Volume 6, Issue 7, July 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Enhanced Energy Aware Virtual Machine Consolidation and Live Migration Considering Multiple Resources

[1]B. Venkata Krishnakanth, [2]G. Vijay Kumar
[1] M Tech IInd Year , Department of CSE, SKU College of Engineering, Anantapur, Andhra Pradesh, India
[2] Lecturer, Department of CSE, SKU College of Engineering, Anantapur, Andhra Pradesh, India

Abstract: ***In the present era consolidation of virtual applications in cloud computing environment enables a significant chance of energy optimization. Cloud Computing acts as source for provisioning of computer resources to end user by pooling servers which are geographically distributed, in the form of accessible resources. This paper addresses the problem of energy inefficiency in the data centers where the ideal power is wasted when the servers run at low utilization. We present an operational model for the live migration of the Virtual Machines in cloud that uses load balancing algorithm for efficient energy optimization. The simulation results show the efficiency of the operational model when compared with the previous model.***

Index Terms: server consolidation, cloud architecture, virtual machines, application scaling

I. INTODUCTION

Today, utility computing, envisioned by John Mc. Carthy1 and others is a social and technical reality, but cloud computing technology must evolve to avoid becoming the victim of its own success. At this time, the average cloud server utilization is low, while the power consumption of clouds based on over-provisioning is excessive and has a negative ecological impact [1]. We live in a world of limited resources and cloud over-provisioning is not sustainable either economically or environmentally. If successful, self-organization would allow cloud servers to operate more efficiently and thus, reduce costs for the Cloud Service Providers (CSPs), provide an even more attractive environment for cloud users, and support some form of interoperability. The pressure to provide new services, better manage cloud resources, and respond to a broader range of application requirements is increasing, as more US government agencies are encouraged to use cloud services . We have known for years that distributed systems which maintain state information are neither scalable nor robust; this is the reason why most Internet services are delivered by stateless servers. We have also known for some time that collecting state information consumes a fair share of system resources and that system management decisions based on obsolete state information are far from optimal; this knowledge is critical for the communication and computing infrastructure built around the Internet but resource management in cloud computing is still based on hierarchical control models where state information is maintained at several levels. We have also known that assembling large collections of systems each with a small, but finite probability of failure requires special design principles to guarantee availability. For several decades we have designed and built heterogeneous computing systems with very large numbers of components interacting with each other and with the environment in intricate ways [2]. The complexity of such systems is undeniable but their design was, and still is, based on traditional, mostly deterministic, system organization and management. This has to change, but the path to change is strenuous. Hypothesis of our research is that self-organization could potentially solve the major challenges related to resource management in large-scale systems. Self-organization is the spontaneous emergence of global coherence out of local interactions. A self-organizing system responds to changes in the environment through adaptation, anticipation, and robustness. The system reacts to changes in the environment, predicts changes and reorganizes itself to respond to them, or is robust enough to sustain a certain level of perturbations [3]. We argue that self-organization could be critical for the future of cloud computing and it is feasible and effective.

II. RELATED WORK

In the last few years packaging computing cycles and storage and offering them as a metered service became a reality. Large farms of computing and storage platforms have been assembled and a fair number of Cloud Service Providers (CSPs) offer computing services based on three cloud delivery models SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service) [1].

2.1 Development in Provisioning Cloud services

Cloud elasticity, the ability to use as many resources as needed at any given time, and low cost, a user is charged only for the resources it consumes, represents solid incentives for many organizations to migrate their computational activities to a public cloud For example, in 2007 the EC2 [5] was the first service provided by AWS; five years later, in 2012, AWS was used by businesses in 200 countries. Amazon's S3 (Simple Storage Service) has surpassed two trillion

objects and routinely runs more than 1.1 million peak requests per second. Elastic MapReduce has launched 5.5 million clusters since May 2010 when the service started [4].

2.2 Energy Consumption.

The rapid expansion of the cloud computing has a significant impact on the energy consumption in US and the world. The costs for energy and for cooling large-scale data centers are significant and are expected to increase in the future. In 2006, the 6 000 data centers in the U.S. reportedly consumed 61 × 109 kWh of energy, 1.5% of all electricity consumption in the country, at a cost of $4.5 billion [6]. The energy consumption of data centers and of the network infrastructure is predicted to reach 10, 300 TWh/year (1 TWh = 109 kWh) in 2030, based on 2010 efficiency levels [7]. These increases are expected in spite of the extraordinary reduction in energy requirements for computing activities. Idle and under-utilized servers contribute significantly to wasted energy. A 2010 survey [8] reports that idle servers contribute 11 million tons of unnecessary CO2 emissions each year and that the total yearly costs for idle servers are $19 billion. Recently, Gartner Research [9] reported that the average server utilization in large data-centers is 18%, while the utilization of x86 servers is even lower, 12%. These results confirm earlier estimations that the average server utilization is in the 10 − 30% range [10].

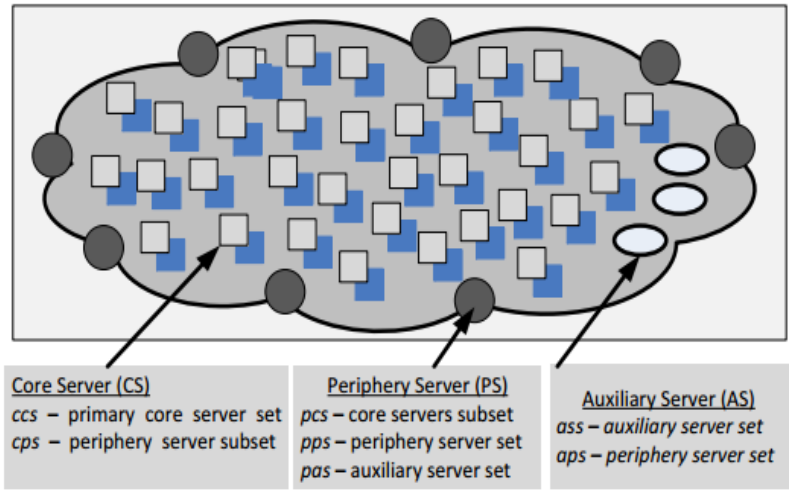
III. SYSTEM MODEL

We present Novel cloud architecture based on the coalition formation and combinatorial auction. The basic tenets of the model include autonomy of individual components, self-awareness and intelligent behavior of individual components.

Self-awareness enables servers to create coalitions of peers working in concert to respond to the needs of an application, rather than offering a menu of a limited number of resource packages.

3.1 Cloud organization:

We distinguish between a core server and a cloud periphery server as shown in Figure 1. The core consists of a very large number of computational and storage servers, CS, dedicated to the cloud mission - the provision of services to the user community The core servers are typically heterogeneous [11]; co-processors and are attached to multi/many core processors.



The cloud consists of a core and a periphery populated by core and periphery servers, respectively; a group of auxiliary servers support internal services. Some of the data structures used for self-awareness are shown:

- **ccs** - the set of primary contacts for a core server and **cps** - the subset of periphery servers known to core server;
- **pcs** - the subset of core servers known to the periphery server**, pps** - the set of all periphery servers, and **pas** - the set of all auxiliary servers; and
- **as** - the set of auxiliary servers and **aps** -the set of periphery servers known to an auxiliary server.

These data structures are populated during the initial system configuration and updated throughout its lifetime.
A relatively small number of periphery servers, PS, known to the outside world, act as access points to the core and as the nerve center of the system. There are also a few, auxiliary servers, AS, providing internal cloud services..

3.2 Operation modes

The cloud computing infrastructure is reconfigurable; a core server is an autonomous entity that can operate in several modes:

(i) MODE 1- M1: the server is configured to support multiple virtual machines, as in existing clouds.

(ii)MODE 2- M2: the server is either the leader or a member of coalition of core servers running under the control of one OS designed to support Single System Image (SSI) operation. This mode will be able to support data-intensive applications which require a very large number of concurrent threads/processes of one application.

(**iii) MODE 3- M3**: the server is either the leader or a member of coalition of core servers where each server runs under the control of one of the operating systems supported by the cloud. This mode is of particular interest for data-intensive applications with a complex workflow involving multiple applications and under strict privacy and security constraints; it is inspired by the clusters supported by AWS and by the MaaS (Metal as a Service) model.
(iv) MODE 4- M4: operation as a storage server.

3.3 Cloud core: the self-configuring monitor (ScM).

The cloud core consists of computational and storage servers providing services to the user community. To address the natural tension between self-awareness and the desire to maintain minimum state information, each core server has a relatively small set of primary contacts, and, when necessary, it has access to a much larger pool of secondary contacts. Secondary contacts, are core servers [12] that can be accessed via the periphery servers known to a core server.

The ScM is a Virtual Machine Monitor (VMM) , its main function is to configure and supervise the system, rather than tightly control the virtual machines (VMs) running on the physical server

The ScM is a component of the software stack running on each core server at all times; its main functions are:

1. To respond to external events following on a set of policies and goals; e.g., choose the operation mode of the server in response to a service request and configure the server accordingly.
2. To evaluate the performance of the server relative to its long- and short-term goals after the completion of a service request. To maintain information about the environment and the past system history such as: the primary contacts, the energy consumption, the ratio of successful versus failed bids for service, the average system utilization over a window of time, the effectiveness of the cost model used for bidding, and all other data relevant for the future behavior of the server.
3. To evaluate the behavior of the application, whether the information in the service request is adequate and the resulting SLA adopted after the bid was successful.
4. To support system security and control communication with the external world. The effectiveness of the model depends on the ability of the ScM to make most decisions using local information thus, minimizing communication with other systems. Minimal intrusiveness is another critical requirement for the ScM; this implies that the ScM should monitor system events by observing control data structures, rather than being directly involved in the flow of control and reacting only to prevent policy violations.

Cloud periphery. A self-organizing cloud includes a relatively small number of periphery servers. The cloud periphery plays a critical role in a self-organizing system, it acts like the nervous system of the cloud and it links the core to the outside world and with the internal services. This strategy allows the core servers to maintain a minimum amount of data about the external and the internal environment and should be dedicated to their main mission of providing services to the user community. At the same time, they should be more agile and able to respond to unforeseen events, to accommodate software updates, and to balance the load placed on the auxiliary servers (AS). The main functions of a periphery server are:

1. To provide an interface of the cloud with the outside world.
2. To aid in self-organization, self-management, and self-repair.
3. To act as a broker - sending service requests, to a subset of core servers and then forwarding bids from core servers to the entity requesting service. Finally, they could be used to mediate the creation of a Service Level Agreement (SLA) between the two parties.

Auxiliary servers. Support internal services such as: accounting, billing, policy and goal management, system statistics, patch management, software license management, and reputation and trust services.

3.4 Distinguishing features of the model

The system is reconfigurable - the autonomous computational and storage servers can operate in several modes to maximize QoS [13]. Coalitions of core servers are created dynamically to respond to service requests. Winning coalitions are determined by the results of auctions.

Virtualization simulates the interface of an object by several means including multiplexing, aggregation, emulation, and multiplexing combined with emulation. While exiting clouds are based exclusively on multiplexing - multiple virtual machines sharing the same server - in our model, in addition to multiplexing, we consider aggregation - the creation of a single virtual server from several physical servers, and federation of servers.

A service request is formulated in a Cloud Service Description Language (CSDL). Once a bid is selected a client-specific, legally-binding Service Level Agreement (SLA) can be negotiated between the cloud service provider and the cloud client. At the completion of a service request the system evaluates the performance of the service provision, the client and the application. This information can be fed back into future service negotiations.

IV. LOAD BALANCING AND VM CONSOLIDATION

To migrate the VM's form the over loaded server we need to identify the overloaded host where the CPU utilization of that particular host crosses the threshold value which could be defined as static for this purpose. The process includes:

1. Detecting the over loaded host
2. Policies to Select the VM's for Migration

3. Target Server for VM to Migrate
4. Detecting Under loaded host

4.1 Detecting the overloaded Host

Three methods are suggested in the literature for adjustment of upper threshold based on historical data of CPU utilization: Median Absolute Deviation (MAD) [10], Interquartile Range (IQR) [14] and Local regression (LR) [11].

Performance evaluation has shown that LR outperforms MAD and IQR [10]. Hence we use LR for deciding whether host is overloaded or not. LR [11] is proposed by Cleveland. The main idea of this method is fitting simple models such as straight line or some well known curve to localized subsets of data to build up a curve that approximates the original data.

4.2 Policies to Select the VM's for Migration

After the host is found to be overloaded, the next step is to select the VMs to be migrated away from that host. There are various polices for this, namely Random Selection (RS) policy, Maximum correlation policy (MC) and Minimum Migration Time (MMT).

Random selection (RS) [10] as the name suggests selects VM at random from the host selected for VM migration. This policy is fairly simple to implement and also running time is quite low.

Maximum correlation (MC) policy is based on the idea proposed by Verma et al. [15]. The higher is correlation between the resource usage by applications running on an oversubscribed server, the higher is the probability of the server overloading. According to this idea, we select those VMs to be migrated that have the highest correlation of the CPU utilization with other VMs. Details of calculation of correlation can be found in Verma et al. [14]. It is shown in [10] that MC is not as good as MMT.

Minimum Migration Time (MMT) [10] policy migrates a VM _v' that requires the minimum time to complete a migration relatively to the other VMs allocated to the host. The migration time is estimated as the amount of RAM utilized by the VM _v' divided by the spare network bandwidth available for the host j currently hosting VM _v'. Let Vj be the set of VMs currently allocated to the host j.

4.3 Target Server for VM to Migrate

Once a VM is selected for migration we need to find a host for migrating the VM. We should take care that the target host should not get overloaded after we place our selected VM.

1. *Average CPU, memory and network utilization of all servers in datacenter is calculated (cpuA, memA, netA).*
2. *For the host under consideration/candidate find its cpu, memory and network utilization (cpuUtil, netUtil and memUtil).*
3. *Calculate integrated average of cpu, memory and network utilizations :*
   iA = (cpuUtil + netUtil + memUtil) / 3
4. *Integrated load imbalance of host is given by,*
   $ILB = \frac{(iA - cpuA) + (iA - memA) + (iA - netA)}{3}$
5. *Now we select host with minimum ILB value given by equation 4.4*

Algorithm 1 : To Select Target Host

Host will not be considered as candidate if

a. After assignment it is getting overloaded.

b. Host is not suitable for VM in terms of resource specifications.

4.4 Detecting Under loaded host

The reason for finding an underloaded host is to move its VMs to other hosts so that this underloaded host can be put to sleep thus saving energy/power

Following is an approach for finding an underloaded host and its VMs migration [10].

1. *Find all overloaded hosts (overHosts) using overloaded host detection algorithm described in Section 4.1. We regard all hosts which are not _overHosts' as _candidateHosts'.*
2. *From _candidateHosts' find a host i which has lowest CPU utilization among all candidateHosts.*
3. *Let targetList = cadidateHosts – host i.*

4. *Try to migrate a VM from host i to a host from _targetList'. Similarly migrate all VMs from host i to hosts from _targetList'. Migration is possible if the target host has sufficient resource requirements (CPU, memory etc.) for the VM under migration.*
5. *If all the VMs can be migrated from host i to hosts from _targetList' then the host i is put to sleep else host i is kept activ*

V. PERFORMANCE ANALYSIS AND RESULTS

It is impractical to experiment with a very large number of physical systems, so we chose numerical simulation to investigate the feasibility of some of the ideas discussed in this chapter. We believe, that at this stage, the emphasis should be on qualitative rather than quantitative results. Thus, our main concern when choosing the parameters of the simulation was to reflect the scale of the system and "typical" situations. The very large number of core servers we chose to experiment with forced us to consider simpler versions of the protocols for bidding. The simulation experiments reported in this section run on the Amazon cloud. Several storage optimized hi1.4xlarge instances1 running on a 64-bit architecture were used.

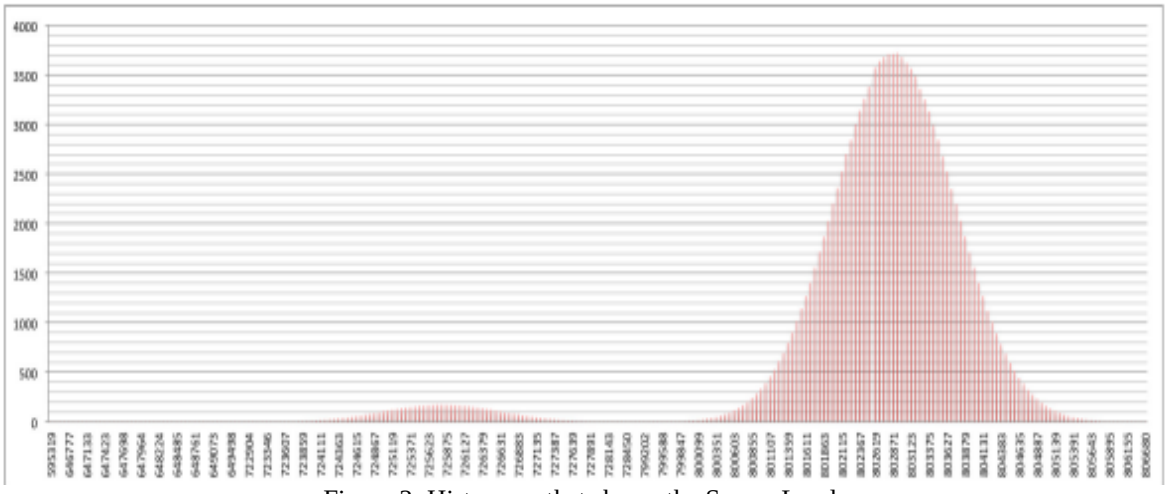


Figure 2: Histogram that shows the Server Load

This choice was motivated by the scale of the system we simulated; the description of the very large number of core servers requires a very large address space thus, systems with access to a very large physical memory. The simulation ran concurrently on 16 virtual cores (vCPUs) delivering 35 ECUs (Elastic Cloud Units) and used 60.5 GB of main memory. The instances were launched on the same cluster and servers were connected by a non-blocking 10 Gbps Ethernet. The simulation, implemented in java, used extensively multi-threading to reduce the execution time; each one of the 16 virtual cores ran a single thread.

VI. CONCLUSION AND FUTURE WORK

The IaaS cloud delivery model, and the Amazon Web Services in particular, support not only a cost-effective, but also a very convenient and elastic computing environment. Any proposal for a novel cloud computing delivery model has to address the feasibility, the performance, and the cost involved. Numerical simulation is widely used in science and engineering for getting insights into physical phenomena, for checking the accuracy and limitations of theoretical models, for testing hypothesis, or for comparing different design options and parameters of the systems we plan to build. In all these cases, we start with a model, an abstraction of a physical system or a phenomena, then we carry out the simulation based on this model and, finally, we validate the simulation results by comparing them with theoretical predictions and, whenever feasible, with measurements of the physical system.

REFERENCES

[1] Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility‖. Future Generation Computer Systems 2009, vol. 25(6), pp. 599-616.

[2] ASHRAE Technical committee 99: ―Datacom equipment power trends and cooling applications 2005.

[3] Barroso LA, Holzle U. ―The case for energy-proportional computing . 2007, vol. 40(12), pp. 33-37.

[4] Fan X, Weber WD, Barroso La. ―Power provisioning for a warehouse-sized computer‖. Proceedings of the 34ths Annual international symposium on computer architecture (ISCA 2007), ACM New York, NY, USA, 2007, pp. 13-23.

[5] Ranganathan P, Leech P, Irwin D, Chase J. ―Ensemble-level power management for dense blade servers‖. Proceedings of the 33rd International symposium on computer architecture (2006), Boston, MA, USA, 2006, pp. 66-77.
[6] B. Addis, D. Ardagna, B. Panicucci, M. Squillante, and Li Zhang. A hierarchical approach for the resource management of very large cloud platforms. IEEE Transactions on Dependable and Secure Computing, 10(5):253–272, 2013.
[7] Amazon Web Services, Inc. Amazon Elastic Compute Cloud User Guide (API Version 2014- 06-15), 2014.
[8] M. Andreolini, S. Casolari, and S. Tosi. A hierarchical architecture for on-line control of private cloud-based systems. In Proc. of 10th World Wide Web Internet Conference (WWWCONF2010), Timisoara, Romania, October 2010.
[9] R. Abbott. "Complex systems engineering: putting complex systems to work." Complexity, 13(2):10–11, 2007.
[10] R. Albert, H. Jeong, and A.-L. Barab´asi. "The diameter of the world wide web." Nature, 401:130–131, 1999.
[11] R. Albert, H. Jeong, and A.-L. Barab´asi. "Error and attack tolerance of complex networks." Nature, 406:378–382, 2000.
[12] D. Ardagna, M. Trubian, and L. Zhang. "SLA based resource allocation policies in autonomic environments." J. Parallel Distrib. Comp., 67(3):259–270, 2007.
[13] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. "Energy-aware autonomic resource allocation in multi-tier virtualized environments." IEEE Trans. on Services Computing, 5(1):2–19, 2012.
[14] J. Y. Arrasjid et. al. VMware vCloud Architecture Toolkit. VMware Press, Upple Saddle River NJ, 2013.
[15] M. Auty, S. Creese, M. Goldsmith, and P. Hopkins. "Inadequacies of current risk controls for the cloud." Proc. IEEE 2nd Int. Conf. on Cloud Computing Technology and Science, pp. 659–666, 2010.

AUTHOR'S BIBLIOGRAPHY

G. Vijay Kumar has obtained his Bachelor of Technology degree in computer science and information technology, affiliated to JNTUH Hyderabad in 2008 and Master of Technology degree in computer science from JNTUA college of Engineering autonomous Ananthapuramu in 2012.
He is currently serving as lecturer in Sri Krishnadevaraya university. He has 7 years of teaching experience to both undergraduate and postgraduate students. His current interest include in security in cloud computing and network security.
Mail-id: vijaykumar.gvk.@gmail.com
Mobile No: 9885887469

B. Venkata Krishna Kanth received the B.Tech(computer science and engineering)degree from the Audisankara institute of technology, gudur, affliated to JNTUA Ananthapuramu in 2014 and presently he is doing M.tech (computer science) degree from sri krishnadevaraya university in anantapur and working on his thesis work. His area of interest is cloud computing and network security and data mining Attachments area.
Mail-id: bkrishnakanth1@gmail.com
Mobile No: 7660807484

