2015 Fifth International Conference on Communication Systems and Network Technologies

Cloudlet Scheduling with Particle Swarm Optimization

Hussein S. Al-Olimat and Mansoor Alam
Dept. of Elec. Engineering and Computer Science
University of Toledo
Toledo, OH 43606
{Hussein.AlOlimat | Mansoor.Alam2}@utoledo.edu

Robert Green and Jong Kwan Lee
Dept. of Computer Science
Bowling Green State University
Bowling Green, OH 43403
{greenr | leej}@bgsu.edu

***Abstract*—Cloud computing is a particularly interesting and truly complex concept of providing services over networks on demand. Many tools have previously been created to simulate the work of the clouds, such as CloudSim. The main use of these tools is evaluation and testing of architectures and services before deployment on network centers and hosts in order to avoid any potential failures or inconveniences. The benefits of using the pay-per-use clouds may be affected by underutilization of the already reserved resources, so the maximization of system utilization while simultaneously minimizing makespan is of great interest to cloud providers wishing to reduce costs. To minimize makespan and increase resource utilization, a hybrid of particle swarm optimization and simulated annealing is implemented inside of CloudSim to advance the work of the already implemented simple broker. The new method maximizes the resource utilization and minimizes the makespan.**

Keywords-Cloud Computing; Particle Swarm Optimization; Task Scheduling; CloudSim;

I. INTRODUCTION

Cloud computing is a term used to describe the on-demand, elastic, and scalable services and resources (computation, servers, storage, applications, etc.) that are offered to consumers over a network. According to NIST [1], one of the essential characteristics of the clouds is to give the consumer the illusion that network capabilities and computing power are unlimited, and can be requested at any time and in any quantity. Achieving such a goal has led to one of the major focuses of IT professionals working in the cloud computing arena: reduction of costs, data center footprints, and improving the computational efficiencies while using virtualization in the cloud. This is particularly important as users expect a high level of service at lower prices. Considering this, the widely used saying "time is money" tends to apply when working with a cloud. When a user requests cloud resources, the cloud should be able to serve the user's request as soon as possible and in a cost-effective manner. In order to satisfy the "unlimited" and "elastic" characteristics of the cloud, consumer requests are often handled by way of the First- Come-First-Serve (FCFS), where the customer request should be the driving factor for workload scheduling [2]. Yet, the FCFS methodology often fails to fully utilize cloud resources, resulting in longer run times and higher costs.

To address this issue, this study presents a solution for improving the makespan scheduled tasks and the utilization of cloud resources through the use of a hybrid variant of popular population-based metaheuristic (PBM) algorithm, particle swarm optimization (PSO). The study also proposes the use of a simulated annealing algorithm to enhance the performance of the binary PSO for scheduling. These methods are integrated with the CloudSim [3] tool and tested using a real workload [4].

The remainder of this paper is organized as follows: Section II contains the background regarding scheduling, clouds, and PSO algorithm as used in this study. Section III discusses the ideas and methods implemented to optimize the makespan time; the results of the implemented methods are examined in Section IV, and Section V closes the paper.

II. BACKGROUND

Wherever When optimizing scheduling in a system such as a cloud, it is important to consider both makespan and resource utilization. Makespan is the total time the network resources take to finish executing all the tasks/jobs and utilization is the measure of how well the overall capacity of the cloud is used. In order to increase the utilization of the resources at a given time, an effective scheduling algorithm should be used to assign the tasks among reserved resources. Makespan and utilization have an inverse relationship, i.e. increasing utilization will decrease the makespan. In other words, to achieve higher utilization, tasks should be distributed more efficiently among all the reserved cloud resources. Such optimization will also reduce consumer costs. As Workload scheduling of this type is known to be a NP-Complete problem, metaheuristic and PBMs have been used to solve such problems (e.g. [5-8]). Some examples of these efforts include a discussion of the features of GAs, Simulated Annealing (SA), and Tabu Search (TS) as three basic heuristics for grid scheduling [9] and the use of Ant Colony (ACO), Particle Swarm Optimization (PSO) and Genetic algorithms (GAs) [10]. In multiple cases, the use of PSO was found to be superior to other algorithms in terms of both performance and clarity [6], [11-12]. PSO has also been used in order to do workflow scheduling in the cloud where the focus has been on increasing utilization and reducing makespans [13-14]. Zhang et al. also implemented a hierarchical swarms in CloudSim for load balancing and scheduling costs reduction, showing good results in contrast to the Best Resource Selection algorithm [15]. While common or standard variants of PSO have typically been used in such studies, the performance of PSO can be improved with the help of other algorithms or by using an improved version of PSO [16]. For example, a hybrid version of PSO and TS was used in CloudSim to maximize the utilization and reduce energy consumption, resulting in an energy reduction of 67.5%. [17].

Due to the superiority of the PSO for optimizing the scheduling of tasks on cloud systems, this study leverages PSO alongside a SA algorithm while also proposing an improved broker method that shows better performance than the typical round-robin broker in CloudSim.

A. CloudSim

Cloud simulation tools allow users to test modeled services in a controlled environment with different workloads and scenarios before deploying them on real clouds [18], [3]. The CloudSim tool [3] starts by creating a network of nodes as illustrated in Fig. 1. A basic node consists of data centers, hosts, and cloud brokers. Data centers (resource providers in CloudSim) are created first with specifications that define the operating system, memory, bandwidth, storage, etc. One or more hosts are then created on each data center with the proper specification of memory, storage, bandwidth, processing elements, and the selection of the scheduling algorithm to schedule virtual machines inside the host. Processing elements are known as cores or CPUs, where each processing element is given a defined processing power measured in millions of instructions per second (MIPS). Hosts are managed by data centers where each data center may manage a single or numerous hosts. The cloud broker, “an entity that creates and maintains relationships with multiple cloud service providers” [19], is also created to distribute work among the available data centers or cloud services. A cloud broker is the middleware between a user and the cloud service providers.

After creating all of the network nodes in CloudSim, a user may create virtual machines (VMs) to run on the specified hosts. Characteristics of each VM are defined by parameters such as processing power in MIPS, memory in megabytes, bandwidth, etc. As is illustrated in Fig. 1, a scheduling algorithm should be chosen to schedule tasks or cloudlets inside the virtual machine.

A final task in developing a CloudSim-based simulation is the generation of tasks/jobs (i.e. cloudlets) either by initializing them through code or from existing workload traces. Cloudlets are defined based on specifications including the task length in millions of instructions (MI), number of processing elements, and a utilization model that states the cloudlet’s execution rate through defining the current requested MIPS from the processing elements.

When generating cloudlets from workload traces, the workload format should be checked to make sure that it follows the standard format described in [20]. A given cloudlets’ length should also be converted to MI instead of the standard’s execution time in seconds by multiplying the execution time by the execution rate, where the default execution rate is 1 MIPS in CloudSim. E.g., a cloudlet with an execution time of 10 seconds is converted to 10 MI. After creating network nodes, VMs, and cloudlets, the list of

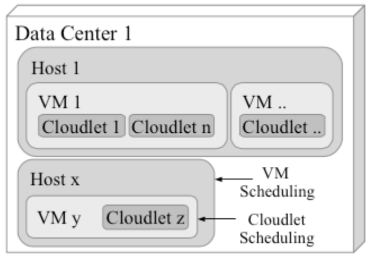

Fig. 1. Network Topology in CloudSim Network

available VMs and cloudlets are submitted to cloud brokers.

Scheduling in CloudSim is performed at the node level. The authors of CloudSim have already implemented two different scheduling algorithms in two different levels of the software: the VM level and the host level. This is shown in Fig. 1. The provisioning scenarios use space-shared and time-shared policies for VMs and task units [3]. Using the space-shared policy allows one VM or cloudlet to be executed at a given moment of time. On the contrary, using the time-shared policy allows multiple cloudlets to multi-task within a VM and will allow VMs to multi-task and run simultaneously within a host.

By default, VMs will be allocated to the hosts on a FCFS basis as mentioned before and as specified in [3]. This gives the user the illusion of having unlimited resources even though the Shortest Job First (SJF) policy was found to be faster than FCFS [21], though the SJF policy also often under performs when computationally large tasks are present. Thus, FCFS is generally ideal policy. Similarly, cloudlets will be executed on the corresponding VM on FCFS bases, after being allocated to the VMs by the broker. The brokering mechanism inside of CloudSim iterates through all cloudlets and assigns them to available VMs one by one. Contrary to this methodology, the proposed method assigns cloudlets to VMs in a more intelligent manner to achieve higher levels of utilization.

B. Particle Swarm Optimization (PSO)

PSO is a PBM algorithm inspired by bird flocking and fish schooling, where each particle learns from its neighbors and itself during the time it travels in space. There are two versions of PSO. The original PSO was first introduced in 1995 and operated in a continuous space [22]. Later, in 1997 the discrete, binary version of the algorithm was presented to operate on discrete binary variables [23]. The binary PSO was utilized by [24] to solve the task assignment problem and by [11] and [25] to schedule jobs in grid computing. Both methods found an optimal solution of the problem.

PSO starts by creating a number of particles to form a swarm that travels in the problem space searching for an optimum solution. An objective function should be defined to examine every solution found by each particle throughout the traveling time. A particle in this method is considered as a position in D-dimensional space, where each element can take the binary value of 0 or 1. Additionally, each particle has a D-dimensional velocity, where each element is in the range [0, 1]. Velocities are calculated as probabilities that change during the time particles move in space, where each velocity value changes from one state to another.

The individuals in this algorithm are a group of particles that move through a search space with a given velocity. At each iteration, the velocity, v_i, and position, p_i, of each particle is stochastically updated by combining the particle's current solution, the particle's personal best solution or $\hat{p}_i$, and the i global best solution or $\hat{g}$ over all particles. The required mathematics are listed in (1) and (2) where ω is the inertial constant, c_1 and c_2 represent cognitive and social constants that are usually about 2.00, and r_1 and r_2 are random numbers. In order to update the velocities and positions of each particle, (3) and (4) are used to add nonlinearity where p_i is the i^{th} component of particle p and r is a uniform random number. (1)-(4) indicate that the velocity of neighbors and the current velocity of the particle itself contribute in deciding the next position of the particle. In order for a particle to be part of the swarm and be able to keep up with the other particles during the search of a solution, each particle adapts to the velocity of the swarm as a whole by learning from itself and its neighbor particles.

$$v_i = \omega v_i + c_1 r_1 \cdot (\hat{p}_i - p_i) + c_2 r_2 \cdot (\hat{g} - p_i) \tag{1}$$

$$p_i = p_i + v_i \tag{2}$$

$$s(p_i) = \frac{1}{1 + \exp(-p_i)} \tag{3}$$

$$p_i = \begin{cases} 0, & s(p_i) \le r \\ 1, & \text{Otherwise} \end{cases} \tag{4}$$

III. Proposed Method

A. Problem Formulation

When using CloudSim, the Cloud broker assigns the cloudlets to the available cloud resources (VMs). The main objective of the proposed solution is to achieve the highest possible utilization of the group of VMs with a minimum makespan. However, the costs of using the resources and the positions of VMs across the system are not considered as parameters in this solution. Task migration between resources is also not allowed which means that every cloudlet can run on one VM until it is done executing.

The objective function of the algorithm is to find the shortest time of running the tasks on the available resources, i.e. to minimize the makespan. So the best combination of the pairs of cloudlets on virtual machines can be found. For this, (5) is used to calculate the fitness value of each PSO particle. The fitness function in (5) calculates the execution times of all possible cloudlet combinations on every cloud resource and then returns the highest execution time as the fitness value of the particle, where $EXC_{VMn}(j_m)$ is the execution time of running the set of cloudlets j_m on VMn. j_n is a normal set, e.g. $j_n = [C_1+C_2+...C_x]$, where x is the number of cloudlets. Also, n and m are the number of VMs and number of possible sets of cloudlets, respectively. The position vector consists of binary variables denoting the VM-Cloudlet assignment. The complete PSO algorithm used in this study is shown in Algorithm 1.

$$Fitness = Max[EXC_{VMi}(j_1), \ldots, EXC_{VMn}(j_m)] \tag{5}$$

Algorithm 1 Binary PSO Algorithm

```
1:  procedure BINARY PSO($nodesList$)
2:      CalculateExecTimes();
3:      initSwarm();
4:      initGlobalBest();
5:      for $i \leftarrow 0 \rightarrow numberIterations$ do
6:          for $j \leftarrow 0 \rightarrow numberParticles$ do
7:              calculateInertiaValue();
8:              calculateNewVelocities();
9:              calculateNewPositions();
10:             calculateFitnessValue();
11:             evaluateSolution();
12:             updateParticleMemory();
13:             updateGlobalBest();
14:         end for
15:     end for
16: end procedure
```

B. The Inertia Weight

The inertia weight ω in (1) is one of the most important adjustable parameters in PSO besides the acceleration coefficients and random variables. The inertia weight value can affects the overall performance of the algorithm in finding a potential optimal solution in less computing time. Many techniques are used to choose or modify the value of ω at runtime, such as the fixed inertia weight (FIW) that use a constant value, linearly decreasing inertia weight (LDIW) implemented by [26] and [27] which changes the value of the inertia weight linearly and per iteration. Also, the adaptive inertia weight (AIW) and random inertia weight (RIW) implemented by [26] and [28], where the inertia weight is slowly scaled from a larger to smaller value as the iterations progress (typically from 0.9 to 0.4).

The inertia weight in Algorithm 1 is calculated using the RIW [29]. This method showed an advantage against the other methods mainly in adjusting the balance between particle's local search and global search abilities. Also, in order to increase the probability of finding a near-optimal solution in fewer iterations and computing time, RIW uses a simulated annealing mechanism in addition to the LDIW method in order to overcome issues related to getting stuck in local optima [30], [29]. The authors of the RIW method have combined the linearly decreasing method with a constrained random inertia weight as thoroughly discussed in [29] to overcome the problems of the linear method. This combination of RIM and SA are used in this study.

IV. Simulations and Results

In our experiments, 100 particles were created where each particle's fitness is evaluated during 1,000 iterations. Additionally, based on a study [31] that compared inertia weights and constriction factors in particle swarm optimization, the values of the acceleration coefficients, C_1 and C_2 in (1), were set to 1.49445 for better performance.

The proposed scheduling method was tested inside CloudSim as part of the cloud broker. The cloud simulation was implemented as follows: (1) One data center was created with the default characteristics defined by CloudSim

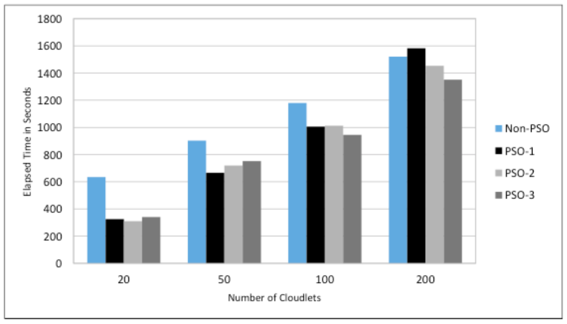

Fig. 2 Makespan Simulations.

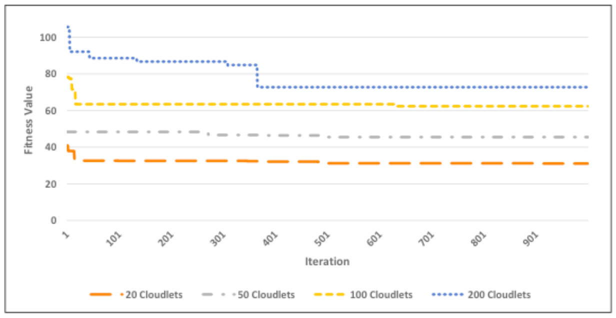

Fig. 3. Fitness Value vs. Iteration

authors as in example 6 provided with the source code, (2) two different hosts were created on the data center with: 2 GB RAM, 1 TB storage, 10 GB/s bandwidth and time-shared scheduling algorithm was chosen to schedule VMs on hosts. The first host has an Intel Core 2 Extreme X6800 processor with 2 cores (PEs) and gives a cumulative processing power of 27,079 MIPS [32]. The second host has an Intel Core i7 Extreme Edition 3960X with 6 cores that gives a cumulative processing power of 177,730 MIPS, (3) cloud broker that implements PSO was used, (4) 5 VMs were created, where each has an Intel Pentium 4 Extreme Edition processor with: 10 GB image size, 0.5 GB memory, 1 GB/s bandwidth and 1 PE that gives 9,726 MIPS processing power. Xen virtual machine monitor [33] was also used for all of them, in addition to, using the time-shared scheduling to schedule cloudlets inside the virtual machines, (5) cloudlets were generated from a standard formatted workload of a high performance computing center called HPC2N in Sweden [4].

Fig. 2 shows the time elapsed between submission to completion when executing a group of cloudlets at the same time on the available cloud resources for 4 different simulations, the first time using the simple brokering and the following 3 using the implemented PSO method. The simulation was developed for 4 different groups of cloudlets, the 1st, 2nd, 3rd and 4th group of cloudlets consisted of 20, 50, 100 and 200 cloudlets, respectively. The figure shows that the makespan was minimized when using PSO in most of the cases. The optimization values ranged from 46% to 51% improvement for the 20 cloudlets, 17% to 26% improvement for the 50 cloudlets, 14% to 20% for the 100 cloudlets, and from 4% to 11% for the 200 cloudlets.

Fig. 3 shows the decrease of the convergence of the global fitness value (the value recorded by the VMs) during the 1,000 iterations of the PSO algorithm. This shows how decreasing the size of the search space and increasing the number of iterations will increase the probability of finding more optimal solutions. Notably, an improvement to the solution for the 20 cloudlets problem was found during the last 100 iterations, with no improvements for approximately the prior 400 iterations. Such findings demonstrate the advantage of using RIW to update the inertia weight, which updates the velocity and position of particles in a way that increased the probability of finding a near-optimal solution even in the later iterations.

To ensure that the results were statistically significant, the Mann-Whitney test was used for the two different sets, where every value was the makespan of executing a workload of 100 cloudlets. The first numeric set was generated after using the regular CloudSim brokering while the other set was generated after using the proposed method. The two distributions were found to be significantly different with a significance level of $p \leq 0.5$. The regular CloudSim scheduling method was found to have constant fitness values for the same set of cloudlets, in contrast to the metaheuristic method, where fitness values were found to be adaptable and variable along the distribution.

The final test was prepared by running the simulation 100 times using the PSO method and reporting the average makespans in seconds and standard deviations were then calculated for the same 4 groups of cloudlets. The results are listed in Table I. The simple brokering of CloudSim was also tested and the output is listed in Table I. Standard deviations, in this case, are zero due to the fact that the simple broker gives the same execution sequences and makespans for the same set of cloudlets and cloud resources. Table I shows that though the proposed PSO integration was able to, on average, improve the makespan by up to 49%, the algorithm also suffers somewhat as the dimensionality of the algorithm increases. In other words, as there are more and more cloudlets to schedule, the algorithm begins to have some difficulty finding a solution that is more optimal than the typical scheduling algorithm. Also of note is that the standard deviations are fairly low, suggesting that the results should be highly repeatable.

Table I. RESULTS (IN SECONDS) OF RUNNING THE PSO METHOD FOR 100 TIMES

Function	20 Cloudlets	50 Cloudlets	100 Cloudlets	200 Cloudlets
Proposed Method	325.45 $\pm$ 12.22	712.32 $\pm$ 35.44	988.22 $\pm$ 30.42	1,462.91 $\pm$ 94.70
CloudSim	634.64 $\pm$ 0.00	902.75 $\pm$ 0.00	1,179.37 $\pm$ 0.00	1,521.06$\pm$ 0.00
Avg. Improvement	49%	21%	16%	4%

V. CONCLUSION

In this study, PSO was used to assign virtual machines to run different cloudlets as part of the broker in CloudSim tool. The results clearly show that PSO was able to minimize the makespan of the workload while excelling at optimizing the simulated scheduling results of CloudSim in a way that will maximize the utilization and minimize the costs of using the on demand cloud services. At the same time PSO, like any other metaheuristic method, does not give any guarantees on finding the most optimal solution.

Consequently, and as shown in Fig. 2, whenever the search space expands, the chance of finding an improved optimal solution becomes harder and harder. However, using the random inertia weight to give the particles the ability to find better solutions during late stages of the search was able to enhance the ability of the PSO method in exploring the problem space. All in all, PSO was able to improve makespan between 4% and 49%.

In the future, this work may be extended in various ways including 1) Considering other single- or multi-objective formulations including timespan, cost, revenue, and energy consumptions; 2) Integrating more realistic and advanced cloud computing concepts like VM and Task migration; and 3) Further optimizing the algorithm to scale effectively with larger workloads.

REFERENCES

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing-Recommendations of the National Institute of Standards and Technology. NIST," *The NIST Special Publication 800-145*, pp. 1–3, 2011.

[2] O. Morariu, C. Morariu, and T. Borangiu, "A genetic algorithm for workload scheduling in cloud based e-learning," *2nd International Workshop on Cloud Computing Platforms*, pp. 1–6, 2012.

[3] R. N. Calheiros, R. Ranjan, A. Beloglazov, and C. A. F. De Rose, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software Practice and Experience*, vol. 41, no. 1, pp. 23–50, August 2010.

[4] Dror G. Feitelson and Dan Tsafrir and David Krako, "Experience with using the Parallel Workloads Archive," *Journal of Parallel and Distributed Computing*, vol. 7, no. 1, pp. 2967–298, 2014.

[5] E. Mokoto, "Scheduling to minimize the makespan on identical parallel Machines: an LP-based algorithm," *Investigacion Operative*, pp. 97– 107, 1999.

[6] P. Pongchairerks, "Particle swarm optimization algorithm applied to scheduling problems," *ScienceAsia*, vol. 35, pp. 89–94, 2009.

[7] W.-n. Chen and J. Zhang, "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 1, pp. 29–43, 2009.

[8] S. Pandey, L. Wu, S. Guru, and R. Buyya, "A Particle Swarm Optimization (PSO)-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," in *IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 400 – 407.

[9] A. Abraham, R. Buyya, and B. Nath, "Nature's Heuristics for Schedul- ing Jobs on Computational Grids," in *IEEE Int'l Conf. on Advanced Computing and Communications*, 2000, pp. 45–52.

[10] M. Rana, S. K. KS, and N. Jaisankar, "Comparison of Probabilistic Optimization Algorithms for Resource Scheduling in Cloud Computing Environment," *International Journal of Engineering and Technology*, vol. 5, no. 2, pp. 1419–1427, 2013.

[11] L. Zhang, Y. Chen, and R. Sun, "A task scheduling algorithm based on pso for grid computing," *International Journal of Computational Intelligence Research*, vol. 4, no. 1, pp. 37–43, 2008.

[12] S. Mirzayi and V. Rafe, "A survey on heuristic task scheduling on dis- tributed systems," *AWERProcedia Information Technology & Computer Science*, vol. 1, pp. 1498–1501, 2013.

[13] A. Bardsiri and S. Hashemi, "A Review of Workflow Scheduling in Cloud Computing Environment," *Int'l Journal of Computer Science and Management Research*, vol. 1, no. 3, pp. 348–351, 2012.

[14] S. Zhan and H. Huo, "Improved PSO-based Task Scheduling Algorithm in Cloud Computing," *Journal of Information & Computational Science*, vol. 13, pp. 3821–3829, 2012.

[15] H. Zhang, P. Li, Z. Zhou, and X. Yu, "A PSO-Based Hierarchical Resource Scheduling Strategy on Cloud Computing," *Trustworthy Computing and Services*, pp. 325–332, 2013.

[16] Y. Wang, J. Wang, C. Wang, and X. Song, "Research on Resource Scheduling of Cloud Based on Improved Particle Swarm Optimization Algorithm," *Advances in Brain Inspired Cognitive Systems*, pp. 118– 125, 2013.

[17] Z. Wang, K. Shuang, L. Yang, and F. Yang, "Energy-aware and revenue-enhancing Combinatorial Scheduling in Virtualized of Cloud Datacenter," *Journal of Convergence Information Technology*, vol. 7, no. 1, pp. 62–70, January 2012.

[18] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, and N. Sharma, "Towards autonomic workload provisioning for enterprise Grids and clouds," in *IEEE/ACM International Conference on Grid Computing*. IEEE, October 2009, pp. 50–57.

[19] N. H. Centers, "Getting Familiar with Cloud Terminology Cloud Dictionary," vol. 1, no. 1, pp. 1–7, 2013.

[20] S. Chapin, W. Cirne, and D. Feitelson, "Benchmarks and standards for the evaluation of parallel job schedulers," *Job Scheduling Strategies for Parallel Processing*, pp. 1–24, 1999.

[21] Y. Cao, C. Ro, and J. Yin, "Comparison of Job Scheduling Policies in Cloud Computing," *Future Information Communication Technology and Applications*, vol. 235, pp. 81–87, 2013.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," *nternational Conf. on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[23] J. Kennedy and R. Eberhart, "A Discrete Binary Version Of The Particle Swarm Algorithm," 1997, pp. 4104 – 4108.

[24] J. L. Pierobom, M. R. Delgado, and C. A. Kaestner, "Particle swarm optimization for task assignment problem," *Brazilian Congress on Computational Intelligence*, vol. 10, pp. 1–8, 2011.

[25] H. Izakian, B. T. Ladani, K. Zamanifar, and A. Abraham, "A novel par- ticle swarm optimization approach for grid job scheduling," *Info. Systems, Technology and Management – Communications in Computer and Information Science*, vol. 31, pp. 100–109, 2009.

[26] K. Deep and Madhuri, "Application of Globally Adaptive Inertia Weight PSO to Lennard-Jones Problem," *International Conference on Soft Computing for Problem Solving*, pp. 31–38, 2012.

[27] S. Sivanandam and P. Visalakshi, "Multiprocessor scheduling using hybrid particle swarm optimization with dynamically varying inertia," *International Journal of Computer Science & Applications*, vol. 4, no. 3, pp. 95–106, 2007.

[28] R. Ojha and M. Das, "An Adaptive Approach for Modifying Inertia Weight using Particle Swarm Optimisation," *IJCSI International Jour- nal of Computer Science Issues*, vol. 9, no. 5, pp. 105–112, 2012.

[29] L. Chong-min, G. Yue-lin, and D. Yu-hong, "A New Particle Swarm Optimization Algorithm with Random Inertia Weight and Evolution Strategy," *Journal of Communication and Computer*, vol. 5, no. 11, pp. 42–48, 2008.

[30] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimiza- tion," in *Congress on Evolutionary Computation*, Washington, DC, pp. 1945–1950.

[31] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Congress on Evolutionary Computation*, vol. 1, no. 7. IEEE, 2000, pp. 84–88.

[32] Wikipedia, "Instructions per second," p. 1, 2013.

[33] Barham, Paul and Dragovic, Boris and Fraser, Keir and Hand, Steven and Harris, Tim and Ho, Alex and Neugebauer, Rolf and Pratt, Ian and Warfield, Andrew, "Xen and the Art of Virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, October 2003.