IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. I (Mar-Apr. 2014), PP 94-101
www.iosrjournals.org

Survey and Comparative Study on Resource Allocation Strategies in Cloud Computing Environment

[1]Vinayak Awasare, [2]Sudarshan Deshmukh
[1]PG Student of PCCOE, University of Pune Maharashtra India
[2]Assistant Professor, PCCOE, University of Pune, Maharashtra, India

Abstract: *Cloud computing is an essential ingredient of modern computing systemsCloud computing provides an on demand service because it offers dynamic resource allocation for reliable and highly available services in pay as-you-consume manner to public. In Cloud computing environment multiple cloud users can request number of cloud services in parallel. So there must be a provision that all resources which are made available to requesting user in efficient manner to satisfy their need. In this Survey paper a review of various strategies for dynamic resource allocation in cloud computing is shown based on Linear Scheduling Strategy for Resource Allocation, Topology Aware Resource Allocation (TARA) and Dynamic Resource Allocation for Parallel Data Processing. Moreover limitations significance and advantages of using Resource Allocation in Cloud computing systems is also discussed.*

Index Terms: *Cloud Computing, Dynamic Resource Allocation, Resource Management, Resource Scheduling*.

I. INTRODUCTION

Cloud Computing is an essential ingredient of modern computing systems. Computing concepts, technology and architectures have been developed and consolidated in the last decades; many aspects are subject to technological evolution and revolution. Cloud Computing is an computing technology that is rapidly consolidating itself as the next step in the development and deployment of increasing number of distributed application. Cloud computing is nothing but a specific style of computing where everything from computing power to infrastructure, business apps are provided as a service. It's a computing service rather than a product. In cloud, shared resources, software and information is provided as s metered service over the network. When the end user accesses some service is cloud, he is not aware of where that service is coming from or what is platform being used or where it is being stored.

Cloud computing platforms, such as those provided by Google, IBM Microsoft, Amazon,etc, let developers deploy applications across computers hosted by a central server. So these all applications can access a large network of computing resources that are deployed and managed by a cloud provider. Software Developers obtain the advantages of a managed computing platform, without having to commit resources to build and maintain the network. One important problem that must be addressed effectively in the cloud is how to manage QoS and maintain SLA for cloud users that share cloud resources.

The cloud computing paradigm makes the resource as a single point of access to the number of clients and is implemented as pay per use basis. Though there are number of advantages of cloud computing such as virtualized environment, equipped with dynamic infrastructure, pay per consume, totally free of software and hardware installations, prescribed infrastructure and the major concern is the order in which the requests are satisfied which evolves the scheduling of the resources. Allocation of resources has been made efficiently that maximizes the system utilization and overall performance. Cloud computing is mainly sold or rented on demand on the basis of time constrains basically specified in hours or minutes. So the scheduling has to be done in such a way that the resource utilization has need done efficiently.

In cloud computing environment, resource allocation or load balancing takes place at two levels. First, when an application is uploaded to the cloud, the load balancer assigns the requested process to physical computers, attempting to balance the computational load of multiple applications across physical computers. Second, when an application receives multiple incoming requests, these requests should be each assigned to a specific requested application instance to balance the computational load across a set of instances of the same requested application. For example Amazon EC2 uses elastic load balancing (ELB) to control how incoming requests are handled. Application designers can direct requests to instances in specific availability zones, to instances demonstrating the shortest response times or to specific instances.

In the following sections a review of existing resource allocation techniques like Linear Scheduling, Topology Aware Resource Allocation and Resource Allocation for parallel data processing is described briefly.

II. RESOURCE ALLOCATION & ITS SIGNIFICANCE

Resource allocation is process of assigning the available resources in an economic way and efficient way Resource allocation is the scheduling of the available resources and available activities required by those activities while taking into consideration both the resource availability and the project time. Resource provisioning and allocation solves that problem by allowing the service providers to manage the resources for each individual request of resource.

Resource Allocation Strategy (RAS) is all about the number of activities for allocating and utilizing scarce resources within the limit of cloud environment so as to meet the needs of the cloud application. It requires the type and amount of resources needed by each application in order to complete a user job

From the perspective of a cloud provider, predicting the dynamic nature of users, user demands, and application demands are impractical. For the cloud users, the number of tasks of job needs to be completed on time with minimal cost. Hence due to limited resources, resource heterogeneity, environmental necessities, locality restrictions and dynamic nature of resource demand, we need an efficient resource allocation system that suits cloud environments.

Cloud resources consist of virtual resources. The physical resources are shared across multiple computer quests through virtualization and provisioning. The virtualized resources are described through a set of parameters detailing the processing, memory and disk needs. Provisioning of cloud can be done by mapping virtualized resources to physical ones. The software and hardware resources are allocated to the cloud applications on-demand basis [1].

III. RELATED WORK

Dynamic resource allocation problem is one of the most important and challenging problem in the resource management problems of cloud computing environment. The dynamic resource allocation in cloud computing environment has attracted attention of researches in the last few years. Many researchers those working on cloud computing around the world have come up with new ways of facing this challenge. Dynamic resource allocation is one of the most challenging problems in the resource scheduling problems. The dynamic resource allocation in cloud infrastructure has capture attention of the number of research community in the last decade. Many researchers around the world have given number of solution for this challenging problem i.e. dynamic resource allocation in cloud infrastructure.

Now a day's number of growing companies has popularized an architectural paradigm based on a huge number of commodity servers. Problems like regenerating a web index or processing crawled documents are split into several independent subtasks, distributed among the available nodes, and computed in parallel. Simplify the development of such number of distributed applications on top of these architectures; some of the cloud provider companies have also built customized data processing frameworks. Examples are Google's Map Reduce, Yahoo's Map-Reduce-Merge etc.

In [8] authors have explained the algorithm for negotiation protocol for resource provisioning in detail. In [1], authors have made a comparison of many resource allocation strategies. In [9] authors propose a model and a utility function for location-aware dynamic resource allocation. Comparison of number of available resource allocation policies is covered in [10]. In [11] author has used a Genetic Algorithm for scheduling of tasks in cloud computing systems. This paper is not only intended to address any specific resource allocation strategy, but to provide a review of some of the existing resource allocation techniques.

Nephele is data processing framework used for dynamic resource allocation offered by todays Infrastructure-as-a-Service **(**IaaS) clouds for both, task scheduling and task execution. Some tasks of a job can be assigned to different types of virtual machines (VMs) which are automatically started and terminated during the job execution. This literature survey focuses on resource allocation mechanisms and its impacts on cloud providers and cloud users. This survey paper would greatly benefit the cloud users and researchers.

IV. RESOURCE ALLOCATION STRATEGIES & ALGORITHM

Recently many resource allocation strategies have come up in the literature of cloud computing environment as this technology has started maturing. Number of Researcher communities around the world have proposed and implemented several types of resource allocation. Some of the strategies for resource allocation in cloud computing environment are discuss here briefly.

A. Topology Aware Resource Allocation (TARA)

Different types of resource allocation strategies are proposed in cloud. The author mentioned in [2] the architecture for resource allocation in Infrastructure-as-a-Service (IaaS) based cloud systems. Current Infrastructure-as-a-Services of cloud providers are usually unaware of the hosted application's requirements and therefore allocate resources independently of its needs, which can significantly impact performance for distributed data-intensive applications.

An architecture which adopts a "what if" methodology to guide allocation decisions taken by the IaaS is proposed to address this resource allocation problem. The architecture uses a prediction engine with a lightweight simulator to estimate the performance of a given resource allocation and a algorithm to find an optimized solution in the large search space. Results showed that Topology Aware Resource Allocation reduced the job completion time of these applications by up to 59% when compared to application-independent allocation policies.

1) Architecture of TARA: Topology Aware Resource Allocation [2] is composed of two major parts: a prediction engine and a fast genetic algorithm-based search technique. The prediction engine is the entity responsible for optimizing resource allocation. When it receives a resource request, the prediction engine iterates through the possible subsets of available resources (each distinct subset is known as a candidate) and identifies an allocation that optimizes estimated job completion time. However, even with a lightweight prediction engine, exhaustively iterating through all possible candidates is infeasible due to the scale of IaaS systems. Therefore a genetic algorithm-based search technique that allows TARA to guide the prediction engine through the search space intelligently is used.

2) Prediction Engine: The prediction engine maps resource allocation candidates to scores that measures their "fitness" with respect to a given objective function, so that TARA can compare and rank different candidates. The inputs used in the scoring process can be seen in Figure1, Architecture of TARA.

3) Objective Function: The objective function defines the metric that TARA should optimize. For example, given the increasing cost and scarcity of power in the data center, an objective function might measure the increase in power usage due to a particular allocation.

4) Application Description: The application description consists of three parts: i) the framework type that identifies the framework model to use, ii) workload specific parameters that describe the particular application's resource usage and iii) a request for resources including the number of VMs, storage, etc.

5) Available Resources: The final input required by the prediction engine is a resource snapshot of the IaaS data centre. This includes information derived from both the virtualization layer and the IaaS monitoring service. The information gathered ranges from a list of available servers, current load and available capacity on individual servers to data centre topology and a recent measurement of available bandwidth on each network link.

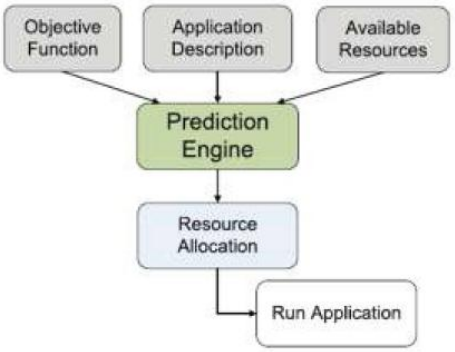

Fig. 1 Basic Architecture of TARA [2]

B. Linear Scheduling Strategy for Resource Allocation

Considering the processing time, resource utilization based on CPU usage, memory usage and throughput, the cloud environment with the service node to control all clients request, could provide maximum service to all clients [3]. Scheduling the resource and tasks separately involves more waiting time and response time. A scheduling algorithm named as Linear Scheduling for Tasks and Resources (LSTR) is designed, which performs tasks and resources scheduling respectively. Here, a server node is used to establish the IaaS cloud environment and KVM/Xen virtualization along with LSTR scheduling to allocate resources which maximize the system throughput and resource utilization.

Resource consumption and resource allocation have to be integrated so as to improve the resource utilization. The scheduling algorithms mainly focus on the distribution of the resources among the requestors that will maximize the selected QoS parameters. The QoS parameter selected in our evaluation is the cost

function. The scheduling algorithm is designed considering the tasks and the available virtual machines together and named LSTR scheduling strategy. This is designed to maximize the resource utilization.

Algorithm [3]:

1) The requests are collected between every predetermined interval of time
2) Resources Ri => {R1,R2, R3,...,Rn}
3) Requests RQi => {RQ1, RQ2, RQ3,..., RQn}
4) Calculate Threshold (static at initial)
5) Th = Σ Ri
6) for every unsorted array A and B
7) sort A and B
8) for every RQi
9) if RQi <Th then
10) add RQi in low array, A[RQi]
11) else if RQi >Th then
12) add RQi in high array B[RQi]
13) for every B[RQi]
14) allocate resource for RQi of B
15) Ri = Ri - RQi; Th = Σ Ri
16) satisfy the resource of A[RQi]
17) for every A[RQi]
18) allocate resource for RQi of A
19) Ri = Ri - RQi; Th = Σ Ri
20) satisfy the resource of B[RQi]

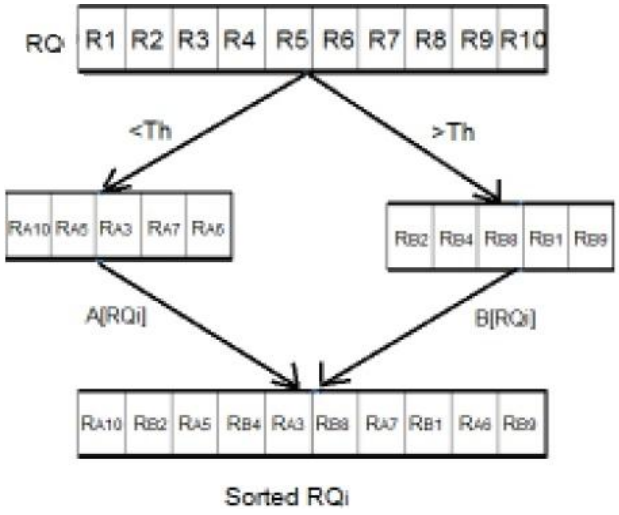

Fig. 2 Example of LSTR algorithm [3]

The dynamic allocation could be carried out by the scheduler dynamically on request for additional resources. This is made by the continuous evaluation of the threshold value. The resource requests are collected and are sorted in different queues based on the threshold value. The requests are satisfied by the VM's. Evaluation is made by creating VM in which the virtual memory is allocated to the longer and shorter queues based on the best fit strategy. This scheduling approach and the calculation of dynamic threshold value in the scheduler are carried out by considering both task and the resource. This improves the system throughput and the resource utilization regardless of the starvation and the dead lock conditions.

C. Dynamic Resource Allocation for Parallel Data Processing

Dynamic Resource Allocation for Efficient Parallel data processing [4] introduces a new processing framework explicitly designed for cloud environments called Nephele Most notably, Nephele is the first data processing framework to include the possibility of dynamically allocating/de-allocating different compute resources from a cloud in its scheduling and during job execution. Particular tasks of a processing job can be assigned to different types of virtual machines which are automatically instantiated and terminated during the job execution.

1) Architecture: Nephele's architecture [4] follows a classic master-worker pattern as illustrated in Figure. Before submitting a Nephele compute job, a user must start a VM in the cloud which runs the so called Job Manager (JM). The Job Manager receives the client's jobs, is responsible for scheduling them, and coordinates their execution. It is capable of communicating with the interface the cloud operator
provides to control the instantiation of VMs. We call this interface the Cloud Controller. By means of the Cloud Controller the Job Manager can allocate or de-allocate VMs according to the current job execution phase. The term instance type will be used to differentiate between Ms with different hardware characteristics. E.g., the instance type "m1.small" could denote VMs with once core, one GB of RAM, and a 128 GB disk while the instance type "c1.xlarge" could refer to machines with 8CPU cores, 18 GB RAM, and a 512 GB disk.

The actual execution of tasks which a Nephele job consists of is carried out by a set of instances. Each instance runs a so called Task Manager (TM). A Task Manager receives one or more tasks from the Job Manager at a time, executes them, and after that informs the Job Manager about their completion or possible errors.

Unless a job is submitted to the Job Manager, we expect the set of instances (and hence the set of Task Managers)to be empty. Upon job reception the Job Manager then decides, depending on the job's particular tasks, how many and what type of instances the job should be executed on, and when the respective instances

must be allocated/deallocated to ensure a continuous but cost-efficient processing. Our current strategies for these decisions are highlighted at the end of this section.

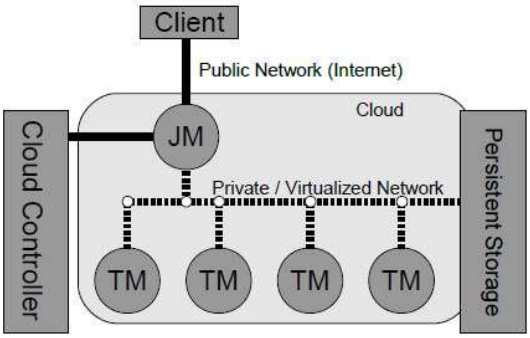

Fig. 3 Design Architecture of Nephele Framework [4]

The newly allocated instances boot up with a previously compiled VM image. The image is configured to automatically start a Task Manager and register it with the Job Manager. Once all the necessary Task Managers have successfully contacted the Job Manager, it triggers the execution of the scheduled job.

Initially, the VM images used to boot up the Task Managers are blank and do not contain any of the data the Nephele job is supposed to operate on. As a result, we expect the cloud to offer persistent storage (likee.g. Amazon S3). This persistent storage is supposed to store the job's input data and eventually receive its output data. It must be accessible for both the Job Manager as well as for the set of Task Managers, even if they are connected by a private or virtual network.

2) Job Description: Jobs in Nephele are expressed as a directed acyclic graph (DAG). Each vertex in the graph represents a task of the overall processing job, the graph's edges define the communication flow between these tasks Job description parameters are based on the following criteria's:

- Number of subtasks
- Data sharing between instances of task
- Instance type
- Number of subtasks per instance

3) Job Graph: Once the Job Graph is specified, the user submits it to the Job Manager, together with the credentials he has obtained from his cloud operator. The credentials are required since the Job Manager must allocate/deallocate instances during the job execution on behalf of the user.

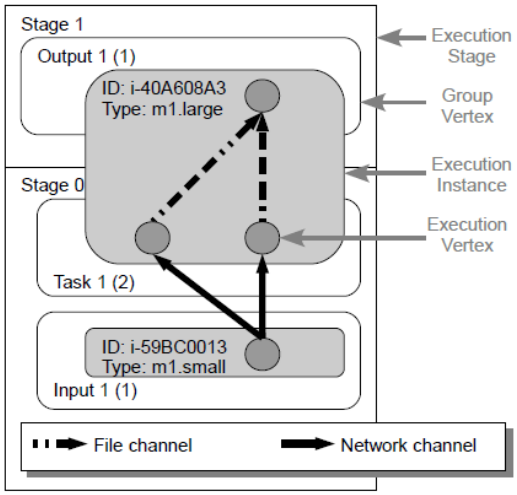

Fig 4: An Execution Graph created from the original Job Graph

4) Job Scheduling and Execution:

After having received a valid Job Graph from the user, Nephele's Job Manager transforms it into a so-called Execution Graph. An Execution Graph is Nephele's primary data structure for scheduling and monitoring the execution of a Nephele job. Unlike the abstract Job Graph, the Execution Graph contains all the concrete information required to schedule and execute the received job on the cloud. It explicitly models task parallelization and the mapping of tasks to instances. Depending on the level of annotations the user has provided with his Job Graph, Nephele may have different degrees of freedom in constructing the Execution Graph. Figure 3 shows one possible Execution Graph constructed from the previously depicted Job Graph (Figure 2). Task 1 is e.g. split into two parallel subtasks which are both connected to the task Output 1 via file channels and are all scheduled to run on the same instance. The exact structure of the Execution Graph is explained in the following figure 4.

There are many benefits in resource allocation while using cloud computing irrespective of size of the organization and business markets. But there are some limitations as well, since it is an evolving technology. Let's have a comparative look at the advantages and limitations of resource allocation in cloud. [1]

D:Advantages and Limitations of Resource Allocation

Advantages:

- The biggest benefit of resource allocation is that user neither has to install software nor hardware to access the applications, to develop the application and to host the application over the internet.
- The next major benefit is that there is no limitation of place and medium. We can reach our applications and data anywhere in the world, on any system.
- The user does not need to expend on hardware and software systems.
- Cloud providers can share their resources over the internet during resource scarcity.

Limitations:

- Since users rent resources from remote servers for their purpose, they don't have control over their resources.
- Migration problem occurs, when the users wants to switch to some other provider for the better storage of their data. It's not easy to transfer huge data from one provider to the other.
- In public cloud, the clients' data can be susceptible to hacking or phishing attacks. Since the servers on cloud are interconnected, it is easy for malware to spread.
- Peripheral devices like printers or scanners might not work with cloud. Many of them require software to be installed locally. Networked peripherals have lesser problems.
- More and deeper knowledge is required for allocating and managing resources in cloud, since all knowledge about the working of the cloud mainly depends upon the cloud service provider.
-

V. COMPARATIVE STUDY

An optimal RAS should avoid the following criteria as follows:

- Resource Contention - Resource contention arises when two applications try to access the same resource at the same time.
- Scarcity of Resource - Scarcity of resource arises when there are limited resources and the demand for resources is high.
- Resource Fragmentation - Resource fragmentation arises when the resources are isolated. There would be enough resources but cannot allocate it to the needed application due to fragmentation into small entities.
- Over Provisioning - Over provisioning arises when the application gets surplus resources than the demanded one.
- Under Provisioning - Under provisioning of resources occurs when the application is assigned with fewer numbers of resources than it demanded.

Sr. No.	Parameter	Linear Scheduling Strategy for Resource Allocation	Topology Aware Resource Allocation	Dynamic Resource Allocation (Nephele)
1]	Resource Contention	This does not avoid	This avoid	This avoid
2]	Scarcity of Resource	This does not avoid	This does not avoid	This avoid
3]	Resource Fragmentation	This does not avoid	This does not avoid	This avoid
4]	Over Provisioning	This does not avoid	This does not avoid	This avoid
5]	Under Provisioning	This does not avoid	This avoid	This avoid

Table 1: Comparative study of Resource Allocation Strategies in Cloud

VI. CONCLUSION

Cloud computing technology is increasingly being used in enterprises and business markets. A review shows that dynamic resource allocation is growing need of cloud providers for more number of users and with the less response time. In cloud paradigm, an effective resource allocation strategy is required for achieving user satisfaction and maximizing the profit for cloud service providers. This paper summarizes the main types of RAS and its impacts in cloud system. Some of the strategies discussed above mainly focus on memory resources but are lacking in other factors. Hence this survey paper will hopefully motivate future researchers to come up with smarter and secured optimal resource allocation algorithms and framework to strengthen the cloud computing paradigm.

REFERENCES

[1] V. Vinothina, Dr. R. Shridaran, and Dr. Padmavathi Ganpathi, A survey on resource allocation strategies in cloud computing, International Journal of Advanced Computer Science and Applications, 3(6):97--104, 2012.

[2] Gunho Lee, Niraj Tolia, Parthasarathy Ranganathan, and Randy H. Katz, Topology aware resorce allocation for data-intensive workloads, ACM SIGCOMM Computer Communication Review, 41(1):120--124, 2011.

[3] Abirami S.P. and Shalini Ramanathan, Linear scheduling strategy for resource allocation in cloud environment, International Journal on Cloud Computing: Services and Architecture(IJCCSA), 2(1):9--17, 2012.

[4] Daniel Warneke and Odej Kao, Exploiting dynamic resource allocation for efficient parallel data processing in the cloud, IEEE Transactions On Parallel And Distributed Systems, 2011.

[5] Atsuo Inomata, Taiki Morikawa, Minoru Ikebe and Md. MizanurRahman, Proposal and Evaluation of DynamicResource Allocation Method Based on the Load Of VMs on IaaS, IEEE, 2010.

[6] Dorian Minarolli and Bernd Freisleben, Uitlity–based Resource Allocations for virtual machines in cloud computing, IEEE, 2011.

[7] Jiyani, Adaptive resource allocation for preemptable jobs in cloud systems, IEEE, 2010.

[8] Bo An, Victor Lesser, David Irwin and Michael Zink, Automated Negotiation with Decommitment for Dynamic Resource Allocation in Cloud Computing, Conference at University of Massachusetts,Amherst, USA.

[9] Gihun Jung and Kwang Mong Sim, Location-Aware DynamicResource Allocation Model for Cloud Computing Environment,International Conference on Information and Computer Applications(ICICA), IACSIT Press, Singapore, 2012.

[10] Chandrashekhar S. Pawar and R.B. Wagh, A review of resource allocation policies in cloud computing, World Journal of Science and Technology, 2(3):165-167, 2012.

[11] Sandeep Tayal, Tasks Scheduling Optimization for the Cloud Computing systems, International Journal of Advanced Engineering Sciences and Technologies (IJAEST), 5(2): 111 - 115, 2011.

[12] Amazon Web Services LLC. Amazon Elastic Compute Cloud(Amazon EC2). http://aws.amazon.com/ec2/, 2009.

[13] Amazon Web Services LLC. Amazon Elastic MapReduce. http://aws.amazon.com/elasticmapreduce/, 2009.

[14] AmazonWeb Services LLC. Amazon Simple Storage Service. http://aws.amazon.com/s3/, 2009.

[15] D. Battr´e, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke.Nephele/PACTs: A Programming Model and Execution Frameworkfor Web-Scale Analytical Processing. In SoCC '10: Proceedingsof the ACM Symposium on Cloud Computing 2010, pages 119–130, New York, NY, USA, 2010. ACM.

[16] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib,S. Weaver, and J. Zhou. SCOPE: Easy and Efficient ParallelProcessing of Massive Data Sets. Proc. VLDB Endow., 1(2):1265–1276, 2008.

[17] H. chih Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-Reduce-Merge: Simplified Relational Data Processing on LargeClusters. In SIGMOD '07: Proceedings of the 2007 ACM SIGMODinternational conference on Management of data, pages 1029–1040,New York, NY, USA, 2007. ACM.

[18] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, andY. Tsang. Maximum Likelihood Network Topology Identificationfrom Edge-Based Unicast Measurements. SIGMETRICS Perform.Eval. Rev., 30(1):11–20, 2002.

Vinayak V. Awasare graduated in Computer Engineering from the University of Pune (India) in 2012. He is pursuing his Master of engineering in Computer Engineering from the University of Pune (India).He is currently research scholar student in Computer Department, Pimpri Chichwad College of Engineering; Pune (India).His research interests include Cloud Computing, Load Balancing and Parallel Computing.

Sudarshan S. Deshmukh graduated in Computer Engineering from the University of Shivaji (India) in 2004. He received his Masters in Computer Engineering from the Bharati University in 2009. He is currently working as an assistant professor, Computer Engg, at PCCOE, University of Pune since 2009. He is a member of the Technical Committee of Parallel Processing (TCPP), IEEE communication society, IAENG etc. Received Nomination for IEEE Technical Committee on Parallel Processing Outstanding Service Award for 2011.Associate Editor of International Journal of Cloud Applications and Computing, also serving as reviewer to several journals and conferences His research interests include distributed systems, resource sharing, load balancing etc.