Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm

Lizheng Guo[1,2]
[1] College of Information Sciences and Technology, Donghua University, Shanghai 201620, China ;
[2]Department of Computer Science and Engineering, Henan University of Urban Construction, Pingdingshan 467633, China
{Hkftjh@yahoo.com.cn}

Shuguang Zhao[1], Shigen Shen[1], Changyuan Jiang[1]
[1] College of Information Sciences and Technology, Donghua University, Shanghai 201620, China
{sgzhao@dhu.edu.cn, kxsg@21cn.com, njjcy@126.com}

***Abstract*—Cloud computing is an emerging technology and it allows users to pay as you need and has the high performance. Cloud computing is a heterogeneous system as well and it holds large amount of application data. In the process of scheduling some intensive data or computing an intensive application, it is acknowledged that optimizing the transferring and processing time is crucial to an application program. In this paper in order to minimize the cost of the processing we formulate a model for task scheduling and propose a particle swarm optimization (PSO) algorithm which is based on small position value rule. By virtue of comparing PSO algorithm with the PSO algorithm embedded in crossover and mutation and in the local research, the experiment results show the PSO algorithm not only converges faster but also runs faster than the other two algorithms in a large scale. The experiment results prove that the PSO algorithm is more suitable to cloud computing.**

***Index Terms*—computing cloud, data intensive, computing intensive, particle swarm optimization, task scheduling**

I. INTRODUCTION

Scientific applications are usually complex and data-intensive. In many fields, such as astronomy [1], high-energy physics [2] and bioinformatics [3], scientists need to analyze terabytes of data either from the existing data resources or from the collected physical devices. The scientific analysis is usually computation intensive and data intensive, and it is natural that it takes a long time for execution. In other aspects, data intensive is not only in the scientific applications, but also in the web environment. Data intensive computing, in the web environment, promotes the distributed application of web clusters because of their scalability and cost-effectiveness instead of one web server with high performance. In order to minimize the response time and the processing time, the task scheduling policy, in such systems, focuses on the manner of scheduling the tasks that decrease the data movement and increase the processing ability of these systems and thus, improve the performance. Running scientific workflow application usually needs not only high performance computing but also massive storage [4]. Nowadays, popular scientific workflows are deployed in grid systems [5] because they have high performance and large storage. However, grid computing is suitable for specialized application and grid computing is not available for users all over the world to use. Cloud computing is a new paradigm for distributed computing. The new emergence of cloud computing technologies provides method to deal with complex applications which are the applications of great deal of data and needing high performance applications. Clouds have been define to be a type of parallel and distributed system consisting of inter-connected and virtualized computers. These computers can be dynamically provisioned as per user's requirements [6]. Berkeley [7] about the define of cloud as following: "Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what we will call a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, we call it a Public Cloud". Reference [8] made a comprehensive comparison of grid computing and cloud computing. Cloud computing system and grid system have like features, for example, they have high performance and massive storage, and all these can meet the needs of scientific workflow. As cloud computing has the advantages of delivering a flexible, high-performance, pay-as-you-go, on-demand offering service over the internet, common users and scientists can use cloud computing resolve complex application.

In terms of our problems, the complex applications can be divided into two classes. The one is computing intensive, the other is data intensive. As far as the data intensive application, our scheduling strategy should decrease the data movement which means decreases the transferring time; but the computing intensive tasks, our

Corresponding author.
E-mail address: kftjh@yahoo.com.cn

doi:10.4304/jnw.7.3.547-553

scheduling strategy should schedule the data to the high performance computer. In order to take cloud computing, scientific workflow will gain a more utilizations. However, we face a lot of new challenges, of which data and task scheduling are one. How to efficient schedule all the tasks of an application is the most important problem. In this paper, we focus on minimizing the total executing cost and transferring time. In order to reduce the executing time, we schedule the computing intensive tasks to the high performance computer. As the tasks scheduling is a NP-complete problem, some heuristic algorithms have been used to resolve this kind of problems. Thus, we achieve the task scheduling by using a method called Particle Swarm Optimization (PSO).

Our main contributions in this paper are as follows:

(1)We formulate a model for task scheduling in cloud computing to minimize the overall time of executing and transmitting.

(2)We design a PSO algorithm to solve task scheduling based on the proposed model, compare and analyze with other algorithm based on PSO.

The rest of this paper is organized as follows. Section Ⅱ presents related work. Section Ⅲ introduces the task scheduling and formulate the mode. Section Ⅳ gives the details of the algorithm of task scheduling. Section Ⅴ demonstrates the simulation result and the evaluation. Section Ⅵ concludes the paper.

II. RELATED WORK

Task scheduling is very important to scientific workflows and task scheduling is challenge problems too. It has been research before in traditional distributed computing systems. Reference [9] is a scheduler in the Grid that guarantees that task scheduling activities can be queued, scheduled, monitored and managed in a fault tolerant manner. Reference [10] proposed a task scheduling strategy for urgent computing environments to guarantee the data's robustness. Reference [11] proposed an energy-aware strategy for task scheduling in RAID-structured storage systems. Reference [12] studies multicore computational accelerators and the MapReduce programming model for high performance computing at scale in cloud computing. They evaluated system design alternatives and capabilities aware task scheduling for large-scale data processing on accelerator-based distributed systems. They enhanced the MapReduce programming model with runtime support for utilizing multiple types of computational accelerators via runtime workload adaptation and for adaptively mapping MapReduce workloads to accelerators in virtualized execution environments. However, none of them focuses on reducing the processing cost and transmitting time between data centers on the Internet. As cloud computing has become more and more important, new data management systems have designed, such as Google's GFS (Google File System) and Hadoop. Their data hide in the infrastructures and the users can not control them. The GFS is designed mainly for Web search applications. Some researchs are based on cloud computing. The Cumulus project [13] introduced scientific cloud architecture for a data centre. And the Nimbus [14] toolkit can directly turn a cluster into a cloud and it has already been used to build a cloud for scientific applications. Within a small cluster, data movement is not a big problem, because there are fast connections between nodes, i.e. the Ethernet，and the processing time is not longer. However, the scientific cloud workflow system is distributed applications which need to be executed across several data centers on the internet.

In recent studies, Reference [15] from the cost aspect studied the compute-intensive and data-intensive application. They formulate a non-liner programming model to minimize the data retrieval and executing cost of data-intensive workflows in clouds. Reference [16] investigated the effectiveness of rescheduling using cloud resources to increase the reliability of job completion. Specifically, schedules are initially generated using grid resources while cloud resources are used only for rescheduling to deal with delays in job completion. A job in their study refers to a bag-of-tasks application that consists of a large number of independent tasks; this job model is common in many science and engineering applications. They have devised a novel rescheduling technique, called rescheduling using clouds for reliable completion and applied it to three well-known existing heuristics. Reference [17] proposed matrix based k-means clustering strategy to reduce the data movement in cloud computing. However, the reducing of data movement and cost do not mean that the processing cost and transmitting time decrease. In this work, we try to schedule the application data based on PSO algorithm in order to reduce the data transmitting time and process cost.

Reference [18] study the deployment selection challenge from two different and usually conflicting angles, namely from the user's and the system provider's perspective. Users want to optimize the execution of their specific requests without worrying about the consequences for the overall system. The provider's objective however is to optimize the system throughput and allow a fair usage of the resources, or a usage mode as defined by the decision makers. While the users are most likely pursuing the same strategy for each request, the system responsible may face a dynamic environment, including changing requirements, changing usage patterns and changing decisions in terms of business objectives. To address this issue, they propose a multi-objective optimization framework for selecting distributed deployments in a heterogeneous environment based on Genetic Algorithm (GA).

In fact, task assignment has been found to be NP-complete [19]. Since task assignment is NP-Complete problem, Genetic Algorithm (GA) has been used for task assignment [20]. But, genetic algorithm may not be the best method. Reference[21] has illustrated that the particle swarm optimization algorithm is able to get the better schedule than genetic algorithm in grid computing. Reference [22] has shown that the performance of Particle Swarm Optimization (PSO) algorithm is better than GA algorithm in distributed system. Not only the

PSO algorithm solution quality is better than GA in most of the test cases, but also PSO algorithm run faster than GA. So, we use a method called Particle Swarm Optimization to optimize the task scheduling problem. In this paper, we focus on minimizing the total executing time and transferring time.

III. TASK SCHEDULING PROBLEM FORMULATION

We denote the task scheduling as a task interaction graph (TIG). We describe the TIG by G(V,E), where V={1,2, …, n} represents the tasks of a application and E={C_{ij}} indicates the information exchange between these tasks. The edge weigh e_{ij} between node i and j denotes the information exchange between these pair of tasks. The node is defined for processor centers. The node weigh w corresponds to the work capacity of the node. Fig. 1 shows an example of the task scheduling in a heterogeneous environment.

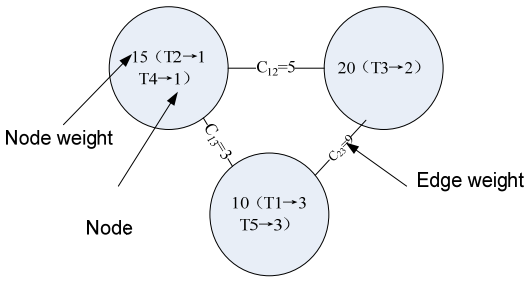

Figure 1. A TIG example on heterogeneous system

In this paper, we consider the task scheduling with the following scenarios. The processors in the cloud computing are heterogeneous and they have different processing ability which depend on their amount units of memory and performance of cup's capacity. A task's processing cost will be variety according to the task being assignment to different processors. On the other hand, the communication cost between two nodes will be changing because between two different node's bandwidth have diversity and changing over time. Our target is how to minimize the communication time and execution cost. In order to formulate the task scheduling, we define T_i i= {1, 2, 3, …, n} as n independent tasks permutation and P_j j={1, 2, 3, …, m} as m computing resources and B_{ij} ,i, j= {1, 2, 3, …, k}as the bandwidth between two nodes and k is the number of node； x_{ik}=1 if task i is assigned to processor k, and x_{ik}=0, otherwise; y_{ijkl}=1 if $k \neq l$ and task i is assigned to processor k and task j is assigned to processor l, and y_{ijkl}=0 otherwise; n is the number of tasks; m is the number of processors; DE_{ik} is the amount of data that the i task assigning to the processor k and P_m and P_c are the processor's memory and CPU's capacity; DT_{ij} is the interchange data amount between task i and task j; Equation (1) and (2) respectively represent the executing cost and the transforming time. Supposing that the processing time is know for task i executing on processor j and the communication time is know for transmitting the data from i node to j node. Our purpose is how to map all the tasks to all the processors make the total time and cost minimizing, which making the (3) value is minimizing.

$$C_{exe}(M) = \sum_{i=1}^{n}\sum_{k=1}^{m} x_{ik} * \frac{DE_{ik}}{Pm * Pc}. \quad (1)$$

$$C_t(M) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} y_{ijkl} * \frac{DT_{ij}}{B_{ij}} . \quad (2)$$

$$Total(M) = C_{exe}(M) + C_t(M) . \quad (3)$$

$$\text{Subject to} \quad \sum_{k=1}^{m} x_{ik} = 1, \forall i = 1,2,...n . \quad (4)$$

$$\sum_{k=1}^{m}\sum_{l=1}^{m} y_{ijkl} = 1, \forall i, j = 1,2,...,n, k \neq l . \quad (5)$$

$$x_{ik}, y_{ijkl} \in \{0,1\}, \forall i, j, k, l . \quad (6)$$

IV. TASK SCHEDULING BASED ON PARTICLE SWARM OPTIMIZATION

PSO is an algorithm proposed by Kennedy and Eberhart in 1995 [23]. Social behavior of organisms such as bird flocking and fish schooling motivated them to look into the effect of collaboration of species onto achieving their goals as a group. A large number of birds or fishes flock synchronously, change direction suddenly, and scatter and regroup together according to the individual and social experience. Each individual is called a particle. Each particle gains good experience from its past and the social past experience. Each particle not only knows its own best position, which is the pbest, but also knows the social best position, which is gbest. In the movement of all the particles, each particle adjusts its direction and velocity in the light of the pbest, gbest and its own current position (x_i^k) and velocity (v_i^k). The pbest and the gbest are dynamic adjustment each iteration. The improvements equations of PSO are listed as (7) and (8). The parameters and their mean of parameters are shown in table Ⅰ.

The PSO algorithm is similar to other evolutionary algorithm. In PSO, each particle is a candidate solution of the underlying problem and has n dimensions which are decided by special problem. Particles position and velocity are initialized randomly. Each particle has a fitness value, which will be evaluated by a fitness function to be optimization in each generation. Each particle knows the pbest and gbest. In each generation the velocity and the position of particle will be update in light of (7) and (8) respectively.

TABLE Ⅰ. PARAMETERS AND THE MEAN OF THE PARAMETERS

Parameters	Mean of the parameters
v_i^k	velocity of particle *i* at iteration *k*
v_i^{k+1}	velocity of particle *i* at iteration *k+1*
x_i^k	position of particle *i* at iteration *k*
x_i^{k+1}	position of particle *i* at iteration *k+1*
ω	inertia weight
c_1, c_2	acceleration coefficients
$rand_1, rand_2$	random number between 0 and 1
$pbest_i$	best position of particle *i*
$gbest$	best position of entire particles in a population

$$v_i^{k+1} = \omega v_i^k + c_1 rand_1 * (pbest_i - x_i^k) + c_2 rand_2 * (gbest - x_i^k) \quad (7)$$

$$x_i^{k+1} = x_i^k + v_i^k \quad (8)$$

A. Particle representation

Our target is resolving the task scheduling, so we should map each underlying solution to a particle. We define every particle as n dimensions vector response to the n tasks. An element delegates a task and the element is an integer value between 1 and n. The particle represents one of the task scheduling. Fig.2 describes an illustrative example for the task assignment to PSO particle mapping.

Task1	*Task2*	*Task3*	*Task4*	*Task5*
Processor3	*Processor1*	*Processor2*	*Processor1*	*Processor3*

Figure 2. Task assignment to PSO particle mapping

B. Initial Swarm Generation

The initial particle population is constructed randomly for PSO algorithm. The position and velocity of each particle initial value produce according to the following formula [24]:

$$x_i^1 = x_{\min} + (x_{\max} - x_{\min}) \times rand \quad (9)$$

$$v_i^1 = v_{\min} + (v_{\max} - v_{\min}) \times rand \quad (10)$$

Where $x_{\max} = v_{\max} = 4.0$ $x_{\min} = v_{\min} = -0.4$ and rand is a random value between 0 and 1.

As the velocity is a continuous value and our task scheduling is a discrete permutation in PSO algorithm, we should transform the continuous value to discrete permutation. The small position value (SPV) rule [10] which borrowed from the random key representation to solve the task assignment can convert the continuous value to discrete permutation.

We use the SPV rule transform a continuous position vector $x_k^i = [x_1^i, x_2^i, ..., x_n^i]$ to a dispersed value permutation vector $s_k^i = [s_1^i, s_2^i, ..., s_n^i]$. In order to counting the processing time, we should map each element of the vector s_k^i into processor's vector $p_k^i = [p_1^i, p_2^i, ..., p_n^i]$. The converting operation is defined as following equation:

$$p_i^k = s_i^k \bmod m + 1 \quad . \quad (11)$$

TABLE Ⅱ. ILLUSTRATE THE RESULT OF PARTICLE x_i^k OF PSO ALGORITHM TO p_i^k FOR 7 TASKS AND 5 PROCESSORS.

Dimension	x_i^k	s_i^k	p_i^k
1	0.1587	2	3
2	3.6189	6	2
3	2.3824	5	1
4	0.0292	1	2
5	0.8254	3	4
6	2.0063	4	5
7	3.8130	7	3

C. The PSO Algorithm

The details algorithm is described in PSO algorithm. The algorithm begins with *k* random particle vector and each particle is n dimensions. Every particle vector is a candidate solution of the underlying problem. The particles are the task to be assigned and the dimensions of the particle are the number of the special tasks in a workflow. Then, each particle moves by the direction on the pbest and gbest until the maximal number of iterations. When the algorithm executes over, the gbest and fitness value are the corresponding task scheduling and the minimal cost of the optimal strategy.

PSO algorithm

1. Initialize particle position vector and velocity vector randomly according (9) and (10). The vector's dimension equal to the size of the special tasks.
2. Convert the continuous position vector ($x_k^i = [x_1^i, x_2^i, ..., x_n^i]$) to discrete vector ($s_k^i = [s_1^i, s_2^i, ..., s_n^i]$) in light of SPV rule. Then, transform the $s_k^i = [s_1^i, s_2^i, ..., s_n^i]$ to processor's vector ($p_k^i = [p_1^i, p_2^i, ..., p_n^i]$) according to (11). Last, calculating each particle's fitness value as in (3).
3. If one particle's fitness value is better than current, setting current value replace previous pbest and as the new pbest.
4. Selecting the best particle from all the particle as the gbest .
5. For all particles update their position and velocity by (7) and (8).
6. If reaching to the maximum iteration or getting the ideal result stops, otherwise repeating from Step 2.

V. EXPERIMENTAL RESULT

A. Performance Metric

As a measure of performance, on one hand, we used cost for the processing and transforming to a placation as a metric. On the other hand, we measure the executing time and the convergence as a metric. We use PSO embed in SPV (SPO) algorithm, PSO algorithm embed in crossover and mutation (CM-PSO), PSO algorithm embed in local search (L-PSO) to compare the above metric.

B. Experimental Settings

In order to presenting the amount of communications between tasks, we define the communication density ρ of G (V,E) as

$$\rho = \frac{|E|}{n(n-1)/2} \quad . \tag{12}$$

Where |E| is the numbers of existing communication between all tasks, n represents the number of the total tasks and n(n-1)/2 indicates the maximal number of communication demand in the all tasks. The other factors are the number of tasks (n) and the number of processors (m).

The testing data set is produce randomly. We normalize the processor's CPU and memory, restricting between 1 and 250, the task data is among 1 and 10000, the communication data is between 50 and 10000, and the bandwidth varies form 10 to 1000. In the following part, all of the experiments are tested on an Intel(R) Pentium(R) Dual CPU E2160 1.80 GHz with 1G RAM. The parameter is as following: size of swarm is 30, self-recognition coefficient c1 is 1.49445, social coefficient c2 is 1.49445, and inertial weight w is 0.729 [25], crossover is 0.8 and mutation is 0.01.

C. Comparative Performance

As PSO algorithm and hybrid PSO algorithm are stochastic and a result may be different for a particular problem, each problem runs 10 times and gets the average value. Moreover, the fitness function is the main source providing the leading the target to optimal solution, the performance test based on the minimum cost obtained when the algorithm has run exact numbers, here, we set the iterations is 10000. Table Ⅲ shows the average result using the three algorithms. From the table Ⅲ, we can get the conclusion: three algorithm performance is almost the same when the task is little, here is 7 tasks and 5 processors, the CPU's running time have a little difference, the PSO is small, the L-PSO is middle and the CM-PSO is the big; when the task is 25 and the processor is 12, the cost of PSO is little better than the other two, but the running time of the L-PSO is almost three times the PSO and the CM-PSO is nearly two time PSO; when the task is 70, processor is 25, in terms of cost and running time, PSO performance is the best, especially in the running time.

D. Convergence Analysis

From Fig. 3 we can see that the each iteration time of PSO is the least in the PSO, CR-PSO and L-PSO when the task=25, processor=12 and ρ =0.75. This represents that PSO algorithm runs faster than other algorithm. Fig. 4 describes the completion time of the three algorithm when task=70, processor=25 and ρ =0.75. It also display that PSO usually had a better average completion time value than the other two. From Fig. 3 and Fig.4 we can get the conclusion that the PSO algorithm runs faster than CR-PSO and L-PSO. The algorithm running time is a very important technology parameter which represents the time of having found the optimal resolution of an application. To some extent it determines good or bad of an algorithm and it is especially important to a time intensive application. Fig. 5 gives the cost of the three algorithms on 25 tasks, ρ =0.75 and 12 processors; it represents the L-PSO convergence quicker than the other two, but the cost of the L-PSO is better than the other algorithm. Fig. 6 shows the cost of PSO, CR-PSO and the L-PSO about 70 tasks, ρ =0.75 and 25 processors; it represents the PSO both converging faster and optimizing better than CR-PSO and L-PSO. As PSO algorithm not only runs faster, but also save the processing time in the large and complex conditions. In cloud computing there are many tasks and processors, so PSO algorithm suits more to cloud computing. Moreover, the spending time of the PSO algorithm is shorter than that of other two algorithms.

VI. CONCLUSIONS

In many different domains, in order to improve the efficiency the optimizing task scheduling is necessary. In cloud computing resources distribute all over the world, and the data usually is bigger and the bandwidth often is narrower, these problems are more important. In this paper, we presented the task scheduling optimizing method in cloud computing, and we formulate a model for task scheduling to minimize the cost of the problem and solved it by a PSO algorithm. By comparing and analyzing particle swarm algorithm with crossover, mutation and local search algorithm based on particle swarm, we propose the particle swarm algorithm embed in SPV, which represents better performance. Experimental result manifests that the PSO algorithm both gains optimal solution and converges faster in large tasks than the other two. Moreover, running time is shorter than the other two too. It is obvious that PSO is more suitable to cloud computing.

In future work, our research is to center on the energy efficiency and service availability in cloud computing system. We aim to improve task scheduling optimization and make optimization policy to optimize not only the efficiency, but also the energy and service level agreement.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that have improved the presentation, quality and correctness of this paper.

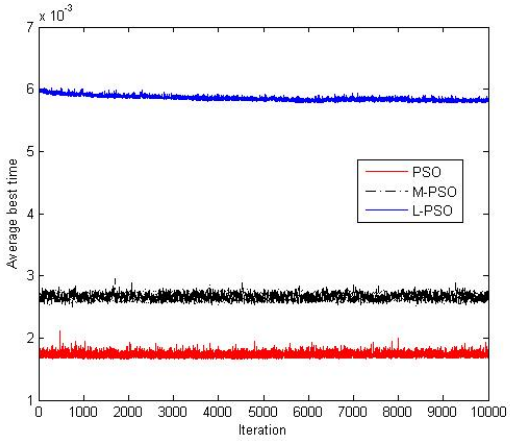

Figure 3. 25 tasks and 12 processors

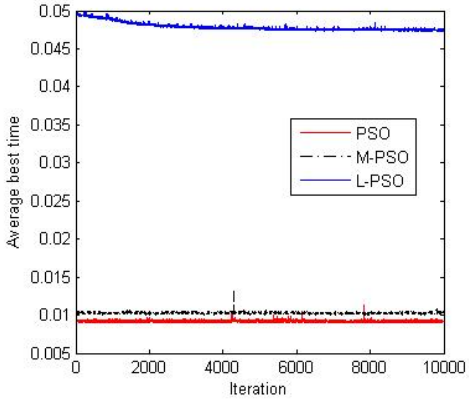

Figure 4. 70 tasks and 25 processors

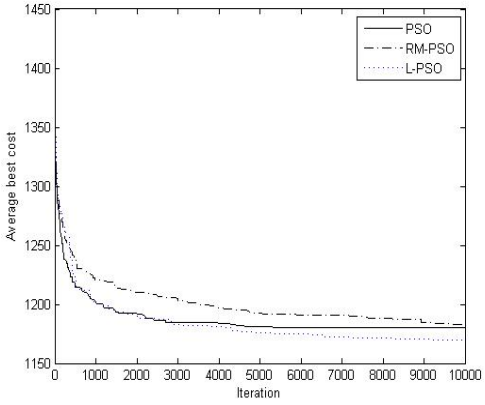

Figure 5. 25 tasks and 12 processors

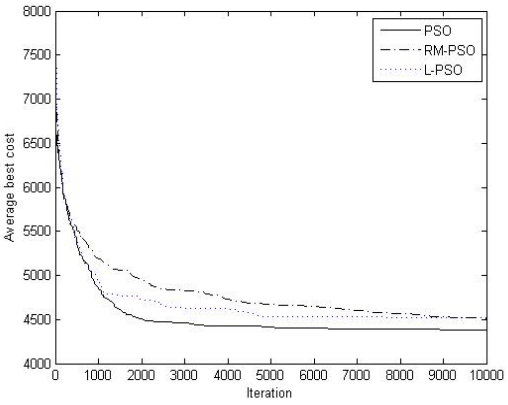

Figure 6. 70 tasks and 25 processors

Table III. The average costs and the used CPU times by PSO, CM-PSO and L-PSO over different problems.

n	m	ρ	PSO		CM-PSO		L-PSO	
			Cost	CPU time	Cost	CPU time	Cost	CPU time
7	5	0.25	204.09	5.63	204.09	14.51	204.09	8.62
		0.5	205.13	5.86	205.13	14.77	205.13	8.90
		0.75	260.70	6.14	260.70	15.01	260.70	9.17
25	12	0.25	1131.9	11.08	1129.8	20.28	1134.2	47.18
		0.5	1206.1	14.27	1210.3	23.57	1211.8	52.78
		0.75	1179.9	17.34	1182.8	26.64	1169.6	58.94
70	25	0.25	3369.8	38.69	3437.5	51.53	3385.2	329.96
		0.5	2477.9	64.81	2497.8	76.39	2493.4	407.10
		0.75	4378.0	92.47	4513.8	102.81	4526.2	478.89

REFERENCES

[1] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi,M. Livny, Pegasus: mapping scientific workflows onto the grid, in: European Across Grids Conference, Nicosia, Cyprus, 2004, pp. 11 – 20.

[2] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee,Scientific workflow management and the Kepler system, Concurrency and Computation: Practice and Experience (2005) 1039 – 1065.

[3] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K.Glover, M.R. Pocock, A. Wipat, P. Li, Taverna: a tool for the composition and enactment of bioinformatics workflows, Bioinformatics 20 (2004) 3045 – 3054.

[4] E. Deelman, A. Chervenak, Data management challenges of data-intensive scientific workflows, in: IEEE International Symposium on Cluster Computing and the Grid, (2008) 687 – 692.

[5] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, Scientific workflow management and the Kepler system, Concurrency and Computation: Practice and Experience (2005) 1039 – 1065.

[6] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, andI. Brandic. Cloud computing and emerging it platforms:

Vision,hype, and reality for delivering computing as the 5thutility. Future Generation Computer Systems,(2009) 25(6):599 – 616 .
[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. H. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html
[8] I. Foster, Z. Yong, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: Grid Computing Environments Workshop, GCE'08, (2008) 1 – 10.
[9] T. Kosar, M. Livny, Stork: Making data placement a first class citizen in the grid, in: Proceedings of 24th International Conference on Distributed Computing Systems, ICDCS 2004,(2004) 342 – 349.
[10] J.M. Cope, N. Trebon, H.M. Tufo, P. Beckman, Robust data placement in urgent computing environments, in: IEEE International Symposium on Parallel &Distributed Processing, IPDPS 2009, (2009)1 – 13.
[11] T. Xie, SEA: A striping-based energy-aware strategy for data placement in RAID-structured storage systems, IEEE Transactions on Computers 57 (2008) 748 – 761.
[12] M. Mustafa Rafique a,* , Ali R. Butt a, Dimitrios S. Nikolopoulos b,c, A capabilities-aware framework for using computational accelerators in data-intensive computing, J. Parallel Distrib. Comput. 71 (2011) 185–197
[13] L. Wang, J. Tao, M. Kunze, A.C. Castellanos, D. Kramer, W. Karl, Scientific cloud computing: Early definition and experience, in: 10th IEEE International Conference on High Performance Computing and Communications, HPCC'08,(2008) 825 – 830.
[14] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, M. Tsugawa, Science clouds:Early experiences in cloud computing for scientific applications, in: First Workshop on Cloud Computing and its Applications, CCA'08,(2008) 1 – 6.
[15] S. Pandey, A. Barker, K. K. Gupta, R. Buyya ,Minimizing Execution costs when using globally distributed cloud services,2010 24th IEEE International Conference on Advanced Information Networking and Applications
[16] Y. C. Lee, A. Y. Zomaya, Rescheduling for reliable job completion with the support of clouds, Future Generation Computer Systems 26 (2010) 1192_1199
[17] D Yuan, Y Yang, X Liu, A data placement strategy in scientific cloud workflows, Future Generation Computer Systems(2010)1200-1214
[18] E. Vineka, P. P. Beranb, E. Schikutab, A dynamic multi-objective optimization framework for selecting distributed deployments in a heterogeneous environment , Procedia Computer Science 4 (2011) 166–175
[19] V.M. Lo, "Task assignment in distributed systems", PhD dissertation, Dep. Comput. Sci., Univ. Illinois, Oct. 1983.
[20] G. Gharooni-fard, F. Moein-darbari, H. Deldari and A. Morvaridi, Procedia Computer Science, Volume 1, Issue 1, May 2010,Pages1445-1454 ,ICCS 2010.
[21] L. Zhang, Y.H. Chen, R.Y Sun, S. Jing, B. Yang. " A task scehduling algorithm based on PSO fro Grid Computing", International Jouranal of Computational Intelligence Research.(2008),pp.37-43.
[22] A. Salman. "Particle swarm optimization for task assignment Problem". Microprocessors and Microsystems, November 2002. 26(8):363–371.
[23] Kennedy J, Eberhart RC (1995), Particle swarm optimization. In:Proceedings IEEE Int'l. Conf. on Neural Networks, vol. IV,1942– 1948.
[24] M. Fatih Tasgetiren, Yun-Chia Liang, Mehmet Sevkli, and Gunes Gencyilmaz, "Particle swarm optimization and differential evolution for single machine total weighted tardiness problem," International Journal of Production Research, (2006) 4737-4754
[25] Y Shi, R C Eberhart. Empirical study of particle swarm optimization. Proc. IEEE Congr. Evol. Comput. (1999)1945-1950.

Lizheng Guo was born in Kaifeng, China. He has completed his Maser's degrees in the College of Computer Science and Technology of Chongqing University of Posts and Telecommunications, Chongqing, China in 2005. He is currently pursuing his Ph.D. in Pattern Recognition and Intelligent System at the College of Information Science and Technology, Donghua University, Shanghai, China. His research interests include distributed system, intelligent computing, cloud computing and grid computing.

Shuguang Zhao received the B.S. degree in information processing from Xidian University, Xi'an, China, 1986, the M.S. degree in signal and information processing and Ph.D. degree in circuit and system from Xidian University, Xi' an, China, in 1992 and 2003, respectively. He is currently Professor at Donghua University. His research interests include evolvable hardware and intelligent signal processing.

Shigen Shen received his B.S. degree in Mathematics Education from Zhejiang Normal University, Jinhua, China in 1995, his M.S. degree in Computer Science and Technology from Zhejiang University, Hangzhou, China in 2005. He is currently pursuing his Ph.D. in Pattern Recognition and Intelligent System at the College of Information Science and Technology, Donghua University, Shanghai, China. His current research interests include *Wireless Sensor Networks* and *Game Theory*.

Changyuan Jiang was born in Anqing, China. He has completed his Master's degrees in the College of Computer Science and Technology of Nanjing Normal University, Nanjing, Jiangsu, China, in 2005. He is currently a Ph.D candidate in the College of Information Science and Technology of the Donghua University, Shanghai, China. His research interests include distributed system, intelligent computing.