Cumhuriyet Üniversitesi Fen Fakültesi
Fen Bilimleri Dergisi (CFD), Cilt:36, No: 3 Özel Sayı (2015)
ISSN: 1300-1949

Cumhuriyet University Faculty of Science
Science Journal (CSJ), Vol. 36, No: 3 Special Issue (2015)
ISSN: 1300-1949

A new approach for scheduling jobs in cloud computing environment

Shahab TAREGHİAN[1*], Zarintaj BORNAEE[1]

[1]*Department of Computer Engineering, Garmsar Branch, Islamic Azad University, Garmsar, Iran*

Received: 01.02.2015; Accepted: 05.05.2015

Abstract. Job dscheduling in cloud computing environment is one of the most important issues that must be considered by cloud computing service providers. Optimal job scheduling enables more efficient utilization of resources, which in turn leads to more customers satisfaction. Solution procedures to the problem of job scheduling in cloud computing environment have mainly focused on optimizing one quality criterion. In this paper, we propose a static solution procedure for the scheduling of jobs in cloud computing environment which is based on particle swarm optimization technique (PSO). Considering the virtual machine capabilities and having secured an appropriate method for request assignments, this solution procedure not only reduces the amount of memory needed, but minimizes the maximum job's makespan. The simulation results show that our proposed method reduces the maximum job makespan by a larger amount when compared to the other PSO based methods.

Keywords: Processing in cloud environment, scheduling, Particle swarm optimization, Computational complexity

1 INTRODUCTION AND BACKGROUND

In recent years, cloud computing has taken the attention of computer scientists and information technologists. This is due to the significant advantages of cloud computing such as cost and efficiency of service. In cloud computing, the end user can avail of various services without investing in the underlying architecture. It includes the delivery of software, infrastructure, and storage over the Internet based on user demand. In fact, cloud computing makes computer infrastructure and services available to users "on-need" basis.

Nearly two decades ago, storage and CPU was very expensive. With the advent of the PC, which brought mass storage and cheap CPUs, the file server gained in popularity as a way to enable document sharing and archiving. In the early 1990s, the idea of "the grid" began to take shape. At that time, the Internet had enough computers attached to it that motivated scientists to begin thinking about how they can connect those machines together to create a massive, shared pools of storage and compute power that would be larger than what any one company could possibly afford to provide. In grid computing, a system program divides and farms out pieces of the end user's computing request as one large system image to several thousand computers. The next stage of this evolvement is cloud computing which provides on-demand resource provisioning. In cloud computing environment a number of servers are provided in data centers to collectively satisfy the user's demands. Should the need arises, the service providers may hire virtual machines from the cloud service providers to serve their customers and achieve the service level required. The challenge for the cloud service providers is to schedule the jobs such that the quality standards set by the service providers are met and the user's needs are satisfied. In fact, scheduling in cloud computing environment concerns the optimal assignment of service requests to the available resources not violating the systems constraints.

Scheduling algorithms are generally categorized as static and dynamic. In static scheduling algorithms, all the required information such as the number of jobs, the number of resources, the

*Corresponding author *Email address*: Tareghian.shahab@gmail.com.

Special Issue: The Second National Conference on Applied Research in Science and Technology

processing times, etc must be known in advance. In contrast, in dynamic scheduling, resources are allocated as and when users require.

The literature on scheduling jobs is relatively rich. In some research, where a single objective has been the focus of the research, the attempt has been to reduce the energy consumptions in data centers by allocation of appropriate virtual machines. In other reseach, multiple objectives have been considered [see e.g. 1,2]. Some studies have focused on reducing cost of utilization of the virtual machines [3,4]. Evolutionary algorithms such as particle swarm optimization [5-7] and ant colony [8,9] have been used to tackle the problem job scheduling in cloud computing environment. These studies have mainly focused on optimizing one criterion. In the present paper, we propose a new model which not only reduces the need for storage, but also minimizes the makespan of the longest job.

The rest of the paper is organized as follows. In Section 2, particle swarm optimization is breifly reviewed. In Section 3, the proposed model is illustrated and its features are discussed. Section 4 contains the simulation results where the effectiveness and efficiency of the proposed model in comparison to other methods are discussed. Finally, in Section 5 some conclusions are drawn and new directions for research are suggested.

2 PARTICLE SWARM OPTIMIZATION ALGORITHM

Particle swarm optimization (PSO) as developed by Kennedy and Eberhart was motivated by the simulation of simplified social behaviour of animals [10]. It soon gained popularity as an optimization algorithm, because when compared to other evolutionary algorithms like genetic algorithm, it is easier to implement and there are fewer parameters to adjust. It has been applied successfully in many application areas, including function optimization, job scheduling, artificial neural network, etc. The basis of the PSO lies in a hypothesis that a solution to an optimization problem can be treated as a group of particles that move through a D-dimensional space, adjusting their movements in the search space according to their own experience and the experience of other particles in the group. To implement PSO for optimization purposes, each single solution is considered as a particle. All of the particles have fitness values, which are evaluated by the fitness function to be optimized, and have velocities that direct the movement of the particles. The particles move through the problem space by following the current optimum particles. In PSO, each particle is associated with three D-dimensional vectors. For the ith particle these vectors are: (a) $x_i(x_{i1}, x_{i2}, \cdots, x_{iD})$ in the D-dimensional space, where $x_{ik} \in [l_k, u_k]$ in which l_k and u_k are the lower and upper bounds of the kth dimension respectively; (b) $v_i(v_{i1}, v_{i2}, \cdots, v_{iD})$ the velocity which is clamped to a maximum velocity V_{max}, provided by the user; and finally (c) $\mathcal{X}_i^{pbest}\left(\mathcal{X}_{i1}^{pbest}, \mathcal{X}_{i2}^{pbest}, \cdots, \mathcal{X}_{iD}^{pbest}\right)$ the best position of the particle in the D-dimensional space. The value of the objective function with regards to the best position of particle i is denoted by F_i^{pbest}. As the algorithm evolves, x_is and v_is are updated. In each iteration, if the current solution is better than the previous best solution, then the current solution replaces the previous best solution. The overal best position that particles have had up to the current iteration, is saved in $\mathcal{X}^{gbest}$. The value of the objective function with regards to $\mathcal{X}^{gbest}$ position of particle i is denoted by f^{best}. Initially, the positions and the velocities of the particles are assigned randomally. Then, uring the execution of the algorithm, the positions and the velocities of the particles in iteration, $t+1$ are built from the positions and velocities of the corresponding particles in iteration t. Generally, to calculate the velocities and the positions of the particles in iteration $t+1$ we use (2.1) and (2.2) respectively, as follows:

$$v_i(t+1) = wv_i(t) + c_1 r_1 \left(x_i^{pbest}(t) - x_i(t)\right) + c_2 r_2 (x^{gbest}(t) - x(t)) \quad (2.1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2.2)$$

where w is the inertia weight, c_1 and c_2 are the acceleration coefficients, with r_1 and $r_2 \sim U(0,1)$. The random numbers r_1 and r_2, somehow provide a measure of diversity in the search space. Coefficients c_1 and c_2, respectively, are learning rates for individual ability (cognitive) and social influence (group). In other words, they represent weight of memory (position) a particle towards memory of the group (swarm).

It is clear that to solve (1), three items of information is required, namely the previous position of the particle, the best position that the particle has experienced so far, and finally the best position that the swarm has experienced. In some variations of the PSO, equations (3.2) and (4.2) have been modified to:

$$v_i(t+1) = wv_i(t) + c_1 R_1 \otimes \left(X_i^{pbest}(t) - x_i(t)\right) + c_2 R_2 \otimes \left(X^{gbest}(t) - x(t)\right) \quad (3.2)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4.2)$$

where R_1 and R_2 are vectors of random numbers in D-dimensional space , operator $\otimes$ is the tensor product.

The steps of the standard PSO algorithm can be summarized in pseudo code as in Fig. 1.

```
For each particle
{
 Initialize particle
}
Do until maximum iterations or minimum error criteria
{
   For each particle
   {
      Calculate fitness value
      If the fitness value is better than pBest
      {
         Set pBest = current fitness value
      }
      If pBest is better than gBest
      {
         Set gBest = pBest
      }
   }
   For each particle
   {
      Calculate particle Velocity
      Use gBest and Velocity to update particle Data
   }
}
```

Figure 1. An overview of the particle swarm algorithm

3 PROPOSED METHOD

The previously proposed algorithms for job scheduling in cloud computing environment such as [13-15] have not considered some of the influential factors. Some of these factors are as follows:

- The issue of load balance between virtual machines has been ignored in previous studies. In the proposed method, through minimization of the longest job, we achieve a measure of load balance between machines.
- In the PSO based optimum algorithm, computational power and storage have not been considered simultaniously. In the proposed method both the storage capacity together with the computattional capacity of virtual machines have been considered.

Let $\mathcal{VM} = \{VM_1, VM_2, \cdots, VM_m\}$ be the set of virtual machines that are used to host the users requests. Moreover, let $\mathcal{T} = \{T_1, T_2, \cdots, T_n\}$ be the set of jobs that are to be executed on virtual machines. The length of the job with the largest makespan that the proposed algorithm is going to reduce it, is defined as follows:

$$CT^* = max\{CT_{ij} | \, i \in \mathcal{T}, j \in \mathcal{VM}\} \tag{5.3}$$

where CT^* is the largest makespan, CT_{ij} is the processing time of job i on machine j.

The solution procedure starts with a swarm containing some particles which is equal to number of tasks. Each particle is assigned a random position and a random velocity. Fig. 2 shows the mapping of jobs to resources. For instance, Task 1 is assigned to Pc1, etc.

Task 1	Task 2	Task 3	Task 4	Task 5
PC1	PC3	PC2	PC3	PC1

Figure 2. Mapping of jobs to resources

In the next stage of the solution procedure the value of fitness function with respect to each particle is evaluated as (6.3)

$$C_{total} = T(s, v) + hp(s, v) + hm(s, v) \tag{6.3}$$

where $T(s, v)$ is the processing time of the set of jobs s, on virtual machine v, $hp(s, v)$ is the computational power relating the set of jobs s, on virtual machine v and finally $hm(s, v)$ is the amount of storage on virtual machine v.

In each iteration after evaluating the particles, the best value of the fitness function is stored in $\mathcal{X}^{gbest}$. In addition, the best positions of particles is saved in $\mathcal{X}_i^{pbest}$. This process is repeated until one of the termination consditions occure. Then the particle with the best value is considered as the solution of the problem. In the sequel, HPSO refers to our proposed model.

4 SIMULATION AND PROPOSED METHOD EVALUATION

We implemented our proposed method on the Cloudsim simulator [16]. We have simulated a data center under cloud computing environment having 5 hosts with virtual capabilities. In fact, we have assumed that on host computers softwares such a Xen has been installed to share the

resources. All the simulation studies are performed on a system that have RAM=4 GB, CPU= core 2 Duo 2.53 Hz. Table 1 shows the machine characteristics that are used to evaluate HPSO. In Table 1, Mips is the number of instructions per processor (in millions), core shows the number of processors for each machine, RAM is the main memory and storage is the secondary memory.

To tune the PSO, an extensive experiments were carried out to determine the appropriate values of the parameters. These are as follows: umber of particles in the swarm: 10; number of iterations: 100; learning rate with regards to individual ability: 1.49445; and finally learning rate with regards to social influence: 1.49445.

Table 1. Characteristics of the machines used to evaluate HPSO

CPU (Mips)	Core	RAM (MB)	Storage (Bs)	Band-Width (MB/s)
27079	2	2048	1048576	102400
177730	6	1024	1048576	102400
27079	2	2048	1048576	102400
12089	4	1024	1048576	102400
177730	6	1024	1048576	102400

Considering 40 virtual machines with different capabilities, we have compared the performance of the HPSO with the optimum PSO based algorithm based on the longest makespan of the jobs. Figure 3 Show the results. Keeping the number of machines fixed at 40, we have varied the number of requests from 100 to 600. As it can be seen from Fig. 3 irrespective of the number of requests, the makespan of the longest job in HPSO has always been smaller than those obtained by the optimum PSO based algortihm.

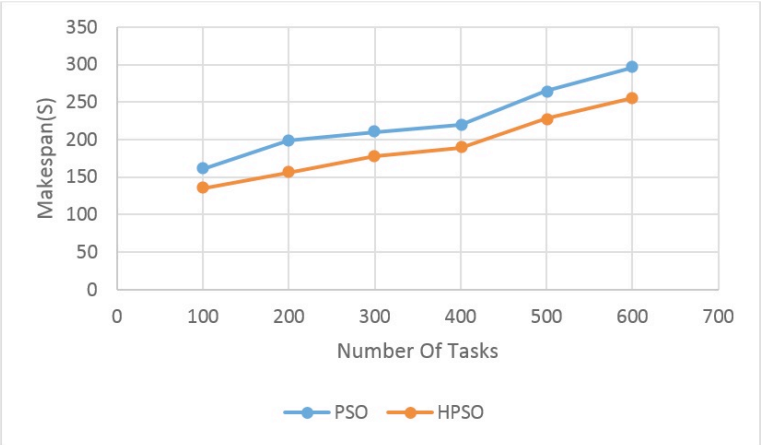

Figure 3. The makespan of the longest job in HPSO vs the optimum PSO based algortihm.

To compare the processing times of the HPSO with those of the optimum PSO based algortihm, we have carried out more experiments with settings as depicted in Table 2.

Table 2. Host Parameters.

Host ID	CPU Mips	Core	RAM (GB)	Storage (TB)	Band-Width
Host-1	27079	2	2	1	10 GB/s
Host-2	177730	6	1	1	10 GB/s

We assume that there are $vmnr$ virtual machines with time shared scheduling [16] in the data center which have characteristics as Table 3.

Table 3. VM Parameters.

VM	CPU Mips	Core	RAM (MB)	Storage (GB)	Band-Width
VMs	9726	1	512	10	1 GB/s

To compare the performance of the HPSO with the optimum PSO based algortihm with regards to the computational power and the utilized memory, we assumed that there are 89 requests. Furthermore, to analyse the effect of the number of virtual machines on the results we considered three levels of resources, i.e. 4, 6 and 10 resources, respectively. The results are depicted in Table 4. As it can be seen, under fixed number of requests and irrespective of the number of resources that are assigned, the longest job processing time under HPSO is always less than those obtained by the optimum PSO based algortihm. In fact the percentage of improvement ranges from 8.6% to 11.3%.

Table 4. Simulation results.

No. of requests	No. of resources	HPSO	PSO	% Improvement
89	4	40.43	44.24	8.6
89	8	22.32	25.12	11.1
89	10	17.65	19.89	11.3

The performance of these two algorithms are also shown in Figure 4. In Fig. 4 the time is dispalyed as a function of number of virtual machines. As it can be seen, the HPSO's performance outperforms that of the optimum PSO based algortihm.

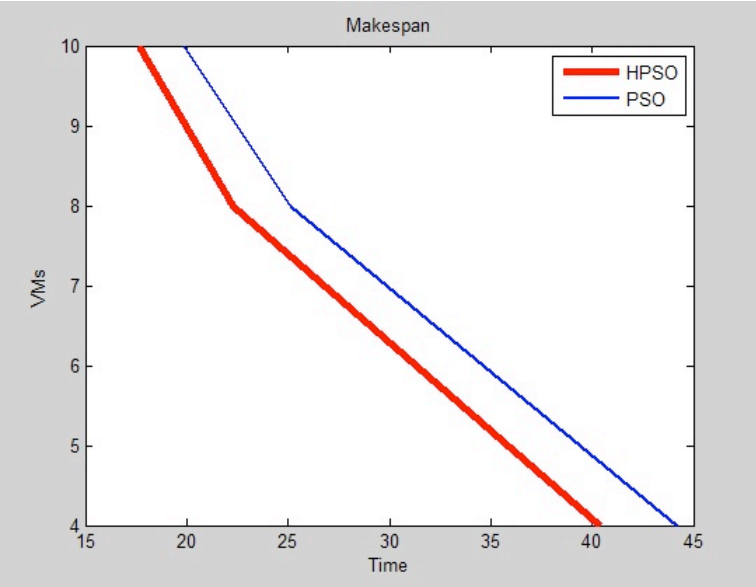

Figure 4. HPSO's performance vs the optimum PSO based algortihm's performance.

5. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

Scheduling jobs in cloud computing environment has attracted a lot of attention in recent years. In this paper, we proposed a new algorithm to schedule jobs in cloud computing environment coupled with some heuristics. The coupling of heuristics into the algorithm, causes

the PSO to search a a smaller space and by keeping the promising areas, not only increases the processing speed, but also generates acceptable solutions. All the simulated results carried out in Clousim environment shows the superiority of the proposed algorithm when compared to the optimum PSO based algortihm with respect to the processing time required to schedule the jobs and also to reduce the makespan of the longest job.

Recently, it has been shown that the hybrid heuristic algorithms produce better results than each of the heuristics individually. Hence, we suggest to implement our proposed algorithm with hybrid heurristics.

REFERENCES

[1] Hu, J., Gu, J., Sun, G., & Zhao, T. (2010, December). A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on (pp. 89-96). IEEE.

[2] Wang, S., & Meng, B. (2007). Resource allocation and scheduling problem based on genetic algorithm and ant colony optimization. In Advances in Knowledge Discovery and Data Mining (pp. 879-886). Springer Berlin Heidelberg.

[3] Abdullah, M., & Othman, M. (2014). Simulated Annealing approach to Cost-Based Multi-Quality of service job scheduling in cloud computing enviroment. American Journal of Applied Sciences, 11(6), 872-877.

[4] Krishnasamy, K. (2013). Task scheduling algorithm based on hybrid partical swarm optimization in cloud computing environment. Journal of Theoretical & Applied Information Technology, 54(1).

[5] Cao, Q., Wei, Z. B., & Gong, W. M. (2009, June). An optimized algorithm for task scheduling based on activity based costing in cloud computing. In Bioinformatics and Biomedical Engineering, 2009. ICBBE 2009. 3rd International Conference on (pp. 1-3). IEEE.

[6] Nan, X., He, Y., & Guan, L. (2013, May). Optimal resource allocation for multimedia application providers in multi-site cloud. In Circuits and Systems (ISCAS), 2013 IEEE International Symposium on (pp. 449-452). IEEE.

[7] Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010, April). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on (pp. 400-407). IEEE.

[8] Li, K., Xu, G., Zhao, G., Dong, Y., & Wang, D. (2011, August). Cloud task scheduling based on load balancing ant colony optimization. In Chinagrid Conference (ChinaGrid), 2011 Sixth Annual (pp. 3-9). IEEE.

[9] Zhu, L., Li, Q., & He, L. (2012). Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm. IJCSI International Journal of Computer Science Issues, 9(5), 54-58.

[10] Eberhart, R. C., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science (Vol. 1, pp. 39-43).

[11] Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. Evolutionary Computation, IEEE Transactions on, 6(1), 58-73.

[12] Dréo, J. (Ed.). (2006). Metaheuristics for hard optimization: methods and case studies. Springer Science & Business Media.

[13] Chiu, C. F., Hsu, S. J., Jan, S. R., & Chen, J. A. (2014). Task Scheduling Based on Load Approximation in Cloud Computing Environment. In Future Information Technology (pp. 803-808). Springer Berlin Heidelberg.

[14] Lin, W., Liang, C., Wang, J. Z., & Buyya, R. (2014). Bandwidth‐aware divisible task scheduling for cloud computing. Software: Practice and Experience, 44(2), 163-174.

[15] Rahman, M., Hassan, R., Ranjan, R., & Buyya, R. (2013). Adaptive workflow scheduling for dynamic grid and cloud computing environment. Concurrency and Computation: Practice and Experience, 25(13), 1816-1842.

[16] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), 23-50.