ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(28), DOI: 10.17485/ijst/2016/v9i28/96635, July 2016* ISSN (Online) : 0974-5645

Two Objectives Big Data task Scheduling using Swarm Intelligence in Cloud Computing

Laouratou Diallo*, Aisha-Hassan A. Hashim, Rashidah Funke Olanrewaju, Shayla Islam and Abdullah Ahmad Zarir

Faculty of Engineering, International Islamic University Malaysia, Jalan Gombak - 53100, Kuala Lumpur, Malaysia; laouratoulelouma@gmail.com

Abstract

Cloud computing is the latest and the most used type of distributed computing systems and also it covers most of their features. It has been widely used for its enormous benefits and its ability to cope with large scale data such as workflows and big data applications. On the other hand, scheduling algorithms; starting from traditional to Hyper-heuristic; are widely used in computing systems such as cloud computing to monitor the use of resources. However, these scheduling algorithms vary in term of their performance and most of these traditional and simple scheduling algorithms may not be efficient for large scale data. Although many scheduling algorithms have been implemented for cloud computing, it has been realized that most of the applications nowadays require different objectives that simple scheduling algorithms fail to achieve. Either one of the objective is violated or the results are far from the optimal solution. In this direction, this paper first gives review of some previous scheduling algorithms used in cloud. Then, it proposes a type of swarm intelligence called Particle Swarm Optimization (PSO) algorithm to diminish cost though meeting deadlines. The proposed method is evaluated using CloudSim and big data applications are used as sample of applications. From the results, it can be seen that PSO works better for big data applications and the cost is reduced to more than half when compared with ordinary scheduling algorithms such as First-Come-First-Serve (FCFS).

Keywords: Cloud Computing, Hadoop and Big Data, Scheduling, Swarm Optimization

1. Introduction

With the recent fast growth of processing methods as well as stowing technologies, computing resources become affordable, more prevailing as well as more ubiquitously accessible than ever earlier. This powerful technological trend has led to the realization and exploration of an innovative computing model named as Cloud Computing (CC), where properties such as CPU as well as storage are conveyed as general utilities which can be used on-demand by users via the Internet. In a CC environment, the starring role of service provider is in two levels: firstly, the infrastructure providers who lease resources to users to develop their own applications conferring to a pay-as-you-go model. Secondly, service providers, who charge resources from some infrastructure providers to assist the end users. Additionally, some advantages of using cloud computing are low cost computing, massive storage capacity, backup, recovery, scaling up or down based on the demands. These services are achieved through the virtualization platform of cloud computing. Cloud computing and mobile computing both transmit data through wireless systems. However, cloud computing can be in distributed networks and users have control over the networks compared to mobile computing. Furthermore, the growth of cloud computing has contributed enormously on the development of Information Technology (IT) industry recently, where big cloud companies such as Google, Microsoft and Amazon are competing to offer reliable as well as cost-efficient cloud platforms. Most of business enterprises are trying to take advantages from this innovative paradigm. In fact, cloud computing put together many captivating features that push it to be more attractive to business, and the pay-as-you-go model has been the key contribution of its popularity. For example, some features are listed below.

* *Author for correspondence*

No need for prior investment: Since cloud computing adopts the pay-as-you-go pricing model. There is no need for a provider to make an investment in the infrastructure to begin using the resources from cloud computing. The resources can be simply leased from the cloud based on the requirement of the service.

Lowering operating cost: In a cloud environment, resources can be easily assigned based on demand. As a consequent, a service provider does not need to reserve capacities even for peak load. This lead to a huge savings, as resources can be rented immediately and automatically according to the exact demand at the particular time.

Highly scalable: Infrastructure providers give access of large amount of resources from data centers and make them easily accessible. Services can be easily scalable as many applications need large scales due to the fast growth of data recently. This is one of the convenience use of cloud computing[1].

Easy access: Services stored in the cloud are most of the cases web-based and are easily through the internet. Therefore, same contents can be accessed in different devices. This brings the flexibility to users since data are stored in the cloud. In addition, most of communication devices can be used to access the contents by users which include portable devices or not.

Decreasing the risks for business and reducing expenses for the maintenance purpose: By keeping data in the cloud, a service provider manages to minimize the risks of hardware failures since infrastructure providers, are more equipped to manage risks almost impossible to lost data. In addition, in cloud, maintenance service costs can be avoided since the subscribers use the services for their needs[2].

The main deployment models in cloud computing are Public Cloud where cloud vendors give services to customers , Private cloud mostly specific to a single organization and Hybrid cloud which is the combination of public and private cloud. In addition, the services deployed in cloud computing are Software as a Service (SaaS) such as Google Docs and Gmail, Platform as a Service (PaaS) such as Google's Apps Engine, Microsoft Azure and Infrastructure as a Service (IaaS) such as Amazon EC2 and Flexiscale. The focus of this paper is on IaaS clouds where users are given a virtual pool of unlimited resources to develop their applications. Therefore, Users can lease resources for what exactly their applications need. Further, this gives the flexibility of elastically leasing and releasing resources with different configurations to best suit the requirements of their applications. Despite the fact that this offers the users the power to control the resources, it also requires an effective elaboration of scheduling techniques for the efficiency of the resources utilized[3].

With the rise of internet of things, social networks, and GPS, petabyte of data are generated repeatedly through the use of Internet. As a result, batch of data need to be processed and analyzed continuously. Therefore, cloud computing is used as a platform for these huge data in order to benefit both low computational cost and satisfaction of users' requirement.

Although cloud computing systems[4] with its wide resources provide scheduling algorithm techniques to perform tasks in the simplest manner concerning responsiveness, scalability, as well as flexibility. Most of those scheduling algorithms are unsuitable for large-scale scientific difficulties for their lack of resources optimization. This means that there is a need to come out with different alternatives of scheduling algorithms for cloud computing systems.

A promising and successful type of scheduling algorithm is the heuristic methods that have attracted several researchers and scientists for their ability to find the optimal solution in a reasonable time while the objectives and the QoS of user are met. Further, heuristic algorithms are proposed in many recent research papers whether in a simple heuristic form, meta-heuristics or hyper-heuristic algorithms.

This paper is a continuous to our previous work where we developed an architecture and suggested a meta-heuristic method for our future work. In this paper, we develop a meta-heuristic scheduling algorithm for large-scale applications in a cloud environment. In addition, our approach uses all features of cloud computing. To achieve this, an analytical model for specifying the users' objectives and a meta-heuristic algorithm to support meeting deadlines and minimizing the cost are elaborated. The main contributions of this paper are summarized as follows:

A comprehensive review of scheduling algorithms is elaborated in order to give strong support to the use of heuristic algorithms;

An analytical model of the users' requirement and definition of their objectives considering deadlines and cost as the two objectives given;

A meta-heuristic algorithm with maximum accuracy in terms of meeting deadlines at smaller costs;

An evaluation of its performance on the cloud simulator CloudSim;

A brief analysis on the performance of the proposed algorithm.

The rest of this paper is organized as follow. Section 2 presents a brief overview to scheduling on cloud computing followed by discussion and comparison on scheduling methods along with their respective related work. Section 3 explains our analytical model and the proposed algorithm is presented in section 4. Section 5 gives the simulation results on CloudSim. Conclusion with future recommendation are shown in Section 6.

2. Scheduling

Cloud possesses a plenty of resources for computation, and the resources are available to be used in a pay as you go manner. Further, users can access cloud resources remotely. Therefore, Scheduling becomes a highly considerable parameter since a lack of appropriate scheduling will result to the failure of using all potential of the cloud system. The two main types of scheduling are mainly static and dynamic respectively.

The scheduling problem on cloud systems can be classified into the workflow problem as well as large-scale scientific problems. The workflow problem is further divided into two levels: service-level (platform layer and static scheduling) and task-level (unified resource layer and dynamic scheduling)[5]. In addition, the main requirements of the above data type are finding an optimal solution for a given set of tasks and meeting the deadlines and other constraints set earlier by the users. As a result, scheduling algorithms are chosen to meet these requirements. In the next sections, types of scheduling are presented.

2.1 Static Scheduling

Static scheduling is where decisions such as task processing time, synchronization requirements, communication mapping resources to the tasks are created before the execution of the scheduling. Therefore, it takes into consideration the initial state of the cloud before execution[6,7]. Furthermore, one of the advantage of static scheduling is that classification of execution before running the program can be determined offline. Thus, the runtime scheduling overheads is minimized[8]. Various schedulers have been created mostly for small-scale scheduling by taking care of few basic decisions. Some static schedulers are First-In-First-Out (FIFO), Round Robin (RR), Shortest Job First (SJF), Longest Job First (LJF), Highest Priority First (HPF), and Fair Share. However, the disadvantage of these scheduling is that they provide simple scheduling methods that are not suitable for cloud computing.

On the other hand, in Hadoop, many static algorithms have been implemented such as The FIFO Default Scheduler of Hadoop developed for Facebook[28]. Next, Hadoop Fair Scheduler (HFS) is proposed to solve the problems experienced in FIFO[28] had designed the HFS with delay scheduling. In overall, Hadoop scheduler is used to schedule tasks in cloud environment, many huge companies which produce big data adopt Hadoop. However, scheduling algorithms used in Hadoop are static and do not provide any optimal solution.

2.2 Dynamic Scheduling

Dynamic scheduling algorithm is a type of scheduling where decisions can change during the execution. It may have complete information about the current active tasks. When new arrivals occur in the future after the process started; which are not known to the algorithm at the time of the beginning the current set; then the decision keeps changing over time. Mostly this kind of scheduling is required for real-time systems. For example, Earliest Deadline First (EDF) uses dynamic method to schedule tasks to resources. In[9], EDF is used for analyzing multiprocessor platforms. They proposed a global Limited Preemptive Earliest Deadline First (g-LP-EDF) scheduling to test it in multiprocessors while violating the real-time non pre-emptible function of EDF. However, EDF performed better for lower scheduling problems but not for large. In addition, it failed to minimize the cost.

2.3 Optimization Algorithms

Optimization algorithms are divided into exact algorithms and heuristic algorithms. They are detailed further in the next sections.

2.3.1 Exact Algorithms

Exact algorithms are guaranteed to find an optimal solution and to prove its optimality for every optimization problems. They are mostly efficient and able to find an

optimal solution with small or moderate sized instances. The finite time, however, may increases exponentially with the large instance sizes. This means that finding good solutions in a limited time is compensated by the guarantee of finding optimal solutions[10]. In this case, the alternative for large instances could be with the other type of optimization which is heuristic. Some exact algorithms are listed with their respective related work.

2.3.1.1 Dynamic Programming

Dynamic programming is a numerical method that is used to solve a problem as a set of series of decisions that are divided into some sub-problems. In[11], the authors proposed a dynamic programing algorithm to develop a scheduling policy for surgical schedule that could minimize the time between patient request and the appointment date. They modeled the problem using a Markov Decision Process and applied Dynamic Programming in the model. However,[12] stated that their model is not directly related to the scheduling problem faced by the hospitals. Therefore, this cannot decrease significantly the crowd.

2.3.1.2 Direct Tree Search Methods: Branch and Bound

The procedure of branch and bound scheduling algorithm is to partition scheduling problem into sub-problems based on the integer values and LP techniques recursively to restraint upper and lower bounds around the optimal solution until they find a promising results in an integer solution. However, for large scheduling problems, the finite time may increase exponentially. In addition, this type of algorithm is time consuming because the number of checks they have to perform is very large. Therefore, this cannot be the optimal candidate in big data applications. Besides that, it may fail to meet the deadline, while deadline is a crucial parameter after which the results would be useless since most those large scale applications are real-time[13].

In[14], Alrokayan et.al proposed in their work to schedule the task of big data applications using different types of branch and bound algorithms. They showed that the branch and bound Pruned tree algorithm (BBPruned) performed better than the other types. However, from their results shown, it can be seen that the branch as well as bound algorithm is not the optimal solution as the deadlines are violated. In addition, even it can be worse when we deal with larger instance sizes.

2.3.1.3 Critical Path or Critical Path Method (CPM)

The authors in[15] have used an exact algorithm for scheduling a single workflow based partial critical paths. They employed cloud computing features in their paper. Nevertheless, the solution is not efficient when more than a single workflow is used. In addition, the optimization technique is not global.

2.4 Integer Programming (IP), Linear Programming (LP) and IP/LP Algorithms

LP is a type of mathematical method used to solve the optimization problem with some objective functions applicable to both linear equality and inequality constraints[16]. IP, LP and mixed IP/LP algorithm are types of analytical algorithms that have been applied widely to solve scheduling optimization problems[17].

The authors in[18] proposed an admission control and scheduling algorithm, which can help to satisfy QoS requirements and to minimize the cost. The proposed scheduling algorithm is based on mixed Integer Linear Programming (ILP). The procedure is that to schedule tasks based on ILP for meeting deadlines and cost, and any task that may be affected by violation of QoS will be redirected to a second phase which is a Greedy Search Algorithm. However, in a case where large instances sizes, the mechanism can turn a worse case which does not satisfy the SLA of users and increase computational effort. Furthermore, this method of algorithm has a single objective as many mathematical algorithms.

To sum up, from the mentioned review, we notify that these algorithms are not capable to find the best solution in a rational time, particularly when the issue becomes too outsized. Therefore, with the huge data that need to be processed in cloud, we realize these scheduling algorithms are inefficient for cloud computing with large applications. As a result, [4] this brings the need to shift to other types of scheduling algorithms that are more efficient and effective, such as heuristics.

2.5 Heuristics Algorithms

The concept of heuristic is reduce the scheduling problem to a much simpler problem, then adding up step by step, solving the problem in consecutive. Thus, any new problem should inherit from its precedent and its solution is used as a key to solve the next one. In addition,

heuristics are in a form of traditional heuristics, meta-heuristics and hyper-heuristics. Next, we will elaborate these form of heuristics and present some of their related work.

2.6 Simulated Annealing (SA)

The approach of SA is to simulate the cooling of material in a heat bath, [19] a technique called as annealing. A solid material is frenzied, pasts its melting point and then cooled back. The final structure of properties relies on how the state of the material (cooling). Furthermore, the strength of SA is that it finds solutions that are better than any others at the current time. The famous example of SA is the travelling salesman problem, where SA is efficient in this case to find the minimum path.

2.7 Genetic Algorithms (GA)

A GA is a random searching algorithm founded on natural selection as well as survival of the rightest in the population called chromosomes. The three supreme significant phases involve in GA are mainly selection, crossover as well as mutation [17].

GA receives too much attention recently for scheduling large scales and it is a promising type of algorithm to be used in order to cope with tremendous increase of data and research issues. In addition, it can help managing efficiently computing resources.

More in line with our work is the solution developed in [20] that proposed a GA scheduling method for different locations in both computational resources and data storages. In the paper, the focus was on grouping the jobs based on the data requirements, the constraints were minimized including the total makes pan while computational resources and the bandwidth are considered as well. Their results showed that the proposed method performed better in replicated sites compared to single site. In addition, the scheduling showed better performance in family groups rather than non-family, as long as the grouped jobs do not exceed the capacity. However, the proposed algorithm did not offer solution for deadline and budget constraints. In addition, group scheduling on the Hadoop cluster may increase the utilization as well.

Furthermore, authors in [21] proposed a GA for task migration. In addition, [22] proposed an improved Genetic Algorithms (IGA). The latter used a "super-scheduler" and added it to the FCFS concept. The optimization objective in the paper was to exploit the resource consumption of the physical machines used as an impartial to reduce resources. The results showed that the IGA performed better than the traditional GA scheduling technique in Grid environment as well as the exploitation is higher. However, this work focused more into energy consumption and deadlines of users are not considered in this case.

2.8 Particle Swarm Optimization (PSO)

Particle swarm is a population-based stochastic algorithm for optimization that uses the concept of social–psychological principles. Contrary to other heuristic algorithms, the PSO does not use selection; rather all population members survive from the beginning of a process until the end. The interaction among the population members produces an iterative improvement of the quality of problem solutions over time[23].

Rodriguez and Buyya[3] proposed an algorithm based on the meta-heuristic optimization technique particle swarm optimization (PSO) to satisfy the parameters such as the deadline and the cost. CloudSim and other techniques are used for evaluation to compare the efficiency of proposed algorithm. Yet, the solution attained does not provide a global peak. In addition, the movement of the population could affect the competency of the algorithm for searching efficient solutions.

More recently, many researches have been done using PSO. Amongst them, [24-27]. This algorithm will be detailed more in the next chapter as our proposed algorithm is based on PSO concept.

2.9 Hyper-Heuristic Scheduling Algorithm

It a combination of two or more heuristic algorithms. It is proposed in many of the recent research work. To illustrate, authors in[4] presented a hybrid scheduling algorithm, named as Hyper-Heuristic Scheduling Algorithm (HHSA), to discover better scheduling resolutions for cloud computing systems compared to traditional single heuristic algorithms. HHSA integrated several heuristic algorithms to be a single heuristic algorithm to increase the optimal solution. From the outcomes, also it is observed that the proposed algorithm performed well and gave better results in both traditional rule-based scheduling algorithms and heuristic scheduling algorithms, in finding solution for the workflow scheduling as well as Hadoop map-task scheduling on cloud computing

environments. Furthermore, HHSA is applied to improve the performance of scheduling problems faced in cloud computing environment. Nevertheless, the system is too complex as it is a combination of two or more heuristics.

3. Analytical Method of the Proposed System Model

In our previous work, in order to define the scheduling algorithm model, we developed an architecture based on Lambda concept[29]. It is a new model architecture for processing big data applications to tackle their complexity.

The system model was formulated into two main parts. The first part is user side where requests are sent from to be performed in the cloud. Once the tasks are sent to the engine (architecture) this will, in return, schedules the tasks, and assigns resources in a transparent fashion. Then, the second part is a cloud provider, where components in charge of provisioning and scheduling provide the necessary resources for processing while taking in consideration the deadlines and cost.

Furthermore, there are four main components that composed our architecture. However, we will focus more in VM scheduling Manager as it is the main focus of this paper. VM manager chooses Cloud resources such as type of VMs and also the pricing model. This module helps to allocate the suitable resource as elected through the task scheduler to avoid unnecessary usage of resources. To put more in details, once we have provisioned the tasks, we can make a query that identifies which resource in the cloud can process the present tasks before deadline. Further, this component will select the resources that minimize the cost. At any new set of data, this component adds or removes resources from the cloud to satisfy user requirements.

In addition, the scheduling manager services a scheduling algorithm to find a suitable Cloud resource for every task. In this paper, we develop this algorithm.

3.2 Mathematical Model and Problem Formulation

We formulate our model with a:

Cloud environment (C)

Heterogeneous VM types: medium, large, xlarge, doubleXlarge, 4xlarge and 10xlarge according to table 2. In addition, the mathematical notation of the formulas is given in table 1.

Each VM
- CPU
- Storage (S)
- Price (P)
- Operating System (OS)
- Memory (M)

VM= {time, VM type};

User Request (UR): SLA= {D, B};

The cost C_t requires for a given task is elaborated below

VM= {MIPS, VMcost, DTT};

DTT= size (data) /Bw; (1)

Texe = size (data)/MIPS (2)

We assume that Data transfer cost among VMs=0.

As a result, we can compute the total cost C_t by this formula below in (4)

Cost Ct, $$Ct = \sum_{1}^{n}(VM\cos t \times Texe) + \sum_{1}^{n}(DTT \times size(data)) \quad (3)$$

On the other hand, Budget B is provided by the user. Thus, to meet the budget requirement;

$B<C_t$; (4)

Furthermore, Makespan (M) is the total time needed to execute the user's request in a datacentre.

M= Tend- Ts; (5)

Where Tend= Texe + DTT; (6)

Deadline (D) the time constraint by when the job must be executed.

Therefore, D>M; (7)

This should be achieved to meet the deadline objective.

Table 1. Mathematical Notation

Name	Description
J	A specific job
Ts	Starting time of the task or submission time
Tend	Ending time of the task
N_{vm}	Number of VMs released for the particular batch or workflow.
C_t	The total cost of running a job
Cvm:	The available capacity of VM
D	the deadline of a job
B	The budget of a job
M	Makespan (total execution time of a job)
DTT	Data transfer time between tasks
DTvm	Data transfer cost among VMs
Bw	Bandwidth between VMs
T exe	Time to execute a task on a VM leased
Vmcost	Cost assigned by the provider to VMs
MIPS	Million instructions per second, known as the capacity processing of the VM
MI	Million instructions, knwn as the length of the task or size of the task

3.3. Objective Function

Deadline and budget being the two constraints. Our objective function of the scheduling problem is diminishing the implementation cost as well as gathering the deadline. Therefore, it is formulated as follow:

$f(S)= \{ M; C_t \};$

Minimizing C_t while considering M<D.

4. Proposed Algorithm

Recently, Cheng et al., [30] tried to give the importance of using swarm optimizations for big data analytics. Particle Swarm Optimization is the most prevalent meta-heuristic algorithm. Therefore, it is proposed to map available resources to the tasks.

4.1 PSO Concept

PSO is an evolutionary computational technique. A particle which is a member of the population moves via the problem space and could be a aspirant solution to the optimization issue. In addition, each particle is defined by its velocity and its position. The fitness function is utilized to regulate and update the particle best position *pbest* as well as of the global best position of the population *gbest* of the particles. The algorithm continues to iterate until a stopping criterion is met; and this is specified by the maximum number of iterations.

Algorithm 1 presents the pseudo-code of the PSO algorithm.

Algorithm 1: PSO

1. Set the population arbitrarily;
2. Compute the fitness function for each particle f;
2.1 Update the velocity and location of the particle grounded on "pbest" and "gbest" particles;
3. Repeat Step 2 till the closure principle met;
4. Output the outcome;

Pbest is the best position of the particle; *gbest* is the best position of the particle in the population.

4.2 Modeling PSO for the Proposed Approach

For the scheduling purpose, a particle represents a task in a Job. Based on the equations we developed earlier, PSO is linked to the schedule problem.

Algorithm 2 gives the mechanism to calculate the parameters of the objective function by the position of the particle. We describe the encoding in concordance with[3], the size of the location of the particle represents the total of tasks, and also the rate of each measurement signifies the resource to be used to execute this task.

Algorithm 2: PSO model in scheduling

Input: a Job J
Output: tasks in J that are allocated on VMs
1. Initialize
C_t=Ø, M=Ø
2. Calculate Texe for each task in the Job J according to formula (3)
3. Calculate the DTT according to formula (1)
4. Compute M according to equation (6)
5. Compute C_t according to equation (4)
6. Wait for the status of tasks scheduled
7. Update list of tasks
8. Update C_t
9. Record the current optimal solution
10. Until stopping criteria is met
11. Output the optimal value of C_t while maintain M not more than D.

5. Simulation Results

In this section, the simulation setup is presented and also the results analysis

5.1 Simulation Setup

CloudSim is used to simulate the proposed algorithm. It was extended to support big data based applications. In particular, it is worth to note that we assume that the service provider offers a computation service like Amazon Elastic Compute Cloud (EC2). The instances are illustrated in Table 2. We assume also the heterogeneity of the cloud, VMs are chosen randomly. Furthermore, MapReduce is chosen as workload with task length varying between 1000, 10000 and 1000000. Also, two different sizes of these applications are chosen; a small set of 30 tasks and large set of 100 tasks. Additionally, only the execution times are considered in the makespan of tasks. Therefore, it is assumed that any data transfers between jobs or within the tasks do not affect considerably scheduling decisions. This assumption can be supported by considering that the datacentre is robust and stable. In order to estimate the

deadlines, a term named deadline factor β is added and it is based on[15]. However, the interval of β is between 1.5 and 6 with a step length of 0.5.

Table 2. EC2 Instances and Pricing

Type	EC2 Units	vECU	Memory (GiB)	Storage (GB)	Cost/ hour
m3.medium	3	1	3.75	4 SSD	$0.13
m3.large	6.5	2	7.5	32 SSD	$0.25
m3.xlarge	13	4	15	40 SSD	$0.50
m3.2xlarge	26	8	30	80 SSD	$1.00
m4.4xlarge	53.5	16	64	EBS only	$1.90
m4.10xlarge	124.5	40	160	EBS only	$4.90

For the PSO parameters, the number of iteration is set to 500. Further, the number of particles in the simulation is set to 100 particles. Then, the experiments were executed for 15 times and in each round the means of the execution time and cost with the number of tasks are evaluated.

5.2 Results Analysis

The execution and Computed time of tasks in respect to the deadline factor are shown in Figure 1 and 2. Tasks were scheduled to meet the deadlines. The execution time was estimated to fall in the range between the slowest makespan and fastest makespan. The deadlines are divide into tight deadlines where β= [1.5-2], medium β=[2-3.5]

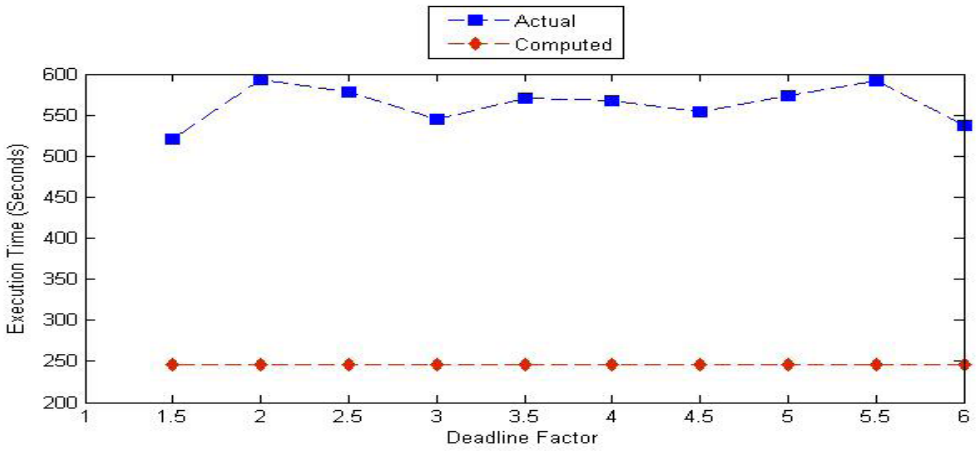

Figure 1. Actual and Computed Execution Time for 30 Map Tasks with the Same Task Length Shown According to Deadline Factor Value.

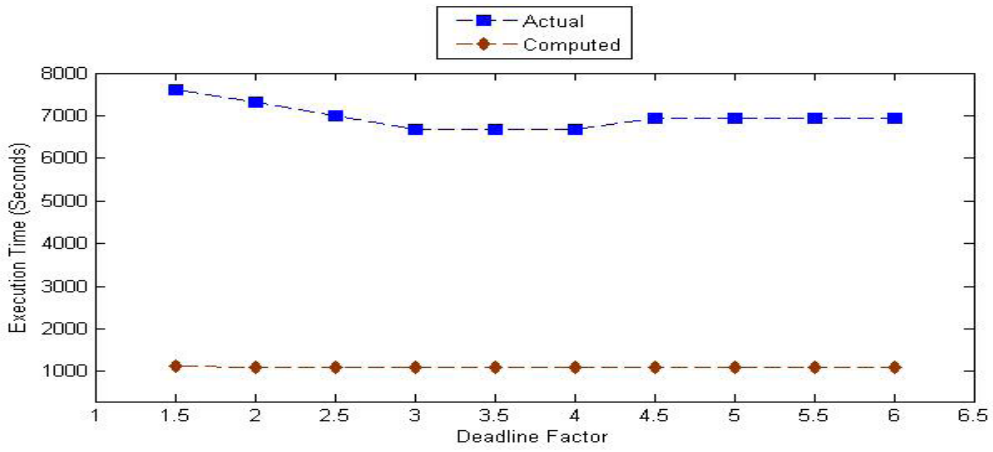

Figure 2. Actual and Computed Execution Time for 100 Map Tasks with Different Task Length Shown According to Deadline Factor Valueβ.

and loose deadlines β= [3.5-6]. In Figure 1, 30 tasks have been simulated with the same task length (1000 MI). The mean of the actual time was recorded and the Computed time is evaluated as well. Therefore, all tasks length have the same value, we noticed a constant value in both Figure 1 and Table 3. Furthermore, there was a variation of the actual time during the simulation. As for the case of large set of tasks (100 tasks), Figure 2 presents the values of the time. In this case, it can be seen that the values of computed time decreases as the deadline becomes looser. An illustration is shown in table 3. The same performance is almost seen in the actual time where the mean of the different runs is shown. Overall, from these results, small changes were noticed in the different range of β of the makespan. Therefore, this can be demonstrated that the cost was minimized even when the deadline was tight.

Table 3. Mean of the Estimated Makespan for Two Sets of Tasks

Deadline factor β	2	2.5	4.5	6
Tasks=30	246.2	246.2	246.2	246.2
Tasks=100	1112.86	1110.39	1109.12	1109.6

6. Conclusion and Future Directions

In this paper, the concept of cloud computing is elaborated and literature review on scheduling algorithms is also discussed. In addition, a scheduling algorithm for big data analytics based on PSO swarm intelligence technique is proposed. Next we presented an analytical model based on PSO along with the algorithm. We presented a summary of the results implementation of PSO in CloudSim.

In the future, the benchmark of the present work against the default First-Come-First-Serve (FCFS) implemented in CloudSim and other meta-heuristic algorithms such as Genetic Algorithms will be carried out it to see the performance.

7. References

1. Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Stoica I. Above the clouds: A Berkeley view of cloud computing. Dept Electrical Eng. and Comput Sciences, University of California, Berkeley, Rep. UCB/EECS. 2009; 28(13).
2. Zhang S, Zhang S, Chen X, Huo X. Cloud computing research and development trend. 2010 Second International Conference on In Future Networks, ICFN'10, Ieee. 2010 Jan; 93–7.
3. Rodriguez MA, Buyya R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE Transactions on Cloud Computing. 2014; 2(2):222–35.
4. Tsai CW, Huang WC, Chiang MH, Chiang MC, Yang CS. A hyper-heuristic scheduling algorithm for cloud. IEEE Transactions on Cloud Computing. 2014; 2(2):236–50.
5. Wu Z, Liu X, Ni Z, Yuan D, Yang Y. A market-oriented hierarchical scheduling strategy in cloud workflow systems. The J Supercomput. 2013; 63(1):256–93.
6. Kwok YK, Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Computing Surveys (CSUR). 1999; 31(4):406–71.
7. Arya LK, Verma A. Workflow scheduling algorithms in cloud environment-A survey. In Engineering and Computational Sciences (RAECS), 2014 Recent Advances, IEEE. 2014 Mar; 1–4.
8. Mohammadi A, Selim G. Akl. Technical Report No. 2005-499 Scheduling Algorithms for Real-Time Systems. 2005 Jul.
9. Thekkilakattil A, Baruah S, Dobrin R, Punnekkat S. The global limited preemptive earliest deadline first feasibility of sporadic real-time tasks. 2014 26th Euromicro Conference In Real-Time Systems (ECRTS), IEEE. 2014 Jul; 301–10.
10. Puchinger J, Raidl GR. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification, Springer Berlin Heidelberg. 2005; 41–53.
11. Astaraky D, Patrick J. A simulation based approximate dynamic programming approach to multi-class, multi-resource surgical scheduling. European Journal of Operational Research. 2015; 245(1):309–19.
12. Geranmayeh S. Optimizing surgical scheduling through integer programming and robust optimization. 2015.
13. Stavrinides GL, Karatza HD. A Cost-Effective and QoS-Aware Approach to Scheduling Real-Time Workflow Applications in PaaS and SaaS Clouds. 2015 3rd International Conference on Future Internet of Things and Cloud (Fi-Cloud), IEEE. 2015.
14. Alrokayan M, Vahid Dastjerdi A, Buyya R. SLA-aware Provisioning and Scheduling of Cloud Resources for Big Data Analytics. 2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), IEEE. 2014 Oct; 1–8.
15. Abrishami S, Naghibzadeh M, Epema DH. Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. Future Generation Computer Systems. 2013; 29(1):158–69.
16. Kantorovich LV. A new method of solving of some classes of extremal problems. In Dokl. Akad. Nauk SSSR. 1940; 28:211–4.
17. Zhou J, Love PE, Wang X, Teo KL, Irani Z. A review of methods and algorithms for optimizing construction scheduling. Journal of the Operational Research Society. 2013; 64(8):1091–105.
18. Zhao Y, Calheiros RN, Gange G, Ramamohanarao K, Buyya R. SLA-Based Resource Scheduling for Big Data Analytics

as a Service in Cloud Computing Environments. 2015 44th International Conference on Parallel Processing (ICPP), IEEE. 2015 Sep; 510–9.
19. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. The journal of chemical physics. 1953; 21(6):1087–92.
20. Kune R, Konugurthi PK, Agarwal A, Chillarige RR, Buyya R. Genetic Algorithm based Data-aware Group Scheduling for Big Data Clouds. 2014 IEEE/ACM International Symposium on Big Data Computing (BDC), IEEE. 2014 Dec; 96–104.
21. Zhang W, Tan S, Lu Q, Liu X, Gong W. A genetic-algorithm-based approach for task migration in pervasive clouds. International Journal of Distributed Sensor Networks. 2015; 14.
22. Zhong H, Tao K, Zhan, X. An approach to optimized resource scheduling algorithm for open-source cloud systems. 2010 Fifth Annual in China Grid Conference (ChinaGrid), IEEE. 2010 Jul; 124–9.
23. Kennedy J. Particle swarm optimization. In Encyclopedia of machine learning . Springer US. 2011; 760–6.
24. Wang X, Cao B, Hou C, Xiong L, Fan J. Scheduling Budget Constrained Cloud Workflows with Particle Swarm Optimization. 2015 IEEE Conference on Collaboration and Internet Computing (CIC), IEEE. 2015 Oct. p. 219–26.
25. Chen H, Guo W. Real-Time Task Scheduling Algorithm for Cloud Computing Based on Particle Swarm Optimization. Cloud Computing and Big Data. Springer International Publishing. 2015; 141–52.
26. Radha K, Rao BT. Slot Utilization and Performance Improvement in Hadoop Cluster. In Information Systems Design and Intelligent Applications. Springer India. 2016; 49–62.
27. Xu L, Qian F, Li Y, Li Q, Yang YW, Xu J. Resource allocation based on quantum particle swarm optimization and RBF neural network for overlay cognitive OFDM System. Neurocomputing. 2016; 173:1250–6.
28. Zaharia M, Borthakur D, Sen Sarma J, Elmeleegy K, Shenker S, Stoica I. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In Proceedings of the 5th European conference on Computer systems. ACM. 2010 Apr; 265–78.
29. Marz N, Warren J. Big Data: Principles and best practices of scalable realtime data systems. Manning Publications Co. 2015.
30. Cheng S, Zhang Q, Qin Q. Big data analytics with swarm intelligence. Industrial Management and Data Systems. 2016; 116(4).