User Centric Scaling System for Dynamic Cloud Resource Sharing Environment

N.Aishwarya,P.Sivaranjani

Abstract— **With the proliferation of web services providing the same functionality, researches about practicability and adaptive capability of web services selection mechanism have gained considerable momentums. This paper presents a dynamic web services composition algorithm based on the optimal path method of the weighted directed acyclic graph. Experiment results show that the algorithm can accelerate the speed of service selection, and bring great benefit to the application of web services composition. There are two directions in our future work: one is to search for the method of implementing run-time whereas, due to underutilization of feedback information, the massive useless redundant iterations result in low solution efficiency of the algorithm. The current approaches can solve some key issues in web services composition and give great illuminations to this paper; however, none of them gives an effective solution to address the issues of low efficiency in the large solution space. In this paper, ant colony algorithm [9] and genetic algorithm [10] combination based algorithm (ACAGA_WSC) are presented. By means of genetic algorithm, ACAGA_WSC overcomes the shortcomings of the ant colony algorithm, and achieves better efficiency and converging speed. Experiment shows that the new algorithm gives better performance for the services composition problem.**

Index Terms— **Cloud computing, virtualization, auto scaling, green computing**

I. INTRODUCTION

Cloud Computing is a novel paradigm for the provision of computing infrastructure, which aims to shift the location of the computing infrastructure to the network in order to reduce the costs of management and maintenance of hardware and software resources. Cloud computing has a service-oriented architecture in which services are broadly divided into three categories: Infrastructure-as-a- Service (IaaS), which includes equipment such as hardware, Storage, servers, and networking components are made accessible over the Internet; Platform-as-a-Service (PaaS), which includes hardware and software computing platforms such as virtualized servers, operating systems, and the like; and Software-as-a-Service (SaaS), which includes software applications and other hosted services. Cloud computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as you-go model to consumers. These services in industry are respectively referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). A Berkeley Report in Feb 2009 states "Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service" [2]

Clouds aim to power the next generation data centers by architecting them as a network of virtual services (hardware, database, user-interface, application logic) so that users are able to access and deploy applications from anywhere in the world on demand at competitive costs depending on users QoS (Quality of Service) requirements [9]. Developers with innovative ideas for new Internet services no longer require large capital outlays in hardware to deploy their service or human expense to operate it [2]. It offers significant benefit to IT companies by freeing them from the low level task of setting up basic hardware (servers) and software infrastructures and thus enabling more focus on innovation and creating business value for their services.

To obtain accurate estimation of the complete probability distribution of the request response time and other important performance indicators. The model allows cloud operators to determine the relationship between the number of servers and input buffer size, on one side, and the performance indicators such as mean number of tasks in the system, blocking probability, and probability that a task will obtain immediate service, on the other.

Middleware clustering technology capable of allocating resources to web applications through dynamic application instance placement is introduced and evaluated. Application instance placement is defined as the problem of placing application instances on a given set of server machines to adjust the amount of resources available to applications in response to varying resource demands of application clusters. The objective is to maximize the amount of demand that may be satisfied using a configured placement. To limit the disturbance to the system caused by starting and stopping application instances, the placement algorithm attempts to minimize the number of placement changes

The Providers such as Amazon [10], Google [11], Salesforce [12], IBM, Microsoft [13], and Sun Microsystems have begun to establish new data centers for hosting Cloud computing application services such as social networking and gaming portals, business applications (e.g., SalesForce.com), media content delivery, and scientific workflows. Actual usage patterns of many real-world application services vary with time, most of the time in unpredictable ways. To illustrate

Manuscript received Mar, 2015.

N.Aiswarya, *Department of Computer Science and Engineering, Anna University/ Rajalakshmi Engineering College, Chennai, India.*

P.Sivaranjani, *Department of Computer Science and Engineering, Anna University/ Rajalakshmi Engineering College, Chennai, India.*

ISSN: 2278 – 1323

this, let us consider an "elastic" application in the business/social networking domain that needs to scale up and down over the course of its deployment. Social networking websites are built using multi-tiered web technologies, which consist of application servers such as IBM Web Sphere and persistency layers such as the MySQL relational database. Usually, each component runs in a separate virtual machine, which can be hosted in data centers that are owned by different cloud computing providers. Additionally, each plug-in developer has the freedom to choose which Cloud computing provider offers the services that are more suitable to run his/her plug-in. As a consequence, a typical social networking web application is formed by hundreds of different services, which may be hosted by dozens of Cloud data centers around the world. Whenever there is a variation in temporal and spatial locality of workload, each application component must dynamically scale to offer good quality of experience to users.

Cloud Architectures are designs of software applications that use Internet-accessible on-demand services. Applications built on Cloud Architectures are such that the underlying computing infrastructure is used only when it is needed (for example to process a user request), draw the necessary resources on-demand (like compute servers or storage), perform a specific job, then relinquish the unneeded resources and often dispose themselves after the job is done. While in operation the application scales up or down elastically based on resource needs.

Cloud Architectures address key difficulties surrounding large-scale data processing. In traditional data processing it is difficult to get as many machines as an application needs. Second, it is difficult to get the machines when one needs them. Third, it is difficult to distribute and coordinate a large-scale job on different machines, run processes on them, and provision another machine to recover if one machine fails. Fourth, it is difficult to auto scale up and down based on dynamic workloads. Fifth, it is difficult to get rid of all those machines when the job is done. Cloud Architectures solve such difficulties.

There are some clear business benefits to building applications using Cloud Architectures. A few of these are listed here:

A) Almost zero upfront infrastructure investment: If you have to build a large-scale system it may cost a fortune to invest in real estate, hardware (racks, machines, routers, backup power supplies), hardware management (power management, cooling), and operations personnel. Because of the upfront costs, it would typically need several rounds of management approvals before the project could even get started. Now, with utility-style computing, there is no fixed cost or startup cost.

B) just-in-time Infrastructure: In the past, if you got famous and your systems or your infrastructure did not scale you became a victim of your own success. Conversely, if you invested heavily and did not get famous, you became a victim of your failure. By deploying applications in-the-cloud with dynamic capacity management software architects do not have to worry about pre-procuring capacity for large-scale systems. The solutions are low risk because you scale only as you grow. Cloud Architectures can relinquish infrastructure as quickly as you got them in the first place (in minutes).

C) More efficient resource utilization: System administrators usually worry about hardware procuring (when they run out of capacity) and better infrastructure utilization (when they have excess and idle capacity). With Cloud Architectures they can manage resources more effectively and efficiently by having the applications request and relinquish resources only what they need (on-demand).

D) Usage-based costing: Utility-style pricing allows billing the customer only for the infrastructure that has been used. The customer is not liable for the entire infrastructure that may be in place. This is a subtle difference between desktop applications and web applications. A desktop application or a traditional client-server application runs on customer's own infrastructure (PC or server), whereas in a Cloud Architectures application, the customer uses a third party infrastructure and gets billed only for the fraction of it that was used.

E) Potential for shrinking the processing time: Parallelization is the one of the great ways to speed up processing. If one compute-intensive or data intensive job that can be run in parallel takes 500 hours to process on one machine, with Cloud Architectures, it would be possible to spawn and launch 500 instances and process the same job in 1 hour. Having available an elastic infrastructure provides the application with the ability to exploit parallelization in a cost-effective manner reducing the total processing time.

II. RELATED WORK

Hypervisor-based fault tolerance (HBFT), which synchronizes the state between the primary VM and the backup VM at a high frequency of tens to hundreds of milliseconds, is an emerging approach to sustaining mission-critical applications. Based on virtualization technology, HBFT provides an economic and transparent fault tolerant solution. However, the advantages currently come at the cost of substantial performance overhead during failure-free, especially for memory intensive applications. This paper presents an in-depth examination of HBFT and options to improve its performance. Based on the behavior of memory accesses among check pointing epochs, Two optimizations, read-fault reduction and write-fault prediction, for the memory tracking mechanism. These two optimizations improve the performance by 31 percent and 21 percent, respectively, for some applications. Then, present software super page which efficiently maps large memory regions between virtual machines (VM). Our optimization improves the performance of HBFT by a factor of 1.4 to 2.2 and achieves about 60 percent of that of the native VM. [3]

Resource allocation in computing clusters is traditionally centralized, which limits the cluster scale. Effective resource allocation in a network of computing clusters may enable building larger computing infrastructures. This problem is considered as a novel application for multiagent learning (MAL). MAL algorithm is proposed and applies it for optimizing online resource allocation in cluster networks. The learning is distributed to each cluster, using local information only and without access to the global system reward. Experimental results are encouraging: our multiagent learning approach performs reasonably well, compared to an optimal

ISSN: 2278 – 1323

solution, and better than a centralized myopic allocation approach in some cases. [4]
Resource management poses particular challenges in large-scale systems, such as server clusters that

simultaneously process requests from a large number of clients. A resource management scheme for such systems must scale both in the in the number of cluster nodes and the number of applications the cluster supports. Current solutions do not exhibit both of these properties at the same time. Many are centralized, which limits their scalability in terms of the number of nodes, or they are decentralized but rely on replicated directories, which also reduces their ability to scale. In this paper, we propose novel solutions to request routing and application placement two key mechanisms in a scalable resource management scheme. Our solution to request routing is based on selective update propagation, which ensures that the control load on a cluster node is independent of the system size. Application placement is approached in a decentralized manner, by using a distributed algorithm that maximizes resource utilization and allows for service differentiation under overload. The paper demonstrates how the above solutions can be integrated into an overall design for a peer-to-peer management middleware that exhibits properties of self-organization. Through complexity analysis and simulation, we show to which extent the system design is scalable. We have built a prototype using accepted technologies and have evaluated it using a standard benchmark. The testbed measurements show that the implementation, within the parameter range tested, operates efficiently, quickly adapts to a changing environment and allows for effective service differentiation by a system administrator [5].
Resource management poses particular challenges in large-scale systems, such as server clusters that simultaneously process requests from a large number of clients. A resource management scheme for such systems must scale both in the in the number of cluster nodes and the number of applications the cluster supports. Current solutions do not exhibit both of these properties at the same time. Many are centralized, which limits their scalability in terms of the number of nodes, or they are decentralized but rely on replicated directories, which also reduces their ability to scale. In this paper, we propose novel solutions to request routing and application placement two key mechanisms in a scalable resource management scheme. Our solution to request routing is based on selective update propagation, which ensures that the control load on a cluster node is independent of the system size. Application placement is approached in a decentralized manner, by using a distributed algorithm that maximizes resource utilization and allows for service differentiation under overload. The paper demonstrates how the above solutions can be integrated into an overall design for a peer-to-peer management middleware that exhibits properties of self-organization. Through complexity analysis and simulation .A prototype is built using accepted technologies and has evaluated it using a standard benchmark. The testbed measurements show that the implementation, within the parameter range tested, operates efficiently, quickly adapts to a changing environment and allows for effective service differentiation by a system administrator. [6]

Given a set of machines and a set of Web applications with dynamically changing demands, an online application placement controller decides how many instances to run for each application and where to put them, while observing all kinds of resource constraints. This NP hard problem has real

usage in commercial middleware products. Existing approximation algorithms for this problem can scale to at most a few hundred machines, and may produce placement solutions that are far from optimal when system resources are tight. In this paper, we propose a new algorithm that can produce within 30 seconds high-quality solutions for hard placement problems with thousands of machines and thousands of applications. This scalability is crucial for dynamic resource provisioning in large-scale enterprise data centers. Our algorithm allows multiple applications to share a single machine, and strives to maximize the total satisfied application demand, to minimize the number of application starts and stops, and to balance the load across machines. Compared with existing state-of-the-art algorithms, for systems with 100 machines or less, our algorithm is up to 134 times faster, reduces application starts and stops by up to 97%, and produces placement solutions that satisfy up to 25% more application demands. Our algorithm has been implemented and adopted in a leading commercial middleware product for managing the performance of Web applications. [7]

III. PROPOSED SCHEME

In Proposed scheme, the task is sent to the cloud center is serviced within a suitable facility node; upon finishing the service, the task leaves the center. A facility node may contain different computing resources such as web servers, database servers, directory servers, and others. A service level agreement, SLA, outlines all aspects of cloud service usage and the obligations of both service providers and clients, including various descriptors collectively referred to as Quality of Service (QoS). QoS includes availability, throughput, reliability, security, and many other parameters, but also performance indicators such as response time, task blocking probability, probability of immediate service, and mean number of tasks in the system, all of which may be determined using the tools of queuing theory. That task will be processed in corresponding cloud server based on user category where scaling depend on it.
Cloud server system is modeled which indicates that the inter arrival time of requests is exponentially distributed, while task service times are independent and identically distributed random variables that follow a general distribution with mean value of u. The system under consideration contains m servers which render service in order of task request arrivals (FCFS).The capacity of system is m þ r which means the buffer size for incoming request is equal to r. As the population size of a typical cloud center is relatively high while the probability that a given user will request service is relatively small, the arrival process can be modeled as a efficient process. Color set algorithm is proposed to decide the application Allocation or request placement and the load distribution.

ISSN: 2278 – 1323

IV. ARCHITECTURAL DESIGN

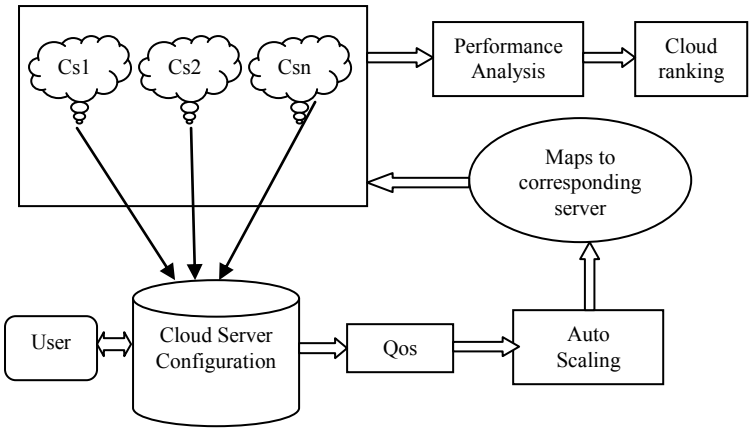

Fig.1 System Architecture

A. Clustered Classification

The server calculates which cloud doing which job. That is Monitoring cloud access, cost calculation and equal sharing of jobs in cloud. Here multiple servers formed into cluster formation. A group application into service performance of classes which are then mapped onto server clusters which parses application level information in Web requests and forwards them to the servers with the corresponding applications running. The switch sometimes runs in a redundant pair for fault tolerance. Each application can run on multiple server machines and the set of their running instances are often managed by some clustering software.

B. Differentiated Services

After the servers may be clustered then allocate the task which can be assigned to each server and calculate the performance and priority. Each server machine can host multiple applications. The applications store their state information in the backend storage servers. It is important that the applications themselves are stateless so that they can be replicated safely.

C. Load Shifting

The load of data center applications can change continuously. We only need to invoke our algorithm periodically or when the load changes cross certain thresholds. Hence, if a flash crowd requires an application to add a large number of servers; all the servers are started in parallel. Our algorithm is highly efficient and can scale to tens of thousands of servers and applications. The amount of load change during a decision interval may correspond to the arrivals or departures of several items in a row. A large load unit reduces the overhead of our algorithm because the same amount of load change can be represented by fewer items. It also increases the stability of our algorithm against small oscillation in load. On the other hand, it can lead to inefficient use of server resources and decrease the satisfaction ratio of application demands.

D. Auto Scaling

The space timing calculates by the reference of cloud usage. That is, the cost also calculates based on cloud space utilization and cloud usage. The server calculates which cloud doing which job. That is monitoring cloud access, cost calculation and equal sharing of jobs in cloud. We analyze and compare the performance offered by different configurations of the computing cluster, focused in the execution of loosely coupled applications. Different cluster configurations with different number of worker nodes from the three clouds Providers and different number of Jobs (depending on the cluster size), as shown in the definition of the different cluster configurations, we use the following acronyms. The use of large-scale distributed systems is enabled for task-parallel applications, which are linked into useful workflows through the looser task coupling model of passing data via files between dependent tasks and potentially larger class of task-parallel Feature Extraction.

E. Resource Utilization

There are also some cloud vendors providing auto-scaling solutions for cloud users .Users are allowed to define a set of rules to control the scaling actions. However, the rules and the load balancing strategies they used are very simple. They perform the scaling actions simply when some conditions are met and balance the load evenly across all instances. Since they do not take the state of the whole system into consideration, they cannot reach a globally optimal decision. They allocate resources on shared cluster serves periodically.

V. CONCLUSION

The design and implementation of a system is presented that can scale up and down the number of application instances automatically based on demand. Color set algorithm is developed to decide the application placement and the load distribution. The system achieves high satisfaction ratio of application demand even when the load is very high. It saves energy by reducing the number of running instances when the load is low.

ACKNOWLEDGMENT

We would like to sincerely thank Assistant Prof. P.Sivaranjani for his advice and guidance at the start of this article. His guidance has also been essential during some steps of this article and his quick invaluable insights have always been very helpful. His hard working and passion for research also has set an example that we would like to follow. We really appreciate his interest and enthusiasm during this article. Finally we thank the Editor in chief, the Associate Editor and anonymous Referees for their comments.

REFERENCES

[1] Dynamic Placement for Clustered Web Applications A.Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. University of California at Berkley, USA. Technical Rep UCB/EECS-2009-28, 2009.

ISSN: 2278 – 1323

[3] Optimizing the Performance of Virtual Machine Synchronization for Fault Tolerance Jun Zhu, Zhefu Jiang, Zhen Xiao and Xiaoming Li

[4] A Multi-Agent Learning Approach to Online Distributed Resource Allocation Chongjie Zhang, Victor Lesser, Prashant Shenoy

[5] Service Middleware for Self-Managing Large-Scale Systems Constantin Adam, Rolf Stadler.

[6] On Balancing the Load in a Clustered Web Farm JOEL L. WOLF AND PHILIP S. YU, IBM T. J.

[7] A Scalable Application Placement Controller for Enterprise Data Centers Chunqiang Tang, MalgorzataSteinder, Michael Spreitzer, and Giovanni Pacifici

[8] Resource Overbooking and Application Profiling in Shared Hosting Platforms Resource Overbooking and Application Profiling in Shared Hosting Platforms

[9] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation Computer Systems, 25(6): 599-616, Elsevier Science, Amsterdam, The Netherlands, June 2009.

[10] Amazon Elastic Compute Cloud (EC2), http://www.amazon.com/ec2/ [17 March 2010].

[11] Google App Engine, http://appengine.google.com [17 March 2010].

[12] Salesforce.com (2009) Application Development with Force.com's Cloud Computing Platform http://www.salesforce.com/platform/. Accessed 16 December 2009

[13] Windows Azure Platform, http://www.microsoft.com/azure/ [17 March 2010].

AUTHORS PROFILE

N.Aishwarya is currently a PG scholar in Computer Science from the Department of Computer Science and Engineering at Rajalakshmi Engineering College, Chennai and Tamilnadu. She received his M.Sc Degree in Computer Science and Technology from SRM Arts and Science College, Chennai and Tamilnadu. Her Research areas include Cloud Computing, Grid Computing and Distributed System.

R.Sivaranjani is currently working as a Asst.Professor from the Department of Computer Science and Engineering at Rajalakshmi Engineering College, Chennai and Tamilnadu. She received his Bachelor Degree in Computer Science and Engineering from Bharath Institute of Science and Technology, Chennai and Tamilnadu. She received his master degree from GKM College of Engineering and Technology, Chennai and Tamilnadu. Her main research interests lie in the area of Wireless Sensor Networks, Network Security and Adhoc Networks.

ISSN: 2278 – 1323