**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Greening Geographical Load Balancing**

**Liu et al.** [18]proposed a method for exploiting the geographical diversity of Internet-Scale systems to provide better power usage. This paper proposes that if the electricity is dynamically priced in proportion to the instantaneous fraction of the total energy that is brown, then geographical load balancing provides significant reductions in brown energy use. The paper also explores the paradox where reductions in energy costs can also be achieved as a result of increased energy usage since the user request can be routed to a datacenter with cheaper electricity price which is father from the source, thus the data center may need to complete the job faster, and so use more service capacity, and thus energy, than if the request was served closer to the source. The paper considers discrete-time model with the timeslot matching the timescale at which routing decisions and capacity provisioning decisions can be updated. The proposed model consists of J user regions and N geographically distributed datacenters, each datacenter consisting of Mi homogeneous servers. The proposed system assumes the systems for provisioning capacity of datacenters and adapting the routing of requests from user regions to datacenters. The model focuses on determining λij(t) which shows the amount of traffic or user request from ith user region which is forwarded to the jth geographically distributed datacenter. It also determines mi(t) which is the number of active servers in datacenter i . These decisions are used for optimizing and controlling the amount of electricity used and for reducing the electricity cost in time slot [1,T] . The paper also compares the performance of the proposed model against three baselines that combine network delays, switching delays and electricity prices.

**2.2 Geographical Load Balancing with Renewables**

**Liu** **et al.** [19] proposed another method for feasibly powering Internet Scale systems with almost renewable energy. The paper determines the feasibility of such a system by performing a trace based study of the issues namely the mix of renewables and the fossil energy, the role of storage and the impact of geographical load balancing. The paper proposes a model that would increase the capacity of renewables by using the energy more efficiently, thus leading to reduced carbon footprint. The paper also proposes a wind dominant, solar complementary system since the availability of wind based energy is high, and thus if aggregated over a range of time intervals, there is lesser variation in availability. The paper proposes an 80% wind and 20% solar portfolio for energy supply to a datacenter and also traces the value of storage when renewables are used. The capacity of renewables is measured as the ratio of the average renewable generation to the minimal energy required to serve the average workload. The paper assumes a mean arrival rate Lj(t) for datacenter j at time t. Real world traces are used to define Lj(t) for every datacenter j. The paper proposes that small scale storage devices can help to increase the capability of the renewable energy and thus shows that the batteries within a datacenter can be used for the storage of energy . Thus the energy cost for a datacenter is taken as the difference between the total energy cost and the energy used by active servers that can be powered by renewables. The delay cost captures the lost revenue incurred because of the delay experienced by the requests, where the delay includes both the network delay from source j to data center i, dij and the queuing delay at i , dij is the distance between source and data center, divided by the speed of 200 km/ms plus a constant (5 ms), resulting in delay ranging in [5 ms, 56 ms]. For large diurnal peak-to-mean ratios the optimal portfolio can be expected to use a higher percentage of solar. The paper compares the performance of the proposed system against another system named “LOCAL” which routes all the requests to the geographically closest datacenter and which adjusts the number of active servers in the region.

**2.3. Honey Bee inspired Load Balancing of Tasks in Cloud Computing Environments**

**Babu et al.** [4] proposed a new algorithm named honey bee behavior inspired load balancing (HBB-LB), which achieves well balanced load across virtual machines for maximizing the throughput. The proposed algorithm also balances the priorities of tasks on the machines in such a way that the amount of waiting time of the tasks in the queue is minimal. The paper considers the case of heterogeneous cloud computing servers and improves the load balancing strategy without introducing much overhead in the computing. The paper shows that the proposed algorithm was effective when compared with existing load balancing and scheduling algorithms. The paper also illustrates that there is a significant improvement in average execution time and reduction in waiting time of tasks on queue. The paper models the algorithm based on the foraging behavior of honey bees. The servers are grouped under virtual servers (VS), each VS having its own virtual service queues. The Virtual Machines are grouped into three categories namely UnderLoaded VMs, OverLoaded VMs , Balanced VMs . The removed tasks from over loaded VMs are considered as the honey bees. Upon submission to the under loaded VM, the task will update the number of various priority tasks and load of that particular VM to all other waiting tasks. This will be helpful for other tasks in choosing their virtual machine based on load and priorities. Whenever a high priority task has to be submitted to other VMs, it should consider the VM that has minimum number of high priority tasks so that the particular task will be executed at the earliest. This paper considers Makespan and Response Time for the allocation of requests to VMs. Makespan which is the overall task completion time where task completion time of task Ti on VM j is denoted as CTij. Current workload of all available VMs can be calculated based on the information received from the datacenter. Based on this, standard deviation and degree of imbalance are calculated to measure deviations of load on VMs.

**2.4. A Multi Stage Load Balancing Environment for Cloud Datacenters**

**Jain et al.** [14]proposed a two level load balancer approach by combining join idle queue and join shortest queue approach. The paper proposes an adaptive approach to tackle the underutilization and over utilization of resources across VMs. The algorithm balances the requests based on changes in system load. The paper considers the factors Response Time, Throughput, Cost, Scalability and Fault Tolerance for the adaptive load balancing strategy. The strategy combines two algorithms Join Idle Queue (JIQ) and Join Shortest Queue (JSQ) . JIQ approach makes use of each scheduler which maintains the list of idle virtual machines in its idle queue. The task is randomly forwarded to any scheduler and the scheduler assigns the task to any idle virtual machine. On the other hand, the JIQ scheduler forwards the task to the VM whose queue length is the smallest. It ensures that the task is completed with minimal response time. The adaptive approach combines the aspects of both the schedulers. The algorithm chooses the scheduler with the largest VM Idle queue and assigns the task to a particular VM. If the Idle queue of all schedulers are empty, then the VM with the smallest queue length is chosen and the task is assigned to be executed on that VM. The limitation with this paper is that it does not consider the case of task migrations during load balancing.

**2.5 Eco Aware Online Power Management and Load Scheduling for Green Cloud Datacenters**

**Deng et al.**[9]proposed an online power management and load scheduling algorithm to reduce the operating power cost of a datacenter. The online algorithm,EcoPower, provides eco-aware power management for geographically distributed green cloud datacenters. The algorithm achieves the objectives while still maintaining user QoS. The problem is modeled as a stochastic optimization problem which is constrained by the maximum and minimum values of the renewable energy generated. The algorithm uses Lyapunov optimization to determine the optimal datacenter to which the request must be routed and extensive simulations illustrate that the algorithm approaches optimality with provable performance bounds. The paper proposes a scenario where there are N geo-distributed datacenters {DCj, j =1, . . . , N} owned by one CSP, which provide cloud services to end users spreading over M regions {Ri, i = 1, . . . , M}. User requests can be routed to any datacenter via the intermediate load balancers. Each datacenter has multiple types of power supplies including Power grid, solar, wind, ESDs. The paper also proposes a new metric “Eco-Factor” which refers to the green degree of the energy in use and has a much broader definition that carbon footprint. For a time slot t, Wi(t) is the workload, i.e., the power load, generated by users in the ith region Ri. λj(t) is the workload dispatched to DCj and λij(t) as the workload generated by users in Ri but dispatched to DCj for service. The service latency function is defined as a nondecreasing convex function g(·), which satisfies g(dij, λij(t)) ≤ Lj , where dij is the geographical distance between Ri and DCj , and Lj is the maximum tolerable latency when requesting services from DCj . The workload dispatched to a datacenter is assumed to be completed timely to ensure service quality, i.e., no workload can be postponed to the next time slot. All the physical servers within a datacenter are assumed to be homogeneous, whose capacities and power consumption properties are the same. The paper proposes two schedulers namely Cheapest First Scheduler and QoS First Scheduler. The Cheapest First Scheduler routes the request to the datacenter with the scheduler. The objective of the scheduler is to reduce the operating cost involved in servicing of the request. The QoS First Scheduler routes the request to the datacenter where the latency involved in servicing of the request is minimum. The strategy guarantees that all requests are serviced within the current time slot. Further, the paper proposes two power plans namely A and B, where the different power sources can automatically make switches based on preset thresholds. Renewable energy is given the highest priority and when it falls below a threshold, the datacenters switch to drain energy from power grid. The different power management and load scheduling methods are combined to create different alternating scenarios. For the different scenarios, capacity of renewables and ESDs are properly provisioned according to the potential workload and electricity price so that the wastage can be minimized. The paper proposes that a wind dominant and solar complementary power portfolio is a better strategy for datacenters to integrate into their power supply. The limitation with this paper is that it does not handle the problem of task migrations, which are one of the main sources for power consumption across datacenters.

**2.6 Cluster based Load Balancing for Cloud Computing**

**Kapoor et al.** [15] proposed a cluster based load balancing strategy for partitioning loads across datacenters. The paper proses the algorithm for heterogeneous node environment considers resource specific demands of the tasks and reduces scanning overhead by dividing the machines into clusters .The paper proposes that the algorithm gives better results in terms of waiting time, execution time, turnaround time and throughput as compared to existing throttled and modified throttled algorithms. The dynamic load balancing strategy considers the factors transfer policy, selection policy, information policy and location policy. Transfer policy decides which job could be transferred from one node to another in order to balance the load. Selection policy aims at deciding which overloaded node will be chosen for conveying its load and which under loaded node will be chosen for receiving the jobs from overloaded node. Location policy will decide the destination to which an incoming task or job will be assigned. The information policy deals with handling of information about server nodes. The proposed algorithm adds clustering approach so as to divide VMs with similar capacities into groups. K-means clustering approach has been used to divide VMs into cluster. The load balancer will maintain a list of all the clusters with the minimum and maximum resource specific capacities of each cluster. This is range specifier list. Load balancer scans the range specifier list of k clusters to see that which cluster can handle the incoming request. Also the load balancer will maintain the list of VMS for each cluster. It reduces the overhead of scanning the entire list of VMs from the beginning.Euclidean distance formula is used to assign VMs to the clusters. The value of K i.e. the number of clusters has been chosen to be the highest prime factor of n where n is the number of VMs. When user submits his task, the load balancer matches task resource specific requirements with capacity range of cluster in order to assign the task to appropriate cluster. Load balancer will then assign the request to the appropriate VM of the chosen cluster by looking into the list of cluster members which will match the specific demands of the task and whose status is available. In case more than one VMs satisfy this, then the first one which is found will get the task. Thus scanning of list gets divided into two levels. This will reduce the time involved in list scanning and also will assign a better and more suitable VM to the task as per its requirements.

**2.7 Efficient and Scalable ACO based Task Scheduling for Green Cloud Computing Environment**

**Ari et al.** [3] proposed a biological inspired scheduling system, based on ant colony optimization to reduce the make span time and achieve better load balancing for green computing. The paper focuses on user level task scheduling in cloud environment and introduces a new algorithm called CACO (Cloud Ant Colony Optimization) that uses the features of ACO to obtain optimality. It exploits the features of Max-Min Ant System (MMAS) to further optimize task scheduling. MMAS prevents the stagnation of search to obtain a rapid convergence time and low computing cost. Early stagnation of search is avoided by setting the pheromone minimum value and maximum value. A virtual machine is removed from the available resources list when the pheromone value is less than minimum value. There are three main steps in the algorithm which are initialization step, construction step, updation of pheromone trial step. In the initialization step number of parameters such as virtual machine parameters, number of ants, number of iterations are initialized. In the construction step a colony of ants are engaged in the construction of a solution independently by pheromone trials and heuristic matrix which is used to generate an array of jobs. In the updating pheromone trial step the pheromone trials are updated according to the best and solution. Based on the simulation results, average relative standard deviation of the proposed CACO increases when the job size increases against the standard deviation of the ACO solutions. The main limitation of the paper is that the GA solution results in high makespan time and dispersion in results which is due to the genome encoding scheme.

**2.8 Load Balancing in cloud Computing based on Improved Particle Swarm Optimization**

**Patel et al.** [25] proposed an improved particle algorithm to achieve resource based load balancing optimization in the cloud environment. Since Load balancing is a discrete space optimization problem, accordingly PSO (Particle Swarm Optimization ) is improved as IPSO(Improved PSO) by adjusting the definition of particle position, velocity and rules for updating, modifying its fitness value. Based on simulation results on IPSO against Max Min algorithm and IABC algorithm, the makespan of IPSO is short and best of stability. Considering the performance of makespan and convergence IPSO provided better results and capable of scheduling resources effectively and shortens the task completion time. The main limitation of the paper occurs under different population size in which algorithm hasn’t been analyzed and compared.

**2.9 Energy-aware Scheduling for Infrastructure Clouds**

**Knauth et al.** [16] proposed a system to quantify the difference in energy consumption caused by virtual machine schedulers. The energy saving potential of virtual scheduler is quantified by varying parameters such as data center size, request distribution, heterogeneous hosts and run time distributions. Multiple algorithms are compared for incorporating virtual machine lease time information. The paper proposes “OptSched” an optimal scheduler to reduce the cumulative machine uptime which is evaluated using data center composition, run time distribution, virtual machine sizes. The main limitation of the paper is processes are assumed to be interruptible and threads can’t be migrated for free cost also unacceptable service unavailability occurs due to migration of virtual machine instances between physical servers.

**2.10 A Novel Approach Towards Improving Performance of Load Balancing Using Genetic Algorithm in Cloud Computing**

**Pilavare et al.** [26] proposed a Logarithmic Least Square matrix algorithm to improve the efficiency of GA which helps to resolve the problem of being idle and starving and to minimize the make span of given task sets. Natural Selection Strategy increases the performance and effective load balancing in Genetic Algorithm. Logarithmic Least Square Matrix technique helps in calculation of priorities in all processors and executes the higher priority jobs. Genetic Algorithm outperforms existing load balancing techniques, giving prioritized input to the genetic algorithm increases the response time. This paper proposes that any crossover and mutation techniques can be modified for better results. The main limitation of the paper is the virtual machine having low fitness value is left that results in starvation and low fitness virtual machine goes idle and no usage.