INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET)

ISSN 0976 – 6367(Print)
ISSN 0976 – 6375(Online)
Volume 4, Issue 4, July-August (2013), pp. 31-44
© IAEME: www.iaeme.com/ijcet.asp
Journal Impact Factor (2013): 6.1302 (Calculated by GISI)
www.jifactor.com

IJCET

©IAEME

# OVERLAPPED CLUSTERING APPROACH FOR MAXIMIZING THE SERVICE RELIABILITY OF HETEROGENEOUS DISTRIBUTED COMPUTING SYSTEMS

**Vinod Kumar Yadav**
ITD, SRMCEM, Lucknow
U.P, India

**Indrajeet Gupta**
ITD, SRMCEM, Lucknow
U.P, India

**Brijesh Pandey**
CSED, GITM, Lucknow
U.P, India

**Sandeep Kumar Yadav**
SCTD, GBU, Greater Noida
U.P, India

## ABSTRACT

For distributed computing system (DCS) where server nodes can fail permanently with nonzero probability, the reliability of the system can be defined as the probability that the system run the entire tasks successfully assign on it before all the nodes fail. In heterogeneous distributed system where various nodes of the system have different characteristics, reliability of the system is highly dependent on the tasks allocation strategies. So, this paper presents a rigorous framework for efficient tasks allocation in heterogeneous distributed environment, with the goal of maximizing the system reliability. Reliability of the system is characterized in the presence of communication uncertainties and topological changes due to node's failure. Node failure has adverse effects on the system reliability. Thus, one possible way to improve reliability is to make the communication among the tasks as local as possible. For this, an overlapped clustering approach has been used. Further, we calculate the reliability of each node of the DCS to determine the actual capabilities of each node. Here, our purpose is to assign the more costly task to more reliable node of the DCS. Then we utilize the load balancing policies for handling the node's failure effect as well as maximizing the service reliability of the DCS. A numeric example is presented to illustrate the importance of incorporating overlapping cluster and load balancing on the reliability study.

**Keywords:** Overlapping cluster, load balancing, reliability, distributed computing, queuing theory, failure propagation, redundancy.

## 1. INTRODUCTION

The introduction of distributed computing paradigms has brought an unprecedented computational power for executing high performance parallel applications, alternative to the very expensive massively parallel machines. A heterogeneous distributed computing system is the

collection of geographically dispersed heterogeneous computing resources fully connected to each other. There is no shared memory in these types of systems. Every system has own local memory. The systems of the DCS communicate with each other via message passing over the network. These messages may take arbitrary delay to deliver from source to destination. Unlike parallel computing environment, various nodes of the distributed computing systems offer heterogeneous computing capabilities. In addition, the communication network typically suffers to both low bandwidth and a significant latency in the information changes. So, in order to exploit the processing capability of DCS, a parallel application is divided into independent executable unit of sub applications that are called tasks and executed concurrently on different nodes in the DCS. In literature, such allocation of tasks on the node's of DCS is referred as tasks assignment.

For a DCS, several studies have been devoted to the problem of tasks assignment. These assignments are concerned with the performance measures such as minimizing the application response time, turnaround time [5, 6, 8, 11, 12, 19] or minimizing the total sum of execution and communication cost (time) [9, 11, 12, 13, 14, 16]. In heterogeneous distributed computing system assignment of tasks among the nodes of the DCS highly affects to its performance. Here, we have used overlapped clustering approach for such assignment. The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters [23]. In overlapping cluster we check out for each task, which nodes of DCS will satisfies to its requirements. In such a way we created the clusters. Some of the requirements for various tasks are common so the clusters overlapped to each other.

Here, our purpose for such assignment is to assign the tasks only on those nodes of the DCS where its requirements get satisfied. It will reduce the network bandwidth problem as well as load balancing problem. This assignment of tasks among the various nodes of the DCS will play an important role. Because for a given context, some of these assignment configurations are obviously more effective than others in terms of some quality of services such as performance, dependability, reliability and so on. The main concern of this is to maximize the reliability of DCS.

Several works have been done for decade to improving the reliability of the DCS, but it is at same. Reliability of a distributed system can be defined as the degree of tolerance against errors and components failure in the system [22]. In distributed computing environment network failure is the most prominent problem. It highly affects to the system reliability and the software application may not provide its expected functionality. For minimizing to this problem we have made the communication as local as possible. Here, it is notable that the tasks that are assigned on the same node of DCS can communicate without affecting to the network status. So many techniques have been given to improve the reliability of DCS. Redundancy is the traditional techniques for improving the reliability of such systems [2, 7, 8, 13, 17, 19], where multiple storage devices and processors are allowed to maintain the multiple copies of critical information. In such a case when one of the processor fails, the computation can be successfully completed at other processor and if one of the storage device fails, the information can be used from other devices. But, redundancy is an expensive approach. In many cases, the system redundancy is not available or infeasible. In that cases, the assignment of tasks have done carefully onto the appropriate processors in the system. We take into account the failure rate of both the processors and communication links. One idea for it is, the tasks need longer execution time should be assigned to more reliable processor.

In this work we have considered that server nodes provide random tasks service time and can fail permanently at any random time. So, we have used the load balancing strategies to handle the failure situations. Load balancing can provide the safe and efficient techniques to the server nodes to process the user requests. Load balancing algorithms generally can be classified as either static or dynamic [3, 4, 10]. Static algorithms use only given node information to make load balancing decisions [4]. But dynamic algorithms dynamically reflect the load of the server nodes [10]. These types of algorithms are very difficult to implement. Here, we are very selective to apply the load

balancing algorithms. We have applied only in the cases of overloaded nodes, idle nodes, and failure cases. Actually, we have followed to this selective nature for load balancing to reduce the network congestion. Load balancing have been done at cluster level.

In this paper a cluster based load balancing approach has been used to maximize the service reliability of peer-to-peer distributed environment. The communication links of the system are the peer-to-peer communication medium with well defined characteristics and behavior [14, 20]. For each node and link of the system we calculate its reliability and stored with them that used to determine the actual capability of each node and link. This information gives us prior knowledge about the various nodes and links of the system that help us in load balancing.

This paper is organized as follows: Section-2 describe the related works. Section-3 defines the problem and its description. In section-4, we have described the Reliability Model. Section-5 discussed the creation of overlapped clusters for appropriate assignment of tasks. Load balancing strategies have been given in section-6. Section-7 includes the brief description of performance of the works and conclusions are presented in section-8.

## 2. RELATED WORKS

To improve the reliability of DCS is a challenging task for researchers. Although so much attention and work have been done in this field but the problem is at same place as it was a decade ago. A distributed computing system becomes complicated in volatile environment in which nodes are prone to fail permanently. To handle to this situation some researchers have proposed the redundancy technique that is an expensive approach. Some of the researchers have proposed the tasks assignment techniques that is not so easy in such situations.

Jorge E. Pezoa [2, 3] proposed an online decentralized solution for maximizing the reliability of DCS. He has presented a framework to characterize the service reliability of DCS in the presence of communication uncertainties and topological changes due to node deletions. His analysis was based upon the effective exchange of load state information among the nodes that is used to estimate whether nodes are imbalanced or not. With the help of this information he calculated both the amount of tasks need to be reallocated to other nodes and the set of nodes receiving the load. Further, he has characterized the dynamics of the service reliability as a function of the balancing instant.

In [24] Bin Yan has presented an efficient and stable cluster system for load balancing of servers. It was based on the improved load balancing algorithm. An improved algorithm that was proposed in this paper combined the two algorithms and improved the efficiency and stability of the cluster system. This cluster system added a redundant node to collect the load balancing parameters of every child node. The control node obtained the children nodes load by exchanging the load balancing information with the redundant node, and then decided which node would be responded to a new user request.

C. C. Hsies [21] proposed the redundancy techniques for improving the reliability of DCS. A distributed computing system is redundant if it processes multiple copies of software and/or hardware. In [17] software redundancy multiple data/files are used among processors and algorithms are proposed to seek the minimal data/files replication while retaining the system reliability. In [2, 7, 8, 13] hardware redundancy multiple processors and communication links are considered and some models are developed under different levels of redundancy. Moreover, redundancy is an expensive approach and many times, it is not available or infeasible.

In [14] Heydernoori maximized the reliability by deployment process. It is a sequence of related activities for installing an already developed application into its target environment, and brings it into an executing state. He had used the graph-based approach for deployment process. Here, his purpose was to make the communication among the processors as local as possible. The main aim of the presented work was to maximize the reliability defined as the probability of failure free software operation for a specified period of time in a specified environment.

## 3. PROBLEM STATEMENT

The problem addressed in this paper concerned with the assignment of tasks of a parallel application on the processors of a distributed computing system with the goal of maximizing the service reliability by handling the failure situation during the execution of tasks with the help of load balancing policies. Suppose that DCS composed of N nodes that can communicate to each other by exchanging the messages. Each node of DCS has some computation facility and communication network has some limited communication capacity. With each node and communication link their failure rate is also associated. There is an appropriate mapping among the tasks and nodes of DCS (where the tasks requirements are getting satisfied). We also assumed that each node of DCS have their own queue-length that shows the number of tasks queued for execution at that node at a particular moment. The queue-length information messages are exchanged by the servers. This information is used by servers to estimate the queue-length of the remaining server in the DCS. The workload to be executed is composed of T individual tasks, such that:

$$\sum_{j=1}^{N} t_j = T$$

Here, $t_j$ shows the amount of tasks that is executing at node j.

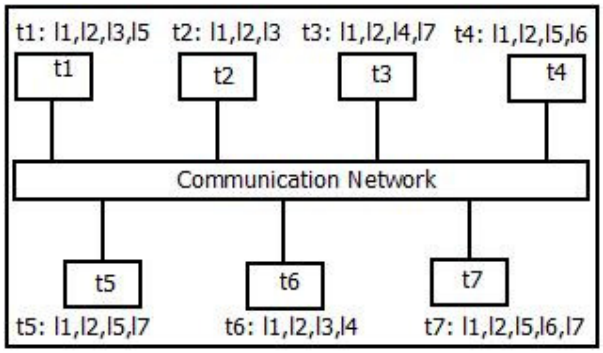



**Figure-1:** Shows the application tasks sets with its requirements

**Table-1:** Shows the requirements of various tasks as shown in fig-1

| Tasks | Requirements |
|---|---|
| $t_1$ | $l_1, l_2, l_3, l_5$ |
| $t_2$ | $l_1, l_2, l_3$ |
| $t_3$ | $l_1, l_2, l_4, l_7$ |
| $t_4$ | $l_1, l_2, l_5, l_6$ |
| $t_5$ | $l_1, l_2, l_5, l_7$ |
| $t_6$ | $l_1, l_2, l_3, l_4$ |
| $t_7$ | $l_1, l_2, l_5, l_6, l_7$ |

**Table-2:** Shows the available resources on nodes of DCS as shown in fig-2

| Nodes | Available Resources |
|---|---|
| $n_1$ | $l_1, l_2, l_3, l_4, l_5, l_6$ |
| $n_2$ | $l_1, l_2, l_3, l_4$ |
| n3 | $l_1, l_2, l_4, l_5, l_7$ |
| n4 | $l_1, l_2, l_5, l_6, l_7$ |
| $n_5$ | $l_1, l_2, l_3, l_4, l_5, l_7$ |

Briefly, we have considered a graph-based approach for tasks assignment. For a set of tasks T, comprising a parallel application (fig-1) have to be executed on a distributed computing system of N nodes (fig-2). Because we have considered heterogeneous DCS, so it's various nodes have different capacity and a failure rate is associated with each node of the system. And communication links have different characteristics such as bandwidth, synchronous, asynchronous, FIFO etc. On the other hand, tasks of the given application required certain capacitated nodes (such as memory,

processing speed etc.) and links (such as bandwidth, synchronous, asynchronous). Here, our purpose is to assign all the T tasks among the N nodes of DCS such that the requirements of tasks gets satisfied and overall reliability of the system could be maximized.

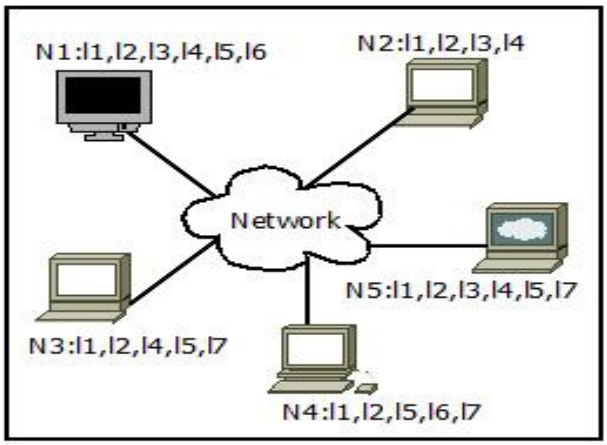



**Figure-2:** Shows the various nodes of DCS with its Available Resources

## 4. RELIABILITY MODEL

In order to maximize the service reliability of distributed computing system, it is necessary to take care of tasks assignment policies. Because we have considered heterogeneous distributed computing system, whose various nodes and communication links have different processing capabilities (hardware and software). On the other hand, various application tasks have different requirements. So, these tasks will perform their expected functionality only when their requirements are getting satisfied by the nodes of DCS on that tasks have been assigned. If the assignment of tasks has not done carefully, nodes of the DCS spend more time to transfer the tasks rather than performing the useful computations. In literature this problem is called thrashing. To handle to this problem, one possible solution is to calculate reliability of each node of the DCS before assigning the tasks. After that assignment is done by taking care of these reliability values as well as requirements of tasks.

### Assumptions

(a) The distributed computing system is heterogeneous in nature. It means the processors may have different processing capabilities (such as processing speed, memory size, failure rate etc.). And, communication links may have different characteristics (such as bandwidth, synchronous, asynchronous, FIFO, failure rates etc.).

(b) Each component (processors and communication links) of distributed computing system may be in one of the two states: operational or faulty. A component will perform their task if it is in operational state. If a component fails in idle time, it will not consider as critical failure.

(c) The failure rates of each component (hardware and software) are associated with them that are constant.

### Notations and Descriptions

The notations and descriptions specified below will be used in the rest of the paper:

T : The task, i.e. the set of modules to be executed.

m : |T|, the number of modules forming the task T.

$m_i$: $i^{th}$ module of task T.

P : The set of processors or nodes (N) in the distributed computing system.

n : |P|, the number of processors in distributed computing system.
$P_j$ : $j^{th}$ processor in distributed computing system.
L : Set of communication links.
$l_{ab}$: Communication link in L connecting the processors $P_a$ and $P_b$.
$P_R$ : Reliability of processor P.
$L_R$: Reliability of communication link L.
$\psi_j$ : Failure rate of processor $P_j$.
$\omega_{ab}$ : Failure rate of communication link $l_{ab}$.
A : A binary matrix (T x N) corresponding to task assignment.
$C_{ij}$ : Accumulative execution cost of task i on processor $P_j$.
$A_{ij}$ : An assignment of $i^{th}$ task on $j^{th}$ processor.
$C_{rsab}$ : Transferring cost of data between task r and s by using communication link $l_{ab}$.
$S_R$: Reliability of distributed computing system.

Reliability of a distributed computing system $S_R$ can be defined as the product of the reliability of its processors ($P_R$) as well as the reliability of its communication links ($L_R$) [13], that means:

$$S_R = P_R * L_R$$

Each component of the distributed computing system may exist in one of the two states: operational or faulty. For successful execution of tasks each component must be operational during the time of execution. Here, it is notable that if a component fails in its idle time then it will not consider as critical failure. For reliable task assignment it is also necessary that the cost of assignment should be minimum. Cost of a task defined in terms of its execution cost and communication cost.

*Processor Reliability ($P_R$):* A processor reliability ($P_R$) is defined as the probability that the processor P is operational during the time interval t for execution of tasks that are assigned to it. If $\psi_j$ is considered as the failure rate of processor $P_j$ during time interval t, then the reliability of processor $P_j$ can be expressed as $\exp(-\psi_j t)$ [1, 12, 13, 15]. Here, time t represents the time required to execute all the tasks assigned to processor $P_j$. Under the assignment A, if $C_{ij}$ be the accumulative execution cost (AEC) of a module running on a processor is the total execution cost incurred for this module during the execution. It means AEC is the product of the number of times this module executed during the process and the average cost unit for each execution on that processor for task i running on it [13, 18]. Then the reliability of processor P is defined as:

$$P_R = \exp(-\psi_j \sum C_{ij} A_{ij}) \quad \text{.....(1)}$$

The above expression shows the total time (cost) taken for executing the tasks assigned on $j^{th}$ processor of distributed computing system.

*Communication link/path reliability ($L_R$):* A communication link reliability ($L_R$) is the probability that the link ab (connecting the two adjacent processor $P_a$ and $P_b$) being operational for communicating data between the terminal processors a and b (intermodule communication between two modules is the product of the number of times they communicates and the average number of words exchanged in each communication) [13, 15] of the link. For a time interval t, if the failure rate of communication link $l_{ab}$ is ($\omega_{ab}$), then the reliability of communication link $l_{ab}$ is $\exp(-\omega_{ab} t)$ [1, 12, 13]. The reliability of communication link for an assignment A and cost of transferring data between two tasks r and s which are assigned to different processors defined as:

$$L_R = \exp\left(-\omega_{ab} \sum_{r} \sum_{r \neq s} C_{rsab} A_{ra} A_{sb}\right) \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

The above expression gives the required time for communication between processors a and b by using link $l_{ab}$.

*System reliability ($S_R$):* Reliability of a distributed computing system may be defined as the probability that the system can run the entire application successfully [15, 21]. In other words, the reliability of the distributed computing system is defined as the product of the reliabilities of its components. It means, the system reliability is the product of the reliabilities of processors as well as the reliabilities of the communication links. Hence, the system reliability may be defined as:

$$S_R = [\prod_a P_R] [\prod_r \prod_{r \neq s} L_R] = \exp(-R) \ldots\ldots\ldots\ldots\ldots (3)$$

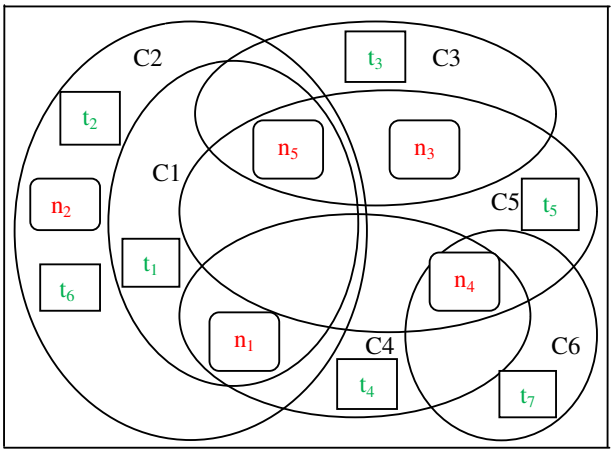
Where

$$R = \sum_a \sum_r \psi_j C_{ij} A_{ij} + \sum_a \sum_{b \neq a} \sum_r \sum_{s \neq r} \omega_{ab} C_{rsab} A_{ra} A_{sb}$$

In above equation first part determines the unreliability due to execution of tasks on the processors of the distributed computing system and second part determines the unreliability due to the inter-processor communication.

## 5. CREATION OF OVERLAPPING CLUSTERS

Load balancing is an expensive approach for maximizing the service reliability of distributed computing system. Because, in dynamic load balancing where a node become unbalance broadcast message to other nodes for averaging the load. So, there are so many messages transmitted over the networks for estimating the load that create network congestion. For example, suppose there are N nodes in DCS that send messages to each other. Then first node send messages to (N-1) nodes, second node send message to (N-1) nodes and so on. In such a way all the N nodes exchange the message to each other. It means (N-1)*N or ($N^2$-N) messages transmitted over the network that cause the network congestion and failure. So by reducing the number of messages exchanging by nodes over network for balancing the load, reliability can be improved. So, one possible way to improve the reliability of DCS is to carefully assignment of tasks among the nodes of DCS.




(**Figure-3:** Shows overlapped clusters for tasks allocation)

**Table-3:** Relatinships among the clusters, nodes and tasks

| Cluster Name | Nodes Joining the Cluster | Tasks Assigned |
|---|---|---|
| C1 | $n_1$, $n_5$ | $t_1$ |
| C2 | $n_1$, $n_2$, $n_5$ | $t_2$, $t_6$ |
| C3 | $n_3$, $n_5$ | $t_3$ |
| C4 | $n_1$, $n_4$ | $t_4$ |
| C5 | $n_3$, $n_4$, $n_5$ | $t_5$ |
| C6 | $n_4$ | $t_7$ |

Because DCS is heterogeneous in nature, so its nodes have different processing capabilities and communication links have different processing capacities. On the other hands, various application tasks have different requirements. So, here our purpose is to determine the set of nodes in DCS for each task where its requirements are get satisfied. The set of nodes for each task where their requirements are getting satisfied are grouped together and called cluster. Clustering is the process of grouping a set of physical or abstract objects into classes of similar objects [23]. Because, some of the nodes of DCS are able to satisfy the requirements of tasks whose requirements are different so they form overlapping clusters. Here, it is notable that the total number of clusters does not exceed to the number of tasks. It means:

$$\sum_{K=1}^{r} C_k \leq T$$

Here, $C_k$ shows the of number overlapped clusters and T shows the number of application tasks. All the steps of clustering process is briefly shown in following algorithm:

```
1.  Ck {N}← 0
2.    k ← 1
3.   for i ← 1 to T
4.     do
5.       for j ← 1 to N
6.         do
7.           if {Ti(Req)}≤ {Nj(Av)}
8.             then
9.                 Ck ← Nj
10.                Ck ← Ti
11.          else
12.              Task can not be executed
13.          end else
14.          end if
15.         j ← j +1
16.      end for
17.  if {Ck-1(N)} ≠ {Ck(N)}
18.    then
19.        k ←k + 1
20.   else
21.        k ← k
22.  end else
23.  end if
24.      i ← i + 1
25.  end for
```

First two steps shows that initially, for k=1 there are none of the nodes in the cluster $C_k$. From step 3 to 7, we have compared the requirements of task $T_i$ with the available resources of each node $N_j$ of DCS. Specially in step 7, when the requirements of tasks $T_i$ is satisfied by the resources available at node $N_j$, then node $N_j$ will includes in cluster $C_k$, and task $T_i$ will assigned to cluster $C_k$ respectively in steps 9 and 10. If the condition of step 7 fails, then task $T_i$ will not be executed at node $N_j$ in step 12. Step 17 shows that if the nodes are not similar in cluster $C_k$ and $C_{k-1}$, then value of k will be incremented by 1 in step 19, otherwise no change in the value of k in step 21. In step 24, value of 'i' will be incremented by 1.

## 6. INCORPORATING THE FAILURE

In order to provide more reliable system, we have used the concepts of load balancing. For it, we have divided the various nodes of DCS among overlapped clusters and assigned the tasks on the basis of available resources on them. Although, still we have assign the tasks very carefully onto the nodes of the DCS, but there are possibility that the tasks could not completely executed on that node (processor) before failure. It means some of the nodes are not as reliable as we required. In such cases nodes may fails before completely executing the tasks assigned on it. Some other cases can be arises such as overloading which means some of the nodes of the clusters are heavily loaded whereas some of them have very low load. To handle to all these situations we have applied the load balancing strategies among the cluster nodes.

The reliability discussed here can be achieved only if the DCS provides tasks redundancy. Redundancy means the backup system that is attached with each cluster. This strategy of task redundancy is a hybrid version of method described in [2, 3]. In [2] redundancy has been used as distributed forms. It means with each system of DCS there was a backup system attached. And in [15] redundancy has been used as centralized form. Here, it is notable that the backup systems does not service to any tasks. In case of node failure backup system must perform the following tasks: 1) Broadcast a failure notice (FN) packet to alert the nodes about the change in the number of functioning nodes; 2) Reallocate all the unfinished tasks among those nodes within the cluster perceived to be functioning and 3) Handle the reception of tasks that were in transit to the $j^{th}$ node before its failure and reallocate the received tasks among the functioning nodes [2, 13]. On the basis of above discussion we can conclude that each backup system of DCS contains the number of functioning nodes, number of tasks queued at each node and the amount of tasks in transit over the network.

Suppose $Q_j(t)$ denotes the queue length of the $j^{th}$ node of DCS at time t. For node $j \neq k$, we use the binary variable $q_{jk}(t)$ that will indicate ("1") if the $j^{th}$ node is informed otherwise ("0") if uninformed about the queue length of the $k^{th}$ node. It means variable $q_{jk}(t)$ describes the status that the queue length information packet broadcasted by the $j^{th}$ node has been received or not by the $k^{th}$ node of the cluster in DCS. Here, we can represent the $Q_j(t)$ and $q_{jk}(t)$, means number of tasks queued in its own queue and its $jk^{th}$ off-diagonal element contains the $q_{jk}(t)$ variable that indicate node $j^{th}$ have informed about the queue length of $k^{th}$ node or not. We can form the Q(t) matrix as the system queue status. It can be represented as:

$$Q(t) = \begin{vmatrix} m_1 & 0 & 1\ldots..1 \\ 1 & m_2 & 0\ldots..1 \\ 1 & 1 & m_3\ldots0 \\ 0 & 1 & 1\ldots m_n \end{vmatrix}$$

For above configuration at time $t = t_0$, first node has $m_1$ tasks in its queue and is uninformed about the queue length of node 2, informed about queue length of node 3 and so on and finally

informed about the queue length of node n. Similarly, node n has $m_n$ tasks in its queue and informed about the queue length of nodes 2 and 3 and uninformed about the queue length of node 1.

Similarly, we will represent the functional states of DCS. Suppose $f_j(t)$ be a binary variable representing the working ("1") or failure ("0") state of the $j^{th}$ node at time t. For $j \neq k$, we define $f_{jk}(t) = 1$ to indicate that the $k^{th}$ node is functioning at time t as perceived by the $j^{th}$ node and $f_{jk}(t) = 0$ to indicate that the $k^{th}$ node is faulty at time t as perceived by the $j^{th}$ node. Now, we arranged all these variables in an n-by-n matrix and described the functional and non-functional state of the nodes. Suppose f(t) denote the system function state at time t, then f(t) can be represented as:

$$f(t) = \begin{vmatrix} 1 & 0 & 0\ldots.1 \\ 1 & 0 & 1\ldots.0 \\ 1 & 0 & 0\ldots.1 \\ 1 & 0 & 0\ldots.1 \end{vmatrix}$$

At time $t = t_0$ corresponds to the configuration for which the first node is functioning and has perceived functioning to $n^{th}$ node and faulty to $2^{nd}$ and $3^{rd}$ nodes. Node 2 is faulty and has perceived functioning to nodes $1^{st}$ and $3^{rd}$ and faulty to node $n^{th}$. Similarly $n^{th}$ node is functioning and perceived functioning to $1^{st}$ node and faulty to $2^{nd}$ and $3^{rd}$ nodes. Here it is noted that as in the case of the queue length information, the random transfer time of failure notice (FN) packets introduces uncertainty or the functioning state that a node perceives about the other nodes in the DCS. For some node's at the same time perceiving as functioning while other perceiving as faulty. It means the message taking an arbitrary delay to deliver from source to destination so that some nodes are not exactly updated at the same time.

Further for reallocation of the tasks, we have followed the load balancing approach. And reallocation has performed only in the cases either when the system nodes or communication links fails or when the node's of DCS become overloaded so that it is unable to perform its expected tasks, or when a system node becomes idle while some other have more than two processes. Here, we have used the concept of load balancing that handles the above problems more efficiently. Because, we have assigned the tasks among the nodes of DCS by analyzing the reliability of the nodes as well as communication links that reduces the failure cases as well as overloading problems. So, we did not apply the general load balancing policies for handling the such problems.

## 7. RESULTS

We have implemented the theoretical achievements discussed in this paper to experimentally validate the things. The test bed architecture consists of the computing nodes, communication links and backup nodes. The set of computing nodes comprises heterogeneous in term of their processing speeds (depends on processors such as Celeron, Pentium-II and Pentium-IV etc.), available resources at each node and failure rate of communication links and nodes that are associated with each. Backup nodes are the same as the working nodes but their roles are different. When a failure occurs, a computing node is switched from working state to failed state. And its loads are needed to be transferred to other working nodes because a node is in failed state cannot process the tasks. The simulated program contains three major parts: first part reads the number of tasks 'T' and the processors 'N'as input. Now, it generate the equivalent parameters: execution time of tasks, memory required, processing capability, and communication capacity required. For distributed system the program algorithm generates the system parameters: available memory, communication capacity and processing load capability.

In step two we have assigned the set of tasks among the nodes of DCS in such a way so that the requirements of tasks are getting satisfied by the nodes of the DCS. Because, we have determined the requirements of each tasks as well as the resources available at each node of DCS. So, we

grouped to those nodes that have approximately common resources and formed clusters. This activity formed the set of overlapped clusters as shown in figure-3. Now, step-3 handle the failure situations by following the load balancing policies. Here, we have considered that requirements of some tasks are not completely satisfied by an individual node. So, some of the tasks partially executed on a node and their remaining work will execute on another nodes. So, we need to transfer them on another node. The failure rate of communication links and the failure rate of processors are given in the ranges (0.00015 – 0.00030) and (0.00005 – 0.00010) respectively.

**Table-4:** Shows the simulation results

| Case | (T, N) | Zϕ | Zc | Δ Z |
|---|---|---|---|---|
| 1 | (2, 6) | 3.833333 | 3.833333 | 7.826087 |
| 2 | (2, 6) | 5.750000 | 5.000000 | |
| 1 | (4, 6) | 8.833333 | 8.833333 | 8.551068 |
| 2 | (4, 6) | 3.555556 | 3.555556 | |
| 3 | (4, 6) | 6.000000 | 5.000000 | |
| 4 | (4, 6) | 5.000000 | 4.000000 | |
| 1 | (6, 6) | 8.833333 | 8.833333 | 13.548387 |
| 2 | (6, 6) | 2.000000 | 2.000000 | |
| 3 | (6, 6) | 2.000000 | 1.000000 | |
| 4 | (6, 6) | 6.000000 | 5.000000 | |
| 5 | (6, 6) | 2.000000 | 2.000000 | |
| 6 | (6, 6) | 5.000000 | 3.500000 | |
| 1 | (8, 6) | 8.833333 | 8.833333 | 20.063694 |
| 2 | (8, 6) | 2.000000 | 2.000000 | |
| 3 | (8, 6) | 3.555556 | 3.555556 | |
| 4 | (8, 6) | 5.000000 | 1.000000 | |
| 5 | (8, 6) | 6.000000 | 5.000000 | |
| 6 | (8, 6) | 2.000000 | 2.000000 | |
| 7 | (8, 6) | 5.000000 | 4.000000 | |
| 8 | (8, 6) | 2.500000 | 2.500000 | |
| 1 | (10, 6) | 8.833333 | 8.833333 | 23.675029 |
| 2 | (10, 6) | 2.000000 | 2.000000 | |
| 3 | (10, 6) | 3.833333 | 3.642857 | |
| 4 | (10, 6) | 2.000000 | 1.000000 | |
| 5 | (10, 6) | 1.407407 | 1.407407 | |
| 6 | (10, 6) | 6.000000 | 5.000000 | |
| 7 | (10, 6) | 2.000000 | 1.000000 | |
| 8 | (10, 6) | 3.000000 | 3.000000 | |
| 9 | (10, 6) | 5.000000 | 3.500000 | |
| 10 | (10, 6) | 1.857143 | 1.857143 | |

Table-4 summarized the simulation results for the discussed problems. It presents the results for the case of six processors with fully connected topology. The first column of table represents the problem sets, where 'T' is the number of tasks and 'N' is the number of processors. The second column represent the values of the cost function (in terms of execution cost of each task) for random allocation 'Zϕ'. It means in this case we did not have care of reliability as well as the requirements of

tasks and resources available on the processors of DCS. Zc considered the values with whole constraints and listed in the third column. The last column represented the average deviation ΔZ in percentage between the random allocation and fully constraints valued. That is:

$$\Delta Z = \frac{(Z\phi - Zc)}{Z\phi} \times 100$$

Where $Z\phi$ and Zc are the values without constraints and with constraints respectively. The result shows that the computation cost with constraints are decreases with the problem size increases. Here, it is also concluded that at constant failure rates of the system components, the system reliability is highly depending upon the tasks distribution. It means that, the reliability of a distributed system does not depends only on the reliability of its components (hardware and software) but also on the allocation of the tasks on the nodes of the DCS. Hence, a parallel application can be executed with high reliability if the various tasks of the application are assigned carefully to the appropriate nodes (by taking care of requirements and availabilities of the resources of tasks and nodes respectively).

## 8. CONCLUSIONS

In heterogeneous distributed computing system where server node can fail permanently, allocation of tasks among the nodes play an important role to improve the reliability of the system. Here, we have taken an approach to analyze the service reliability of DCS in the presence of communication and node uncertainty. We have allocated the tasks among the nodes of DCS very carefully by taking care of requirements of tasks and resources available on the nodes of DCS. Because allocation of the tasks among those nodes of DCS where their requirements are not satisfying, just increases the communication costs. So, we have selected the nodes where their requirements are gets satisfied. And then calculate reliability of such nodes and then assign the tasks on appropriate node that reduces the communication and execution cost as well as failure chances. Our experimental result shows not only an improvement in the service reliability, but also the remarkable accuracy in our predictions. In terms of scalability, our approach improves the reliability linearly with the number of tasks increases. For a small test bed of 10 tasks system reliability improved more than 23 percent.

In future we will improve in our study on this topic by taking into consideration of the following aspects. Firstly, we handle the failure situation without redundancy. In this paper we have considered redundant systems for handling the failure situation. Because we have determined the reliability of each node as well as communication links, so on the basis of these reliability values we will predict that the system will further proceed the tasks queued on it or not. In the case if a node is being unable to proceed the tasks before being fail, it transfer its load to functioning nodes of DCS. It means a node will work as backup for own. Secondly, we have considered the nodes in only two states either working (“1”) or faulty (“0”). So enhance the working states of nodes on the basis of percentage. It means how much computational capability is left in the nodes.

## REFERENCES

[1] Gamal Attiya and Yskandar Hamam, “Task allocation for maximizing reliability of distributed systems: A simulated annealing approach”, J. Parallel Distributed Computing 66 (2006) Elsevier, 1259-1266.

[2] Jorge E. Pezoa, sagar Dhakal, and Majeed M. Hayat, “Maximizing Service Reliability in Distributed Computing Systems with Random Node failures: Theory and Implementation”, IEEE trans.on parallel and distributed system, vol. 21, no. 10,oct.2010.

[3] Jorge E. Pezoa, sagar Dhakal, and Majeed M. Hayat, "Decentralized Load Balancing for Improving Reliability in Heterogeneous Distributed Systems", International conference on parallel processing workshops, 2009.
[4] S. Dhakal, "Load Balancing in Communication Constrained Distributed Systems: A Probabilistic Approach", Ph.D. dissertation, University of New Mexico, 2006.
[5] Y. S. Dai et al., "Performance and reliability of Tree-Structured Grid Services Considering Data Dependence and failure Correlation", IEEE T. Computers, v. 56, pp. 925-936, 2007.
[6] Yang, B., Hu, H., & Jia, L., "A Study of Uncertaity in Software Cost and its Impact on Optimal Software Release Time", IEEE trans. on Software Engineering, 34(6), 813-825, 2008.
[7] Yang, B., Hu, H., Guo, S., & Huang, H. Z., "Syste Cost Oriented Task Allocation and Hardware redundancy Strategy for Distributed Computing System", In proceedings of the first international conference on maintenance engineering (ICME2006) (pp. 872-878). Chengdu, china, Oct. 2006.
[8] Bo Yang. Huajun Hu, Suchang Guo, "Cost-Oriented task Allocation and Hardware Redundancy Policies in Heterogeneous Distributed Computing Systems Considering Software Reliability",Computers & Industrial Engineering 56 (2009) 1687-1696.
[9] Jorge E. Pezoa, Majeed M. Hayat, Zhuoyao Wang and Sagar Dhakal, "Optimal Task Reallocation in Heterogeneous Distributed Computing Systems with Age-Dependent Delay Statistics", 39th International Conference on Parallel Processing, IEEE, 2010.
[10] S. Dhakal, M. M. Hayat, J. E. Pezoa, C. Yang, and D. A. Barder, "Dynamic Load Balancing in Distributed Systems in the Presence of delays: A Regeneration-Theory Approach", IEEE Trans. Parallel and Dist. Systems, vol. 18, pp. 485-497, 2007.
[11] Y. S. Dai and G. Levtin, "Optimal Resource Allocation for Maximizing Performance and Reliability in Tree-Structure Grid Services", IEEE Trans. Reliability, vol. 18, pp. 485-497, 2007.
[12] V. K. Yadav, S. R. Sahoo, D. K. Yadav, "Reliable Task Allocation in Distributed Systems", 11th international conference on Software Engineering Research and Practice (SERP-12), July 16-19, IEEE 2012.
[13] V. K. yadav, M. P. Yadav, D. K. Yadav, "Reliable Task Allocation in Heterogeneous Distributed Systems with Random Node Failure: Load Sharing Approach", Turing 100 International Conference on Computing Science (ICCS), IEEE, 2012.
[14] Abbas Heydarnoori, Farhad Mavaddat, "Reliable Deployment of Component-Based Application into Distributed Environments", Proceeding of the 3rd international conference on Information Technology New Generations (ITNG'06), IEEE, 2006.
[15] G. Attiya and Y. Hamam, "Reliability Oriented Task Allocationin Heterogeneous Distributed Computing Systems", Proc. Ninth Int'l Symp. Computers and Comm., pp. 68-73, 2004.
[16] R. S. Montero, R. M. Vozmediano, I. M. Llorente, "An Elasticity Model for High Throughput Computing Clusters", J. Parallel Distrib. Comput. 2010.
[17] A. Lisnianski and Y. Ding, "Redundancy Analysis for Repairable Multi-State Systems by using Combined Stochastic Processes Methods and Universal Generating Function Technique", Reliability Engineering and System Safety, vol.94, pp. 1788-1795, 2009.
[18] Y. Hamam and K. S. Hindi, "Assignment of Program Modules to Processors: A simulating Annealing Approach", J. of operational research 122. pp. 509-513, 2000.
[19] G. Attiya and Y. hamam, "Optimal Allocation of Tasks onto Networked Heterogeneous Computers Using Minimax Criterion", Proc. of the international network optimization conferencr (INOC' 03), pp. 25-30, Evry Paris, france 2003.
[20] Arbab, F. Reo, "A Channel-Based Coordination Model for Component Composition", Mathematical structure in computer science, 14.3 (June, 2004), 329-366.
[21] C. C. Hsies, "Optimal Task Allocation and Hardware Redundancy Policies in Distributed Computing Systems", European J.Oper. Res 147 (2003), 430-447.

[22] Pradeep K. Sinha, Distributed Operating Systems Concepts and Design, PHI Learning Private Limited, New Delhi – 110001, 2010.

[23] J. Han and M. Kamber, "Data Mining Concepts and techniques", Morgom Kaufmann Publishers, 2nd edition, March 2006.

[24] Y.Bin, Q. Liu, B. Cheng Yang Hu, W. Zheng, "An Efficient and Stable Cluster System Based on Improved Load Balancing Algorithm", 3rd international conference on computer science and information technology (ICCSIT), July 9-11, IEEE, 2010.

[25] Preeti Gupta, Parveen Kumar and Anil Kumar Solanki, "A Comparative Analysis of Minimum-Process Coordinated Checkpointing Algorithms for Mobile Distributed Systems", International Journal of Computer Engineering & Technology (IJCET), Volume 1, Issue 1, 2010, pp. 46 - 56, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.

[26] D.Asir, Shamila Ebenezer and Daniel.D, "Adaptive Load Balancing Techniques in Global Scale Grid Environment", International Journal of Computer Engineering & Technology (IJCET), Volume 1, Issue 2, 2010, pp. 85 - 96, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.

[27] R. Lakshman Naik, D. Ramesh and B. Manjula, "Instances Selection using Advance Data Mining Techniques", International Journal of Computer Engineering & Technology (IJCET), Volume 3, Issue 2, 2012, pp. 47 - 53, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.