MATEC Web of Conferences **28**, 06001 (2015)
DOI: 10.1051/matecconf/20152806001

Particle Swarm Optimization with Time Varying Parameters for Scheduling in Cloud Computing

Zhao Shuang [1,2] Lu Xianli[1] Li Xuejun[3]

[1]*Beijing Institute of Tracking and Telecommunication Technology, Beijing, 100094, China.*
[2]*The College of Postgraduate, Academy of Equipment, Beijing, 101416, China.*
[3]*Department of Information Equipment, Academy of Equipment, Beijing, 101416, China.*

Abstract. Task resource management is important in cloud computing system. It's necessary to find the efficient way to optimize scheduling in cloud computing. In this paper, an optimized particle swarm optimization (PSO) algorithms with adaptive change of parameter (viz., inertial weight and acceleration coefficients) according to the evolution state evaluation is presented. This adaptation helps to avoid premature convergence and explore the search space more efficiently. Simulations are carried out to test proposed algorithm, test reveal that the algorithm can achieving significant optimization of makespan.

1 Introduction

Cloud computing has gained popularity in recent years, which involves virtualization, parallel and distributed computing, networking, software and web services etc. Cloud computing is a type of distributed and parallel system which consists of a collection of internet-based virtualized computers and consists of elements such as clients, datacenter and distributed servers 1.Cloud computing is an emerging technology and has the high performance and it means that you pay as you really need. The scheduling is very significant in the Cloud computing for using the distributed resources efficiently. Because of its theoretical and practical significance it has been attracting more and more attention from researchers. The scheduling problem is an NP-hard optimization problem. Scheduling algorithm is the method by which threads, processes or data flows are assigned to system resources (e.g. VM, processor, communications bandwidth). The speed, efficiency, utilization of resources in optimized way depends largely on the type of scheduling algorithm selected for the cloud computing environment 2.

The rest of this paper is organized as follows. Section II gives a brief review on the original PSO algorithms and scheduling with PSO. In Section III, the proposed PSO is presented. Comparison studies are shown in Section IV and the conclusions are finally summarized in Section V.

2 PSO and scheduling with PSO

Particle Swarm Optimization is a typical intelligent optimization algorithm introduced by Kennedy and Eberhart 3. After that a number of modified PSO algorithms have been proposed to enhance efficiency performance of the algorithm. To balance the exploration (global search) and exploitation (local search) ability of PSO , inertia weight is added to the original one by Shi，Y. and Eberhart4 and this is regarded as the standard algorithm. Particles update velocity and position in each iteration according to the formula as shown in Eq.(1) and (2).

$$v_i(t+1) = wv_i(t) + c_1r_1(t)(p_i(t) - x_i(t)) + c_2r_2(t)(p_g(t) - x_i(t)) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

where $v_i(t)$ is the velocity of particle *i*. $p_i(t)$ is the best position yielded by the *i*th particle and $p_g(t)$ is the best position so far obtained by the whole swarm. w is inertia weight. c_1 and c_2 are acceleration coefficients. c_1 is called the cognitive acceleration coefficient, while c_2 the social acceleration coefficient. r_1, r_2 are random number in the interval of [0 1]. A number of modified PSO algorithms have been proposed to achieve better optimization by adaptive control of parameters and combination with auxiliary search operators. Most of time-varying controlling strategies for the parameters are based on the iterations number. Parameters are linearly decreasing4 or nonlinear modified567. The parameters sometimes are dynamically adjusted with a fuzzy system using fitness feedback. Some use a self-adaptive method by encoding the parameters into the particles and optimizing them together with the position during run time7. Strategies based on generation number have improved efficiency of the algorithm greatly, but they may run into the risk of adjusting the parameters inappropriately without

[a] Corresponding author: zhsh2002@163.com

evolutionary state information that reflects the swarm distribution or assembling state and fitness values.

Due to its simplicity, fast convergence, ease of implementation and effectiveness, PSO has drawn much attention from various fields and has become popular in wide range of application. A substantial number of PSO-based algorithm has been applied to solve Scheduling problem and task allocation 89.Chen and Wang 10 proposed PSO for task scheduling in heterogeneous grid to minimize application completion time. Zhan et al. 11combine PSO with annealing algorithm to improve search speed. Zhang et al12 applied revised discrete PSO for workflow Scheduling to minimizing data transmission and computation cost. Liu et al.13 proposed a scheme to solve meta task scheduling problem. In this approach, fuzzy matrices are used to represent positions and velocities of particles in PSO. The elements in each matrix represent fuzzy relations between grid nodes and jobs. Suraj Pandey et al. 14 present a particle swarm optimization (PSO) based heuristic to schedule applications to cloud resources that takes into account both computation cost and data transmission cost.

3. Proposed PSO

3.1 Particle encoding method

To apply PSO successfully to scheduling problems, how to map the particle position to the problem solution is important and must be considered first. In this paper, a particle is expressed as a N（N is the number of tasks） dimensions vector and each dimension represents a task. Value of the particle in each dimension represents the resource number that a task assigned to. The solution of each dimension of particles keeps changing in search process. The position of a particle represents a potential scheduling, the position can be ‘‘decoded’’ to obtain a feasible solution 15.

One of the particles is constructs as shown in Table 1.The particle has 5 dimensions, representing 5 tasks, and the value of the particle in each dimension represents the number of resource which the task assigns to. The particle is coded as shown in Table 2.

Table1. Task-resource allocation

task	1	2	3	4	5
resource	3	1	2	3	2

Table 2. Particle encoding

3	1	2	3	2

3.2 Fitness function

The scheduling target is to find optimal task-resource assignment and hence minimize application complete time. There are several objectives can be measured for the mapping of tasks to resources, such as makespan, cost, Qos etc. In this paper the scheduling target only focus on the minimizing of makespan .

The expected execution time ET_{ij} of task t_i on machine m_j is defined as the time taken by m_j to execute t_i given that m_j has no load when t_i is assigned. In this paper, the task execution time is represented by the quotient of task length of t_i and computing power of resource m_j:

$$ET_{ij} = Task\ length\ of\ task\ i\ /\ Computing\ power\ of\ resource\ j \tag{3}$$

The makespan is the amount of time that from the first tasks start running to the last tasks finished. And from the view of resource pool, the makespan is the maximum running time of all the resource to finish tasks that assigned to it. Short makespan mean that the scheduling is optimized.

$$CT = \underset{j=1}{\overset{m}{Max}} \sum_{i=1}^{n} ET_{ij} \tag{4}$$

where ***m*** is the resource number and ***n*** is the task number that execution on resource ***j***.

The objective function *f (x)* is mathematically stated as to minimizing the makespan of all tasks:

$$f(x) = Min(CT) \tag{5}$$

3.3 Proposed PSO algorithm

Due to its simple concept and effectiveness, the PSO has become a popular optimizer and has widely been applied in practical optimization problem, but at the same time it’s easy to get trapped in the local optima and premature convergence. Much research on performance improvements has been presented, including dynamic modified parameters, topological structures and combination with auxiliary operations. This paper present time varying parameters of inertia weight and acceleration coefficient to modify the learning strategies of particles. When swarms trapped into premature convergence, introduce “mutation” to the velocity and position updating strategy to enhance the global search .The algorithm is shown in Fig. 1, whose structure is similar to the original PSO.

1) swarm initialization
2) **while**(termination condition is not meted)
3) calculate new fitness, average fitness value, update *pbest* and *gbest*
4) adjust parameters(w_t,$c1_t$,$c2_t$, *AccFator*, *AssFactor*)
5) updating new velocity and the position
6) **end while**
7) output the scheduling scheme and its fitness value

Figure 1. the general adaptive algorithm

where *AccFator* denote convergence speed and *AssFactor* denote the aggregation degree of swarm. In the evolution process the best fitness of this iteration is better than that of last iteration, the quotient of this two value (AccFator) can reflect the convergence speed. *AccFator* is in the range (0, 1], when *AccFator* is 1, the evolution is in the state of convergence. The quotient of the best fitness and average fitness (*AssFactor*) of the swarm can

reflect the population distribution state. *AssFactor* is also in the range (0, 1]. At an early stage, *AssFactor* value is small, the distribution is dispersive. When particles cluster together and converge to a locally or globally optimal area, the *AssFactor* value is 1.

$$AccFactor = F(gbest_T) / F(gbest_{T-1}) \tag{6}$$

$$AssFactor = F(gbest_T) / avgF(X_T) \tag{7}$$

$F(gbest_T)$ is the best fitness of this iteration, $F(gbest_{T-1})$ is the best fitness of last iteration and $avgF(X_T)$ is the average fitness of this iteration.

3.3.1 Inertia weight

The parameter of inertia weight is critical for the convergence behavior of PSO. The inertia weight *w* controls the influence of the previous velocity on the current one. Its value is usually initialized a little big value around 0.9 and gradually decreasing towards a small value around 0.4 as the evolution progresses. Meanwhile, a better strategy is to apply adaptive approaches in which *w* is adaptively modified according to the evolution state of the particles. In order to improve the convergence of PSO, a time variant *w* is used in this paper as shown in Eq.(8).The value of *w* is allowed to change with the convergence speed and evolution state of particles16.

$$w = w_{ini} - dAccFactor * w_{Acc} + dAssFactor * w_{Ass} \tag{8}$$

where *AccFator* and *AssFactor* are as shown in Eq.(6) and (7).w_{ini} is the initial value of *w*, set as 1.w_{Acc} set as 0.5 and w_{Ass} set as 0.2.

3.3.2 Acceleration coefficients

Though fuzzy adaptive inertia weight can balance the global and local search and fine tune the optimum solution, acceleration coefficients are also important for PSO. Parameter c1 represents the "self-cognition" that pulls the particle to its own historical best position, helping explore local niches and maintaining the diversity of the swarm. Parameter c2 represents the "social influence" that pushes the swarm to converge to the current globally best region, helping with fast convergence7.Many lectures are proposed to improve algorithm by time varying acceleration coefficients, As shown in Eq.(9) and (10) ,c1 and c2 is linear changing with iteration 56 1517 as shown in Eq.(9) and (10).Adaptively controlled acceleration coefficients according to the evolutionary state are proposed in7.

$$c1 = (c_{1f} - c_{1i}) \frac{iter}{MAXITER} + c_{1i} \tag{9}$$

$$c2 = (c_{2f} - c_{2i}) \frac{iter}{MAXITER} + c_{2i} \tag{10}$$

where c_{1f}, c_{1i}, $c_{2f}$and c_{2i} are constants, iter is the current iteration number and MAXITER is the maximum number of allowable iterations.
In this paper a time variant c1 and c2 is presented as shown in Eq.(11) and (12).

$$c1 = c1_{ini} - AccFator * c_{Acc} - AssFactor * c_{Ass} \tag{11}$$

$$c2 = c2_{ini} - AccFator * c_{Acc} + AssFactor * c_{Ass} \tag{12}$$

where $c1_{ini}$ is 2.5, $c2_{ini}$ is 1.5, c_{Acc} and c_{Ass} are 0.5(range from 0.4 to 0.6).*AccFator* and *AssFactor* are shown in Eq. (6) and (7).

3.3. 3 Velocity and Position Updating

Velocity represents the moving direction and trends of a particle. Mutation is a genetic operator that alters one or more gene values in chromosome of genetic algorithm. Mutation is important in genetic search progress as it helps to prevent the population from trapped into local optima 18. In this paper, when swarms trapped into convergence, adaptive mutation operator is added to the velocity and position updating strategy of the particles to enhance the global search capability and avoid premature convergence. The velocity and position updating rules are as followings:

$$v_i(t+1) = wv_i(t+1) + c_1 r_1(t)(p_i(t) - x_i(t)) + c_2 r_2(t)(p_{gi}(t) - x_i(t)) - c_3((p_i(t) + p_g(t))/2) \tag{13}$$

$$x_i(t+1) = x_i(t+1) + r_3 p_{\max} / 3 \tag{14}$$

where $p_{ij}(t)$ is particle best position, $p_{gi}(t)$ is global best position. p_{max} is maximum position of the particles.c_3 is mutation factor. r_3 are random number in the interval of [0 1].

1) **For** each particle i(i=1.2…m)
2) Velocity updating according to Eq. (1)
3) Position updating according to Eq. (2)
4) **If**(convergence condition is met)
5) Velocity updating according to Eq. (13)
6) Position updating according to Eq. (14)
7) **End if**
8) **End for**

Figure 2. Updating velocity and Position

4 Experiment and Result

The scheduling algorithm is tested on CloudSim. Simulations are carried out to observe the quality of the optimum solution and the rate of convergence of the new methods. In order to compare the performance of the algorithms, **s**equential assignment scheduling algorithm that provided by CloudSim, standard PSO, time varying

parameter PSO and time varying parameter PSO with mutation are tested.

The number of iterations is 100 and the particle number is 30.The task length is generated randomly ranging in the interval [1000, 8000] MIPS (million instructions per second).The resource number is 10 and 20,computing power ranging from 100 to 800 MIPS. Test results are averaged over 30 runs in order to maintain the reliability of the results. Experimental parameters setting are shown in Table 3.

Table 3. Parameters setting of the algorithm

Algorithm	Inertial weight(w)	Acceleration coefficient C1	Acceleration coefficient C2	mutation
STD_PSO	Linear decreasing from 0.9 to 0.4	2	2	None
TV_PSO	Adaptive change as proposed in Eq.(8)	Adaptive change as proposed in Eq.(11)	Adaptive change as proposed in Eq.(12)	None
MTV_PSO	Adaptive change as proposed in Eq.(8)	Adaptive change as proposed in Eq.(11)	Adaptive change as proposed in Eq.(12)	Velocity and position as proposed in Eq.(13) and (14).c3 is 0.5

4.1 Performance comparison

Table 4 and Figure.3 shows the average makespan values of tasks, the task number is range in [10,100]. From the results it is evident that the makespan produce by MTV_PSO algorithm is the least in comparison with other approaches. The performance of TV_PSO algorithm and standard PSO is similar and TV_PSO is a little better than standard PSO. They are all better than Sequential Assignment.

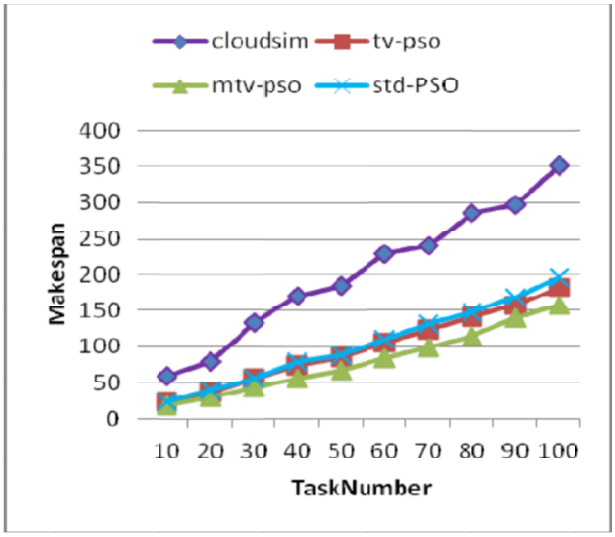
Table 4 Comparison of Task Completion Time on 10 and 20 resources

(a) The resource number is 10

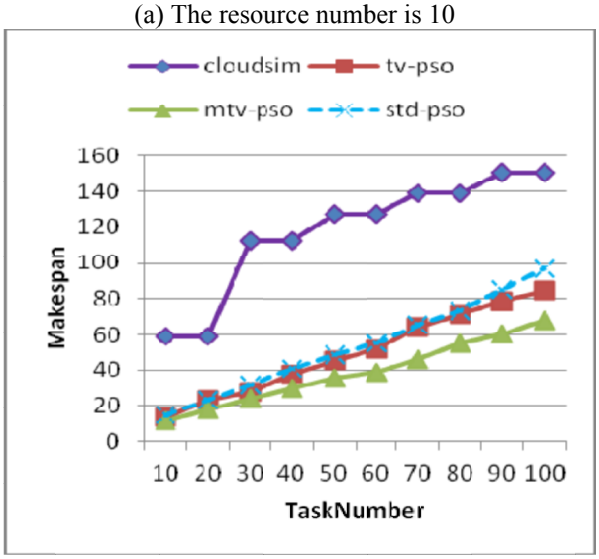
Task NO	Sequential Assignment	TV_ PSO	MTV_ PSO	STD_ PSO
10	59.19	23.01	19.01	24.33
20	79.67	37.21	30.06	40.56
30	132.8	56.57	42.69	56.16
40	170.69	74.45	55.98	79.53
50	185.47	86.3	67.22	88.4
60	228.08	105.29	83.38	109.59
70	240.28	122.31	98.16	131.49
80	285.88	141.77	114.69	147.5
90	297.2	157.47	139.97	168.65
100	350.34	181.43	156.85	195.87

(b) The resource number is 20

Task NO	Sequential Assignment	TV_ PSO	MTV_ PSO	STD_ PSO
10	59.19	13.44	11.34	14.68
20	59.19	23.25	18.21	23.38
30	112.31	27.77	23.54	31.34
40	112.31	37.25	29.92	40.71
50	127.1	45.52	35.6	48.64
60	127.1	52.08	38.99	55.04
70	139.29	64.27	45.96	65.61
80	139.29	71.05	54.95	73.12
90	150.61	78.45	60.14	85.31
100	150.61	84.49	68.02	96.69



(a) The resource number is 10



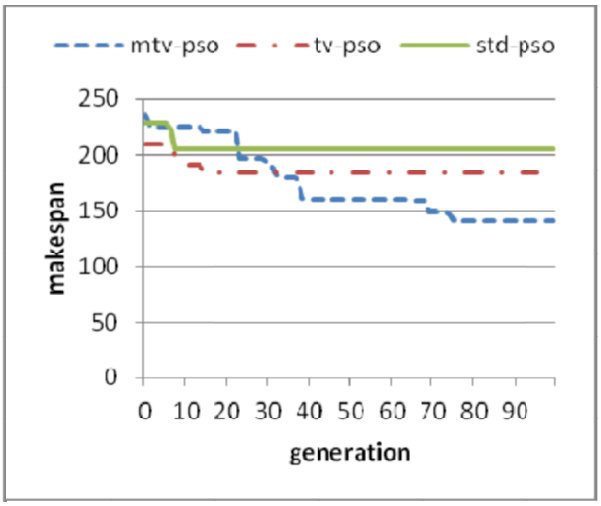
(b) The resource number is 20

Figure 3. Performance of job scheduling

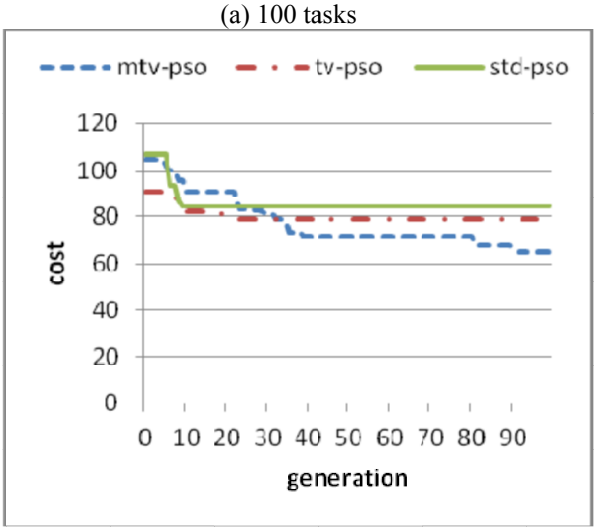
4.2 Comparisons on the Convergence Speed

Figure.4 plots the convergence of total makespan computed by PSO over the number of iterations. As the algorithm progresses, the standard PSO convergence is drastic and it finds a global minima very quickly, but it trapped premium convergence and can't jump out. The

number of iterations needed for the convergence is about ten. The TV-PSO convergence is similar to standard PSO.MTV_PSO convergence is slow, but when it trapped premium convergence, it can jump out by mutation to the velocity and position and get better optimization.



(a) 100 tasks



(b)50tasks

Figure .4 The trend of convergence of PSO

5. Conclusion

This paper presents a modified PSO to achieve the scheduling goals of minimizing the completion time of application on Cloud computing environments. To overcome the premature convergence and explore the search space more efficiently, a dynamic changing inertia weight and acceleration coefficients are applied. At the same time, when search trapped local optimums, introduced mutation is more effective for particles jumping out. The results obtained by our heuristic are compared against PSO and heuristic that provided by Clouds. We found that modified PSO based task resource scheduling performs best as compared to other algorithms in this paper.

References

1. Swarnkar, N., Singh, A. P. A. K., & Shankar, R. (2013). A Survey of Load Balancing Techniques in Cloud Computing. *International Journal of Engineering*, *2*(8),800-804
2. Lin, C. T. (2013). Comparative Based Analysis of Scheduling Algorithms for Resource Management in Cloud Computing Environment. *International Journal of Computer Science and Engineering*, *1*(1), 17-23.
3. Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (Vol. 4, No. 2, pp. 1942-1948).
4. Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference on* (pp. 69-73). IEEE.
5. Chaturvedi, K. T., Pandit, M., & Srivastava, L. (2009). Particle swarm optimization with time varying acceleration coefficients for non-convex economic power dispatch. *International Journal of Electrical Power & Energy Systems*, *31*(6), 249-257.
6. Tripathi, P. K., Bandyopadhyay, S., & Pal, S. K. (2007). Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. *Information Sciences*, *177*(22), 5033-5049.
7. Zhan, Z. H., Zhang, J., Li, Y., & Chung, H. H. (2009). Adaptive particle swarm optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *39*(6), 1362-1381.
8. Liu, H., Abraham, A., & Hassanien, A. E. (2010). Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. *Future Generation Computer Systems*, *26*(8), 1336-1343.
9. Xue, S. J., & Wu, W. (2012). Scheduling workflow in cloud computing based on hybrid particle swarm algorithm. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, *10*(7), 1560-1566.
10. Chen, R. M., & Wang, C. M. (2011, February). Project scheduling heuristics-based standard PSO for task-resource assignment in heterogeneous grid. In *Abstract and Applied Analysis* (Vol. 2011). Hindawi Publishing Corporation.
11. Zhan, S., & Huo, H. (2012). Improved PSO-based Task Scheduling Algorithm in Cloud Computing. *Journal of Information & Computational Science*, *9*(13), 3821-3829.
12. Wu, Z., Ni, Z., Gu, L., & Liu, X. (2010, December). A revised discrete particle swarm optimization for cloud workflow scheduling. In *Computational Intelligence and Security (CIS), 2010 International Conference on* (pp. 184-188). IEEE.
13. Liu, H., Abraham, A., & Hassanien, A. E. (2010). Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. *Future Generation Computer Systems*, *26*(8), 1336-1343.
14. 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.

15. Ratnaweera, A., Halgamuge, S., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *Evolutionary Computation, IEEE Transactions on*, *8*(3), 240-255.
16. Xuanping, Z., Du Yuping, Q. G., & Zheng, Q. (2005). Adaptive Particle Swarm Algorithm with Dynamically Changing Inertia Weight. Journal of xi'an jiaotong university, (39, 10), 1039-1042.
17. Mohammadi-Ivatloo, B., Rabiee, A., Soroudi, A., & Ehsan, M. (2012). Iteration PSO with time varying acceleration coefficients for solving non-convex economic dispatch problems. *International Journal of Electrical Power & Energy Systems*, *42*(1), 508-516.
18. Masrom, S., Abidin, S. Z., Omar, N., & Nasir, K. (2013). Time-Varying mutation in particle swarm optimization. In Intelligent Information and Database Systems (pp. 31-40). Springer Berlin Heidelberg.
19. 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