J. Basic. Appl. Sci. Res., 2(11)11659-11667, 2012

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

A Study about Servers Count and Branch Factoring Tasks of Workflow in Cloud Computing

Rohollah Esmaeli Manesh[1], Diana Sarokhani[2], Mohammad Taher Sarokhani[3], Nasrin Naderian[4], Ahmad Zareie[5], Mohammad Khalaf Sayadiyan[6]

[1] Department of Computer _Gilan_E_ Gharb Branch, Islamic Azad University, Gilan_E_ Gharb , kermanshah, Iran.
[2]Department of Computer Engineering, Malayer University, Malayer, Iran.
[3]Department of Computer Engineering, science And Research Branch, Islamic Azad University, kermanshah, Iran.
[4]Engineering Department, Faculty of Engineering, Tarbiat Moallem University, Karaj/Tehran, Iran.
[5]Young Researchers, Songhor Branch, Islamic Azad University Songhor, kermanshah, Iran.
[6]Young Researchers, Eslam Abad_E_ Gharb Branch, Islamic Azad University, Eslam Abad_E_ Gharb, kermanshah, Iran.

ABSTRACT

The architecture of a cloud includes several key modules user interaction interface, system resource management module with a services catalog, and
resource provisioning module. The system resource management module manages a massive network of servers running in parallel. Often it also uses virtualization techniques to dynamically allocate computing resources. In a distributed system Timing and mapping the priority of tasks among processors is one of the issues attracted most of attention to itself. This issue consists of mapping a Direct Acyclic Graph with a set of tasks on a number of parallel processors and its purpose is allocating tasks to the available processors in order to satisfy the needs of priority and decency of tasks, and also to minimize the duration time of execution in total graph.
In this article, we'll represent an algorithm that useful Servers Count and Branch Factoring Tasks of Workflow in distributed systems.
Simulations is showed that we approach is better than about the list schedule algorithm.
KEYWORDS: Cloud, mapping, distributed system, cloud computing

1. INTRODUCTION

The timing issue of a graph, the functions of a parallel plan on the distributed computing system, is a Np-Complete issue and as one of the challenges in parallel computing problems has aroused a lot of attention [1]. Scheduling and timing of task's graph is an available process for mapping effective tasks on the number of processors that could be done considering purposes such as minimizing the application of execution time, increasing confidentiality, increasing resources profitability, decreasing data transmission costs and using saving resources and etc. [2].Therefore, the efficiency of distributed system greatly depends on the way of timing and task allocation method on them. In one hand, delays of communicative links among processors and their weak bandwidth among them may challenge task's graph application on parallel processors. One of the computing systems that are represented recently is Cloud computing system. The concept of this system has represented since 2007[7] and was successful in many file [3, 4, 5, 6]. The most important purpose of this system is decreasing economical costs and offering computing service such as water, electricity, gas and telephone. I.e. users use resources and services when they need according to the amount of their needs and also pay based on the rate of their use [8]. This system consists of many data centers that are distributed geographically all over the world and are accessible using internet. Each data center consists of many computing and saving servers and other resources. Servers in each center are attached with a high bandwidth in every spot and we can consider zero as their communicative delay among them.

According to the stated features cloud computing system is suitable condition for applying priority task's graph and while mapping graph on inter-center computing servers, it is possible to benefit a wide range of advantages such as, removing communicative delays using powerful saving computation servers, decreasing additional costs, profitability of high scaling cloud computing system and etc. But while

***Corresponding Author:** Rohollah Esmaeli Manesh, Department of Computer _Gilan_E_ Gharb Branch, Islamic Azad University, Gilan_E_ Gharb ,kermanshah, Iran.

mapping the parallel application on cloud computing system, some computing servers may remain unused effectively, and their profitability decreases to some extent. Executing task's graph on some processors always is not monotonous, since according to the parallel nature of tasks it might need four computing server in a moment and after completing tasks the number of required servers decrease to three server, these changes will be continued till the complementation of all graph . As in cloud computing system there is another simultaneous requests in which processors might serve them, therefore consecutive switches of computing servers are needed between simultaneous requests of other users and a user that represented the task graphs to the cloud computing system. In addition to decreasing profitability of computing resources of these switches, they may not be possible sometimes for severe oscillations, because based on the rate of overload that will have, it is better instead of switching to another use, let the computing server idle for a moment and restart executing other tasks of graph. To understand this issue clearly pays attention to the following example:
If task graph be like the fig.1

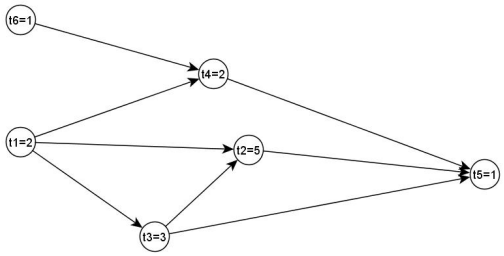

Fig 1- a sample of tasks graph

Two different mapping states are shown in figures 2 and 3 on two computing server for this graph:

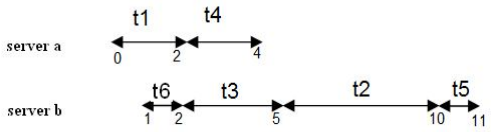

Fig 2- First mapping for graph in the figure 1

And second mapping for this graph is:

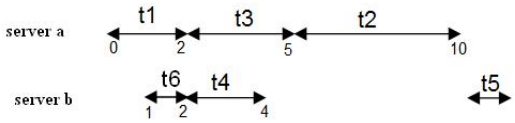

Fig 3- Second mapping for graph figure1

As it is being observed in the first mapping server 'a' will be released in the moment of '4' that in the interval '0 to 4' completely uses computing server and also server 'b' that from moment 1 to 11 serve the graph is completely profitable , but in the second mapping server 'b' from moment 1 to 11 serve the graph and from the moment 4 to 10 it is useless and idle, so in some extent its profitability decreases because if in this interval the processor execute the other users 'simultaneous requests and again return to the application based on overload it is inefficient and it is of high costs. In this article we want to represent an approach for mapping task's graph on computing servers in a data center of cloud computing system in order to make profit from available servers and to decrease its idle and useless time.

Related works

Timing and mapping are one of the main challenges in executing the parallel applications on distributed systems and aroused a lot of attention. To map and schedule task graph various approaches are

presented that in each one some limitations are considered for system and timing has been done schedule has been done for different purposes. This approach can be divided into the following classes [28, 34]:

- Task Duplication Based (TDB) [9][10][11]
- Bound Number of Processors (BNP)Scheduling[12][13][14][15]
- Unbounded Number of Clusters (UNC) Scheduling[15][16]
- Arbitrary Processor Network (APN) scheduling[17][18][19][20]
- List scheduling algorithms[35][36][37][38][39]

In the Task Duplication Based (TDB) scheduling with the purpose of decreasing Schedule length some tasks duplicated and run on several servers, but using this technique in cloud computing system is inefficient and leads to increase in computation costs. Some algorithms suppose that there exist an unlimited number of processors to apply tasks. But strategies of second class schedule consider a high limit for processors that are available. According to the limitation of computing servers in the centers of cloud computing systems a high limitation was considered for processors. Schedule algorithms with unlimited number of clusters, in the beginning of schedule process each node is considered as a cluster and in the next step when two clusters combination decrease the completion time, two cluster combine and this combination continue recessively. The reason for this kind of algorithms is using plenty of processors to decrease the application time. Therefore, this approach needs another stage for clusters mapping on processors, because available processors might be less than clusters.

Another important feature in schedule is considering the type of server attachment and network topology. As servers in a cloud computing data center are attached they completely connected with a high bandwidth, in our work, topology is considered completely attached. Of the other represented schedule we can refer to following cases.

Kumar represented an approach for scheduling independent works in a distributed system that the only relationship between tasks is the need of reaching shared files and no task is needed for the results of the other application task [21]. In [22] a schedule on grid system that is represented data replacement activities are closely similar to computing tasks and they can be queuing, scheduling and managing. In this article [23] also a data placement approach represented in the cloud environment and the only task placement criteria for place of processing tasks has set a center with tasks data and the short length of schedule is not considered.

In distributed system and in the field of task designation the other kinds of timing with different purposes are represented. In [25], genetic algorithm is used for mapping a task graph in the distributed systems to decrease the length of schedule and load balancing on the servers. In [9], an approach to scheduling tasks has been stated for the purpose of fulfillment of task completion deadlines and optimization of execution costs. in approach [21] schedule tasks in order to decrease schedule length and increasing reliability of executing application in distributed environment heterogeneous . This Article [28] represented a method for scheduling the workflow on processor with various abilities. Of the other approaches that are represented for allocation of independent tasks in homogenous distributed environment to increase reliabilities we can refer to [29, 30]. In the previous work [31] we represented an approach for scheduling a work flow graph. In this approach we tried to decrease storage costs, as possible as, during the executing the graph. In this work we want to represent an approach in order to increase profitability and usage of computing servers.

Stating the main problem

In this article we map the workflow of graph on some computing servers in a data center, which are servers completely connected. The number of servers are shown using M such as m1,m2,……,mM.

The workflow graph of DAG consists of N nodes that we have n1,n2,……,nN that each node shows just one task of work graph. Each task consists of a set of instructions that should be executed on a computing server without retaking and respectively. Each node has one or more parent task and when it starts running that running of all parent tasks being ended. In this article we call a node without parent 'fn' and the node without child 'ln'. Also graph has the E directed edge where shows the priority and the consequence of running tasks. It means $n_i \rightarrow n_j$ edge show that nj is a child for ni and not until the end of ni, nj can not start. In some researches usually there is a weight for edges that show communicative cost or duration till output data of n_i are given to server n_j (if t_i and t_j are mapped in different centers). But as we map graph on computing servers in a center and communicative links between servers in a data center have

a high bandwidth. Communicative costs from the time giving ni data to nj is zero, therefore we consider no weight for edges. If mapping ni on server mj, the beginning time will be shown by ST(ni,mj), and its completion time is shown by ET(ni,mj). In which according to being homogeneous computing servers:

(1) $$ET(n_i, m_j) = ST(n_i, m_j) + w(n_i)$$

That $w(n_i)$ shows the task of executing time. Our purpose in this work is increasing profitability and decreasing the idle time of computing servers, it means that when two tasks of ni , nt are mapped as two consecutive tasks on server mj the interval between completion ni and beginning nt means that:

(2) $$ST(n_t, m_j) - ET(n_i, m_j)$$

It should be somehow that be able to use server mj to process simultaneous requests of other users, if this amount is small enough for processing other requests to use , server should be idle in this time interval and this lead to decline in server profitability.

So if

(3) $$0 < ST(n_t, m_j) - ET(n_i, m_j) < \lambda$$

Computing server could not be used in the time interval between these two tasks, therefore it will be useless for sometimes. The value of λ should be considered according to the application time for works as well, that in recent c request are sent to consider data center. So, if in equation 2 is not appointed we can retake computing server from the workflow graph and use in the time interval between beginning and completion of tasks and set them for applying other requests, but in spite of this according to the overload that lead to computing server switch, it is better that intended server stay for work graph user and in this time stay useless and free.

Therefore when work graph usage has the computing server will be calculated by equation 3.

(4)

$$\text{Allocate}(m_j) = w(n_{j1}) + \sum_{k=2}^{l} w(n_{jk}) + \alpha$$

IF $ST(n_{jk+1}, m_j) - ET(n_{jk}, m_j) < \lambda$ then $\alpha = ST(n_{jk+1}, m_j) - ET(n_{jk}, m_j)$ else $\alpha = 0$

That $w(n_{jk})$ intention the time of kth task that mapped on mj computing server and l is the number of these tasks.

And the time that m_j computing server is to serve workflow graph usage will be calculated using the relation 5.

(5) Process (mj) = $\sum_{k=1}^{l} w(n_{jk})$

Therefore profitability rate of computing server mj will be obtained from relation 6.

(6) $$useful(m_j) = \frac{Process(mj)}{Allocate(mj)}$$

And the rate of profitability from all of the calculating servers that serve workflow graph usage application will be obtained from relation 7.

(7) $$Useful\_Allservers = \sum_{k=1}^{M} Useful(m_k)$$

Therefore, the purpose of represented approach is defined as follow:

(8) Max(Useful All servers)

In other hand, the application of whole graph should be ended in an acceptable time; it means that, if possible, schedule length or N completion time might be minimized.

(9) Min($ET(n_N,m)$)

Presented approach

Before stating represented approach it is necessary to define two terms. As for each task till not completion of all parents the possibility of starting application is not supplied, therefore the earliest starting time for task ti is shown by es(ti) and is defined as follow :

(10) $$\begin{cases} \max_{tj \in pred(ti)} (et(ti)) & else \\ 0 & \text{if} \quad i = 1 \end{cases}$$

Then we express a suggested approach.

For each pair of work (ti , tj) we calculate consistency rate based on the following relation.

$$\text{Compatibility}_{ij} \begin{cases} \text{IF } St(tj)\text{-}Et(ti) \geq \lambda \quad \text{Compatibility}_{ij} = 1 \\ \text{ELSE} \quad \text{Compatibility}_{ij} = 1 - \frac{|St(t_j) - Et(t_i)|}{St(t_N) - Et(t_1)} \end{cases}$$

(11)

Also earliest start times equal:

(12)

$$es(t_i) = \begin{cases} \text{IF} \quad i=1 \quad es(t_i) = 0 \\ \text{ELSE} \\ es(t_i) = \max_{tj} \in \text{pred}(ti)(et(tj)) \end{cases}$$

And earliest end time:

(13) $ee(t_i) = es(t_i) + w(t_i)$

Then we create an n*n matrix named AM that we set the element AMi,j is consistency(ti,tj)This matrix will be a polar matrix.

Then using BEA (Bond Energy Algorithm)we make some changes on matrix. In 1972 this algorithm [32] in the system of distributed information bank is used for vertical sectioning the great tables [33].

This algorithm is a transformational algorithm that with transformations on columns and rows, categorize elements that are more consistent.

In this work we give AM matrix to BEA algorithm as entrance. After doing following relation will be selected as breaking point and then we converse the matrix to two sub-matrixes.
By this break, most consistent elements will be categorized in one group.

(14)

$$\text{break point} = \max\left[\sum_{i=1}^{p}\sum_{j=1}^{n} AM_{ij} \times \sum_{i=P+1}^{n}\sum_{j=P+1}^{n} AM_{ij} - \left(\sum_{i=1}^{p}\sum_{j=P+1}^{n} AM_{ij}\right)^2\right]$$

That in the above relation for p=1,2,…,n-1 find the maximum amount and we divide the matrix into two groups. Now for each matrix we calculate break point recessively and the sub-matrix that has the highest break point value has more inconsistent elements and is selected to break it into two sub-matrixes and will be sent to BEA algorithm as entrance data. The stated process is done recessively till the number of sub-matrixes (groups) reach to the number of servers (M) or break point for all sub-matrixes equals to zero (it means that all of the elements in a matrix from all sub-matrixes are completely consistent) then works of each group are arranged based on the most early possible time to start (ES) in an ascending form and each of these groups could be selected for executing on one of the computing servers.

	t1	t2	t3	t4	t5
t1	.2	.1	.2	0	0
t2	.1	.3	.1	.2	.1
t3	.2	.1	.2	0	0
t4	0	.2	0	.2	.1
t5	0	.1	0	.1	.2

BEA

	t1	t3	t2	t4	t5
t1	.2	.2	.1	0	0
t3	.2	.2	.1	0	0
t2	.1	.1	.3	.2	.1
t4	0	0	.2	.2	.1
t5	0	0	.1	.1	.2

One example sorted matrix with BEA algorithm.
In this offers cods for purpose algorithm.

	Input : a directed Acyclic graph of scientific workflow with n task and the availlble server count in datacenter
1	Calculate es and ee for each task of graph
2	For each two task t_i and t_j calculate compatibility based on eqation 11 and insert in AM_{ij}
3	Execute BEA to transform on AM matrix
4	Calculate break point and find p base on equation 12
5	While(break point>0 and group count<server count)
5-1	Break AM matrix to submatrix AM_{Low} and AM_{High}
5-1-1	$AM_{Low} = AM[1 \dots p, 1..p]$
5-1-2	$AM_{high} = AM[p+1 \dots n, p+1..n]$
5-2	Calculate break point base on equation 12 for $submatrix$
5-3	Goto step 4 and running algorithm for submatrix with max breakpoint
6	Insert tasks in each submatrix in a group
7	Sort tasks in each group based on es
8	Map each group on an availlable server

Experiments and Assessment of the represented approach

To evaluate the efficiency of represented approach we manipulate their codes in C# environment and apply them on different graphs with different parameters and in each state the completion workflow time and profitability rate from available servers are calculated and then we show the results using Matlab software in the form of graphs. For analyzing the amount of success in the represented approach we compare the result with schedule approaches based on scheduling list. In this represented approach the main idea is allocating the priority to each task and creating an ascending list of tasks based on priorities, then not until losing all elements in the list in the beginning of the list a deletion will be mapped on a free server [28].to contrast the completion time of workflow we used a normal schedule length that is defined as following relation [30].

(15) NSL= $\frac{et_n}{\sum_{t_i \in cp} w(t_i)}$

And also to contrast profitability rate of computing servers we used the following relation:

(16)

$$Useful_{Rate} = Useful\_AllServer * 100$$

We have applied both approaches on graphs with the same parameters and also in each experiment we evaluated the effects of changing one parameter from workflow graph on approaches. All of the data in graphs is the average of the results getting from the applied approaches on 50 different entrance graphs with the same parameters.

Experiment 1

In this experiment we had applied algorithms on graphs with the same parameters on different number of servers. We have shown the useful rate and NSL resulted from two algorithms.

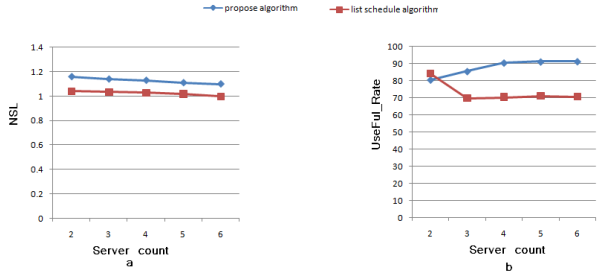

Fig 4- Analyzing efficiency of two approaches based on changing the number of servers.

According to the results of third experiment by increasing the number of servers, the represented approach looking for better and more effective use of them because by increasing the number of servers sub-matrixes are increasing, so the tasks in each matrix have higher consistency and this issue result in a higher profitability of servers. Also based on figure 6-a by increasing the number of servers, schedule length will decrease in both algorithms, but we can see a higher schedule length in represented approach again.

Experiment 2

In this experiment we have applied algorithms on graphs with the same parameters and also we have shown the results of two algorithms affected by changing communicative coefficient among tasks. Figure 5 shows the useful rate and NSL for two algorithms.

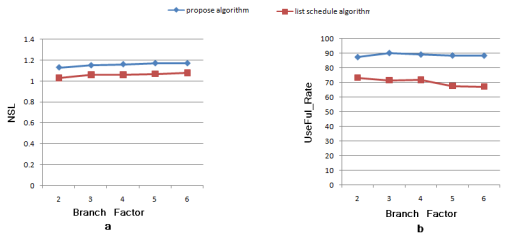

Fig 5- Comparing two approaches effected by changing coefficient of communication among workflow graph tasks

According to the results getting from the second experiment, the represented approach shows stable results by increasing communicative coefficient among tasks, but in the schedule list approach the profitability of servers will be decreased by increasing communicative coefficient and also because of the interval for applying tasks on each server. But according to this experiment the schedule duration is greater in the represented approach.

RESULTS AND DISCUSSIONS

Scientific workflows play an important role in today's science. Many disciplines rely on workflow technologies to orchestrate the execution of thousands of computational tasks. Much research to-date focuses on efficient, scalable, and robust workflow execution, especially in distributed environments. However, many challenges remain in the area of data management related to workflow creation, execution, and result management. According to the paying computing costs, saving, network and …based on the rate of usage, geographical domain and cloud computing system- based on internet and also simultaneous requests from different users, algorithms that are used to map applications in this system should prevent from being uselessness. the advantages of this issue are increasing the usage efficiency of computing servers, decreasing payment costs to the clients of the cloud computing system services , increasing the ability of cloud computing service suppliers in response to simultaneous requests from the other clients, decreasing workflow completion time that lead to reducing clients patience.

REFERENCES

[1] H. El-Rewini, T.G. Lewis and H.H. Ali, Task Scheduling in Parallel and Distributed Systems, Prentice-Hall International Editions (1994).

[2] M. Rahman, R. Ranjan and R. Buyya, Cooperative and decentralized workflow scheduling in global grids, Future Generation Comp. Syst. 26(5) (2010), PP 753-768.

[3] M. Brantner, D. Florescuy, D. Graf, D. Kossmann, T. Kraska, Building a Database on S3, in: SIGMOD, Vancouver, BC, Canada, 2008, pp. 251–263.

[4] R. Grossman, Y. Gu, Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere, in: SIGKDD, 2008, pp. 920–927.

[5] R. Buyya, C.S. Yeo, S. Venugopal, Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities, in: 10th IEEE International Conference on High Performance Computing and Communications, HPCC-08, Los Alamitos, CA, USA, 2008.

[6] C. Moretti, J. Bulosan, D. Thain, P.J. Flynn, All-Pairs: An abstraction for data-intensive cloud computing, in: IEEE International Parallel & Distributed Processing Symposium, IPDPS'08, 2008, pp. 1–11.

[7] A. Weiss, Computing in the Cloud, vol. 11, ACM Networker (2007) 18–25.

[8] R. Buyya, C. Yeo, S. Venugopal, J. Broberg and I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems, 25(6) (2009), pp. 599-616.

[9] I. Ahmad and Y.-K. Kwok, On exploiting task duplication in parallel program scheduling, IEEE Trans. Parallel Distrib. Syst. 9 (9) (1998), pp. 872–892.

[10] S. Darbha and D.P. Agrawal, Optimal scheduling algorithm for distributed -memory machines, IEEE Trans. Parallel Distrib. Syst. 9 (1) (1998), pp. 87–95.

[11] C.H. Papadimitriou and M. Yannakakis, Towards an architecture-independent analysis of parallel algorithms, SIAM J. Comput. 19 (2) (1990), pp. 322–328.

[12] M.K. Dhodhi, I. Ahmad and A. Yatama et al., An integrated technique for task matching and scheduling onto distributed heterogeneous computing system, J. Parallel Distrib. Comput. 62 (9) (2002), pp. 1338–1361.

[13] H.J. Park and B.K. Kim, An optimal scheduling algorithm for minimizing the computing period of

cyclic synchronous tasks on multiprocessors, J. Syst. Softw. 56 (3) (2001), pp. 213–229.

[14] Xiaoyong Tang, Kenli Li, Guiping Liao and Renfa Li, List scheduling with duplication for heterogeneous computing systems, J. Parallel Distrib. Comput. 70 (4) (2010), pp. 323–329.

[15] D. Kim and B.G. Yi, A two-pass scheduling algorithm for parallel programs, Parallel Comput. 20 (6) (1994), pp. 869–885.

[16] Y.-K. Kwok and I. Ahmad, Dynamic critical-path scheduling: an effective technique for allocating task graphs onto multiprocessors, IEEE Trans. Parallel Distrib. Syst. 7 (5) (1996), pp. 506–521.

[17] S. Bansal, P. Kumar and K. Singh, An improved duplication strategy for scheduling precedence constrained graphs in multiprocessor systems, IEEE Trans. Parallel Distrib. Syst. 14 (6) (2003), pp. 533–544.

[18] G.C. Sih and E.A. Lee, A compile-time scheduling heuristic for interconnection-constrained heterogeneous machine architectures, IEEE Trans. Parallel Distrib. Syst. 4 (2) (1993), pp. 175–187.

[19] Xiaoyong Tang, Li Kenli and D. Padua, Communication contention in APN list scheduling algorithm, Sci. China Ser. F 52 (1) (2009), pp. 59–69.

[20] H. El-Rewini and T.G. Lewis, Scheduling parallel program tasks onto arbitrary target machines, J. Parallel Distrib. Comput. 9 (2) (1990), pp. 138–153.

[21] K. Kaya, B. Uçar and C. Aykanat, Heuristics for scheduling file-sharing tasks on heterogeneous systems with distributed repositories, J. Parallel Distrib. Comput. 67 (2007), pp. 271 – 285.

[22] 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), pp. 342–349.

[23] D. Yuan, Y. Yang, X. Liu, J. Chen, A Data Placement Strategy in Cloud Scientific Workflows , Future Generation Computer Systems (FGCS), 26(8)(2010), pp. 1200-1214.

[24]T. Kosar, S. Son,G. Kola,M. Livny, Data placement in widely distributed environments , Advances in Parallel Computing 14(2005), pp. 105-128.

[25] Fatma A. Omara,Mona M. Arafa, Genetic algorithms for task scheduling problem , Journal of Parallel and Distributed Computing 70(1)(2010), pp. 13-22.

[26] Y. Yuan, X.Li,Q.Wang,X. Zhu, Deadline division-based heuristic for cost optimization in workflow scheduling , Information Sciences 179(15) (2009),pp. 2562-2575 .

[27] X.Tang, K.Li, R. Li,B.Veeravalli, Reliability-aware scheduling strategy for heterogeneous distributed computing systems , Journal of Parallel and Distributed Computing 70(9)(2010),pp. 941-952 .

[28] Z. Shi,Jack J. Dongarra, Scheduling workflow applications on processors with different capabilities , Future Generation Computer Systems 22(6)(2006),pp. 665-675.

[29] Q. Kang, H. He, H. Song and R. Deng, Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization, Journal of Systems and Software 83(11) (2010), pp. 2165-2174.

[30] C. Chiu, Y. Yeh,J.Chou, A fast algorithm for reliability-oriented task assignment in a distributed system, Computer Communications 25(17)(2002),pp. 1622-1630 .

[31] A. zareie, M. M. pedram, M. kelarestaghi, A. kosari, International Journal of Computational Intelligence and Information Security 2(7)(2011), pp . 13-20.

[32] W.T. McCormick, P.J. Sehweitzer and T.W. White, Problem decomposition and data reorganization by a clustering technique, Operations Research 20 (1972), pp. 993–1009.

[33] M.T. Ozsu and P. Valduriez, Principles of Distributed Database Systems, Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1991).

[34] X. Tang, K. Li, R. Li and B. Veeravalli , Reliability-aware scheduling strategy for heterogeneous distributed computing systems, Journal of Parallel and Distributed Computing 70(9)(2010), pp. 941-952.

[35] M.Y. Wu and D.D. Gajski, Hypertool: A programming aid for message-passing systems, IEEE Trans. Parallel Distrib. Syst. 1 (1990) (3), pp. 330–343.

[36] G.C. Sih and E.A. Lee, A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures, IEEE Trans. Parallel Distrib. Syst. 4 (1993) (2), pp. 175–187.

[37] H. El-Rewini and T.G. Lewis, Scheduling parallel program tasks onto arbitrary target machines, J. Parallel Distrib. Comput. 9 (1990) (2), pp. 138–153.

[38] G. Liu, K. Poh and M. Xie, Iterative list scheduling for heterogeneous computing, J. Parallel Distrib. Comput. 65 (2005) (5), pp. 654–665.

[39] R. Sakellariou, H. Zhao, A hybrid heuristic for dag scheduling on heterogeneous systems, in: Proceedings of 13th Heterogeneous Computing Workshop, HCW2004, Santa Fe, NM, 2004.