Asian Journal of Information Technology 15 (19): 3703-3707, 2016
ISSN: 1682-3915

A Task Scheduling Based on Hybrid Self Organizing Migrating Algorithm in Cloud Environment

[1]B. Gomathi and [2]Karthikeyan krishnasamy
[1]Department of IT, Hindusthan College of Engg. and Tech., Coimbatore, India
[2] Department of IT, Sri Krishna College of Engg. and Tech., Coimbatore, India

Abstract: Cloud computing aspires to share a pool of virtualized resources. On the contrary, Task scheduling is an indispensable issue in cloud computing. To map the tasks of users to virtualized resources, this study advises Hybrid V-SOMA (Self Organizing Migrating Algorithm and Variable Neighborhood Search) algorithm to balance the load of the virtualized resources in the process of task scheduling. On bottom of the simulation results produced by cloudsim, this algorithm can drastically reduce the makespan of task scheduling as well as execute the good load balancing compared to other scheduling algorithms like PSO,SOMA etc.

Key words: Balance, load, resources, cloudsim, execute

INTRODUCTION

Cloud computing system (Dikaiakos *et al*., 2009) is a new paradigm of parallel and distributed system and it affords virtualized computing and storage technologies as a service to the end users by low cost on a demand basis. One of the main exigent issues in cloud environment is dispatching the tasks of users to virtualized resources in an dexterous manner which belongs to task scheduling. Task scheduling is one of the problems in cloud computing. Since, it is tough to find optimal solution in polynomial time for the task scheduling in cloud environment, it is said to be NP-Complete problem. Now a day's numerous heuristic algorithms like GA, PSO, ACO etc are extensively proposed by many researchers to resolve NP-Completeness of the scheduling problems. In order to exploit the cloud resources effectively, SOMA (Self Organizing Migration Algorithm) and Variable Neighborhood Search (VNS), named V-SOMA algorithm is proposed for task scheduling in cloud environment. Further, the strictures of SOMA are redefined to make this algorithm more apposite for task scheduling problem. In the scheduling process, each task is owed to pertinent resources with minimum completion time over all available resources. So V-SOMA algorithm minimizes the makespan of assigned tasks and enhances the performance by balancing the load among the virtual machines in cloud environment. To appraise this proposed algorithm, an extensive cloud simulation platform was developed based on cloudsim (Calheiros *et al*., 2011) toolkit package. The tentative results show that the proposed V-SOMA radically outperforms when compared to PSO and SOMA in terms of performance.

MATERIALS AND METHODS

Task scheduling model: In order to conform to the parallel and distributed cloud environment, we conquest the dynamic batching mode, where tasks are collected in a set that is analyzed for mapping in preference to mapping tasks into resources as they arrive. In accordance to the collected information such as real status of resources and task details, we can devise more rational scheduling strategy to balance the load across resources. In this study, SOMA-VNS scheduling algorithm is proposed to recuperate the resource exploitation in cloud environment.

In this study, we deem that cloud environment has heterogeneous resources with diverse processing capability. The processing time of task may show a discrepancy according to the task scheduling on different resources. The following metrics are used to measure the performance of proposed algorithm. First metric is the makespan which is delineated as the finishing time of last task for the solution s as follows:

$$\text{minimize } f(s) = Fmax(s) \quad (1)$$

Where:
f(s) = The objective function
Fmax(s) = The finishing time of the last task (also called makespan). The smaller makespan shows the better feat of the algorithm

Another metric is Load Balancing Index (β) which is computed to gauge the deviations of load on VMs as follows (Kruekaew and Kimpan, 2014).

Corresponding Author: B. Gomathi, Department of IT, Hindusthan College of Engg. and Tech., Coimbatore, India

$$\beta=\sqrt{\frac{\sum_{i=1}^{n}(Li-\bar{L})}{n}} \quad (2)$$

Where:

n	=	The number of virtual machines
Li	=	The load of the virtual machine
i and $\bar{L}$	=	The average load of all virtual machines
The smaller β	=	The better load balancing in cloud environment

To augment the performance of task scheduling in cloud, we hub on balancing the load across VMs to minimize makespan in cloud environment

Self Organizing Migrating Algorithm (SOMA): SOMA is a population based stochastic optimization algorithm which is modeled on self-organizing, competitive and co-operative social behavior of animals searching together to find the food source. By the haphazard initialization, population is arbitrarily distributed over the search space and this algorithm has latent to converge towards the global optimum. Instead of engendering new individuals in search, positions of the individuals are changed based on two operations such as perturbation and migration. At each iteration, the population is weighed up and the individual with highest fitness is considered as leader. The other individuals will traverse towards the leader in n steps of defined length. PRT (expanded as Perturbation) factor is used to perturb this path randomly. Prior to an individual proceeds towards leader, PRT vector whose range is (0, 1) is created. An individual moves towards a leader using the following equation:

$$X_{i,j}^{MLnew}=X_{i,j}^{ML}+(X_{L,J}^{ML}-X_{i,j}^{ML})tPRTVector_j \quad (3)$$

Where:

t <0,byStepto,Pathlength>

ML	=	The actual migration loop
$x_{i,j}^{MLnew}$	=	The new position of an individual
$x_{i,j}^{ML}$	=	The position of an active individual
$x_{L,j}^{ML}$	=	The position of leader

Akin to other evolutionary algorithm, SOMA algorithm is impinged on by the premature convergence due to the faster convergence feature. For that reason, Variable Neighborhood Search (VNS) is utilized by SOMA to build on the searching ability in solution space.

Variable Neighborhood Search (VNS): Variable Neighborhood Search (VNS) is a simple and effective meta-heuristic approach proposed by Mladenovic and Hansen (Mladenovic and Hansen, 1997). It is one of the most prevalent procedure because of its ease of use and triumph in problem solving. The efficiency of this algorithm depends on the neighbouring structure with which the neighbouring solutions can be unwavering to move further. VNS endows with this structure with which one can escape from local optima by others. For task scheduling problem, the following two neighbouring structures are used owing to the simplicity and ease of use features. Exchange is the function which is used to swap any two randomly selected tasks between uth and vth dimensions where u≠v. Insert is another fine-tuning function that is used to remove the task at uth dimension and insert it in the vth dimension where u≠v. Pseudo code of VNS local search (Tasgetiren *et al*., 2004).

Algorithm 1:

Get best solution, x_0S
Set outloop=0
while(outloop<(P_{ls}*PopSize)) {
 Set u=rnd(1,n) and v=rnd(1,n)
 Assign x_1=insert(x_0,u,v)
 Set inloop=0
 while(inloop<(n*(n-1))) {
  Set k=0 and K_{max}=2
  while(k< K_{max}) {
   Set u=rnd(1,n) and v=rnd(1,n)
   If(k=0) then x_2=insert(x_1,u,v)
else if(k=1) then x_2=interchange(x_1,u,v)
   if f(x_2)<f(x_1) then x_1= x_2 and k=0
   else k=k+1 }
 inloop=inloop+1 }
 outloop=outloop+1
 if f(x_1)<=f(x_0) then x_0= x_1 }

Where n is the number of tasks in task scheduling problem and u and v are the integer random numbers between 1 and n. The task sequence of global paramount solution is represented as S^k, on which the VNS local search is applied to get the unsurpassed solution. π = insert(π_0,u,v) means removing task from uth dimension in the sequence π_0 and insert it in the vth dimension in the sequence π_0 that provides a sequence π The probability of the local search P is taken as 10%. To improve individuals in the population as well as to shun premature convergence in task scheduling problem, VNS local search is used with SOMA algorithm.

Hybrid V-SOMA: This section imparts a hybrid SOMA, where V-SOMA draws on Iterated VNS algorithm which is applied to the sequence of global preeminent solution at each iteration in V-SOMA. VNS algorithm is used to spurn poor local optima by changing neighborhood

structures as well as to uphold the diversity of the population. This hybrid algorithm destines to balance the load across the resources in the cloud environment while the makespan of the system is taken into account. While applying V-SOMA algorithm to task scheduling problem, each individual in the population should map with one of the solution in the problem. For the crisis in n tasks, the dimension of the individual is n and content of each cell in the individual represents resource allocation to that task. After initializing the population randomly, V-SOMA is sustain the preliminary population throughout the migration process and only the positions of the individual are changed. Pseudo code for V-SOMA is given in Algorithm 2.

Algorithm 2:

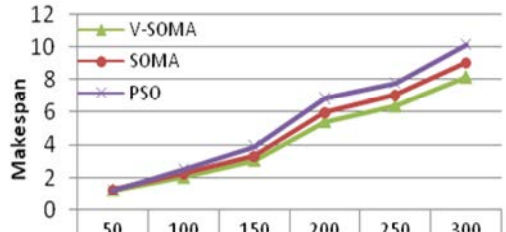
Input the scheduling problem with n tasks and m VMs
Output an optimized VM allocation to the specified task set
Generate initial population randomly
Evaluate individuals in the population using fitness function
Select best individual as leader
Create PRT vector for all individuals
All individual except leader migrate towards the leader using equation() until individual reaches the final position given by pathlength
For each step, fitness is evaluated at each position and move individual to the new position if it gives best fitness value
Apply VNS procedure to improve the leader
If max iteration is reached or stopping criteria is satisfied, then stop. Otherwise repeat from step4
Report the best individual solution as optimal solution.

RESULTS AND DISCUSSION

Experimental evaluation: Cloudsim is a cloud simulation tool which is used to estimate the projected task scheduling algorithm. The pragmatic analysis is conducted on machine which runs cloudsim, with intel Pentium dual-core processor 2.66 GHz, 4 GB memory configuration and Cloudsim can be used to construct VMs as resources to persuade the user prerequisite by data center and cloudlets are crafted from the workload of high performance computing center called HPC2N(8). Then, the cloudlets are premeditated with help of the proposed V-SOMA scheduling algorithm which is employed as part of cloud broker in cloudsim and judged against with well-known algorithms like PSO and SOMA algorithm using the following metrics: makespan and load balancing indexing. Table 1 shows the parameter settings of the above three algorithms. Here, PSO (9) used constraint random inertia weight to boost the global searching knack. This simulation experiment is used to scrutiny the above metrics for V-SOMA, SOMA and PSO algorithms with 50-300 tasks in 20 or 50 virtual machines, respectively.

Table 1: Parameter settings for PSO, SOMA and V-SOMA algorithms

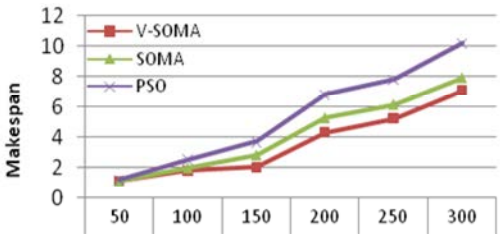
Algorithm	Parameters
PSO	Number of particles = 100
	acceleration coefficient c1 = c2 = 1.49
SOMA and V-SOMA	Number of individuals = 100
	Path length = 3
	Step size = 0.23
	PRT = 0.1



	50	100	150	200	250	300
V-SOMA	1.22339	2.04585	3.08307	5.45053	6.41585	8.17136
SOMA	1.26208	2.25211	3.29643	6.00106	7.07579	9.03102
PSO	1.20973	2.47982	3.91338	6.83735	7.80194	10.1648

Number of Tasks

Fig. 1: Average makespan using 20VM



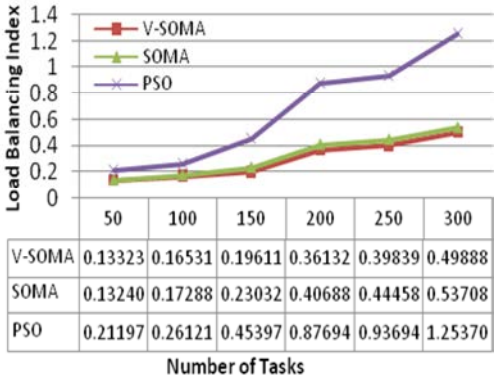
	50	100	150	200	250	300
V-SOMA	1.13323	1.79694	2.03031	4.31640	5.26242	7.08056
SOMA	1.13517	2.00694	2.83673	5.27128	6.14352	7.89978
PSO	1.20825	2.53848	3.70555	6.81672	7.81241	10.1801

Number of Tasks

Fig. 2: Average makespan using 50VM

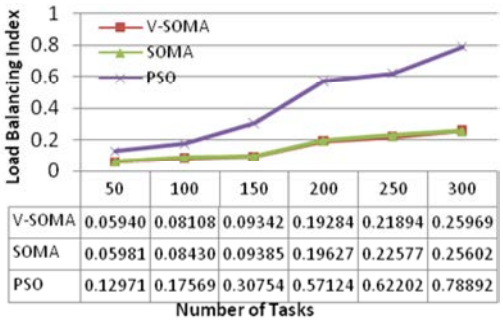
Makespan: We ordeal the average of makespan of three algorithms by running simulation for 10 times with 50 - 300 tasks in 20 or 50 virtual machines as shown in Fig.1.

Figure 1 and 2 shows that the makespan acquired from V-SOMA is lower than other two algorithms because VNS local search algorithm in V-SOMA progresses the individuals in the population. This leads to optimal resource allocation for all the tasks. Based on the annotations, we can easily know their performance. For example, for small datasets, the convergence speed of V-SOMA is quite closer to other algorithms. However, as the number of tasks increases, the local search in V-SOMA has a higher chance to find a improved upshot than other algorithms because V-SOMA is able to steer clear of



	50	100	150	200	250	300
V-SOMA	0.13323	0.16531	0.19611	0.36132	0.39839	0.49888
SOMA	0.13240	0.17288	0.23032	0.40688	0.44458	0.53708
PSO	0.21197	0.26121	0.45397	0.87694	0.93694	1.25370

Fig. 3: Average of load balancing using 20VM



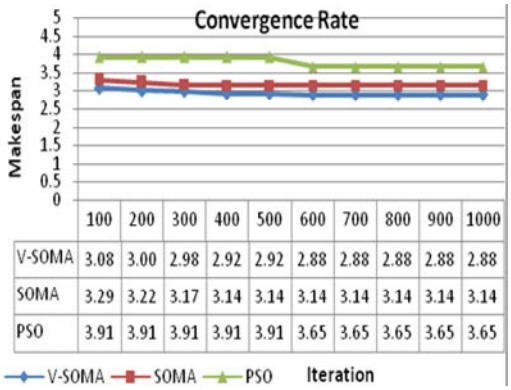
	50	100	150	200	250	300
V-SOMA	0.05940	0.08108	0.09342	0.19284	0.21894	0.25969
SOMA	0.05981	0.08430	0.09385	0.19627	0.22577	0.25602
PSO	0.12971	0.17569	0.30754	0.57124	0.62202	0.78892

Fig. 4: Average of load balancing using 50VM

premature convergence by maintaining the diversity of the population. Also, it can be shown that the Makespan is reducing with increase of VMs for the same number of tasks.

Load balancing index: We examine the average of load balancing index of three algorithms by running simulation for 10 times with 50-300 tasks in 20 or 50 virtual machines as shown in Fig. 3 and 4. demonstrates that load balancing index in PSO is highest because all the tasks were allocated to few optimal resources according to its scheduling strategy. This directs to load imbalance in the system. The load balancing index in the V-SOMA and SOMA are quite closer because they used modulus operator (Zhang *et al*., 2008) to find resource allocation for each task which leads to proper load balancing amid set of resources. Also, it shows that load balancing index is plummeting with the increase of VMs for the same number of tasks.

Convergence rate: We analyze the convergence rate of three algorithms by running simulation with 150 tasks in 20 virtual machines at 1000 iterations as shown in Fig. 5. The above figure illustrates that SOMA converges faster



	100	200	300	400	500	600	700	800	900	1000
V-SOMA	3.08	3.00	2.98	2.92	2.92	2.88	2.88	2.88	2.88	2.88
SOMA	3.29	3.22	3.17	3.14	3.14	3.14	3.14	3.14	3.14	3.14
PSO	3.91	3.91	3.91	3.91	3.91	3.65	3.65	3.65	3.65	3.65

Fig. 5: Convergence analysis

than V-SOMA and PSO because PSO used constraint random inertia weight and V-SOMA used VNS local search algorithm to broaden the global probing ability (Zhang *et al*., 2008). In toting up, VNS local search in V-SOMA used to evade premature convergence as well as retain the miscellany of the population, it shows that the end results of V-SOMA are usually better than the other scheduling algorithms, principally for complex and large-scale problems.

CONCLUSION

The proposed V-SOMA algorithm uses VNS local search to pass up premature convergence by preserving the assortment of the population. The proposed algorithm is deployed to poise the load across the entire system while trying to curtail the makespan of a given task sets. The simulation results be evidence for that the proposed algorithm achieves better performance than SOMA and PSO for cloud computing systems.

REFERENCES

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A. DeRose and R. Buyya, 2011. Cloud Sim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software Pract. Experience, 41: 23-50.

Dikaiakos, M.D., D. Katsaros, P. Mehra and G. Pallis and A. Vakali, 2009. Cloud Computing: Distributed internet computing for it and scientific research. IEEE Comput. Soc., 13: 10-13.

Kruekaew, B. and W. Kimpan, 2014. Virtual machine scheduling management on cloud computing using artificial bee colony. Proceedings of the International MultiConference of Engineers and Computer Scientists 2014 Vol. I, IMECS 2014, March 12-14, 2014, MECS Publisher, Hong Kong, ISBN: 978-988-19252-5-1, pp: 12-14.

Mladenovic, N. and P. Hansen, 1997. Variable neighborhood search. Comput. Operat. Res., 24: 1097-1100.

Tasgetiren, M.F., M. Sevkli, Y.C. Liang and G. Gencyilmaz, 2004. Particle swarm optimization algorithm for single machine total weighted tardiness problem. Proceedings of the Congress on Evolutionary Computation CEC2004, June 19-23, 2004, IEEE, Istanbul, Turkey, ISBN:0-7803-8515-2, pp: 1412-1419.

Zhang, L., Y. Chen, R. Sun, S. Jing and B. Yang, 2008. A task scheduling algorithm based on PSO for grid computing. Int. J. Comput. Intell. Res., 4: 37-43.