Abstract—This paper presents a rigorous framework of efficient task allocation in heterogeneous distributed environment where server nodes can fail permanently. The system performance can be improved by increasing the probability of serving queued tasks in the distributed computing system (DCS) before all the node fails. For a large set of tasks that is being allocated into a distributed environment, several allocation methods are possible. These allocations can have significant impact on quality of services such as reliability, performance etc. In distributed environment reliability is highly dependent on its network and failures of network have adverse impact on the system performance. So, one possible way for improving reliability is to make the communication among the tasks as local as possible. Firstly, it divides the whole workload into small and independent units, called tasks and determines expectant hosts from environment. It then applying a two phase hybrid algorithm: Simulated Annealing and Branch-and-Bound (SABB). The Simulated Annealing algorithm finds a near optimal allocation from expectant hosts and then find an optimal allocation by applying the branch-and-bound (BB) techniques by considering the Simulated Annealing algorithm as initial solution. This algorithm overcomes the less improved quality obtained by heuristics as well as the computational cost of the exact algorithms.

Key Words: Reliability, Queuing Theory, Distributed Computing, Communication Links, Renewal Theory.

I. INTRODUCTION

A distributed computing system is a collection of heterogeneous processors interconnected by a communication network. Each processor has its own local memory and other peripherals, and the communication between any two processors of the system takes place by message passing over the communication network. Such systems provide very powerful platform for executing high performance parallel applications, alternative to the very expensive massively parallel machines. But the performance of the system is highly dependent on the tasks allocation onto the available machine. Because different application requires various hardware and software, So these application components will provide their expected functionality only when their requirements will be satisfied. Such type of problem in distributed computing system is referred as task allocation problem.

Typically the communication networks interconnecting the servers suffer the problem of low bandwidth. In order to enhance the processing capabilities of distributed computing system, workloads are partitioned into small independent units called tasks. To allocate these tasks among the processors is the main concern while maintaining the system reliability. For complex application where tasks can be allocated to different hosts in distributed environment, several tasks allocation method are available. Some of them are more effective than others for some given context in terms of quality of services such as communication cost, network congestion, dependability, performance etc.

In distributed environments network failure is the most potential problem that can lead to disastrous effects on the systems reliability and the software application may not provide its expected functionality. For minimizing to this risk one way is to make the communication among the tasks as local as possible. In this manner the tasks that are allocated on the same host of the DCS can communicate without any respect to the networks status.

In this paper reliability can be achieved by carefully assigning the tasks onto the processors of the DCS by taking into account to failure rate of both the communication links as well as the processors. Here the idea is to assigning tasks with longer execution time to more reliable processors and communication links. This paper also solves the problem of finding the optimal solution for reliable task allocation problem. It first determines the expectant hosts and then develops a mathematical model based on a cost function(execution time, inter-processor communication etc) and then apply a two phase hybrid algorithm: first is Simulated Annealing (SA) [1,3] and second is Branch-and-Bound(BB)[2,4,14,15]. The Simulated Annealing algorithms finds a near optimal solution and then find an optimal solution by applying Branch-and-Bound algorithm by considering the results of Simulated Annealing algorithm as initial solution.

This paper considers a graph - based approach for task allocation. For maximizing the reliability of the application, peer-to-peer distributed environment has taken into consideration. A communication link is a peer-to-peer communication medium with well defined characteristics and behavior [5]. In peer-to-peer architecture two or more computers can directly communicate with each other without requiring any intermediate devices [6,13,16]. In our consideration the tasks are allocated
with respect to the various parameters of communication links that different hosts can supports in distributed environment.

Rest of the paper is organized as follow: Section II formally defines the problem statement. Section III determines the hosts over which tasks can be assigned. Section IV presents a reliable task allocation model. Section V describes the algorithm used for task allocation. Section VI shows the experimental results. Finally Section VII concludes and outline future works.

II. PROBLEM STATEMENT

In a distributed computing system which consists of a set of N heterogeneous computers communicating over a fully connected network and M independent tasks have to be processed by the system. Each computer of the DCS has some capabilities and communication links have some capacities and a probability of failure is associated with each component of the system. We also associate a vector with each task which represents its execution cost at the different computers in the system. The purpose is to allocate M tasks onto the processors of the DCS in such a way that the requirements of tasks gets satisfied and the reliability of the system is maximized.

III. DETERMINATION OF EXPECTANT HOSTS

For determining expectant hosts following steps are considered: Firstly, we allocated the application components secondly, the tasks are allocated in distributed environment and at the end expectant hosts for application components are determined.

A. Allocation of application components

This step specifies the set of application components or tasks that has to be allocated onto the distributed environment for execution. These tasks are connected by a set of communication links that has different characteristics. These components can be compared with nodes of the graph and edges as communication links. These communication links (channels) can be of several types like synchronous, asynchronous, FIFO, etc.

B. Application components graph

Let \( n_i \) represents the various components of the application and \( l_j \) represents the various communication links. Then the component graph \( C_g = (V_{cg}, E_{cg}) \) is defined as a graph where \( V_{cg} = \{n_1, n_2, n_3, ..., n_N\} \) and \( E_{cg} = \{l_1, l_2, l_3, ..., l_M\} \).

C. Distributed environment

In distributed environment the tasks are allocated independently on the basis of their requirements.
Figure 2 shows the heterogeneous DCS consists of five hosts (H1, H2, H3, H4, H5). These hosts are connected by different set of links (11-16).

D. Expectant hosts

For a given component graph $C_g = (V_{cg}, E_{cg})$, host $H_j$ is an expectant host for the allocation of component $N_i$, only if $l_{Ni} \subseteq l_{H_j}$, where $l_{Ni}$ is the communication links required by component $N_i$ in component graph and $l_{H_j}$ is the communication links supported by host $H_j$ in distributed environment.

<table>
<thead>
<tr>
<th>Component name</th>
<th>Expectant host</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>H1,H4,H5</td>
</tr>
<tr>
<td>n2</td>
<td>H2,H5</td>
</tr>
<tr>
<td>n3</td>
<td>H1,H2</td>
</tr>
<tr>
<td>n4</td>
<td>H1,H4</td>
</tr>
<tr>
<td>n5</td>
<td>H4,H5</td>
</tr>
<tr>
<td>n6</td>
<td>H2,H4</td>
</tr>
<tr>
<td>n7</td>
<td>H3,H4,H5</td>
</tr>
</tbody>
</table>

Table-1: Shows the expectant hosts for components

Table-1 shows the candidate hosts for allocating the components of the application from fig-1 to the target distributed environment presented in fig-2. Here we can see that the application components n1 can be allocated either on host H1 or H4 or H5 and task n2 can be allocated on only H2 and H5, and task n3 can be allocated on host H1 and H2 and so on. Only these set of hosts can satisfy the requirements of the respective components.

IV. RELIABLE TASK ALLOCATION MODEL

Reliability of a distributed computing system can be seen as the reliability of its processors as well as the reliability of its communication links. Each component of the distributed system may exist in one of the two states: operational or faulty. For successful execution each processor must be operational during the time of execution of tasks. The communication links must be active during the communication between the end processors. For reliable task allocation it is also necessary that the cost of a task should be minimum. Cost of a task defined in terms of its execution cost and communication cost.

The system reliability for a mission is defined as the time interval during which the system to be active. During a process a task may execute more than once. The Accumulative Execution Time (AET) of a module running on a processor is total execution time incurred for this module running on that processor during the mission i.e. the product of the number of times this module executes during the process and the average time unit for each execution on that processor [7]. And the inter-module communication (IMC) between two modules is the product of the number of times they communicates and the average number of words exchanged in each communication [7]. A detailed discussion of AET and IMC can be found in [8].

A. Notations and Descriptions

The notations specified here used in rest of the paper:

- $R_p$: Reliability of processor $P$
- $T$: The set of tasks.
- $t_n$: $n^{th}$ module of the task set $T$.
- $P$: The set of processors.
- $P_i$: $i^{th}$ processor in $P$.
- $L$: Set of communication links.
- $l_{ij}$: $j^{th}$ communication link in $L$ connecting the processors $P_a$ and $P_b$.
- $\psi_i$: Failure rate of processor $P_i$.
- $X$: A binary matrix $(M \times N)$ corresponding to a task assignment.
- $C_{ni}$: Accumulative execution cost of task $n$ on processor $P_i$.
- $X_{ni}$: An assignment of $n^{th}$ task on $i^{th}$ processor.
- $\omega_{ab}$: Failure rate of communication link $(l_{ij})$ connecting the processors $P_a$ and $P_b$.
- $C_{mnab}$: The cost of transferring data between task $m$ and $n$ by using communication link $l_{ij}$ (connecting to two processors $P_a$ and $P_b$).
- $R_s$: Reliability of the distributed system i.e. the product of the reliability of the components of distributed computing system.
- $R_l$: Reliability of communication links.

B. Reliability of processors ($R_p$)

The reliability of a processor $p_i$ is the probability of being operational during the time interval ‘t’ till the execution of tasks are completed that are assigned to it. If a failure rate of processor $p_i$ is $\psi_i$, then the reliability of processor $p_i$ is $\exp(-\psi_i \cdot t)$ [7, 9, 10]. The reliability of processor $p$ for an assignment $X$, and accumulative execution cost $C$ for task $n$ running on it is defined as:

$$R_p = \exp(-\psi \sum_n C_{ni} X_{ni}) ....(4.1)$$

This expression gives the total time taken for executing the tasks assigned on $n^{th}$ processor.

C. Reliability of communication links ($R_l$)

The reliability of a communication link $l_{ij}$ (connecting two adjacent processor $P_a$ and $P_b$) is the probability of being operational during the time interval ‘t’ till the communication of task has completed between adjacent processors. If the failure rate of communication link is $\omega_{ab}$, then the reliability of communication link $l_{ij}$ is $\exp(-\omega_{ab} \cdot t)$ [7, 9, 10]. The reliability of communication link for an assignment $X$ and cost of transferring data between two tasks $m$ and $n$ which are assigned to different processors given as:

$$R_l = \exp(-\omega_{ab} \sum_{m \neq n} C_{mnab} X_{ma} X_{nb}) ....(4.2)$$

The summation gives the required time for communication between processors ‘a’ and ‘b’ by using link $l_{ij}$. 

So the reliability of the distributed computing system \( R_s \) is defined as the product of the reliability of components of the distributed computing system [12].

\[
R_s = \prod_a R_p \prod_m \prod_{n \neq m} R_{li} = e^{exp(-s)} \quad \ldots(4.3)
\]

Where

\[
S = \sum_a \sum_m \psi_i C_{ni} X_{mi} + \sum_a \sum_{b \neq a} \sum_m \sum_{n \neq m} \omega_{ab} C_{mnab} X_{ma} X_{nb} \quad \ldots(4.4)
\]

In eqn 4.4 first part of cost function determines the unreliability due to execution of tasks on the processors of the system and second part determines the unreliability due to the inter-processor communication.

The reliability of processors and communication links with respect to tasks are vectored and associated with each task, representing the execution cost of the tasks at the different processors in the distributed system. In figure-1 the set of vertices ‘V’ shows the set of tasks, edges ‘E’ shows the set of communication links. In figure-2 H1, H2, H3, H4 and H5 shows the processors of distributed computing system. Now the execution cost of each task at various nodes are calculated. Table-2 represents the communication costs of tasks at expectant hosts. The cost infinity represents the requirements of task is not satisfied at that processor.

Table-2 shows the execution cost of each task at the various processors

<table>
<thead>
<tr>
<th></th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>H4</th>
<th>H5</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>10</td>
<td>∞</td>
<td>∞</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>n2</td>
<td>∞</td>
<td>15</td>
<td>∞</td>
<td>∞</td>
<td>20</td>
</tr>
<tr>
<td>n3</td>
<td>7</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>n4</td>
<td>14</td>
<td>∞</td>
<td>∞</td>
<td>40</td>
<td>∞</td>
</tr>
<tr>
<td>n5</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>n6</td>
<td>∞</td>
<td>10</td>
<td>∞</td>
<td>5</td>
<td>∞</td>
</tr>
<tr>
<td>n7</td>
<td>∞</td>
<td>∞</td>
<td>11</td>
<td>18</td>
<td>35</td>
</tr>
</tbody>
</table>

V. ALGORITHM FOR TASK ALLOCATION

This section presents a two phase hybrid algorithm for task allocation. In the first phase Simulated Annealing(SA) algorithm and in second phase Branch-and-Bound(BB) algorithm has been used. The SA algorithm determines the near optimal allocation and BB algorithm determines the optimal allocation.

A. Simulated Annealing algorithm

Simulated Annealing Algorithm is a global optimization technique which attempts to find the lowest point in an energy landscape [3,11]. The SA method emulates the physical concepts of temperature and energy to represents and solves the optimization problem. It is often used when the search space is discrete. For certain problems, Simulated Annealing Algorithm may be more efficient than exhaustive enumeration provided that the goal is merely to find amount of time, rather than the best possible solution.

Simulated Annealing technique comes from metallurgy, involves heating and controlled cooling of a material to increase the size of its crystal and reduce their defects. Due to heat, atoms unstuck from their position (a local minimum of the internal energy) and moves randomly through state of higher energy. The slow cooling gives them more chances of finding configurations with lower internal energy than the initial one. Each step of the Simulated Annealing algorithm replaces the current solution by a random nearby solution, chosen with a probability that depends both on the difference between the corresponding function values and also a global parameter T (called temperature). That decreased gradually during the system equilibrium state through elementary transformations which will be accepted if they reduce the system energy. However as the temperature decreases, small energy increment may be accepted and the system eventually settle down to a low energy state.

The probability of accepting an uphill move (transition) from the current state \( S \) to a new state \( S_1 \) is specified by an acceptance probability function \( P(e,e_1,T) \), where T is the temperature. It depends on the energies \( e = E(S) \) and \( e_1 = E(S_1) \) of two states. Here it is noted that the probability function \( P \) must be nonzero when \( e_1 > e \). It means that the system may move to the new state even when it is worse (has a higher energy) than the current one. Here we are using the probability function \( \exp (-\delta/T) \) where \( \delta = e_1 - e \) (energy difference).

B. Pseudo code

The pseudo code given below presents the Simulated Annealing heuristic as described above [11]. It starts by randomly selecting an initial state \( S_0 \) and then calculates the energy (cost) \( E_{s0} \) for this state. After setting an initial temperature T, it generate a random chosen neighbor \( (S_1) \) from given set of states and calculate the corresponding energy \( E_{s1} \). If the energy of \( S_1 \) is less than the energy of \( S_0 \), then this solution is accepted as new solution. Otherwise, a probability function \( \exp (-\delta/T) \) is evaluated to ensure that the new solution may be accepted as a current solution. When a thermal equilibrium is reached at the current temperature T, the value of T is decreased by a cooling factor \( \alpha \), and inner repetition is increased by an increasing factor \( \beta \).

In this algorithm the neighborhoods defines procedure to move from a solution point to another solution point. Here it is very simple to determine neighboring solution. It can be obtained by choosing a random task n from the current allocation vector and assign it to randomly selected processor P from expectant hosts processor for the task n. Selection of initial temperature is very important parameter in Simulated Annealing algorithm because if the initial temperature is very high, then the execution time of algorithm becomes very
long and if it is very low then poor results are obtained. So
the initial temperature must be only hot enough to allow an
almost free exchange of neighboring solution.

The term cooling factor represents the rate at which the
temperature T is reduced. This is also an important factor for
the success of Annealing process [11]. The increasing factor
represents the rate at which the inner number of repetition
is increased with respect to reduction in temperature. It is
important to spend long time at lower temperature. In this
paper temperature is used as stopping condition.

C. Branch-and-Bound Algorithm

Branch-and-Bound Algorithm is a systematic method for
solving optimization problems. It is more suitable for solution
of those problems where the Greedy method and Dynamic
programming fails.

Branch-and-Bound Algorithm requires two steps. The first
one is a way of covering the feasible region by several smaller
feasible sub-regions (splitting into sub-regions), this is called
branching and second is bounding, which is a fast way of
finding upper or lower bounds for the optimal solution within
a feasible sub-region. In this paper we are using Simulating
Annealing algorithm for calculating the bound. The Branch-
and-Bound algorithm has all the elements of backtracking,
except simply stopping the entire search process any time a
solution is found. We continue processing until we get the best
solution. In addition, the algorithm has a scoring mechanism
to always choose the most promising configuration to explore
in each iteration. Because of this approach, Branch-and-
Bound is sometime called a best-first-search strategy. Starting
by considering the root problem (the original problem with
the complete feasible region), the bounding procedure are
applied to the root problem. If the bound is less than the
calculated value, prune that branch and choose another branch.

In case bound is greater than the calculated value choose
that one as the new bound. Otherwise, the feasible region is
divided into two or more regions. The algorithm is applied
recursively to the sub-problems until all the nodes have
been solved or pruned, or until some specified threshold is
met between the best solution found and the bounds on all
unsolved sub-problems

The efficiency of this method is highly depends on the
effectiveness of the bounding algorithm (Simulated Annealing
algorithm) used, bad choice could lead to repeated branching,
without any pruning, until the sub-regions become very small.

VI. EXPERIMENTAL RESULTS:

The proposed idea is tested for a large number of tasks
that are allocated onto a distributed system. The idea contains
two major parts. The first part determines the expectant hosts.
In second part it uses two phase hybrid algorithm (SABB)
for efficient task allocation. The hybrid method is coded with
object-oriented programming language over Unix platform.
The nodes are considered as heterogeneous on the basis of
reliability function.

In first part we reduce the number of nodes where a task can
be executed. For that we calculate the reliability of each node
of the DCS. On the basis of the requirements of the tasks we
set the reliability value, below that no task can be executed on
a particular node. It can be easily described by following table:

<table>
<thead>
<tr>
<th>Sl.No</th>
<th>Max Value(1000)</th>
<th>Final Cost</th>
<th>Node Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>75</td>
<td>106</td>
</tr>
<tr>
<td>2</td>
<td>700</td>
<td>75</td>
<td>142</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>75</td>
<td>142</td>
</tr>
<tr>
<td>4</td>
<td>300</td>
<td>75</td>
<td>163</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>75</td>
<td>163</td>
</tr>
</tbody>
</table>

Table-3 Shows the variation of node counts

In the above table we can see that when the individual cost
assigned to any worker node is equal or greater than the Max
bound then the number of nodes visited is minimum (shown in
$1^{st}$ row of the table). In the next of the cases the visiting node
is increases (that increase the cost in terms of execution time).
The above simulation results are shown in the following graphs:

Graph.1 shows the bound generated by the simulated annealing algorithm with the consideration of expectant hosts is lower than the bound generated without expectant hosts. $C_{Asa}$ shows the average cost in Simulated Annealing method.

Graph.2 shows the total number of node visited considering both the cases explained in Graph.1. Here the number of node visited is depends upon the bound generated by simulated annealing algorithm. Avn shows the average number of visited nodes.

VII. Conclusions and Future Work:

The task allocation problem in terms of reliability is discussed in this paper. The graph based mathematical approach has been taken to solve the problem. The number of visited nodes have been minimized during the allocation of task by suitably changing the bound value. The determination of expectant hosts has reduced the number of node where a particular task can be assigned. It enhanced the capability of Simulated Annealing Algorithm to determine the bound for Branch and Bound Algorithm. By dynamically updating the bound value we have assigned the tasks in more optimal way. Finally the total number of visited node has been reduced that minimized the over all assignment time, which makes the system more reliable.

The further enhancement would be the content storing strategy of nearest worker node in current node. This may lead to faster allocation of jobs in working nodes.

References