International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 5 159 – 165

Proposing Optimus Scheduler Algorithm for Virtual Machine Placement Within a Data Center

Dr. M. Sughasiny
Assistant Professor, Department of Computer Science
SrimadAndavan Arts and Science College (Autonomous)
Trichy, Tamilnadu, India
sughasiny5.cs@gmail.com

M. Hari Prakash
Research Scholar, Department of Computer Science
SrimadAndavan Arts and Science College (Autonomous)
Trichy, Tamilnadu, India
line 4: e-mail: name@xyz.com

Abstract—With the evolution of the Internet, we are witnessing the birth of an increasing number of applications that rely on the network; what was previously executed on the user's computers as stand-alone programs has been redesigned to be executed on servers with permanent connections to the Internet, making the information available from any device that has network access. Instead of buying a copy of a program, users can now pay to obtain access to it through the network, which is one of the models of cloud computing, Software as a Service (SaaS). The continuous growth of Internet bandwidth has also given rise to new multimedia applications, such as social networks and video over the Internet; and to complete this new paradigm, mobile platforms provide the ubiquity of information that allows people to stay connected. Service providers may own servers and data centers or, alternatively, may contract infrastructure providers that use economies of scale to offer access to servers as a service in the cloud computing model, i.e., Infrastructure as a Service (IaaS). As users become more dependent on cloud services and mobile platforms increase the ubiquity of the cloud, the quality of service becomes increasingly important. A fundamental metric that defines the quality of service is the delay of the information as it travels between the user computers and the servers, and between the servers themselves. Along with the quality of service and the costs, the energy consumption and the CO2 emissions are fundamental considerations in regard to planning cloud computing networks. In this research work, an Optimus Scheduler algorithm to be proposed for Add, Remove or Resize an application by using Tabu Search Algorithm.

Keywords-*Optimus Scheduler, Tabu Search Heuristic Algorithm, Virtual Machine, Data Center, First Fit*

*****

I. INTRODUCTION

The algorithm proposed in this chapter is characterized as online because application requests are received consecutively, each one at a different time. For each application request, the decision to make is which server will host each requested VM for that application. After the decision is made, the hypervisors of the target servers create the VMs in real time. The objective is to minimize delay among VMs, maximize the network throughput available among VMs, and minimize the server energy consumption. The application elasticity is also taken into account, that is, the workload is expected to increase and decrease over time. Our placement strategy can be characterized as:

1. based on application graph,
2. communication-aware,
3. energy-efficient,
4. and elastic.

In what follows, we explain each of these points

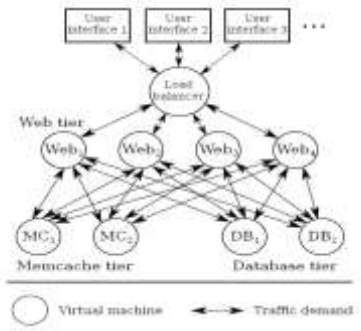

Figure 1: Graph of a multi-tier application.

II. RELATED WORKS

VM placement problem is a non-deterministic problem. Number of virtual machine placement algorithms has been studied in [1] that run under cloud computing environment. This section explains some of the exiting virtual machine placement approaches and their anomalies.

First Fit: It is a greedy approach. In this approach scheduler visits the PM sequentially one by one and placed VM to first PM that has enough resources. Each time when the new VM arrived, scheduler starts searching the PM sequentially in the data center till it finds the first PM that has enough resources. If none of the physical machine satisfied the resource requirement of the VM, then new PM is activated and assigns VM to the newly activated PM. Main problem with this approach is that load on the system can imbalanced.

Single Dimensional Best Fit: This methods use the single dimension (CPU, memory, bandwidth etc.) for placing a VM. When VM arrived, scheduler visit the Physical Machines in the decreasing order of their capacity used in a single dimension and place the VM to the first PM that has the enough resources. That means VM place to the PM which used the maximum capacity along with the given dimension. Problem with approach is that it can increase the resource imbalancing because resource in the cloud is multi-dimension (CPU, memory, bandwidth etc.). So there may be a situation where a host utilize their full CPU capacity while other resources such as memory and bandwidth are underutilized.

Volume Based Best Fit: This heuristic used the volume of the VM for placing a VM. This approach visits the Physical Machines in a decreasing order of their volume and place VM

to the first PM that has the enough resources. That means Physical Machine which has the maximum volume will be considered first.

Dot Product Based Fit: In this heuristic resource requirement of virtual machine and the total capacity of the physical machine along the specified dimensions are expressed as vectors. Dot product of these two vectors is calculated and then PMs are arranged into the decreasing order of their dot product. For the proper utilization of the resources it is necessary that the virtual machine which required more CPU and less memory should be placed on the physical machine which has low CPU and more memory utilization. This method seems good, but it can choose the wrong PM, because they the not using the length of virtual machine and the remaining capacity of the physical machine.

Constraint based approach: Constraint based approaches are use in the combinatorial search problems [2]. In these approach some constraint are apply and these constraint must be fulfil during VM placement. These constraints are Capacity Constraints: For all dimension (CPU, memory and bandwidth) of a given physical machine, sum of the resource utilize by all VMs running in that host should be less than or equal to the total available capacity of that physical machine. Placement Constraints: All virtual machines must be placed on to the available host. SLA constraint: VM should be placed to the PM where it fulfils the SLA. Quality of services (QoS) constraint: Some quality of services constraint such as throughput, availability etc. must be considered during the VM placement. Constraint Programming is useful where the input data is already known. That means we know the demands of the VMs, before calculating the cost function.

Bin packing problem: Bin packing problem [3] is a NP hard problem. If PM and the VM are consider as a three dimension object, then VM placement problem is similar to the bin packing problem where item represent the VM and container represent the PM. In the bin packing problem number of item (VM) are placed inside a large container (PM). The aim is to places a number of items into a single container as possible. So that the number of container required to packing the item is minimized. Bin packing problem is different from the VM placement problem. In the bin packing problem bins can be placed side by side or one on top of the other. But in the case of VM placement, placing VMs side by side or one on top of the other is not a valid operation. This is because the resource cannot be reused by any other VM, once a resource is utilized by a VM.

Stochastic Integer Programming: Stochastic Integer Programming is used to optimize the problems, which involve some unknown data. VM placement problem can be consider as a Stochastic Integer Programming because resource demand of the VM are known or it can be estimated and the objective is to find the suitable host which consume less energy and minimize the resources wastage. Stochastic Integer Programming can be use where the future demands of the resources are not known, but their probability distributions are known or it can be calculated.

Genetic Algorithm: Genetic algorithm is a global search heuristics. It is useful where objective functions changed dynamically. This approach is inspired by the evolutionary biology such as inheritance. Genetic Algorithm can be use to solve the bin packing problem with some constraints. It is useful for the static VM placements, where the resource demands do not vary during a predefine period of time.

Ahmad NaharQuttoum et al. [4] presented a Smart Placement Approach (SPA) that provides for smart placement maps of VDNs over CDNs. In this, author pointed out that choosing the placement maps for such VDNs should satisfy its requirements while: maintaining load-balancing over the hosting CDNs, guaranteeing its Quality of Service (QoS) levels, and assuring low placement costs. The VDNs are usually hosted over physical networks; overlaying its resources to gain the dynamic required services. Cloud-service Datacenter Networks (CDNs) can provide such a service under a delivery model that is called Infrastructure as a Service (IaaS).

CHONGLIN GU et al. [5] proposed a tree regression-based method to accurately measure the power consumption of VMs on the same host. The merits of this method are that the tree structure will split the data set into partitions, and each is an easy-modeling subset. Cloud computing is developing so fast that more and more data centers have been built every year. This naturally leads to high-power consumption. Virtual machine (VM) consolidation is the most popular solution based on resource utilization. In fact, much more power can be saved if we know the power consumption of each VM. Therefore, it is significant to measure the power consumption of each VM for green cloud data centers. Since there is no device that can directly measure the power consumption of each VM, modeling methods have been proposed. However, current models are not accurate enough when multi-VMs are competing for resources on the same server. One of the main reasons is that the resource features for modeling are correlated with each other, such as CPU and cache.

Zhaoning Zhang et al. [6][7] defined that Infrastructure as a service (IaaS) allows users to rent resources from the Cloud to meet their various computing requirements. The pay-as-you-use model, however, poses a nontrivial technical challenge to the IaaS cloud service providers: how to fast provision a large number of virtual machines (VMs) to meet users' dynamic computing requests? We address this challenge with VMThunder, a new VM provisioning tool, which downloads data blockson demand during the VM booting process and speeds up VM image streaming by strategically integrating peer-to-peer (P2P) streaming techniques with enhanced optimization schemes such as transfer on demand, cache on read, snapshot on local, and relay on cache. In particular, VMThunder stores the original images in a share storage and in the meantime it adopts a tree-based P2P streaming scheme so that common image blocks are cached and reused across the nodes in the cluster.

Jiaxin Li et al. [1] proposed a Layered Progressive resource allocation algorithm for multi-tenant cloud data centers based on the Multiple Knapsack Problem (LP-MKP). The LP-MKP algorithm uses a multi-stage layered progressive method for multi-tenant VM allocation and efficiently handles unprocessed tenants at each stage. This reduces resource fragmentation in cloud data centers, decreases the differences in the QoS among tenants, and improves tenants' overall QoS in cloud data centers.

Gursharan Singh et al. [2] proposed a technique that reduces the size of data image stored on source host before migration. When a Virtual Machine migrates to another host, the data image for that VM is kept in the source host after removing unwanted data according to the probability factor. When the VM migrates back to the original host later, the kept memory image will be "reused", i.e. data which are identical to the kept data will not be transferred and comparative to existing

system the size of memory image is small. To validate this approach, results evaluated using different threshold levels and probability factor of change in data. Proposed system required less memory to store the memory image and allow more VMs to be hosted.

Narander Kumar et al. [3] focused on quantitative analysis of live migration within a cloud data centre with the aim of understanding the factors which are responsible for cloud's efficiency. Various key parameters, such as, virtual machine size, network bandwidth available and dirty rate of a cloud application are discussed in detail and given the comparisons also, to give a clear view of their role in live migration's performance. The analysis presented in this paper gives a proper platform for considering future enhancements and/or modifications in the existing migration technology.

Aarti Singh et al. [7] proposed an Autonomous Agent Based Load Balancing Algorithm (A2LB) which provides dynamic load balancing for cloud environment. Cloud Computing revolves around internet based acquisition and release of resources from a data center. Being internet based dynamic computing; cloud computing also may suffer from overloading of requests. Load balancing is an important aspect which concerns with distribution of resources in such a manner that no overloading occurs at any machine and resources are optimally utilized. However this aspect of cloud computing has not been paid much attention yet. Although load balancing is being considered as an important aspect for other allied internet based computing environments such as distributed computing, parallel computing etc. Many algorithms had been proposed for finding the solution of load balancing problem in these fields. But very few algorithms are proposed for cloud computing environment. Since cloud computing is significantly different from these other types of environments, separate load balancing algorithm need to be proposed to cater its requirements.

S. Sohrabi et al. [8] introduced two new virtual machine selection policies, Median Migration Time and Maximum Utilization, and show that they outperform existing approaches on the criteria of minimising energy consumption, service level agreement violations and the number of migrations when combined with different hotspot detection mechanisms. Applications are first assigned to virtual machines which are subsequently placed on the most appropriate server host. If a server becomes overloaded, some of its virtual machines are reassigned. This process requires a hotspot detection mechanism in combination with techniques that select the virtual machine(s) to migrate.

Mohammad Mehedi Hassan et al. [9] proposed a cost effective and dynamic VM allocation model based on Nash bargaining solution. With various simulations it is shown that the proposed mechanism can reduce the overall cost of running servers while at the same time guarantee QoS demand and maximize resource utilization in various dimensions of server resources.

Christina Terese Josepha et al. [10] proposed a novel technique to allocate virtual machines using the Family Gene approach. Experimental analysis proves that the proposed approach reduces energy consumption and the rate of migrations, The concept of virtualization forms the heart of systems like the Cloud and Grid. Efficiency of systems that employ virtualization greatly depends on the efficiency of the technique used to allocate the virtual machines to suitable hosts. The literature contains many evolutionary approaches to solve the virtual machine allocation problem, a broad category of which employ Genetic Algorithm.

Antony Thomas et al. [11] introduced an improved scheduling algorithm after analyzing the traditional algorithms which are based on user priority and task length. High prioritized tasks are not given any special importance when they arrive. Min-Min algorithm is used to reduce the make span of tasks by considering the task length. Keeping this in mind, cloud providers should achieve user satisfaction. Thus research favours scheduling algorithms that consider both user satisfaction and resources availability.

DoshiChintanKetankumar et al. [12] proposed a green cloud broker for resource procurement problem by considering the metrics of energy efficiency and environmental friendly operations of the cloud service provider. Author used mechanism design methods to decide the allocation and payment for the submitted job dynamically and performed experiments and show the results of comparisons of energy consumption and emission of greenhouse gases between the allocation decided by the proposed green cloud broker and a without taking the green metric into consideration.

A. I. Awad et al. [13] proposed mathematical model using Load Balancing Mutation (balancing) a particle swarm optimization (LBMPSO) based schedule and allocation for cloud computing that takes into account reliability, execution time, transmission time, make span, round trip time, transmission cost and load balancing between tasks and virtual machine .LBMPSO can play a role in achieving reliability of cloud computing environment by considering the resources available and reschedule task that failure to allocate. Our approach LBMPSO compared with standard PSO, random algorithm and Longest Cloudlet to Fastest Processor (LCFP) algorithm to show that LBMPSO can save in make span, execution time, round trip time, transmission cost.

Arunkumar. G et al. [14] outlined the existing cloud technologies, interoperability issues and possible solution to overcome the problems. Most of the consumers are analyzing the appropriateness of cloud to employ themselves for their enterprise or personalized operations. Customers are self-satisfied at the inception, but expectation changes. Based on their business escalation it needs further adoption of modern cloud services the existing cloud provider fails to offer. Hence the user needs interoperability and portability to ship their assets from one cloud to other cloud. The complication faced by the customers in shifting their assets remains as a challenge to be addressed.

AtulVikasLakra et al. [15] proposed a multi-objective task scheduling algorithm for mapping tasks to a VMs in order to improve the throughput of the datacenter and reduce the cost without violating the SLA (Service Level Agreement) for an application in cloud SaaS environment. The proposed algorithm provides an optimal scheduling method. Most of the algorithms schedule tasks based on single criteria (i.e execution time). But in cloud environment it is required to consider various criteria like execution time, cost, bandwidth of user etc.

Narander Kumar et al. [16] proposes a demand-based preferential resource allocation technique that designs a market-driven auction mechanism to identify users for resource allocation based on their payment capacities and implements a payment strategy based on a buyer's service preferences. A comparison is drawn between the proposed allocation technique and the famous off-line VCG auction mechanism and results show a performance benefit in revenues to service provider, payments of cloud users besides ensuring an optimum resources use.

NidhiBansal et al. [17] developed a method to calculate cost of QoS-driven task scheduling algorithm and compare with traditional task scheduling algorithm in cloud computing environment. It also defined many parameters that are to be considered in QoS driven like makespan, latency and load balancing. But allocation cost parameter is not considered in QoS-driven scheduling algorithm. Minimizing the total allocation cost is an important issue in cloud computing.

Mehmet Sahinoglua et al. [18] addressed a discrete event CLOUD simulator, namely CLOURAM (CLOUD Risk Assessor and Manager) to estimate the risk indices in large CLOUD computing environments, comparing favorably with the intractably theoretical Markov solutions or hand calculations that are limited in scope. The goal is to optimize the quality of a CLOUD operation and what countermeasures to take to minimize threats to the service quality by reserve planning of reserve crew members.

Mohammad Mehedi Hassan et al. [19] proposed an automatic method, based on Multi-Objective Particle Swarm Optimization, for the identification of power models of enterprise servers in Cloud data centers. The average consumption of a single data center is equivalent to the energy consumption of 25.000 households. Modeling the power consumption for these infrastructures is crucial to anticipate the effects of aggressive optimization policies, but accurate and fast power modeling is a complex challenge for high-end servers not yet satisfied by analytical approaches.

III. PROPOSED OPTIMUS SCHEDULER ALGORITHM FOR ADD, REMOVE OR RESIZE AN APPLICATION BY USING TABU SEARCH ALGORITHM

Given that a VM scheduler in a cloud data center could need to scale to more than 100,000 servers and VMs, we developed a very efficient hierarchical heuristic based on tabu search. We name the proposed method Optimus Scheduler (OS). We exploit the fact that data center topologies are split in clusters and sub clusters (pods). As the topology, each pod in the test cases contains 1,600 servers that can host a total of 6,400 VMs. Then each pod has a separate scheduler that handles VMs assigned to it. Each pod is split in racks of servers-e.g., 40 racks of 40 servers. Each server in a rack is symmetrical to the other servers in the same rack because they are all connected to the same rack switch with the same link capacity.

The general structure of the heuristic shown in flowing Optimus Scheduler Algorithm has important differences in the neighborhood relation between solutions, objective function calculation, tabu list size, number of iterations before stopping. the execution, and in the possibility to resize applications on the fly.

In this problem, a solution S is a mapping between VMs and servers. The neighborhood N(S) is the set of solutions that differ in the placement of one of the VMs that is being scheduled. To take advantage of the symmetry between solutions, each possible movement is to move each VM to the first available server in each rack. Thus, the number of possible movements is reduced to the number of VMs to schedule multiplied by the number of racks in a pod. The reduction of movements is valid because the servers in the same rack are equal in terms of delay. Furthermore, choosing the first one available optimizes the second objective that is the power consumption because a server will be activated only if the previous ones were full. This assumes that servers are sorted by energy efficiency in each rack. Thus, the first available server will also be the most efficient one among all availabilities.

The number of iterations q to keep a movement in the tabu list is the square root of the neighborhood size. In this case, a second restriction was added to the tabu movements. Each VM that is moved to a server is then kept in that server for a number of iterations that is half of the number of VMs to schedule. Keeping a VM fixed in a new rack for a number of iterations allows the other VMs to be moved to that rack, and that solution with the whole application in a different rack can be evaluated by the algorithm. When the solution does not improve for MAX ITERATIONS the algorithm stops. That value was set to the number of VMs to schedule multiplied by the number of racks.

ALGORITHM

Step 1: function optimus Scheduler (Request, M)
Step 2: an← Request application
Step 3: if Request type is add then
Step 4: M ← Initial Greedy Solution(an, M)
Step 5: M ←TabuSearch(an, M)
Step 6: end if
Step 7: if Request type is remove then
Step 8: Remove an's VMs from M
Step 9: end if
Step 10: if Request type is resize then
Step 11: Remove an's VMs from solution M
Step 12: Add or remove VMs from an
Step 13: Place existing VMs in the same servers than before in M
Step 14: if an has new VMs to schedule then
Step 15: M←InitialGreedySolution(an, M)
Step 16: M ←TabuSearch(an, M)
Step 17: end if
Step 18: end if
Step 19: end function
Step 20: function InitialGreedySolution(an, S)
Step 21: for each vm in vmsToSchedule(an) do
Step 22: Move vm to the server that least increases z(S)
Step 23: end for
Step 24: return S
Step 25: end function
Step 26: function TabuSearch (an, Current)
Step 27: S ← Current
Step 28: Best ← Current
Step 29: L← {}
Step 30: i ← 0
Step 31: repeat
Step 32: Choose the pair (c; s) ϵ N(S) - L that minimizes z when moving vm to s in S.
Step 33: Move c to s in S
Step 34: Add (c; s) to tabu list L for q iterations
Step 35: i ← i + 1
Step 36: if z(S) <z(Best) then
Step 37: Best ← S
Step 38: i ← 0
Step 39: end if
Step 40: until i = MAX ITERATIONS

Step 41: return Best
Step 42: end function

IV. RESULTS AND DISCUSSIONS

This section analyzes the proposed Optimus Scheduler (OS) for the case study, and compares OS with First Fit (FF) policy. We first analyze an initial period when the applications are deployed, and then the whole day with a varying workload.

A. Initial Period

We first added 800 applications with a total of 5,027 VMs corresponding to the workload between 5 PM and 6 PM to a pod of 1,600 servers. For each application, OS uses the tabu search algorithm to place its VMs. In another experience with the same applications and VMs, the policy to place each VM was FF that is to choose for each VM the first server with enough capacity.

Table 1 shows the results obtained by the two policies. The average transmission delay was 70% higher in FF than in OS, and the average link utilization was 9.9% higher in FF than OS. That is because OS has the minimization of delay as first priority, and FF does not take traffic into account to place VMs. When an application with multiple VMs is deployed, FF picks the first the server for the first VM. If that server has not enough space for the second VM, then it will be placed in a second server. The traffic between these VMs will use the links between the servers and the rack switch. On the other hand, OS will pick a server with enough space for two VMs, and the traffic between both VMs will be kept within the server and will not use any link.

Table 1: Optimus Scheduler vs First-Fit after adding 800 applications

Solution	OS (Optimus Scheduler)	FF (First-Fit)
Average Delay	12.3μs	19.6 μs
Average Link Utilization	12.3%	24.0%
Average Inter Switch Link Utilization	11.2%	12.3%
Maximum Inter Switch Link Utilization	32.2%	36.7%
Number of VMs	5,027	5,027
Number of servers	1,258	1,257
Total power	267.5 kW	267.4 kW

Considering the links that connect rack and pod switches, the average link utilization was 1.4% higher in FF than OS. That improvement is because OS minimizes delay by placing the VMs of each application in a single rack, and reduces the use of inter-rack links. Because FF places the VMs of an application in consecutive servers, they will often be placed in the same rack. However, when a rack is almost full, an application will be split and the links between rack and pod switches will be used. As we will see, the difference between OS and FF is larger when applications are resized over the day.

With respect to power consumption, both mechanisms are power-efficient because they tend to fill an active server with VMs before turning on an inactive server. In some cases, OS will temporally consume more power because it will prefer to turn on a server to place VMs of the same application in the same rack. However, the VMs of the next applications will use the server that had been kept partially occupied, and the total number of servers used is almost the same.

OS minimizes power consumption by taking into account the consumption of each type of server. In the case shown, all the racks have servers with the same type of energy consumption. When racks of energy-efficient servers and energy-inefficient servers are alternated, OS consumes 332kW and FF, 357 kW, an 8% advantage for OS.

B. Workload

Once the 800 applications are deployed, we consider a workload that varies over the day. Each application is resized each hour according to the maximal workload expected for that hour. New VMs are deployed when the workload increases in size, and VMs are removed in periods where the workload decreases. Figure 7.1 shows the number of VMs used to match the workload in each period and the power consumption achieved by CC in a pod of 1,600 servers.

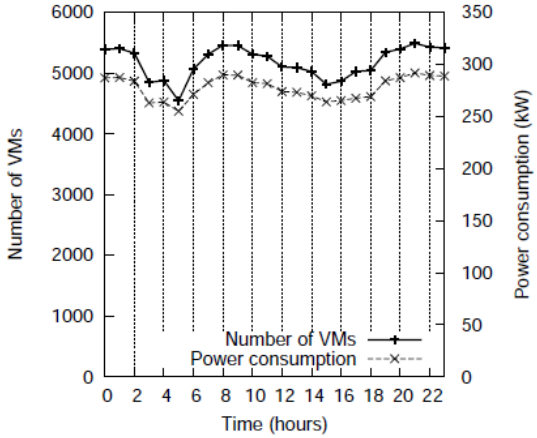

Figure 2: Number of VMs used and power consumption in each period.

First, we compare how much energy is saved by considering the variation of workload instead of dimensioning for the peak period. In this case, the peak period is at 9 PM when 291.8 kW are consumed by the VMs placed by OS in a pod, or 23.3 MW if the pod power consumption is extrapolated to the whole data center. If the servers used in that peak period consume that power during a year, 204.4 millions kWh would be consumed. Multiplying by an electricity price of $0.16 / kWh gives a total of $32.7 millions for the electricity employed by the data center during a year. Using OS to resize applications during the day and suspending unused servers saves 4.9% of energy consumption, which becomes $1.6 millions per year. These savings are from the cloud provider point of view. Users that deploy applications would achieve higher savings in the cost paid to the provider to host VMs. Applications whose workload have more variability during the day benefit the most.

OS is then compared to the FF policy at the time of resizing applications each hour of the day. Table 2 presents results of both policies in the peak period. The first remark that can be highlighted is that FF produces 47% more delay than OS. That also implies that the links are 10.3% more used, thus increasing the queuing delay and degrading the QoS. In particular, links that connect rack switches with pod switches reach up to 80% utilization in FF. That is because 32 racks are used and 8 racks remain free after placing the applications in the initial period. VMs added in high activity periods are then placed in the last racks.

Table 2: Optimus Scheduler vs First-Fit in the highest period (9 PM)

Solution	Optimus Scheduler	First-Fit
Average Delay	13:5 µs	19:9 µs
Average link utilization	20.7%	31.0%
Average Inter Switch Link Utilization	15.2%	27.3%
Maximum Inter Switch Link Utilization	47.0%	80.0%
Number of VMs	5,490	5,490
Number of servers	1,385	1,373
Total power	291.8 kW	290.6 kW

The traffic of the new VMs is directed to the load balancers placed in the first racks using the links between rack and pod switches. On the other hand, OS uses 40 racks from the starting period leaving available space in each rack for VMs that arrive in high periods. Then, OS places each new VM in the rack that hosts the corresponding application and thus reduces the link utilization between racks and improves the QoS. The cost to pay is that 12 more servers are used in the pod of 16,000 servers, and power consumption increases 0.4% compared to FF in the case where all servers are equally energy-efficient. The energy consumed in the whole day is 0.3% higher in OS than FF because a few more servers were used to improve the QoS.

C. Execution Time

The execution time of OS was evaluated for different application sizes and number of servers. To vary the application size, an empty pod with 1,600 servers was gradually filled with applications with between 6 and 40 VMs each. For each application, the time to schedule the application was recorded. With that information, the average execution time for each application size was calculated as seen in Figure 2.

To analyze the variation of scheduling time as a function of the network size, the number of VMs in each application was set as 20 and we varied the number of racks, with 40 servers per rack. For each network size, applications are scheduled until all the servers are full with VMs. The average scheduling time was calculated for each test. Figure 3 shows that the scheduling time of an application grows linearly as a function of the number of racks or servers. In particular, scheduling an application with 20 VMs in a pod of 1,600 servers take an average of 283 ms.

Concerning the workload variation over the day, the scheduler must resize each application each hour of the day. That implies choosing the servers to place new VMs in the periods when an application expands. The fact that only the new VMs are deployed makes the operation very quick compared to the operation of initial application deployment. Adding800 applications with their 5,027 VMs in a pod of 1,600 servers took 33 seconds in the initial period. Resizing those same applications only took a total of half a second in each period.

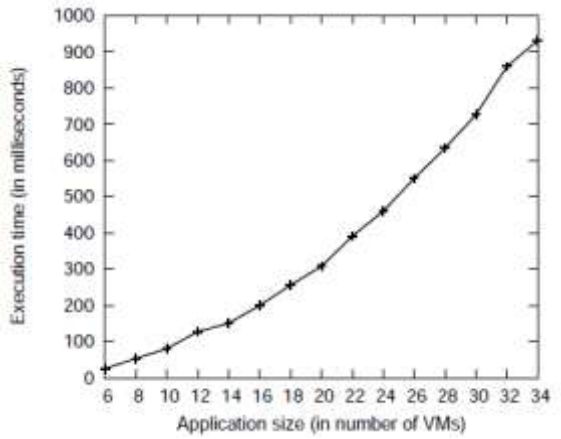

Figure 3: Average scheduler execution time over the application size

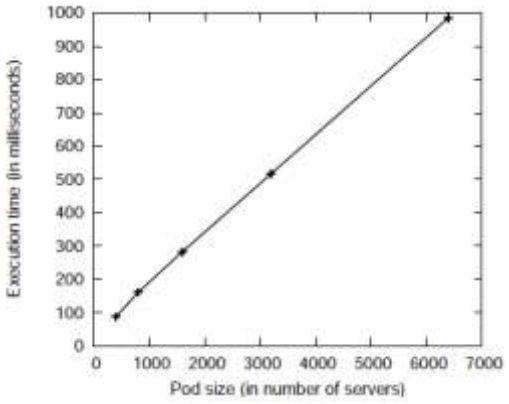

Figure 4: Average scheduler execution time over the network size

D. Discussions of the Results

The method proposed in this chapter shows the importance of taking the traffic among VMs into account to improve the QoS. The minimization of delay implies that VMs that communicate among themselves are placed in nearby servers and have more communication bandwidth available. This also provokes that links are less used and thus reduces the queuing delay. The reduction in delay, probability of congestion, and increase in throughput implies that the application QoS is improved.

An alternative to reduce the queuing delay and increase throughput among VMs is to augment the link capacity between switches. Topologies such as Fat tree [20] and VL2 [21] achieve full bisectional bandwidth and each pair of servers can communicate close to the maximal capacity of their network cards. However, that approach has a high cost in number of switches and power consumption that could be not justified when not all the applications require high throughput among VMs. These topologies are indeed useful for specific applications that do require an all to all communication pattern such as Hadoop clusters.

Experiences in this chapter study communication delay, throughput, and link utilization, metrics that vary depending on which server host each VM. The server processing time was not included because the processing power is a fixed requirement of each VM type, e.g., in number of virtual CPUs in a specific frequency. The number of VMs that will be assigned to an application will determine the number of requests per unity of time that each VM will handle and its processing time. However, the total response time is composed

of the processing time of each VM that participates, e.g., load balancer, web server, database, memcache, and the communication delay. The way these times are combined is difficult to predict precisely because that will depend how the application is implemented. For instance, if a web VM makes requests to multiple databases in parallel or in a sequential fashion. We believe that the relation between communication delay and total response time should be analyzed through measurements in specific implementations. The fact that the transmission time is expressed in microseconds should not mislead to think that is negligible compared to a total response time in milliseconds because each application request can require multiple internal requests among VMs and because if congestion is not avoided the queuing delay could increase. As this chapter shows, our approach reduces congestion and communication delay and the total response time as a result.

V. CONCLUSION

This chapter presented an online method to place VMs in servers that optimizes communication among VMs and power consumption. The proposed scheduler considers server heterogeneity and VMs with different types of requirements. Taking advantage of regular network topologies makes possible to scale in the delay calculation and parallelization on multiple clusters.

Furthermore, the proposed scheduler considers the elasticity of applications to improve the QoS of VMs that arrive in peak periods. The method was formalized through a Mixed Integer Programming (MIP) model and a heuristic was developed scaling to more than 100,000 servers and applications. Applications can be added and removed on the y, and the number of VMs of an application can be changed to match the workload that varies over the day. A case study shows the advantages of the proposed approach. Compared to first-fit policy, the delay among VMs is reduced by 70% and the most used network links are decreased a 33%. We also found that a 4.9% of power consumption is saved compared to statically provisioning of resources for the peak period. In a large data center with 128,000 servers, these annual savings represent $1.6 millions when the energy is charged $0.16 / kWh. Using first-fit in each period of the day taking into account the varying workload achieved 0.3% less energy consumption than our approach. That is because a few more servers were activated, which is a small price to pay that gives important benefits on the QoS. These results show that the proposed scheduler can be a key element in the management of a cloud provider contributing competitive advantages in QoS, energy consumption, and costs.

REFERENCES

[1] Jiaxin Li, Dongsheng Li, Yuming Ye, Xicheng Lu, "Efficient Multi-Tenant Virtual Machine Allocation in Cloud Data Centers", TSINGHUA SCIENCE AND TECHNOLOGY, ISSN: 1007-0214, Volume 20, Number 1, February 2015, pp: 81-89.

[2] Gursharan Singh, Sunny Behal, Monal Taneja, "Advanced Memory Reusing Mechanism for Virtual Machines in Cloud Computing", 3rd International Conference on Recent Trends in Computing, Vol: 57, 2015, pp: 91-103.

[3] Narander Kumar, Swati Saxena, "Migration Performance of Cloud Applications- A Quantitative Analysis", International Conference on Advanced Computing Technologies and Applications, Vol. 45, 2015, pp: 823-831.

[4] Ahmad Nahar Quttoum, Mohannad Tomar, Bayan Khawaldeh, Rana Refai, Alaa Halawani, Ahmad Freej, "SPA: Smart Placement Approach for Cloud-service Datacenter Networks", The 10th International Conference on Future Networks and Communications, Vol: 56, 2015, pp: 341-348.

[5] Chonglin Gu, Pengzhou Shi, Shuai Shi, Hejiao Huang, Xiaohua Jia, "A Tree Regression-Based Approach for VM Power Metering", Special Section On Big Data For Green Communications And Computing, IEEE, June 1, 2015.

[6] Zhaoning Zhang, Ziyang Li, Kui Wu, Dongsheng Li, Huiba Li, Yuxing Peng, Xicheng Lu, "VMThunder: Fast Provisioning of Large-Scale Virtual Machine Clusters", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Vol. 25, No. 12, DECEMBER 2014.

[7] Aarti Singh, Dimple Juneja, Manisha Malhotra, "Autonomous Agent Based Load Balancing Algorithm in Cloud Computing", International Conference on Advanced Computing Technologies and Applications, Vol: 45, 2015, pp: 832-841.

[8] S. Sohrabi, I. Moser, "The Effects of Hotspot Detection and Virtual Machine Migration Policies on Energy Consumption and Service Levels in the Cloud", ICCS, Vol: 51, 2015, pp: 2794-2798.

[9] Mohammad Mehedi Hassan, Atif Alamri, "Virtual Machine Resource Allocation for Multimedia Cloud: A Nash Bargaining Approach", International Symposium on Emerging Inter-networks, Communication and Mobility, Vol: 34, 2015, pp: 571-576.

[10] Christina Terese Josepha, Chandrasekaran K, Robin Cyriaca, "A Novel Family Genetic Approach for Virtual Machine Allocation", International Conference on Information and Communication Technologies, Vol: 46, 2015.

[11] Antony Thomas, Krishnalal G, Jagathy Raj V P, "Credit Based Scheduling Algorithm in Cloud Computing Environment", International Conference on Information and Communication Technologies, Vol: 46, 2015, pp: 913-920.

[12] Doshi Chintan Ketankumar, Gaurav Verma, K. Chandrasekaran, "A Green Mechanism Design Approach to Automate Resource Procurement in Cloud", Eleventh International Multi-Conference on Information Processing, Vol: 54, 2015, pp: 108-117.

[13] A.I.Awad, N.A.El-Hefnawy, H.M.Abdel_kader, "Enhanced Particle Swarm Optimization For Task Scheduling In Cloud Computing Environments", International Conference on Communication, Management and Information Technology, Vol: 65, 2015, pp: 920-929.

[14] Arunkumar. G, Neelanarayanan Venkataraman, "A Novel Approach to Address Interoperability Concern in Cloud Computing", 2nd International Symposium on Big Data and Cloud Computing, Vol: 50, 2015, pp: 554-559.

[15] Atul Vikas Lakra, Dharmendra Kumar Yadav, "Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization", International Conference on Intelligent Computing, Communication & Convergence, Vol: 48, 2015, pp: 107-113.

[16] Narander Kumar, Swati Saxena, "A Preference-based Resource Allocation In Cloud Computing Systems", 3rd International Conference on Recent Trends in Computing, Vol: 57, 2015, pp: 104-111.

[17] Nidhi Bansal, Amitab Maurya, Tarun Kumar, Manzeet Singh, Shruti Bansal, "Cost performance of QoS Driven task scheduling in cloud computing", Third International Conference on Recent Trends in Computing, Vol: 57, 2015, pp: 126-130.

[18] Mehmet Sahinoglua, Sharmila Ashokan, Preethi Vasudev, "Cost-Efficient Risk Management with Reserve Repair Crew Planning in CLOUD Computing", Cost-Efficient Risk Management with Reserve Repair Crew Planning in CLOUD Computing, Vol: 62, 2015, pp: 335-342.

[19] Patricia Arroba, Jos´e L. Risco-Mart´ın, Marina Zapater, Jos´e M. Moya, Jos´e L. Ayala, Katzalin Olcoz, "Server Power Modeling for Run-time Energy Optimization of Cloud Computing Facilities", 6th International Conference on Sustainability in Energy and Buildings, Vol: 62, pp: 2014, pp: 401-410.

[20] U.S. Energy Information Administration, "Average retail price of electricity to ultimate customers by end-use sector," May 2013.

[21] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, 2008, pp. 63-74.