Cost Effective Cloud Environment Setup to Secure Corporate Data

Mrs.Ashwini Prakash Sawant
M.Tech(Computer)Student
Bharati Vidyapeeth Deemed University College of Engineering, Pune,India
ashwinisawant0@gmail.com

Prof. Sandeep Vanjale
Ph.D. Research Scholar
Bharati Vidyapeeth Deemed University College of Engineering, Pune,India
sbvanjale@bvucoep.edu.in

Mrs. Mousami Vanjale
Asst. Professor, AISSMS IOIT,Pune.
Ph.D. Research Scholar, BVDUCOE, Pune
vmousami@gmail.com

***Abstract*-In recent years ad-hoc parallel processing has emerged to be one among the killer applications for Infrastructure-as-a-Service (IaaS) clouds. Major Cloud computing firms have began to integrate frameworks for parallel processing in their product portfolio, creating it simple for purchasers to access these services and to deploy their programs. However, the process frameworks that area unit presently used are designed for static, consistent cluster setups and disrespect the actual nature of a cloud. Consequently, the allotted calculate resources could also be inadequate for giant components of the submitted job and unnecessarily increase time interval and value. During this paper we tend to discuss the opportunities and challenges for economical parallel processing in clouds and present our research HPSSD. HPSSD is the data processing framework to overtly utilize the dynamic resource portion presented by today's IaaS clouds for each with hacker protection, task programming and execution. Specific tasks of a process job are often assigned to differing types of virtual machines that are mechanically instantiated and terminated throughout the task execution.**

Keywords*-cloud computing; distributed cloud; virtual Networks; resource allocation system; network virtualization environment, HPSSD (Hacking Protection and secured software Delivery using Cloud Data Storage Environment)*

I. INTRODUCTION

Now a days a number of companies have to process vast amounts of data in a cost-efficient manner. Classic representatives for these companies are operators of Internet search engines, like Google, Yahoo, or Microsoft. The huge quantity of knowledge they need to take care of daily has created ancient info solutions prohibitively high-priced [5]. Instead, these firms have popularized the architectural paradigm supported an outsized range of goods servers. Issues like process crawled documents or create an online index are split into many freelance subtasks, distributed among the offered nodes, and computed in parallel.

In order to change the event of distributed applications on high of such architectures, several of those firms have additionally engineered custom-made data processing frameworks. Examples are Google's MapReduce [9], Microsoft's dryad [14], or Yahoo!'s Map-Reduce-Merge [6].

They can be classified by terms like high turnout computing (HTC) or many-task computing (MTC), betting on the amount of knowledge and therefore the number of tasks concerned within the computation [20]. Though these systems dissent in style, their programming models share similar objectives, particularly concealing the trouble of parallel programming, fault tolerance, and execution optimizations from the developer. Developers will usually still write consecutive programs. The process framework then takes care of distributing the program among the offered nodes and executes every instance of the program on the acceptable fragment of knowledge. For firms that solely need to method massive amounts of information often running their own data center is clearly not the choice. Instead, Cloud computing has emerged as a promising approach to rent an outsized IT infrastructure on a short-run pay-per-usage basis. Operators of so-called Infrastructure-as-a-Service (IaaS) clouds, like Amazon EC2 [1], let their customers apportion, access, and management a collection of virtual machines (VMs) that run within their knowledge centers and solely charge them for the amount of time the machines are allotted. The VMs are usually offered in numerous varieties, every kind with its own characteristics (number of CPU cores, amount of main memory, etc.) and cost.

Since the VM abstraction of IaaS clouds fits the architectural paradigm assumed by the information process frameworks delineated above, comes like Hadoop [5], a well-liked open supply implementation of Google's MapReduce framework, have already got begun to push using their frameworks within the cloud [9]. Very recently, Amazon has integrated Hadoop jointly of its core infrastructure services [2]. However, rather than hold its dynamic resource allocation, current processing frameworks rather expect the cloud to imitate the static nature of the cluster environments they were originally designed for. E.g., at the instant the categories and range of VMs allotted at the start of a reckon job cannot be modified within the course of process, though the tasks the task consists of might need fully completely different demands on the setting. As a result, rented resources could also be inadequate for large components of the process job, which can lower the process performance and increase the value.

In this paper we would like to debate the actual challenges and opportunities for economical parallel processing in clouds and present HPSSD, a replacement process framework expressly designed for cloud environments. Most notably, HPSSD is that the 1st processing framework to incorporate the chance of dynamically allocating/deallocating completely different reckon resources from a cloud in its programming and through job execution.

II. CHALLENGES AND OPPORTUNITIES

Current processing frameworks like Google's MapReduce or Microsoft's dryad engine are designed for cluster environments. This is often mirrored during a range of assumptions they create that aren't essentially valid in cloud environments. During this section we tend to discuss however abandoning these assumptions raises new opportunities however conjointly challenges for economical parallel processing in clouds.

A. OPPORTUNITIES

Today's process frameworks usually assume the resources they manage include a static set of unvaried figure nodes. Though designed to subsume individual nodes failures, they think about the quantity of accessible machines to be constant, particularly once planning the process job's execution. Whereas IaaS clouds will actually be accustomed produce such cluster-like setups, a lot of of their flexibility remains unused.

One of an IaaS cloud's key options is that the provisioning of figure resources on demand. New VMs may be allotted at any time through a well-defined interface and become accessible in an exceedingly matter of seconds. Machines that are not any longer used may be terminated instantly and also the cloud client is going to be charged for them no additional. Moreover, cloud operators like Amazon let their customers rent VMs of various varieties, i.e. with totally different procedure power, totally different sizes of main memory, and storage. Hence, the pc resources accessible in a cloud are extremely dynamic and probably heterogeneous.

With relation to parallel processing, this flexibility results in a spread of latest potentialities, significantly for programming processing jobs. The question a hardware has got to answer is not any longer "Given a collection of compute resources, a way to distribute the actual tasks of employment among them?", however rather "Given employment, what compute resources match the tasks the work consists of best?".

This new paradigm permits allocating computer resources dynamically and only for the time they're needed within the process work flow. E.g., a framework exploiting the chances of a cloud might begin with one VM that analyzes an incoming job so advises the cloud to directly begin the desired VMs in line with the job's process phases. Once every part, the machines might be free and no longer contribute to the price for the process job.

Facilitating such use cases imposes some necessities on the look of a process framework and therefore the method its jobs square measure delineated. First, the hardware of such a framework should become responsive to the cloud setting employment ought to be dead in. It should comprehend the various styles of accessible VMs likewise as their price and be able to portion or destroy them on behalf of the cloud client.

Second, the paradigm accustomed describe jobs should be powerful enough to precise dependencies between the various tasks the roles consists of. The system should remember of that task's output is needed as another task's input. Otherwise the hardware of the process framework cannot decide at what purpose in time a selected VM is not any longer required and deallocate it. The MapReduce pattern may be a model of an unsuitable paradigm here: though at the end of employment solely few reducer tasks should be running, it's insufferable to finish off the idle VMs, since it's unclear if they contain intermediate results that are still needed.

Finally, the hardware of such a process framework should be ready to verify that task of employment ought to be executed on which sort of VM and, possibly, what percentage of these. This data may well be either provided outwardly, e.g. as an annotation to the task description, or deduced internally, e.g. from collected statistics, equally to the manner information systems attempt to optimize their execution schedule over time [4].

B. CHALLENGES

The key challenge we see is that the cloud's opaqueness with prospect to exploiting information locality:

In a cluster the figure nodes are generally interconnected through a physical superior network. The topology of the network, i.e. the approach the figure nodes are physically wired to every different, is sometimes accepted and, what's more necessary, doesn't modification over time. Current processing frameworks supply to leverage this information regarding the network hierarchy and conceive to schedule tasks on compute nodes so data sent from one node to the opposite has got to traverse as few network switches as doable [9]. That approach network bottlenecks will be avoided and also the overall throughput of the cluster will be improved.

In a cloud this topology info is often not exposed to the client [9]. Since the nodes concerned in process a knowledge intensive job usually ought to transfer tremendous amounts of knowledge through the network, this disadvantage is especially severe; components of the network could become full whereas others are basically unutilized. Though there has been analysis on inferring possible network topologies alone from finish-to-end measurements (e.g. [7]), it's unclear if these techniques are applicable to IaaS clouds. For security reasons clouds usually incorporate network virtualization techniques (e.g. [8]) which might hamper the logical thinking method, especially when supported latency measurements.

Even if it absolutely was attainable to see the underlying network hierarchy during a cloud and use it for topologyaware programming, the obtained data wouldn't essentially stay valid for the complete interval. VMs are also migrated for administrative functions between completely different locations within the information center with none notification, rendering any previous information of the relevant network infrastructure obsolete.

As a result, the sole way to ensure vicinity between tasks of a process job is presently to execute these tasks on identical VM within the cloud. This could involve allocating fewer, however a lot of powerful VMs with multiple CPU cores. E.g., take into account an aggregation task receiving information from seven generator tasks. However, presently no processing framework includes such ways in its programming algorithms.

III. PARALLELIZATION AND SCHEDULING STRATEGIES

As mentioned before, constructing an Execution Graph from a user's submitted Job Graph may leave different degrees of freedom to HPSSD. Using this freedom to construct the foremost economical Execution Graph (in terms of time interval or financial cost) is presently a significant focus of our analysis. Discussing this subject very well would transcend the scope of this paper. However, we wish to stipulate our basic approaches during this subsection:

Unless the user provides any job annotation that contains a lot of specific directions we presently pursue a straightforward default strategy: every vertex of the work Graph is reworked into one Execution Vertex. The default channel varieties are network channels. Every Execution Vertex is by default assigned to its own Execution Instance unless the user's annotations or alternative programming restrictions (e.g. the usage of in-memory channels) command it. The default instance sort to be used is that the one with the bottom value per time unit out there within the IaaS cloud.

One elementary plan to refine the programming strategy for revenant jobs is to use feedback knowledge. We developed an identification system for HPSSD which may continuously monitor running tasks and therefore the underlying instances. Supported the Java Management Extensions (JMX) the identification system is, among alternative things, capable of breaking down what share of its time interval a task thread really spends process user code and what share of time it's to attend for knowledge. With the collected knowledge HPSSD is able to notice both computational as well as I/O bottlenecks. Whereas procedure bottlenecks counsel a better degree of parallelization for the affected tasks, I/O bottlenecks offer hints to change to quicker channel sorts (like in memory channels) and rethink the instance assignment.

Since HPSSD calculates a cryptologic signature for every task, continual tasks is known and also the antecedently recorded feedback information is exploited. At the instant we tend to solely use the identification information to notice these bottlenecks and facilitate the user to decide on cheap annotations for his job. It provides immediate visual feedback concerning the present utilization of all tasks and cloud instances concerned within the computation. In additional advanced versions of HPSSD we envision the system to mechanically adapt to detected bottlenecks, either between consecutive executions of an equivalent job or perhaps throughout job execution at run time.

While the allocation time of cloud instances is decided by the beginning times of the appointed subtasks, there are completely different attainable methods for example de-allocation. So as to mirror the very fact that the majority cloud suppliers charge their customers for example usage by the hour, we integrated the chance to use instances. An instance of a selected kind that has become obsolete within the current Execution Stage isn't directly de-allocated if an instance of constant kind is needed in an approaching Execution Stage. Instead, HPSSD keeps the instance allotted till the end of its current lease amount. If ensuing Execution Stage has begun

before the tip of that amount, it's reassigned to an Execution Vertex of that stage, otherwise it de-allocated early enough to not cause any extra value.

Besides the utilization of feedback knowledge we recently complemented our efforts to supply affordable job annotations mechanically by a higher-level programming model bedded on prime of HPSSD. instead of describing jobs as impulsive DAGs, this higher-level programming model referred to as PACTs [4] is targeted round the concatenation of second-order functions, e.g. just like the map and scale back function from the well-known MapReduce programming model. Developers will write custom initial order functions and fix them to the required second order functions.

IV. GENERAL HARDWARE SETUP

An experiment is carried out on our locally configured ubantu cloud of two servers. We used two desktop for cloud setup to reduce cost of machines. Both are connected through regular cross cables to avoid use of VLAN. The main operating system was Linux (kernel version 2.xx).

To study the cloud, we originated Eucalyptus [16]. The same as Amazon EC2, Eucalyptus offers a predefined set of instance varieties a user will choose between. Throughout our experiments we used two totally different instance sorts' algorithms: the primary instance type was "file" and second instance was "data in file" which we used to implement data security by inventing new technique "data roars".

As a demo testing the 100 to 1000 KB computer file set of random numbers has been generated in step with the principles of the Jim grey type benchmark [18]. Inline to the information accessible to Hadoop, we started an HDFS [25] knowledge node on every of the allotted instances before the process job and distributed the information equally among the nodes.

V. RESULTS

Figure 1 show the performance results of our experiment, severally. This plot illustrates the typical instance utilization over time, i.e. the typical utilization of all CPU cores all told instances allotted for the task at the given purpose in time. The employment of every instance has been monitored with the UNIX operating system command "top" and is lessened into the quantity of time the hardware cores spent running the several processing framework, the kernel and its processes (SYS), and also the time looking ahead to I/O to finish (WAIT). So as let's say the impact of network communication, the plots in addition show the typical quantity of information processing traffic flowing between the instances over time.

We compared our results with existing Hadoop results. And the main difference is Hadoop is a standalone application and HPSSD is in network and over the web. During the map phase instances show an average utilization of about 80 %. However, after approximately 50 minutes, HPSSD starts transmitting the sorted output stream of subtasks to the two instances which are scheduled to remain allocated for the upcoming random Execution Stages to protect from possibility of hacking.

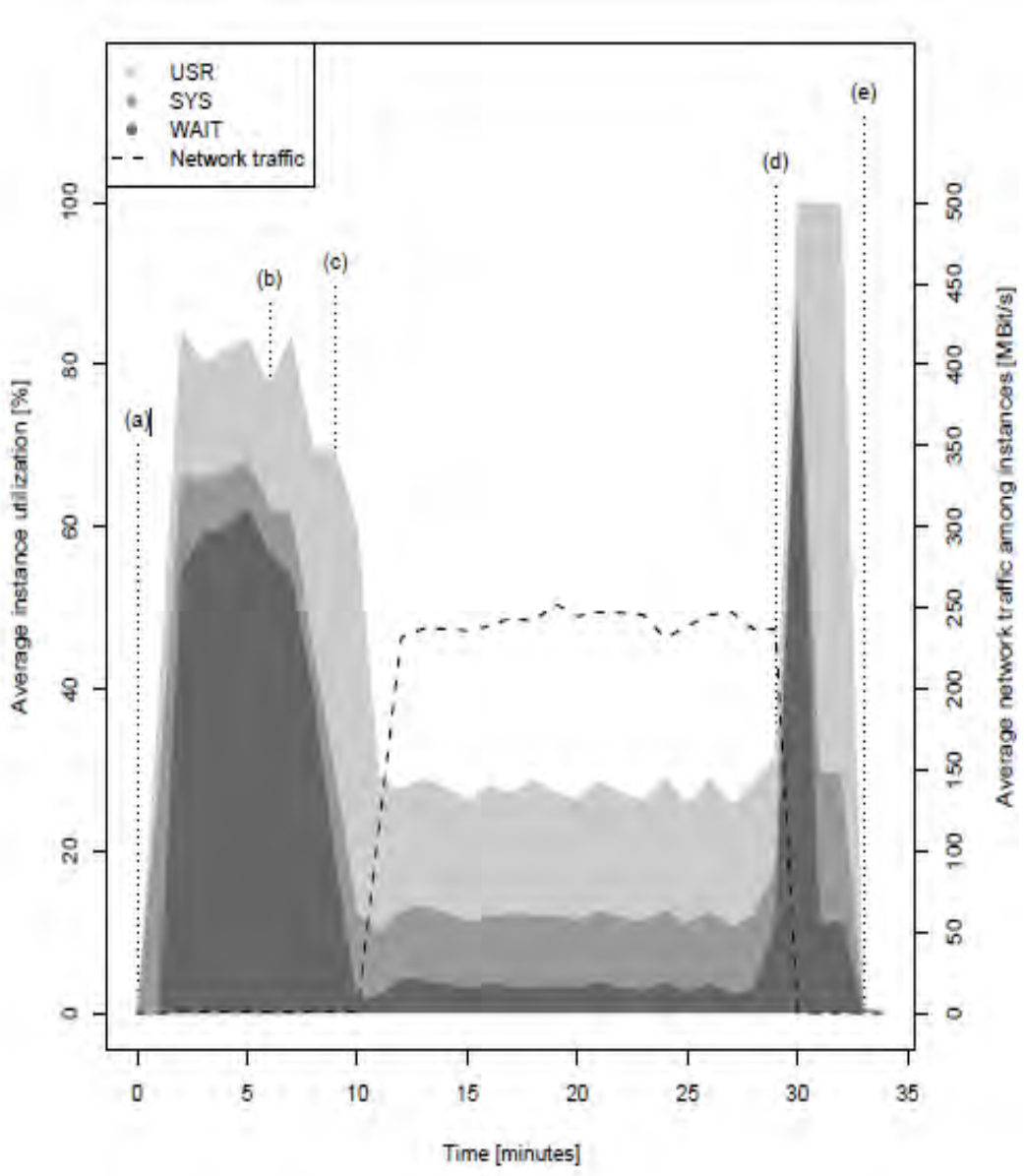

Figure 1. Performance Results Of Our Experiment

Figure 1 show the performance results of our experiment, severally. This plot illustrates the typical instance utilization over time, i.e. the typical utilization of all CPU cores all told instances allotted for the task at the given purpose in time. The employment of every instance has been monitored with the UNIX operating system command "top" and is lessened into the quantity of time the hardware cores spent running the several processing framework, the kernel and its processes (SYS), and also the time looking ahead to I/O to finish (WAIT). So as let's say the impact of network communication, the plots in addition show the typical quantity of information processing traffic flowing between the instances over time.

We compared our results with existing Hadoop results. And the main difference is Hadoop is a standalone application and HPSSD is in network and over the web. During the map phase instances show an average utilization of about 80 %. However, after approximately 50 minutes, HPSSD starts transmitting the sorted output stream of subtasks to the two instances which are scheduled to remain allocated for the upcoming random Execution Stages to protect from possibility of hacking.

VI. CONCLUSION

In this paper we've mentioned the challenges and opportunities for economical parallel processing in cloud environments and bestowed HPSSD, the primary processing framework to take advantage of the dynamic resource provisioning offered by today's IaaS clouds. We've delineated basic cloud design and bestowed a performance comparison to the well-established processing framework Hadoop. The performance analysis provides a primary impression on however the power to assign specific virtual machine varieties to specific

tasks of a process job, likewise because the risk to mechanically allocate/deallocate virtual machines within the course of employment execution will facilitate to enhance the resource utilization and, consequently, scale back the process value.

REFERENCES

[1] Amazon Web Services LLC. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/, 2013.

[2] Amazon Web Services LLC. Amazon Elastic MapReduce. http: //aws.amazon.com/elasticmapreduce/, 2012.

[3] AmazonWeb Services LLC. Amazon Simple Storage Service. http: //aws.amazon.com/s3/, 2010.

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

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

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

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

[8] R. Davoli. VDE: Virtual Distributed Ethernet. Testbeds and Research Infrastructures for the Development of Networks & Communities, International Conference on, 0:213–220, 2005.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[10] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. Sci. Program., 13(3):219–237, 2005.

[11] T. Dornemann, E. Juhnke, and B. Freisleben. On-Demand Resource Provisioning for BPEL Workflows Using Amazon's Elastic Compute Cloud. In CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 140–147, Washington, DC, USA, 2009. IEEE Computer Society.

[12] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. Intl. Journal of Supercomputer Applications, 11(2):115–128, 1997.

[13] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing, 5(3):237–246, 2002.

[14] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pages 59–72, New York, NY, USA, 2007. ACM.

[15] A. Kivity. kvm: the Linux Virtual Machine Monitor. In OLS '07: The 2007 Ottawa Linux Symposium, pages 225–230, July 2007.

[16] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems. Technical report, University of California, Santa Barbara, 2008.

[17] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: A Not-So-Foreign Language for Data Processing. In SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 1099–1110, New York, NY, USA, 2008. ACM.

[18] O. O'Malley and A. C. Murthy. Winning a 60 Second Dash with a Yellow Elephant. Technical report, Yahoo!, 2009.

[19] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the Data: Parallel Analysis with Sawzall. Sci. Program., 13(4):277–298, 2005.

[20] I. Raicu, I. Foster, and Y. Zhao. Many-Task Computing for Grids and Supercomputers. In Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on, pages 1–11, Nov. 2008.

[21] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde. Falkon: a Fast and Light-weight tasK executiON framework. In SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, pages 1–12, New York, NY, USA, 2007. ACM.

AUTHORS PROFILE

Mrs.Ashwini Prakash Sawant is student of M.Tech Computer in BVDUCOE,Pune. She has completed Diploma in Electrical Engineering From Walchand College Of Engineering; Sangali(MSBTE) And BE in Computer Science and Engineering from KBP College of Engineering Satara Of Shivaji University, Kolhapur.

Prof.S.B.Vanjale is Professor in Computer Engineering Department of Bharati Vidyapeeth University college of engineering, Pune (Maharashtra).He is pursuing Ph.d from Bharati Vidyapeeth University,Pune.He obtained his ME (Computer) degree from Bharati Vidyapeeth University Pune in 2004. He has guided 16 post graduate students.He has thirteen years of experience in teaching. His research interest include Computer Network, Network Security. He attended many conferences and workshops and published more than 25 Papers in national and international Journals.

Mrs. Mousami Vanjale, Asst. Professor, AISSMS IOIT Pune. She is pursuing Ph.d from Bharati Vidyapeeth University,Pune. She has ten years of experience in teaching. She obtained her M.Tech(Electronics) degree from Govt. College Of Engineering,Pune in 2010. Her research interest include Mobile Ad-Hoc Network, Robotics and automation.