**CHAPTER 3**

**PROPOSED SYSTEM**

**3.1 OVERVIEW OF PROPOSED SYSTEM**

Originally, load scheduling and power management [9] was considered for geographically distributed datacenters with integration of renewables in the datacenter. The main aim of this approach is to reduce the eco-aware operating power cost of green cloud datacenters. This approach uses lyapunov optimization [21] for optimal scheduling and power management while ensuring user Quality of Service (QoS). But this approach does not consider the problem of task migrations [4] that increase the operating power of datacenter. We propose a modified bacterial foraging optimization based load balancing algorithm to reduce the number of task migrations and balance loads across the hosts in the system.

The following are proposed to overcome the limitations of the existing system:

* Introduction of task migrations from overloaded to underloaded servers.
* A new modified bacterial foraging optimization algorithm based load balancing approach.
* Estimation of number of requests per time slot to reduce the operating cost.

The architecture of the proposed system is depicted in fig 3.1. Here, the user requests are generated from the user region or user base which is sent to the Datacenter Broker. The Datacenter Broker forwards the user requests to the VM Load balancer which works on the proposed algorithm based on Modified Bacterial Foraging Optimization Algorithm. The VM Load balancer balances schedules the requests in the datacenters and periodically balances loads across the datacenters. The datacenters run on grid energy, solar energy or wind energy which is selected using Energy Selector.

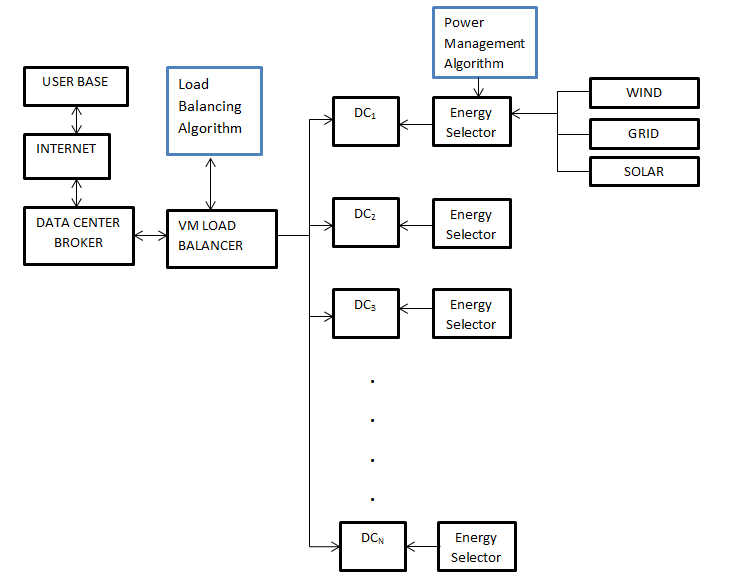


Figure 3.1 Architecture of the proposed system

**3.2 DATASETS USED**

The proposed algorithm takes the following inputs:

* Real time user request traces
* Wind energy dataset
* Solar energy dataset
* Grid energy dataset
* Cost of energy dataset

Figure 3.2 Trace File

The workload for simulation [32] is based on the real time user request traces taken from PARALLEL WORKLOADS archive , a repository that holds information and workloads from parallel machines and is based on Los Alamos National Laboratory(LANL) CM-5 log file .The trace file contains cloudlet requests with their submit times , cloudlet sizes , requested memory , number of PEs ,etc.,. The submit times of the cloudlets are scaled to fit the simulation.

Figure 3.3 Wind Energy Dataset

The wind energy dataset [34] consists of five different measurements of wind speed at different distances which is used to calculate the average wind speed measure and the amount of wind energy generated. The wind speeds are measured in m/s and wind energy generated is measured in KWh.

Figure 3.4 Solar Energy Dataset

The solar energy dataset [33] tabulates the solar irradiance per time slot in W/m2.This is used to determine the amount of solar energy generated in KWh.

Figure 3.5 Grid Energy Dataset

The grid energy dataset [35] consists of amount of grid energy supplied to each datacenter in each time slot measured in KWh.

Figure 3.6 Energy Cost Dataset

The energy cost dataset [36], obtained from Energy Information Administration, USA , consists of energy cost per unit time slot , measured in Dollars/time slot.

**3.3 PROPOSED METRICS**

The following metrics are defined for developing the algorithm:

**Eco Factor :**

Eco factor refers to the green degree or the level of environmental impact caused by the type of energy used. Eco factor gives a much more broader definition for environmental impact than carbon footprint. Generally , the eco factor of brown energy (fossil fuels) is high considering its environmental impact. The eco factor of renewable energy(green energy) is typically low. The eco factor is denoted by the symbol **ε** with energy type in superscript. The algorithm depends more on the relative relationship of the eco factors of different types of energies than on its exact values. Hence the eco factor for renewable energy follow the uniform distribution **εren** **∼** U(0,1) and the eco factor for brown energy follow the uniform distribution **εgrid** **∼** U(8,9)

**Eco aware power cost :**

Eco aware power cost is defined as the product of the eco factor of the energy type and its corresponding power cost .Since eco factor is relative , eco aware power cost can be used to determine the actual measure of environmental impact of the type of energy. It is denoted by **Cj(t)** where **C** is the eco aware power cost of **jth** datacenter in time slot **t**.

**Solar energy conversion efficiency :**

Solar energy conversion efficiency refers to the efficiency at which solar irradiance is converted to electricity. It is used to denote the amount of energy lost while converting solar energy to electricity. It is denoted by **α** and is generally between **15-20%**

**Wind energy conversion efficiency :**

Wind energy conversion efficiency refers to the efficiency at which energy of wind speeds is converted to electricity. It is used to denote the amount of energy lost while converting wind energy to electricity. It is denoted by **β** and is generally between **20-30%**

**3.4 PROPOSED FACTORS**

The following factors have been considered for the algorithm:

* Latency
* Energy constraints (solar, wind, grid energy)
* VM load
* VM capacity
* Cost per type of energy
* CPU utilization
* Location or Propagation delay

**3.5 SYSTEM MODULES**

**3.5.1 LOAD SCHEDULING MODULES**

Load scheduling is used to route requests to specific datacenters based on the scheduling strategies derived. This is applied to workloads generated across all regions. The total workload generated across all user regions is denoted by

**λ(t) = {λij(t), i = 1, . . . , M; j = 1, . . . , N}**

where ,

**λ(t)** – workload at time slot **t**

**λij(t)** –workload generated at time slot **t** in **jth** user region and routed to datacenter **i**

**CHEAP FIRST LOAD SCHEDULING**

The aim of cheap first load scheduling strategy is to minimize the total cost of drawing electricity from power grid. This implies that user requests are served first by datacenters with the cheapest electricity price.

**λ(t) ε min(Cj(t)) ∀ j=1 to N**

User requests in time slot **t** are routed to the datacenter with the cheapest electricity price.

**QoS FIRST LOAD SCHEDULING**

The aim of QoS first load scheduling is to minimize the latency involved in routing / scheduling requests to datacenters. The latency experienced by a user request must be bounded to ensure good user QoS. The strategy ensures that all user requests generated should be serviced without exceeding the maximum tolerable latency threshold. All physical servers within a datacenter are assumed to be homogeneous and hence the maximum tolerable latency is defined as a constant for all the servers. The service latency function is bounded as,

**g(dij, λij(t))<=Lj**

where,

**g(.)** – service latency function

**dij –** geographical distance between **DCj** and user region **i**

**λij(t) –** workload sent to **DCj** from user region **i**

**Lj** – maximum tolerable latency when requesting **DCj**

The QoS factors considered are :

* Latency
* Propagation delay
* Bandwidth / bit rate

**3.5.2 POWER MANAGEMENT MODULES**

The aim of power management modules is to dynamically manage power supplies to the datacenter and to automatically make switches between different sources based on preset thresholds. This implies that when the power supply for a particular type of energy falls below the threshold i.e. the minimum operating power of a datacenter, the datacenter switches to energy supplied from the grid for uninterrupted operation.

The power management strategies are used to dynamically determine the following:

* The amount of grid energy in time slot **t**

**Egrid(t)={Ejgrid (t), j=1,2….N}**

Where,

**Egrid(t) –** total grid energy supplied to all datacenters in time slot **t**

**Ejgrid (t) –** grid energy supplied to **DCj**

* The amount of renewable energy generated in **DCj** time slot **t**

**Ejren(t)={Ejwind (t) + Ejsolar (t), j=1,2….N}**

Where,

**Ejwind(t) –** wind energy generated in **DCj** in time slot **t**

**Ejgrid (t) –** solar energy generated in **DCj** in time slot **t**

**Esolar(t) = α . A . s(t) . l**

**Ewind(t) = β . (1/2) . B . ρ . v3(t) . l**

Where ,

**α –** solar to electricity conversion efficiency

**β –** wind to electricity conversion efficiency

**A –** total active irradiation area of all solar panels

**B –** total rotor area of all wind turbines

**s(t) –** solar irradiance in time slot **t**

**v3(t) –** wind speed

**l –**length of time slot **t**

**ρ –** air density

**POWER MANAGEMENT PLAN A**

The power management plan A considers both renewables (solar, wind) and grid energy as power sources to the datacenter. Here, the different power sources automatically make switches based on preset thresholds. Renewable energy is given the highest priority and if it falls below the minimum required operating power of the datacenter, then datacenters use energy from the power grid. Renewable energy cannot be used exclusively because of its intermittent generation.

**Ej(t) = Ejren(t) if Ejren(t) >=T**

**Ejgrid(t) if Ejren(t) <T**

Where,

**Ej(t) –** Energy used by **DCj** in time slot **t**

**Ejren(t) –** Renewable energy generated in **DCj** in time slot **t**

**Ejgrid(t) –** Grid energy generated in **DCj** in time slot **t**

**T –** Preset threshold derived from operating power of datacenter

**GRID BASED POWER MANAGEMENT PLAN**

The grid based power management plan exclusively uses energy from the power grid. This implies that datacenters typically rely on energy supplied from the power grid for operation. Hence this power plan can result in increased time average eco aware power cost since the eco factor or green degree of grid energy is high.

**Ej(t) = Ejgrid(t)**

**3.5.3 STRATEGIES USING LOAD SCHEDULING AND POWER MANAGEMENT PLANS**

The load scheduling and power management plans are combined to create four different strategies for comparing the time average eco aware power cost of the datacenters while serving user requests. The four different strategies in comparison are:

* Grid based power plan with Cheap first scheduling
* Power Plan A with Cheap first scheduling
* Grid based power plan with QoS first scheduling
* Power Plan A with QoS first scheduling

**LYAPUNOV OPTIMIZATION FUNCTION DEFINITION**

The eco aware power cost of datacenter at time slot **t** is given by,

**Cj(t)= εgrid . Pj(t).Ejgrid(t) +Ejren(t) . εren**

where ,

**εgrid –** eco factor of grid energy

**Pj(t) –** power cost of grid energy in time slot **t** in region of **DCj**

**Ejgrid(t) –** grid energy used by **DCj**  in time slot **t**

**Ejren(t) -** renewable energy generated in **DCj**  in time slot **t**

**εren** – eco factor of renewable energy

The aim is to reduce the eco aware power cost of a datacenter while still maintaining user QoS to ensure reduction in total energy consumption. Hence this problem is modeled as a stochastic optimization problem.

Problem **P1** : **min**

Constraints : **0<= Ejgrid(t) <= Ej,maxgrid(t)**

**0<= Ejsolar(t) <= Ej,maxsolar(t)**

**0<= Ejwind(t) <= Ej,maxwind(t)**

**Wi(t) = λj(t) =**

**g(dij, λij(t))<=Lj**

where,

**Wi(t)** – workload generated in **ith** user region in time slot **t**

**λj(t) –** workload dispatched to **DCj** in time slot **t**

The problem P1 can be solved using dynamic programming but the computational complexity is high. With the inclusion of many factors, dynamic programming results in higher order polynomial time complexity which results in serious overload on cloud platforms causing service disruptions and SLA violations. Hence the problem P1 can be solved using Lyapunov optimization framework to design an online algorithm which only depends on the current system state. The algorithm can obtain sub-optimality with provable performance bounds.

Lyapunov optimization [31] requires defining a virtual queue vector which is used to satisfy the linking constraints in the problem. The virtual queue vector is represented as a shifted version of actual energy level in the datacenter in time slot t,

**Hj(t)=Ejsolar(t)+Ejwind(t)-V εgridPj,max –E j,minsolar-Ej,minwind**

Where,

**E j,minsolar -** minimum solar energy generated in **DCj**

**Ej,minwind -** minimum wind energy generated in **DCj**

**Pj,max –** maximum electricity price at **DCj**

**V –**tunable parameter between [0,1]

The virtual queue update is given by,

**Hj(t+1)= Hj(t) – Ej(t)**

Where,

**Ej(t) -** Energy used by **DCj** in time slot **t**

The one slot lyapunov function and its corresponding lyapunov drift is given as,

**L(H(t)) ≜ 0.5 [H12 + H22+…+ HN2]**

**Δ(H(t)) ≜ E[L(H(t+1))-L(H(t))|H(t)]**

Lyapunov drift is a variation of virtual queues and it is upper bounded. This can be used to convert problem P1 into problem P2.

Problem **P2** : **min λ(t) Egrid(t)**

The problem P2 can be solved to minimize the time average eco aware power cost of the system.

**3.5.4 LYAPUNOV OPTIMIZATION BASED LOAD SCHEDULING AND POWER MANAGEMENT**

Algorithm : **Eco Aware Online Power Algorithm**

Input : **Egrid(t) ,Esolar(t),Ewind(t),V, g(.), L , εgrid , εren ,P(t) , W(t)**

Output : **Power management and Load scheduling strategies**

Eco Aware Online Power Algorithm :

1. Start
2. T->0
3. While service is active , do,
   1. Simplify P2 based on H(t)
   2. Calculate load scheduling strategy λ(t)for each user region by solving P2 using linear programming
   3. Derive power management strategy for the datacenter based on workload
   4. Schedule workload based on power management and load scheduling strategies
   5. Update virtual queue of each datacenter
   6. T->T+1
4. Stop

**MODIFIED BACTERIAL FORAGING OPTIMIZATION**

Bacterial foraging optimization algorithm [29] mimics the foraging or chemotactic behavior of bacteria in virtual search space. The algorithm has three stages:

* Chemotaxis
  + Tumble step
  + Swim step
* Reproduction
* Elimination Dispersal

**CHEMOTAXIS STEP**

The chemotaxis step simulates the movement of an E.coli cell through swimming and tumbling. The bacterium can either swim in the same direction for a period of time in favorable condition or tumble in a different direction in unfavorable environment. The computational chemotaxis is represented by,

**θi(j+1,l) = θi(j,l)+C(i) \* (Δ(i)/ √( ΔT(i)\* Δ(i)))**

Where,

**θi(j+1,k,l)** – position of bacterium **i** in **j+1th** chemotactic step and **lth** elimination step.

**θi(j,k,l) -** position of bacterium **i** in **jth** chemotactic step and **lth** elimination step.

**C(i) –**step size

**Δ(i) –** random unit direction vector [-1,1]

Here, at each new chemotactic step, the bacterium or cloudlet tumbles to a new underloaded VM and its cost is evaluated with respect to its position among other bacteria. If the bacterium’s cost at the new position is lower than its previous cost, it swims, i.e. it tries other underloaded VMs in the same datacenter. The cost of the bacterium is evaluated as the sum of its cost at that position and its attraction and repulsion with other bacteria.

**ELIMINATION DISPERSAL STEP**

Elimination dispersal step is used to simulate the accidental changes that happen in the environment in which the bacterium forages. This can cause certain bacteria to be displaced to a new environment or position. Elimination dispersal step can destroy chemotactic progress but can help to achieve global optimum without being stuck in local optima. Here, each bacteria / cloudlet is randomly assigned a probability Ped, which is evaluated to determine if the bacterium should be dispersed to a new location or not.

**θi(0,l+1) = θi(j,l)+ (Δ(i)/ √( ΔT(i)\* Δ(i)))** if **P(j,l)>=Ped**

**θi(j,l)** if **P(j,l)<Ped**

**CAPACITY OF VM**

CVM=PEVM \* MIPSVM + BWVM

Where,

PEVM – Number of Processing Elements allocated to VM

MIPSVM - Millions of Instructions per second of all PEs for VM

BWVM – Bandwidth allocated to VM

**LOAD ON VM**

LVM = N(t)/S(t)

Where,

N(t) – Number of tasks in time t in VM

S(t) –Service rate of VM at time t

**COST FUNCTION**

The cost function J for modified BFO algorithm is defined as ,

**Ji(j,l)=Ji(j,l)+Jcc(Δi(j,l) , Pi(j,l))**

Where,

**Ji(j,l)** - cost of the ith bacterium in the jth chemotactic and lth elimination dispersal steps.

**Jcc(Δi(j,l) , Pi(j,l)) -**  additional cost due to the attraction and repulsion between different bacteria in the same position.

Here the cost function is defined as ,

**Ji(j,l)=LiVM+UiVM-CiVM-(Eisolar(t)+Eiwind(t)- V εigridPi,max –E i,minsolar-Ei,minwind)**

Where,

**LiVM  -** Load on VM (under loaded) where the bacteria (cloudlet) is currently present

**UiVM  -** CPU utilization at VM

**CiVM  -** Capacity of VM

Constraints : **0<= Ejgrid(t) <= Ej,maxgrid(t)**

**0<= Ejsolar(t) <= Ej,maxsolar(t)**

**0<= Ejwind(t) <= Ej,maxwind(t)**

**Wi(t) = λj(t) =**

**g(dij, λij(t))<=Lj**

**3.5.5 MODIFIED BACTERIAL FORAGING OPTIMIZATION BASED LOAD BALANCING ALGORITHM**

In order to achieve load balancing across VMs in the system , we use modified BFOA at each time slot to migrate cloudlets from overloaded to underloaded VMs. Cloudlets from each overloaded VMs are considered as bacteria and forage/ search for underloaded VMs which are considered as food sources.

Since BFOA deals with foraging / nutrient search of bacteria over a virtual search space, the algorithm can be used to identify optimal VM among the underloaded VMs to migrate the cloudlet. The original BFO algorithm is modified to remove the reproduction loop since this step causes only the healthy half of the bacteria i.e. a certain number of cloudlets from overloaded VMs, to be reproduced causing the rest of the cloudlets to be eliminated. Since we need to consider the position of a cloudlet with respect to all other cloudlets, we remove the use of reproduction loop. Further, this leads to reduce in overhead due to the elimination of reproduction loop.

Algorithm : **Modified BFOA based Load balancing Algorithm**

Input : **Egrid(t) ,Esolar(t),Ewind(t),V, g(.), L , εgrid , εren ,P(t) , W(t)**

Output : **Power management and Load balancing strategies**

Modified BFOA based Load balancing algorithm:

1. Start
2. While simulation not terminated , do at each time slot ,
   1. Compute standard deviation of loads across all VMs
   2. If σ<Ts
      1. System is Balanced ,return
   3. If total\_load >total\_capacity
      1. Enqueue tasks ,return
   4. Initialize colony of bacteria with cloudlets from overloaded VMs
   5. Compute initial costs
   6. While Ned >0 ,do
      1. While Nc >0 , do
         * Tumble to new VM
         * If variance(new costs) <variance(old costs)
           + While Ns >0 and variance reduces ,do

Try other VMs in same datacenter

Update colony

* + - * Continue
    1. Assign random Ped to all bacteria
    2. Randomly disperse bacteria based on Ped
  1. Migrate cloudlets to new positions (VMs) based on bacterial positions.

1. Stop