An Optimal Model for Priority based Service Scheduling Policy for Cloud Computing Environment

Dr. M. Dakshayini
Dept. of ISE,
BMS College of Engineering,
Bangalore, India.

Dr. H. S. Guruprasad
Dept. of ISE,
BMS College of Engineering
Bangalore, India.

ABSTRACT
Cloud computing refers to the model, which is the pool of resources. Cloud makes on-demand delivery of these computational resources (data, software and infrastructure) among multiple services via a computer network with different load conditions of the cloud network. **U**ser will be charged for the resources used based upon time. Hence efficient utilization of cloud resources has become a major challenge in satisfying the user's requirement (QoS) and in gaining benefit for both the user and the service provider. In this paper, we propose a priority and admission control based service scheduling policy that aims at serving the user requests satisfying the QoS, optimizing the time the service-request spends in the queue and achieving the high throughput of the cloud by making an efficient provision of cloud resources.

Keywords
Subscription-category, service scheduling policy, Priority, admission control and deadline.

1. INTRODUCTION
Cloud computing provides on-demand delivery of various services like computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. As the user is charged for the resources used, resource scheduling strategy plays significant role in cloud computing environment. Users or clients can submit a job to the service provider, without actually possessing the software or hardware. The consumer's computer may contain very little software or data (perhaps a minimal operating system and web browser only), serving as a basic display terminal connected to the Internet. Earlier, both data and software had to be stored and processed on or near the computer. The design of cloud computing technology allows the functional separation between the resources used and the user's computer, usually residing outside the local network. Consumers regularly use data intensive applications driven by cloud computing technology earlier which were unavailable due to the cost and the complexity involved in deployment [1,4]. An analogy to explain cloud computing is that of public utilities such as electricity, gas, and water. Centralized and standardized utilities have made individuals free from the difficulties of generating electricity or pumping water. The development and maintenance tasks involved were drastically reduced with cloud technology.

It is very much useful for small organizations that cannot afford huge investment on their IT sector but in order to survive in today's complex competitive business world they expect maximum benefit from such supporting industry. Cloud computing can help such small organizations by providing massive computing power, unlimited storage capacity, less maintenance cost, availability of useful web-services etc [2].

As per Buyya et. all [3], "A Cloud is a type of parallel as well as distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumer". The definition clearly implies that there is a Service Level Agreement (SLA) between the provider and the consumer for getting services from cloud on pay per use basis. Hence efficient scheduling system is one of the core and challenging area in cloud and grid computing.

Actually in Cloud computing there are three types of services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS). SaaS provides different types of applications as a Service for the end user. It includes different useful web-services. PaaS provides a standard platform for better execution of application with proper exploitation of physical resources. PaaS includes Database services, Middleware Services etc, IaaS provides the infrastructure of cloud consisting of physical resources like CPU, Storage, and Network etc [3-5].

The rest of this paper is organized as follows. In Section 2, related works in the area are discussed. Section 3 analyzes the various system parameters used in the system model. Proposed cloud architecture and the scheduling policy are described in Section 4. Section 5 describes simulation model and performance evaluation. Finally, in section 6, we conclude our work and refer to future work.

2. RELATED WORK
In case of Cloud computing environment, there are some critical QoS parameters to be considered, such as time, cost (service charge for the user and servicing charge for provider), reliability and trust/security. In particular, QoS requirements are not static and need to be updated dynamically over the time due to continuous changes in the operating environments. That is greater importance should be given to user's time as they pay for using services from the Clouds based on time. In addition, dynamic negotiation of SLAs between the users and the service provider is not completely supported in the Cloud computing environment. S. Venugopal, X. Chu, and R. Buyya [9] have

developed negotiation mechanisms based on alternate offers protocol for establishing SLAs. Rajkumar Buyya, Chee Shin Yeo and Srikumar Venugopal have presented a 21st century vision of computing in their keynote paper [3]. They also have identified various computing paradigms promising to deliver the vision of computing utilities. Cloud computing definitions and the architecture for creating market-oriented Clouds by leveraging technologies such as VMs are also discussed. Buyya et al. [8] have proposed scheduling policies to address the time minimization and cost minimization problem in the context of Grid computing. K.Mukherjee, G.Sahoo, [2] have given a Mathematical Model for Market-Oriented Cloud Computing. They also have proposed a Bee and Ant colony system based scheduling policy. Qiang Li and Yike Guo [10] have proposed a model for resource scheduling in cloud computing based on stochastic integer programming technique, but none of these papers have considered the concept of admission control and priority of user's requests. In this paper we are proposing an efficient priority based scheduling policy [PSP] and the supporting cloud architecture with appropriate components to achieve optimization. We also have incorporated an admission control technique based on deadline of the service-request, that allows the cloud to accept the service-requests only if the cloud can provide the service satisfying the required QoS.

3. SYSTEM MODEL

According to the analysis of the behavior of the cloud computing network with multiple servers and service-requests for service, the cloud can be considered as a pool of resources. The priority based group of service-requests in the cloud computing environment can be considered as M/G/c queue, and all the queues together can make a queuing network. Hence applying multiple server queueing system, we give a model for the proposed Priority based scheduling policy [PSP]. The parameters considered in this system model are listed in the table 1.

Table 1. Parameters considered for the queuing system model

Parameter	Definition
Q_H	High Priority Queue
Q_M	Medium Priority Queue
Q_L	Low Priority Queue
$\|Q_H\|$	Size of Q_H
$\|Q_M\|$	Size of Q_M
$\|Q_L\|$	Size of Q_L
T1	Threshold time for deadline at level 1
T2	Threshold time for deadline at level 2
λ	Mean request arrival rate
λ_e	Effective arrival rate
W_i^Q	Total time a Req_i spends in the queue
μ_i	Total service time taken by the Req_i
W	Average time spent by the user request in the cloud
P	Server utilization

We define the PSP with the following assumptions

a. Subscription: Before demanding for the service every user must subscribe themselves to the cloud manager using one of the 3 subscription category (SB_{CAT}).

   SB_{CAT} – High,

   SB_{CAT} - Medium or

   SB_{CAT} - Low.

   For each subscription category subscription charge varies.

b. Requests arrival pattern: The user's request arrivals occur randomly according to a Poisson distribution with λ arrivals per unit time.

c. SLA between the cloud providers and the cloud users: is an agreement on guaranteed high quality service and cost for the service

d. *QoS* of the service requests: There are 3 attributes: guaranteed service, high quality service and cost for the service

e. Queue behavior: Request is selected from one of the three queues based on the priority.

   In this model, whenever the request for the service from the user *i* (service-request^i) arrives at the cloud, the *Req-control-mgr* estimates the service time ST_{est}^i required to complete that service-request^i based on the type of the service-request and the average service time taken (by experience) for that type of service-request.

$$delay_i^T = (DL_i^T - C_i^T) \text{ ---------- } (1)$$

where

C_i^T - Current time

DL_i^T - Deadline given by service-request^i

$delay_i^T$ - Maximum tolerable time of ith ser-req

Req-control-mgr also estimates the total servicing time required ST_{est}^T for all the service requests in all the three queues (Y) and is computed as:

$$ST_{est}^T = \sum_{i=1}^{Y} ST_{est}^i \text{ ---------- } (2)$$

Where $Y = |Q_H| + |Q_M| + |Q_L|$

According to the tolerable delay computed ($delay_i^T$) and SB_{CAT} of ith service-request (ser-req^i), arrived service-request will be placed in one of the 3 queues (Q_H – High priority queue, Q_M - Medium priority queue or Q_L-Low priority queue)

{ If (($delay_i^T$ = ((ST_{est}^T /C) + ST_{est}^i)) and

SB_{CAT} ==1or 2 or 3)

Place the *ser-req*^{i} in Q_H

If (($delay_i^T$ > ((ST_{est}^T /C) + ST_{est}^i) by T_1))

{ If and SB_{CAT} ==1)

Place the *ser-req*^{i} in Q_H

If and SB_{CAT} ==2)

Place the *ser-req*^{i} in Q_M

If and SB_{CAT} ==3)

Place the *ser-req*^{i} in Q_L }

If (($delay_i^T$ > ((ST_{est}^T /C) + ST_{est}^i) by T_2))

{ If and SB_{CAT} ==1)

Place the *ser-req*^{i} in Q_H or in Q_M based on the space availability

If and SB_{CAT} ==2)

Place the *ser-req*^{i} in Q_M or in Q_L based on the space availability

If and SB_{CAT} ==3)

Place the *ser-req*^{i} in Q_L based on space availability }

} ---------- (3)

With this system model, the probability that there are *N* customers in the system is P_N . The effective arrival rate, that is the mean number of arrivals per time unit who enter and remain in the system is λ_e.

$$\lambda_e = \lambda (1 - P_N) \quad \text{---------- (4)}$$

The total time a *ser-req*^{i} (w_i) spends in the cloud is:

$$w_i = w_i^Q + \mu_i \quad \text{---------- (5)}$$

Where

w_i^Q - is the total time a *ser-req*^{i} spends in the queue

μ_i - is the total service time taken by the *ser-req*^{i}

The average time spent by the user request *ser-req*^{i} in the cloud is the average *req-to-service delay/ ser-req* in the Cloud.

$$w = \frac{1}{N}\sum_{i=1}^{N} w_i \quad \text{---------- (6)}$$

The average number of user requests being served in the cloud is the average *server utilization p*

$$P = \frac{\lambda_e}{\sum_{i=1}^{C} \mu_i}$$

Where

C – is the number of servers in the system

The maximum delay $delay_i^T$ the *ser-req*^{i} can tolerate is specified in SLA. To meet this QoS requirement for the *ser-req*^{i}

$$w_i < \text{delay}_i^T$$

In addition to meet the QoS requirement of the service-request, the profit ω_i for the cloud in providing the service to user-requests should also be considered and computed.

We assume that:

- The unit of cost for service from the cloud as ∂/min
- Total amount of unit time for which service is provided to *ser-req*^{i} as H_μ .

Charge per Request *ser-req*^{i} is

$$\omega_i = H_\mu * \partial \text{/min}$$

Total profit of servicing all the requests

$$\omega = \sum_{i=1}^{|QH|} \omega_i + \sum_{j=1}^{|QM|} \omega_j + \sum_{k=1}^{|QL|} \omega_k$$

Our optimization problem is to

- Minimize the total time a *ser-req*^{i} spends in the queue Min w_i^Q
- Guaranteed high quality of service

  $w_i \leq delay_i^T$
- Maximizing the throughput

4. PRIORITY BASED SCHEDULING ALGORITHM AND ARCHITECTURE

4.1 Architecture

Our proposed architecture consists of 2 levels, cloud service provider level (SPL) and user level (UL). SPL provides a set of services to the user with suitable communication among several components of the cloud. Various components of the cloud are "Request Control-Manager (*Req-Cntr-mgr*)", "Service Manager (*Ser-mgr*) in association with Resource usage Accounting-Manager" (*Res-mgr*) etc; whereas the UL provides secured access point between the user and the service provider.

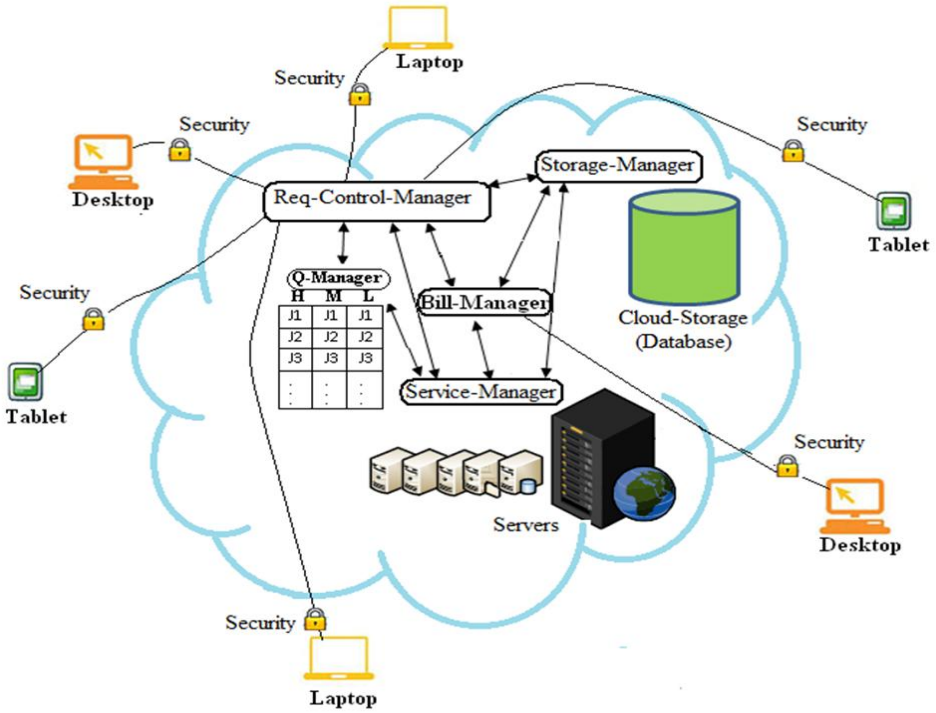

Figure 1: Cloud architecture to support Priority and Admission control based service scheduling system

Whenever the request from UL arrives at the cloud the *Req-Cntr-mgr* accepts the request after ensuring that the service-request can be served with the required QoS (Admission control). For this, the Req-Cntr-mgr interacts continuously with *Ser-mgr* regarding resource availability; while the *Bill-mgr* decides the charges for the finished job. The final billing charge of the finished job is fixed by the *Bill-mgr*, based on the actual time taken for the service by hiring the required resources. Thus, it ensures that there is no overloading of resources whereby many service requests cannot be fulfilled successfully due to limited resources available. The main role of *Ser-mgr* is to keep track of the availability of processors (Virtual Machines (VM)), assigning the available required resources to the service-request and initiating the servicing of the service-request on the allocated Virtual Machines. The Virtual Machines execute the service- request on physical machines. It is observed that performance monitoring of any application in cloud computing is always complex, different and challenging.

Figure 1 shows the architecture of the system model supporting Priority, subscription-category (SUB_{CAT}) and SLA based service scheduling in Clouds and Data Centers. There are basically five main components involved in this design:

- *Users:* Users can submit their requests for servicing the jobs from anywhere in the world to the Cloud through secured connection.
- *Request-Control-Manager (Req-Cntr-mgr)*: The *Req-Cntr-mgr* behaves as an interface between the Cloud service provider and external users. In order to schedule the services, it requires the interaction with the other entities to support priority, Admission control and subscription-category based service management. When the service-request is first submitted, the *Req-Cntr-mgr* checks whether the service-request can be admitted to the cloud or not by computing tolerable delay for that request using the equation (1)

---

if ($delay_i^T \geq ((ST_{est}^T / C) + ST_{est}^i)$)
  Admit the service-request
else
  reject the service-request

---

the *Req-Cntr-mgr* assigns the priority to each user request based on the $delay_i^T$ and SB_{CAT} to which user belongs.

Every user has to subscribe themselves to the cloud before accessing the cloud for submission of *service request*. During this subscription phase SLA is signed between the cloud service provider and the user. The *Req-Cntr-mgr* analyzes the submitted request for QoS requirements based on the priority and adds the request to one of the 3 queues High priority queue (Q-H), Medium priority queue (Q-M), low priority queue (Q-L) based on the priority computed using equation (3), SUB_{CAT} and on the availability of the resources. These *ser-req*s will be taken from these queues and serviced based on the availability of the resources required.

Before deciding whether to accept or reject the request *Req-Cntr-mgr* communicates with the *Ser-mgr*. Thus, it ensures that

there is no overloading of resources whereby many service requests cannot be fulfilled successfully due to limited resources available. It also needs the latest status information regarding resource availability (from *Ser-mgr*) in order to make resource allocation decisions effectively. Then, it assigns requests to VMs.

- *Queuing-manager (Q-mgr)* : is responsible for keeping all the *ser-reqs* in an appropriate queue based on the priority decided (computed by Assign-Priority module) by *Req-Cntr-mgr*. It releases the *ser-req*s for servicing according to the instruction given by *Req-Cntr-mgr* with *ser-mgr*.

(Pr^i) = *Assign-priority*$(delay_i^T , ST_{est}^T , SUB_{CAT})$

If $(Pr^i == H)$

Q_H = *ser-req*^{i}

If $(Pr^i == M)$

Q_M = *ser-req*^{i}

If $((Pr^i == L)$

Q_L = *ser-req*^{i}

- *Storage-Manager (Sto-mgr): Sto*-mgr stores and retrieves the data required for processing of the jobs.
- Billing-Manager (Bill-mgr): The Billing-mgr decides (computed by Amt-to-be-paid module) how the completed serviced requests should be charged. For instance, service-request can be charged based on the total time taken to complete the service $H\mu$ and billing rates (fixed/changing).

4.2 Algorithm

[Nomenclature :
Req_i - Service –request from user i
$H\mu$ - Total time (mins) served
SB_{CAT} - Subscription category
$delay_i^T$ - Tolerable delay for the request i
]

When a request for the service from the user i *ser-req*^{i} arrives at the Cloud (Service provider)

Req-control-mgr checks for the admission by computing the maximum tolerable delay and required service time for that *ser-req*^{i}

if $(delay_i^T \geq ((ST_{est}^T / C) + ST_{est}^i))$

Admit the *service-request*

else

reject the *service-request*

Req-control-mgr assigns the priority by calling *Assign-priority* module.

(Pr^i) = *Assign-priority*$(delay_i^T , ST_{est}^T , Td_{max})$

If $(Pr^i == H)$

Q_H = *ser-req*^{i}

If $(Pr^i == M)$

Q_M = *ser-req*^{i}

If $((Pr^i == L)$

Q_L = *ser-req*^{i}

Based on priority and SUB_{CAT} of the *ser-req*^{i} service scheduling happens as follows

While $(Q_H \neq NULL)$

{For each *service-request* in Q_H

Req-ser-mgr Checks for the resource availability by communicating with the *Ser-mgr*

if (required resources are available)

{ Schedule the *ser-req*^{i} for the service and initiate the servicing of *ser-req*^{i}

Once the μ_i is completed

Billing is done by *Bill-mgr* to charge for the service provided to *ser-req*^{i}

Amt-to-be-paid(Hμ) }

else

dynamically reallocate the resources based on $delay_i^T$ and ST_{est}^i and SB_{CAT} }

While $(Q_M \neq NULL)$

{For each *service-request* in Q_M

Req-ser-mgr Checks for the resource availability by communicating with the *Ser-mgr*

if (required resources are available)

{ Schedule the *ser-req*^{i} for the service and initiate the servicing of *ser-req*^{i}

Once the μ_i is completed

Billing is done by *Billing-mgr* to charge for the service provided to *ser-req*^{i}

Amt-to-be-paid(Hμ) }

else

dynamically reallocate the resources based on $delay_i^T$

and ST_{est}^i and SB_{CAT}

}

While ($Q_L \neq NULL$)

{For each *service-request* in Q_L

Req-ser-mgr Checks for the resource availability by communicating with the *Ser-mgr*

if (required resources are available)

{ Schedule the *ser-req*^{i} for the service and initiate the servicing of *ser-req*^{i}

Once the μ_i is completed

Billing is done by *Billing-mgr* to charge for the service provided to *ser-req*^{i}

Amt-to-be-paid(Hμ) }

else

dynamically reallocate the resources based on $delay_i^T$

and ST_{est}^i and SB_{CAT} }

Priority assignment Module

char Assign-Priority($delay_i^T$, ST_{est}^T ,)

{ If (($delay_i^T$ = ((ST_{est}^T /C) + ST_{est}^i)) and

SB_{CAT} ==1or 2 or 3)

Rethrn(H)

If (($delay_i^T$ > ((ST_{est}^T /C) + ST_{est}^i) by T_1))

{ If and SB_{CAT} ==1)

Rethrn(H)

If and SB_{CAT} ==2)

Rethrn(M)

If and SB_{CAT} ==3)

Rethrn(L) }

If (($delay_i^T$ > ((ST_{est}^T /C) + ST_{est}^i) by T_2))

{ If and SB_{CAT} ==1)

Place the service-request in Q_H or in Q_M

If and SB_{CAT} ==2)

Place the service-request in Q_M or in Q_L

If and SB_{CAT} ==3)

Place the service-request in Q_L }

}

Billing Module

Int Amt-tobe-paid (H_μ)

{

$\omega_{Reqi} = H_\mu * \partial$ /min

Return (ω_{Reqi})

}

5. SIMULATION MODEL AND PERFORMANCE EVALUATION

In our simulation model, we have a single cloud with group of 4 servers. There are 3 queues namely high priority queue-Q_H, medium priority queue-Q_M and low priority queue-Q_L with queue size of 15 each. We assume that, there is sufficient bandwidth in the cloud network. Simulation has been conducted for 5 hours. The table 2 lists the performance parameters considered for the simulation.

Table 2:. Parameters for the Simulation model

Parameter	Definition
$\|Q_H\|$	15
$\|Q_M\|$	15
$\|Q_L\|$	15
T_1	25 mins
T_2	60 mins
λ	46 request/hr
λ_e	30 to 45/hr
C	4
Unit time	Minutes

Service Completion Rate with QoS Vs Priority: In comparison with the traditional service scheduling technique(TSC) with out considering any priority and admission control [TSC-(AC+P)], Figure 2 shows service completion rate for the service-requests with our proposed priority based scheduling policy with admission control [PSC+AC]. During the admission of the service-request itself this algorithm checks whether the guaranteed quality service can be provided by

computing delay $_i^T$ and ST_{est}^i. Hence almost 99% of the service-requests have been provided with guaranteed high quality service. Since ST_{est}^i is the estimated service time required to complete the service-request, as λ and λ_e increases ≤1% of the requests could not finish their service within the specified deadline as shown in the figure 2. Whereas the traditional service scheduling policy in which, the priority and admission control policy are not considered accepts all the requests. Hence it is not able to complete all the requests with specified QoS as shown in figures 2 and 3. Thus the overall performance of the cloud throughput has been increased in the proposed priority based scheduling policy with admission control technique.

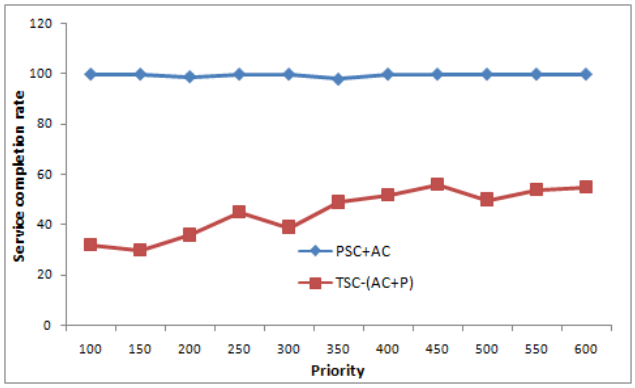

Figure 2: Service Completion Rate Vs Priority

Impact of Priority and admission control based scheduling policy on an average time a user request spends in the Queue: Keeping all the ser-reqs in the appropriate queues the proposed algorithm ensures the guaranteed quality service to every user request. As [PSP+AC] gives higher precedence to high priority service-request, the average time service-request waits in the queue also decreases as the priority increases as shown in the figure 3.

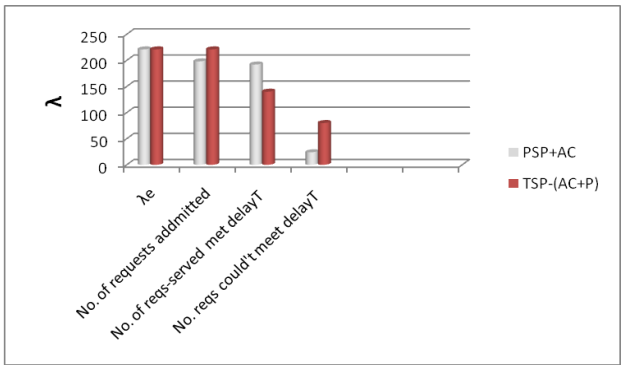

Figure 3: No. of reqs met delay^T Vs No. of reqs violated delay^T

6. CONCLUSION

In this paper an efficient Priority and admission control based service scheduling policy[PSP+AC] and an optimization model are proposed. This policy with the proposed cloud architecture has achieved very high (99%) service completion rate with guaranteed QoS over the traditional scheduling policy which does not consider the priority and admission control techniques[TSP-(AC+P)]. As this policy provides the highest precedence for highly paid user service-requests, overall servicing cost for the cloud also increases. In our future work, we are planning to extend this model to hire resources from other clouds and provision of security to improve the performance of the cloud system.

7. REFERENCES

[1] Cloud computing - Wikipedia, the free encyclopedia.htm

[2] K.Mukherjee, G.Sahoo, "Development of Mathematical Model for Market-Oriented Cloud Computing", International Journal of Computer Applications (0975 – 8887), Volume 9– No.11, November 2010.

[3] Rajkumar Buyya, Chee Shin Yeo and Srikumar Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", The 10th IEEE International Conference on High Performance Computing and Communications, IEEE Computer Society, 2008, pages 5-13.

[4] P. Mell and T. Grance, "Draft NIST working definition of cloudcomputing," Referenced on June. 3rd, 2009 Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, 2009.

[5] Liu Peng. Cloud computing principle. Web Page http://www.chinacloud.cn/show.aspx?id=1929&cid=12.

[6] Chen Ming1, Li Mengkun1,Cai Fuqin1 "A model of scheduling optimizing for cloud computing resource sevices based on", 2010 IEEE International conference on Granular Computing. DOI 10-1109/GiC 2010.180

[7] A Sachin V. Solanki, B. Minal Gour and C. A.R. Mahajan, "An overview of Different Job Scheduling Heuristics Strategies for Cloud Computing Environment", www.icett.com.

[8] R. Buyya, M.M. Murshed, D. Abramson, and S. Venugopal. Scheduling parametersweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm. Software Practice and Experience, 35(5):491{512, 2005.

[9] S. Venugopal, X. Chu, and R. Buyya. A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol. In *Proceedings of the 16th International Workshop on Quality of Service (IWQoS 2008)*, Twente, The Netherlands, June 2008.

[10] Qiang Li, Yike Guo. "Optimization of Resource Scheduling in Cloud Computing", 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 978-0-7695-4324-6/10© IEEE, DOI 10.1109/SYNASC.2010.8.