**International Journal of E-Health and Medical Communications**
Volume 10 • Issue 2 • April-June 2019

# A Novel Performance Enhancing Task Scheduling Algorithm for Cloud-Based E-Health Environment

Vijayakumar Pandi, Department of Computer Science and Engineering, University College of Engineering Tindivanam, Tindivanam, India

Pandiaraja Perumal, Department of Computer Science and Engineering,M.Kumarasamy College of Engineering, Thalavapalayam, India

Balamurugan Balusamy, Department of Computer Science and Engineering, Galgotias University, Greater Noida, India

Marimuthu Karuppiah, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India

## ABSTRACT

The fast-growing internet services have led to the rapid development of storing, retrieving and processing health-related documents from a public cloud. In such a scenario, the performance of cloud services offered is not guaranteed, since it depends on efficient resource scheduling, network bandwidth, etc. The trade-off which lies between the cost and the QoS is that the cost should be variably low on achieving high QoS. This can be done by performance optimization. In order to optimize the performance, a novel task scheduling algorithm is proposed in this article. The main advantage of this proposed scheduling algorithm is to improve the QoS parameters which comprises of metrics such as response time, computation time, availability and cost. The proposed work is simulated in Aneka and shows better performance compared to existing paradigms.

## KEYWORDS

Cloud Computing, E-Health, Quality Of Service, Resource Scheduling, Virtual Machines

## 1. INTRODUCTION

Many Organizations store health related documents in a secure way and provide them to their customers in electronic way which is denoted as E-Health. In such a scenario, to minimize the cost and to improve the performance, the organization used to store their documents in a cloud. Cloud computing is a technology that provides the efficiency, scalability and flexibility for the services it offers. It is a framework implemented for providing anything as a service over the Internet. Cloud is a network grid that relies on virtualization and strives to offer multi-tenancy. It works on a pay-per-service model that delivers on-demand services. Cloud computing deployment models are classified into public cloud, private cloud, hybrid cloud and community cloud. It also comprises of different service models namely software-as-a-service (SAAS), platform-as-a-service (PAAS) and infrastructure-as-a-service (IAAS). SAAS allows cloud users to consume application software over Internet on demand based requests without installations to the local machines. PAAS model is utilized to develop applications and for hosting them on cloud, consists of operating systems, resource allocation and sharing, databases, etc. IAAS forms the base for the other two service models that consists of data centers, physical computing devices and also enables virtualization. Various vendors provide these services directly

DOI: 10.4018/IJEHMC.2019040106

to the cloud users. Some of the well-known cloud giants are Amazon, Microsoft, salesforce, google and IBM. PAAS vendors are google app engine, azure, salesforce, bluecloud, amazon web services and Manjrasoft Aneka. IAAS providers are open nebula, eucalyptus, etc. It consists of stacks, libraries and runtime environment to develop and maintain services across the network (Buyya et al., 2013; Buyya et al., 2010; Ferry et al., 2013). Cloud computing is a technology that enforces a connective environment that allows concurrent execution of services across the network. The accessibility to cloud is made easier by its deployment models, though which utilization of cloud is limited by pay per use model. The distributed applications are migrated to cloud from desktop grids. PAAS model is utilized to develop applications and for hosting it on cloud, consists of operating systems, resource allocation and sharing, databases, etc. Multiple developers use tools to develop web apps provided by PAAS. Though development is constrained to certain languages like java, python, .Net, ruby on rails and few more, it provides efficiency, scalability, interoperability and quick deployment with reduction in cost (Banerjee et al.,2014). Various risk factors are associated with PAAS such as access control, security, technical support from vendors and flexibility. It comprises of user management, resource allocation and database management. The scalability and performance of applications on cloud environment should be comparatively high over traditional distributed computing. To ensure this, various service models support enormous functionalities which offer the quality of service and the performance metrics.

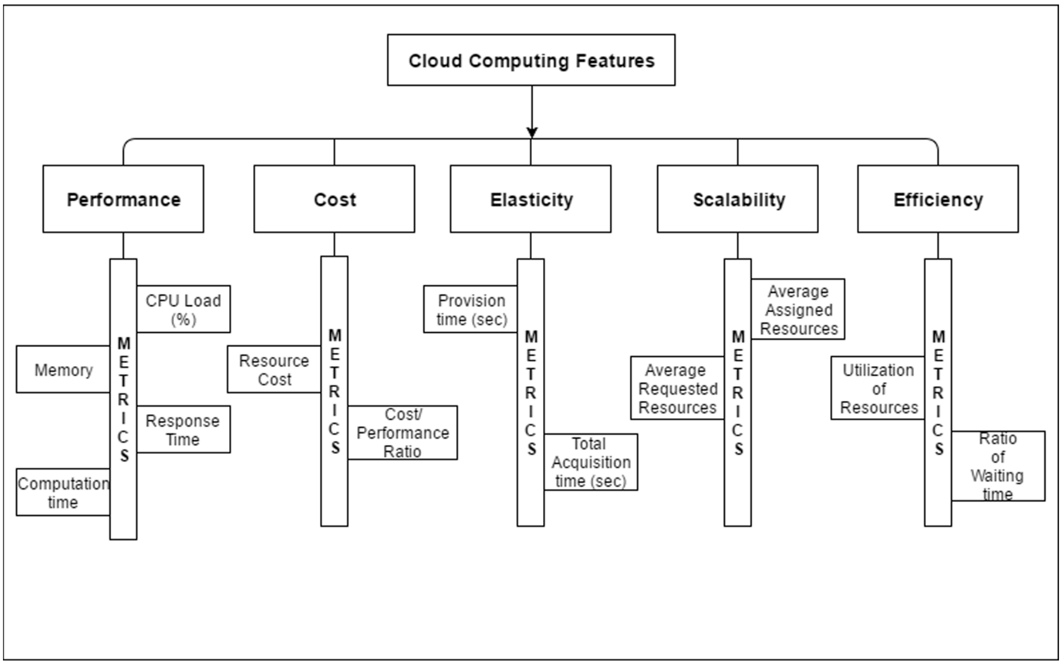
Aneka is purely a PAAS provider and it supports linux and windows platforms. It also allows third party cloud deployment which is useful for dynamic scheduling. Aneka is extensible, i.e. the aneka built-in API's can be used for developing application on top of it, the programming model and algorithms can be extended or customized by the developers. The platform abstraction layer (PAL) in aneka allows the cloud computing platform to work upon different operating systems. This layer interacts with the physical machine on which aneka is installed and supports a run time environment to scale applications on various platforms. The detailed architecture of the aneka cloud is illustrated in (Vecchiola et al., 2009). It comprises of four layers namely application development and management, middleware, PAL layer and infrastructure. Inbuilt API's and various tools are implemented in the application layer. The middleware hosts distinct services offered by aneka, the fabric services, the foundation services, execution services that includes different programming models and the transversal services that provides persistence and security. The cloud paradigm is associated with many challenges while deploying a real time application on top of it. The important aspect for a scalable application is the accessibility, the flexibility and concurrency. Many cloud computing platforms are providing different services, whereas aneka cloud outperforms the other PAAS vendors. Aneka provides the extensibility over the libraries and also portability. Aneka cloud can be utilized for optimizing all kinds of applications.

There are some aspects through which the consumers can evaluate when using a cloud service. The service is measurable in terms of performance and this impacts the quality of service. The QOS based resource scheduling is an important factor for efficient resource utilization and cost optimization. A priority based scheduling for achieving multi-QOS parameters and user satisfaction is proposed and evaluated (Lawrance & Silas, 2013). Resource scheduling can be affected by certain factors namely CPU speed, bandwidth, stability (task states) and task size. The cloud performance needs to be measured to assess the quality of service offered. The evaluation is based upon metrics that varies for each feature in cloud computing (Bardsiri & Hashemi, 2014). The cloud computing features can be measured in order to evaluate the quality of service. Each feature can be measured with the help of certain metrics which is depicted in Figure 1. For scalability, a resource assignment is measured. To check the availability, response time is measured. Likewise, the efficiency of the service is based on proper resource utilization.

## 2. RELATED WORKS

Distributed applications deployed on cloud are being managed across different geographical places. Parallel processing and distributive access to the applications on cloud is essential. Due to the increasing number of users, workload and application management is at risk. In order to recover this issue, a federated inter cloud model is proposed and evaluated in terms of performance by using the simulator tool, Clouds (Buyya et al., 2010). The work compares the various tools that offer cloud computing services to the end users. The comparisons are made for eucalyptus, Microsoft azure, amazon EC2, google app engine, aneka and go grid. These tools are compared for its load balancing and auto-scaling capabilities. The proposed framework is still a conventional method, because federation in clouds deals with a centralized mechanism only. Though it has resulted with less cost and flexibility, decentralized cloud outperforms the centralized cloud. Applications should dynamically adapt to the workloads which change based on the user demands. This improves auto-scaling and load balancing. In (Campos et al., 2015), a performance evaluation model is proposed to improve elasticity of the cloud environment. The Markov chain model is used for the performance analysis based on virtual machines in private cloud. This showed an improvement in performance and increased auto-scaling. Migration performance of cloud applications is an essential parameter, since virtual machine migration and scheduling are the most important factors that enforce the efficient cloud into existence. A quantitative analysis of migration performance is represented in (Kumar & Saxena, 2015) that include parameters such as virtual machine size, network bandwidth and dirty rate of applications on cloud. The migration model is analyzed in CloudSim with respect to the migration time taken. Scientific computing is referred as applications that involve large scientific problems which are traditionally solved by using high performance computing. Some examples of scientific computing apps are brain imaging, ECG recording system, etc. Nowadays scientific computing applications are being migrated to cloud. This created an evolution in cloud models which results in vendors offering QOS based services. Manjrasoft aneka is a framework that provides PAAS for

**Figure 1. Cloud computing general metrics**



developers to create and run applications on cloud with different language base as support. It acts as a user interface with API's and SDKs, and also supports virtualization and IAAS with third party deployment (Vecchiola et al., 2009). Aneka offers various programming models and the models offered can also be extended with customized models. Scientific experiments can be implemented using aneka with its flexible programming models and execution services. A case study of gene expression classification is stated in (Vecchiola et al., 2009), the comparison is done with different classifier models. Aneka acts as a middleman that provides the distributed access and build on amazon EC2 public cloud cluster, S3 for storage services. The QOS levels attained and the performance of scientific applications on cloud is optimized and analyzed. HPC in cloud is addressed and evaluated in (ExpóSito et al., 2013). HPC can be defined as, "High Performance Computing (HPC) most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering or business". The HPC applications are evaluated in terms of performance by using amazon EC2 platform and it is proved by better provisioning and network communication (ExpóSito et al., 2013). In (Gupta et al., 2016), HPC applications for cloud and its performance, type of applications that uses HPC and how effectively HPC applications are hosted on cloud were addressed with the help of analysis using Clouds. By using HPC, the number of users and heterogeneous network of systems are expanded. In accordance with this, the cloud where HPC applications are hosted should provide the scalability and performance in a large scale. Different applications are analyzed across different platforms using cloud services such as EC2, open cirrus and so on (Gupta et al., 2016). The experiments show how optimized cloud for HPC performs and increased throughput reflects how HPC is possible in cloud perspective. However further optimization is required in order to attain QOS and security. Improving the performance of HPC is feasible by migrating them on cloud. Performance of such applications are improved and analyzed over amazon EC2 cloud system in (Hassani et al., 2014). This is implemented using parallel radix sort algorithm, where MPI technique is used for faster task scheduling and execution. An optimal solution by using amazon EC2 cloud is proposed and evaluated for HPC applications (Hassani et al., 2014). Parallel HPC applications are moving their implementation towards cloud where certain improvements need to be deployed in terms of security, scalability and availability. Various applications are being deployed over cloud of distinct providers. The key parameters that improve the performance of the applications are taken into account and platforms/tools are chosen to host these applications. A brief overview about different types of applications on cloud is reviewed. Multiple cloud systems and applications deployed on it is discussed below with the context of the type of application, services utilized, performance required and various programming models.

The scalability and communication performance of HPC applications on windows Azure cloud is tested with certain benchmarks in terms of speed ups, communication among the nodes, kernel performance in the cluster and security (Hassan et al., 2015). Most of the cloud providers are supporting Linux-based platform, only Microsoft azure and Manjrasoft Aneka are windows-based platform that uses .Net framework to build applications on it. Performance analysis on a HPC application using Windows Azure is established and a comparative study is made between normal machine, amazon EC2 and azure platform (Roloff et al., 2012). It is proven that azure performance is good and cost efficient than the other platforms. Also challenges faced during porting application form Linux to windows have been mentioned. In (Ostermann et al., 2009), identified that amazon EC2 cloud is not sufficient to handle application based on scientific computing. Performance evaluation on VM's schedule in accordance with the requests per second and compared between metrics like bandwidth, stream rate, and page views, etc. Further enhancements on the cloud system and efficiency need to be implemented to handle all type of workloads from HPC and scientific applications. In the proposed work, (Masdari et al., 2016) the algorithm workflow is classified into bio-inspired/scientific workflow, heuristic workflow and a combination of both the work flows, which constitutes to the hybrid workflow. These scheduling algorithm are tested with cloud computing deployment models and a comparison with respect to certain factors like time, cost, QOS, etc., are evaluated. A cost optimized approach using scientific workflow scheduling is proposed in (Alkhanak et al., 2016), where cost-based factors are focused in order to yield optimized results. In (Zhang et al., 2016), resource provisioning algorithms are classified under different cloud deployments to obtain optimal solutions. It guides to best algorithm selection for cloud based issues and developments. Optimized performance guarantees quality of services. Many metaheuristic algorithms are used for scheduling process. These algorithms are well known bio-inspired algorithms that can be utilized for getting optimal solutions. The survey of certain algorithms used in cloud is discussed and a comparative analysis is presented in (Kalra & Singh, 2015). A queuing model based scheduling algorithm is proposed that results with improved performance and energy efficient (Cheng et al., 2015). The task waiting time is reduced when compared to other metaheuristic scheduling algorithms. Different ways to attain QOS in cloud computing and various research works on its importance is briefly reviewed (Abdelmaboud et al., 2015). The study focuses on QOS approaches need to be followed with respect to the service models in cloud computing. This also impacts for QOS improvements.

The scientific application is implemented using genetic algorithm, the performance of the application is evaluated in terms of cost and the execution time. ECG data analysis with patient's data in autonomous cloud is experimented using aneka and third party authentication with amazon EC2 infrastructure (Pandey et al., 2012). The dynamic provisioning results with maximum response time, user satisfaction and reduced cost. The challenges associated with distributed applications on aneka include security and virtual machine management. A cloud web portal is developed for testing and deploy application using aneka framework (Alrokayan & Buyya, 2013). It compares the thread and task programming model with its applications and proved that thread model executes faster than task model. The results achieved shows that the execution time can be decreased by increasing the number of worker nodes used. Implementing distributed applications on multi-clouds using aneka integration with Azure and EC2 is evaluated in (Buyya & Barreto, 2015). The work emphasizes the customizable and extensible feature of aneka cloud. The execution time taken by the application on azure and amazon EC2 is compared to establish the integration successful. A prototype of building IOT applications using Amazon EC2 on top of aneka cloud is proposed in (Khodadadi et al., 2015). Cloud workflow engine is designed using aneka framework and workflow model is established to improve the scalability and efficiency of the cloud systems (Zhou et al., 2014).

As a result, all the existing methodology that was used for task scheduling performs in a way that the service time and the task duration are unknown. The values are not predetermined and there is no deadline for a particular task, due to this resources are not properly utilized. Hence, most of the scheduling algorithms are not pre-emptive. Therefore, a new scheduling algorithm is to be proposed to overcome the limitations observed in order to attain maximum performance and reduced cost. This ensures the quality of service to minimize the performance in cloud computing.

## 3. PROPOSED WORK

A scheduling policy for wireless networks with QOS guaranteed is proposed in (Iftikhar et al., 2015). The proposed algorithm uses queuing theory with Markov chain process analysis. The scheduling strategy follows priority based scheduling and also ensures that the QOS parameters are improved. The parameters validated are bandwidth, delay and queue delay time. A queuing theory based model for cloud computing is presented in (Vilaplana et al., 2014). It ensures QOS by validating the response time, arrival time and service rate of the processors. The simulation results show the efficiency of the proposed model in terms of improved performance. A queuing theory based approach is represented in (Maguluri & Srikant, 2014) which is throughput optimal and addressed the previous limitations by scheduling jobs with unknown durations, i.e. job size duration is not bounded to a particular value. The tasks should be efficiently distributed to the resources available, which would help to achieve performance. The traditional FCFS algorithm is extended to handle parallel workloads in cloud environment (Liu et al., 2015). Though the proposed algorithm is faster than conventional FCFS scheduling, the job size and resources required are not estimated. Service oriented resource provisioning is implemented using the aneka framework with integrated amazon EC2 virtual instances (Calheiros et al., 2012). An algorithm of spot instances for task scheduling is discussed and highlighted for the improved performance achieved. A deadline driven mechanism for provisioning resources in hybrid cloud to execute scientific applications using aneka is proposed in (Vecchiola et al., 2012). The algorithm executes the given task based on resource availability and the resource is selected from the resource pool. The deadline driven algorithm supports provisioning services, where resources are provisioned according to the deadline calculated. The resources are provisioned in such a way that it attains the deadline.

Though the deadline driven provisioning algorithm is useful for allocating resources from different sources, it creates a trade off in the QOS offered. The provisioning of resources is optimized in terms of estimating a deadline for the number of task requests. The limitation found here is that the cost will be increased while provisioning resources from public sources. Proper resource utilization should be focused in order to optimize the cost factor. The proposed algorithm in (Vecchiola et al., 2012) is efficient only for certain applications that generates less amount of workload. While scaling a high performance computing application, the workloads will be high and dynamic. To schedule these kinds of workloads, we propose a scheduling algorithm that ensures optimized QOS and efficient resource utilization.

## 3.1. QOS Enhanced Scheduling Algorithm

The proposed scheduling algorithm is based on queuing theory (M/M/c). The algorithm is modeled to improve the performance of the cloud by reducing the total execution time of the application by optimized resource scheduling. It follows pre-emptive scheduling where tasks waiting in queue for a longer time period are rescheduled. This uses join operation to interrupt a thread from its execution. The algorithms is presented below, the terms used are briefed as follows. The tasks, 't' to be executed which will be queued in ready queue, 'Q'. 'R' represents the resources which can be statically or dynamically deployed, desktop machines and virtual machines. The similar tasks are grouped together under separate groups, G (G exists from 1 to n). The value 'S' denotes the current state at which the task is present. The task deadline, D is estimated by the product of resource capacity and the waiting time. The arrival time of each task is taken into account to estimate the waiting time. The probability mass function is used to calculate the arrival time for tasks. The arrival time of tasks are not known, hence it is stochastic, is assumed as random. The arrival time is calculated by continuous-time Markov process.

```
1         Begin
2         Load entry task t into ready queue Q
3         t ← task
4         R ← available resources for the request submitted
5         G ← group of tasks that requires same service rates to
complete
6         S ← state at which the task is currently present
7         foreach task with time t
8         t ← first task from Q
9         for t(1 to N) tasks do
10         Calculate arrival time of each task,t
11         Arrival time, at ← probabilistic mass function
12         End for
13         Task deadline, D ← (capacity of resource, c) ×(Waiting
time, w) - 1
14         Where c← average no. of tasks completed by R in a unit
```

```
time
15          W ← total time required for a task from s1 to s2
16          for t do
17          get R with min(D)
18          if min(D)<=c
19          schedule t
20          else
21          reschedule t with R having C> D
22          end
23          if R > tasks in ready queue
24          release ← no. of task in waiting state - R
25          end for
26          for next task to be scheduled
27          add tasks to Q
28          repeat step 12 to 25
29          end foreach
30          end
```

The input of the algorithm is the request submitted by the user at a unit time and resource required for each task, its capacity and price. A task state varies from queued, scheduled, completed, aborted, failed and rescheduled. The threshold value for the capacity of resource is taken from the execution logs of resource by the scheduler.

## 3.2. Workload

Workload in cloud computing is a challenging factor which is dynamic and more complex. The management of different kinds of workloads promotes performance and guarantees QOS. Workload modeling to ensure proper resource usage analysis is presented (Magalhães et al., 2015). The work also implements a model for web application to handle dynamic workloads. The simulation results show the efficiency of the proposed model. Workload can simply be defined as a task or job that needs to be completed by a system. The tasks required to complete certain actions of a system. For example, for a web application when the user clicks on a button, a request and response from both the user and the server side are referred as the tasks. This transforms as a workload on a large scale usage, i.e. when multiple users are active on the same application. An energy efficient task scheduling model is proposed for cloud data centers (Dong et al., 2015). The model is developed with the usage of greedy algorithm on constraint based scheduling that saves energy. The workload includes large number of tasks that demands quick response time and proper resource utilization. Task deadlines and proper resource provisioning can be used to handle different workloads.

## 3.3. Scheduling Strategy

The stochastic process is referred as chains, where the variables are random. Markov process constitutes the fundamental theory underlying the concept of queuing systems. The queuing theory forms the basis of scheduling. The arrival time for each task/job in queue can be calculated using continuous-time Markov chains.

T denotes all tasks, $T_{t1}, T_{t2}, T_{t3}, \ldots, T_{tn}$ where each task in queue varies with different time t. $t_0 < t_1 < t_2 < \ldots\ldots < t_n < t_{n+1}, \forall n \in N$ and $\forall S_i \in S = N_0$ for the conditional probability mass function, $T_{tn+1}$ depends only on the last previous value, $T_{tn}$ and not on the earlier values, $T_{t1}, T_{t2}, T_{t3}, \ldots, T_{tn-1}$.

$$\textbf{Arrivaltime,} P\left(T_{tn+1} = S_{n+1} \mid T_{tn} = S_n, T_{tn-1} = S_{n-1}, \ldots.., T_{t0} = S_0\right) = P\left(T_{tn+1} = S_{n+1} \mid T_{tn} = S_n\right)$$

**Waiting time,** the mean waiting time is distributed exponentially and it is represented as $P_{ij}(u,v)$ $= P(T_v = j \mid T_u = i)$. $P_{ij}(u,v)$ the continuous time markov chain to travel from state i to state j during the period of time (u, v) where u<=v.

**Deadline estimation, D** is the product of resource capacity, c and waiting time of task, w D = (c × w) - 1

The queue length, $\overline{Q} = \sum_{x=1}^{\infty} x.\prod_x$ where $\prod_x = P[x\_jobs\_in\_Queue]$.

Resource utilization, $\psi = \dfrac{\lambda}{m\mu}$ where $\lambda$ is the arrival rate and μ is the service rate for m machines (multiple machines). To reschedule a task, the response time of each task and expected completion time of the task is recorded.

The scheduling strategy deals with calculating the inter arrival time of each tasks in the queue followed by estimating the task waiting time. This helps to formulate the deadline for a task. A task with minimum deadline is scheduled first and resources are released when not utilized. The main factor that helps to overcome the deadline driven algorithm is task completion time. In the existing deadline driven scheduling algorithm [20], the deadline is calculated for resource provisioning. However, this kind of scheduling involves more number of resources on an average. The proposed algorithm overcomes this by means of proper resource utilization. This is achieved by estimating the deadline with resource capacity and waiting time of tasks. The working and efficiency of the QOS enhanced scheduling algorithm is evaluated and experimented using aneka. The experimental setup and analysis is presented in the following section.

## 4. EXPERIMENTAL SETUP

The experimentation is done by using the proposed scheduling algorithm in aneka cloud. The aneka cloud is created by adding local and virtual machines. The master and worker containers together form the aneka cloud deployment. Initially, the local machine is added by installing a master and three worker containers. To create a private/public cloud, third party cloud provider can be integrated. We use the Xeon server to obtain dynamic resources. The virtual machines are managed by the Xeon server pool and resource pool manager in aneka. The VM instances are also connected to Aneka by means of resource provisioning service. The machines provide 1.3 GB of memory and 20GB storage with windows and Linux platforms. The proposed algorithm can be implemented inside aneka using the scheduling service. The work uses thread programming model, one of the model offered by aneka which is comparatively faster and enables parallel processing and supports distributed applications. The model uses

the thread scheduler where the scheduler uses an algorithm for scheduling the tasks to execute the application. By extending the scheduling service, the novel algorithm can be implemented and

**Table 1. Components**

| Containers | Static Resources | Dynamic Resources | Consumption |
|---|---|---|---|
| Aneka Master | tcp://192.168.1.108:9090/localhost | 1 Xen server VM | 141.77 |
| Aneka Workers | tcp://192.168.1.108:9092 | 2VM instances Xen server pool | 1.57 |
| | tcp://192.168.1.108:9093 | | 2791.21 |
| | tcp://192.168.1.108:9096 | | 20.358 |

used for application scheduling. The aneka cloud components and resources provisioned are listed in Table 1 with average resource consumptions.

The applications that can be developed using aneka can be a web application, multithreading application, scientific computing or also analytics application. All these applications can be executed using task or thread model except the analytics application which deals with large data can be executed using the Map Reduce programming model. We developed accounting software that can be utilized for storing and retrieving data across different geographical areas. It supports multi-tenancy and can be accessed anywhere at any time. This actually fulfills a basic constraint of a cloud application. The application can be used by companies for certain services like project management, ticket raising, employer's details and communication, screen sharing, etc. A prototype that facilitates this kind of application is developed using aneka libraries. The developed application runs on aneka using the thread programming model. Firstly, the application is executed using the deadline driven scheduling algorithm which is existing in aneka. Later on, the same application is executed by our proposed QOS Enhanced scheduling algorithm. The observed improvements are marked in Table 2 with respect to the total execution time of the application.

The experimentation shows that the application is faster while using the proposed algorithm. The total execution time of the application is represented in milliseconds. The algorithm is efficient in terms of resource utilization and tasks execution. These kinds of applications will have dynamic workloads. The performance of the application should be high and for achieving this, we use our proposed algorithm which helps to get optimized performance that yields enhanced QOS. The task deadlines are followed and minimum of those is executed first followed by the rest. Also the pre-emptive scheduling reduces the average waiting time.

## 5. PERFORMANCE EVALUATION

The results show that the proposed algorithm is efficient in utilization of resources when compared with the deadline driven scheduling algorithm. This shows improved performance and efficiency of the proposed algorithm. Performance is increased with reduced execution time and optimized resource utilization is observed. As a result, cost associated with resources can also be reduced. The QoS parameters achieved are performance, availability and cost. The application is distributed and optimized when using the thread programming model. The performance metrics are compared to the conventional scheduling algorithm and the efficiency of the proposed model is analyzed. The novel algorithm ensures improved QoS of the application deployed over aneka. The QoS parameters achieved are the performance and cost. The availability can be achieved by provisioning on demand resources on the basis of requests. Performance is measured by reducing the total execution time of the application. Cost can be optimized by proper resource utilization. Figure 2 clearly shows that performance is increased in our proposed work when number of nodes is increased.

**Table 2. Comparative analysis on total execution time of the application**

| Application | Cloud | Submitted Node | Total task Size | Duration (ms) | |
|---|---|---|---|---|---|
| | | | | Deadline driven Scheduling | QOS Enhanced Scheduling |
| Accounting Software | Private Cloud | tcp://192.168.1.108:9093 | 100 | 333135 | 84122 |
| | Private Cloud | XenServer instance (VM) | 1200 | 612745 | 603726 |
| | Private Cloud | tcp://192.168.1.108:9093 | 41 | 304682 | 302576 |
| | Private Cloud | tcp://192.168.1.108:9092 | 60 | 72156 | 72124 |
| | Private Cloud | XenServer instance (VM) | 130 | 300022 | 245670 |

Figure 2. Performance improvement

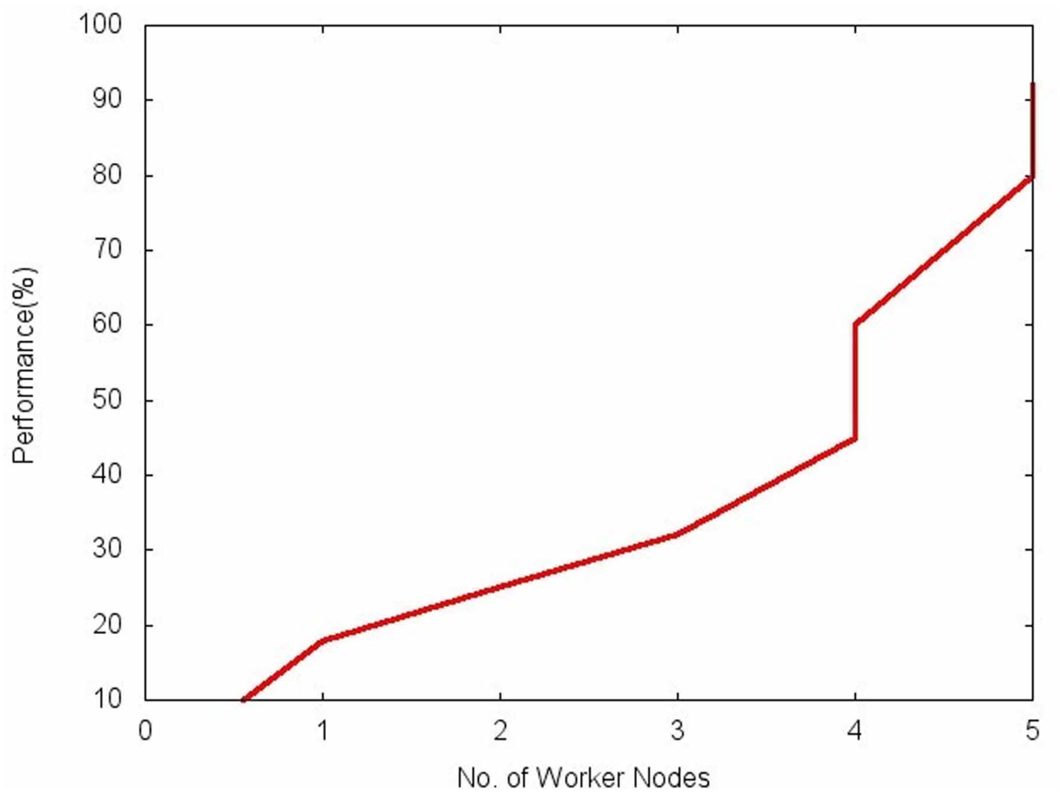


Figure 3 shows the efficiency of our proposed algorithm with existing algorithm. It is very clear to understand from the figure is that our proposed algorithm takes less execution time in comparison with the existing algorithm. Figure 4 shows the execution time when more tasks are assigned to a particular node. From this figure, it is easy to observe that our thread programming model takes less execution time and increased efficiency of our proposed algorithm with existing algorithm. It is very clear to understand from the figure is that our proposed algorithm takes less execution time in comparison with the existing algorithm. Figure 5 shows energy efficiency of our proposed work. Increasing the worker node instances result with faster execution. No. of jobs / tasks relies to peak load to the system. The graphs depicted above shows (Figure 2 to Figure 5), the Total Execution time is inversely proportional to Performance and Efficiency. Proper resource utilization constitutes to better cost.

## 6. CONCLUSIONS

The proposed algorithm results yields increased performance and optimized resource utilization which results in enhanced QoS. The QoS Enhanced Scheduling Algorithm is comparatively faster than the deadline driven scheduling and hence it is suitable for more efficiently storing and retrieving E-health documents from the cloud. By means of the proposed algorithm, the total execution time of the applications deployed on aneka cloud is reduced. In future, the work can be extended by creating a hybrid environment for the evaluation of high performance computing applications. Also, the thread programming model can be extended and a new programming model can be developed which can be utilized for deploying distributed and high performance computing applications.

**Figure 3. Algorithm efficiency**

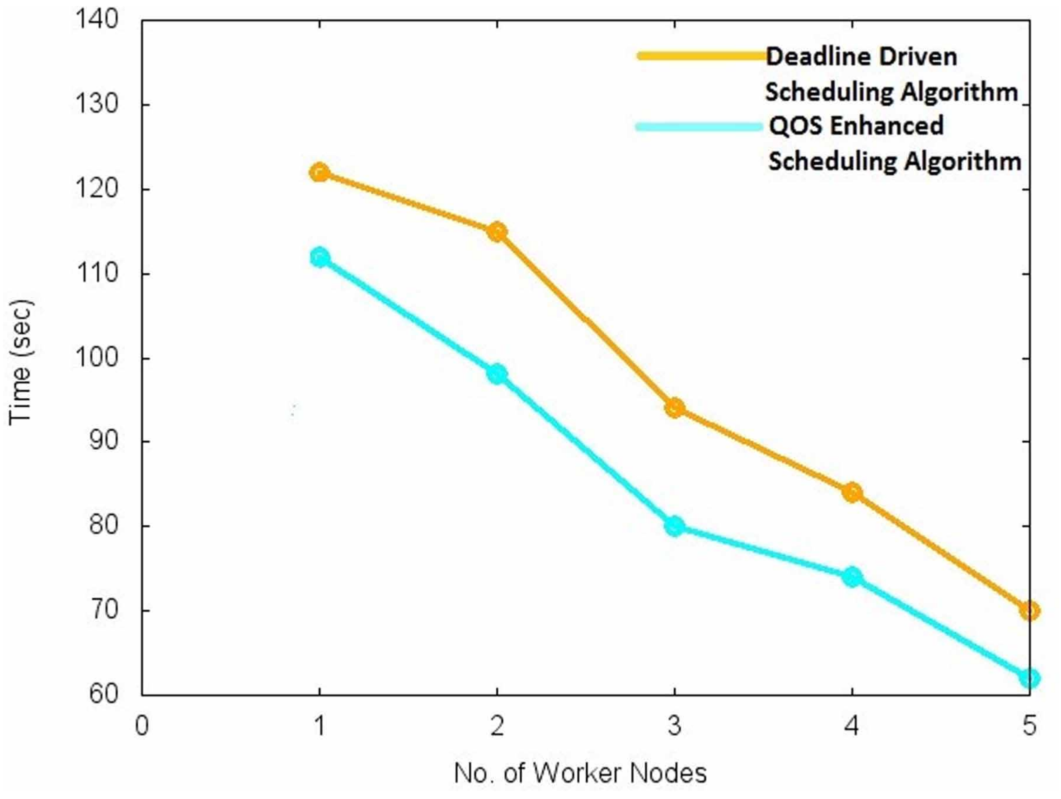


**Figure 4. Comparative analysis of task and thread programming model**

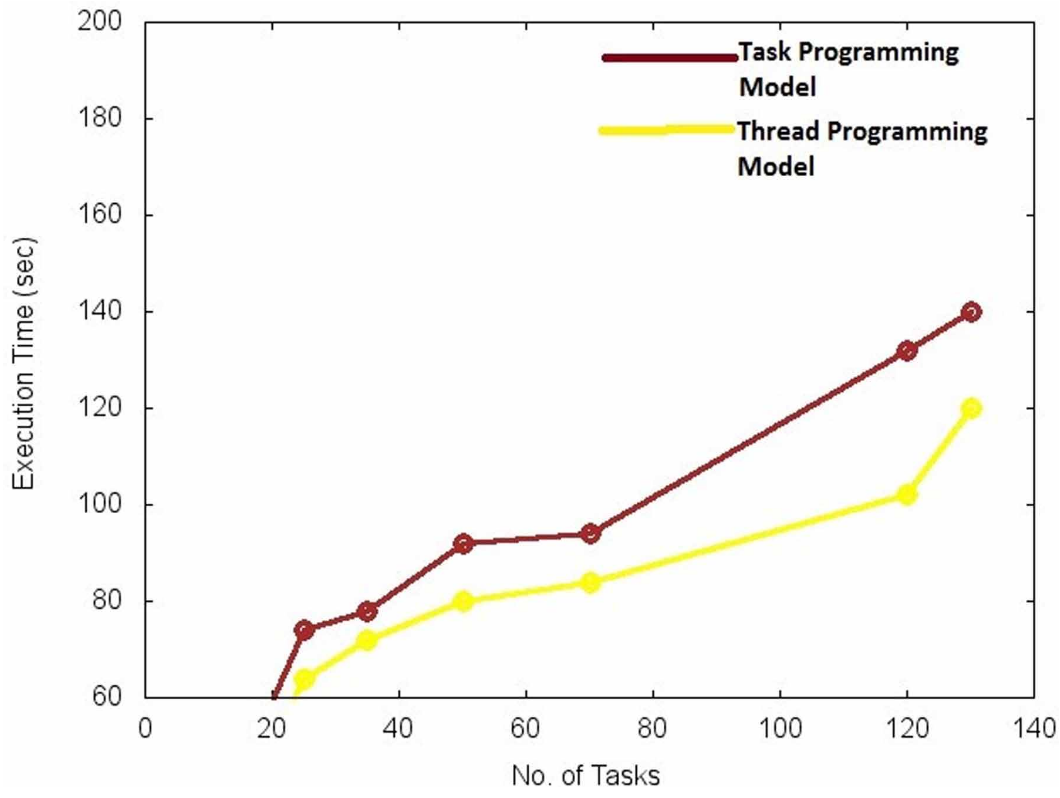
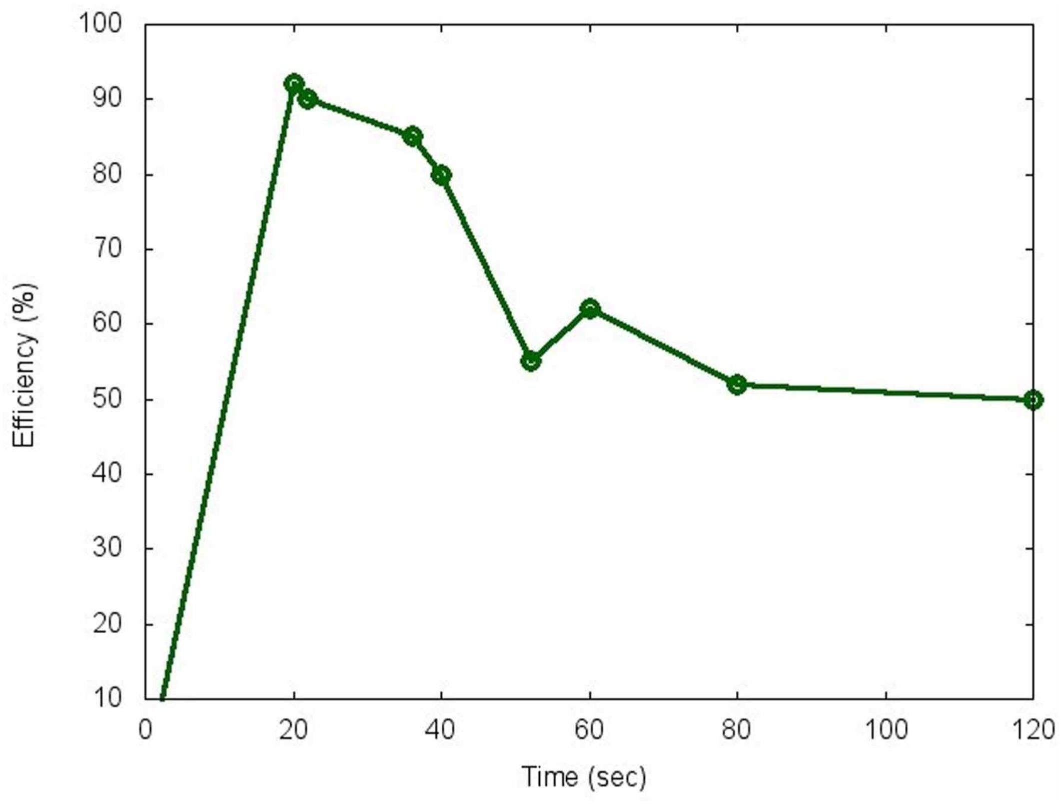


**Figure 5. Energy efficiency**



## REFERENCES

Abdelmaboud, A., Jawawi, D. N., Ghani, I., Elsafi, A., & Kitchenham, B. (2015). Quality of service approaches in cloud computing: A systematic mapping study. *Journal of Systems and Software*, *101*, 159–179. doi:10.1016/j.jss.2014.12.015

Alkhanak, E. N., Lee, S. P., Rezaei, R., & Parizi, R. M. (2016). Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues. *Journal of Systems and Software*, *113*, 1–26. doi:10.1016/j.jss.2015.11.023

Alrokayan, M., & Buyya, R. (2013). A Web portal for management of Aneka-based MultiCloud environments. In *Proceedings of the Eleventh Australasian Symposium on Parallel and Distributed Computing* (pp. 49-56).

Banerjee, S., Gupta, N., & Gupta, V. (2014). Implementation and Management of framework for PaaS in Cloud Computing. *International Journal of Innovations & Advancement in Computer Science*, *2*(2), 38–49.

Bardsiri, A. K., & Hashemi, S. M. (2014). QoS metrics for cloud computing services evaluation. *International Journal of Intelligent Systems and Applications*, *6*(12), 27–33. doi:10.5815/ijisa.2014.12.04

Buyya, R., & Barreto, D. (2015). Multi-Cloud Resource Provisioning with Aneka: A Unified and Integrated Utilisation of Microsoft Azure and Amazon EC2 Instances. arXiv:1511.08857

Buyya, R., Broberg, J., & Goscinski, A. M. (2010). *Cloud computing: principles and paradigms*. John Wiley & Sons.

Buyya, R., Ranjan, R., & Calheiros, R. N. (2010). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In Algorithms and architectures for parallel processing (pp. 13-31). Springer.

Buyya, R., Vecchiola, C., & Selvi, S. T. (2013). *Mastering cloud computing: foundations and applications programming*. Newnes.

Calheiros, R. N., Vecchiola, C., Karunamoorthy, D., & Buyya, R. (2012). The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds. *Future Generation Computer Systems*, *28*(6), 861–870. doi:10.1016/j.future.2011.07.005

Campos, E., Matos, R., Maciel, P., Costa, I., Silva, F. A., & Souza, F. Performance. (2015). Evaluation of Virtual Machines Instantiation in a Private Cloud. In *Proceedings of the IEEE 2015 Services (SERVICES)* (pp. 319-326). doi:10.1109/SERVICES.2015.55

Cheng, C., Li, J., & Wang, Y. (2015). An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing. *Tsinghua Science and Technology*, *20*(1), 28–39. doi:10.1109/TST.2015.7040511

Dong, Z., Liu, N., & Rojas-Cessa, R. (2015). Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. *Journal of Cloud Computing*, *4*(1), 1–14.

Expósito, R. R., Taboada, G. L., Ramos, S., Touriño, J., & Doallo, R.ExpóSito. (2013). Performance analysis of HPC applications in the cloud. *Future Generation Computer Systems*, *29*(1), 218–229. doi:10.1016/j.future.2012.06.009

Ferry, N., Rossini, A., Chauvel, F., Morin, B., & Solberg, A. (2013). Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In *Proceedings of the Sixth IEEE International Conference on Cloud Computing (CLOUD)* (pp. 887-894). doi:10.1109/CLOUD.2013.133

Gupta, A., Faraboschi, P., Gioachin, F., Kale, L. V., Kaufmann, R., Lee, B. S., & Suen, C. H. (2016). Evaluating and Improving the Performance and Scheduling of HPC Applications in Cloud. *IEEE Transactions on Cloud Computing*, *4*(3), 307–321. doi:10.1109/TCC.2014.2339858

Hassan, H. A., Mohamed, S. A., & Sheta, W. M. (2015). *Scalability and communication performance of HPC on Azure Cloud*. Egyptian Informatics Journal.

Hassani, R., Aiatullah, M., & Luksch, P. (2014). Improving HPC Application Performance in Public Cloud. *IERI Procedia*, *10*, 169–176. doi:10.1016/j.ieri.2014.09.072

Iftikhar, M., Mathkour, H., Imran, M., Bedaiwi, A., & Vasilakos, A. V. (2015). A novel framework for G/M/1 queuing system based on scheduling-cum-polling mechanism to analyze multiple classes of self-similar and LRD traffic. *Wireless Networks*, *22*(4), 1269–1284. doi:10.1007/s11276-015-1001-5

Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud computing. *Egyptian Informatics Journal*, *16*(3), 275–295. doi:10.1016/j.eij.2015.07.001

Khodadadi, F., Calheiros, R. N., & Buyya, R. (2015). A data-centric framework for development and deployment of Internet of Things applications in clouds. In *Proceedings of the tenth IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)* (pp. 1-6). doi:10.1109/ISSNIP.2015.7106952

Kumar, N., & Saxena, S. (2015). Migration Performance of Cloud Applications-A Quantitative Analysis. *Procedia Computer Science*, *4*, 823–831. doi:10.1016/j.procs.2015.03.163

Lawrance, H., & Silas, S. (2013). Efficient QoS based resource scheduling using PAPRIKA method for cloud computing. *International Journal of Engineering Science and Technology*, *5*(3), 638.

Liu, X., Zha, Y., Yin, Q., Peng, Y., & Qin, L. (2015). Scheduling parallel jobs with tentative runs and consolidation in the cloud. *Journal of Systems and Software*, *104*, 141–151. doi:10.1016/j.jss.2015.03.007

Magalhães, D., Calheiros, R. N., Buyya, R., & Gomes, D. G. (2015). Workload modeling for resource usage analysis and simulation in cloud computing. *Computers & Electrical Engineering*, *47*, 69–81. doi:10.1016/j.compeleceng.2015.08.016

Maguluri, S. T., & Srikant, R. (2014). Scheduling jobs with unknown duration in clouds. *IEEE/ACM Transactions on Networking*, *22*(6), 1938–1951. doi:10.1109/TNET.2013.2288973

Masdari, M., ValiKardan, S., Shahi, Z., & Azar, S. I. (2016). Towards workflow scheduling in cloud computing: A comprehensive analysis. *Journal of Network and Computer Applications*, *66*, 64–82. doi:10.1016/j.jnca.2016.01.018

Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., & Epema, D. (2009). A performance analysis of EC2 cloud computing services for scientific computing. In Cloud computing (pp. 115-131). Springer.

Pandey, S., Voorsluys, W., Niu, S., Khandoker, A., & Buyya, R. (2012). An autonomic cloud environment for hosting ECG data analysis services. *Future Generation Computer Systems*, *28*(1), 147–154. doi:10.1016/j.future.2011.04.022

Roloff, E., Birck, F., Diener, M., Carissimi, A., & Navaux, P. O. (2012). Evaluating high performance computing on the windows azure platform. In *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD)* (pp. 803-810). doi:10.1109/CLOUD.2012.47

Vecchiola, C., Calheiros, R. N., Karunamoorthy, D., & Buyya, R. (2012). Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka. *Future Generation Computer Systems*, *28*(1), 58–65. doi:10.1016/j.future.2011.05.008

Vecchiola, C., Chu, X., & Buyya, R. (2009). Aneka: A software platform for. NET-based cloud computing. *High Speed and Large Scale Scientific Computing*, *18*, 267–295.

Vecchiola, C., Pandey, S., & Buyya, R. (2009). High-performance cloud computing: A view of scientific applications. In *Proceedings of the IEEE 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN)* (pp. 4-16).

Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Abella, F., & Rius, J. (2014). A queuing theory model for cloud computing. *The Journal of Supercomputing*, *69*(1), 492–507. doi:10.1007/s11227-014-1177-y

Zhang, J., Huang, H., & Wang, X. (2016). Resource provision algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, *64*, 23–42. doi:10.1016/j.jnca.2015.12.018

Zhou, J., Sun, C., Fu, W., Liu, J., Jia, L., & Tan, H. (2014). Modeling, design, and implementation of a cloud workflow engine based on aneka. Journal of Applied Mathematics.

*P.VijayaKumar completed his Ph.D in Computer Science and Engineering in Anna University Chennai in the year 2013. He completed Master of Engineering in the field of Computer Science and Engineering in Karunya Institute of Technology in the year 2005. He completed his Bachelor of Engineering under Madurai Kamarajar University in the year 2002. He is presently working as Dean at University College of Engineering, Tindivanam. He is guiding for many Ph.D scholars in the field of network and cloud security. He has published various quality papers in the reputed journals like IEEE Transactions, Elsevier, Springer, IET, Taylor & Francis, Wiley etc. His main thrust research areas are Key management in Network Security and Multicasting in Computer Networks. Pandiaraja Perumal received a Bachelor of Engineering in Computer Science and Engineering Degree from Anna University, Chennai, India, in 2006 and the Master of Engineering Degree in Computer Science and Engineering from Anna University, Chennai, India in 2009.*

*Balamurugan Balusamy is an Associate Professor in School of Information Technology and Engineering,VIT University,Vellore,India. Marimuthu Karuppiah received his B.E. degree in Computer Science & Engineering from Madurai Kamaraj University, Madurai, India in 2003, M.E. degree in Computer Science & Engineering from Anna University, Chennai, India in 2005. In 2005, he has appointed as a Lecturer in Department of Computer Science & Engineering at Syed Ammal Engineering College, Ramanathapuram, India. He is now an Assistant professor (Senior) in School of Computing Science & Engineering, VIT University, Vellore, India. Now, he is pursuing his Ph.D. degree at VIT University, Vellore, India. He has published more than 10 research papers in reputed international conferences and journals. He is a life member of Cryptology Research Society of India (CRSI). His main research interests include cryptography and wireless network security, in particular, authentication and encryption schemes.*