Power Consumption Based Cloud Scheduler

Wu Li*
School of Software, Shanghai Jiaotong University Shanghai, 200240, China.

* Corresponding author. Tel.: 18621114210; email: defaultuser@sjtu.edu.cn
Manuscript submitted September 25, 2014; accepted May 28, 2015.
doi: 10.17706/jcp.10.4.221-228

Abstract: After years of development and trial usage, cloud computing has finally entered into mainstream and wide spread commercial use. Among the various cost elements in operating a cloud computing infrastructure, power usage stands out prominently. In fact, how to reduce the power consumption of cloud computing platforms has received considerable attention of the cloud service provider as well as academic researchers. This paper proposes a power awareness scheduler (PAS) system to manage the resource scheduling based on power usage patterns in the cloud. By obtaining and analyzing the power consumption-load curve, PAS is able to make power-efficient virtual machine deployment for the cloud. Our implementation of the proposed PAS on OpenStack shows that PAS can reduce total power consumption by up to 17%.

Key words: Cloud, OpenStack, power, scheduler.

1. Introduction

Commercial cloud platforms [1] often contain vast number of virtual machines and cost millions of dollars per month in operation. Different scheduling policies for virtual machine creation and migration have impacts on both the quality of service and operation cost. Among the various cost elements, energy cost constitutes a large part of the total expense for cloud service providers [2]. According to a report by the US Environmental Protection Agency, the 2007 electricity usage by US data centres has taken up 1.5% of the total electricity usage in the US and this proportion increases to 3.5% in 2010 [3].

A power consumption based cloud resource scheduler not only improves the resource utilization, but also reduces power consumption of the cloud [4]. This paper proposes a Power Awareness Scheduler (PAS) system to manage the resource scheduling based on power usage patterns in the cloud. By obtaining and analyzing the power consumption-load curve of relevant VMs and/or physical servers, PAS is able to make power-efficient virtual machine deployment for the cloud without sacrificing the quality of service. Specifically, PAS is able to choose the best deployment policy for a new virtual machine instance.

The rest of the paper is organized as follows: next section introduce the PAS design model, followed by a detailed description of our PAS implementation in OpenStack framework in Section 3. Section 4 is the evaluation and Section 5 discusses related work. Section 6 concludes the paper.

2. Design Model of PAS

2.1. Power Consumption Model

According to the research by Intel Corporation, power consumed by the CPU and memory makes up the

majority of the whole power consumption. As the development of DVFS (dynamic voltage and frequency scaling), the power consumption by CPU has decreased a lot. The memory power consumption is mainly related to the hit rate which depends on the allocation policy by operation system and running programs.

The power consumption can be divided to fixed power consumption and floating power consumption. Fixed consumption refers to the consumption when the system load is close to zero. The rest is floating consumption value and floats as the load of the system varies.

Since the power consumption of CPU makes up the majority of the total, CPU utilization can be used to model the power consumption-load. Assuming that P_{static} refers to the fixed consumption, P_{sys} refers to the total consumption and U refers to utilization of CPU, a formula can be got:

$$P_{sys} = a * U + P_{static} \tag{1}$$

2.2. Design Target

The basic target of PAS is to make the resources allocation power efficient without affecting the quality of services. Apart from the basic target, PAS has several other characteristics:

Platform-independent: PAS does not need to be customized for different hardware. When additional servers join the cluster, PAS can handle it without human configuration thus providing the extensibility.

Compatibility: PAS is implemented on OpenStack and completely compatible with OpenStack.

2.3. Design Method

As is shown in Fig. 1, first PAS gathers power consumption information from each server and generates the power consumption-load curve. Power consumption data and system load data are collected separately and the load of CPU is a transient value. The changes of power consumption caused by the changes of CPU load only reflect after a short period of time. So the implementation of PAS should take the delay and deviation into consideration. PAS uses software synchronization method to make the time stamp to be the same in order to get a more precise curve. PAS adds a mean filter to the CPU load data to make the curve smooth and match to the power consumption curve.

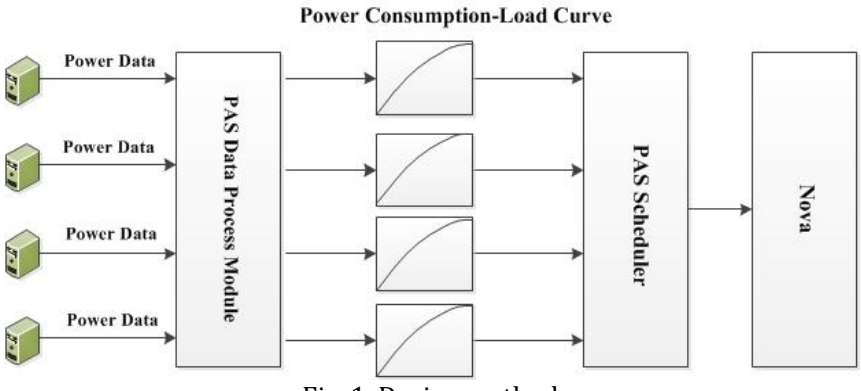

Fig. 1. Design method.

Combined with virtual machine information, PAS can decide which server to deploy the new virtual machine to save power consumption. The floating consumption of each server can be calculated by the demand load of virtual machine and the power consumption-load curve. The PAS scheduler can choose the server with lowest floating consumption without affecting the quality of services. The calculating method is given below. In the formula, P refers to power consumption and n refers to system load. To simplify the effect of virtual machine to system load of server, it is assumed that each virtual machine will make the maximum use of its VCPU. According to the characteristics of KVM, the VCPU of virtual machine is mapping to a logical

CPU of a server. So the adding system load can be calculated by below formula [5].

$$\begin{cases} n_{\text{after}} = n_{\text{now}} + vcpu_{\text{require}} \div vcpu_{\text{total}} \times 100\% \\ P = f(n) \\ \delta P = P_{\text{after}} - P_{\text{now}} = f(n_{\text{after}}) - f(n_{\text{now}}) \end{cases} \quad (2)$$

OpenStack does not provide any interfaces to gather power consumption information. In order to get rid of dependency on specific hardware, PAS leverages the universal sensor as hardware interface to monitor the power consumption. The total power consumption of CPU, memory, fan and disk is called platform consumption. PAS scheduler makes decisions based on the platform consumption data.

2.4. Architecture and Process

Fig. 2 illustrates the main components and the dependencies of PAS. The parts of gray color refer to PAS. The PAS power module is responsible for gathering power consumption data from server. The PAS data processing module is responsible for generating the power consumption-load curve. The PAS scheduler implements specific interfaces of Nova through configurations. The Nova-Scheduler will choose the best scheduling policy to schedule.

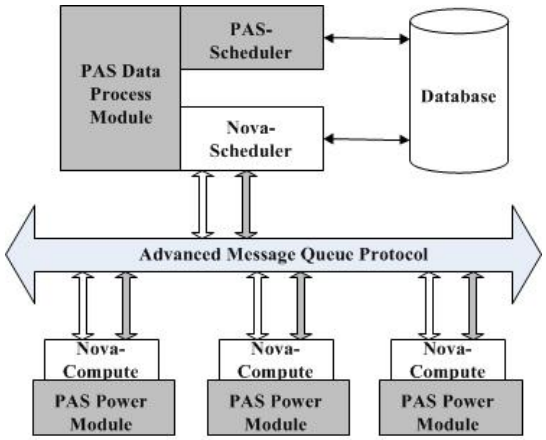

Fig. 2. Architecture of PAS.

The scheduling policy of PAS is based on power consumption curve. The power consumption curve is established as below. First PAS reads raw data of power consumption from the gathering module and processes the data using k-means algorithm. Baseline will be added if the distributed data dots cannot form a curve.

When a new node server joins the cluster, there is no record in PAS and PAS cannot make it as a candidate server when scheduling. PAS will run random load generating program on the new added server and generate a random load. When PAS gathers enough samples for generating a baseline, PAS changes it to the stage of running. As sample data increases, power consumption curve will be replaced by the new gathered data and the random baseline will not be used.

The scheduling process is divided into two phases. First PAS filters the unsatisfactory servers based on the request. The virtual CPU and virtual memory must be enough to hold the request virtual machine and the system load is no more than 80%. Second PAS chooses the lowest power consumption server to deploy the request virtual machine. PAS first reads data from database and calculates the increased power consumption based on the power consumption curve. Then PAS chooses the minimum one to deploy.

3. Implementation

The design of PAS can be applied to any cloud resource scheduler. In this paper, PAS is implemented on OpenStack for an exploration. PAS is implemented based on the Essex version of OpenStack. The operation system is based on Ubuntu Server 12.04 LSE of 64 bits.

3.1. Data Gathering and Processing

Because power consumption data and system load data are collected separately and the load of CPU is a transient value. The changes of power consumption caused by the changes of CPU load only reflect after a short period of time. So the implementation of PAS should take the delay and deviation into consideration. PAS uses software synchronization method to make the time stamp to be the same in order to get a more precise curve.

3.2. Power Consumption-Load Curve Generation

The power consumption increases with CPU load monotonically. PAS uses below function to generate the curve:

$$f(x) = a_0 * x^5 + a_1 * x^4 + a_2 * x^3 + a_3 * x^2 + a_4 * x + a_5 \quad (3)$$

PAS will periodically calculate the parameters of the generated curve and save the parameters in database.

As the time running by the system increases, the data sampled increases monotonically. It takes longer time to generate the curve with larger data. The calculating time is acceptable when there are 3000 sample data. This strategy also makes the data time associative. Even when the server is replaced by a different server, the generated curve can be matched to the new server. The time interval between the update of the curve and match to the new server is about three days.

Many noisy data dots are introduced when sampling. PAS uses k-means algorithm [6] to eliminate the noisy data. The process speed of curve generation increases as the data dots are decreased after the noisy data elimination process.

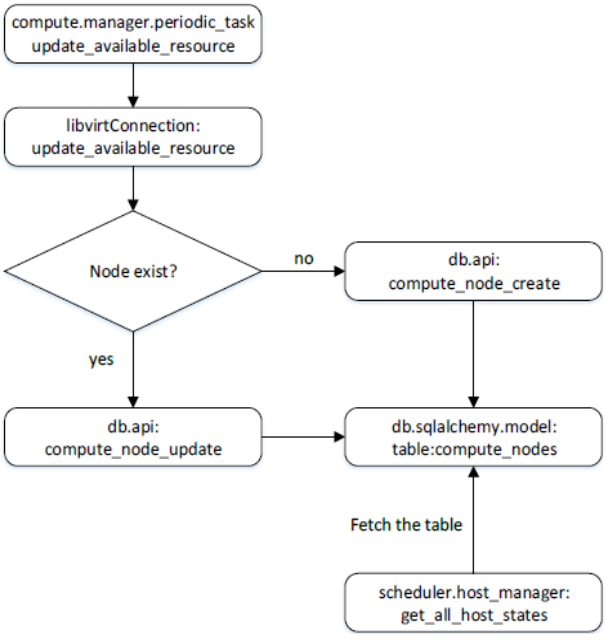

Fig. 3. Process of updating status for compute node.

3.3. Integration with OpenStack

PAS uses the interfaces of OpenStack to implement the scheduler [7] to replace the initial scheduler. PAS will call the method schedule_run_instance when there is a virtual machine creation request. The scheduling method _schedule is called after the context switch. Scheduler reads the server information from database and filters the unsatisfactory servers to ensure the service quality. Then PAS reads other data including six parameters of the curve and restore the curve. Every floating power consumption value is calculated and PAS chooses the lowest server to deploy the new virtual machine [8].

Nova is responsible for creating the virtual machine as well as reporting the status to the controller node including available resources and status of node. A timing task is added to nova-compute to call the power consumption module interface of PAS so that nova-compute can deal with power consumption data.

Nova-compute has nothing to do with curve generating. All the curve generating tasks are performed on the controller node. The process for nova-compute to report its status is illustrated as Fig. 3. All the power consumption update operations are done in update_available_resource of libvirtConnection. Nova-compute periodically use period_task decorator to call the functions in libvert module to update the status of libvirt and server. In the version of E, compute-node can automatically register its status to database. PAS adds the power consumption data by editing the function update_available_resource.

Nova-scheduler is initially responsible for accepting the creation request and completing the scheduling. PAS takes place of nova-scheduler to complete the scheduling process and then handle the control to nova-scheduler.

4. Test Benchmark

Use To make the test more precise and representative, all the tests are done on the same test environment and data is defined as average value from multiple tests.

Table 1. Server Configuration

Name	CPU	Memory	OS
cc01(Control)	Xeon 5678*2(16core)	6G	Ubuntu Server 12.04
cn01(Compute)	Intel SB(ES)1.8G(8core)	8G	Ubuntu Server 12.04
cn02(Compute)	Intel SB(ES)1.8G(8core)	4G	Ubuntu Server 12.04
cn03(Compute)	Intel SB(ES)1.8G(8core)	6G	Ubuntu Server 12.04
cn04(Compute)	Intel SB(ES)2.7G(8core)	8G	Ubuntu Server 12.04
cn05(Compute)	Intel SB(ES)2.7G(8core)	6G	Ubuntu Server 12.04

Table 2. Virtual Machine Configuration

Item	Parameter
CPU	2 VCPU
Memory	512MB
Disk	20G
Fix IP	1
OS	Ubuntu 12.04

The detail configurations for the cluster are listed in Table 1. There are 100 logical CPUs of the seven servers. The overbooking and load restriction function are closed during the test. This ensures that the total count of VCPU [9] of a server is no more than the count of logical CPU. The process of testing starts from zero load to load with 50 virtual machines. The configuration of virtual machine is listed in Table 2.

4.1. Simulation Test

PAS schedules the virtual machine to the lowest power consumption server until the VCPU is not available. The initial scheduler does not take power consumption into consideration. The initial scheduler just makes the counts of virtual machines on each server balanced. The scheduling policy of PAS takes effect

from the comparison of Fig. 4.

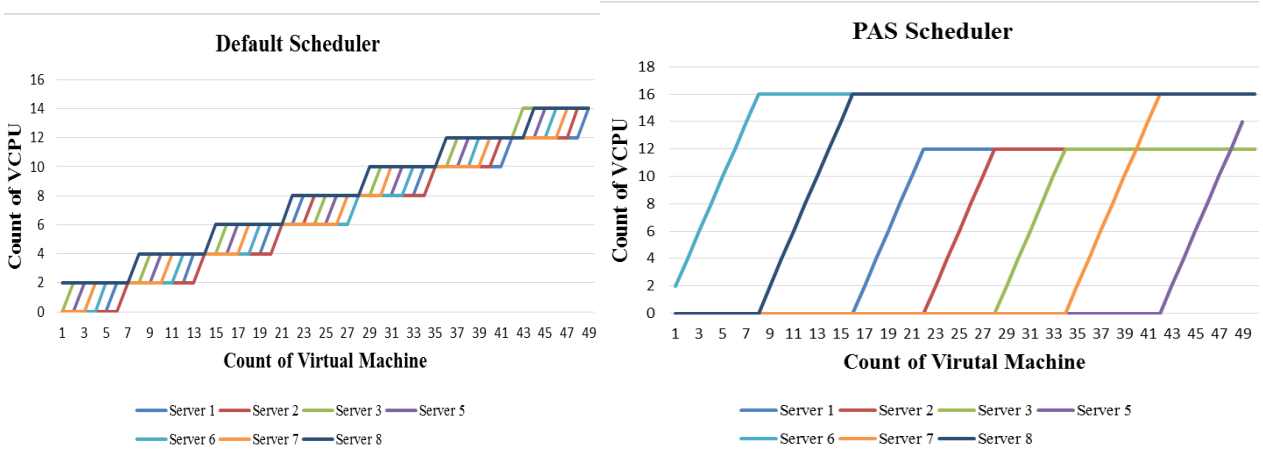

Fig. 4. Comparison of scheduling policy.

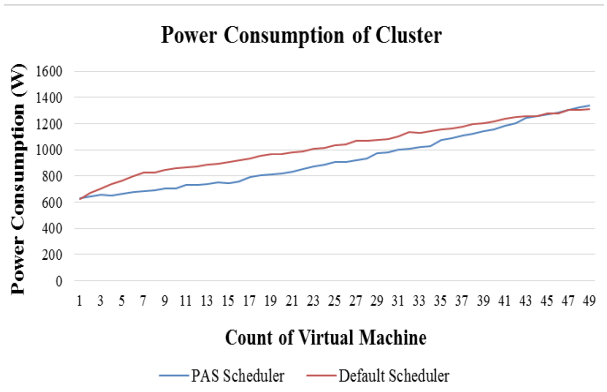

Fig. 5. Power consumption comparison of cluster.

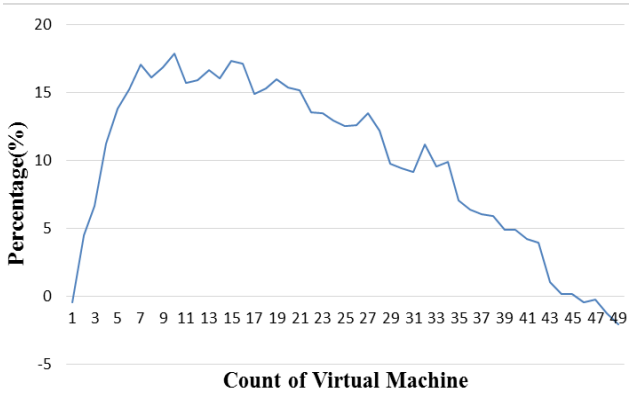

Fig. 6. Percentage of cluster power consumption saving.

Fig. 5 describes the relationship between the counts of virtual machines and cluster power consumption [10]. When the counts of virtual machines do not reach the upper limit, the power consumption using PAS of the cluster is less than the cluster using initial scheduler. As the counts of virtual machines increase, the power consumption gets closer. This result meets the expected effect.

Fig. 6 describes the percentage of power consumption savings as the counts of virtual machine increase. As the load increases, high power consumption CPU joins the cluster and the percentage of power consumption drops.

The power saving of the cluster is low at the beginning because of the fixed consumption. As the count of virtual machine increases, the percentage of power saving increases to a top value then decreases to zero. The top percentage can reach to 17%. As the restriction of the test environment, the result might be different in a varied test environment.

5. Related Work

Now servers are equipped with dynamic power management modules to reduce power consumption of computer or computing cluster by turning down unneeded service whenever possible. There are two main techniques to implement dynamic power management. One is called Dynamic Voltage (DVFS) and Frequency Scaling, and the other is the Distributed Power Management (DPM) [11]. DVFS is implemented in hardware and is designed for single servers whereas DPM is implemented in software and is designed for computing clusters. The quality of service and power consumption can also be combined [12].

6. Conclusion

PAS has implemented power consumption based cloud resources scheduler which helps the cloud platform decrease power consumption without affecting the quality of service. PAS adopts power consumption data from each server and processes the data to make the power consumption data match with the changes of system load [13]. Then PAS generates power consumption-load curve which is hardware independent. PAS implements the interfaces defined by OpenStack and chooses the lowest power consumption server to schedule. A series of tests have verified the correctness and efficiency of PAS. The OpenStack platform integrated with PAS consumes less power than the initial platform.

When there are many virtual machine immigrations, the scheduling result is not correct. Additionally, PAS is suitable for computing intensive scenarios [14]. The following research will focus on a stable PAS which is efficient for both virtual machine immigrations and floating system load scenarios [15].

Acknowledgment

The research work is done under the guide of my mentor Zou Hengming. The testing environments are provided by System Software Lab in Shanghai Jiaotong University.

References

[1] Open source software for building private and public clouds, OpenStack open source cloud computing software. From http://www.openstack.org/

[2] Beloglazov, A., Buyya, R., Lee, Y. C., & Zomaya, A. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers, 82(2),* 47-111.

[3] Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. *The New York Times, 49(3).*

[4] Van, H. N., Tran, F. D., & Menaud, J. M. (2010, July). Performance and power management for cloud infrastructures. *Proceedings of IEEE 3rd International Conference on Cloud Computing (CLOUD)* (pp. 329-336).

[5] Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., & Zhao, F. (2008, April). Energy-aware server provisioning and load dispatching for connection-intensive internet services. *NSDI*, *8,* 337-350.

[6] Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, 100-108.

[7] Armstrong, P., Agarwal, A., Bishop, A., Charbonneau, A., Desmarais, R., Fransham, K., & Sobie, R. J. (2010). Cloud Scheduler: A resource manager for distributed compute clouds. *ArXiv Preprint arXiv: 1007.0050*.

[8] Lucas, S. J. L., Moreno-Vozmediano, R., Montero, R. S., & Llorente, I. M. (2011, July). Dynamic placement of virtual machines for cost optimization in multi-cloud environments. *Proceedings of International Conference on High Performance Computing and Simulation* (pp. 1-7).

[9] Danish, M., Li, Y., & West, R. (2011, April). Virtual-CPU scheduling in the quest operating system. *Proceedings of 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (pp. 169-179).

[10] Kansal, A., Zhao, F., Liu, J., Kothari, N., & Bhattacharya, A. A. (2010, June). Virtual machine power metering and provisioning. *Proceedings of the 1st ACM Symposium on Cloud Computing* (pp. 39-50).

[11] Infrastructure, V. (2006). Resource management with vmware drs. *VMware Whitepaper*.

[12] Kim, K. H., Beloglazov, A., & Buyya, R. (2009, November). Power-aware provisioning of cloud resources for real-time services. *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science* (p. 1).

[13] Beloglazov, A., & Buyya, R. (2010, May). Energy efficient allocation of virtual machines in cloud data

centers. *Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing* (pp. 577-578).

[14] Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M. Q., & Pentikousis, K. (2010). Energy-efficient cloud computing. *The Computer Journal*, *53(7),* 1045-1051.

[15] Lee, Y. C., & Zomaya, A. Y. (2012). Energy efficient utilization of resources in cloud computing systems. *The Journal of SUPERCOMPUTING*, *60(2),* 268-280.

Wu Li was born in 1989 and got the bachelor degree in Wuhan University in Wuhan, China in 2012 majoring in software engineering. He is pursuing his master degree in Shanghai Jiaotong University, in Shanghai majoring in software engineering.

His mainly research areas include system software, cloud computing and storage. He once worked as a software development intern in Alibaba Corporation and Intel Corporation. His current research concentrates on Microsoft Azure Cloud platform for big data.

Mr. Wu is now working with Microsoft to apply natural language processing for Microsoft Azure and make the advantage of extension of cloud computing and cloud storage.