
      Calculate current Load Pattern of $PM_j \rightarrow U_j = \{U_j^{cpu}, U_j^{mem}\}$
      If($U_j^{cpu} \geq Upper_i^{cpu}$) && ($U_j^{mem} \geq Upper_i^{mem}$)
        {
        continue;
        }
        else
        {
        break;
        }
      End if
  End For
End For
```

Return $selectedVM_{List}$ to RUF based VM Placement and Migration Algorithm

3.3 RMCM-VM PLACEMENT AND MIGRATION

Along with results of an RMCM-VM selection algorithm, for VMs which require migrating, an RMCM-VM placement algorithm selects another PM to migrate the VMs. Algorithm 4 presents RMCM-VM placement and migration algorithm. This algorithm calculates the Resource Utilization Factor (RUF) among the VM's load model (from selectedVMList) and PM's load model (from otherPMList).

$$RUF = [0.5(U_j^{cpu} + U_i^{cpu})] + [0.5(U_j^{mem} + U_i^{mem})] \quad (4)$$

This algorithm suggests which PM has the highest Resource Utilization Factor value that is suitable for VM migration. For Example, Let VM_1 = (5 MIPS, 6 MB) is waiting at the queue for the position. Let, there is 3 PMs are available PM_1 = (8 MIPS, 9 MB), PM_2 = (6 MIPS, 7 MB) and PM_3 = (9 MIPS, 10 MB). After that, this algorithm calculates RUF values,

$$PM_1=(8MIPS,9MB) \rightarrow (8+5)/2+(9+6)/2 = 14$$

$$PM_2=(6MIPS,7MB) \rightarrow (6+5)/2+(7+6)/2 = 12$$

$$PM_3=(9MIPS,10MB) \rightarrow (9+5)/2+(10+6)/2 = 15$$

Here, the RUF value of PM_3 is high. Therefore, the suitable PM is PM_3.

Algorithm 4: RMCM-VM Placement and Migration

Input: $selectedVM_{List}$ (group of VMs are waiting at a queue for position), $otherPM_{List}$ (set of suitable PMs for placement), all_{List}

Output: RUF based VM Placement and Migration

```
For each $VM_i$ in $selectedVM_{List}$ do
  $VM_i$ Load Pattern $\rightarrow U_j = \{U_i^{cpu}, U_i^{mem}\}$
  rufList = {}
  For each $PM_j$ in $otherPM_{List}$ do
    $PM_j$ Load Pattern $\rightarrow U_j = \{U_j^{cpu}, U_j^{mem}\}$
    //check $PM_j$ is sufficient for $VM_i$ placement
    If(!(($U_i^{cpu} \geq U_j^{cpu}$) && ($U_i^{mem} \geq U_j^{mem}$)))
      {//Resource Utilization Factor Calculation
        $RUF = [0.5(U_j^{cpu} + U_i^{cpu})] + [0.5(U_j^{mem} + U_i^{mem})]$
        $ruf_{List} = ruf_{List} \cup RUF$ with $PM_j$
      }
  End For
    if(!(rufList.isEmpty()))
      {
        Sort $RUF$ using Descending Order in $ruf_{List}$
        $best_{PM} = PM$ has Highest $RUF$ value
        Migrate $VM_i$ to $best_{PM}$
        Remove $VM_i$ from $selectedVM_{List}$
      }
        else
      {
        For each $PM_j$ in $all_{List}$ do
        $PM_j$ Load Pattern $\rightarrow U_j = \{U_j^{cpu}, U_j^{mem}\}$
        //check $PM_j$ is sufficient for $VM_i$ placement
```

```
            If(!(($U_i^{cpu} \geq U_j^{cpu}$) && ($U_i^{mem} \geq U_j^{mem}$)))
             {
              Migrate $VM_i$ to $PM_j$
              Remove $VM_i$ from $selectedVM_{List}$
              break;
             }
          End For
        }
End For
```

4. OPTIMAL COST-EFFECTIVE TECHNIQUE

In this section, we present Optimal Cost-Effective Technique (OCET) for replication and migration of information at cloud data centres with multiple storage classes. OCET is performed by cloud service providers mainly focuses on minimizing the cost to maintain all physical machines. The cost minimization typically achieved by reducing electricity consumption. A proposed approach involves dynamically halting physical machines and virtual machine migration. Conversely, for cost optimization executed by cloud consumers is to choose the correct cost-efficient CSP selection. The Fig.1 shows cost optimization based on OCET.

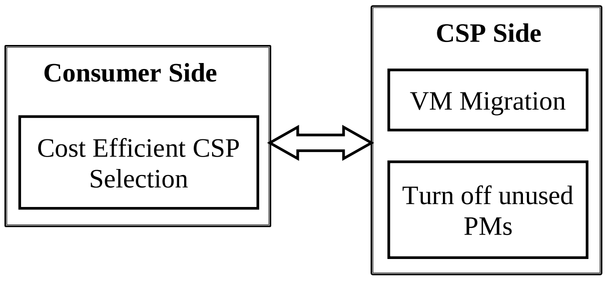

Fig.1. Cost Optimization based on OCET

Algorithm 5: Optimal Cost-Effective Technique

Input: Virtual Machine Monitor (VMM), All Available Datacenters (AADs), Datacenter (DC), PM Set (pm_{List}), VM Set (vm_{List}), Energy Threshold (ET)

Output: Cost effective data center ($bestDC_{Id}$) for storage

```
String bestDCId="";
Double minCost=0;
For datacenter DC from AADs
  {
    VMM collect data from the DC
                $pm_{List} = \{PM_1, PM_2, \ldots, PM_n\}$
        $PM_1 = \{PM_1^{cpu}, PM_1^{mem}\}$, Similarly $PM_2, PM_3, \ldots$
    Normalize CPU
    Normalize Memory Utilization.
    //Normalization alters CPU and Memory to the data without
    dimension in the range [0, 1]
  }
Based on the resource usage of PMs with an equivalent pre-defined
Reduce power usage by proper incorporation of PM thresholds, where a group of PMs selected
Measure the following migration situations based on the resource usage of a PMs at a previous step,
  Move entire VMs positioned in a PM must be moved in an initial situation
  Move merely a fraction of the VMs must be moved in the second situation
OCET-VM Replication and Migration: Based on the results of the
OCET-VM selection, for VMs which require to be moved, a VM placement technique selects another PM to
Migrate the VMs
The PM which has Inferior Threshold Resource Usage (≤ $Lower_i$) could be shut down.
  Double Energy = calculateEnergyConsumption ($pm_{List}$, $vm_{List}$, $migration_{Count}$);
                Double Cost = energy * ET;
if (minCost > cost)
{
minCost = cost;
bestDCId = datacenterId;
}
```

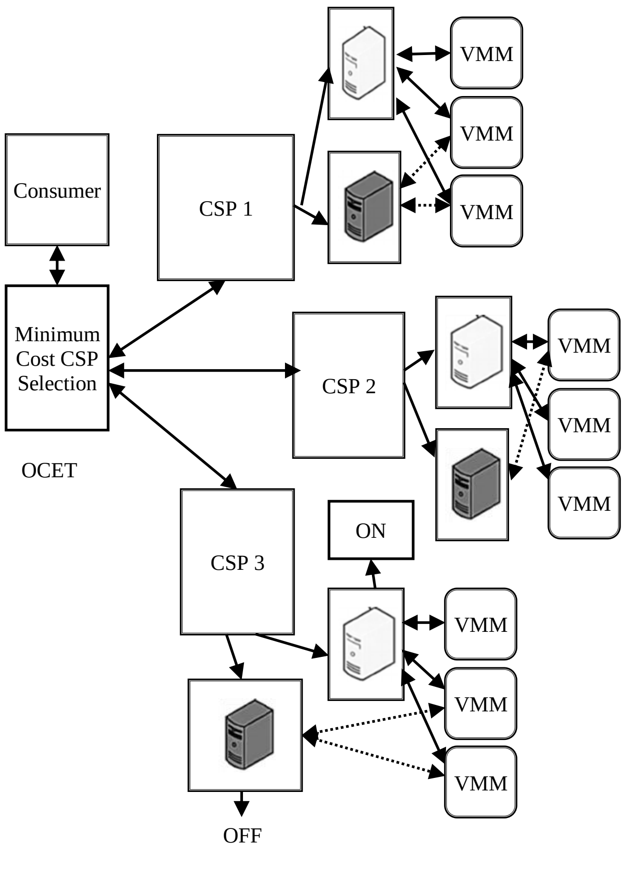

Fig.2. OCET system architecture

Furthermore, Algorithm 5 shows the optimal cost-effective technique. First, the virtual machine monitor (VMM) collects all data centre Metadata. It utilized to watch and gather information from a data centre (Step 5). This data contains PM Set.

Furthermore, this set has each PMs CPU and Memory Utilization values. These values have different dimensions. So, first, normalize it into dimensionless data in the interval 0 to 1 (Step 6). After normalization, this algorithm applies OCET-PM Selection (Step 7). It selects suitable PMs, which has the resource utilization less than a threshold value or greater than the threshold value. Followed by, this algorithm applies OCET-VM Selection (Step 8). It selects suitable VMs from selected PMs Set. Furthermore, this algorithm applies to OCET-VM placement and Migration (Step 9). After VM migration which PM has less resource utilization, it should power off to reduce energy consumption and cost in the datacenter. Now the customer can choose less cost consumption datacenter as the best datacenter (Step 16). The Fig.2 shows the proposed OCET architecture.

5. RESULTS AND DISCUSSIONS

To confirm the efficiency of the proposed Optimal Cloud Service Provider Selection Algorithm and execute a vast number of frequent experiments CloudSim toolkit [18], to simulate the experimentation used. The CloudSim permits its user to generate cloud simulation including cloud initialization, datacenter, physical machine, virtual machine generation, datacenter selection, VM migration and task scheduling etc. The extended mechanism could give execution time and statistics of a simulation. Furthermore, the CloudSim could further simulate the optimal data centre selection. Based on the CloudSim toolkit, this work generated a data centre consisting of 100 physical machines. It contains two kinds of PMs, that is, the IBM X3550 and the HP ProLiant ML110 G5. There are 5 to 20 VMs which contain various load modes in every PM. Furthermore, this work set the Lower and Upper Threshold Resource Utilizations,

$$Lower_i = \{Lower_i^{cpu}, Lower_i^{mem}\} = \{0.30, 0.30\}$$

$$Upper_i = \{Upper_i^{cpu}, Upper_i^{mem}\} = \{0.60, 0.60\}$$

Followed by, four existing algorithms used for comparison, namely FFA [19], ACSVMC [9], BMH [10], and VRDI [11]. First, this work compares existing algorithms with the proposed RMCM algorithm based on the whole power usage of a datacenter. Under a situation of containing a similar quantity of VMs, an FFA [19], ACS-VMC [9], BMH [10], and VRDI [11] algorithms compared with the proposed RMCM algorithm. The Table.1 shows the comparison of energy consumption between different VMs.

Table.1. Energy consumption comparison

Algorithm	Number of VMs				
	200	400	600	800	1000
FFA	36	38	37	36	36.5
ACS-VMC	22	26	27	30.5	30
BMH	22.5	23.5	27.5	31	31
VRDI	20	20.5	23.9	27.8	30.5
RMCM	12.24	172	23.784	28.024	30.096

Compared with existing algorithms, Table.1 shows the proposed RMCM algorithm consumes less energy. Secondly, this work compares the three algorithms with the proposed RMCM algorithm based on a quantity of moved VMs. The procedure of movement of virtual machines would consume power which might affect a QoS of cloud appliances. Therefore, it is necessary to decrease the quantity of VMs to migrate. The Table.1 shows the results of VM migration count for the three algorithms with the proposed RMCM algorithm.

Table.2. Comparison of a quantity of VMs to moved

Algorithm	Number of VMs				
	200	400	600	800	1000
ACS-VMC	160	260	360	370	400
BMH	162	290	350	400	450
VRDI	163	300	390	450	440
RMCM	138	151	310	254	193

Compared with existing algorithms, Table.2 shows the proposed RMCM algorithm migrate a minimum number of VMs. So, it saves a lot of energy in datacenter. Further, Table.3 shows the number of PMs which shut down while utilizing the ACS-VMC [9], BMH [10], and VRDI with proposed RMCM algorithm.

Table.3. Comparison of an amount of shut downed PMs

Algorithm	Number of VMs				
	200	400	600	800	1000
ACS-VMC	90	79	62	59	39
BMH	90	70	60	51	42
VRDI	90	84	71	63	52
RMCM	94	90	73	72	59

The Fig.5 shows a number of PMs which shut downed while based on the ACS-VMC [9], BMH [10], and VRDI with proposed RMCM algorithm. As stated above, a reason for resource incorporation of a datacenter is to transmit VMs to shut down a few PMs that contain low usage to enhance the power competence of a datacenter. Thus, a lot of shuts downed PMs, a lot of is an efficiency of the algorithm.

Compared with existing algorithms, Table.3 shows the proposed RMCM algorithm closed more the number of PMs. So, it saves a lot of energy in datacenter. To check the efficiency of a presented optimal cost-effective technique (OCET) algorithm, and execute the massive amount of repetitive experiments, java used for simulation. This section shows OCET minimum cost CSP selection, VM migration and PM turn off results. This experiment takes three cloud service providers, and each has 100 physical machines and 200 virtual machines. After applied the OCET algorithm, results noted. The Table.4 shows the total no of virtual machines migrated results.

Table.4. Total No of VMs migrated results

CSP ID	VMs to be migrated
CSP1	125
CSP2	145
CSP3	131

The Table.4 concludes CSP 1 migrate few numbers of VMs compared with others. So, it may be consuming less energy. Further, Table.5 shows total no of hosts powered off results.

Table.5. Total No of Hosts Powered Off results

CSP ID	Hosts Powered Off
CSP1	94
CSP2	94
CSP3	90

The Table.5 concludes CSP 1 powered off a lot of Hosts compared with others. The Table.6 shows energy consumption results. The Table.6 concludes CSP 1 consumes less energy compared with others. The Table.7 shows the replication and migration cost of three CSPs.

Table.6. Energy Consumption results

CSP Id	Energy Consumption
CSP1	10.2 kWh
CSP2	10.68 kWh
CSP3	10.344 kWh

Table.7. Replication and Migration Cost of three CSPs

CSP Id	Replication and Migration Cost
CSP1	5100 Rs
CSP2	5340 Rs
CSP3	5172 Rs

The Table.7 concludes CSP 1 consumes minimum replication and migration cost compared with others. Therefore, this experiment results suggest based on these performance metrics, Cloud Service Provider 1 is the best CSP compared with others.

6. CONCLUSION

To reduce the price of data placement for appliances and time changeable workloads, developers should optimally use the cost variation among network with storage services across numerous providers. This paper proposed two algorithms to achieve this goal. The First algorithm is Replication and Migration Cost Minimization (RMCM) algorithm which reduces the costs and power usage of the datacenter through combining virtual resources. The second algorithm is the Optimal Cost-Effective Technique (OCET) algorithm for duplication and movement of information in cloud data centres with multiple storage classes. This algorithm objectives at attaining price deduction on a load allotment procedure on the multi-datacenter situation, where VMs allocated to the specified datacenter through regard as both power price differences with the existence of local renewable power manufacture, to decrease a power receipt.

The experimental outcomes demonstrated which a proposed RMCM and OCET algorithm contain the significant benefit based on decreasing the costs and power usage of the datacenter. Thus, a proposed RMCM and OCET algorithm is helpful on a building of the green data centre. The results showed which a presented RMCM and OCET algorithm reduced a replication and migration cost and decreased an energy usage of a datacenter in green cloud computing efficiently.

REFERENCES

[1] S. Muralidhar, "F4: Facebook's Warm Blob Storage System", *Proceedings of International Symposium on Operating Systems Design and Implementation*, pp. 383-398, 2014.

[2] G. Skourletopoulos., "An Evaluation of Cloud-Based Mobile Services with Limited Capacity: A Linear Approach", *Soft Computing*, Vol. 34, No. 2, pp. 1-8, 2016.

[3] A. Bourdena, "Using Socio-Spatial Context in Mobile Cloud Offload Process for Energy Conservation in Wireless Devices", *IEEE Transactions on Cloud Computing*, Vol. 32, No. 9, pp. 1-9, 2016.

[4] A. Kathpal, "Analyzing Compute vs Storage Tradeoff for Video-Aware Storage Efficiency", *Proceedings of 4th USENIX Conference on Hot Topics in Storage and File Systems*, pp. 1-13, 2012.

[5] D. Bermbach, "Meta Storage: A Federated Cloud Storage System to Manage Consistency-Latency Tradeoffs", *Proceedings of International Conference on Cloud*, pp. 452-459, 2011.

[6] K. P. Puttaswamy, "Frugal Storage for Cloud File Systems", *Proceedings of ACM European Conference on Computer Systems*, pp. 71-84, 2012.

[7] Z. Wu, "Spanstore: Cost-Effective Geo-Replicated Storage Spanning Multiple Cloud Services", *Proceedings of 24th ACM Symposium on Operating Systems Principles*, pp. 292-308, 2013.

[8] Y. Wu, "Scaling Social Media Applications into Geo-Distributed Clouds", *IEEE/ACM Transactions on Networking*, Vol. 23, No. 3, pp. 689-702, 2017.

[9] F. Farahnakian and H. Tenhunen, "Using Ant Colony System to Consolidate VMs for Green Cloud Computing", *IEEE Transactions on Services Computing*, Vol. 8, No. 2, pp. 187-198, 2015.

[10] S. Sohrabi, A. Tang, I. Moser and A. Aleti, "Adaptive Virtual Machine Migration Mechanism for Energy Efficiency", *Proceedings of International Conference on Green Sustainable Software*, pp. 8-14, 2016.

[11] Yingyou Wen, Zhi Li, Shuyuan Jin, Chuan Lin and Zheng Liu, "Energy-Efficient Virtual Resource Dynamic Integration Method in Cloud Computing", *IEEE Access*, Vol. 5, pp. 1-18, 2017.

[12] J. Li, M. Qiu, J. W. Niu, Y. Chen and Z. Ming, "Adaptive Resource Allocation for Pre-Emptable Jobs in Cloud Systems", *Proceedings of International Conference on Intelligent System Design and Application*, pp. 31-36, 2011.

[13] Y.O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti and Y. Coady, "Dynamic Resource Allocation based on Distributed Multiple Criteria Decisions in Computing Cloud", *Proceedings of International Conference on Cloud Computing*, pp. 91-98, 2010.

[14] Sahar Hosseinzadeh and M.S. Shirvani, "Optimizing Energy Consumption in Clouds by using Genetic Algorithm", *Journal of Multidisciplinary Engineering Science and Technology*, Vol. 2, No. 6, pp. 1431-1434, 2015.

[15] D.M. Quan, F. Mezza, D. Sannenli and R. Giafreda, "T-Alloc: A Practical Energy-Efficient Resource Allocation Algorithm for Traditional Data Centres", *Future Generation Computer Systems*, Vol. 28, No. 5, pp. 791-800, 2012.

[16] B. Anton, J. Abawajy and R. Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centres for Cloud Computing", *Future Generation Computer Systems*, Vol. 28, No. 5, pp. 755-768, 2012.

[17] N. Bobroff, A. Kochut and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations", *Proceedings of International Conference on Integrated Network Management*, pp. 119-128, 2007.

[18] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose and R. Buyya, "CloudSim: A Toolkit for Modelling and Simulation of cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", *Software: Practice and Experience*, Vol. 41, No. 1, pp. 23-50, 2011.

[19] K.M. Baalamurugan and S.V. Bhanu, "Analysis of Cloud Storage Issues in Distributed Cloud Data Centres by Parameter Improved Particle Swarm Optimization (PIPSO) Algorithm", *International Journal on Future Revolution in Computer Science and Communication Engineering*, Vol. 4, pp. 303-307, 2018.

[20] Y. Mansouri, A.N. Toosi and R. Buyya, "Cost Optimization for Dynamic Replication and Migration of Data in Cloud Data Centres", *IEEE Transactions on Cloud Computing*, Vol. 7, No. 3, pp. 705-718, 2016.