International Journal of Recent Engineering Research and Development (IJRERD)
Volume No. 02 – Issue No. 02, ISSN: 2455-8761
www.ijrerd.com, PP. 51-59

Analysis Of SLA-Aware Multi Level Scheduling In Federated Cloud Environment

Sruthi K K
PG Student
Department of Information Technology
Government Engineering College,Idukki,India

K R Remesh Babu
Assistant Professor
Department of Information Technology
Government Engineering College,Idukki,India

Abstract: The heart of computing is the available resources and no matter what type of environment is. Cloud is one of the ongoing research area which provides better and flexible computing with novel features. The main attractive part of cloud computing is the way of approach to the resource scarcity. Even in the cloud environment the availability of resources at the right moment is the big issue and the solution for this leads to the idea federated cloud. In federated cloud the resources will have the ability to adjust their capabilities according to sudden variations in service demands. In this paper the importance of federation is studied, surveyed different approaches to the scalability problem. This work is basically focused on to create a federated cloud environment for the purpose of auto scaling and the new scheduling method is also incorporated. Along with this, SLA violation minimization is considered for better service.

Keywords: Cloud Computing, federation ,Scaling

I. INTRODUCTION

Cloud computing is a new terminology which is emerged from Distributed, Parallel and Grid computing. . Cloud computing offers end consumers a pay as go model .Cloud computing provides different types of resources like hardware and software as services via internet. With cloud computing, user can increase the computing power and storage. Also variety of software and hardware services can be utilized from the wide range of available applications on the basis of pay-per-use, irrespective of the location. Cloud computing can also helps reduce the overall computing and servicing cost. These characteristics attract an increasing number of customers and corporations to rent cloud service to run their applications.

Cloud infrastructure providers (IaaS providers) have established data centers in multiple geographical locations to serve a large number of software and hardware application service consumers from around the world, and it provide redundancy and ensure reliability in case of site failures. This approach has many issues , which include the difficulty for Cloud customers to pre-determine the best location for hosting their services as they may not know origin of consumers of their services and Cloud SaaS providers may not be able to meet QoS requirements of their service consumers from the multiple geographical locations. To meet QoS requirements of Cloud customers ,a building mechanisms is needed for seamless federation of datacenters of a Cloud provider or providers supporting dynamic scaling of applications across multiple domains . A single Cloud infrastructure provider will not be able to establish their data centers at all possible locations throughout the world. Which means Cloud application service (SaaS) providers will have difficulty in meeting QoS requirements for all their consumers. Hence, make the use of services of multiple Cloud infrastructure service providers who can provide better service for their specific consumer needs helpful to solve this issue. This kind of requirements often arises in firms with global operations and applications such as Internet service, media hosting, and Web applications. This shows the necessity of building mechanisms for federation of Cloud infrastructure service providers for better provisioning of services across different Cloud providers. There are many challenges and issues involved in the creation of such Cloud interconnections through federation process. To meet these requirements, the Cloud service providers should be able to dynamically expand or resize their capabilities based on sudden variations in workload demands by leasing the available computational capabilities and storage resources from other Cloud service providers. It should operate as parts of a market driven resource leasing federation and host their services based on Service Level Agreement (SLA) contracts driven by competitive market .The SLA is about prices; and deliver on demand, reliable, QoS aware and cost effective services based on virtualization technologies while ensuring high minimizing service costs and QoS standards . For provisioning of federated

hardware infrastructure and virtualized software services among users with heterogeneous applications and QoS targets, They need to be able to utilize market-based utility models as the basis [1].

Managing cloud computing elasticity is typically connected with each application task and it is the process of mapping performance requirements of the task to the underlying available resources. This process of getting resources to the on-demand requirements of an application called scaling, and the scaling process can be very challenging. Resource under-provisioning will inevitably hurt performance and create SLA violations, while resource over-provisioning can result in idle instances, thereby incurring unnecessary costs. The first thought could lead us to plan capacity for the average load or for the high load. When planed for the average load, there is less cost incurred, but performance will be a problem if high load occurs. Bad performance will discourage customers, and rating will be affected. On the other hand if capacity is planned for the high workload, the resources will remain usable most of the time. Therefore, it seems the necessary of more sophisticated technique for resource allocation, that automatically scales resources according to demand. These are called auto-scaling techniques. To date, cloud researchers have pursued schedule-based and rule-based mechanisms to attempt to automate this matching between computing requirement and computing resources. Schedule-based techniques take into account the cyclical pattern of the daily workload. These scaling actions are configured manually, based on the time of the day, so the system cannot adapt to the unexpected changes in the load. For this reason, schedule-based techniques will not be discussed in this work.

II. LITERATURE REVIEW

The paper [8] presents vision, challenges, and architectural elements of InterCloud for utility-oriented federation of Cloud computing environments. The proposed environment supports scaling of applications across multiple vendor clouds known as InterCloud. Paper contains the validation of approach by conducting a set of rigorous performance evaluation study using the CloudSim toolkit. The results shows that federated Cloud computing model has immense potential to provide
significant performance gains as related to response time and cost under dynamic workload scenarios.

The paper [4] says that the cloud federation helps to enables cloud providers and IT companies to collaborating and sharing their resources, that are facing with many portability and Interoperability issues. The paper analyzed that, Cloud de researchers and developers have researched or implemented various federation architectures, including cloud bursting, brokering, and aggregation. These architectures are classified according to the coupling level or interoperation among the cloud instances involved. The cloud instances ranging from loosely coupled that have no or little inter operability among cloud instances to tightly coupled that have full interoperability among cloud instances. The cloud hybrid architecture combines the existing private cloud infrastructure with remote resources from one or more public clouds to provide extra capacity to satisfy peak demand periods. The central component of broker architecture is a broker that serves various users and has access to several public cloud infrastructures. Cloud aggregation architecture consists of two or more partner clouds that interoperate to aggregate their resources and provide users with a larger virtual infrastructure. The multitier architecture, consists of two or more cloud sites, each running its own cloud OS and usually belonging to the same corporation, that are managed by a third cloud OS instance following a hierarchical arrangement.

Gang Chen [5] explored the challenges of building a real cloud system . Most existing cloud systems use an idealized representation of the cloud , which is of value in initial conceptualization in an algorithm development . However , one has to go beyond this ideal environment to to build and manage a real system. This paper also considered the issue in deploying multiple data management systems on a cloud infrastructure. The main highlighted portion of the work is resource sharing and data processing issues introduced by hosting a federated management system.

In [6] Lena Mashayekhy proposed a mechanism that improves the cloud providers dynamic resource scaling capabilities to fulfil users demands.The paper proposed a cloud federation formation game that characterizes the process of federation formation and then proposed a novel cloud federation formation mechanism. In the proposed mechanism, cloud providers dynamically cooperate to form a federation in order to provide the requested resources to a user. The resources are provisioned as VM instances of different types. The proposed mechanism forms cloud federations yielding the highest total profit. In addition, the proposed mechanism produces a stable cloud federation structure, that is, the participating cloud providers in the federation do not have incentives to break away from the federation. In this work, they performed extensive experiments to investigate the properties of proposed mechanism. The results showed that the proposed mechanism is able to form stable federations with total profit very close to the optimal profit. In addition , the proposed mechanism finds the stable cloud federation in a reasonable amount of time making it suitable for real cloud settings. For the future work, they planned to incorporate the data privacy concerns into the federation formation problem and to investigate the influence of cloud providers policies on the federation formation process.

In [7] Javier Diaz-Montes proposed the CometCloud which aims to provide infrastructure and programming support for enabling such workflws via flexible, software-defied synthesis of custom cyber infrastructure through the autonomic, on -demand federation of geographically distributed compute and data resources.CometCloud is an autonomic framework designed to enable highly heterogeneous, dynamically federated computing and data platforms that can support end-to-end application workflws with diverse and changing requirements.This occurs through autonomic, on-demand federation of geographically distributed compute and data resources as applications need them, and by exposing the federation using elastic cloud abstractions and science-as-a-service platforms. Consequently, CometCloud can create a nimble and programmable environment that autonomously evolves over time, adapting to changes in infrastructure and application requirements.

The Paper[9] introduced FederatedCloudSim, a very flexible cloud simulation framework that can be used to simulate many federated cloud scenarios while respecting SLAs (service level agreements). In this paper we also present first simulation results and explain different simulation scenarios. FederatedCloudSim is a new software for simulating such cloud federations. It is based on CloudSim 3.0.3, a widely used java framework for simulating distributed clouds . Leaving the original CloudSim unchanged, the code is provided as an extra package adding new classes and often extending existing cloudsim classes using inheritance.

III. PROPOSED SYSTEM DESIGN

To manage heterogeneous resources in cloud computing environment in an optimized way of resource management is required. The dynamic nature of the cloud environment makes it as a challenging problem. The efficient and effective task scheduling algorithm not only aims to reduce job completion time but has to consider QoS requested by the customer. An efficient task scheduling strategy must also aim to yield less response time so that the execution of a submitted task takes place within a minimum time and reduce rejection rate.

The main Objective of this work is to propose a new scheduling mechanism for cloud, and to create a federated environment or scaling process. Along with this, SLA violation minimization is also considered. Auto Scaling of cloud resources is the important thing which is going to be considered in this work .Auto scaling of virtual machines and datacenters will be done based on the load. The Fig. 6 is the proposed architecture of the work.

A. Design Components

The proposed design consists of four main components such as Coordinator, Job Scheduler, Resource scaling and SLA Monitor. All the components have specific functionalities.

1) Cloud Coordinator

The Cloud Coordinator is the main component of this architecture and have the responsibility for the management of Clouds and their membership to the overall federation. The Cloud Coordinators are able to exports the services of a cloud to the federation by implementing basic functionalities for resource management such as scheduling, allocation, virtualization, dynamic sensing/monitoring etc.

2) Job Scheduler

This component allocates virtual machines to the Cloudlets based on user's QoS requirements and the present resource availability. The main job of the scheduler is to rearrange the cloudlets based on priority value and then schedule them according to the priority

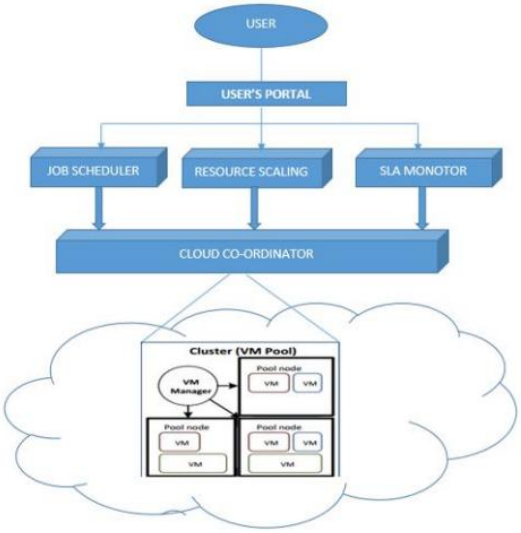

Fig. 6 Proposed System Design

3) Resource Scaling Resource Scaling

As the name indicated, the main function of this module is the scaling process.Because of the unpredictable dynamic load, the resources can't serve them within the SLA and some violations will occur due to the insufficient resources.In this case,the federated cloud environment allow to lease the resources from other providers.The management of this process is the main function of Resource Scaling module.

4) SLA Monitor

The funcion of SLA monitor component is to monitoring of the service request and the QoS requirements and make the positive step to serve feasibly.The SLA module stores the service terms and conditions that are being supported by the Cloud and Cloud Broker on a per user basis.

B. User Defined Multl level Scheduling

The working of user defined scheduling algorithm is based on the priority of each job. Here the priority value is completely depends on the user interest. User can specify their requirements through user's portal. Once they specified their requirements, they are navigated to the other gui for setting the priority to the specified jobs. Here the values from the user portal will saved for further movements. The algorithm takes the job list and the corresponding priority value as input. Then initialize a new list for storing the jobs in the order of priority. The loop will take the first job as the current job, then the priority value of the current job will compared to all other jobs .If the current job has the higher priority than all other jobs, then the current job is added to the new list. Otherwise the loop will continue without adding the current job to the list. The algorithm will stop when there in no jobs for comparison.

Algorithm: Degree of Satisfaction of SLA Calculation

- Start
- Create a list of jobs
- For all jobs in the list do
  - Assign priority to the jobs /* priority given by the user */
  - If *Tex > Td*, go to step 6
  - Create a new list for adding the jobs based on priority
  - Compare the current job's priority with others
    - If priority of current job is > others, go to step 6
    - *Else*
      - Set next job as current job, go to step 5
- Add job to the list
- Stop

C. Proposed Degree of Satisfaction of SLA Calculation

The objective proposed SLA monitoring algorithm is to check the SLA satisfaction of services. Every user have their on requirements, so that the satisfaction of every user requirements is desirable.

Input to the algorithm is the requested cloudlets in the case of simulation and deadline values. Here the deadline values means the deadline to the execution time ***Td*** .The execution time ***Tex*** compares with the ***Td.***

If Tex >Td Then

SLA violation occurred

Else

Successful execution

The SLA violation is calculated according to the execution time taken by the cloud to run the specific tasks. The comparison with the deadline gives whether the particular execution satisfied the SLAs. The algorithm Degree of Satisfaction of SLA Calculation is proposed for finding the number of cloudlets which are violated the execution deadline and to calculate the degree of satisfaction by using the following equations.

$$degreeofSatisfactionofSLA = \frac{NumberofCloudletsExceedstheDeadline}{TotalNumberofCloudlets}$$

The Degree of Satisfaction of SLA is represented as *Ds* ,Number of Cloudlets Exceeds the deadline is *Nv* and The Total Number of Cloudlets *Nt.*The above equation can be rewrite in to the following form.

$$Ds=Nv/Nt \quad (1)$$

Algorithm: Degree of Satisfaction of SLA Calculation

- Start
- INPUT : cloudlets(n),***Td*** */* Finish time is taken as deadline */*
- OUTPUT : *Nv* ,*Ds /* Count of deadline violated cloudlets */*
- For all items in cloudlets(n) do
  - *Tex*=getFinishTime(n);
  - If *Tex* > *Td*, go to step 6
    - *Nv* = *Nv* + 1
    - continue the loop with next item
  - Else
    - Continue the loop with next item
  - *Ds=Nv / Nt*
- Stop

IV. EXPERIMENTAL SETUP AND RESULT

To evaluate the performance of federated cloud, the result were Simulated in Windows 8 (64 bit),i3 processor,2350 M processor ,2.3 GHz of speed with 4GB memory.

The work is implemented on Cludsim with jdk 1.7.The implementation work is done on the basis of firstly the NetBeans 8.0.2 is installed successfully and then CloudSim is imported into the NetBeans. After the installation the CloudSim will run successfully and finally the algorithm and scaling process implemented and run into the cloud. The inputs are taken from the graphical user interface provided to the user.

The two phases of the work is analysed in this section. A federated cloud environment is simulated for running the experiments. The process federation is evaluated under different cloud conditions which are created by varying the number of DCs and Vms.The obtained output shows how the process federation is helpful to maintain the cloud service capability. This section analysed the Multi - Level scheduling policy.The scheduling purely based on user demand on the jobs.The scheduling part of the work is based on the following, firstly all the requirements should be analysed then serving of the cloudlets based on that requirements is the next step.The priority to the cloudlets can set directly or by random value.first level scheduling is based on the priority of the cloudlets and then scheduled them either by using TimeShared Policy or SpaceShared Policy

The table 2 is the comparison table that shows the makespan for executing different number of cloudlets in different situation. For getting this results, different number of cloudlets are evaluated under scaling environment and without scaling environment. The cloud resource specification should be same in order to evaluate it. The number of DC is 2 and each DC consists of 2 hosts. Each host having different power elements, one is dual core machine and another one is quad core machine. And the cloud consists of total 4 virtual machines so that each DC consists of 2 VMs.

TABLE 2 : COMPARISON OF MAKESPAN (2 DC)

No. of Cloudlets	With Scaling (ms)	Without Scaling(ms)
10	9.71	20.1
20	18.28	40.1
30	27.37	60.1
50	45.55	100.1
70	63.74	140.1
100	91.01	200.1
150	136.46	300.1
200	181.92	400.1

The process of federation will be more easier as well as powerful when the number of DCs are sufficient. The table 3 is the comparison table for analyzing the efficiency of the process federation in the case of 5 DCs. This particular cloud environment consists of total 5 DCs and each DC having 2 hosts. Total 10 Vms are created which means each DC consists of 2 VMs .In this case the process federation will be more powerful because of the availability of sufficient resources.

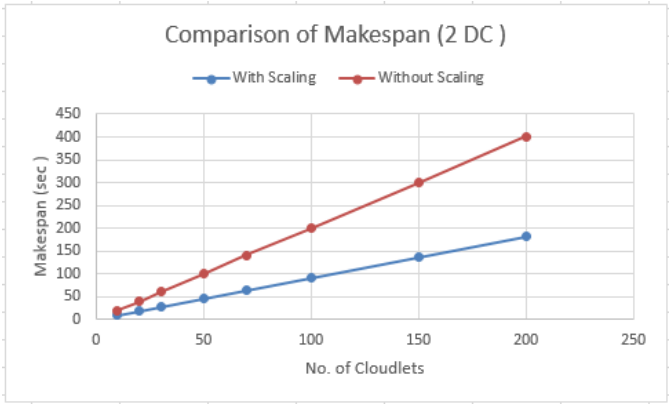

Fig. 7 Comparison of Makespan (2 DC)

TABLE 3 : COMPARISON OF MAKESPAN (5 DC)

No. of Cloudlets	With Scaling (ms)	Without Scaling(ms)
10	5.55	12.09
20	9.19	20.1
30	14.65	32.1
50	23.74	52.05
70	32.83	72.04
100	45.56	100.1
150	61.19	151.96
200	91.01	100.1

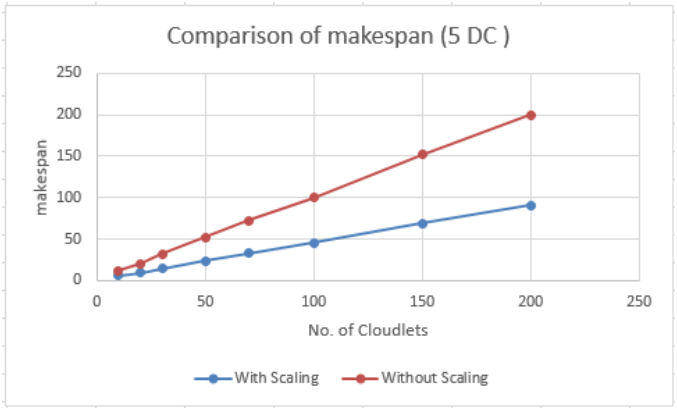

Fig. 8 Comparison of Makespan (5 DC)

The figure 9 shows the analysis of makespan of serving the cloudlets. The execution time taken by different number of cloudlets are analyzed in both scaled and non scaled environment. The cloud environment is fixed for every analysis. Cloud consists of 2 datacenters 4 virtual machines and 2 hosts in every datacenter. The graph shows the analysis graph from that it is very clear that, the cloud without scaling consume more time to serve the cloudlets.

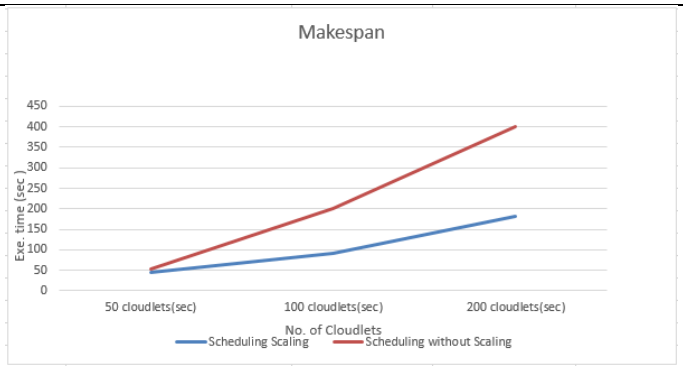

Fig. 9 Comparison of Makespan

The figure 10 represents the comparison of scaling process in different scaling environment. This part of experiment is used to analyze the variation in the scaling process when number of DC is different. The graph shows that when the number of Data centers increases, the scaling process will be more flexible and it results to better performance.

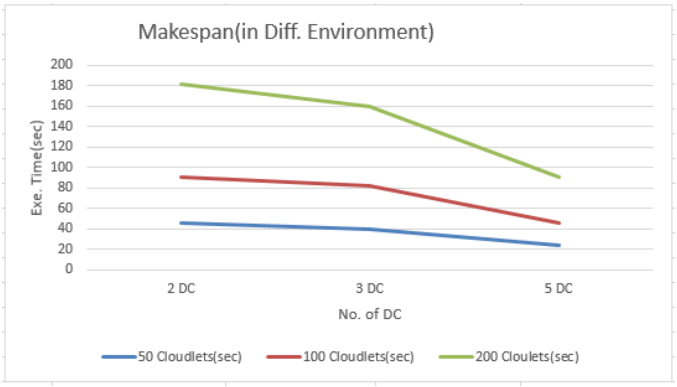

Fig. 10 Makespan in Different Environment

The table 4 and figure 11 representing the analysis of degree of satisfaction of SLA when the number of resources in the environment changed. Here the SLA requirements are Execution time deadline, MIPS rate and the RAM capacity. The graph shows that, when the number of DC increases it results to the better SLA satisfaction because of adequate available resources. The graph shows the degree of satisfaction of SLA for serving 200 cloudlets in the cloud with scaling environment and without scaling environment. The deadline for this particular experiment is 160 seconds. Only 174 cloudlets are served within the deadline time in the cloud with 2 DCs and 87 percent of requirement is satisfied. The cloud with 3 DCs give much more degree of satisfaction around 89 percent. The results shows that the cloud with 5 DC's give around 100 percent satisfaction in this particular case.

TABLE 4 : DEGREE OF SATISFACTION

NO. of DCs	Degree of Satisfaction (%)
2	87
3	89
5	100

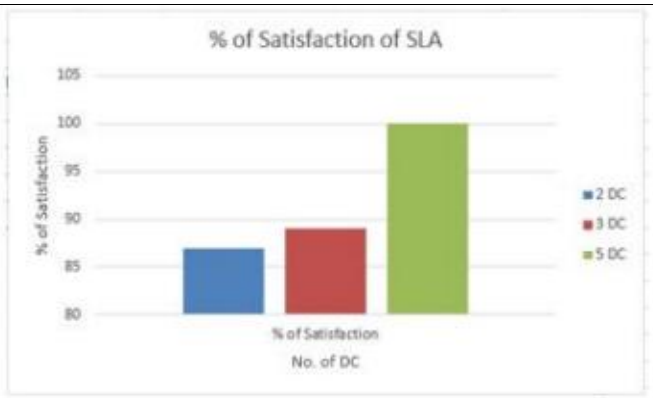

Fig. 11 Degree of Satisfaction

It is observed from the above result is, the process federation is more suitable for the cloud environment because of its dynamic nature. The experiment is conducted on both scaled and unscaled environment. The results shows that the federated environment is giving the better performance and the degree of satisfaction of SLA is also high.

V. CONCLUSIONS AND FUTURE SCOPE

In past few years, cloud computing has attracted an increasing amount of attention from the digital world. Job Scheduling is the most important step in cloud computing environment because user demand may vary upon time and have to pay for resources used based on utilization. Hence efficient utilization of resources depends on the task scheduling and it must be properly handled in order to get maximum benefit from the resources. A new scheduling mechanism named Multi Level User Defined Scheduling is implemented for cloud based on users QoS requirements.The most important part of the work is the creation of Federated Cloud Environment. A federated cloud is successfully created and the performance of scaling in the federated cloud is also evaluated and compared with the cloud without scaling environment. The experimental result shows that the federated cloud is more suitable for the cloud environment because of its dynamic nature. This work aims to provide the motivation to the readers to think about the importance of federation process in cloud computing and highlighted some keys for the future works.

REFERENCES

[1] Rajkumar Buyya, Rajiv Ranjan,"Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities", Grid Computing and Distributed Systems (GRIDS) Laboratory Department of Computer Science and Software Engineering The University of Melbourne, Australia.

[2] Tania Lorido-Botran, Jose Miguel-Alonso,"Autoscaling Techniques for Elastic Applications in Cloud Environments",Department of Computer Architecture and Technology University of the Bas que Country September 5, 2012.

[3] Hanieh Alipour, Yan Liu,"Analyzing Auto-scaling Issues in Cloud Environments".

[4] Rafael Moreno-Vozmediano, Ruben S. Montero,"IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures",Published by the IEEE Computer Society,DECEMBER 2012.

[5] Gang Chen,"Federation in Cloud Data Management: Challenges and Opportunities",IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 7, JULY 2014.

[6] Lena Mashayekhy,"Cloud Federations in the Sky: Formation Game and Mechanism",IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 3, NO. 1, JANUARY-MARCH 2015.

[7] Javier Diaz-Montes, Moustafa AbdelBaky,"CometCloud Enabling Software-Defied Federations for End-to-End Application Workflws",Published by the IEEE Computer Society,January/February 2015.

[8] Rajkumar Buyya,Rajiv Ranjan,"InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services",loud Computing and Distributed Systems (CLOUDS) Laboratory Department of Computer Science and Software Engineering The University of Melbourne, Australia.

[9] Andreas Kohne,"FederatedCloudSim: A SLA-aware Federated Cloud Simulation Framework",acm.org. "CrossCloud Brokers "14 December 8, 2014, Bordeaux, France

[10] Peiravi, Mashhadi, HamedJavadi,An optimal energy efficient clustering method in WSN using multi objective genetic algorithm.
[11] D. Jiang, G. Chen, B. C. Ooi, K.-L. Tan, and S. Wu, epiC: An extensible and scalable system for processing big data, in Proc. Int. Conf. Very Large Data Bases, 2014, pp. 541552.
[12] B. Cui, H. Mei, and B. Ooi, Big data: The driver for innovation in databases, Nat. Sci. Rev., vol. 1, no. 1, pp. 2730, 2014.
[13] U. Hoelzle and L. A. Barroso, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 1st ed. San Rafael, CA, USA: Morgan and Claypool Publishers, 2009.
[14] (2014) [Online]. Available: http://aws.amazon.com/ec2/
[15] M. Stonebraker, The case for shared nothing, IEEE Database Eng. Bull., vol. 9, no. 1, pp. 49, Mar. 1986.
[16] H. T. Vo, S. Wang, D. Agrawal, G. Chen, and B. C. Ooi, LogBase: A scalable log-structured database system in the cloud, Proc. VLDB Endowment, vol. 10, pp. 10041015, 2012
[17] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, Mesos: A platform for finegrained resource sharing in the data center, in Proc. 8th USENIX Conf. Netw. Syst. Des. Implementation, 2011, pp. 2235.
[18] (2013) [Online]. Available: http://hadoop.apache.org/docs/ current/hadoop-project-dist/hadoop hdfs/Federation.html.
[19] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, Disklocality in datacenter computing considered irrelevant, in Proc. 13th USENIX Conf. Hot Topics Operating Syst., 2011, pp. 1216.
[20] M. Al-Fares, A. Loukissas, and A. Vahdat, A scalable, commodity data center network architecture, in Proc. ACM SIGCOMM Conf. Data Commun., 2008, pp. 6374.
[21] H. Zhang, B. Tudor, G. Chen, and B. Ooi, Efficient inmemory data management: An analysis, in Proc. Int. Conf. Very Large Data Bases, 2014, pp. 833836.
[22] J. Ousterhout, P. Agrawal, D. Erickson, The case for RAMClouds: Scalable high-performance storage entirely in DRAM, ACM SIGOPS Operating Syst. Rev., vol. 43, no. 4, pp. 92105, Jan. 2010.
[23] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation, 2012, pp. 215.