Asian Journal of Information Technology 15 (16): 2965-2972, 2016
ISSN: 1682-3915

Process Size Based Dynamic Scheduling for Resource Optimization in Cloud Environment

[1]V.G. Ravindhren and [2]S. Ravimaran
[1]Seshasayee Institute of Technology, Tiruchirappalli, Tamil Nadu, India
[2]Department of Computer Science and Engineering, Software System Group Lab,
Anna University, M.A.M. College of Engineering, Tiruchirappalli, Chennai, Tamil Nadu, India

Abstract: In the current business era, the cloud computing model provides a pool of resources for clients that are easily accessible, available, scalable and providing "pay as you use" provisioning for clients. Therefore, the cloud service providers are outfitted with client satisfaction and thereby attracting more clients and getting more profits. This could be achieved by implementing a reliable, effective and efficient resource management approaches. When the client real time requirements are rapid and dynamically changes from time to time, all conventional resource allocation mechanism may fail to meet the client satisfaction. Therefore, this study presents a new dynamic resource management framework using workflow scheduling called Process Size based Dynamic Scheduling(PSDS) which maximizes customer satisfaction as well as providers profit. The results demonstrate the efficiency of our proposed framework and shows it has a better option in terms of increase in resource utilization, reduced waiting time and reduction in missed process considerably.

Key words: Cloud computing, resource management, waiting time, client satisfaction, PSDS

INTRODUCTION

Now a days, researchers face with numerous kinds data and most of data cannot be classified into regular relational and structured type data. Hence new solutions are required for data processing. In the current trend data are generated and processed rapidly, so we need powerful platforms and infrastructure support. Extracting valuable information from raw data is especially difficult considering the velocity of growing data from year to year and the fact that 80% of data is unstructured. In addition, data sources are heterogeneous and are located in different situations or contexts. Cloud computing which concerns large-scale interconnected systems with the main purpose of aggregation and efficient exploiting the power of widely distributed resources, is a viable solution. Resource management and process scheduling play an essential role in cases where one is concerned with optimized use of resources. This is why, cloud computing infrastructures runs reliably and permanently to provide the context as a "public utility" to different services.

Cloud systems are highly dynamic in its structure because the user requests must be respected as an agreement rule, so SLA play an essential role. When ubiquitous systems become clients for cloud systems new algorithm for events and process scheduling and new methods for resource management should be designed in order to increase the performance of such systems. The adaptive methods used in context are oriented on: self-stabilizing, self-organizing and autonomic systems; dynamic, adaptive and machine learning based distributed algorithms; fault tolerance, reliability, availability of distributed systems.

However, the cloud is "a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured for optimum resource utilization. This pool of resources is typically exploited by a pay-per-user model which offers by the Infrastructure by the Provider by means of customized SLA's. Despite the numerous advantages that the Cloud offers, business organizations hesitated and expressed a level of doubtfulness about adopting the services. This was because the thought of their data residing on someone else' server which heighted their fear concerning security of these data. Also, because resources in Cloud computing are obtained as a service, questions about the Quality of Service (QoS) arose.

Scheduling is a major process in a cloud computing environment. In cloud computing environment datacenters take care of these processes. The datacenters receive processes from the datacenter brokers which arrive from different users and maintained in queue. In that we propose a process scheduling for dynamic

Corresponding Author: V.G. Ravindhren, Seshasayee Institute of Technology, Tiruchirappalli, Tamil Nadu, India

mapping of available resources associated with priorities. Our model considers these priorities and it is responsible for assigning the task. The available algorithms like Just in time will not consider user priority. A better scheduling algorithm is needed to achieve full utilization of resources. Just in time algorithm consider the task length as the crumb for executing the process. An dynamic workflow scheduling allocate and execute the process with minimum length first. But on the other side heavy weighted processes are executed by workflow scheduling in order of process may change which may be different from the order when the process length is considered. The main idea behind the proposed approach is considering both the process size, bandwidth and available resources. Based on these three criteria, order of execution of the process is considered in our proposed model for improving the performance of the cloud environment.

The proposed model enhances the dynamic resource management using workflow application in cloud environment which are evaluated and compared with existing models by simulation. And it is proved that the proposed DRS increase the resource utilization and reduce waiting time in a cloud environment.

Literature review: Nagpure *et al.* (2015) implemented a dynamic resource allocation system that avoid overload in server effectively by allocating resource evenly among VMs. The capacity of a physical machine should be sufficient so that it satisfy the resource needs of all virtual machines. Otherwise, physical machine is overloaded and it can decreases performance of virtual machines. We used the concept of skewness to calculate the uneven utilization of multiple resources on the server among VMs and checks available server resource and predict future load to avoid overload on server.

Wang *et al.* (2015) proposed an economic method and bio-inspired algorithm, an intelligent combinatorial double auction based dynamic resource allocation approach is proposed for cloud services. The system framework is devised to provide a comprehensive solution. A reputation system is used to suppress dishonest participants. A price formation mechanism is proposed to predict price and determine eligible transaction relationship. Winner Determination Problem (WDP) is optimally solved by the improved Paddy Field Algorithm (PFA). Simulation results validate the effectiveness of our proposed approach and demonstrate its superiority on economic efficiency and trustfulness.

Lu Di, MA Jianfeng, Xi ning proposed a framework for dynamic evaluation of fairness based on dominant share. Aiming to the dynamic resource demand and computing node in cloud computing, we introduce time and probability factor to establish two sub-models: Dynamic Demand Model (DDM) and Dynamic Node Model (DNM) (Lu *et al*., 2015).

Fu *et al.* (2015) describe a novel dynamic resource scheduler for real-time streaming analytics in a cloud-based. Dynamic Resource Scheduling (DRS) overcomes several fundamental challenges, including the estimation of the required resources necessary for satisfying real-time requirements, effective and efficient resource provisioning and scheduling and the efficient implementation of such a scheduler in a cloud-based. The performance model of DRS is based on rigorous queuing theory, and it demonstrates robust performance even when the underlying conditions of the theory are not fully satisfied.

Fakhfakh *et al.* (2015) proposed algorithm enables to determine the resource amount to execute a workflow on IaaS cloud. It aims to minimize the execution cost while meeting a user defined deadline. To further reduce the cost, we used an adjustment process which takes into account the wasted time fractions. Then, we have presented a first step towards the provisioning for a dynamic workflow by addressing only the case of adding and deleting tasks at runtime.

Rahman *et al.* (2013) defined the workflow scheduling problem in grid computing environment and discussed the existing well-known workflow scheduling techniques. Nevertheless, these techniques are static in nature and do not take into account dynamic resource behavior. Therefore, we have proposed a dynamic and adaptive scheduling approach, named Dynamic Critical Path for Grid (DCP-G) for scheduling grid workflows. DCP-G determines an efficient mapping of workflow tasks to grid resources by calculating the CP in the workflow task graph at every step and assigns priority to a task in the CP that is estimated to complete earlier. We have compared the performance of DCP-G with other existing heuristic-based and meta heuristic-based scheduling strategies for different types and sizes of workflows. The results show that DCP-G can generate better schedule for most of the workflow types, irrespective of their sizes particularly when the resource availability changes frequently. To the best of our knowledge, only few research works have proposed for Resource allocation and workflow scheduling in cloud environment. The proposed model suggest a Dynamic Resource allocation, which provides workflow application for process scheduling which result, less failure rate, less cost, waiting time of process is reduced and increase the utilization of resource. The prime focus of this study is on

developing a Dynamic Resource Scheduling for various workflow applications of user process as well as protocols in distributed cloud platform.

MATERIALS AND METHODS

Proposed model: In this study, we present our proposed Process Size based Dynamic Scheduling model along with the system architecture and design model of resource management for maximum utilization, reduce in waiting time and cost strategy in cloud environment. The overall architecture of Process Size based Dynamic Scheduling (PSDS) model is shown in Fig. 1.

The cloud user move their data's from on-premise to cloud-based data centers and it is accessed by mobile devices. In that scenario, challenges arise from data residency, accessing mechanism, mobility, bandwidth restrictions, number of wireless and wired access, number of communication messages and privacy in order to protect sensitive information. The primary solutions for these problems include encryption of data stored in the cloud.

The conventional scheduling module in cloud computing is a mapping mechanism from users' tasks to the appropriate selection of resources and its execution. In that, we find out the challenges to spread the load on processors and maximize their utilization, minimize the execution time without affecting cloud service. And we find out the proper sequence in which the process is executed under our algorithm to meet and overcome the challenges.

Out of the conventional existing scheduling algorithm namely, first come first server, shortest job first, priority scheduling, greedy approach and round Robin algorithm we proposed two algorithm in our study for sequence and Workflow Scheduling for Resource Management namely Just-Sequence-in-Time (JST) and Process size based Dynamic Scheduling Algorithm (PSDS) which perform the process management and Dynamic Resource Management to obtain the maximum resource utilization and less waiting time of the process.

The rest of the section describe about of these two algorithms of process scheduling and resource management.

The process Scheduling compares with the credentials of suitable resources and allocation of the tasks on those resources. It decides the relocation of process from a heavily loaded Virtual Machine (VM) or idle VM or least loaded VM to allocate the current process to the available resource. In process size based dynamic scheduling, the heavy weight processes are divided into different-different small co-process, these co-process or sub processes are allocated to available resources in such a way that achieve some pre-defines objective.

Process size based dynamic scheduling design: The various modules of the Process size based Dynamic scheduling are discussed below:

Resource pool: The resource pool is the existing available n number of Data Center's in a cloud environment. The each data center has m number of Virtual Machine's, in which all Virtual Machines of a particular data center are integrated with computation resources, network resources and storage resources. The cloud environment with their configured data centers and VMs forms the set of resource available for process computing. The available resources are maximum utilized for a heavy weight cloud user process are effectively allocated in the resource are managed.

- Compute resources: It is typically collection of Physical Machines (Pms) which all together provide the computational capacity of a cloud environment for end user
- Networking resources: The physical Machines within a data center are packaged as clusters of thousands of nodes for resource allocation purposes
- Storage resources: public cloud providers provides persistent storage services of different types, ranging from virtual disks to storage of different accessing speeds

Workflow interface: This interface is integrated and controlled by process management with cloud user process along with their workload and available resource management, finally intimate the notification of the process and resource management. The interface has the sub models are discussed below:

Resource available: From the resource pool the control are received based on their available resource in cloud environment in the form of index table which intimate to process management.

Workload analyzer: This analyzer analysis all the cloud user's process requests in the order of their arrival with their arrival time along with the weight of the process are mapped and it will be moved to process management.

Process management: This module receives index table of the resource available and mapped table of the cloud

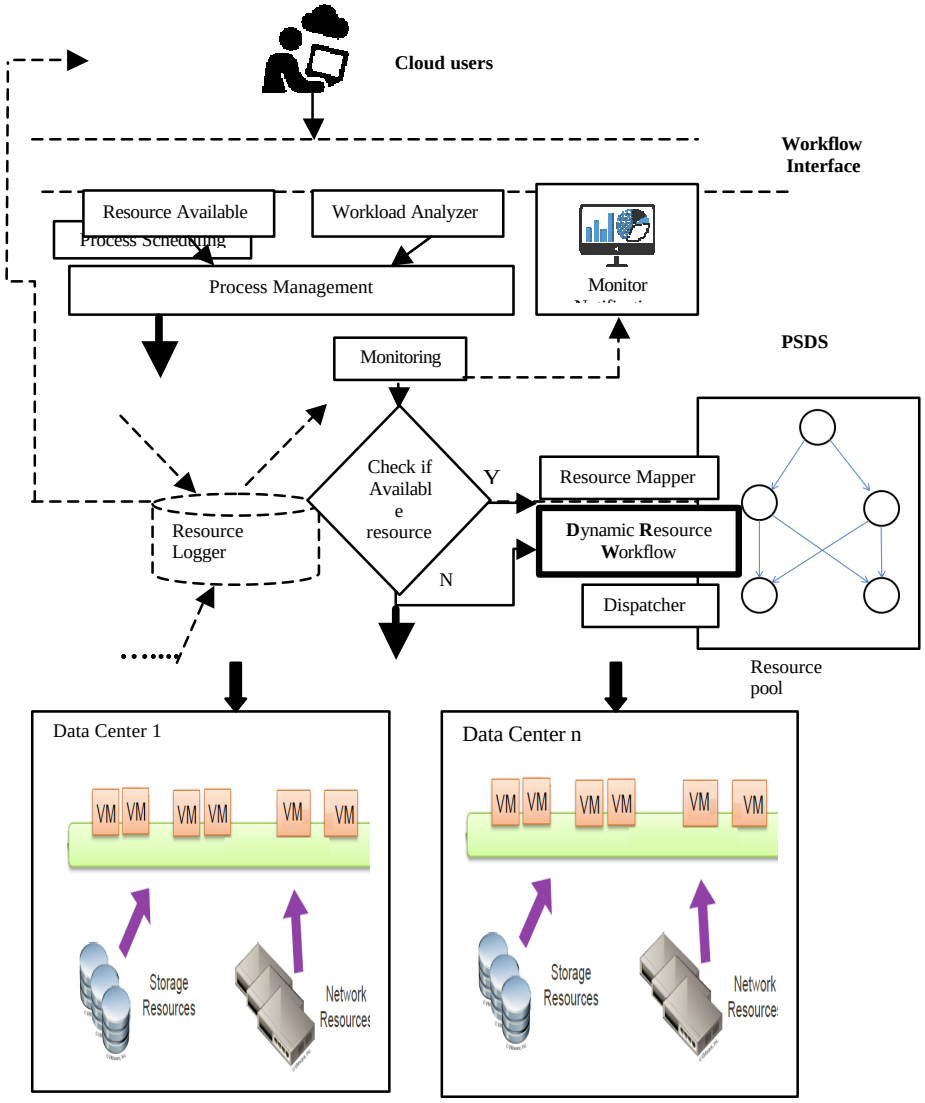

Fig. 1: Architecture of process size based dynamic scheduling

user's process and verifies the process arrival time and weight of the process whether it is a light weight or heavy weight. Further, it checks whether the processes are fully independent or inter dependent between the multiple processes. If it is fully independent process it is directly moved to process scheduling algorithm with process mapped table which contain the arrival time and weight. If it is multiple process then it verifies the inter dependency process within the multiple process and assigned to process scheduling algorithm in the form of map table which contain arrival time, weight and inter link of the process of parent and child process are executed.

Monitor notification: This notification intimated the process management of all cloud user's request process and status of the execution.

PSDS model

Process scheduling: The Process scheduler selects the appropriate VMs based on the algorithms. This scheduler collects the process and resources information through the process management, which calculates the processing, based on process management map table and has a log in resource logger data base used for monitoring.

Monitoring: Resources monitor is part of process management and it communicates with all the resources

and collects the VM capabilities, current load on each VM, number of processes in execution/waiting/explode in each VM through resource pool. Now this control check the available of resources based on the map table and it assign the process to VM based on derived algorithms for single independent or light weighted process through Resource Mapper. In case the process is heavy weighted or multiple process or inter dependent process the Process size based Dynamic scheduling algorithm assigns the process to VMs through dispatcher.

Resource logger DB: This is resource cache data base, maintain the log details of the process management and resource pool of a process which store and provides the information to process scheduling.

Resource mapper: The process management map table provides the information of weight and dependency of the process. The check condition on monitor module is success for single independent or light weighted process and the process is submitted to VMs through dispatcher.

Dispatcher: This module dispatch the allotted process to VMs from the resource mapper and process size based dynamic scheduling module to the resource pool for utilizes the maximum resource, and reduce the waiting time.

Process size based dynamic scheduling: The dynamic resource management based on work flow scheduling performs with two distinct algorithm namely Just-Sequence-in-Time (JST) and Process size based Dynamic Scheduling Algorithm (PSDS). The JST algorithm proposed is an improved algorithm over the existing Just In Time Scheduler algorithm. It consider the map table generated by process management based on that the algorithm works with coordination of resource mapper and dispatcher.

Algorithm 1: Just-sequence-in-time scheduler

For each root process $p_i \in P_{roots}$ do
Assign p_i to an available resource r_n
end for
repeat
For all $p_i \in P_{(1..n)nonroots}$ do
Assign ready process p_i to any available resource r_n
end for
Dispatch all the mapped process
Update the ready process list
Until there are independent processes in the map table

The resources of Virtual Machine in resource pool are dynamic allocated based on allotted or limit size share based on resource demand is computed as its actual resource consumption, resource allotted , plus a scaled portion of resource ready , the time it was ready to execute:

$$\text{Resource}_{demand} = \text{Resource}_{alloted} + \frac{\text{Resourcer}_{run}}{\text{Resource}_{run} + \text{Resource}_{sleep}} \times \text{Resource}_{ready} \quad (1)$$

The resource demand values have been computed, they are aggregated up the resource workflow tree. Demand values are updated to resource pool, it always be no less than the capacity and no more than the limit value. Thus, at each resource, the following adjustments are made:

$$\text{demand} = \text{MAX}(\text{demand}, \text{capacity}) \quad (2)$$

$$\text{demand} = \text{MIN}(\text{demand}, \text{limit}) \quad (3)$$

The resource is to assign dynamically arriving dependent/independent process in data centers to reduce response latency and maximize resource utilization. Let us consider there are n number of VMs and m number of process. The set of all VMs are represented as r_i where i varies from 1 to n and the set of all process requests are represented as p_j where j varies from 1 to m. The Processing time of all process in a resource r_i can be defined as:

$$r_i = \sum_{j=1}^{m} pt_{i,j} \quad (4)$$

where, pt_{ij} is the processing time of j^{th} process p_j on i^{th} resource r_i. The factor gives the minimum time of the resources.

The processing capacity of each VMs in data center's are obtained from resource demand from resource pool. The execution of allotted process in resource by Dispatcher calculates by resource demand and start and finished time. The process complete executions based on independent or inter dependent process. This shows how effectively resources are being allocated and executed. It is represented as follows:

$$\sum_{i=1}^{T} \sum_{j=1}^{ALL\ VMS} \left(\frac{\text{Satisfied demand of resource at time t}}{\text{Cluseter capacity at t}} \right) \times 100 \quad (5)$$

Here, T denotes the total simulation time. If all the VMs in data centers have equal shares and have no

capacity or limit, a higher number is better. Furthermore, this value is normalized by dividing the area by sum of capacity of the entire host's time's length of the experiment. The factor is the CPU payload thus captures the total number of CPU cycles in the cluster. The factor also shows how the resources are effectively utilized.

Algorithm 2: Process Size based Dynamic Scheduling (PSDS)

Construct a tree structure for all available resources
Get N resources $(R)_N$ with highest weight w
For each process p_i starting from the root do
Set minimum Start Time (minST)=8
For each resource $r_i \in (R)_N$ do
Probe each connected neighbor $r^{neighbor}{}_i$ of r_i for calculating instantaneous bandwidth (max of w probes)
Split input process based on probe values: $\{f^{fsplit-1}, \bullet\bullet\bullet, f^{fsplit-w}\}p_i$?$\{f\}p_i$
Estimate total workflow time $t_t(\{f\}p_i, r_i)$ for interdependent or multiple process, the split process $\{fsplit\}p_i$ from each $r^{neighbor}{}_i$ to r_i
StartTime(ST)=$ST(p_i,r_i)+t_t(\{f\}p_i,r_i)$
If(ST=minST) then
MinST=ST,minR=r_i
End if
End for
Assign p_i to the minR ?the r_i that gives minimumST
Wait for polling time
Update the ready process in db
Distribute output data of process p_i to resources that host the process required by successors of p_i
End for

Here, the workflow scheduling problem with all of its constraints is discussed. Information symbolized in this section are categorized in four different groups:

- Work flow info (number of process, process dependencies, etc.)
- Process info (deadline, input output process size, etc.)
- VM info (number of VM, VM type, transfer cost, etc.)
- Overall algorithm info (utilize, time, process to VM assignment, etc.)

There is a workflow of process along with their dependency unknown in advance. We know the deadline of each process. The approximate size of input process and output process of each process is also known approximately. In each step, it can be determined whether the input file of each process is ready or not. There are a set of reserved and on-demand cloud VMs available and the cost of execution of each process on each VMs is known according to some calculations and historical data. By on-demand VM we mean the maximum allowable VM that can be applied to workflow. We introduce an algorithm which assigns process to VMs, in a way that the weight and cost(deadline miss, execution and transfer times) is minimized. In this approach, the workflow application is modeled as a Directed Acyclic Graph (DAG). Let p be the finite set of process p_i . Let A be the set of directed arcs of the form (p_i, p_j) where p_i is called a parent task of p_j and p_j is the child task of p_i . The problem assumes that a child task cannot be executed until all of its parent tasks have been completed.

The weight of the process is assumes that the cloud data center varies depending on the cloud user requests and the type of the process requested. The weight threshold values Th1 and Th2 which are dynamically calculated.

$$\text{if } w(t) < Th1(t)$$

- Weights are assigned to all the tasks, with sequence as the preference

$$\text{else if } w(t) >= Th1(t) \text{ and } w(t) <= Th2(t)$$

- Weights are assigned to all the tasks, with process lengths as the preference
- Weights are assigned to all the tasks, with waiting time as the preference

Where w(t) is a weight of the process p at time 't' dynamically.

RESULTS AND DISCUSSION

Performance analysis: We analyze the performance of PSDS by comparing it with the other models proposed in related work. Most of the existing methods are not properly use the resource utilization. The performance metrics employed are the average response time of process requests, the number of aborts of process, and the number of allotted resource to the resource database log and workflow scheduling for dependent and inter dependency process with weight of process for the final validation. The performance was measured by a simulation by varying the number of process requests with various resources. The scheme also presents a simulation study about the potential benefit in a service provider when doing workflow level. The rate of annual growth due to resource management, the frequency of process execution due to performance issues and rate of workflow scheduling time dynamically are also presented through simulation data. To simulate this we carried out in Cloud Analyst simulator tool.

The parameters we have take for Process size based dynamic scheduling model in cloud analyst simulator, this simulation is performed using CPU core i3, memory 4GB and 4MB cache memory. The results have been obtained with a system simulator program setting are shown in Table 1.

Table 1: Cloud analyst parameters

Entity type cloud	Process parameter	Value of resource
Process of cloud analyst	Length of process	1-1000
	Total number of process	100
	Weight of Process	High, Medium and Low
Data center		
	Number of Datacenter	1
	Number of Hosts	2-4
	VMM and VMs Scheduler	Xen and Resource shared, Time shared and multiple threshold
	Operating System	Linux
	Architecture	X86
Virtual Machine		
	Number of VMs	10
	MIPS	512-2048
	VM Memory	512-2048
	Bandwidth	500-1000
	Cloud analyst scheduler	Resources shared and Time shared
	Number of processor equipments	1-4

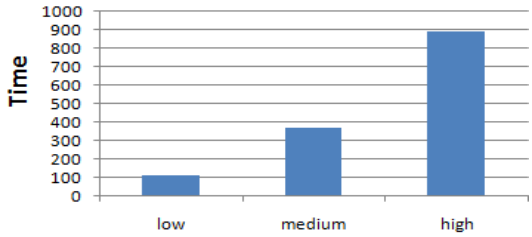

Fig. 2: Waiting time for different weighted process

Fig. 3: Heavy weighted process waiting time

Figure 2 shows the average waiting times of low weighted process, medium weighted process and high weighted process. The process with low weighted and medium process has less waiting time than the process with high weighted process. The three categories of processes base on their weight is classified based on the waiting time as follows:

- Low weight process:Waiting time <100 units
- Medium weight process:Waiting time in the range of 100-600 units
- Low weight process:Waiting time in the range of 500-1000 units

Table 2: Multiple Process Planning and execution flow

Level	#Process	Avg. Process Exec. Time	Planned Time	Planned cost	Assigned Vms no. of process
L1	2	20	8	80	DC1 A(2)
L2	2	10	2	45	DC1 A(1), B(1)
L3	1	25	4	50	DC1 A(1)

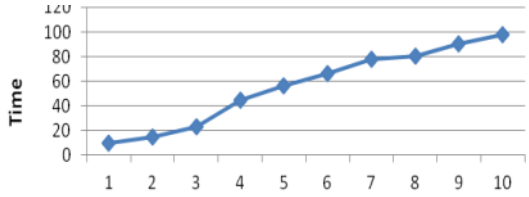

Fig. 4: Resource utilization

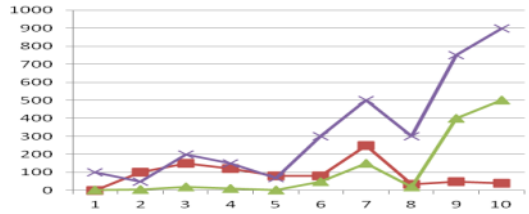

Fig. 5: Overall performance

The following graph show the waiting time of heavy weighted processes which have the variation because of the change in dynamic workflow scheduling algorithm (Fig. 3).

From the graph, it is seen that the waiting time is high compared to low weighted process. There is a variation because of the change in the scheduling policy then the assignments of tasks to VMs change dynamically whenever the actual execution time differs from the estimated one. The figure 4 shows the overall resource utilization. It can be seen from the graph that the resource utilization is increasing stepwise as the number of data centers increases.

To illustrate the operation of Process size based Dynamic scheduling algorithm, for multiple process or inter dependency process the following Table 2 illustrate the multi level process with heavy weighted or multiple process. The assignments of tasks to VMs change whenever the actual execution time differs from the estimated one.

Figure 5 shows the overall performance of our system. It demonstrates that the waiting time of submitted processes are reduced and resource utilization is enhanced depending upon the process weight.

CONCLUSION

In this study, a process size based dynamic scheduling algorithm on cloud environments was

introduced. The algorithm considered resource utilization and waiting time is reduce are the major quality of services parameters in scheduling. The proposed design benefitting from existing scheduling algorithm advantages. Our experimental result shows about 85% resource utilization, in comparison to existing implementations, 90% of waiting time reduce comparison to non-dynamic workflow scheduling model and 30% cost reduction in comparisonm to existing gravitational search algorithm. One of the improvement needed in future works is finding an algorithm to choose number of reserved VMs according to historical data in those points of view are types of cloud environment, types of cloud workflow application, cloud workflow scheduling algorithms, dynamic resource allocation and constraints are objectives of cloud workflow scheduling. Finally, we introduced a novel dynamic resource management system using Process size based Dynamic scheduling, taking into consideration, followed by challenges accomplishing the system.

REFERENCES

Fakhfakh, F., H.H. Kacem and A.H. Kacem, 2014. Workflow scheduling in cloud computing: A survey. Proceedings of the 2014 IEEE 18th International Conference on Enterprise Distributed Object Computing and Demonstrations, September 1-2, 2014, IEEE, Sfax, Tunisia, ISBN: 978-1-4799-5467-4, pp: 372-378.

Fakhfakh, F., H.H. Kacem and A.H. Kacem, 2015. A provisioning approach of cloud resources for dynamic workflows. Proceedings of the 2015 IEEE 8th International Conference on Cloud Computing, June 27- July 2, 2015, IEEE, Sfax, Tunisia, ISBN: 978-1-4673-7287-9, pp: 469-476.

Fu, T.Z., J. Ding, R.T. Ma, M. Winslett and Y. Yang *et al*., 2015. DRS: Dynamic resource scheduling for real-time analytics over fast streams. Proceedings of the 2015 IEEE 35th International Conference on Distributed Computing Systems (ICDCS), June 29-July 2, 2015, IEEE, Guangzhou, China, ISBN: 978-1-4673-7214-5, pp: 411-420.

Lu, D., J. Ma and N. Xi, 2015. A universal fairness evaluation framework for resource allocation in cloud computing. China Commun., 12: 113-122.

Nagpure, M.B., P. Dahiwale and P. Marbate, 2015. An efficient dynamic resource allocation strategy for vm environment in cloud. Proceedings of the 2015 International Conference on Pervasive Computing (ICPC), January 8-10, 2015, IEEE, Nagpur, India, ISBN:978-1-4799-6272-3, pp: 1-5.

Rahman, M., R. Hassan, R. Ranjan and R. Buyya, 2013. Adaptive workflow scheduling for dynamic grid and cloud computing environment. Concurrency Comput. Pract. Experience, 25: 1816-1842.

Wang, X., X. Wang, H. Che, K. Li and M. Huang *et al*., 2015. An intelligent economic approach for dynamic resource allocation in cloud services. IEEE. Trans. Cloud Comput., 3: 275-289.