Optimized Load Balancing based Task Scheduling in Cloud Environment

Elrasheed Ismail Sultan
Muscat College

Noraziah .A.
Faculty of CS and SE
Universiti Malaysia Phahang
Kuantan, Malaysia

Faisal Alamri
Faculty of CS and SE
Universiti Malaysia Phahang
Kuantan, Malaysia

Abdulla. A
Universiti Malaysia Phahang
Kuantan, Malaysia

ABSTRACT
Abstract. The fundamental issue of Task scheduling is one important factor to load balance between the virtual machines in a Cloud Computing network. However, the optimal broadcast methods which have been proposed so far focus only on cluster or grid environment. In this paper, task scheduling strategy based on load balancing Quantum Particles Swarm algorithm (BLQPSO) was proposed. The fitness function based minimizing the *makespan* and data transmission cost. In addition, the salient feature of this algorithm is to optimize node available throughput dynamically using MatLab10A software. Furthermore, the performance of proposed algorithm had been compared with existing PSO and shows their effectiveness in balancing the load.

Keywords
Cloud computing, scheduling, Load balancing, Storage System, Virtual machines.

1. INTRODUCTION
Cloud computing is emerging as the latest distributed computing paradigm and attracts increasing interests of researchers in the area of Distributed and Parallel Computing [1], Service Oriented Computing [2] and Software Engineering [3]. Generally speaking, the function of a cloud workload system and its role in a cloud computing environment, is to facilitate the automation of user submitted workload applications where the tasks have precedence relationships defined by graph-based modeling tools such as DAG (directed acyclic graph) and Petri Nets [4], or language-based modeling tools such as XPDL (XML Process Definition Language) [17].

Among many others, one of the most important aspects which differentiate a cloud workload system from its other counterparts is the market-oriented business model. Such a seemed small change actually brings significant innovations to conventional computing paradigms since they are usually based on non-business community models where resources are shared and free to be accessed by community members [5]. Meanwhile, application data can be hosted on different storage resources at the global cloud infrastructure. When one task needs to process data from different datacenters, moving the data becomes a challenge [6]. In order to efficiently and cost effectively schedule the tasks and data of applications among cloud services, end user QoS-based scheduling strategies are implemented, such as those for minimizing *makespan*, minimizing total execution cost and balancing the load of resources [7]. In this paper, we focus on minimizing the execution time and the execution cost of applications on these resources provided by Cloud service providers, such as Cisco and Amazon.

The particle swarm method for function optimization has been introduced by Kennedy and Eberhart [8]. The ability of groups of some species of animals to work as a whole in locating desirable positions in a given area is simulated. It has better ability of global searching and has been successfully applied to many areas [9]. This algorithm is predominately employed to find solutions for continuous problem without prior information. Unfortunately, workload scheduling which is one of a variety of NP-completes is a discrete and very complicated optimization issue. Several approaches have been developed for PSO to solve discrete problem, such as swap operation [10], angle modulation [11], space transformation [12] and priority-based representation [13]. Although various discrete PSO variants have been proposed, their performance is generally not satisfactory when compared with other meta-heuristics for discrete optimization [18].

More recently, set-based concept is introduced into PSO to solve combinatorial optimization problems, such as determining RNA secondary structure [14], traveling salesman problem (TSP) and multidimensional knapsack problem (MKP) [15]. This concept has been proved to be promising. Based on the set-based scheme, we use QPSO to minimize the total computation cost of cloud workload.

2. TASK SCHEDULING MODEL
Figure 1 shows the architecture model for which the proposed algorithm is implemented. Here the works are submitted by the user to the computing system. As the submitted user work arrive to the cloud they are queued in the stack. The cloud controller estimates the work size and checks for the availability of the virtual machine and also the capacity of the virtual machine. Once the work size and the available resource (virtual machine) size match, the work scheduler immediately allocates the identified resource to the user work in queue. Unlike the round robin scheduling algorithm, there is no overhead of fixing the time slots to schedule the works in a periodic way. The impact of the proposed algorithm is that there is an improvement in response time and the processing time. The works are equally spread, the complete computing system is load balanced and no virtual machines are underutilized. The novel advantage, is that the computational cost and the data transfer cost are minimized.

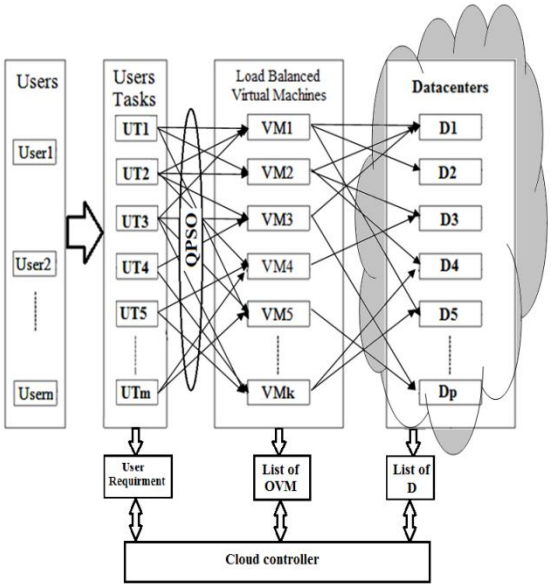

Fig. 1: Scheduling and QPSO load balancing model

The main parts of this model consist as follows.

User Requirement: to store the user submits the task, and tasks are queued. Responsible for the classification and record the user tasks. This represent as m users, as $w_1, w_2, \cdots, w_m$, n independent tasks, as $t_1, t_2, \cdots, t_n$.

List of Virtual Machine: responsible for collecting and recording information about the list of the currently idle machine resources. This represents as kVMs, as $VM_1, VM_2, \cdots, VM_k$.

List of Datacenters: responsible for selecting available host in a datacenter, which meets the memory, storage, and availability requirement for a VM deployment [14, 15, 16]. This represent as p datacenters, as. $D_1, D_2, \cdots, D_p$

Cloud Controller: responsible for the collection of user requirements; the user task submitted by the task scheduling resources to a virtual machine to complete the scheduling task.

Works scheduling aims at assigning work to victuals machines in the cloud so that the execution time (*makespan*) of the overall tasks of work is minimized. This problem can be formulated as follows.

$$\text{Userwork} = w_1, w_2, \cdots, w_n \quad (2.1)$$

a set of 'n' works to be scheduled. Moreover, we consider for each Userwork'i'

$$\text{UserworkiT} = UTi_1, UTi_2, \cdots, UTi_n \quad 2.2)$$

as a set of 'm' task partitions of worki disseminated among 'm' cloud datacenters (D) in order to be executed. Consequently, each cloud datacenter can carries out a disjoint subset of the decomposed jobs set. For its assigned jobs, MVj ensures the execution of their tasks as follows:

$$VMj\text{Tasks} = \{UTaj, UTbj, \cdots, UTrj\} \quad (2.3)$$

The union of these overall disjoint subsets gives the whole set of works. For example, VMj carries out

$$VMj\text{Tasks} = \{UT3j, UT6j, \cdots, UT9j\} \quad (2.4)$$

which are tasks of userworkw3, w6,…, and w9, respectively.

Therefore, the total processing time of all work ('r' tasks) assigned to VMj would be:

Makespan(VMjTasks) =Max (UTkj.StartTime + UTkj.ExeTime), (2.5)

whereUTkj.StartTime is the time when work task 'k' UTkj starts executing on VMi and UTkj.ExeTime is the execution time of UTkj at VMj. Thus, the work scheduling problem in the cloud computing could be defined as searching of a set:

MvTasks = {VM1T, VM2T,…,VMpT} (2.6)

and

VMjTasks = {UTaj, UTbj, ...,UTrj} with $0 < r \leq n$ (2.7)

Which reduces: Makespan(VMjTasks)

In order to evaluate the quality of the requested solution (VMTasks), a fitness function is defined as follows (used to calculate the above makespan):

Fitness(VMTasks) = Σ (Fitness(UTij, VMj)) (2.8)

where ($1 \leq j \leq m$)

and

Fitness(UTij, VMj) = UTij.TimeToExe (2.9)

where,

UTij.TimeToExe is the execution time of task of job 'i' needs to run in VMj.

Each population contains 'N' individuals (solutions) where each one is represented by a set of datacenter. Each datacenter carried out a set of job tasks as follows:

Set of datacenters.

VMTasks = {VM1T, VM2T,…,VMpT} (2.10)

Each datacenter contains a set of affected work tasks as follows:

VMjTasks = {UTaj, UTbj, ...,UTrj} (2.11)

The load balancing initialization aims at the generation of the first population in which 'N' individuals are randomly selected. For example, the following individuals are selected:

VMTasks={VM1T, VM2T,…,VMmT}

={<UTa1,UTb1,...,UTr1>,<UTa'2,UTb'2,...,UTr'2>,…,<UTa" m,UTb"m, ...,UTr"m>} (2.12)

To evaluate each individual, the above fitness function is applied. For this step, each task is characterized by its execution time (UTij.TimeToExe).

We choose to apply a dynamic stopping criterion. The load balancing iterations are carried out and stopped only when the fitness does not change during 'NS'. It is the stagnation state. The number 'NS' is a user parameter. Note that this process is limited by an iteration maximum number 'ItMax'.

3. THE PROPOSED QPSO LOAD BALANCING ALGORITHM

We utilize the characteristics of particle algorithms mentioned above to schedule task. We can carry out new task scheduling depending on the result in the replication based task scheduling. It is very efficient in the cloud environment. In

contrast to other, PSO algorithm, the QPSO algorithm inherits the basic ideas from PSO algorithm to decrease the computation time of tasks executing, it also considers the loading of each VM. We can carry out new task scheduling depending on the result in the past task scheduling.

3.1 Initialize Pheromone of VMj

At the beginning, particles are distributed on the virtual machines randomly, and then it will initialize the VMj pheromone value based on

$$VMj\text{Tasks} = \{UT3j, UT6j, \cdots, UT9j\} \quad (3.1)$$

Where pe_numj is the number of VMj processor, pe_mipsj is the MIPS (Million Instructions Per Second) of each processor of VMj and the parameter VM_bwj that is related to the communication bandwidth ability of the VMj.

3.2 The Rule of Choosing VM for Next Task

The k-particle chooses VMj for next task with a probability that is defined as:

$$p_j^k = \sum[\tau_j(t)]^\alpha [EV_j]^\beta [LB_j]^\gamma \quad (3.2)$$

Where

- τj(t) is the VMj pheromone value at time t.
- EV j is the computing capacity of VMj, it is defined

as follows:

$$EV_j = pe\_num_j \times pe\_mips_j + vm\_bw_j \quad (3.3)$$

Where pe_numj is the number of VMj processor, pe_mipsj is the MIPS of each processor of VMj and the parameter VM_bwj that is related to the communication bandwidth ability of the VMj.

LBj is the load balancing factor of VMj, to minimize the degree of imbalance, which is defined as follows:

$$\boldsymbol{LB_j = \frac{res_j - lastAver\_res}{res_j - lastAver\_res}} \quad (3.4)$$

Where lastAver_res is the average execution time of the virtual machines in the last iteration of the optimal path, and resj is the expected execution time of the task in the VMj, which is defined as follows:

$$res_j = \frac{total\_tasklengt\ h}{EV_j} + \frac{InputFilesize}{vm\_bw_j} \quad (3.5)$$

Where total task length is the total length of the tasks that have been submitted to VMj, and

Input File size is the length of the task before execution.

α, β and γ are three parameters that control the relative weight of the pheromone trail, the computing capacity of VMs and the load balancing factor of VMs.

Once some VMs are loading heavy, it becomes a bottleneck in the cloud and it influences the makespan of a given tasks set. Therefore we define the load balancing factor LBj in the particle algorithm to improve the load balancing capability, and the bigger LBj of VMj should be chosen with high probability, that means the comprehensive ability of VMj is power now, and then it is high desirable.

4. EVALUATION

The experiment is implemented using CloudSim platform. The scheduling algorithms of the experiment include the QPSOLB, the basic PSO [6] and ACO.

4.1 Assumptions

Adopting the Scheduling and the load balancing model introduced in chapter 3, we assume that

- Tasks are mutually independent, i.e., there is no precedence constraint between tasks.
- Tasks are computationally intensive.
- Tasks are not preemptive and they cannot be interrupted or moved to another processor during their execution.

Assume all tasks are executed on the Amazon Elastic Compute Cloud (http://aws.amazon.com), all the data are stored in Amazon Simple Storage Service and data transmissions are fulfilled through the Amazon Cloud Front. And assume that Service 1 to be in US, Service 3 in Malaysia and Service 4 in Japan. Due to the varying price of service, in the following simulation, the price at this moment is adopted.

Cost of execution of Ti on Servicej is $0.17 per hour (resources for high-CPU, on-demand instance medium instances, Windows 7). Taskcost = Tasktime * Price.

The scheduling problem aims to minimize the total execution time of tasks as well as to achieve a well-balanced load across all VMs in Cloud. That is, there are two factors considered here. One is the minimization of the tasks completion time. The other is to distribute workload evenly among virtual machines

4.2 Experiment Result

We compared our QPSO algorithm with the Ant Colony Optimization (ACO) and the basic Particle Swarm System (PSO). The ACO and basic PSO algorithm aims to find the earliest completion time of each task individually.

The QPSO algorithm aims to minimize the makespan of a given set of tasks. The QPSO algorithm chooses optimal resources to perform tasks according to resources status and the size of given task in the cloud environment. Not only does it minimize the makespan of a given set of tasks but it also balances the entire system load.

In the following experiments, we compared the average makespan of the basic PSO, ACO and QPSO algorithm with different iterations; we also compared the average makespan of 100-500 tasks set, and the average degree of imbalance (DegreeImb) of each algorithm in the following experiments.

The average makespan of the basic PSO, ACO andQPSO algorithm with different iterations is shown in Figure 5.1. In this experiment, we used 300 tasks set to compare the average performance of the basic PSO, ACO and the QPSO algorithm, and we recorded the makespan using the time in the CloudSim (ms).

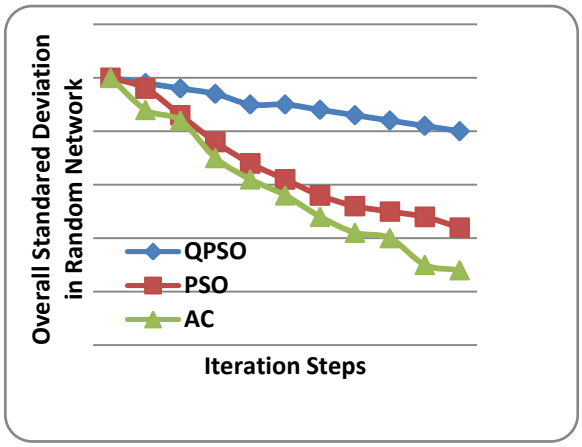

Figure 5.1 The average makespan of 300 tasks set

Figure.5.1 shows that the average makespan of the basic PSO, ACO and QPSO algorithm reduced roughly with the number of iterations increased. But for the basic PSO, ACO and QPSO algorithm, this change became slow after 50 iterations.

Hence, we used 50 iterations for other experiments in this chapter. The average makespan of each algorithm with the number of tasks varying from 100 to 500 is shown in Figure 5.1. In this experiment, we also use the time in the CloudSim (ms) to record the makespan. At last the average degree of imbalance (DegreeImb) of each algorithm with the number of tasks varying from 100 to 500 is shown in Figure 5.1.

It can be seen from the Figure 5.1 and Figure 5.2, the average performance of the QPSO algorithm is better than the basic PSO algorithm and ACO algorithm. It means that the QPSO can achieve good system load balance in any situation and take less time to execute tasks. In other words, these results demonstrated the effectiveness of the QPSO algorithm.

5. CONCLUSION

In this work, we presented task scheduling based on Quantium Particle Swarm Optimization for load balancing. We used the scheduling to minimize the makespan of execution of scientific application workload ows on Cloud computing environments.

We compared the results against basic PSO and Ant Colony. We found that QPSO based task-resource mapping can achieve at least three times cost savings as compared to PSO and AC based mapping. In addition, QPSO balances the load on compute.

6. ACKNOWLEDGEMENT

Appreciation conveyed to Ministry of Higher Education Malaysia for project financing under Exploratory Research Grant Scheme, RDU120608.

7. REFERENCES

[1] B. Raghavan, et al., "Cloud control with distributed rate limiting," Proc. SIGCOMM'07, pp. 337 - 348, Kyoto, Japan, 2007.

[2] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," IEEE Transactions on Software Engineering, pp. 369-384, 2007.

[3] K. Bhattacharya, et al., "ICSE Cloud 09: First international workshop on software engineering challenges for Cloud Computing," Proc. 31st International Conference on Software Engineering - Companion Volume,. (ICSE-Companion 2009), pp. 482-483. 2009

[4] W. Van der Aalst and K. Van Hee, Workflow management: models, methods, and systems: The MIT press, 2004.

[5] I. Foster, Zhao Yong, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared", Proc. Grid Computing Environments workshop, 2008. GCE '08, pp. 1-10, 2008.

[6] D. Yuan, et al., "A data placement strategy in scientific cloud workflows," Future Generation Computer Systems, pp. 1200-1214 2010.

[7] S. Pandey, et al., "A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," in Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on, pp. 400-407,2010.

[8] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, 2007, pp. 120-127.

[9] M. Clerc, "Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem," New optimization techniques in engineering(Springer), 2004.

[10] G. Pampara, et al., "Combining particle swarm optimisation with angle modulation to solve binary problems," Proc.The IEEE Congress on Evolutionary Computation , pp. 89-96, vol.1,2005.

[11] D. Sha and C. Hsu, "A hybrid particle swarm optimization for job shop scheduling problem," Computers & Industrial Engineering, pp. 791-808, vol. 51,2006.

[12] J. Grobler, et al., "Metaheuristics for the multi-objective FJSP with sequence-dependent set-up times, auxiliary resources and machine down time," Annals of Operations Research, pp. 1-32, 2008.

[13] M. Neethling and A. P. Engelbrecht, "Determining RNA Secondary Structure using Set-based Particle Swarm Optimization," IEEE Congress on Evolutionary Computation, BC, Canada,pp. 1670-1677, 2006.

[14] C. Wei-Neng, et al., "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems," IEEE Transactions on Evolutionary Computation, , vol. 14, pp. 278-300, 2010.

[15] Z. Wu, et al., "A Market-Oriented Hierarchical Scheduling Strategy in Cloud Workflow Systems," Journal of Supercomputing, Special issue on Advances in Network&ParallelComptg, to be appeared,2010.

[16] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," Journal of Grid Computing, no. 3, pp. 171-200, 2005.

[17] X. Liu, J. Chen, Z. Wu, Z. Ni, D. Yuan, Y. Yang, Handling Recoverable Temporal Violations in Scientific Workflow Systems: A Workflow Rescheduling Based Strategy. Proc. of 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2010), pages 534-537, Melbourne, Australia, May 2010.