International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume II, Issue V, September- 2014

EXTENDED FRAMEWORK FOR DYNAMIC RESOURCE ALLOCATION USING ASJS ALGORITHM IN CLOUD COMPUTING ENVIRONMENT

R.Gurumoorthy, V.S.Suresh Kumar
PG StudentAssistant Professor,Nandha College of Technology,Erode
gurumoorthy.xor@gmail.com,sureshkumar@nandhatech.org

Abstract: *Cloud computing allows business customers to scale up and down their resource usage. Here we grant a system that uses virtualization technology to allocate data center resources dynamically based on application demands and support green computing by optimizing the number of servers in use. We initiate the concept that to reduce "skewness" which means the unevenness in the multidimensional resource utilization of a server. By reducing skewness, we achieve good performance and the overall utilization of server resources. We broaden a set of heuristics that avoid overload in the system effectively while saving energy used. Adaptive Scoring Job Scheduling algorithm (ASJS) is applied for cloud nodes resource scheduling so that the given job is split into 'N' tasks along with Replication Strategy.*

INTRODUCTION

The elasticity and the lack of upfront capital investment offered by cloud computing is appealing in business. There is much discussion on the benefits and costs of the cloud model and on how to move legacy applications onto the cloud platform.

We intend to achieve two goals in our algorithm are:

1. **Overload avoidance**. The capacity of a PM should be sufficient for all running VMs. Otherwise, the PM get overloaded and lead to degraded the performance of VMs.
2. **Green computing**. By using VM to reduce the PM Utilization.

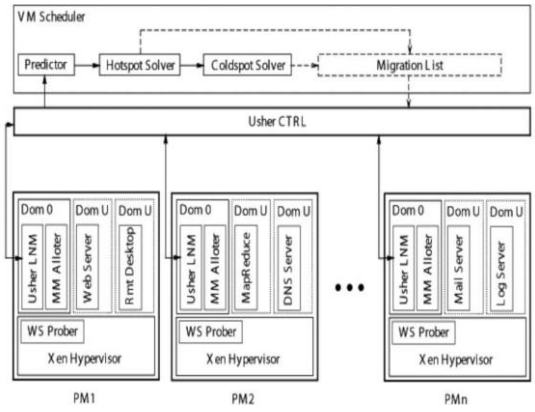

Fig 1. Overview of Xen Hypervisor

On the design and implementation of an ARMS that reach a good balance between the two goals. We construct the following contributions:

1. Develop a resource allocation system that can avoid overload in the system effectively while minimizing the number of servers used.
2. Introduce the concept of "skewness" to measure the irregular consumption of a server. By reducing skewness, we can progress the overall utilization of servers in the face of multidimensional resource constraints.
3. Design a load prediction algorithm that can capture the future resource usages of applications accurately without looking within the VMs. The algorithm can capture the rising trend of resource usage patterns and help reduce the placement agitate significantly.

2. RELATED WORK

Resource Allocation at the Application Level- Automatic scaling of Web applications was previously studied in [1] and [2] for data center environments. Each server has replicas of all web applications running in MUSE[3].Based on the dispatch algorithm in a frontend L7-switch makes sure requests served to minimizing the number of not utilizing servers. Work [7] uses network flow algorithms to allocate the load of an application among its running works. For relation slanting Internet services, work [10] presents an integrated approach for load dispatching and server provisioning. The works require the applications be structured in a multitier architecture with load balancing provided through an front-end dispatcher. The contrasted work targets Amazon EC2-style environment where it places no restriction on what and how applications are constructed within the VMs. A VM is treating like a blackbox. Resource management is completed at the small of whole VMs.

Resource Allocation by Live VM Migration-The HARMONY system applies virtualization technology across multiple resource layers [4]. It uses VM and data migration to mitigate hot spots on the servers, network devices and the storage nodes. Extended Vector Product (EVP) as resource utilization indicator. Multidimensional knapsack problem is variant of load balancing algorithm. By minimizing SLA violations Dynamic placement of virtual servers . Packing problem and use the well-known first-fit approximation algorithm for calculate the VM to PM layout periodically. That algorithm is designed mostly for offline use and It likely to invite a large number of migrations when applied in online environment where the resource needs of VMs change dynamically.

Sandpiper combines multidimensional load information into a singleVolume metric [8]. It sorts the list of PMs based on theirvolumes and the VMs in each PM in their volume-to-sizeratio (VSR). This unfortunately abstracts away critical information needed when making the migration decision.Based on the presorted order PM and VM machine are cosidered. We give a concrete example in Section 1 of the additional file, where their algorithm selects the wrong VM to migrate away during overload and fails to alleviate the hot spot. We also measure up to our algorithm and theirs in existent research. The results are analyzed in Section 5 of the additional file, which is available online, to show how they behave differently

Green Computing-Many efforts have been made to restrain energy consumption in data centers. Novel thermal design for less cooling power or power-proportional and low-power hardware are hardware based approach. Work [22] uses dynamic voltage and frequency scaling (DVFS) to adjust CPU power basis of its load. We do not use DVFS for green computing, in the supple-mentary file. PowerNap [23] resorts to new hardware technologies such as solid state disk (SSD) and Self-Refresh DRAM to implement rapid transition(less than 1ms) between full operation and low power state, so that it can "take a nap" in short still intervals. When a server goes to sleep, Somniloquy [4] notifies an embedded system residing on a special designed NIC to hand over the main operating system. It gives the delusion that the server is always active.

Our work belongs to the category of pure-software low-cost solutions [5], [6]. Similar to Somniloquy [4], SleepServer [7] initiates virtual machines on a devoted server as hand over, instead of depending on a special NIC. LiteGreen [10] does not use a delegate.The desktop can sleep so that Instead it moving the desktop OS left. It requires the shared storage desktop is virtualized with shared storage

3. THE SKEWNESS ALGORITHM

We introduce the concept of skewness to quantify the unevenness in the utilization of multiple of a server. We consider n is the number of server and ri be the utilization of the ith resource. We define the resource skewness of a server p as

$$skewness(p) = \sqrt{\sum_{i=1}^{n}\left(\frac{r_i}{\bar{r}} - 1\right)^2},$$

Where r is the average utilization of all resources. Not all types of resources in practice are performance critical and hence we only need to consider bottleneck resources in the above calculation. We can combine different types of workloads nicely and improve the overall utilization of server resources only to reduce skewness. In the following, about the details of our algorithm. By comparision between SPAR and FUSD are

4. HOT AND COLD SPOTS

Our algorithm executes periodically to evaluate the resource allocation status based on the predicted future resource demands of VMs. We define a server as a hot spot if the utilization of any of its resources is above a hot threshold. This indicates that the server is overloaded and hence some VMs running on it should be migrated away. We define the temperature of a hot spot p as the square sum of its resource [8] utilization beyond the hot threshold:

$$temperature(p) = \sum_{r \in R}(r - r_t)^2,$$

Where R is the set of overloaded resources in server p and rt is the hot threshold for resource r. (Note that only overloaded resources are considered in the calculation.) The temperature of a hot spot reflects its degree of overload. If a server is not a hot spot, its temperature is zero.

We define a server as a cold spot if the utilizations of all its resources are below a cold threshold. This indicates that the server is mostly idle and a potential candidate to turn off to save energy. However, we do so only when the average resource utilization of all actively used servers [10] (i.e., APMs) in the system is below a green computing threshold.

A server is actively used if it has at least one VM running. Otherwise, it is inactive. Finally, we define the warm threshold to be a level of resource utilization that is sufficiently high to justify having the server running but not as high as to risk

becoming a hot spot in the face of temporary fluctuation of application resource demands

. Different types of resources can have different thresh-olds. For example, we can define the hot thresholds for CPU and memory resources to be 90 and 80 percent, respectively. Thus a server is a hot spot if either its CPU usage is above 90 percent or its memory usage is above 80 percent.

5. ADD VIRTUAL MACHINE AND CLOUD DATA CENTER

We compare the execution of our algorithm with and without load prediction. When load prediction is disabled, the algorithm simply uses the last observed load in its decision making. It shows that load prediction significantly reduces the average number of hot spots in the system during a decision run. Notably, prediction prevents over 46 percent hot spots in the simulation with 1,400 VMs [9]. This demonstrates its high effectiveness in preventing server overload proactively. Without prediction, the algorithm tries to consolidate a PM as soon as its load drops below the threshold. With prediction, the algorithm correctly foresees that the load of the PM will increase above the threshold shortly and hence takes no action. This leaves the PM in the "cold spot" state for a while. However, it also reduces placement churns by avoiding unnecessary migrations due to temporary load fluctuation.

We evaluate the effectiveness of our algorithm in overload mitigation and green computing. We start with a small scale experiment consisting of three PMs and five VMs so that we can present the results for all servers in Fig. 7. Different shades are used for each VM[10] . All VMs are configured with 128 MB of RAM. An Apache server runs on each VM. We use httperf to invoke CPU intensive PHP scripts on the Apache server. This allows us to subject the VMs to different degrees of CPU load by adjusting the client request rates. The utilization of other resources are kept low.

We first increase the CPU load of the three VMs on PM1 to create an overload. Our algorithm resolves the overload by migrating VM3 to PM3. It reaches a stable state under high load around 420 seconds. Around 890 seconds, we decrease the CPU load of all VMs gradually. Because the FUSD prediction algorithm [15] is conservative when the load decreases, it takes a while before green computing takes effect. Around 1,700 seconds, VM3 is migrated from PM3 to PM2 so that PM3 can be put into the standby mode. Around 2,200 seconds, the two VMs on PM1 are migrated to PM2 so that PM1 can be released as well. As the load goes up and down, our algorithm will repeat the above process: spread over or consolidate the VMs as needed.

Initially, the two VMs on PM1 are CPU intensive while the two VMs on PM2 are network intensive. We increase the load of their bottleneck resources gradually. Around 500 seconds, VM4 is migrated from PM2 to PM1 due to the network overload in PM2 [11]. Then around 600 seconds, VM1 is migrated from PM1 to PM2 due to the CPU overload in PM1. Now the system reaches a stable state with a balanced resource utilization for both PMs—each with a CPU intensive VM and a network intensive VM. Later we decrease the load of all VMs gradually so that both PMs become cold spots. We can see that the two VMs on PM1 are consolidated to PM2 by green computing.

6. ADAPTIVE SCORING JOBSCHEDULING ALGORITHM

Recall that the goal of the skewness algorithm is to mix workloads with different resource requirements together so that the overall utilization of server capacity is improved. In this experiment, we see how our algorithm handles a mix of CPU, memory, and network intensive workloads. We vary the CPU load as before. We inject the network load by sending the VMs a series of network packets [12]. The memory intensive applications are created by allocating memory on demand. Again we start with a small scale experiment consisting of two PMs and four VMs so that we can present the results for all servers in Fig. 11. The two rows represent the two PMs. The two columns represent the CPU and network dimensions, respectively. The memory consumption is kept low for this experiment.

Initially, the two VMs on PM1 are CPU intensive while the two VMs on PM2 are network intensive. We increase the load of their bottleneck resources gradually. Around 500 seconds, VM4 is migrated from PM2 to PM1 due to the network overload in PM2. Then around 600 seconds, VM1 is migrated from PM1 to PM2 due to the CPU overload in PM1. Now the system reaches a stable state with a balanced resource utilization for both PMs—each with a CPU intensive VM and a network intensive VM. Later we decrease the load of all VMs gradually so that both PMs become cold spots [13], [14]. We can see that the two VMs on PM1 are consolidated to PM2 by green computing.

CONCLUSION

We have presented the design, implementation, and evaluation of a resource

management system for cloud computing. Based on the changing demand our system multiplexes virtual to physical re-sources adaptively. The capacities of servers are well utilized only we use the skewness metric to combine VMs with different resource characteristics appropriately. Our algorithm mainly reaches both overload avoidance and green computing for systems with multi-resource constraints.

REFERENCES

[1] M. Armbrust et al., "Above the Clouds: A Berkeley View of Cloud Computing," technical report, Univ. of California, Berkeley, Feb. 2009.
[2] L. Siegele, "Let It Rise: A Special Report on Corporate IT," The Economist, vol. 389, pp. 3-16, Oct. 2008.
[3] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast Transparent Migration for Virtual Machines," Proc. USENIX Ann. Technical Conf., 2005.
[4] M. McNett, D. Gupta, A. Vahdat, and G.M. Voelker, "Usher: An Extensible Framework for Managing Clusters of Virtual Ma-chines," Proc. Large Installation System Administration Conf. (LISA '07), Nov. 2007.
[5] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," Proc. Symp. Networked Systems Design and Implementation (NSDI '07), Apr. 2007.
[6] C.A. Waldspurger, "Memory Resource Management in VMware ESX Server," Proc. Symp. Operating Systems Design and Implementa-tion (OSDI '02), Aug. 2002.
[7] G. Chen, H. Wenbo, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services," Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI '08), Apr. 2008.
[8] P. Padala, K.-Y. Hou, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated Control of Multiple Virtualized Resources," Proc. ACM European conf. Computer Systems (EuroSys '09), 2009.
[9] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," Proc. IFIP/IEEE Int'l Symp. Integrated Network Management (IM '07), 2007.
[10] "TPC-W: Transaction Processing Performance Council," http:// www.tpc.org/tpcw/, 2012.