Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 12, Number 2 (2016), pp. 1647-1662

http://www.ripublication.com

A Novel Hierarchical Scheduling Method for Managing Parallel Workloads in Cloud

R. Valarmathi [1*] and Dr. T. Sheela [2]

[1]Research Scholar, Sathyabama University and*
Assistant Professor, Sri Sairam Engineering College,
Chennai, Tamil Nadu, India.
[2] Professor, Department of Information Technology,
Sri Sairam Engineering College, Chennai, Tamil Nadu, India.

Abstract

Nowadays one of the most useful computing paradigm is cloud computing which can be accessed as "Pay-n-Use" method and it provides services in terms of software, utility and platform. Cloud computing leads people to run any kind of complex applications and it allows accessing data centers remotely and costing effectively. It is well known that most of the complex applications need parallel processing efficiency. Transferring data, messages, any kind of communication and synchronization are some of the parallel processes involved in the cloud where these processes don't require CPU utilization high due to parallelism. Also cloud computing supports parallel processing, distributed processing and basically it is multi-tenant environment. Scheduling the task for parallel processing among multi-tenant involves time complexity and it affects the performance of the cloud computing. The existing studies were concentrating on consolidating the parallel workloads using virtualization techniques. But in this paper, a novel hierarchical scheduling approach is used to schedule the workloads by considering the attributes based weight value, generated dynamically. The weight value is calculated by examining the various attributes of the workloads in terms of resources, time taken to complete the job and availability of the resources. The best value of the attributes is elected by Artificial Immune System and the server mode is verified if it is in Sleep, Idle or in Busy and assign the job in order to save the energy. The simulation based experiment is carried out in GreenCloud software and the performance is evaluated.

Keywords: Cloud Computing, Scheduling, Resource Allocation, Time Complexity, Parallel Processing.

Introduction

The general concept of cloud computing is that the user's data are stored in the network data center rather than in a personal hard disk or any other storage device, so that the user can use the data from any system from anywhere in the world over a network connection. The data stored in the data center are maintained by the service provider of the individual. The users access the stored data from anywhere in the world using Application Programming Interface (API) software. This software is also available as freeware and shareware in public networking sites and also in the service provider's site. Service providers offer three types of services such as Infrastructure, Software or Platform. These services provided in a cloud network are named as Infrastructure as a Service (IaaS), Software as a Service (SaaS) and Platform as a Service (PaaS). The main benefits of using cloud computing is remote access of resources, low cost and reliability.

Background Study

The cloud computing model agree a cost-effective solution for running professional applications through the necessity of virtualization technologies, highly scalable distributed computing, and data administration techniques as well as a pay-as-you-go cost plan. In modern years, it also attempts high-performance computing efficiency for applications to resolve complicated problems [2]. Improving, resource use is necessary for realizing cost efficaciousness. Low use has protracted an event in data centers. Servers in an exemplary data center perform at 10 to 50 percent of their highest use level [3]. 10 to 20 percent use is, ordinary in data centers [4]. In order to achieve high performance in terms of delay during parallel processing it is essential apply concurrent process on jobs. For clients, cloud computation afford the deduction of the basic model which reserve them the price of designing, construction and maintaining a data center to host their Application Environments (AE) [1]. By operating leverage system virtualization in a data center, diverse Virtual Machines (VM), which are the components of AEs, can be joined on one Physical Machine (PM). A VM is loosely conjugated with the PM it proceeds on; as an effect, not only can a VM be originate on any PM, but also, it can be wander to other PMs.

Currently, VM migration processes can be accomplished in seconds to minutes confide on the size, activity and bandwidth of the VM and PMs [1]. The process migration makes it possible to dynamically accommodate the data center usage and tune the resource arrangement to AEs. Also, these adjustments can be machine-controlled through characteristically determined strategies in order to continuously control the resources in the data centers with less human intervention. We relate to this task as the procedure of dynamic and self-governing resource administration. It is well known that various numbers of complex applications are entering into cloud environment and data center, which are remotely connected with the cloud. The applications coming into cloud environment requires parallel execution. While executing the jobs parallel, it is essential to provide hierarchical order in order to avoid conflict, congestion and overhead. In this paper the attributes of the jobs are considered and according to the value of the attributes the jobs are arranged in a

hierarchical manner. Whenever parallel processing increases, the CPU utilization also gets increased, which is a challenging problem in cloud environment? In order to provide a better solution, a hierarchical scheduling algorithm is used. In parallel Job processing there are two different factors to be considered in order to reduce the node utilization.

1. In order to execute a parallel job, a number of nodes are required. Also when jobs enter with different node requirement, it is necessary to classify the jobs or nodes for proper execution. If the classifications of nodes under jobs are not perfect, then some of the jobs or nodes are idle [5-8].
2. In parallel processing waiting time is high, since the utilization of the node is low.

One of old and famous batch scheduling algorithm used in parallel jobs is FCFS-First Come First Serve [9]. According to the entry, the scheduler processes the job and the job has a specification about the number of nodes needed. When a request coming into cloud server, the attribute "number of nodes" is verified, and if it finds the number of nodes availability is sufficient then the processing time and scheduling is efficient. Else the jobs waiting time is extended due to the long execution time of the nodes in the cloud. The FCFS handles the node separation, scheduling and job assignment to the resources in an improved manner. Among the several works proposed earlier for resource allocation in cloud computing systems, some papers like [10], [11] discussed about Quality of Service in resource allocation. According to these papers the demands of highly prioritized applications are solved first. In [12] a utility based resource allocation is proposed. This method regulates the cost-service trade-off. In [13], [14] the system uses dynamic migration. Since they use policy based algorithms, the migration process becomes much more complex.

In this paper, we propose a hierarchical priority-based scheduling method which is applied for parallel jobs with the following goals:

1) Improve the utilization of servers allocated to these jobs;
2) Preserve the Hierarchical order of jobs when available resources satisfy the needs of these jobs.

Existing System

The earlier parallel scheduling algorithms commonly takes care of response generation by initializing top level to bottom level priority allocation where recent cloud needs this activity to assign priority due to heavy work load entering into cloud. In the existing approach a priority based method is used to consolidate the workloads in parallel by concentrating memory capacity. The virtualization technology is used to partition the CPU capacity in terms of tiers such as background and foreground tier. The foreground VM is assigned to high priority CPU and background VM is assigned to low priority CPU [16]. A scheduling algorithm is applied to parallel jobs to make efficient use of two tier VMs to improve the responsiveness of these jobs. Generally virtualization is used to reduce the request queue. Controlling the VMs is too difficult

dynamically. Also VMs are restricted in terms of memory size and it takes more time to activate the VM dynamically according to the new request arrives.

Proposed System

In this paper, the hierarchical scheduling algorithm is proposed to improve the resource utilization, reduce the execution time, and assign priority in terms of job size and requirement hierarchically. The hierarchical utilization examines the jobs with same resources, nearest resources, amount of resource utilization in terms of job size and time taken to complete the resource utilization in terms of attributes of the Jobs. Higher priority can be assigned for the jobs which need same kind of resources and less execution time. The proposed method provides an examination based systematic way to schedule the workloads on attribute evaluation. In contrast with the existing system, the proposed approach examines the attributes of the incoming jobs and evaluated.

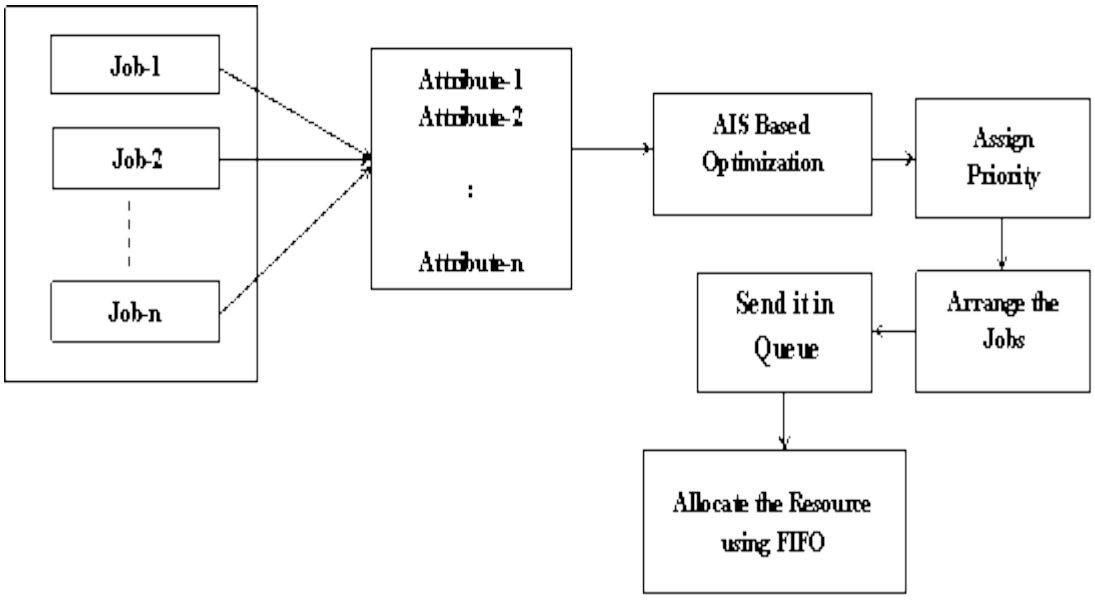

Figure-1: Over All Functionality of the Proposed Approach

The set of jobs (N number of Jobs) entering into the cloud will be fetched and the attributes are extracted for examination. All the attributes of each job is given to AIS approach and the FF (Fitness Function) value is calculated. According to the FF value the jobs are arranged in ascending order and given to Job queue. The FF value defines the hierarchy value of each job where the FF value lies from 0 to 6 and it is calculated using the equation (6). Finally the queued jobs are allocated by the appropriate resources. If any two or more jobs require same resource, then the jobs are allocated parallel to the resources.

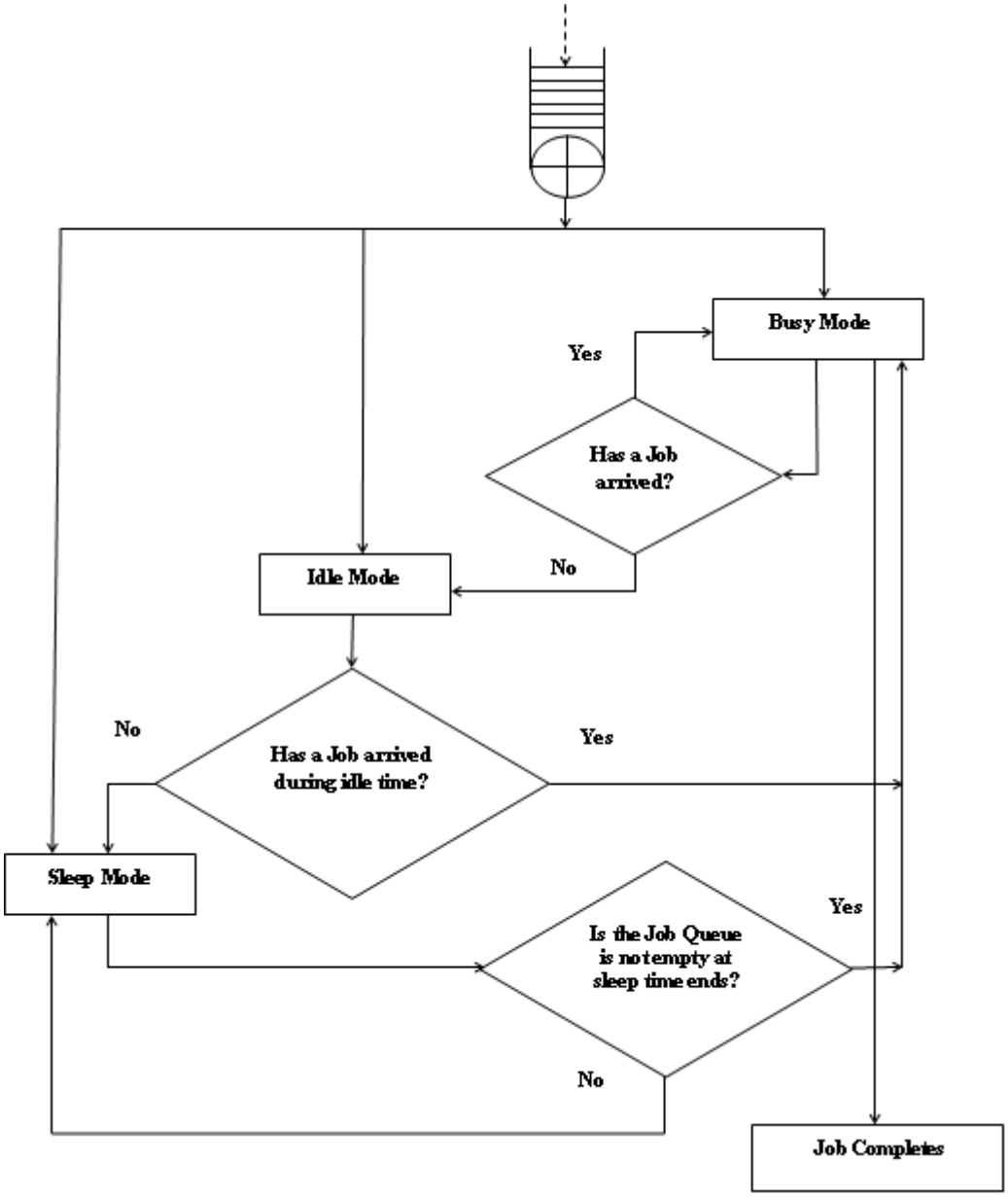

Figure-2: Job Allocation In terms of Server Mode [15].

In this paper the job–resource allocation follows predefined scenarios taken from [15] where it helps to examine the resources and server in order to save the energy. The status of the resources and server is defined in three modes as idle, sleep and busy [15]. All three modes of the server are related to one another. If the server is busy, it means that all the current job requests are finished. Server is idle it denotes that a server stays in an idle mode and waits for subsequently arriving jobs before switching into a sleep mode. If a job arrives during an idle period, a server can switch into a busy mode and start to work immediately. A server begins a next idle period until all

job requests have been successfully completed. If there is no job arrival, a server switches into a sleep mode when an idle period expires. A server remains in a sleep mode, if the number of jobs in the queue is fewer than the controlled N value. Otherwise, a server switches into a busy mode and begins to work.
In this paper the some of the assumptions are made in order to improve the efficiency. They are:

1. The job size is known. From the job size the execution time is calculated.
2. When a job enters into the cloud, it is assumed that the scheduler and the server can find out and identify the requirements of the job.
3. The data center is identified by the server and the dispatcher.
4. According to the size and the execution time, the CPU utilization time can also be calculated.

The execution time and the CPU utilization time can be calculated according to timer integrated in the CPU. Also the data center is identified from the request command, since all the requests have the database name.
Let us assume the number of incoming jobs ***J*** at time ***T*** is

$$J = \{J1, J2, \dots, Jn\} \text{---} \quad (1)$$

The properties P_{Ji} of a job $\boldsymbol{J_i}$ is

$$PJi = \{PJ1, PJ2, \dots, PJm\} \text{---} \quad (2)$$

Where,

$$P_{Ji} \in \{S, ET, CPUT, NN, DCN, DIST\} \text{---} \quad (3)$$

Where S denotes the size of the job and
ET: Execution Time
CPUT: CPU utilization time
NN: Number of Nodes
DCN: Data Center Number
DIST: Distance between requester and the server

In order to provide higher priority for the job $\boldsymbol{J_i}$ the properties of the job $\boldsymbol{J_i}\ \boldsymbol{P_{Ji}}$ should satisfy the fitness function defined in the AIS approach. One of the optimization techniques is Artificial Immune System. It is used to find the best value for large problems in various domains. AIS are used to optimize large size problems which can emulate processes by utilizing the mechanisms of the biological immune system. AIS have local and global searching methodologies by creating affinity based mutation and cloning. AIS compare the results of the small problems with the results of the large problems. Through which it can able to obtain a best solution for large problems. Since, in this paper AIS is utilized to fetch the optimum attribute value based job to be scheduled [means priority allocation], when a request comes for execution or for a resource, the system calls the AIS for choosing the best resource from resource queue. Before obtaining an optimum solution with AIS, first get the list of free resources available in the queue. Once a request comes from a cloud user, the AIS will be executed to optimize the strategies and finds the best attributes and

arrange the jobs in a sequential manner according to the priority value in ascending order. In the arranged queue, the first job has top most priority and the last job has low level priority. The overall job scheduling procedure verifies the objectives in terms of attributes.

Initially the jobs and the resources are mapped using a normal mapping method using their attributes. To do this a set of initial population ***P(0)***{ individual solution representing a job} is generated randomly. The attributes of each individual population $\boldsymbol{P_i^J}$ is represented as $\boldsymbol{P_i^J} = \left(\boldsymbol{p_{J1}^T}, \boldsymbol{p_{J2}^T}, \ldots., \boldsymbol{p_{Jm}^T}\right)$ where, ***T*** says the present population generation at the time ***T***, **i = 1,2,…., s**, and ***s*** says the size of the entire population in a stipulated time interval.

Fitness Function

That is the FF value is changed from 0 to 6. The job which is having FF value of 6 is assigned with higher priority. In case of same level of FF, the jobs are scheduled using FCFS [17]. Whereas the jobs which are having the Fitness Function (FF) value is 0 are assigned with lower priority. The priority is changed from higher to lower according to the FF value from 0 to 6 in increased order.

$$\text{FF} = \left\{ \begin{array}{c} S = 1\ if(\ S == \min(all\ S(J))) \\ else\ S = 0 \\ ET = 1\ if\left(ET == \min(allET(J))\right) \\ else\ ET = 0 \\ CPUT = 1\ if(\ CPUT == \min(all\ CPUT(J))) \\ else\ S = 0 \\ DIST = 1\ if\left(DIST == \min(all\ DIST(J))\right) \\ else\ DIST = 0 \\ NN = 1\ if(NN\ \subseteq NN(PJ) \\ else\ NN = 0 \\ DCN = 1\ if\ DCN\ is\ available \end{array} \right\} = 6\text{---} \quad (4)$$

Every antibody (individual) is treated as a candidate solution which can be represented by a binary string of bits. The user can set the string length to obtain the best solution for the problem. 0 or 1 represents the gene in the chromosomes. After generating the populations successfully, the affinity value of each individual is investigated and stored for future operations. The AIS approach is applied for job scheduling to contract with the optimization problem where the affinity function contracts with energy efficiency, time and cost. The affinity function in AIS algorithm is defined as follows:

$$\text{aff(x)} = e^{\min \text{FF}_\text{J}}\text{---} \quad (5)$$

Cloning operation combines identical cells in the human body which is generated from the same ancestor and only antibodies with high affinity will be cloned to attack pathogens. Also cloning is an antibody random map, persuaded by affinity. According to the affinity function, all the antibodies are evaluated and stored in decreasing order.

After arranging in decreasing order the higher affinity values are selected for next generation. Then the selected antibodies proliferate into certain copies, the copied and original ones are replicated in the current population. Afterwards, the antibodies in the population will implement mutation operation. Then the antibodies in the population are mutated. AIS have two different mutation operations as pair wise and inverse mutation which can generate a new mutated individual $\boldsymbol{P_J^{T'}}$.

The entire AIS process is applied for Job scheduling approach where the Job attributes and the resource behavior such as speed, time taken to process, cost to process all are optimized and the best resource is chosen and scheduled. The AIS selects the JOB with less cost, less time, less energy consumption, available resource, nearest data center, less number of nodes need for processing the job requested from the user. Hence AIS based hierarchical approach provides best scheduling approach for Job execution and resource allocation strategy for cloud computing. The AIS algorithm is given in the form of pseudo code, where it can be implemented and evaluate the performance.

Algorithm-1: Artificial Immune System (AIS)

***Input*:** P is the population generated randomly
S is the total number of population
K is the number of iteration
R is the rejection ratio
Initialize OFV, the objective function value

1. Initialize a random population P
2. Compute OFV for each P
3. Compute the affinity value for each P using AFV= 1/OFV
4. Compute the rate of cloning value $\mathrm{ROC} = (\frac{1}{\mathrm{OFV}} * \mathrm{P}) / \mathrm{total}$
5. Generate clones to the problem according to ROC value
6. According to the cloning size, the value of S is increased
7. Do mutation in terms of pair wise and inverse on S
8. Arrange all population in ascending order and maintain the initialized size P by eliminating R% populations having highest OFV
9. Generate R% new populations P in order to maintain the Initial population size.
10. Repeat step 2 until getting best OFV.

The entire functionality of the AIS is described in terms of optimizing the properties for efficient resource allocation. The OFV is obtained using certain number of parameter assignment as P, K and R. In this paper,
P=100, K=500 and R=20%.
These above values may be changed to get best solution.

Initialization

It is clear that the population-P, Iteration-K and replacing factor-R are initialized. The population is generated randomly to create possible combinations through which each combination represented by one string called as clones and the number of clones is P.

The entire set of population is taken into account of the initial population. Example, the string is

$$SE \in \{S, ET, CPUT, NN, DCN, DIST\}$$

Each resource has more values in SE and it can be mutated with different value for obtaining optimum value as OFV.

Objective function

The main objective function of proposed model is to allocate the best resource according to the optimum values of the incoming jobs. The OFV calculated for all the solutions P and affinity value is computed using the following equation is:

$$Affinity\ value\ =\ 1/OFV$$

It is denoted that the affinity value is inversely proportional to the OFV value.

Clonal Selection

The String SE is selected and cloned. This number of cloning is generated and applied directly proportional to the affinity value with a rate of cloning-[ROC] and it can be calculated by:

$$\text{ROC} = \frac{affinity\ value\ of\ P}{total\ affinity\ value\ in\ the\ solution}$$

Clonal Expansion

Original copy of one string is called as a clone. According the ROC value, new clones can be generated. Example if ROC value is 1. 4, then the number of new clones is 2. It increases the population for obtaining best value as optimum.

Mutation

Mutation is a process that can make new clones of the population. Mutation can change the clone using reverse mutation and pairwise mutation. The following Figure-3 illustrates the inverse mutation of the string S, where the numbers of resources are 12, and the inverse mutation starts at location 3 to 8.

Figure-3: Inverse Mutation

After the inverse mutation, the OFV is calculated for the mutated string and check that the mutated OFV is lesser than the clone OFV, the original string is replaced by the mutated string, else, the original string is retained. The following Figure-4

illustrates the pairwise mutation of the string S, where the numbers of resources are 12, and the pairwise mutation starts at location 3 and 8.

Figure-4: Pairwise Mutation

After the pairwise mutation, the OFV is calculated for the mutated string and if the mutated OFV is lesser than the clone OFV, then the original string is replaced by the mutated string, else, the original string is retained. After Mutation, the number of strings in the improved string is higher than the population size. To maintain the initial population size, arrange the improved strings in ascending order and remove the strings having highest OFV.

Receptive Editing Process

R=20%, where 20% of the highest OFV based clones are replaced by newly generated population and repeat the same process described above to obtain the best OFV. Repeat the above steps until the number of iterations mentioned. The AIS selects the resource with less cost, less time, less energy consumption for processing the job requested from the user. Hence AIS model provides communication aware, best resource allocation strategy for cloud computing, according to scheduling. The above algorithm describes the entire logical functionality of the AIS for optimizing the attributes of the jobs entering to the cloud. If the scheduling process schedules the jobs in a hierarchical manner it avoids congestion and reduces execution time and power consumption to make the cloud as efficient.

Simulation Settings

The simulation is setup using TCL files located in./***src/scripts/ directory***. The mainfile *main.tcl* determines the data center topology and simulation time. It also calls a set of the following simulation scripts:

Table 1: Simulation Parameter Considered for Simulation

Sr. No	**Parameters**
1.	Job Number
2.	Job size
3.	Number of Nodes required
4.	Approximate time need for execution
5.	Approximate distance among the Job request place and the data center
6.	Exact Resource need
7.	Resource available Freely

_ ***setup_params.tcl*** contains general configuration of servers, switches, tasks, monitoring and migration
_ ***topology.tcl***Creates the data center network topology
_ ***dc.tcl*** Creates the data center servers and VMs
_ ***user.tcl*** Defines behavior of cloud users
_ ***record.tcl*** Sets up runtime results reporting procedures
_ ***finish.tcl*** Calculates and reports simulation statistics.
The simulation is run by executing the.***/run/ script***. It contains the following three simulation parameters:
_ ***Data center:*** *Load* defines the size of the data allocated

Data center**:** *Load* describes the quantity and requirements need for computing and jobs incoming with respect to the data center capacity. Usually the load should be between 0 and 1. The load close to 0 represents an idle data center, while the load equal or greater than 1 would saturate data center.
Simulation time: Shows the maximum time allowed for task execution, while tasks deadlines have impact on the timesharing behavior of tasks. Longer deadlines allow more tasks to be executed in parallel on a single host or a VM.
Memory requirement: Defines the maximum size of the simulated memory resource that can be used in multitasking.
The number of jobs, resources with more number of requests is given as input parameters and evaluated according to the time, cost, distance, and data center. Number of nodes needed, energy and available resources are optimized using AIS algorithm. The number of nodes N and the number of attributes R is changeable to fetch more graphical representation based results. The simulation is repeated with various numbers of nodes as 20, 30, 40 and 50.

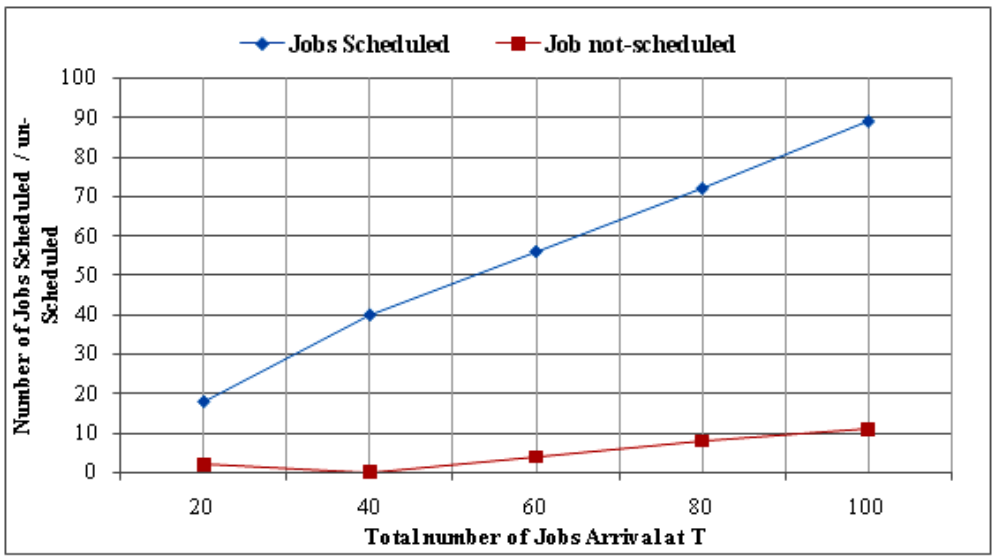

Figure-5: Number of Jobs versus Scheduled\Un Scheduled

Figure-6: Number of Jobs versus Priority Assigned

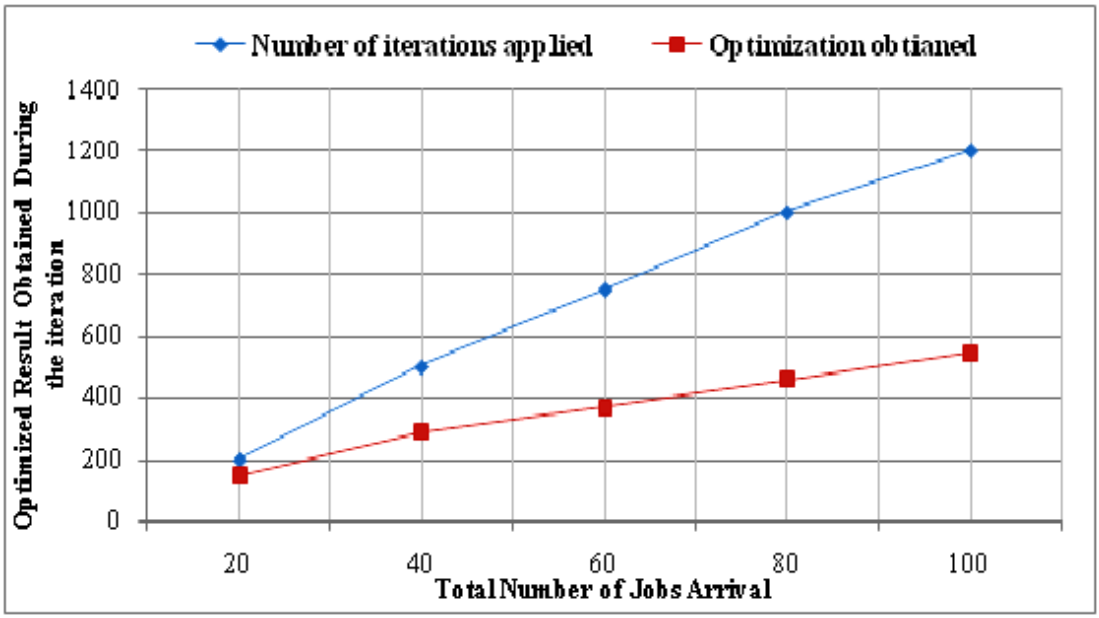

Figure-7: Number of Jobs versus Result Obtained at the Iteration Number

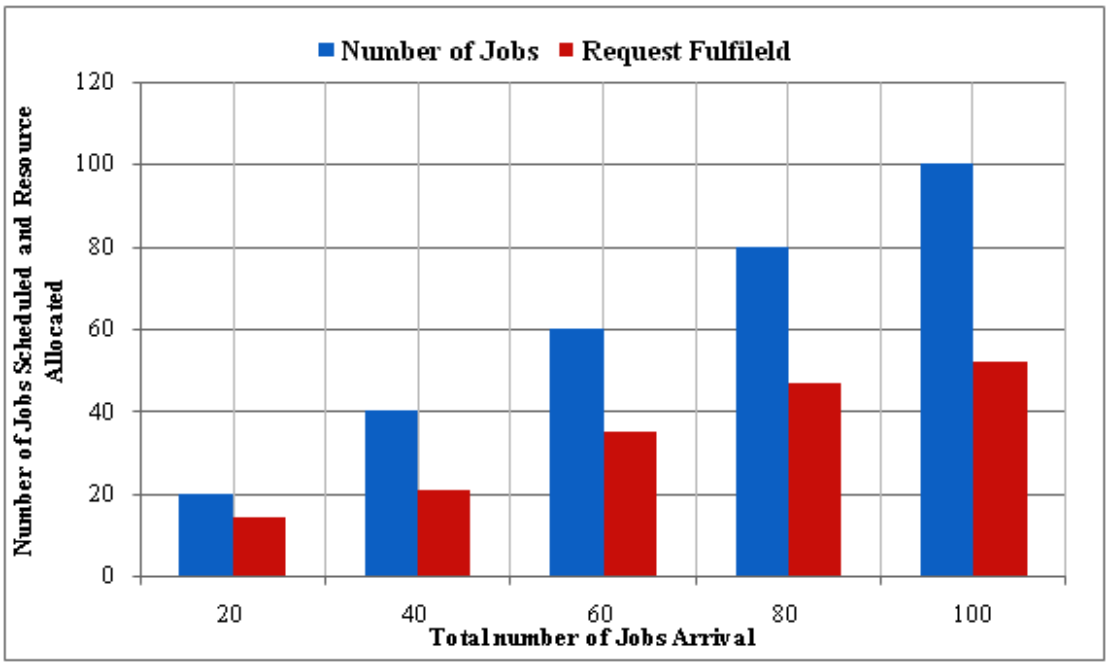

Figure-8: Number of Jobs Scheduled and Resource Allocated

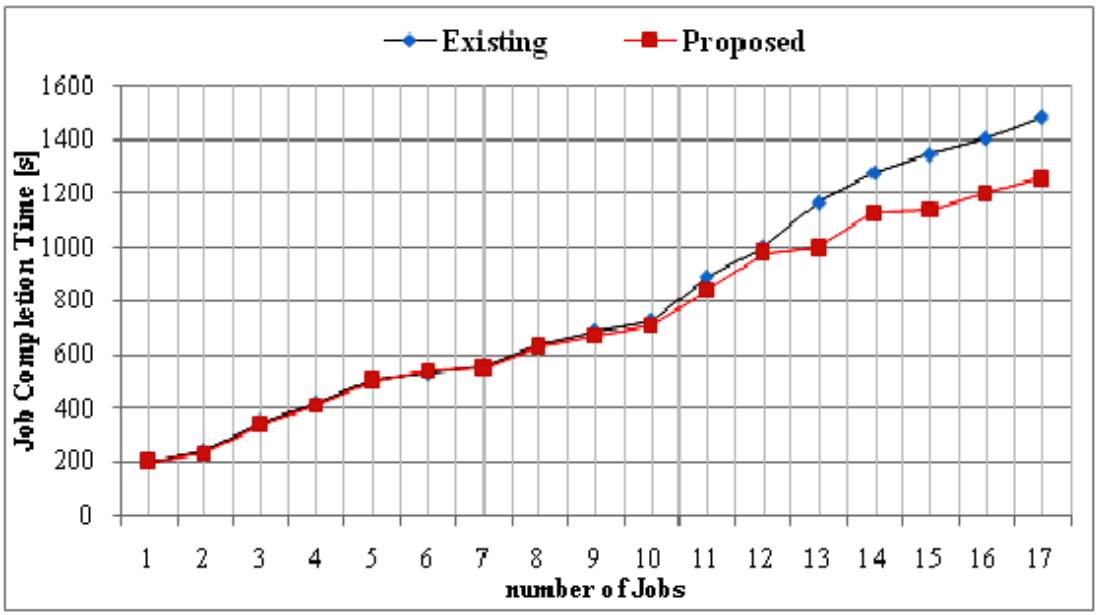

Figure-9: Number of Jobs versus Response Time

There are 5 rounds of operations simulated for completing the experiment. In each round, the number of job arrival rate is changed as 20, 40, 60, 80 and 100. It is because to evaluate the performance of the proposed approach. Each job needs a

schedule and priority level within 0 to 6 for allocating the resource, then according to the priority level the resource is allocated. In this paper, the job scheduling is done by verifying the attributes given in equation (3). These properties are evaluated using AIS by comparing the attribute values with the Fitness Function. From the arrived jobs the number of scheduled jobs and un-scheduled jobs according to the fitness function is shown in Figure-5.

From this Figure-5, it is clear and can be understood that the number of un-scheduled jobs are increased 2% higher in each round than the previous round. In some occasion all the incoming jobs are scheduled. It may happen according to the resource availability and if the size of the jobs is small. Once the jobs are scheduled, it is assigned by a priority level. According to the FF value calculated from the properties using AIS, the priority value is assigned. The number of jobs assigned with higher priority and lower priority depends on the optimization process. A job receives a higher priority if and only if, the job is small, take less time, the resource is available and the parallel jobs are also less. The priority assignment for the total number of job during the time T is shown in Figure-6. The job getting higher priority and lower priority is not constant, it will get vary due to network situation. All optimization techniques are executed with a profound variable as number of iterations. The number of iteration can provide more accurate best value in the entire domain, due to the mutation, cloning and affinity of the AIS approach. In this paper a constant number of iteration is applied to all the rounds, where the best result is obtained in certain iteration number. Here in all the five rounds, the number of iterations assigned is 200, 500, 750, 1000 and 1200, where the optimized result obtained during the iteration of 145, 287,367, 456 and 544. It is shown in Figure-7.

To compute the efficiency of the proposed approach, the number of jobs which is scheduled and resource allocated is calculated. Out of 20, 40, 60, 80 and 100 number jobs 14, 21, 35, 47 and 52 is scheduled and resource allocated and this is shown in Figure-8. The jobs is allocated by a resource if and only if the job is scheduled and resource allocated in terms of scheduling. First the response time obtained by existing and proposed algorithms are compared and shown in Figure-9. The number of jobs and the response time is represented in X-axis and in Y-axis respectively. These above results are showing the efficiency of the proposed approach in terms of scheduling and resource allocation.

In order to compute the load balancing efficiency, it is essential to take a right decision by evaluating the client-server communication. Since, all the nodes in the cloud are communicated in heterogeneous manner all the request, response time taken to process the request and the response and load balancing are examined and investigated. According to the load the computational memory is also investigated, since, virtualization can be handled to avoid memory problems. During the examination the server, resource types (computing, memory, storage and networking), resource suppliers (system configuration) and resource capabilities ($core^1$, $core^2$, $core^3$, $core^4$, RAM in GBs) are investigated. According to the investigation the scheduling methodology is effective and efficacy is shown in Figure-10. From Figure-10, it is clear that the efficacy of the scheduling methodology depends on the resource configurations.

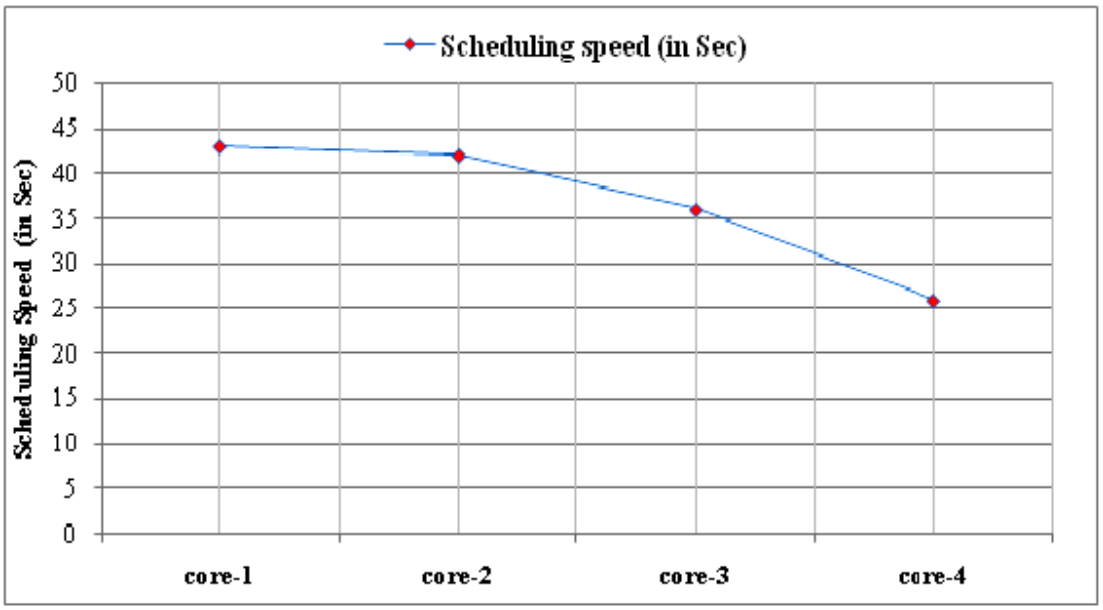

Figure-10: Scheduling Versus Resource Capabilities

Conclusion

The main objective of this paper is to provide a scheduling mechanism to allocate the resources efficiently. To do this, the paper proposed a novel method which assigns priority as a hierarchical order from 0 to 6, where the job having higher priority value is assigned the resources first. After calculating the priority value, the jobs are arranged in a queue and passed to resource allocation. The jobs are allocated by resources, according to the priority value. To select the best jobs artificial immune system approach is used where the jobs are optimized by computing the value of the attributes assigned to the incoming jobs. From the obtained results, it is concluded that 50% of the incoming jobs are resource allocated and executed efficiently.

References

[1] YagızOnatYazır, Chris Matthews, Roozbeh Farahbod, "Dynamic Resource Allocation in Computing Clouds using Distributed Multiple Criteria Decision Analysis", IEEE 3rd International Conference on Cloud Computing, 2010.

[2] "High Performance Computing (HPC) on AWS," Amazon Inc., http://aws. amazon. com/hpc-applications/, 2011.

[3] L. Barroso and U. Holzle, "The Case for Energy-Proportional Computing," Computer, vol. 40, no. 12, pp. 33-37, Dec. 2007.

[4] J. Hamilton, Cloud computing economies of scale, keynote at AWS Genomics & Cloud Computing Workshop, Seattle, WA, 08. 06. 2010, 2010. Available

from http://www. mvdirona. com/jrh/TalksAndPapers/JamesHamilton_GenomicsCloud20100608. pdf.

[5] D. Feitelson, "A Survey of Scheduling in Multi Programmed Parallel Systems", IBM TJ Watson Research Center, 1994.

[6] D. Feitelson and B. Nitzberg, "Job Characteristics of a Production Parallel Scientific Workload on the Nasa Ames ipsc/860," Proceedings of Workshop Job Scheduling Strategies for Parallel Processing, pp. 337-360, 1995.

[7] J. Jones and B. Nitzberg, "Scheduling for Parallel Super Computing: A Historical Perspective of Achievable Utilization," Proceeding of Workshop Job Scheduling Strategies for Parallel Processing, pp. 1-16,1999.

[8] L. G. Valiant, "A Bridging Model for Parallel Computation," Comm. ACM, vol. 33, no. 8, pp. 103-111, 1990.

[9] U. Schwiegelshohn and R. Yahyapour, "Analysis of First-Come-First-Serve Parallel Job Scheduling," Proc. Ninth Ann. ACM-SIAM Symposium on Discrete Algorithms, pp. 629-638, 1998.

[10] D. Gmach, J. Roliaand, L. cherkasova, "Satisfying service level objectives in a self-managing resource pool", In Proc. Third IEEE international conference on self-adaptive and self-organizing system. (SASO'09) pages 243-253. IEEE Press 2009.

[11] X. Zhu et al," Integrated capacity and workload management for the next generation data center", In proc. 5th international conference on Automatic computing (ICAC'08), pages 172-181,IEEE Press 2008.

[12] Dorian Minarolli and Bernd Freisleben," Utility –based Resource Allocations for virtual machines in cloud computing", (IEEE, 2011), pp. 410-417.

[13] A. Singh,M. Korupolu and D. Mohapatra, "Server-storage virtualization: Integration and Load balancing in data centers", In Proc. 2008 ACM/IEEE conference on supercomputing (SC'08) pages 1-12, IEEE Press 2008.

[14] T. Wood et al, "Black Box and gray box strategies for virtual machine migration", In Proc 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI 07), pages 229-242.

[15] Yi-Ju Chiang, Yen-ChiehOuyang, Ching-Hsien (Robert) Hsu, "An Efficient Green Control Algorithm in Cloud Computing for Cost Optimization", IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 3, NO. 2, APRIL/JUNE 2015.

[16] Xiaocheng Liu, Chen Wang, Bing Bing Zhou, Junliang Chen,Ting Yang, and Albert Y. Zomaya, "Priority-Based Consolidation of Parallel Workloads in the Cloud", IEEE Transactions On Parallel And Distributed Systems, VOL. 24, NO. 9, SEPTEMBER 2013.

[17] Y. Etsion and D. Tsafrir, "A Short Survey of Commercial Cluster Batch Schedulers," Technical Report 2005-13, The Hebrew Univ. of Jerusalem, May 2005.