# K-Means Clustering based Task Allocation Model for Distributed Real-Time System

Urmani Kaushal
Mody University of Science and Technology
Lakshmangarh, Sikar, Rajasthan, India

Avanish Kumar
Bundelkhand University
Jhansi, U.P.
India

Narendra Kumar
Mody University of Science and Technology
Lakshmangarh, Sikar, Rajasthan, India

## ABSTRACT
The distributed real-time system [DRTS] is the great platform for parallel application. Multiple tasks will be formed of the parallel application, which are to be allocated over the nodes available in DRTS. Numbers of tasks are much more than available nodes in the system. The tasks should be grouped or clustered in a very efficient manner and allocated over the nodes of the system efficiently for the minimization of overall system cost and maximization of system performance.

Task allocation is NP-hard problem. A model based on k-mean clustering has been proposed in this paper. In the suggested model, the limitation of memory and the number of tasks allowed over the processor has been considered. MATLAB 7.11.0 has been used to simulate the proposed model.

## Keywords
Distributed Real Time System, k-mean cluster, NP-hard, Parallel Application, Task Allocation.

## 1. INTRODUCTION
Task allocation problem in DRTS deals with finding appropriate assignment of tasks to available nodes in the system so that the system performance can be maximized. A large body of literature exists for providing the solution and models of task allocation. In the literature [***1-10***] several models for allocation, like graph theoretical, integer programming, fuzzy logic based, genetic algorithm based, simulated annealing based and heuristic methods have been proposed [***10,11***].

Graph theoretical models proposed in [***1,9***] makes the use of a graph to represent tasks. By performing a min-cut algorithm on the graph it reduces the inter task communication cost.

In integer programming method [***2***] implicit enumeration algorithm is used. Here constraints can be enforced easily.

In Genetic Algorithm (GA), a unique encoding scheme with Partially Matched Crossover (PMX) is being used. In this approach a population of strings is being created and Reproduction, Crossover, and Mutation operations are performed over it to get the desired allocation [***12***].The simulated annealing (SA) is based on exponential cooling schedule of Newtonian cooling process. The number of iterations should be chosen at each step of experimentation [***13***].

In fuzzy logic based model proposed in [***14***] two round of operation have been performed for the task allocation. In this model each node is associated with to a learning agent. These agents in turn evaluate themselves by using fuzzy approximate reasoning.

Task assignment model proposed in [***2,4,15***] has used the clustering to reduce inter task communication cost first and then device the mechanism to assign the task clusters to appropriate processor or node. While clustering the execution cost of tasks has not been taken into consideration.

By efficient allocation of load available in the form of clusters, performance can be improved of distributed system. For this purpose load sharing policies can be adopted while allocation.

Load sharing policies can be either static or dynamic. Static load sharing policies do not require system state information in making load distribution decisions [***7***]. Dynamic policies make their load distribution decisions based on the current system state. Dynamic load sharing policies provide significant performance improvements compared to static policies [***5***]. This paper considers dynamic load sharing in heterogeneous distributed systems.Most of the models ignore programs' needs, real load conditions and users' activities [***16***]. The task allocation is a NP-complete problem. The task allocation algorithms proposed in [***4,5,7,8,15,17,18***] are using static load sharing policy.

The objective of the model proposed in this paper is to maximize the overall performance of distributed system by efficient allocation mechanism. A new heuristic method for task allocation has been proposed and deployed here.

The inter task communication should be avoided while making the decision of task allocation to the available processors. The decision of task allocation over any node depends on the number of tasks already in execution on it. At the same time, available memory capacity of the processing node should be greater than the required memory by the task. The above constraints should be considered while designing the strategies for task allocation.

We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace the content with your own material.

## 2. PROBLEM FORMULATION
The parallel application should complete its execution over a DRTS in lesser cost than over a standalone system. The parallel applications running over the DRTS is having *m* number of tasks which are to be allocated on *n* number of nodes of the distributed system for execution. This allocation of task should be done in such a manner that inter task

communication can be reduced. Execution constraints of the tasks must be fulfilled. A processor can hold a limited number of tasks and memory is also the limit. So these constraints had been taken into consideration in the proposed model, which provide an optimal solution for the assignment of the set of *"m"* tasks of programs over the set of *"n"* processors / nodes (*where, m > n*) DRTS.

Performance enhancement of the distributed system is the objective of this problem and it will be achieved by appropriate allocations of tasks over the system.

## 2.1 Assumptions

In the proposed task allocation model following assumptions [***1,2,4,18***] have been made:

**1** The processors available in the system is not having any specific interconnection structure and they are heterogeneous in nature.

**2** The parallel application is the collection of *m* independent tasks, which are to be allocated over a set of *n* available nodes processors with different attributes.

**3**The tasks will not be relocated once allocated on processors without completing its execution.

**4T**he inter task communication cost (*ITCC*) between the tasks in the same cluster will be zero.

**5**The clusters will be formed in accordance with processors / nodes available in the system.

**6** For clustering data points will be the collection of vectors i.e. *ECM (,)*.

**7** Number of tasks *m* will be greater than the number of processors *n* every time while making the allocation decision (*m>>n*) as in real life situation.

## 2.2 Proposed Mathematical Model for Task Allocation

In this section, a task allocation model has been developed to find the optimal system cost so that system performance could be enhanced. Effective allocation of parallel applications' tasks may increase the performance of the distributed system.

Execution cost, inter task communication cost, memory required by each task and the task accommodation capacity of processor should be known for the allocation of tasks. Obtaining all such information of tasks and processors / nodes is beyond the scope of this paper. The clustering has been done at two levels in the model, one at initial stage by using *k-mean* algorithm and the second at the time of allocation. *TMSV* (Task& Memory Status Vector) data structure and *TMSV*'s Collection proposed in [***5***] are being used to solve the problem in this paper.

### *2.2.1 Execution Cost*

The task allocation given as: *X: T→P, X (i)* = l. For the task allocation *X*, the execution cost $ec_{ij}$ represents the execution of task $t_i$ on processor $P_j$[***4***]. For the task allocation *X*, execution cost of $n^{th}$ processor can be calculated as:

$$\mathrm{EC(X)} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \sum_{\mathrm{j=1}}^{\mathrm{m}} ec_{ij}\, x_{ij} \qquad (1)$$

$$where, x_{ij} = \begin{cases} 1\ if\ i^{th}\ task\ is\ assigned\ to\ \ j^{th}\ processor \\ 0, otherwise \end{cases}$$

### *2.2.2 Communication Cost (CC)*

The cost incurred because of data exchange between $t_i$ and $t_j$ allocated over different processors represented by $cc_{ij}$. $cc_{ij}$ = 0 is the constraint if tasks $t_i$ and $t_j$ are residing on the same node or processor[***5***].

### *2.2.3 Task Clustering*

The total distance of each data point of a cluster from the cluster mean i.e. the compactness of cluster ($Z_{ki}$) can be evaluated as

$$\sum_{x_i IC_k} \left\| X_i - \overline{X}_k \right\|^2 = \sum_{i=1}^{m} Z_{ki} \left\| X_i - \overline{X}_k \right\|^2 \qquad (2)$$

$Z_{ki}$ is an indicator variable designating the fittingness of the $i^{th}$ data point $X_i$ to be a part of the $k^{th}$ cluster [***5***].

Where,

Cluster Mean calculated as:

$$\overline{\mathrm{X}}_{\mathrm{k}} = \frac{1}{\mathrm{m_k}} \sum_{x_i IC_k} X_i$$

Total number of points allocated to cluster *k* is

$$m_{\mathrm{k}} = \sum_{i=1}^{m} \mathrm{Z}_{ki}$$

Overall goodness of clustering using indicator variables defined as:

$$\varepsilon_{\mathrm{k}} = \sum_{i=1}^{m} \sum_{k=1}^{k} Z_{ki} \left\| X_i - \overline{X}_k \right\|^2 \qquad (3)$$

$\overline{X}_{\mathrm{k}}$ should be calculated to minimize the value of $\varepsilon_{\mathrm{k}}$.

# 3. PROPOSED TASK ALLOCATION TECHNIQUE AND ALGORITHM

## 3.1 Technique

In the addressed problem, There are two sets, $P = \{P_1, P_2, P_3, \ldots\ldots P_n\}$and $T = \{t_1, t_2, t_3, \ldots\ldots t_m\}$of *'n'* processors and '*m*' tasks respectively. Execution Cost Matrix *ECM(,)*of order *m x n* contains the cost of each task over every processor and Inter Task Communication Cost Matrix *ITCCM (,)* of order *m x m* stores the communication cost of tasks.

By minimizing the total system cost, the performance could be enhanced. There are *m* tasks to be processed over *n* processors and the number of tasks is more than the number of processors (*m>n*), so tasks should be clustered into *k* clusters. *k* clusters are to be allocated on *n* processors. For clustering, *k-mean* clustering algorithm will be used. Each task with its processing cost over every processor is forming a vector. Therefore, *m* vectors of task will be framed intended to be placed in *k* clusters.

Find *k* initial points for each cluster represented by task vector. These points represent centroids. Assign each task vector to the cluster that has the closest centroid. When all task vectors have been assigned, recalculate the positions of *k* centroids. Repeat the work of assignment and recalculation of centroid's positions until the centroids no longer move. This produces a separation of the task vectors into clusters from which the metric to be minimized is calculated by using equation (2) [***5***].

Modify the *ECM (,)* according the *k* clusters by adding the processing time of those tasks that occurs in the same cluster. Modify the *ITCCM(,)* by putting the communication zero amongst those tasks that are in the same cluster.

In the process of assigning *k* clusters to *n* processors, next level of clustering will be done on the basis of $ec_{il}$ (execution cost) constraint. If there is any change in the clusters the *ECM (,)* and *ITCCM (,)* should be recalculated accordingly. Once the final assignments are in hand, the optimal cost of assignment is to be computed using eq. (1). The objective function to calculate total system cost is as follows:

$$\text{Total Cost} = EC + CC \quad (4)$$

## 3.2 Proposed Algorithm

The algorithm consists of following steps:

**Step-1:** Start

**Step-2:** Read the number of processors in *n*, number of tasks in *m*, number of clusters in *k*.

**Step-3:** Read the *ECM(,)* of the task of order *n x m*, *ITCCM (,)*of order m *x m*,*TMSV* Collection, Memory Requirement of each task

**Step-4**: Apply *k-mean* clustering algorithm on *ECM (,)*

**Step-5:** Store Cluster Information *CIM*

**Step-6:** For all clusters

Ifthe cluster can't be assigned because of *TMSV* Collection or Memory Requirement over any processor than

mark it as restricted assignment

**Step-7:** Check if any of cluster can't be assign because of *TMSV* Collection or memory requirement over all processors then

Goto Step 7

**Step-8:** Add the processing cost of tasks in each cluster and update *ECM (,)*

**Step-9:** Update the *ITCCM (,)*on the basis of *CIM*

**Step-10:** Divide Modified *ECM (,)* in n column matrix

**Step-11:** Sort each column matrix

**Step-12:** For each column matrix *i*=1 to *n*

For *j*=1 to *m*

If $t_j$ is not assigned

Assign $t_j$ on $P_i$ processor

Else

If cost on pre assignment > cost of current assignment

Assign $t_j$ on $P_i$ processor

Else

Skip current assignment

End if

End if

End for

End for

**Step-13:** Modify *TMSV* Collection according to the assignment made.

**Step-14:** Calculate total EC and ITCC

**Step-15:** Calculate the Optimal Cost by eq. (4)

**Step-16:** End

# 4. IMPLEMENTATION

To illustrate the proposed algorithm, which is implemented in MATLAB is using the following data set. It assumed that Execution Cost Matrix, *ITCC* Matrix and *TMSV's* Collection table are given for each task in units of time. Given a set of seven tasks $\{T_1,T_2,T_3,T_4,T_5,T_6,T_7\}$ and a set of three processors $\{P_1, P_2, P_3\}$. Execution Cost Matrix has been given in Table 1.

**Table 1: Execution Cost Matrix**

| | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| $T_1$ | 15 | 25 | 15 |
| $T_2$ | 10 | 10 | 30 |
| $T_3$ | 40 | 25 | 20 |
| $T_4$ | 5 | 20 | 5 |
| $T_5$ | 10 | 15 | 10 |
| $T_6$ | 10 | 5 | 5 |
| $T_7$ | 15 | 20 | 5 |

Inter-task communication detail has been provided in Table 2. Memory requirement of all tasks is in the Table 3. The detail of memory capacity and the number of tasks allowed by the each node is given in Table 4. Clustering information is provided in Table 5. The Final allocation of all tasks is provided in Table 6. Table 7 shows the final optimized system cost.

**Table 2: Inter Task Communication Cost Matrix**

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ |
|---|---|---|---|---|---|---|---|
| $T_1$ | 0 | 1 | 5 | 4 | 2 | 7 | 8 |
| $T_2$ | | 0 | 2 | 0 | 9 | 5 | 1 |
| $T_3$ | | | 0 | 4 | 2 | 2 | 5 |
| $T_4$ | | | | 0 | 3 | 1 | 6 |
| $T_5$ | | | | | 0 | 9 | 5 |
| $T_6$ | | | | | | 0 | 1 |
| $T_7$ | | | | | | | 0 |

**Table 3: Memory Requirement of Tasks in Units**

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ |
|---|---|---|---|---|---|---|
| 5 | 6 | 3 | 2 | 1 | 2 | 3 |

## 5. CONCLUSION

The model proposed in this paper is based on effective two level clustering and straightforward and efficient algorithm to obtain optimal system cost. The task allocation is done under load sharing scheme by allocating the task dynamically. *ITCC* had been reduced by effective two levels clustering in this model. The overall complexity of the proposed algorithm is $O(m^2)$.

**Table 4: TMSV's Collection**

| Processor | No. of Tasks (Maximum) | Memory Capacity (Maximum) | Tasks Assigned | Available Task Capacity | Memory Available |
|---|---|---|---|---|---|
| $P_1$ | 9 | 40 | 0 | 8 | 40 |
| $P_2$ | 6 | 30 | 0 | 4 | 30 |
| $P_3$ | 4 | 10 | 0 | 4 | 10 |

**Table 5: Cluster Information**

| Cluster | Clustered Tasks | Restricted Assignment | Final Assignment |
|---|---|---|---|
| C - 1 | $T_3$ | - | $P_3$ |
| C - 2 | $T_1, T_4, T_5, T_6, T_7$ | $P_2$ | $P_1$ |
| C - 3 | $T_2$ | $P_2$ | $P_3$ |

**Table 6: Status of TMSV's Collection after Allocation**

| Processor | No. of Tasks (Maximum) | Memory Capacity (Maximum) | Tasks Assigned | Available Tasks Capacity | Memory Available |
|---|---|---|---|---|---|
| $P_1$ | 9 | 40 | C - 2 | 4 | 27 |
| $P_2$ | 6 | 30 | - | 6 | 30 |
| $P_3$ | 4 | 10 | C – 1, C - 3 | 2 | 1 |

**Table 7: Optimal System Cost**

| Processors | Cluster | Processor Load | Optimal System Cost | | |
|---|---|---|---|---|---|
| | | | EC | ITCC | EC + ITCC |
| $P_1$ | C - 2 | 10 | 70 | 8 | 78 |
| $P_2$ | - | - | | | |
| $P_3$ | C – 1, C - 3 | 60 | | | |

In the illustrated problem, the number of the tasks is much more than the number of processors of the DRTS. To measure the performance of the model a problem has been solved. In this problem many constraint have been taken into consideration. Before allocation of tasks on the processor, node's memory capacity has been checked and the number of tasks allowed by the processor has also been checked. Restricted assignments given by the initial phase of the algorithm has also been considered at the time of the allocation.

## 6. ACKNOWLEDGMENTS

We gratefully acknowledge support from Dean and faculty members of Department of Computer Science, FASC, Mody University Science and Technology, Lakshmangarh, Sikar and Department of Mathematical Sciences & Computer Applications, Bundelkhand University, Jhansi for the same.

## 7. REFERENCES

[1] B. Ucara, C. Aykanata, K. Kayaa and M. Ikincib, "Task Assignment in heterogeneous computing system," J. Parallel Distrib. Comput., vol. 66, pp. 32-46, 2006.

[2] P.-Y. RICHARD MA, E. Y. S. LEE and M. TSUCHIYA, "A Task Allocation Model for Distributed Computing Systems," IEEE Transactions on Computers, vol. C 31, no. 1, pp. 41-47, January 1982.

[3] A. Elsadek and B. E. Wells, "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.

[4] K. Govil and A. Kumar, "A Modified and Efficient Algorithm for Static Task Assignment in Distributed Processing Environment," International Journal of Computer Applications, vol. 23, no. 8, pp. 1-5, June 2011.

[5] U. Kaushal and A. Kumar, "Improving the Performance of DRTS by Optimal Allocation of Multiple Tasks under Dynamic Load Sharing Scheme," International Journal of Scientific & Engineering Research, vol. 4, no. 7, pp. 1316-1321, July 2013.

[6] U. Kaushal and A. Kumar, "Modified Clustered Approach for Performance Escalation of Distributed Real-Time System," in ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol II, S. C. Satapathy, P. Avadhani, S. K. Udgata and S. Kakshminarayana, Eds., Vishakapatnam, Springer International Publishing, 2014, pp. 9-16.

[7] U. Kaushal and A. Kumar, "Performance Intensification of DRTS under Static Load Sharing Scheme," International Journal of Computer Applications, vol. 71, no. 16, pp. 55-59, June 2013.

[8] U. Kaushal. A. Kumar and N. Kumar, "Algorithm for Performance Improvement of DRTS Under Static Load Sharing Scheme," IUP Journal of Information Technology, vol. 9, no. 3, pp. 43-52, September 2013.

[9] H. S. Stone, "Multiprocessor scheduling with the aid of network flow," IEEE Trans. Software Eng., vol. SE 3, pp. 85-93, January 1977.

[10] R. Mohan and N. P. Gopalan, "A Modified Parallel Heuristic Graph Matching Approach for Solving Task Assignment Problem in Distributed Processor System," I.J. Information Technology and Computer Science, vol. 5, no. 10, pp. 78-84, 2013.

[11] Karimi, F. Zarafshan and A. b. Jantan, "A New Fuzzy Approach for Dynamic Load Balancing Algorithm," International Journal of Computer Science and Information Security, vol. 6, no. 1, pp. 1-5, 2009.

[12] Sahu and R. Tapadar, "Solving the Assignment problem using Genetic Algorithm and Simulated Annealing," IAENG International Journal of Applied Mathematics, vol. 36, no. 1, pp. 762-765, February 2006.

[13] G. Attiya and Y. Hamam, "Task allocation for maximizing reliability of distributed systems:A simulated annealing approach," J. Parallel Distrib. Comput., vol. 66, no. 10, p. 1259 – 1266, 2006.

[14] Z. Khan, R. Singh and R. Alam, "Tasks Allocation Using Fuzzy Inference in Parallel And Distributed System," Journal of Information and Operations Management, vol. 3, no. 2, pp. 322-326, 2012.

[15] K. Govil, "A Smart Algorithm for Dynamic Task Allocation for Distributed Processing Environment," International Journal of Computer Applications, vol. 28, no. 2, pp. 13-19, 2011.

[16] G. A. Geist and V. S. &Sunderam, "Concurrency: Practice and Experience," Network Based Concurrent Computing on the PVM System, vol. 4, no. 4, pp. 293-311, 1992.

[17] Folliot and P. Sens, "Load Sharing and Fault Tolerance Manager," High Performance Cluster Computing Architectures, p. 841, 2008.

[18] P. Yadav, M. Singh and K. Sharma, "An Optimal Task Allocation Model for System Cost Analysis in Hetrogeneous Distributed Computing Systems: A Heuristic Approach," International Journal of Computer Applications, vol. 28, no. 4, pp. 30-37, August 2011.