From Concept to Algorithmic Implementation: Optimized Sharing of Resources in Cloud Computing Environment

P. K. Suri
Dean, Research and Development; Chairman
HCTM Technical Campus, Kaithal, Haryana, India

Himanshi Goyal
HCTM Technical Campus

ABSTRACT

Cloud computing environment is referred as a collection of services which are delivered via the Internet. It depends upon sharing of resources to maximize the utilization of shared resources, and to achieve consistency and economies of scale. Resource management is very important for every system. Performance, functionality and cost are the three basic factors that are affected by resource management for system evaluation. Cloud resource management means to allocate and schedule computing resources. In this paper, various resource allocation and scheduling strategies are considered that helps in achieving high resource utilization and users demands. Various resource allocation strategies that are discussed in this paper are based on various parameters such as: location, time, topology, applications, hardware, priority, QoS etc. to meet the needs of cloud application. Similarly, scheduling strategies are based on parameters: cost, time, location, Qos, priority, load-balancing etc. to achieve high performance computing and best system throughput.

General Terms

Resource Scheduling, Simulator

Keywords

Cloud Computing, Resource Management, Resource Allocation Strategies, Scheduling Strategies

1. INTRODUCTION

As the name suggests, cloud computing allocates resources of local servers or personal devices, enabling applications to be handled via a resource cloud. Cloud computing has been lifeline of present day systems that rent computing resources on-demand, billing is done on pay-as-you-go basis, and enable geographically segmented users to work on the same physical infrastructure. Thus, cloud computing acts as a mirage of infinite computing resources to cloud users so that the availability can be increased or decreased as per the resource consumption rate.

1.1 History

Centralized Computing: At the onset of the '60s and '70s, a centralized computing model was considered to be the best way of resource sharing. It consisted of supercomputers strategically and remotely located in an internal data center. These supercomputers, with all the software, network and storage devices etc. were not only expensive but under-utilized, costing millions of dollars and hours of wasted manpower and energy. Revolutionized phase of the 1980s brought demand for increasingly more powerful and less expensive microprocessors. The way for low costs and simplicity was led by personnel computers at that time.

Distributed Computing [1]:

- Peer to peer network: A peer-to-peer (P2P) network is defined as distributed network architecture with interconnected nodes ("peers"). The idea of peer-to-peer sprang up in 1960s, with creation of ARPANET network to share files within US research facilities. Despite such success, the first peer-to-peer network was introduced after almost 50 years in 1999.
- Cluster Computing: Cluster computing computes of basic computing in which several nodes in the same physical location, directly connected with very high speed connections (LAN) to operate as a single device.
- Grid and Utility Computing played its part in the 1990s as the Internet and the World Wide Web found their way into the general computing world moving from obsolete centralized models to Internet-based computing. Term utility computing finds its roots from the real world where service providers maintain basic utility services, ranging from electrical power, gas, and water to consumers. Consumers pay in return to service providers based on basic usage. All grid/cloud platforms enact as utility service providers. On the better part, cloud computing offers a broader concept than utility computing. Grid computing can be largely related to large-scale cluster computing. Grid computing systems are more heterogeneous, loosely coupled and geographically dispersed as compared to cluster computing systems. For processing a single task, grid computing uses the capabilities of different computing units. A task is divided into sub-tasks and then assigned to different machines and on completion they are sent back to the machine which is responsible for all the tasks.

Application Service Providers (ASP) took the game to another level in the late 1990's creating the first of many Internet-enabled applications. It was termed as "on-demand software." ASP is generally used to provide computer-based services over a network. It provides access to a particular application program by indulging a standard protocol.

Software as a Service (SaaS) is a software delivery model and is often used in place of ASP. Unlike ASP, SaaS vendors typically develop and manage their own software. In SaaS, users pay for software's as per usage, not as per a license.

Cloud computing today acts as a mirage of infinite computing resources to cloud users so that the availability can be increased or decreased as per the resource consumption rate. In cloud computing data is stored in the data center of internet but not locally, and with the use of application programming interface (API), users can access the stored data. Cloud computing has been lifeline of present day systems that rent computing resources on-demand, billing is done on pay-as-you-go basis, and enable geographically segmented users to work on the same physical infrastructure.

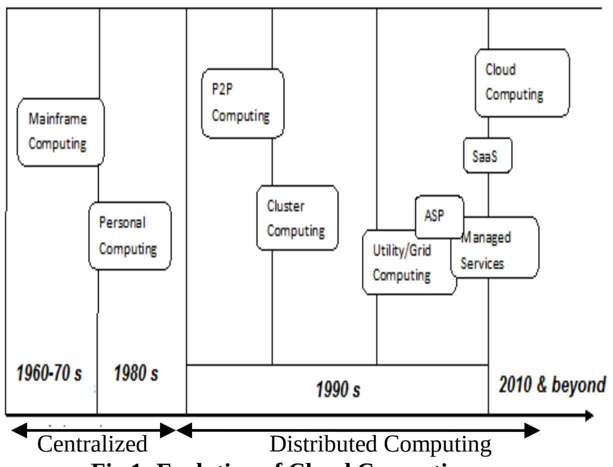

Fig 1: Evolution of Cloud Computing

1.2 Types of Clouds

Cloud computing system is classified according to the following types and each is having significant characteristics [2]:

- Private Cloud: Private cloud is a cloud model which suits better to the organization that wants to operate solely, and the services are provided to specified client only.
- Public Cloud: It is a type of model that promotes shared environment and services are provided to the clients across the globe.
- Community cloud: It is a type of model which is shared between several infrastructures, having a common concern.
- Hybrid Cloud: It is a composition of two or more form of clouds that remain distinct entities but are bound together.

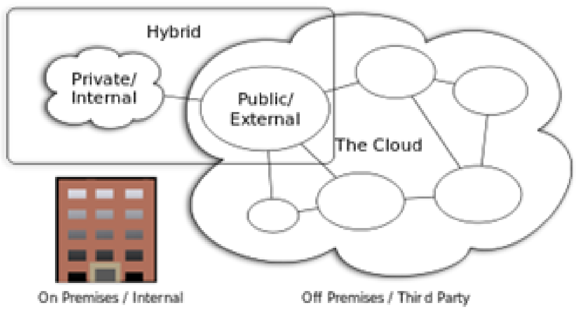

Fig 2: Cloud Computing Types

2. RESOURCE MANAGMENT

Resource management is very important for every system. Performance, functionality and cost are the three basic factors that are affected by resource management for system evaluation. If resource management is not efficient then it directly affects all the three factors i.e. performance, functionality, and cost in a negative way. Cloud computing is a complex system and acts as a mirage of infinite computing resources to cloud users so that the availability can be increased or decreased as per the resource consumption rate. Cloud computing rents computing resources on-demand and billing is done on pay-as-you-go basis. Cloud resource management strategies are associated with three models (cloud delivery models) named as, Platform as a Service, Infrastructure as a Service, and Software as a Service. All the three models are different from one another. Cloud providers and users both play an important role in cloud computing environment. Cloud providers are the one who hold computing resources in their large data centers and users have applications that require resources from providers to run. Providers provide resources on rent to users only on demand and they pay as per usage.

Cloud resource management is concerned with two main aspects i.e. resource allocation and job scheduling. Resource allocation means to allocate resources to the applications so as to satisfy SLA requirements. In cloud environment, resources are allocated only on demand i.e. when a user makes a request. An application may consist of multiple jobs to which resources are allocated. Hence after allocating the resources, an efficient scheduling strategy is required to schedule these jobs to allocated resources so as to provide high resource utilization and to achieve best system throughput.

2.1 Resource Allocation

Resource Allocation (RA) is the process of allocating resources which are available to the needed applications. Resources are allocated only on demand, providers rent computing resources on pay-as-you-go basis. An efficient and optimized allocation strategy is required to allocate scarce resources and to utilize them within the limit of cloud environment so as to meet QoS requirements of cloud applications. The type and amount of requested resources is decided by the user and is provided in the request made. Then providers place the requested resources, according to their availability onto nodes in data centers. The type and amount of requested resource should be sufficient so as to match the workload characteristics and to meet the constraints respectively. An optimal resource allocation strategy should give a wide berth to the following measures [3]:

- Resource contention
- Scarcity of resources
- Resource fragmentation
- Over-provisioning
- Under-provisioning of resources

2.2 Job Scheduling

An application may consist of multiple jobs to which resources are allocated. Once the resources (virtual machines) are allocated to the user, procedure is required to schedule tasks or jobs on the resources to achieve maximum profit and efficient resource utilization. In cloud computing resources are allocated to the user on pay-as-per-use basis, hence job scheduling is an important task in cloud environment. Job scheduling strategy is responsible for scheduling jobs on allocated resources so that resource utilization effectively increases. There should be efficient and optimized strategies for job scheduling so as to meet QoS requirements for users. The objectives of job scheduling strategy are: first, to maximize the profit second, to meet user's QoS requirements third, efficient resource utilization. Forth, high performance computing and fifth, increase in throughput.

Job scheduling algorithms can be classified into two main categories:

- Batch mode heuristic scheduling algorithms, and
- Online mode heuristic scheduling algorithms.

In batch mode algorithm, jobs are not scheduled immediately after arriving in a system. They first entered the queue and then scheduled after a fixed interval of time (say 1 hr.). First-come-first-serve, max-min, min-min etc. are the examples of batch mode heuristic algorithm. But in online mode algorithm, jobs are scheduled immediately even after they arrive in a system. Most fit task scheduling is an example of online mode algorithm. For cloud environment, on-line mode heuristic algorithms are more appropriate than batch mode algorithms because it is a heterogeneous system and speed of processor's varies quickly. The various job scheduling algorithms are the following [4]:

- First-Come-First-Serve Algorithm: A type of algorithm in which job which comes first will be served first.
- Round Robin algorithm: In round-robin scheduling, jobs are scheduled on first-come-first-serve basis but each is having a limited amount of CPU time to process and if it is not completed before that time period then CPU is pre-empted and given to the next job in a queue.
- Min–Min algorithm: A type of algorithm in which short jobs execute in parallel and then followed by long jobs.
- Max – Min algorithm: It is very much similar to min-min algorithm but here short jobs execute in parallel with long jobs.
- Most fit task scheduling algorithm: Job that fits best to the requirements will be served first.

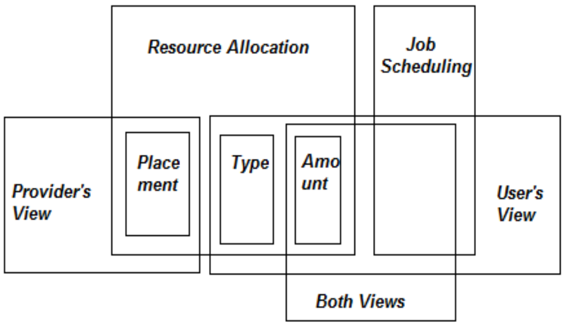

Fig 3: Elements of Resource Allocation and Job Scheduling

3. LITERATURE SURVEY

Cloud resource management means to allocate and schedule computing resources. The main goal of cloud resource management is to provide high resource utilization, to achieve best system throughput, and to fulfil user demands. Efficient resource management directly influences the efficiency of the whole system. In this section we are focusing on various resource allocation and scheduling methods that are already present in the cloud computing environment.

3.1 Resource Allocation Strategies

3.1.1 Topology Aware Resource Allocation (TARA)

In [5], architecture for optimized resource allocation is proposed that is based on IaaS i.e. Infrastructure-as-a-Service based cloud systems. Current IaaS systems allocate resources independent of the application requirements because they are unaware of the user's needs and hence, can affect the performance for distributed data-intensive applications. To address this problem, an architecture based on what-if methodology is proposed to help IaaS system in allocation decision. The two main aspects of proposed architecture are:

- Prediction Engine: Estimate the performance of resource allocation by using light weight simulator.
- Genetic algorithm: To find an optimized solution.

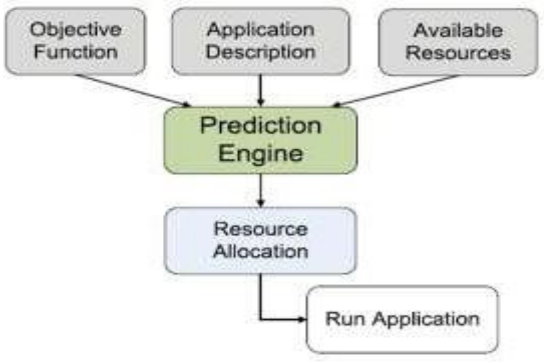

Fig 4: Basic Architecture of TARA[5]

3.1.2 Virtual-Machine Based Allocation

In [6], a mechanism is designed for a non-cooperative cloud environment to allocate virtualized resources among selfish virtual machines. By non-cooperative means, virtual machines care only about their own benefits. Stochastic approximation approach has been considered in the proposed model. The proposed stochastic mechanism and management approaches enforced to allocate the virtual resources efficiently. In [6], QoS performance is also analyzed under various virtual resource allocations. The proposed method is very complex and it is not implemented in a practical virtualization cloud system with real workload.

3.1.3 Linear Scheduling Technique

In [7], a scheduling algorithm is proposed to schedule tasks and resources. Hence it is named as Linear Scheduling for Tasks and Resources (LSTR). LSTR scheduling mainly focuses on two main aspects to allocate the resources: first, to maximize system throughput and second, efficient resource utilization. The basic idea of linear scheduling strategy is to distribute the resources among those requestors that will maximize the selected QoS parameters. Linear scheduling strategy does not allocate resources as they arrive. The initial response is made only after collecting the resource for a fixed time period (say 1 day or 1 hr) but not allocating the resource as they arrive.

Algorithm:

Step1: Find all requests that arrive, if any, during a pre-determined time period.
Step 2: Initialize the threshold value.
Step 3: For every request, if it is less than threshold value then add to array A else add to array B.
Step 4: Sorts array A[RQ_i] & B[RQ_i].
Step 5: For every request in array B, allocate resources and updates the available resources and threshold value.
Step 6: Then for every request in array A, allocate resources and updates the available resources and threshold value.

3.1.4 Most-fit Processor Strategy

In [8], a new approach for resource allocation is proposed named as, most-fit processor policy. The proposed policy controls resource fragmentation in multi-cluster environment. In the most-fit policy a job is allocated to the cluster that produces a leftover processor distribution. It requires a complex searching process to determine the target cluster, and may also involve simulated activities. It is assumed that the clusters are homogeneous and geographically distributed and the number of processors in each cluster is binary compatible. The proposed policy presented that job migration is required to be done at the time of load sharing activities. It is also showed that the most-fit policy has higher time complexities and negligible time overheads as compared to the system long time operation.

3.1.5 Nephele Framework

In [9], a new project called Nephele is presented. Nephele is the first data processing framework for both, task scheduling and execution. It exploits the dynamic resource allocation offered by IaaS clouds. A job can have number of tasks that can be assigned to different types of virtual machines at the time of job execution, particular task(s). Nephele's architecture follows a classic master-worker pattern.

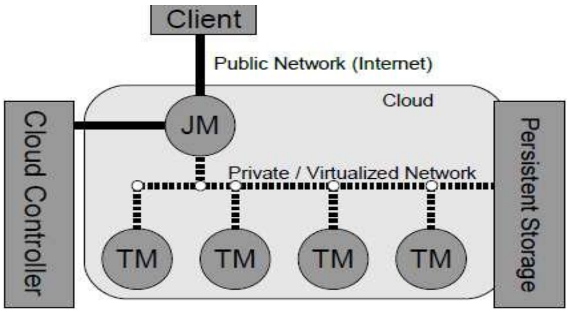

Fig 5: Architecture of Nephele Framework[9]

Working: In Nephele, a user must start a virtual machine before submitting a job, which runs the Job Manager (JM). He is the one who receives client's jobs, schedule them, and coordinates their execution. In Nephele, a job is represented by a Directed acyclic graph i.e. (DAG). Each vertex in the graph represents a task of the processing job, and edges represent the communication flow between these tasks. After receiving a valid job graph from the user, Job Manager transforms it into an Execution Graph. An Execution Graph is a primary data structure in Nephele, used for scheduling and monitoring the execution of a job. Job Manager communicates with the Cloud Controller (CC) which is provided by cloud operator to controls VM instantiation. According to the current job execution phase, Job Manager can allocate or deallocate virtual machines with the help of Cloud Controller. The actual execution of tasks is carried out by a set of instances, and each instance runs a so-called Task Manager (TM). A Task Manager is the one who receives tasks from the Job Manager and executes them. After then, it informs the Job Manager about the tasks completion or possible errors.

3.1.6 Gossip Strategy

In [10], a gossip-based protocol for resource allocation is proposed in large-scale cloud environments. The system is modeled as a dynamic set of nodes. Each node represents the machines of cloud computing environment. Each node has a specific CPU capacity and memory capacity. The basic idea of this protocol is to allocate cloud resources to a set of applications that have time-dependent memory demands. The simulation results show that optimal allocation is produced only when memory demand is smaller than the available memory in the cloud and the number of applications and the number of machines does not affect the quality of allocation. The protocol dynamically maximizes a global cloud utility function. But additional functionalities are required to make resource allocation scheme robust to machine failure.

3.1.7 Priority based Allocation

In [11], a new approach is proposed i.e. based on the priority of various parameters and hence named as priority algorithm. The motive of the proposed algorithm is to minimize the wastage and to provide maximum profit. This priority algorithm decides priority among different user request on the basis of many parameters like cost of resource, task type, number of processors needed to run the job or task, time needed to access etc. In the proposed model, first of all client sends a job request to the cloud server and then server, the service provider in cloud computing environment will run the task submitted by client. The priority among the different user request is decided by the cloud administrator and hence plays an important role to make efficient resource allocation.

Algorithm:

```
Step 1: Insert all values of client request i.e. time, price etc. into the linked list.
Step 2: Then the priority value is assigned to each request and its tasks based on the predefined conditions. Priorities that are assigned in the proposed algorithm are:
    • Node priority, and
    • Time priority.
Step 3: For each client‘s request check,
If (input value is within the threshold limit), then calculate sum for each request by adding priority values and other parameters.
Step 4: Sort the value that is calculated in above step, and request with least value is ready to execute.
Stop
```

3.1.8 Auction Based Allocation

In [12], resource allocation is addressed by auction mechanism. The proposed mechanism is based on basic idea of sealed-bid auction. Like sealed-bid auction, the cloud service provider collects bids from all the users and then on the basis of bid, determines the price. As per according to the bid-auction, the resource is distributed to the first k^{th} highest bidders under the price of the $(k+1)^{th}$ highest bid. It is a simplified approach because allocation is done only on the basis of bid provided by user. The proposed system reduces the resource problem into ordering problem, hence simplifies the cloud service provider decision rule and therefore the allocation rule. But due to its truth telling property under constraints, this mechanism does not ensure profit maximization.

In [13], allocation is done by using market based resource allocation strategy named as, RSA-M strategy. This market based allocation strategy uses the concept of equilibrium theory. The aim of the proposed resource allocation strategy is to maximize the profits of both the customer agent and the resource agent in a large data center by using the equilibrium theory i.e. to maintain balance between the demand and supply in the market. RSA-M strategy determines the number of fractions used by one VM and can be adjusted dynamically according to the varied requirement for resources. The resource type which is delegated by resource agent is used to publish the resource's price, and the one delegated by the customer agent participates to obtain the maximum benefit for the consumer. Market Economy Mechanism is responsible to maintain balance between the resource supply and demand in the market system.

3.1.9 Application Based Allocation

Application based means allocation is done on the basis of nature of application. In [14], virtual infrastructure allocation strategy is designed where allocation is done on the basis of the workflow representation of the application. To produce an estimation of executed schedule, the application logic for work flow based applications can be interpreted and exploited. And, for each run of the application, helps the user to estimate the exact amount of consumed resources. In [14], four strategies are designed to allocate resources and schedule computing: Naive, FIFO, Optimized and services group optimization.

3.1.10 Queueing Model Based Strategy

In [15], a cloud computing model is proposed for allocation of resources to the jobs that enter into the cloud by using queuing models. The arrival of jobs follows non-homogeneous Poisson process. The proposed model is analyzed by using various performance factors such as Utilization, Throughput, Mean number of job requests in the cloud, and Mean Delay in the cloud. The arrival of job requests follows non-homogeneous Poisson process stored in a buffer, scheduled for resource allocation using queuing models. The scheduling is carried out using request dependent strategy. In request dependent strategy the resource (virtual machines (VMs)) allocation rate linearly depends upon the number of request jobs in the buffer depending on the buffer content.

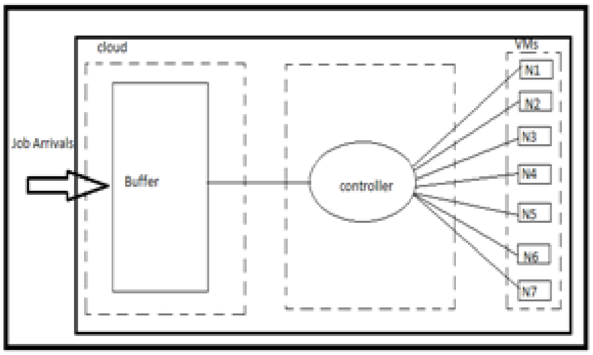

Fig 6: Job requests in buffer to cloud system[15]

Following are the characteristics of the system (during a small interval of time h) in [15]:

- The arrival of the jobs that follows non-homogeneous poison process is statistically independent.
- [λh+o(h)] is the probability that only one job arrives.
- [n μ h + o(h)] is the probability that only one job is serviced through the cloud, when there are n jobs in the buffer
- [o(h)] is the probability of other than above jobs.
- [1- λh- n μ h+o(h)] is the probability that no job arrives in the buffer and no job servicing occurs, when there are n jobs in the buffer.

3.1.11 Pre-Copy Approach

In [16], it is suggested that migration of the operating system instances across distinct physical hosts offers a separation between hardware and software and is a great tool for the administrator of data centers and clusters. It provides many features like load balancing, low level system maintenance and fault management. Clark et al. on the basis of this idea proposed a new approach: "pre-copy approach". In this approach memory pages are repeatedly copied to the destination host. But the most important point that is considered here is, all these things are done without ever stopping the execution of the system and to make sure that a consistent snapshot is transferred, page level protection hardware is provided. In the proposed system, rate-adaptive algorithm is used for controlling the traffic of different running services. And in last, it pause the virtual machine and copies any leftover pages to the destination and afterwards resumes the execution there.

3.1.12 Negotiation Strategy

In [17], a new approach is presented where providers and consumers automatically negotiate resource leasing contracts. A negotiation mechanism that is proposed is distributed in nature because resource demand and supply can be uncertain and dynamic. Agents negotiate not only over contract price but also over a decommitment penalty. By decommitment penalty means, agents can decommit from contracts at a penalty (cost). In negotiation mechanism, agents make contracts according to which resources are provided to users for a fixed time interval. Feature of decommitment penalty is introduced to accommodate the highly dynamic nature of cloud computing platforms.

3.1.13 Location Aware Dynamic Allocation

In [18], a dynamic resource allocation model on cloud computing environments is proposed that depends upon two factors: first, utilization level of Physical machines in data centers and second, location of user and data center. In [18] dynamic resource utilization management architecture is proposed to perform location-aware Virtual machine placement by permitting provider to place a new VM in an appropriate PM for better performance. The system prevents performance degradation of the data center by guarantying the maximized utilization level,

Proposed Model: The system uses utility function, to find out which Physical machine is appropriate for a new Virtual machine or migration as follows:

$$U_m = (\alpha * u^u) + (\beta * u^t) + (\gamma * u^g)$$
$$(0<=u_m<=1)$$

Utilization level (U^u):

For ($w_c <= w_e < w_t$):
$$U_{min} + 1-[(w_e – w_c) / (w_t – w_c)]$$

For ($w_t < = w_e$): U_{min}

Response Time utility (U^t):

For ($t_c < = t_e < t_t$):
$$U_{min} + 1-[(t_e – t_c) / (t_{SLA} – t_c)]$$

For ($t_{SLA} <= t_e$): U_{min}

Geographical Distance Utility:
$$U^g = 1- [dif_m / dif_{range}]$$

3.1.14 Just-In-Time Resource Allocation

In [19], the cost based workload provisioning and "just-in- time resource allocation" is illustrated. In this type of method, optimization is dispensed by taking into concern the step-down of the cost incurred to the application. The cost can be a mixture of various factors like leasing cost of resources, cost associated with the changes to the configuration, and the cost of SLA violations. A perfect solution for efficient resource utilization is to set a time interval and resources are changed within the limit of this interval unceasingly in accordance with the modification in load. In JITRA, there are three components of the cost function that calls for the penalty:

- cost for violation of SLA bounds,
- cost of leasing a machine, and
- cost of reconfiguring the application.

But recursive data structures are required to be implemented for the look-ahead implementation of the time interval for each task.

3.1.15 Utility Function Based Allocation

In [20], resource allocation is based on response time for multitier cloud computing systems (heterogeneous servers) as a measure of utility function by considering communication resources, memory, and CPU. Servers are characterized on the basis of following characteristics: memory usage, capacity of processing powers, and communication bandwidth. For each tier, requests from the users are distributed among available servers and each available server is assigned to exactly one of the application. We can say that, servers can serve only those requests that are assigned to them. In the proposed strategy, queuing theory is used to dispatch each client to the server. This system meets the requirement of SLA (Service Level Agreement) such as response time and utility function based on its response time.

But in [21], specific resource allocation (CPU, RAM) is done by considering the utility function as a measure of application satisfaction. Local Decision Module (LDM) is used to compute utility function by taking current work load of the system. Then the module interacts with Global Decision Module (GDM). GDM is the decision making entity within the autonomic control loop. The proposed mechanism is based on a two-tier architecture and resource arbitration process that can be controlled by using different factors such as application's weight.

3.1.16 Adaptive Resource Allocation

In [22], an adaptive approach is presented in runtime to allocate resources in order to satisfy one of the most important QoS requirements, and throughput of multiple workflows in Service Based Systems. A system that adopts service-oriented architecture (SOA) is known as service-based systems (SBS) eg: grid computing, web services, cloud computing etc. In SOA, services are categorized as: atomic and composite. Atomic services are those that serve only one type of service-request and cannot be decomposed into smaller services. All other are considered as composite services.

In [22], first of all a model is developed for an atomic service named as Resource Allocation Throughput (RAT) model, and then it is extended for the entire SBS to analyze the relationship between resource allocation and throughputs of the multiple workflows. Based on the RAT model, a linear programming problem is formulated to find the optimal resource allocation to serve the user's requests.

RAT model for atomic services: To an atomic service limited system resources are allocated. As request for any service arrive at the queue, the atomic service creates multiple threads, which utilize the system resource to process the request and send out the responses. To estimate the amount of resources, consumed by the atomic service for the processing of user's requests following five factors are considered:

Service-request rate (R): The average number of service-requests arrives at atomic service per second.

Critical resource: Resources of the server that will become a bottleneck.

Percentage of Allocated critical resource (A): It is provided by the server over the total available critical resource of the server.

Throughput of an atomic service (P): The average number of service responses per second. It is determined by using the values of R and A.

Throughput requirement of a workflow: The minimum number of service responses required by the users for a workflow per second.

A linear programming problem that is formulated to allocate critical resources of all servers in a SBS is represented as:

1). W, set of workflows in SBS
2). Sv, set of servers in SBS\
3). for each workflow, define priority and throughput as TH and Pr.
4). Objective Function:

$$\text{Max. } \Sigma_W (TH \times Pr)$$

5). Constraints:

- Throughput (TH) of w is less than equal to Service request rate (SR) of workflow (w).
- Throughput of w is greater than equal to Service request rate of w, only if throughput-requirement (TR) of w is less than equal to Service request rate of workflow (w).
- Cost (C) is less than equal to the percentage of available critical resource of service-request.

Then the Simplex algorithm [23] is used to solve the linear programming problem and the optimal throughput of each workflow is evaluated in SBS.

3.1.17 Hardware Dependent Allocation

In [24], a resource allocation strategy is proposed by categorizing the cluster in the system. Clusters are categorized on the basis of the data storage, number and type of computing and communication resources that they control. All of the computing resources are allocated within each server. Generalized Processor Sharing (GPS) is used to allocate different resources (except disk resource) in the servers and clusters. The disk resource is allocated based on the client's constant need. The proposed strategy performs distributed decision making by parallelizing the solution to reduce the decision time. Greedy algorithm is used to find the best initial solution. By changing resource allocation, the solution could be improved. But large changes in those parameters cannot be handled by the system, which are used for finding the solution.

In [25], on the basis of CPU consumption amount, an adaptive resource co-allocation approach is presented. The presented resource co-allocation is done in stepwise manner in three phases.

- First, the co-allocation scheme is determined by considering the CPU consumption amount for each physical machine (PM).
- Second, simulated annealing algorithm is used to determine whether to put applications on PM or not. The configuration solution can be altered by randomly changing one element.
- In the third step, the exact CPU share that is occupied by each VM occupies is determined and that can be optimized by using gradient climbing approach.

The proposed strategy does not consider the dynamic nature of resource request; it mainly focuses on CPU and memory resources for co-allocation.

3.2 Scheduling Strategies

3.2.1 Enhanced Max-min Task Scheduling

In [26], an Enhanced Max-min task scheduling algorithm is proposed. It is a modification of improved max-min algorithm. Improved task scheduling algorithm is proposed to improve the efficiency of max-min algorithm. In improved max-min algorithm, firstly largest task (task with maximum execution time) is assigned to the slowest resource (resource having minimum completion time). Then the scheduled task is removed from the set and all the timings (i.e. ready time of selected resource and total completion time) are updated and finally max-min algorithm is applied on remaining tasks. If the largest task is too large than other tasks in a set, then it may happen that tasks other than the largest task complete their execution before the completion of largest task by fastest resource. This leads to increase in make span because the largest task is executed by slower resource and load imbalance across resources. To solve this problem, enhanced max-min algorithm is proposed. In enhanced algorithm instead of scheduling largest task, task with average execution time is scheduled first to slowest resource, this helps in reducing the make span and balancing the load across resources.

Algorithm:

Step 1: Compute expected completion time for all tasks on all resources.

Step 2: Assign task with average (or nearest greater than the average) execution time to the slowest resource.

Step 3: Then remove schedules task from the task-set, and then update both the ready time of selected resource and completion time for all tasks on all resources.

Step 4: Now apply max-min algorithm to schedule remaining tasks:

- First, compute minimum completion time for all the tasks then among these minimum time values select the maximum value.
- Second, schedule that task on the resource on which it takes minimum time and remove from the task set.
- Third, update the ready time of that resource for all the other tasks and completion time for all other remaining tasks on all resources and delete the scheduled task from the task-set.

3.2.2 Priority based Job Scheduling

In [27], a priority based job scheduling algorithm for cloud computing environment is presented. This job scheduling algorithm is named as PJSC algorithm. Priority job scheduling algorithm considers priority at three levels: scheduling level, resources level, and job level. In priority job scheduling, every job that is required to schedule has a pre-determined priority and scheduling is done on the basis of that priority.

Algorithm:

Step 1: Suppose there are n jobs that are required to be scheduled on m resources, where (m<<n).

Step 2: Create a n*n comparison matrix of jobs for each resource, such a matrix is known as consistent comparison matrix. The matrix is created according to the priority of resources accessibilities.

Step 3: As there are m resources, m such matrices are created (one for each resource). Let the matrices are M_1, M_2… M_d and for i=j, the value of matrix is 1, otherwise:

$$m^{ij} = (1 / m^{ji}).$$

Step 4: Compute priority vector for each of the matrix, let W_1, W_2….. W_d are the priority vectors for matrix M_1, M_2… M_d respectively, the value of w can be calculated as:

W=Eigen value of Matrix*Corresponding Eigen vector.

Step 5: Make a matrix with priority vectors as:

$$\Delta= [W1, W2 ... Wd]$$

Step 6: Create a d*d consistent comparison matrix for resources to determine which resource has a higher priority than others on the basis of decision makers.

Step 7: Let R be the matrix for resource level, calculate priority vector for matrix R same as that of step 5. Let pr be the priority vector of matrix R.

Step 8: Calculate priority vector of scheduling jobs. Let PVS is the priority vector, computed as:

$$PVS= \Delta . pr$$

Step 9: Finally on the basis of PVS, a job with maximum priority value is scheduled to a suitable resource and then update the list of jobs.

3.2.3 Heterogeneous Earliest Finish Time algorithm

In [28], algorithm is presented for distributed environment on the basis of earliest finish time. Hence it is named as, heterogeneous earliest finish time (HEFT) algorithm. The two main aspects that is considered in the proposed algorithm are:

- priority, and
- execution time

Algorithm:

Step 1: Compute the average execution time for each task on each resource.

Step 2: Compute the average communication time between the resources of two tasks.

Step 3: Ordered the tasks in the workflow on the basis of a rank function.

Step 4: Assign higher priority to a task with higher rank value.

Step 5: Then schedule tasks in priorities, in the resource selection phase

Step 6: Finally tasks are assigned according to the earliest completion time.

3.2.4 Modified Genetic Algorithm

Genetic algorithm (GA) is a search heuristic that copies the procedure of natural evolution. In [29], a modified genetic algorithm (MGA) is presented. The algorithm is proposed by combining two existing scheduling algorithms such as, smallest cloudlet (job) to fastest processor and largest cloudlet to fastest processor. MGA schedule tasks according to the computing capacity of processing elements and computational complexity. It main goal is to minimizes the execution time and execution cost as well.

Algorithm:

Step 1: Generate an initial population of individuals by using the output of following techniques:

- Smallest Cloudlet (job) to Fastest Processor (SCFP),
- Longest Cloudlet (job) to Fastest Processor (LCFP), and
- 8 Random Schedules.

Step 2: Evaluate each individual.

Step 3: Repeat following steps until termination condition occur:

- Select individuals with minimum execution time.
- Crossover between individuals, crossover operator that is used in the proposed algorithm is two –point crossover.
- Mutate the resultants by using various operators such as: move, swap, rebalancing etc. but the one that is used here is swap operator.
- Evaluate modified individuals having relevant fitness.
- Generate a new population

3.2.5 Bees Life Algorithm

In [30], an efficient algorithm that is inspired by the bees' colony life is proposed. Hence it is known as Bees Life Algorithm (BLA). Two behaviors of bee's life are represented in the algorithm: one is reproduction and other is food source searching. The proposed algorithm aims at scheduling jobs to resources in minimum completion time.

Algorithm:

Step 1: Generate an initial population at random.

Step 2: Evaluate fitness of each individual. Fitness is specified as:

- fittest bee is queen
- fittest following bees are drones
- remaining bees are workers

Step 3: Repeat the following until terminate condition occur:

/* reproduction behavior */

Step 4: Select drones
Step 5: Crossover between individuals by using two-point crossover.
Step 6: Mutate the resultants by selecting a operator in which randomly selected task is replaced by another random task.
/* food foraging behavior */
Step 9: Search of food source by W workers in W regions.
Step 10: Recruit bees for each region and then select the fittest bee from each region
Step 11: Evaluate fitness of population

3.2.6 Improved Cost-Based Algorithm

In [31], an improved cost-based scheduling algorithm is proposed to schedule tasks in cloud computing environment. It is designed for the environment where resources have different computation costs and performance. The proposed algorithm divides tasks into three different criteria on the basis of resource processing capabilities. Changes in traditional cost based strategy implicates that no relation stands between the varied tasks leading to overhead resource cost and overhead application base in user friendly cloud computing environment. Due to job grouping, the proposed scheduling algorithm improves the computation/communication ratio.

3.2.7 Short Job Scheduling

In [32], an efficient scheduling algorithm is proposed for multiple clouds computing environment. The algorithm is named as, Short Job scheduling algorithm. The proposed algorithm uses middle-layer architecture to perform allocation in case of under load and overload conditions. That means the algorithm is able to handle load conditions. The concept of process migration is used to handle over load conditions. As the middle layer exists between the clouds and users, therefore the request from user will be first accepted by the middle layer and make the analysis of servers. The middle layer is responsible for three main tasks:

- First, scheduling the user requests,
- Second, monitor the servers for its capabilities and to perform the process allocation.
- Third, process migration in overload conditions

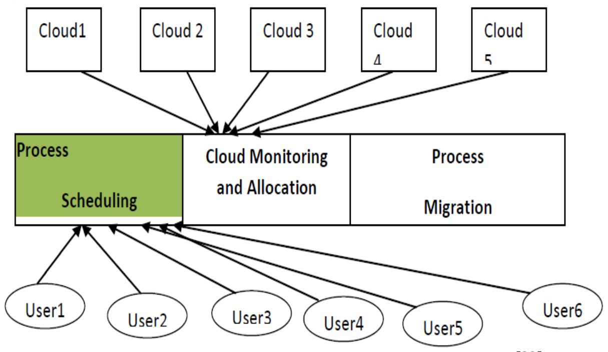

Fig 7: Proposed Three-Layer system[32]

Algorithm:
Step 1: Suppose there are M numbers of Clouds and each cloud is having L number of Virtual Machines associated with them, and N is the number of user's requests having some parameters i.e. process time, required memory etc.
Step 2: For each virtual machine, compute load and available memory.
Step 3: Assign the priority to each cloud.
Step 4: Arrange the requests according to one of its parameter, let it be required memory.
Step 5: For each user request, identify cloud and associate virtual machine for which available memory is greater than required memory.
Step 6: Now allocate the process to that particular cloud and virtual machine.
Step 7: For each request, compute free Time slot on priority cloud. Then record the turnaround time, process time, start time, and the deadline of the process.
Step 8: Now for each request, check migration is required or not. If finish time of process is greater than the deadline of that process then yes otherwise no.
Step 9: For migration, the process is migrate to the next high priority cloud, that having the free memory and the time slot

3.2.8 DRR Scheduling

In [33], an optimized algorithm is proposed that satisfies the user requirements like reliability, deadline etc. Hence it is named as, Deadline-Reliability-Resource-aware (DRR) scheduling algorithm. The proposed algorithm considers communication model and the realistic network topology. The theory analysis of the model demonstrates that, the algorithm can satisfy the reliability and deadline requirements of the user. In [33], it is shown that the proposed algorithm can complete the job under the deadline, if the deadline of DRR algorithm is set to the value less than that of the make span of the MaxRe algorithm.

3.2.9 Qos Based Scheduling

In [34], a QoS based predictive max-min, min-min switcher algorithm named as, $QPS_{MAX-MIN<>MIN-MIN}$ is presented for scheduling jobs. Before scheduling the next job, the proposed algorithm makes an appropriate selection on the basis of heuristic applied, among the QoS based min-min or QoS based max-min algorithm. Min–Min algorithm is a type of algorithm in which short jobs execute in parallel and then followed by long jobs but in max-min short jobs execute in parallel with long jobs. The proposed scheduling algorithm selects the best algorithm between QoS max-min and QoS min -min according to the length of jobs while making each scheduling decision.

Algorithm:
Step 1: Compute expected completion time for all tasks on all machines.
Step 2: For all jobs, compute minimum completion time and the machine on which it takes minimum time.
Step 3: Compute standard deviation of completion time for all unassigned jobs.
Step 4: Find a position in a sorted array (in increasing order of jobs completion time) of jobs where, computed standard deviation is less than the difference of two consecutive completion times.
Step 5: If either that position is less than equal to half of jobs or standard deviation is less than threshold value then go to step 6 else go to step 7.
Step 6: Assign first job of the sorted set to machine on which it takes minimum time and go to step 8.
Step 7: Compute no of jobs in a set (say n) then assign j_n job to the machine on which it takes minimum completion time.
Step 8: Update the ready time of corresponding machine and update the completion time of all jobs on all machines.

3.2.10 Generalized Priority Algorithm

In [35], a Generalized Priority algorithm is presented for efficient execution of task. In the proposed algorithm, user assigns priority to the jobs according to various parameters such as: bandwidth, size, scheduling policy, memory etc. In the proposed strategy, priority is assigned to both tasks and virtual machines on the basis of parameters size, bandwidth, processing speed etc. In the proposed strategy, tasks are prioritized according to their size and virtual machines are prioritized according to their MIPS value. That means, tasks having highest size has highest priority and virtual machine with highest MIPS has the highest rank. Thus, size and MIPS are the key factor for prioritizing tasks and virtual machines respectively.

Algorithm:

Step 1: According to computational power of host/physical server, create VM to different Data center and allocate cloudlet length. Computational power of a server can be defined in terms of speed, memory, cost etc.

Step 2: Maintain an index table of virtual machines using load balancer

Step 3: Assign highest MIPS of virtual machine to highest length of cloudlet.

Step 4: Sends request with specific id to virtual machine and update the available resource.

3.2.11 QoS Based Workflow Scheduling

In [36], QoS based strategy is proposed for those tasks that processed in a specific order according to their required service. This strategy schedule tasks on the basis of various QoS parameters such as: Reliability, Deadline, Cost etc. provided by the user. The idea behind this strategy is to schedule tasks on the basis of QoS negotiation between user requirements and the services delivered by servers. QoS negotiation is achieved by using distribution of QoS parameters among tasks.

3.2.12 Load- Balanced Scheduling

In [37], an improved load balanced algorithm is introduced that uses the basics of min-min algorithm. Hence it is named as "LBIMM" i.e. Load Balance Improved Min-Min scheduling algorithm. Cloud providers provide the resources on pay-as-per-usage basis. Therefore the cost per resource unit depends on the services selected by the user. In return, the user receives guarantees regarding the provided resources. To perceive the promised guarantees, user priority aware algorithm is proposed named as "PA-LBIMM" so that user's demand could be satisfied more completely.

LBIMM Algorithm:

Step 1: Compute expected completion time for all tasks on all resources.

//min-min algorithm:

Step 2: For all tasks do step 3 to 5:

Step 3: Compute minimum completion time for all the tasks then among these minimum time values select the minimum value.

Step 4: Schedule that task on the resource on which it takes minimum time and remove from the task set.

Step 5: Update the ready time of that resource for all the other tasks and completion time for all other remaining tasks on all resources.

// rescheduling steps:

Step 6: Repeat step 7 to 10 until less completion time is produced for smallest task on heavy load resource by any other resource.

Step 7: Compute task with minimum execution time on heavy load resource.

Step 8: Compute minimum completion time for that task and resource on which it takes minimum time.

Step 9: Reassign task to the resource on which it takes minimum time, if computed minimum completion time is less than that of make span.

Step 10: After reassigning, update the ready time of both the resources i.e. heavy load resource and resource on which it takes minimum time.

PA-LBIMM Algorithm:

Step 1: Divide the tasks into two groups i.e. VIP group and ordinary group, according to user-priority demand.

Step 2: Compute expected completion time for all tasks of VIP group on all VIP qualified resources.

Step 3: Apply min-min algorithm for all tasks of VIP group, same as that of LBIMM algorithm.

Step 4: Compute expected completion time for all tasks of ordinary group on all resources.

Step 5: Now apply min-min algorithm for all tasks of ordinary group.

Step 6: Perform rescheduling steps, which are stated in LBIMM algorithm.

3.2.13 Real Time Scheduling

In [38], a strategy is proposed to schedule real-time tasks non-pre-emptively in cloud computing environment. The motive of the proposed strategy is to maximize the utilization. But increase in throughput and minimize average response time are the two main considerations of proposed strategy. Two different time utility functions are associated with each task at the same time such as:

- a profit time utility function, and
- a penalty time unit function

This approach not only provides the minimum completion time but also penalizes if not completes within a given deadline or if real-time tasks abort.

3.2.14 TPD Scheduling

In [39], an efficient scheduling algorithm is proposed in which first users select their method on the basis of application requirements and then prioritized. The proposed algorithm addresses major challenges of scheduling in cloud computing environment such as: resource utilization, maximum profit, minimum execution cost etc. As the users select their method and then prioritized, so the algorithm is named as "TPD Scheduling Algorithm", Here T stands for Task Selection, P Stands for Priority and D stands for Deadline. In the proposed algorithm, user selects from following two methods: deadline-based and cost-based.

Algorithm:

Step 1: When users send requests to cloud, valid one's are allowed to select method that best fit the tasks requirements.

Step 2: After selecting any method, prioritize them using following:

- Assign higher priority to the task that will arrive first i.e. serve the requests on first-come-first-serve basis.
- If two requests have same start time, then assigns priority on the basis of number user has used the cloud i.e. to the old user (whose count value is more).
- If cloud count is also same, then priority is assigned according to the cost based count.
- If it is also same then assign priority on the basis of id, provided at the time of registration.

Step 3: After assigning priority, schedules the task and update the application status.

3.2.15 Locality Driven Scheduling

In [40], a heuristic scheduling algorithm is proposed. It occurs in two phases: in first, tasks allocation is done and in second, total job completion time is reduced. Hence it is known as Balance-Reduce (BAR). It is a locality driven algorithm, because it can dynamically change the locality of data, by regularly examine the network state. If a job consists of number of tasks, then it is considered to be completed only when all tasks finish their execution. The first phase of BAR is balance phase, in which tasks are scheduled to resources in a balanced way. And second phase is reduce, in which total job completion time (make span) is reduced.

4. CONCLUSION

Cloud computing has been lifeline of present day systems that rent computing resources on-demand, billing is done on pay-as-you-go basis, and enable geographically segmented users to work on the same physical infrastructure. This paper presents various resource allocation and scheduling strategies for resource management in cloud computing environment. In cloud environment, resource management is required to meet QoS requirements, to achieve high resource utilization, better system throughput etc. The strategies that are analyzed above mainly focus on various parameters such as: time, Qos, cost, location, and priority etc.

5. ACKNOWLEDGEMENTS

Sincere Thanks to HCTM Technical Campus Management Kaithal, Haryana, India for their continuous encouragement.

6. REFERENCES

[1] Brijender Kahanwal and Tejinder Pal Singh, “The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle”, International Journal of Latest Research in Science and Technology, Vol. 1, No. 2, pp. 183-187 , 2012.

[2] Sagar Girase, Rahul Samant, Mayank Sohani, and Suraj Patil, “Review on: Resource Provisioning in Cloud Computing Environment”, International Journal of Science and Research, Vol. 2, No. 11, 2013.

[3] V.Vinothina, Dr.R.Sridaran, and Dr. Padmavathi Ganapathi, “A Survey on Resource Allocation Strategies in Cloud Computing”, International Journal of Advanced Computer Science and Applications, Vol. 3, No. 6, 2012.

[4] Pinal Salot, “A Survey of Various Scheduling Algorithm in Cloud Computing Environment”, International Journal of Research in Engineering and Technology, Vol. 2, No. 2, 2013.

[5] Gunho Lee, Niraj Tolia, Parthasarathy Ranganathan, and Randy H. Katz, “Topology-Aware Resource Allocation for Data-Intensive Workloads”, ACM SIGCOMM Computer Communication Review, Vol. 41, No. 1, pp. 120-124, 2011.

[6] Zhen Kong et.al, “Mechanism Design for Stochastic Virtual Resource Allocation in Non-Cooperative Cloud Systems”, IEEE 4th International Conference on Cloud Computing, pp. 614-621, 2011.

[7] Abirami S.P. and Shalini Ramanathan, “Linear Scheduling Strategy for Resource Allocation in Cloud Environment”, International Journal on Cloud Computing: Services and Architecture, Vol. 2, No. 1, pp. 9-17, 2012.

[8] Kuo-Chan Huang and Kuan-Po Lai, “Processor Allocation Policies for Reducing Resource Fragmentation in Multi Cluster Grid and Cloud Environments”, IEEE, pp. 971-976, 2010.

[9] Daniel Warneke and Odej Kao, “Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud”, IEEE Transactions on Parallel and Distributed Systems, 2011.

[10] FetahiWuhib and Rolf Stadler, “Distributed Monitoring and Resource Management for Large Cloud Environments”, IEEE, pp. 970-975, 2011.

[11] K C Gouda, Radhika T V, and Akshatha M, “Priority Based Resource Allocation Model for Cloud Computing”, International Journal of Science, Engineering and Technology Research, Vol. 2, No. 1, 2013.

[12] Wei-Yu Lin et al, “Dynamic Auction Mechanism for Cloud Resource Allocation”, IEEE/ACM 10th International Conference on Cluster, Cloud and Grid Computing, pp. 591-592, 2010.

[13] Xindong YOU, Xianghua XU, Jian Wan, and Dongjin YU, “RAS-M :Resource Allocation Strategy based on Market Mechanism in Cloud Computing”, IEEE,pp. 256-263, 2009.

[14] Tram Truong Huu and John Montagnat, “Virtual Resource Allocations Distribution on a Cloud Infrastructure”, IEEE, pp.612-617, 2010.

[15] Satyanarayana .A, Dr. P. Suresh Varma, Dr. M.V.Rama Sundari, and Dr. P Sarada Varma, “Performance Analysis of Cloud Computing under Non Homogeneous Conditions”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 5, 2013.

[16] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hanseny, Eric July, Christian Limpach, Ian Pratt, and Andrew Warfield, “Live Migration of Virtual Machines”, 2nd Symposium on Networked Systems Design and Implementation, 2005.

[17] Bo An, Victor Lesser, David Irwin, and Michael Zink, “Automated Negotiation with Decommitment for Dynamic Resource Allocation in Cloud Computing”, Proceedings of 9th International Conference on Autonomous Agents and Multi-agent Systems, Vol. 1, 2010.

[18] Gihun Jung and Kwang Mong Sim, “Location-Aware Dynamic Resource Allocation Model for Cloud Computing Environment”, International Conference on Information and Computer Applications, Vol. 24, 2012.

[19] Nilabja Roy, Abhishek Dubey and Aniruddha Gokhale, “Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting”, Cloud Computing IEEE International Conference, pp. 500-507, 2011.

[20] HadiGoudaezi and MassoudPedram, “Multidimensional SLA-based Resource Allocation for Multi-tier Cloud Computing Systems”, IEEE 4th International conference on Cloud Computing, pp. 324-331, 2011.

[21] Hien Nguyen et al, “SLA-Aware Virtual Resource Management for Cloud Infrastructures”, IEEE 9th International Conference on Computer and Information Technology, pp. 357-362, 2009.

[22] Stephen S. Yau and Ho G., “An Adaptive Resource Allocation for Service-Based Systems”, International Journal of Software and Informatics, Vol. 3, No. 4, pp. 483–499, 2009.

[23] Griva I, Nash SG, and Sofer A., “Linear and Nonlinear Optimization”, 2nd Education Society for Industrial Mathematics, 2008.

[24] HadiGoudarzi and MassoudPedram, “Maximizing Profit in Cloud Computing System Via Resource Allocation”, IEEE 31st International Conference on Distributed Computing Systems Workshops, pp. 1-6, 2011.

[25] Patricia Takako Endo et al., “Resource Allocation for Distributed Cloud: Concept and Research Challenges”, IEEE, pp. 42-46, 2011.

[26] Upendra Bhoi, “Enhanced Max-min Task Scheduling Algorithm in Cloud Computing”, International Journal of Application or Innovation in Engineering and Management, Vol. 2, No. 4, 2013.

[27] Shamsollah Ghanbari, and Mohamed Othman, “A Priority based Job Scheduling Algorithm in Cloud Computing”, International Conference on Advances Science and Contemporary Engineering, 2012.

[28] Wieczorek M., Prodan R. and Fahringer T., ‘‘Scheduling of Scientific Workflows in ASKALON Grid Environment’’, SIGMOD Rec., Vol. 34, No. 3, pp. 56–62, 2005.

[29] Shaminder Kaur and Amandeep Verma, “An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment, International Journal of Information Technology and Computer Science, Vol. 10, pp. 74-79, 2012.

[30] Salim Bitam, “Bees Life Algorithm for Job Scheduling in Cloud Computing”, 2nd International Conference on Communications and Information Technology, 2012.

[31] Mrs.S.Selvarani and Dr.G.Sudha Sadhasivam, “Improved Cost-Based Algorithm for Task Scheduling in Cloud Computing”, IEEE, 2010.

[32] Poonam Devi, “Implementation of Cloud Computing By Using Short Job scheduling”, International Journal of Advanced Research in Computer Science and Software Engineering, 2013.

[33] Laiping Zhao, Yizhi Ren, and Sakurai, K., “A Resource Minimizing Scheduling Algorithm with Ensuring the Deadline and Reliability in Heterogeneous Systems", IEEE, 2011.

[34] M. Singh and P.K. Suri, “$\text{QPS}_{\text{MAX-MIN<>MIN-MIN}}$: A Qos Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid”, International Journal of Information and Technology, Vol. 7, No. 8, pp. 1176-1181, 2008.

[35] Dr. Amit Agarwal and Saloni Jain, “Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment”, International Journal of Computer Trends and Technology, Vol. 9, No. 7, 2014.

[36] Jayadivya S K and S. Mary Saira Bhanu, “QoS Based Scheduling of Workflows in Cloud Computing”, International Journal of Computer Science and Electrical Engineering, Vol. 1, No. 1, 2012.

[37] Huankai Chen, Professor Frank Wang, Dr Na Helian, and Gbola Akanmu, “User-Priority Guided Min-Min Scheduling Algorithm for Load Balancing in Cloud Computing”, Parallel Computing Technologies National Conference, pp. 1-8, 2013.

[38] Shuo Liu, Gang Quan, and Shangping Ren, "On-Line Scheduling of Real-Time Services for Cloud Computing", IEEE, 2010.

[39] Dr.V.Vaithiyanathan, R.Arvindh Kumar, S.Vignesh, B.Thamotharan, and B.Karthikeyan, “An Efficient TPD Scheduling Algorithm for Cloud Environment”, International Journal of Engineering and Technology, Vol. 5, No. 3, 2013.

[40] Jiahui Jin, Junzhou Luo, Aibo Song, Fang Dong and Runqun Xiong, “BAR: An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing”, 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2011.