ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

Efficient Load Balancing in Cloud Computing Using Weighted Throttled Algorithm

B. Nithya Nandhalakshmi [1] Mahalingam [2]

M.E, Dept. of C.S.E, J.K.K Nattaraja College of Engineering, Erode, India[1]

HOD, Dept. of C.S.E., J.K.K Nattaraja College of Engineering, Erode, India[2]

ABSTRACT: Load balancing is one of the critical components for efficient operations in the cloud computing environment. In recent years many clients from all over the world are demanding the various services at rapid rate. Many algorithms have been designed to carry out the client's request towards the cloud nodes. Accordingly the cloud computing platform will dynamically configure its servers and these servers may be present physically or virtually in the computing environment. Hence, selecting the virtual machines or servers has to be scheduled properly by using an appropriate load balancing approach. In the present work, a weighted based optimized load balancing approach is proposed for distributing of incoming jobs uniformly among the servers or virtual machines. Further, the performance is analyzed using Cloud simulator and compared with existing Round Robin and EIPR algorithms. Simulation results have demonstrated that the proposed algorithm has distributed the load uniformly among virtual machines.

KEYWORDS: Infrastructure-as-a-service; acyclic graphs; Weight based Throttled algorithm; Cloud Service Broker Policy

I. INTRODUCTION

Over the last decade, cloud computing emerged as a form of distributed computing, in which computational resources can be provisioned on-demand over the Internet [1]. In the Infrastructure-as-a-service (IaaS) model of cloud computing, computational resources of any scale can be rented in the form of virtual machines (VMs) from commercial cloud providers like Amazon or Microsoft [2]. The convenience of its pay-as-you-go model along with the aggressive promotion by its providers has led to an exponential growth in the usage of cloud computing over the last years the performance of rented cloud infrastructure and the requirements of the to-be-deployed application. These characteristics are hard to quantify and vary depending on the application and cloud provider. Since benchmarking a given application on cloud infrastructure of large scale repeatedly under various experimental conditions is both tedious and expensive, simulation constitutes a convenient and affordable way of evaluation prior to implementation and execution on real hardware [3, 5].

Unfortunately, available cloud simulation toolkits like CloudSim [6] do not adequately capture in homogeneity and dynamic performance changes inherent to non-uniform and shared infrastructures like computational clouds. The effect of these factors of uncertainty and instability is not negligible and has been repeatedly observed to strongly in fluency the runtime of a given application on commercial clouds such as Amazon's Elastic Compute Cloud (EC2).

Scientific work flows are directed, acyclic graphs (DAGs), in which nodes correspond to data processing tasks and edges constitute data dependencies between these tasks. They have recently gained attention as a flexible programming paradigm for modeling, representing, and executing complex computations and analysis pipelines in many different areas of scientific research. A considerable number of scientific work how management systems (SWfMS) has been developed Many of these systems provide the functionality to design work flows in a drag-and-drop user interface, share work flows with other users in public repositories, or execute work flows on appropriate computational architectures.

Proposed work present heuristic based Dynamic CloudSim, an extension to CloudSim which provides an array of capabilities to model the instability inherent to computational clouds and similar distributed infrastructures. We

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

showcase the applicability of heuristic dynamic CloudSim in a series of experiments involving the scheduling of a computationally intensive scientific work flow which has been repeatedly used for evaluation purposes. We believe scientific work flow scheduling to be a suitable use case for demonstrating the capabilities of a cloud simulation framework for two reasons: (1) Scheduling computationally intensive scientific work flows on distributed and potentially shared architectures presents many opportunities for optimizing robustness to instability (2) Simulation has been repeatedly made use of for evaluating scientific work flow schedulers.

II. RELATED WORK

Recent research focused on algorithms that are aware of the intricacies of Cloud environments when using them to schedule workflow applications. Reynolds et al. [7] proposed the utilization of Clouds to complement desktop Grid resources. However, Cloud resources are deployed with the goal of replicating slow tasks to increase the chance of an early completion of the workflow. The proposed method is not optimized either for budget or for execution time; instead, it operates in a best-effort basis when late tasks are detected. Xu et al. [8] and Mao and Humphrey [9] proposed algorithms for scheduling multiple workflows in Clouds. Rahman et al. [10] proposed an algorithm for hybrid Clouds, where at least part of the resources can be used without cost and with higher level of control over their performance. Different from this approach, we focus on the scheduling of single workflows in pure Cloud environments.

Scheduling of workflows (also referred as Direct Acyclic GraphsV DAGs) in parallel and distributed systems has been subject of extensive research. Kwok and Ahmad [11] presented a survey describing and comparing different algorithms for static scheduling of workflows in parallel systems. Shi and Dongarra [12] present an algorithm for scheduling of workflows in heterogeneous clusters. These algorithms can be used in the context of Cloud computing if the machines are provisioned a priori. However, they cannot decide the optimal number of resources to reduce the cost of use of the infrastructure, and therefore they are not suitable for elastic Cloud environments using a pay-per-use economic model.
Byun et al. [13] proposed the Partitioned Balanced Time Scheduling (PBTS) algorithm for cost-optimized and deadline-constrained execution of workflow applications on Clouds. The PBTS algorithm considers only one type of Cloud resource, chosen a priori, for its provisioning and scheduling decision. Abrishami et al. [14] proposed two algorithms for cost-optimized, deadline-constrained execution of workflows in Clouds. These algorithms do not consider all data transfer times during provisioning and scheduling, increasing the execution budget. Our proposed algorithm is based on one of such algorithms (called IC-PCP), but also accounts for data transfer times and Cloud resources boot time during the provisioning and scheduling process. Furthermore, it explores possibility of tasks replication to increase the probability of meeting application deadlines.

Existing CloudSim Tool is an extension of the GridSim framework for simulation of resource provisioning and scheduling algorithms on cloud computing infrastructure developed by Calheiros et al. [15] at the University of Melbourne's CLOUDS Laboratory. It provides capabilities to perform simulations of assigning and executing a given workload on a cloud computing infrastructure under different experimental conditions. Cloud Simulation for instance has been used to (1) measure the effects of a power-aware VM provisioning and migration algorithm on datacenter operating costs for real-time cloud applications, (2) evaluate a cost-minimizing algorithm of VM allocation for cloud service providers, which takes into account a acculturating user base and heterogeneity of cloud VMs, (3) develop and showcase a scheduling mechanism for assigning tasks of different categories yet without data dependencies.

Cloud Simulation tool operates event-based, i.e., all components of the simulation maintain a message queue and generate messages, which they pass along to other entities. A Cloud simulation can instantiate several datacenters, each of which is comprised of storage servers and physical host machines, which in turn host multiple VMs executing several tasks (named cloudlets in CloudSim). A datacenter is characterized by its policy of assigning requested VMs to host machines (with the default strategy being to always choose the host with the least cores in use). Each datacenter can be configured to charge different costs for storage, VM usage, and data transfer.

Workflow partitioning can be classified as a network cut problem where a sub-workflow is viewed as a sub-graph. But there are two differences with our approach. First, we must consider the problem of data overlap when a new job is added to a sub-workflow. Second, valid workflows require no cross dependencies although it is possible to make that cut in network cut problem.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

A variety of sophisticated algorithms for scheduling scientific workflows on shared infrastructures such as computational clouds have been developed. As an application example for Dynamic Cloud Simulation, we compare the performance of several established scientific workflow schedulers at different levels of instability. Since some of the investigated schedulers have been developed to handle heterogeneity, dynamic performance changes, and failure, we expect our experiments to replicate the advertised strengths of the different workflow schedulers. Results from an extensive number of simulation runs confirm these expectations, underlining the importance of elaborate scheduling mechanisms when executing workflows on shared computational infrastructure.

III. PROPOSED SYSTEM

Efficient scheduling is the critical concept of the load balancing cloud computing based on the performance. In complex and large systems, there is a tremendous need for load balancing. Weight based Throttled algorithm is completely based on virtual machine. In this client first requesting the load balancer to check the right virtual machine which access that load easily and perform the operations which is give by the client or user. In this algorithm the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation.

In this algorithm the load balancer maintains an index table of virtual machines as well as their states (Available or Busy). The client/server first makes a request to data centre to find a suitable virtual machine (VM) to perform the recommended job. The data centre queries the load balancer for allocation of the VM. The load balancer scans the index table from top until the first available VM is found or the index table is scanned fully. If the VM is found, the load data centre. The data centre communicates the request to the VM identified by the id. Further, the data centre acknowledges the load balancer of the new allocation and the data centre revises the index table accordingly. While processing the request of client, if appropriate VM is not found, the load balancer returns -1 to the data centre. The data centre queues the request with it. When the VM completes the allocated task, a request is acknowledged to data centre, which is further apprised to load balancer to de- allocate the same VM whose id is already communicated. The total execution time is estimated in three phases. In the first phase the formation of the virtual machines and they will be idle waiting for the scheduler to schedule the jobs in the queue, once jobs are allocated, the virtual machines in the cloud will start processing, which is the second phase, and finally in the third phase the cleanup or the destruction of the virtual machines. The throughput of the computing model can be estimated as the total number of jobs executed within a time span without considering the virtual machine formation time and destruction time The proposed algorithm will improve the performance by providing the resources on demand, resulting in increased number of job executions and thus reducing the rejection in the number of jobs submitted.

A. *Cloud Service Broker Policy:*

A service broker decides which datacenter should provide the service to the requests coming from each user. And thus, service broker controls the traffic routing between user and datacenters. So in simple words, it is datacenter selection policy. Here some datacenters and users are shown in the figure. When request come from the user then service broker policy helps to decide which datacenter will provide service for upcoming request.

1. Closest Data center Policy
2. Optimize Response Time Policy
3. Dynamically reconfigurable routing with load balancing

- Closest Data center Policy

The data center which is having least proximity from the user is selected. Proximity in term of least network latency. If more than one Data centers having same proximity then it will select data center randomly to balance the load.

- Optimize Response Time Policy

First it identifies the closest data center using previous policy but when Closest Data centres performance (considers response time) starts degrading it estimates current response time for each data center then searches for the data center which having least estimated response time. But there may be 50:50 chances for the selection of closest and fastest data center.

- Dynamically reconfigurable routing with load balancing

This is an extension to Closest Data center Policy where the routing logic is similar. But it has one more responsibility of scaling the application deployment based on the load it is facing. It also increases or decreases the no. of VMs

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

accordingly. This will be done taking under consideration the current processing times and best processing time ever achieved.

B. *Weighted Response Time:*
The purpose of these algorithms is to calculate the expected response time. We use the following formula for calculation Response

$$\text{Time} = F_{int} + A_{rrt} + T_{Delay} \qquad \text{eq. (1)}$$

where, Arrt is the arrival time of user request and Fint is the finish time of user request and Tdalay is the transmission delay. However, Tdelay can be calculated as

$$T_{Delay} = T_{latency} + T_{transfer} \qquad \text{eq. (2)}$$

Here, Tlatency is the network latency and Ttransfer is the time taken to transfer the size of data of a single request (D) from source location to destination.Tlatency is taken from the latency matrix (after applying Poisson distribution on it for distributing it)held in the internet characteristics.

$$T_{transfer} = {}^{D}/_{BW_{peruser}} \qquad \text{eq. (3)}$$

where Bwperuser = Bwtotal / N; Bwtotal is the total available bandwidth (held in the internet characteristics) and N is the number of user requests currently in transmission. The internet characteristics also keep track of the number of user requests in-flight between two regions for the value of N.

C. *Throttled Load Balancer Algorithm:*
This algorithm is a dynamic load balancing algorithm. It is used for load balancing in the case of the virtual machines to be used. Here it first checks the index values of all the virtual machine in the system. The request is sent where load balancer parses a table for the allocation of the resources in the system. It assigns the request to a particular load balancer which passes or responds reverse the request to the requester and updates the allocation policy. After the successful allocation of the system the whole process for the de-allocation of the system also starts. This mechanism provides a greater a higher amount of resource sharing and allocation on a whole in the system resulting in the higher performance and utilization. The throttling threshold maintained generally is 1. It could be modified easily to make the threshold a configurable value.

Throttled algorithm is completely based on virtual machine. In this client first requesting the load balancer to check the right virtual machine which access that load easily and perform the operations which is give by the client or user. In this algorithm the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation.

The total execution time is estimated in three phases. In the first phase the formation of the virtual machines and they will be idle waiting for the scheduler to schedule the jobs in the queue, once jobs are allocated, the virtual machines in the cloud will start processing, which is the second phase, and finally in the third phase the cleanup or the destruction of the virtual machines. The throughput of the computing model can be estimated as the total number of jobs executed within a time span without considering the virtual machine formation time and destruction time The proposed algorithm will improve the performance by providing the resources on demand, resulting in increased number of job executions and thus reducing the rejection in the number of jobs submitted.

D. *Algorithm Implementation:*
Step 1: Initially VM index table will be 0 as all the VMs are in available state.
Step 2: Data Center Controller receives a new request.
Step 3: Data Center Controller queries new Load Balancer for next allocation.
Step 4: Data Center Controller parses the VM list to get next available VM: If found: Load Balancer returns the VM id to Data Center Controller Step2 continues If not found: Using weight fashion VM index is re-initialized to 0 and in increment manner VMs are checked to find VM in available state.
Step 5: When the VM finishes the processing the request, and the Data Center Controller receives the cloulet response, it notices the load balancer of the VM de-allocation
Step 6: The Load Balancer updates the status of VM in allocation table to available.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

Step 7: Continue from Stp2 the purpose of the algorithm is to find the expected Response Time of each Virtual Machine, which is calculated as: refer eq. (1) Where,Arrt is the arrival time of user request and Fint is the finish time of user request and the transmission delay can be determined using the following formulas as eq.(2)
Where, TDelay is the transmission delay T latency is the networklatency and T transfer is the time taken to transfer the size of data of a single request (D) from source location to destination.

$$Weight = {}^{D}/_{BW_{peruser}} * BW_{total} = \frac{BW_{total}}{N_r} \qquad eq.(4)$$

Where, Bwtotal is the total available bandwidth and Nr is the number of user requests currently in transmission. The Internet Characteristics also keeps track of the number of user requests in between two regions for the value of Nr.

E. *Scientific Workflow Schedulers:*
The workflow from the end to the beginning, computing the upward rank of each task as the estimated time to overall workflow completion at the onset of this task. The computation of a given task's upward rank incorporates estimates for both the runtimes and data transfer times of the given task as well as the upward ranks of all successor tasks. The static schedule is then assembled by assigning each task in decreasing order of upward ranks a time slot on a computational resource, such that the task's scheduled finish time is minimized.

IV. SIMULATION RESULTS

To showcase a possible application of Dynamic Cloud Simulation, we simulate the execution of a computationally intensive workflow using different mechanisms of scheduling and different levels of instability in the computational infrastructure. We expect the schedulers to differ in their robustness to instability, which should be reflected in diverging workflow execution times. Here outline the evaluation work-flow, the experimental settings, and the schedulers which we used in our experiments.

Montage is able to generate workflows for assembling high-resolution mosaics of regions of the sky from raw input data. It has been repeatedly utilized for evaluating scheduling mechanisms or computational infrastructures for scientific workflow execution a schematic visualization of the Montage workflow and an example of output generated by a Montage workflow. In these experiments, we used a Montage workflow which builds a large-scale (twelve square degree) mosaic of the m17 region of the sky. This workflow consists of 43,318 tasks reading and writing 534 GB of data in total, of which 10 GB are input and output files which have to be uploaded to and downloaded from the computational infrastructure.

The comparative evaluation of proposed optimized weight throttled algorithm and the existing algorithms for heterogeneous system such as Round robin, EIPR for DAGs with various characteristics by simulation. For this purpose, we consider two sets of graphs as the workload for testing the algorithms: randomly generated task graphs and the graphs that represent some of numerical real world problems. We have used Intel Xeon processors with 1 GHz speed for our experiments.

Comparison metrics: We have used the following metrics to evaluate the proposed algorithm.
Schedule length ratio (SLR): SLR is the ratio of the parallel time to the sum of weights of the critical path tasks on the fastest processor.
Speedup: Speed up is the ratio of the sequential execution time to the parallel execution time.
Running time of the algorithms: The running time (the scheduling time) of an algorithm is its execution time for obtaining the output schedule of a given task graph.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

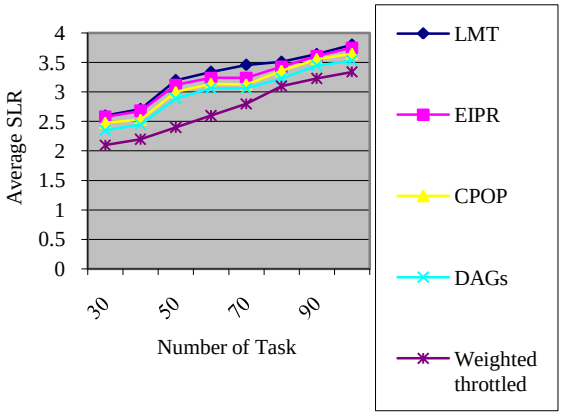

Fig.1. Comparison of no of task Vs SLR

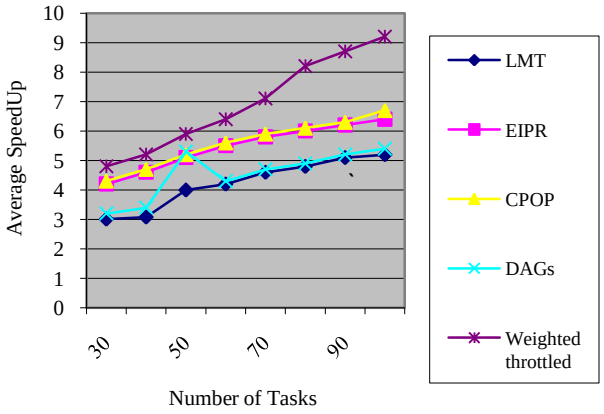

Fig. 2. Comparison of no of task Vs Average Speedup

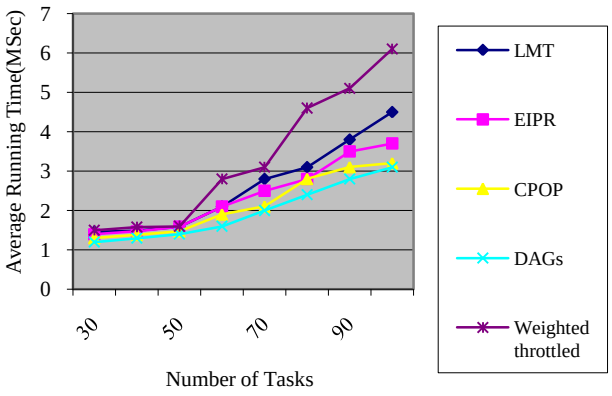

Fig. 3. Comparison of no of task Vs Average Run Times

V. CONCLUSION AND FUTURE WORK

Choosing right load balancer at the beginning is imperative to the success of complex implementations later. So far the proposed techniques weighted based load balancing algorithms. The proposed algorithm least VM assign method distribute workload across multiple computers to achieve optimal resource utilization with minimum response time. Cloud simulation simulator is used for algorithm implementations. Cloud simulation is a framework which enables modeling and simulation and experimenting on designing Cloud computing infrastructure self-contained platform which can be used to model data centers, hosts, service brokers, scheduling and allocation policies. The future work includes overcoming the problem of deadlocks and server overflow. It can also implement a new service broken policy in the simulator. To have optimized service broker policy. To have new load balancing algorithms in the simulator.

REFERENCES

[1] P.Mell, T. Grance, The NIST De_nition of Cloud Computing, National Institute of Standards and Technology (2009).
[2] I. Foster, Y. Zhao, I. Raicu, S. Lu (2008), Cloud Computing and Grid Computing 360-Degree Compared, in: Proceedings of the 1st Workshop on Grid Computing Environments, Austin, Texas, pp. 1-10.
[3] A. Beloglazov, R. Buyya (2012), Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers, Concurrency and Computation: Practice and Experience 24(13):1397-1420.
[4] L. Wu, S. K. Garg, R. Buyya (2011), SLA-Based Resource Allocation for Software as a Service Provider in Cloud Computing Environments, in: Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Ieee, Newport Beach, California, USA (2011), pp. 195-204.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2015

[5] S. Sadhasivam, N. Nagaveni, R. Jayarani, R. V. Ram (2009), Design and Implementation of an Efficient Two-level Scheduler for Cloud Computing Environment, in: Proceedings of the 2009 International Conference on Advances in Recent Technologies in Communication and Computing, Kottayam, India, pp. 884-886.
[6] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, R. Buyya (2011), CloudSim: a toolkit for modeling and simulation of cloud computing environ ments and evaluation of resource provisioning algorithms, Software - Practice and Experience 41(1):23-50.
[7] C.J.Reynolds, S.Winter,G.Z.Terstyanszky, T. Kiss,P.Greenwell, S. Acs, and P. Kacsuk, ''Scientific Workflow Makespan Reduction through Cloud Augmented Desktop Grids,'' in Proc. 3rd Int'l Conf. CloudCom, 2011, pp. 18-23.
[8] M. Xu, L. Cui, H. Wang, and Y. Bi, ''AMultiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing,'' in Proc. Int'l Symp. ISPA, 2009, pp. 629-634.
[9] M. Mao and M. Humphrey, ''Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows,'' in Proc. Int'l Conf. High Perform. Comput., Netw., Storage Anal. (SC), 2011, p. 49.
[10] M. Rahman, X. Li, and H. Palit, ''Hybrid Heuristic for Scheduling Data Analytics Workflow Applications in Hybrid Cloud Environment,'' in Proc. IPDPSW, 2011, pp. 966-974.
[11] Y.-K. Kwok and I. Ahmad, ''Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors,'' ACM Comput. Surveys, vol. 31, no. 4, pp. 406-471, Dec. 1999.
[12] Z. Shi and J.J. Dongarra, ''Scheduling Workflow Applications on Processors with Different Capabilities,'' Future Gener. Comput. Syst., vol. 22, no. 6, pp. 665-675, May 2006.
[13] E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, ''Cost Optimized Provisioning of Elastic Resources for Application Workflows,'' Future Gener. Comput. Syst., vol. 27, no. 8, pp. 1011-1026,Oct. 2011.
[14] S. Abrishami, M. Naghibzadeh, and D. Epema, ''Deadline-Constrained Workflow Scheduling Algorithms for IaaS Clouds,'' Future Gener. Comput. Syst., vol. 29, no. 1, pp. 158-169, Jan. 2013.

BIOGRAPHY

B. Nithya Nandhalakshmi is a Master of Engineering in the Computer science engineering Department, J.K.K Nattaraja College of Engineering. She received B.E (Information Technology) degree in 2010 from Karunya University, Coimbatore, India. Her research interests are Cloud Computing, Big Data, Data Mining etc.