International Journal of Computer Applications (0975 – 8887)
Volume 64– No.2, February 2013

An Optimized Algorithm for Enhancement of Performance of Distributed Computing System

Pankaj Saxena
Department of Computer Applications
Teerthanker Mahaveer University, Moradabad
(U.P), INDIA.

Kapil Govil, PhD.
Department of Computer Applications
Teerthanker Mahaveer University, Moradabad
(U.P), INDIA.

ABSTRACT
Distributed Computing System (DCS) presents a platform consisting of multiple computing nodes connected in some fashion to which various modules of a task can be assigned. A node is any device connected to a computer network. Nodes can be computers or various other network applications. A task should be assigned to a processor whose capabilities are most appropriate for the execution of that task. In a DCS, a number of tasks are allocated to different processors in such a way that the overall performance in terms of time, cost should be minimized and reliability should be maximized. For a large set of tasks that is being allocated into a DCS, several allocation methods are possible. These allocations can have significant impact on quality of services such as time, cost or reliability. Execution time is the time in which a single instruction is executed. Execution cost can be termed as the amount of value of resource used. In DCS reliability is highly dependent on its network and failures of network have adverse impact on the system performance. In DCS the whole workload is divided into small and independent units, called tasks and it allocates onto the available processors. In this paper a simple algorithm for task allocation in terms of optimum time or optimum cost or optimum reliability is presented where the numbers of tasks are more then the number of processors.

Keywords
Distributed Computing System (DCS), Task, Time, Cost, Reliability.

1. INTRODUCTION
In this paper we consider the problem of designing an algorithm in a Distributed Computing System (DCS) for handling a set of tasks for getting the optimized results in terms of time or cost or reliability. A task is a piece of code that is to be executed and task allocation [10, 18, 20] refers to the way that tasks are chosen, assigned, and coordinated. It is the process that results in specific processor being engaged to process in specific tasks. Task allocation [14, 19, 23] algorithms assign any particular task to specific processor in the distributed network for execution. In DCS for optimized results the cost [3] of system and time factors should be

minimized. A highly reliable system is one that will continue working for a long period of time. Reliability [5, 24] analysis is one of the important parameters to achieve the system efficiency. In a DCS [7], a number of tasks may need to be allocated to different processors such that the reliability of processing successfully these tasks modules is maximized. Execution time is the time in which a single instruction is executed. Execution cost [9] can be termed as the amount of value of resource used [8]. It means paying something to achieve some services. Reliability [16] is defined to be the probability that the system will not fail during the time that it is processing the tasks. Different processors used in distributed systems typically vary in cost [13] based depending on their computing efficiencies. Task allocation [15, 17] is also often done based on estimates of the computation time of each task on each processor. In this paper, we have presented an algorithm, considering DCS with heterogeneous processors in order to achieve optimal time, cost and reliability by allocating the tasks to the processors, in such a way that the load of tasks [25] on each processor is balanced [6]. The required processing power for these applications can not be achieved with a single processor. One approach to this problem is to use DCS that concurrently process an application program by using multiple processors. To optimize the performance of a DCS, several issues arise such as the minimization of time and cost as well as maximization of system reliability.

2. OBJECTIVE
The objective of the present paper is to determine a task allocation [4, 22] scheme so as to enhance the performance of Distributed Computing System (DCS) by minimizing the overall execution cost, or execution time or maximizing the reliability [4] in order to optimize system utilization. The type of assigning task [11] to the processor is static [1, 2, 12,] in nature in this paper. Here we have taken an example of distributed network [21] where the number of processors is lesser in comparison to the number of tasks. Performance is measured in terms of either time or cost or reliability of the modules of a task that have to process on the processors of the network.

3. NOTATIONS

T	:	Set of tasks
P	:	Set of processors
CM	:	Communication Matrix
PCTR	:	Processor Cost Time Reliability
MPCTR	:	Modified Processor Cost Time Reliability
FPCTR	:	Fused Processor Cost Time Reliability

4. TECHNIQUE

For obtaining the optimal time or cost or reliability for each task initially the emphasis will be on those modules of tasks which have the maximum probability of data transfer. Now, in case of time and cost the elements will be added and in case of reliability they will be multiplied. We have considered a set of task T, which contains three tasks t_1, t_2, and t_3, a set of processors P which contains three processors p_1, p_2 and p_3, also every task contains some number of modules. Now we have taken a matrix in which the time, cost and reliability of modules are defined and define a communication matrix by considering the communication between tasks. On the basis of highest communication we get a matrix namely FPCTR (,,). Now from this table we can get the separate tables for time, cost and reliability. Load count is taken as an integer variable which contains binary values either 1 or 0. We will assign it a value 0 to the processor if no task is assigned otherwise a value 1 will be assigned to the load count. By considering that the preference should be given to the idle processor we assign load count as 1 or 0. The function for obtaining the overall execution time [Etime], cost [Ecost], and reliability [Ereliablity] is as follows-

$$\text{Etime} = \left[\sum_{i=1}^{n} \left\{ \sum_{j=1}^{n} ET_{ij} x_{ij} \right\} \right]$$

(i)

$$\text{Ecost} = \left[\sum_{i=1}^{n} \left\{ \sum_{j=1}^{n} EC_{ij} x_{ij} \right\} \right]$$

(ii)

$$\text{Ereliablit y} = \left[\prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} ER_{ij} x_{ij} \right\} \right]$$

(iii)

$$\text{Where, } x_{ij} = \begin{cases} \geq 1, \text{if } i^{th} \text{ task is assigned to } j^{th} \text{ processor} \\ 0, \text{Otherwise} \end{cases}$$

Macintosh, use the font named Times. Right margins should be justified, not ragged.

5. ALGORITHM

Step 1: Start Algorithm

Step 2: Take the set of different tasks T, Set of different processors P and different modules in each task.

Step 3: Input the matrix PCTR (,,). Select time/cost/reliability data corresponding to each task as needed.

Step 4: Input matrix CM (,,) by considering the communication time between modules of each task.

Step 5: Consider each task on the basis of time or cost or reliability.

Step 6: On the basis of Step 5 we get the matrix MPCTR (,,).This matrix will be derived from matrix PCTR (,,).

Step 7: Fused the modules of tasks in MPCTR (,,), on the basis of highest communication we get the matrix FPCTR (,,).

Step 8: From FPCTR(,,) we can take bifurcate tables for time, cost and reliability with table names Table I for Time, Table II for Cost and Table III for Reliability. The tasks will be allocated on the basis of minimization of time and cost and maximization of reliability.

Step 9: Assign a load count 0 to the processor if no task is assigned and give preference to this processor in such a way that an idle processor should be busy for allocating task; otherwise assign a load count as 1.

Step 10: Calculate Etime, Ecost and Ereliability form the table in step 9.

Step 11: End algorithm

6. IMPLEMENTATION

Let us consider a set of tasks T. This set consists three tasks t_1, t_2, and t_3.We may define it as, T= $\{t_1, t_2, t_3\}$. Now, consider the task t_1 has a set of task M_1. This set consists the five modules. We may define it as, M_1= $\{m_{11}, m_{12}, m_{13}, m_{14}, m_{15}\}$. Now, for task t_2 consider the set of task M_2. This set consists the four modules. We may define it as, M_2= $\{m_{21}, m_{22}, m_{23}, m_{24}\}$. Now, for task t_3 consider the set of task M_3. This set consists the six modules. We may define it as, M_3= $\{m_{31}, m_{32}, m_{33}, m_{34}, m_{35}, m_{36}\}$. The total number of processors are three and it can be define as, P= $\{p_1, p_2, p_3\}$. The graphical representation of this problem is shown in figure 1.

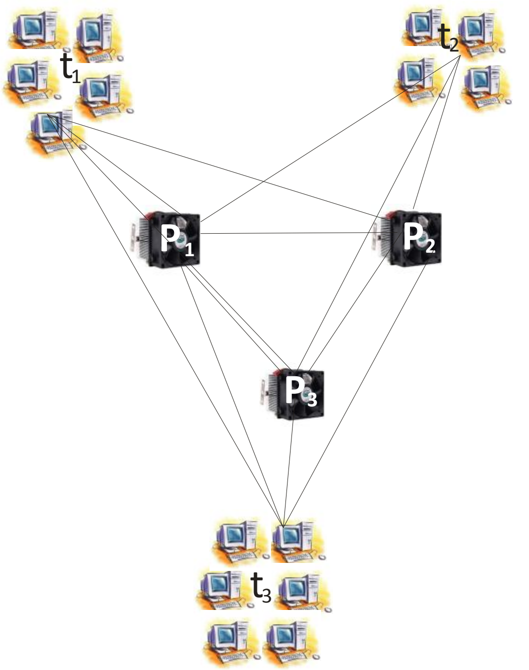

Figure 1: Processors and Tasks

For different task t_1, t_2 and t_3 there are three set of tasks M_1, M_2 and M_3. These set of tasks contains different individual tasks components which are called modules. The processing time (t), cost (c) and reliability (r) of each module of every task on various processors are known and mentioned in the matrix namely, PCTR (,,)-

	Processors	p_1	p_2	p_3
Tasks	Modules	t – c – r	t – c – r	t – c – r
	m_{11}	132 – 2500 – 0.999276	127 – 2300 – 0.999132	140 – 2500 – 0.999234
	m_{12}	140 – 2700 – 0.999188	135 – 2200 – 0.999234	130 – 2300 – 0.999787
t_1	m_{13}	150 – 2600 – 0.999231	145 – 2700 – 0.999342	150 – 2600 – 0.999899
	m_{14}	160 – 2700 – 0.999150	170 – 2200 – 0.999456	160 – 2300 – 0.999988
	m_{15}	120 – 2200 – 0.999120	130 – 2800 – 0.999566	110 – 2000 – 0.999700
	m_{21}	110 – 2800 – 0.999100	160 – 2600 – 0.999666	120 – 2400 – 0.999800
	m_{22}	180 – 2700 – 0.999227	170 – 2100 – 0.999765	175 – 2200 – 0.999678
t_2	m_{23}	150 – 2800 – 0.999123	160 – 2560 – 0.999786	180 – 2360 – 0.999899
	m_{24}	160 – 2700 – 0.999555	170 – 2390 – 0.999334	170 – 2390 – 0.999333
	m_{31}	170 – 2800 – 0.999567	165 – 2300 – 0.999412	150 – 2400 – 0.999123
	m_{32}	180 – 2300 – 0.999444	150 – 2100 – 0.999229	140 – 2000 – 0.999200
	m_{33}	170 – 2800 – 0.999123	140 – 2500 – 0.999447	130 – 2750 – 0.999555
t_3	m_{34}	140 – 2900 – 0.999222	110 – 2400 – 0.999444	120 – 2200 – 0.999788
	m_{35}	120 – 2700 – 0.999221	140 – 2500 – 0.999111	130 – 2300 – 0.999667
	m_{36}	170 – 2400 – 0.999221	150 – 2300 – 0.999671	140 – 2900 – 0.999899

The considered communication time amongst the modules of each task is mentioned in the matrices, namely CM (,,).

For task t_1, the matrix CM (1,) is as:

$$\begin{bmatrix} & m_{11} & m_{12} & m_{13} & m_{14} & m_{15} \\ m_{11} & 0 & 4 & 6 & 8 & 5 \\ m_{12} & & 0 & 6 & 4 & 3 \\ m_{13} & & & 0 & 4 & 1 \\ m_{14} & & & & 0 & 4 \\ m_{15} & & & & & 0 \end{bmatrix}$$

For task t_2, the matrix CM (2,) is as:

$$\begin{bmatrix} & m_{21} & m_{22} & m_{23} & m_{24} \\ m_{21} & 0 & 2 & 5 & 3 \\ m_{22} & & 0 & 8 & 2 \\ m_{23} & & & 0 & 5 \\ m_{24} & & & & 0 \end{bmatrix}$$

For task t_3, the matrix CM (3,) is as:

$$\begin{bmatrix} & m_{31} & m_{32} & m_{33} & m_{34} & m_{35} & m_{36} \\ m_{31} & 0 & 2 & 4 & 3 & 5 & 1 \\ m_{32} & & 0 & 3 & 4 & 4 & 2 \\ m_{33} & & & 0 & 7 & 7 & 6 \\ m_{34} & & & & 0 & 8 & 1 \\ m_{35} & & & & & 0 & 6 \\ m_{36} & & & & & & 0 \end{bmatrix}$$

Here it is considered that task t_1 is based on the constraint of execution time (one may choose the either cost or reliability constraint), task t_2 is based on the constraint of cost (one may choose the either time or reliability constraint), and task t_3 is based on the constraint of reliability (one may choose the either time or cost constraint).

Hence, we can use the following form of data from matrix PCTR (,,) i.e. execution time for task t_1, execution cost for task t_2, and execution reliability for task t_3 and can get the matrix namely MPCTR (,,) in the following way-

Tasks	Processors / Modules	p_1 t − c − r	p_2 t − c − r	p_3 t − c − r
	m_{11}	132−⋯−⋯	127−⋯−⋯	140−⋯−⋯
	m_{12}	140−⋯−⋯	135−⋯−⋯	130−⋯−⋯
t_1	m_{13}	150−⋯−⋯	145−⋯−⋯	150−⋯−⋯
	m_{14}	160−⋯−⋯	170−⋯−⋯	160−⋯−⋯
	m_{15}	120−⋯−⋯	130−⋯−⋯	110−⋯−⋯
	m_{21}	⋯−2800−⋯	⋯−2600−⋯	⋯−2400−⋯
	m_{22}	⋯−2700−⋯	⋯−2100−⋯	⋯−2200−⋯
t_2	m_{23}	⋯−2800−⋯	⋯−2560−⋯	⋯−2360−⋯
	m_{24}	⋯−2700−⋯	⋯−2390−⋯	⋯−2390−⋯
	m_{31}	⋯−⋯−0.999567	⋯−⋯−0.999412	⋯−⋯−0.999123
	m_{32}	⋯−⋯−0.999444	⋯−⋯−0.999229	⋯−⋯−0.999200
	m_{33}	⋯−⋯−0.999123	⋯−⋯−0.999447	⋯−⋯−0.999555
t_3	m_{34}	⋯−⋯−0.999222	⋯−⋯−0.999444	⋯−⋯−0.999788
	m_{35}	⋯−⋯−0.999221	⋯−⋯−0.999111	⋯−⋯−0.999667
	m_{36}	⋯−⋯−0.999221	⋯−⋯−0.999671	⋯−⋯−0.999899

Now, the task t_1 have five modules so on the basis of highest communication the modules m_{11}&m_{14}, m_{12}&m_{13} will be fused. The task t_2 have four modules so on the basis of highest communication the modules m_{22}&m_{23} will be fused. The task t_3 have six modules so on the basis of highest communication the modules m_{34}&m_{35}, m_{33}&m_{36}, m_{31}&m_{32} will be fused. The resulting matrix namely FPCTR (,,) will be-

Tasks	Processors / Modules	p_1 t − c − r	P_2 t − c − r	P_3 t − c − r
	$m_{11} * m_{14}$	292	297	300
t_1	$m_{12} * m_{13}$	290	280	280
	m_{15}	120	130	110
	$m_{22} * m_{23}$	5500	4660	4560
t_2	m_{21}	2800	2600	2400
	m_{24}	2700	2390	2390
	$m_{34} * m_{35}$	0.998443	0.998555	0.999455
t_3	$m_{33} * m_{36}$	0.998344	0.999118	0.999454
	$m_{31} * m_{32}$	0.999011	0.998641	0.998323

Now, from the table FPCTR (,,), we can get the three tables namely Table I, Table II and Table III which are given below-

Table I

	$m_{11} * m_{14}$	$m_{12} * m_{13}$	m_{15}
p_1	292	290	120
p_2	297	280	130
p_3	300	280	110

Table II

	$m_{22} * m_{23}$	m_{21}	m_{24}
p_1	5500	2800	2700
p_2	4660	2600	2390
p_3	4560	2400	2390

Table III

	$m_{34} * m_{35}$	$m_{33} * m_{36}$	$m_{31} * m_{32}$
p_1	0.998443	0.998344	0.999011
p_2	0.998555	0.999118	0.998641
p_3	0.999455	0.999454	0.998323

In the context of Table I we have to assign unallocated task to processor where load count will take place when we assign any task to the processor. In Table I we have to allocate module $m_{11}*m_{14}$ to any processor which has load count 0 and/or less execution time. So we assign $m_{11}*m_{14}$ to processor $p_1$and mark load count 1 to processor p_1.for $m_{12}*m_{13}$, we assign it to processor p_2 with load count 1 and m_{15} to processor p_3 with load count 1.

Table IV

Processors	Alllocated Tasks	Time	Load Count	Etime
p_1	$m_{11} * m_{14}$	292	1	
p_2	$m_{12} * m_{13}$	280	1	682
p_3	m_{15}	110	1	

Results of Table IV can be shown graphically as in figure 2-

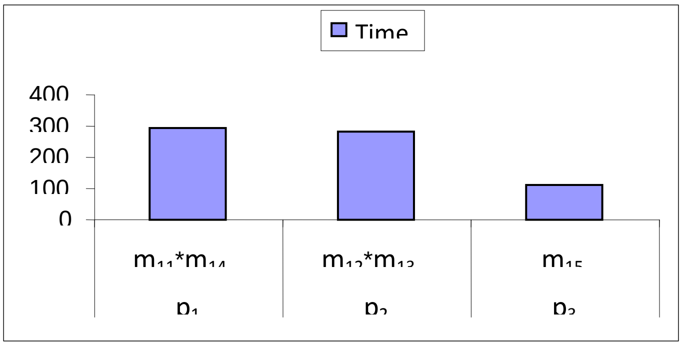

Figure 2: Time for tasks

Similarly we can compute allocation on the basis of execution cost and are mentioned in Table V

Table V

Processors	Alllocated Tasks	Cost	Load Count	Ecost
p_1	$m_{22}*m_{23}$	5500	1	
p_2	m_{24}	2390	1	10290
p_3	m_{21}	2400	1	

Results of Table V can be shown graphically as in figure 3-

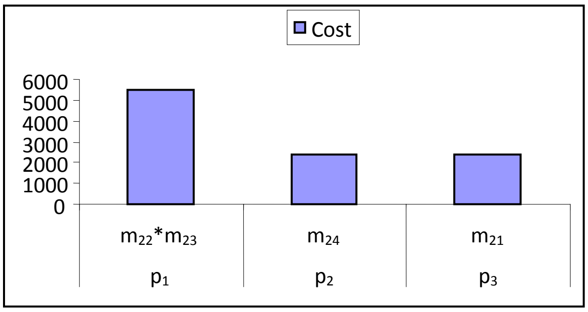

Figure 3: Cost for tasks

Computation for execution reliability is mentioned in table VI-

Table VI

Processors	Alllocated Tasks	Reliablity	Load Count	Ereliablit y
p_1	$m_{31}*m_{32}$	0.999011	1	
p_2	$m_{33}*m_{36}$	0.999118	1	0.997585
p_3	$m_{34}*m_{35}$	0.999455	1	

Results of Table VI can be shown graphically as in figure 4-

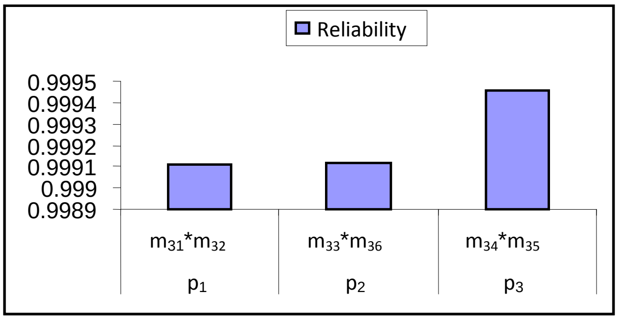

Figure 4: Reliability for tasks

4. CONCLUSION

The performance tests show that the proposed algorithm is almost always superior in comparison with others by providing different inputs. In the present paper we have taken a problem where the numbers of tasks are greater then the number of processors in a Distributed Computing System (DCS). The presented algorithm describes an efficient way to calculate the optimal results for time, cost and reliability. The three tasks taken in this paper which are t_1, t_2, and t_3 are solved in such a way to process the task t_1 in minimum time, to process the task t_2 in minimum cost and to process the task t_3 in maximum reliability. The calculated optimal results can be shown the following table-

Tasks	p_1	p_2 (Processors)	p_3	Optimal Etime	Optimal Ecost	Optimal Ereliablit y
t_1	$m_{11}*m_{14}$	$m_{12}*m_{13}$	m_{15}	682	⋯	⋯
t_2	$m_{22}*m_{23}$	m_{24}	m_{21}	⋯	10290	⋯
t_3	$m_{31}*m_{32}$	$m_{33}*m_{36}$	$m_{34}*m_{35}$	⋯	⋯	0.997585

5. REFERENCES

[1]. Braun Tracy, D., Siegel Howard Jay, Maciejewski Anthony, A., and Hong Ye, 2008. Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions. Journal of Parallel and Distributed Computing, Volume 68, Issue 11, 1504-1516.

[2]. Bo Yang, Huajun Hu, and Suchang Guo, 2009. Cost-oriented task allocation and hardware redundancy policies in heterogeneous distributed computing systems considering software reliability. Journal of Computers & Industrial Engineering, Volume 56, Issue 4, 1687-1696.

[3]. Dr. Kapil Govil, 2011. Processing Reliability based a Clever Task Allocation Algorithm to Enhance the Performance of Distributed Computing Environment. International Journal of Advanced Networking and Applications, Volume 3, Issue 1, 1025-1030.

[4]. Deo Prakash Vidyarthia, and Anil Kumar Tripathib, 2001. Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm. Journal of Systems Architecture, Volume 47, Issue 6, 549-554.

[5]. Ghjh Edward, A., Billard, Joseph, C., and Pasquale**,** 1997. Load balancing to adjust for proximity in some network topologies. Journal of Parallel Computing, Volume 22, Issue 14, 2007-2023.

[6]. Herlihy Maurice, and Luchangco Victor, 2008. Distributed computing and the multicore revolution. Journal of ACM SIGACT News, Volume 39, Issue 1, 62-72.

[7]. Henri Casanova, Frederic Desprez, and Frederic Suter, 2010. On cluster resource allocation for multiple parallel task graphs. Journal of Parallel and Distributed Computing, Volume 70, Issue 12, 1193-1203.

[8]. Hsieh, Chung-Chi, and Hsieh Yi-Che, 2003. Reliability and cost optimization in distributed computing systems. Journal of Computers & Operations Research, volume 30, Issue 8, 1103-1119.

[9]. Kołodziej Joanna, and Xhafa Fatos, 2007. Modern approaches to modeling user requirements on resource and task allocation in hierarchical computational grids.

International Journal of Applied Mathematics and Computer Science, Volume 21, Issue 2, 243–257.

[10]. Marwa Shouman, Gamal Attiya, Ibrahim, Z., Morsi, 2011. Static Workload Distribution of Parallel Applications in Heterogeneous Distributed Computing Systems with Memory and Communication Capacity Constraints. International Journal of Computer Applications, Volume 34, Issue 6, 18-24.

[11]. Manoj, B.S., Sekhar Archana, and Siva Ram Murthy, C., 2009. A state-space search approach for optimizing reliability and cost of execution in distributed sensor networks, Journal of Parallel and Distributed Computing, Volume 69, Issue 1, 12-19.

[12]. Manisha Sharma, Harendra Kumar, and Deepak Garg, 2012. An Optimal Task Allocation Model through Clustering with Inter-Processor Distances in Heterogeneous Distributed Computing Systems. International Journal of Soft Computing and Engineering, Volume 2, Issue 1, 50-55.

[13]. Nirmeen, A., Bahnasawy, Fatma Omara, Magdy, A., Koutb, and Mervat Mosa, 2011. A new algorithm for static task scheduling for heterogeneous distributed computing system. International Journal of Information and Communication Technology Research, Volume 1, Issue 1, 10-19.

[14]. Pankaj Saxena, Kapil Govil, Rajendra Belwal, and Umesh Kumar, 2011. An efficient approach for optimal task allocation through optimizing processing time in Distributed Network. In Proceeding of International Conference on the Next Generation Information Technology Summit, Amity University, Noida, January 27-28.

[15]. Pankaj Saxena, Dr. Kapil Govil, Neha Agrawal, Saurabh Kumar, and Deep Narayan Mishra, 2012. An approach for allocating tasks in optimized time in a distributed processing environment. International Journal of Innovative Research and Development, Volume 1, Issue 5, 431-437.

[16]. Pankaj Saxena, and Dr. Kapil Govil, 2012. A time efficient algorithm for static allocation of tasks on processors in a distributed computing system. In Proceeding of International Conference on System Modeling & Advancement in Research Trends, Teerthanker Mahaveer University, Moradabad, Oct 20-21.

[17]. Pankaj Saxena, Dr. Kapil Govil, Gaurav Saxena, Saurabh Kumar, and Neha Agrawal, 2012. An algorithmic approach and comparative analysis of task assignment to processor for achieving time efficiency in process completion. International Journal of Applied Engineering and Technology, Volume 2, Issue 1, 114-119.

[18]. Peng-Yeng Yin, Shiuh-Sheng Yu, Pei-Pei Wang, and Yi-Te Wang, 2007. Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization, Journal of Systems and Software, Volume 80, Issue 5, 724-735.

[19]. Pradeep Kumar Yadav, Singh, M.P., and Kuldeep Sharma, 2011. Task Allocation Model for Reliability and Cost optimization in Distributed Computing System, International Journal of Modeling, Simulation and Scientific Computations, Volume 2, Issue 2, 1-19.

[20]. Qin-Ma Kang, Hong He, Hui-Min Song, and Rong Deng, 2010. Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization, Journal of Systems and Software, Volume 83, Issue 11, 2165–2174.

[21]. Santhanam Srinivasan, and Niraj, K., Jha, 1999. Safety and Reliability Driven Task Allocation in Distributed Systems, IEEE Transactions on Parallel and Distributed Systems, Volume 10, Issue 3, 238-251.

[22]. Shatz, S.M., Wang, J.P., Goto, M., 1992. Task Allocation for Maximizing Reliability of Distributed Computer Systems, IEEE Transaction on Computers, Volume 41, Issue 9, 1156-1168.

[23]. Ueno Yoichiro, Miyaho, and Suzuki Shuichi, 2009. Disaster recovery mechanism using widely distributed networking and secure metadata handling technology, Workshop on Use of P2P, GRID and agents for the Development of Content Networks, 45-48.

[24]. Vidyarthi, D.P., and Tripathi, A.K., 2001.Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm, Journal of System Architecture, Volume 47, Issue 6, 549-554.

[25]. Zubair Khan, Ravinder Singh, and Jahangir Alam, 2012. Task allocation using fuzzy inference in parallel and distributed system, Journal of Information and Operations Management, Volume 3, Issue 2, 322-326.