Optimal Tasks Allocation Process Based on Fusion of the Unallocated Tasks for Distributed Systems

Sarvesh Kumar Dubey Virendra Upadhyay

dubey.sarvesh25@gmail.com drvirendra.upadhyay@gmail.com

Department of Physical Sciences

Mahatma Gandhi Chitrakoot Gramodaya Vishwavidyalay

Chitrakoot, Satna- 485780, M. P; INDIA

Avanish Kumar

Department of Mathematical Sciences & Computer Applications

Bundelkhand University

Jhansi-284128, U. P; INDIA

dravanishkumar@yahoo.com

***Abstract*- Tasks partitioning and systematic tasks allocation to processors over the distributed real time system for proper utilization of processor's capacity are very much important factors in distributed systems. Efficient use of available processors enhances the performances of the distributed systems and provide an optimal solution. This paper presents a new mathematical model consisting of a set of m tasks to n processors (where, m >> n) which performs static allocation of such software program modules that minimizes the application program's parallel EC, ITCC and maximizes the overall throughput of the system in such a way that the allocated load on all the processors are balanced. The heuristic model deals with the optimal tasks allocation by the fusion of the unallocated tasks with the other allocated tasks.**

Keywords: *DCS, Task Allocation, Parallel Computation, EC, ITCC, Processors Utilization, Throughput, Load Balancing.*

I. INTRODUCTION

In the past years, Distributed Computing Systems(DCS) have become a key platform in order to achieve optimal cost by allocating number of tasks to different processors for execution in task allocation problems. The configuration of a DCS is the set of autonomous processors inter-connected by a high bandwidth communication link to coordinate their actions to achieve a common goal. DCS has become a dominant paradigm in computer architecture, mainly in the form of multi-core processors. If two, or three, or more processors are available, than many of the programs can be executed more quickly. While one processor is doing one aspect of computation, others can work on another and share the data. In this way DCS employs different processing components simultaneously to solve a problem. In this form of computing, the work is accomplished by breaking the large problems into the smaller ones, which are than solved concurrently. DCS has attracted many researchers by posing several challenging problems. A major problem encountered in DCS is the task allocation to the processors so that the system cost i.e. sum of execution cost and communication cost is minimized without violating any of the system constraints [1]. A best allocation of tasks leads to a best balancing of the system [2]. The performance of the DCS may suffer if the distribution of resource is not carefully implemented. In order to make the best use of resources in a DCS, it becomes essential to maximize the overall throughput by allocating the tasks to processors in such a way that the allocated load on all the processors should be balanced [3]. The tasks partitioning and task allocation activities influence the distributed software properties such as IPC [4, 5]. Many approaches have been reported for solving the task assignment problems in DCS. The problem of finding an optimal dynamic assignment of a modular program for a two processor system is analyzed by [6]. The main incentives for choosing DCS are higher throughput, improved availability and better access to a widely communicated web of information [7]. One measure of usefulness of a general purpose DCS is the system's ability to provide a level of performance commensurate to the degree of multiplicity of resources present in the system. Taxonomy of approaches to the resource management problem is reported [8]. The taxonomy, while presented and discussed in terms of distributed scheduling, is also applicable to most types of resource management. A model for allocating information files have been reported by [9] the model considers storage cost, transmission cost, file lengths, and request rates, as well as updating rates of files.

The coordination of dynamic task allocation based on available resources is very challenging in DCS. Task allocation and load balancing has been extensively researched in past decades and a large number of related studies and results have been presented concerning this topic [10, 11, 12, 13]. A task allocation model that allocates application tasks among processors in DCS satisfying: minimum inter-processor communication cost, balanced utilization of processors and all engineering requirements has been reported by Perng –Yi Richard Ma et. al. [14]. This problem of task allocation in heterogeneous distributed systems with goal of maximizing the system reliability has been addressed [11]. The model is based on well-known simulated annealing (SA) technique. Yadav et. al. [15] have formulated an algorithm for Reliability Evaluation of Distributed System based on Failure Data Analysis. J. B. Sinclair [16] asserted for finding an optimal assignment of the modules of a program to processors in a distributed system by multiprocessor scheduling with the aid of the network flow algorithms. A module incurs an execution cost that may be different for each processor assignment, and modules that are not assigned to the same processor but that communicate with one another incur a communication cost. Sagar et. al. [17] used the matrix reduction technique and reported that a task is selected randomly to start with and then assigned to a processor. Kumar et. al. [18, 19] proposed algorithms for distributed processing system and they clustered heavily communicated tasks and allocate them to same processors. DCS offers the potential for improved performance and resource sharing. To make the best use of the computational power available in multi processing system, it is essential to assign the tasks dynamically to that processor whose characteristics are most appropriate for the execution of the tasks in distributed processing system [13, 20]. Elsadek et.al. [21] proposed a new methodology which expands the Maximally Linked Module algorithm by incorporating simulated annealing and produces complete allocations for assigning static modules to processors of a distributed system.

Communication technology has completely changed the modern life standard and web services play a crucial role in this technology. It has opened new avenues for processing, sharing and transferring of data. Apart from sharing data and I/O devices, DCS also share the computational power among the nodes which improves the system performance [22]. Yadav et. al. [23, 24] and Singh et. al. [24,25] reported task scheduling and represented ANN based task scheduling model for load distribution in fully connected client server network using feedback neural network architecture. Load distribution is important for performance enhancement of the distributed systems. It means the allocation of workload over the distributed system optimally [26]. Analysis of load distribution in DCS through systematic Allocation Task and an exhaustive approach of performance analysis to the distributed systems based on cost assignments have been reported by Kumar et. al. [27, 28].

Our objective of this paper is to minimize the total program Execution Cost(EC). A heuristic model utilized the mathematical programming technique for execution of the tasks assuming, when a task is assigned to a processor, it remains there while the characteristic of computation change, and a new assignment must be computed. During formation of an algorithm it is assumed that the number of tasks is greater than the number of processors. Let it be **m** > **n**, where **m** tasks is to be executed on a set of **n** processors, which have different processing capabilities. It is considered that the EC of a task on each processor is known, if a task is not executable on any of the processor due to the reason of some missing resources than the EC of that task is taken to be infinite (∞). When a task is executed on a processor, it stores the output data in its local memory and when needed, the processor read it from there. Whenever a group of task is assigned to the same processor, the Inter Task Communication Cost(ITCC) between them is zero. Completion of a program from computational point of view means that all related tasks or modules have got executed. The developed model is based on matrix reduction technique and the performance, effectiveness and efficiency of it has been implemented on several sets of input data. It is found that the algorithm is suitable for arbitrary number of tasks and processors with random program structure.

II. NOTATIONS

T : the set of tasks of a parallel program to be executed
P : the set of processors in DCS
n : the number of processors
m : the number of tasks forming a program
k : the number of clusters
t_i : i^{th} task of the given program
P_j : j^{th} processor in set of processor P
X_{ij} : the decision variable s.t. $X_{ij} = 1$, if i^{th} task is allocated to j^{th} processor, $X_{ij} = 0$, otherwise.
ec_{ik} : incurred execution cost (EC), if i^{th} task is executed on k^{th} processor
cc_{ij} : incurred inter task communication cost between task t_i and t_j, if they are executed on separate processor

T_{ass} { } : a linear array to hold assigned tasks
T_{non_ass} { } : a linear array to hold non assigned tasks.
$L_{avg}(P_j)$: Average load on a processor
W_j : Total work load i.e. summation of a maximum module execution cost on different processors.
$T_{RP}(j)$: Total throughput of the processors

III. FORMULATION OF TASK ASSIGNMENT PROBLEM

The problem being addressed here is concerned with an optimal task allocation in DCS, which consists of an application program of a set of **m** communicating tasks T = { $t_1, t_2, t_3, \ldots\ldots\ldots t_m$ } and a DCS with **n** processors P = { $p_1, p_2, p_3, \ldots\ldots p_n$ } interconnected by communication network and let **m** > **n**. An allocation of tasks to processors is defined by a function A_{alloc} from the set T of tasks to the set P of processors s.t. :

$A_{alloc} : T \rightarrow P$, where $A_{alloc}(i) = j$ if task t_i is assigned to processor P_j, $1 \le i \le m$, $1 \le j \le n$.

Our objective is to minimize the ITCC and balance the processing load. The processing cost of these tasks on all the processors is given in the form of Execution Cost Matrix[ECM (,)] of order **m**x**n**, the ITCC is taken in the form of a symmetric matrix named as Inter Task Communication Cost Matrix[ITCCM (,)]. It is very essential to optimize the overall throughput of the processors by assigning the tasks in a manner that the allocated load on all the processors shall have to be balanced. The proposed methodology includes:

- Identification of allocation by Minimally-linked modules MLM
- Calculate total processing cost of each processor
- Calculate the throughput of the processors
- Identification of optimal response time of the system
- Calculation of average load on each processors

A. *Definitions*

1. *Execution Cost*(EC): The EC ec_{ij} where $1 \le i \le m$, $1 \le j \le n$ of each task t_i depends on the processor p_j to which it is assigned and the work to be performed by each of tasks of that processor p_j. The processing Execution cost EC of the tasks on all the processors is given in the form of [ECM (,)] of order **m** x **n**. The EC of a given assignment on each processors are calculated by Equation (1)

$$PEC(j) = \sum_{j=1}^{n} ec_{ij} x_{ij}, \quad i = 1, 2, 3, \ldots\ldots m, \qquad \ldots(1)$$

where, $x_{ij} = \begin{cases} 1, & \text{if task } t_i \text{is assigned to processor } p_j \\ 0, & \text{otherwise} \end{cases}$

2. *Inter Task Communication Cost*(ITCC): The ITC cost cc_{ik} of the interacting tasks t_i and t_k is the minimum cost required for the exchange of data units between the processors during the process of execution.

$$ITCC(j) = \sum_{i<k} cc_{ik} x_{ij}, \quad j = 1, 2, 3, \ldots\ldots n, \qquad \ldots(2)$$

where, $x_{ij} = \begin{cases} 1, & \text{if task } t_i \text{is assigned to processor } p_j \\ 0, & \text{otherwise} \end{cases}$

3. *Response Time Cost of the System*(RTC): The RTC of the system is a function of amount of computation to be performed by each processor and the computation time. This function is defined by considering the processor with the heaviest aggregate computation and communication load. Response time of the system for a given assignment is defined by

$$\text{RTC (Aalloc)} = \max_{1 \le j \le n} \{ PEC(Aalloc)_j + IPC(Aalloc)_j \} \qquad \ldots(3)$$

4. *Average Load*: For each node there are *N* values representing the execution cost required for the corresponding task to be processed on each of the *N* processors. These values are in the matrix [ECM(,)]. Each edge is labeled with a value that represents the communication cost needed to exchange the data when the tasks reside on different processors. Average load on a processor is the sum of EC of the tasks assigned to it then the average load that must be assigned to each processor P_j is calculated as

$$Lavg(P_j) = \frac{W_j}{m}, \quad j = 1, 2, 3, \ldots\ldots n \qquad \ldots(4)$$

where, $W_j = \sum_{1 \le i \le m} et_{ij}, \quad j = 1, 2, 3, \ldots\ldots n.$

The total workload *W* is the summation of the maximum module execution cost on the different processors as shown

$$Tlod = \sum_{1}^{n} Lavg(P_j) \qquad \text{….(5)}$$

The *Lavg(*P_j) on a processor depends upon the different tasks on each processor. The system is considered to be balanced if the load on each processor is equal to the processor average load within a given (small percentage) tolerance.

B. *Assumptions*

Several assumptions have been made to keep the algorithm reasonable in size while designing. The program is assumed to be collection of **m** tasks to be executed on a set of **n** processors, which have different processing capabilities. A task may be portion of executable code or a data file. The number of tasks to be allocated is more than the number of processors (**m** > **n**), as normally is the case in real life distributed processing environment. It is assumed that the EC of a task on each processor is known, if a task is not executable on any of the processor due to absence of some resources. The EC of that task is on that processor is taken to be infinite (∞). We assume that once a task has completed its execution on a processor, the processor stores the output data of the task in its local memory, if the data is needed by some other task being computed on the same processor, it reads the data from the local memory. Using this fact, the algorithm tries to assign heavily communicating tasks to same processor. Whenever groups of tasks or cluster are assigned to the same processor, the data transfer between them is zero. Completion of a program from computational point of view means that all related tasks have got executed.

IV. THE TASKS ALLOCATION PROCESS

Initially the average load on the processors P_j is obtained by using equation (4) and the total load may be calculated by using the equation (5). The load on each processor is equal to the average load within a reasonable tolerance. In the present study a tolerance factor of 20% of average load has been considered. Determine the **n** Minimally Linked Task (MLT) by using the equation (6) and store the result in two-dimensional array [MLT (,)] the first column represents the task number and second column represents the sum of ITCC of task t_i with all task t_{k-i}. Rearrange the [MLT (,)] in ascending order assuming the second column as sorted key.

$$MLT(i,j) = \sum_{i,k=1}^{m} cc_{ik}$$

….(6)

Augmented ECM(,) by introducing the MLM(,) and sort ECM(,) in increasing order considering MLT as sorting key. To determine initial allocation apply the Yadav et. al. algorithm [28] and store in an array $T_{ass}(j)$ (where j = 1,2,…,n). The processor position are also store in a another linear array $A_{alloc}(j)$. The value of $T_{TASK}(j)$ is also computed by adding the values of $A_{alloc}(j)$ if a task t_i is assigned to processor p_j otherwise continue. The remaining (m-n) task are then store in a linear array $T_{non\_ass}(\ )$. Tasks assigned to processors p_j and stored in $T_{non\_ass}(\ )$ which are obviously:

$$T = T_{ass}(\ ) \cup T_{non\_ass}(\ )$$

All the tasks stored in $T_{non\_ass}(\ )$ fused with those assigned tasks stored in $T_{ass}(\ )$ on the bases of minimum average of EC and ITCC.

The FEC of a task $t_a \in T_{non\_ass}(\ )$ with some other task $t_i \in T_{ass}(\ )$ on processor p_j is obtained as:

$$FEC(j)_{ai} = [ec_{aj} + ec_{ij}\ ],\ (\ 1 \le a \le m,\ 1 \le i \le m,\ 1 \le j \le n,\ i \ne a\ )$$

Let cc_{ai} be the ITCC between $t_a \in T_{non\_ass}(\ )$ and $t_i \in T_{ass}(\ )$. The FITCC for t_a with t_i is computed as:

$$FITCC(j)_{ai} = \sum_{i/t_i \in T_{ass()}} [cc_{ai}]$$

Here, $c_{ai} = 0$ if fused with t_i or a = i and remaining c_{ai} value are added and MAFC is calculated as:

$$MAFC(j)_{ai} = \min\{(FEC_{a1} + FITCC_{a1}), (FEC_{a2} + FITCC_{a2}), \ldots\ldots, (FEC_{an} + FITCC_{an})\}$$

This process will be continued until all the tasks stored in $T_{non\_ass}()$ are fused. After complete allocation is achieved PEC(j) and ITCC(j) by using equation (1) and (2) respectively. Finally, calculate the Total Optimal Cost(TOC) by summing up the values of PEC(j) and ITCC(j), and store the result in a linear array over TOC(j), where j = 1, 2, ….., n. The maximum value of TOC(j) will be the optimal cost of the system:

$$TOC(j) = \text{Max}\{\ PEC(j) + ITCC(j)\ \}$$

The Mean Service Rate(MSR) of the processors in terms of $T_{ass}(j)$ to be computed as and store the results in *MSR(j)* , where, j = 1, 2, …., n.

$$MSR(j) = \frac{1}{PEC(j)},\ \text{where, j = 1, 2, …., n.}$$

The overall throughput of the processors are calculated as and store the results of throughput in the linear arrays *TRP(j),* where, j = 1, 2, ….., n.

$$T_{RP}(j) = \frac{T_{TASK}(j)}{PEC(j)},\ \text{where, j = 1, 2, ….., n.}$$

V. IMPLEMENTATION OF THE ASSIGNMENT MODEL, RESULTS AND DISCUSSIONS

Algorithm has been formulated to model the execution process of tasks in the distributed computing environment. The algorithm can be used for efficient execution of the tasks allocation in distributed computing system without overloading the processors. To justify the application and usefulness of the present method an example of a distributed computing system is considered with the following Inputs:
Number of processors of the system (n) = 3
Number of tasks to be executed (m) = 4
ECM (,), ITCCM(,), MLM(,)

$$ECM(,) = \begin{array}{c|ccc|} & p_1 & p_2 & p_3 \\ t_1 & 9 & 2 & 6 \\ t_2 & 3 & 8 & 7 \\ t_3 & 7 & 10 & 3 \\ t_4 & 3 & 4 & 9 \end{array} \qquad ITCCM(,) = \begin{array}{c|cccc|} & t_1 & t_2 & t_3 & t_4 \\ t_1 & 0 & 1 & 4 & 6 \\ t_2 & 1 & 0 & 2 & 0 \\ t_3 & 4 & 2 & 0 & 8 \\ t_4 & 6 & 0 & 8 & 0 \end{array}$$

Actual average load to be assign to the processor after introducing the 20% Tolerance Factor(TF) the average load may be assign to the processors is calculated and same may be store in linear array *Lavg*( ).

$$Lavg() = \begin{array}{c|c|} p_1 & 9 \\ p_2 & 9 \\ p_3 & 10 \end{array}$$

Determine the Minimally Link Task (MLT) and store the result in MLM(,) as follows:

$$MLM(,) = \begin{array}{c|c|} t_1 & 11 \\ t_2 & 3 \\ t_3 & 14 \\ t_4 & 14 \end{array}$$

Augmented ECM(,) by introducing the MLM(,) and sort ECM(,) in increasing order considering MLT as sorting key.

$$ECM(,) = \begin{array}{c|ccc|c} & p_1 & p_2 & p_3 & MLT \\ t_1 & 9 & 2 & 6 & 11 \\ t_2 & 3 & 8 & 7 & 3 \\ t_3 & 7 & 10 & 3 & 14 \\ t_4 & 3 & 4 & 9 & 14 \end{array} \quad ECM(,) = \begin{array}{c|ccc|c} & p_1 & p_2 & p_3 & MLT \\ t_2 & 3 & 8 & 7 & 3 \\ t_1 & 9 & 2 & 6 & 11 \\ t_3 & 7 & 10 & 3 & 14 \\ t_4 & 3 & 4 & 9 & 14 \end{array} \quad \begin{matrix} Modified \\ ITCCM(,) \end{matrix} = \begin{array}{c|cccc|} & t_1 & t_2 & t_3 & t_4 \\ t_1 & 0 & 1 & 4 & 0 \\ t_2 & 1 & 0 & 2 & 0 \\ t_3 & 4 & 2 & 0 & 8 \\ t_4 & 0 & 0 & 8 & 0 \end{array}$$

Apply Yadav et.al. [28] algorithm to determine the initial allocation. The initial allocation than store in a linear array $T_{ass}()$ and the processor position are store in $A_{alloc}()$. The remaining m-n tasks are stored in $T_{non\_ass}(\ )$. The results are as follows:

$T_{ass}(j)$ = [t_1, t_2, t_3]
$A_{alloc}(\ )$ = [p_2, p_1, p_3]
$T_{non\_ass}(\ )$ = [t_4]

After getting the initial allocation the task stored in $T_{non\_ass}(\ )$ has been selected for assignment i.e., t_4FEC(j), for t_4 with all the stored in $T_{ass}(\ )$ are:

Tasks	Processors	FEC(j)
$t_4 + t_1$	p_2	6
$t_4 + t_2$	p_1	6
$t_4 + t_3$	p_3	12

FITCC(j) for t_4 with the other assigned tasks stored in $T_{ass}(\ )$ are:

Tasks	Processors	FITCC(j)
$t_4 + t_1$	p_2	13
$t_4 + t_2$	p_1	17
$t_4 + t_3$	p_3	12

MAFC(j) by summing up the values of FEC(j) and FITCC(j) are:

Tasks	Processors	FEC(j)	FITCC(j)	MAFC(j)
$t_4 + t_1$	p_2	6	13	19
$t_4 + t_2$	p_1	6	17	23
$t_4 + t_3$	p_3	12	12	24

Final Results

Processors	PEC(j)	ITCC(j)	TOC(j)	MSR(j)	T_{RP}(j)
P_1	3	3	6	0 .333	0.333
P_2	6	13	19	0.167	0.334
P_3	3	14	17	0.333	0 .333

The $MAFC(2)_{41}$ is minimum i.e. 19. Therefore task t_4 is fused with task t_1 executing on processor p_2. We compute PEC(j), ITCC(j), TOC(j), MSR(j) and T_{RP}(j) given in table final results.

The maximum of TOC(j) is 19 i.e. the total busy cost of the system is 19 which corresponds to processor p_2. The model deals with the problem of optimal task allocation and load balancing in DCS. The load balancing mechanism is introduced in the algorithm by fusing the unallocated tasks on the basis of minimum of the average impact of EC and ITCC. It can also be perceived from the example presented here that wherever the algorithm of better complexity is encountered [21] present technique gets an upper hand by producing better optimal results with slight enhancement in the cost due to minor in complexity factor. The worst case run time complexity of the algorithm suggested by [21] is $O(n^2+m^2+ m^2n+m+n)$ and the run complexity of the algorithm presented in this paper is $O(m^2n+2mn)$. Figure1 shows the run time complexity of present methods and [21] considering different cases of tasks and processors. The utility of our algorithm will ultimately depend on how it can be incorporated into the designing and managing activities of DCS.

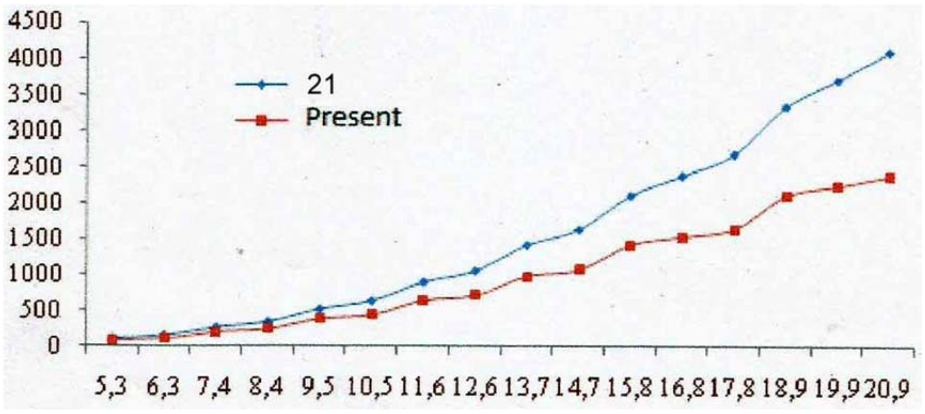

Figure : Run time complexity compression

VI. REFERENCES

[1] Ajith, Tom P. and Murthy, C. Siva Ram, "An Improved Algorithm For Module Allocation in Distributed Computing Systems", Journal of Parallel and Distributed Computing Systems, 42, pp. 82-90 (1997).

[2] Mostaph Zbakh and Mohamed Dafir El Kettani, "A Task Allocation Algorithm For Distributed Systems", Journal of Theoretical and Applied Information Technology, Vol. 33 No. 1 (2011).

[3] Sharma Abhilasha and Gupta Surabhi**,** "An optimal approach for the tasks allocation based on the fusion of EC and ITCC in the Distributed Computing System", American Journal of Research in Science, Technology, Engineering & Mathematics. 5(2), pp. 184 - 189, Dec. (2013) -Feb. (2014).

[4] SHATZ, S/M., and WANG, J.P., "Introduction to distributed software engineering", Computer, 20 (10), pp. 23-31 (1987).

[5] Baca, D.F., "Allocation Tasks to Processor in a Distributed System", IEEE Transactions on Software Engineering, Vol. SE-**15**, 1427-1436, (1989).

[6] Bokhari, S.H., " Dual Processor Scheduling with Dynamic Re-Assignment", IEEE Transactions on Software Engineering, vol. SE-5, 341-349 (1979).

[7] Kumar Avanish, Sharma Abhilasha, and Dhagat, V. B., “Maximal Link Module Algorithm For Task Allocation In Distributed Computing System”, Proceedings of the 4th National Conference; INDIACom- 2010.

[8] Casavent, T.L. and Kuhl, J.G., “A Taxonomy of Scheduling in General Purpose Distributed Computing System”, IEEE Trans. On Software Engineering, Vol.14 pp.141-154, (1988).

[9] Chu, W.W., “Optimal File Allocation in a Multiple Computing System”, IEEE Trans. On Computer, Vol.C-18 pp.885-889, (1969).

[10] Yadav, P.K., Singh, M.P. and Sharma, K., “Task Allocation Model for Reliability and Cost Optimization in Distributed Computing System”, International Journal of modeling, simulation and scientific computations, vol-2, pp. 1-19 (2011).

[11] Attiya Gamal and Haman Yskandar, “Task allocation for maximizing reliability of distributed systems: A simulated annealing approach”, Journal of Parallel and Distributed Computing, Volume 66, Issue 10, 1259-1266 (October 2006).

[12] Kumar, V., Yadav, P.K. and Bhatia, K, “Optimal Task Allocation in Distributed Computing Systems Owing to Inter Tasks Communication Effects”, Proc. of the 33rd Annual convention of system society of India, held at New Delhi, India 369- 378 (1998).

[13] Yadav, P.K., Singh, M.P. and Kumar Harendra, “Scheduling Algorithm: Tasks Scheduling Algorithm for Multiple Processors with Dynamic Re-assignment”, Journal of Computer Systems, Networks, and Communications, Vol. 2008, pp. 1-9, (2008).

[14] Ma, P.-Y.R., LEE, E.Y.S. and Tsuchiya, M., “A Task Allocation Model for Distributed Computing Systems", IEEE Trans. on Computers, Vol.C-31, No. 1, pp. 41-47, (Jan.1982).

[15] Yadav, P. K., Bhatia K. and Sagar Gulati, “Reliability Evaluation of Distributed System Based on Failure Data Analysis”, International Journal of Computer Engineering, Vol. 2(1), pp. 113-118, (2010).

[16] Sinclair, J.B., “Efficient Computation of Optimal Assignments for Distributed Tasks”, J. Parallel Dist. Comput., Vol. **4**, pp. 342-362, (1987).

[17] Sagar, G. and Sarje, A.K., “Task Allocation Model for Distributed System”, Int. J. System Science, Vol.(no.)-22(9), pp.1671-1678, (1991).

[18] Kumar, V., Singh, M.P. and Yadav, P.K., “A Fast Algorithm for Allocation Task in Distributed Processing System”, Proceedings of the 30th Annual Convention of Computer Society of India, Hyderabad, pp.347-358, (1995).

[19] Kumar, V., Singh, M.P., and Yadav, P.K., “An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System”, Proceedings of the Nineteenth National System Conference (NSC-95), PSC College of Technology, Coimbatore, Organized by System Society of India, New Delhi, 82-87, (1995).

[20] Kumar, V., Singh, M.P. and Yadav, P.K., “An Efficient Algorithm for Multi-Processor Scheduling with Dynamic Re-Assignment”, Proceedings of the Sixth National Seminar on Theoretical Computer Science, Banasthali Vidya Pith, pp. 105-118, (1996).

[21] Elsadek, A.A. and Wells, B.E., “A Heuristic model for task allocation in heterogeneous distributed computing systems”, The International Journal of Computers and Their Applications, Vol. 6, no. 1, pp. 0-35, (March-1999).

[22] Beniwal Payal and Garg Atul, “A comparative study of static and dynamic Load Balancing Algorithms”, International Journal of Advance Research in Computer Science and Management Studies, Vol. 2, Issue 12, pg. 386-392, (December 2014).

[23] Yadav, P.K., Singh, J. and Singh, M.P., “An efficient method for task scheduling in computer communication network”, International Journal of Intelligent Information Processing Vol. 3(1) pp. 81-89, (2009).

[24] Yadav, P.K., Singh, M.P., Kumar Avanish and Agrawal Babita, “An efficient task scheduling model in Distributed Computing Systems Using ANN”, International Journal of Computer Engineering Vol. 1(2) pp. 57-62, (2009).

[25] Singh, M.P., Kumar Avanish, Yadav, P.K. and Krishna Garima, “Study of Load Distribution in fully connected client server Network using feedback neural architecture”, Jour. of Engi. and Technology Research, Vol. 2(4) pp. 58-72, (2010).

[26] Kaushal Urmani and Kumar Avanish, “Improving the Performance of DRTS by Optimal Allocation of Multiple Tasks Under Dynamic Load Sharing Scheme”, Int. J. of Scientific and Engineering Research, Vol. (4), Issue 7, pp. 1316- 1321 (July 2013).

[27] Kumar Avanish, Yadav P.K., and Sharma Abhilasha, “Analysis of Load Distribution in Distributed Computing systems Through Systematic Allocation Task”, International Journal of Mathematics, Computer Science and Information Technology, Vol. 3(1), pp.101-114, (2010).

[28] Yadav, P. K., Kumar, Avanish and Singh, M. P., “An Algorithm for Solving the Unbalanced Assignment Problems”, International Journal of Mathematical Sciences, Vol. 12(2), pp. 447-461 2004.

AUTHOR’S PROFILE

Sarvesh K Dubey, M.Phil.: He is a Doctoral student in Department of Physical Sciences of Mahatma Gandhi Chitrakoot Gramodaya Vishwavidyalay at Chitrakoot (MP), INDIA. His research interest include task allocation and HPC with cluster, grid, cloud and parallel distributed computing. He received his Master of Philosophy degree in Operation Research from Bundelkhand University at Jhansi (UP), INDIA. Contact him at dubey.sarvesh25@gmail.com.

Virendra Upadhyay, Ph.D.: He is an Associate Professor , Head & Former Dean of Faculty of Science in Mahatma Gandhi Chitrakoot Gramodaya Vishwavidyalay at Chitrakoot (MP), INDIA. His research area of interest include Fluid Mechanics, Convention flow, applied Fluid Mechanics etc. Contact him at drvirendra.upadhyay@gmail.com.

Avanish Kumar, Ph.D.: He is an Associate Professor & Former Head in Department of Mathematical Sciences & Computer Applications of Bundelkhand University at Jhansi (UP), INDIA. He conducts applied research in the various areas of Operation Research, Distributed Computing, Software Engineering and Enterprise integration. Contact him at dravanishkumar@yahoo.com.

. . . .