Reliability Aware Task Scheduling In Wireless Heterogeneous Systems

Sonali T. Bodkhe  
G.H. Raisoni Academy of Engineering & Technology, Nagpur, India.  
Ph.D Scholar at SGB Amravati University

Anjali R. Mahajan, PhD  
HOD, Department of Information Technology, Government Polytechnic, Nagpur, India.

ABSTRACT
A distributed operating system is one that looks to its user like an ordinary centralized operating system, but runs on multiple, independent nodes. A distributed operating system should:

- control resource allocation to allow their use in the most effective way;
- provide the user with a convenient virtual computer that serves as a high-level programming environment;
- hide the distribution of the resources;
- provide mechanisms for protecting system resources against access by unauthorized users.

The main purpose of this research is to explore and use the idle resources and to share the wireless distributed system fairly among the processes. Reliability is taken into account to achieve the goal of scheduling. The comparison study, based on both randomly generated graphs and the graphs of some real applications, shows that proposed scheduling algorithm can achieve reliable task scheduling and improves system reliability significantly. For experimental performance study, a real world application as well as synthetic workloads have been considered. It can be stated that shortening scheduling times improves performance of the system. Hence if scheduling algorithms are applied in parallel to reduce scheduling times, the performance of the heterogeneous clusters will be further enhanced.

General Terms
Distributed Computing, reliable task scheduling.

Keywords
Dynamic scheduling, heterogeneous systems, performance, evaluation, processor utilization, reliability, schedulability.

1. INTRODUCTION
In heterogeneous cluster computing systems, processor failures and network failures are a common occurrence and can have a negative effect on the applications running on such systems. If failures have to be taken into account effectively, then a reliable scheduling algorithm needs to be employed. However, most of the existing scheduling algorithms for heterogeneous systems consider only common features like scheduling length, and hence do not effectively satisfy the reliability requirements of any task. The problem becomes more prominent in wireless distributed systems where in spite of consideration of various factors, reliability cannot be assured. Therefore a WDS is proposed where some of the parameters of a static distributed system will be considered since scheduling itself is a NP complete problem.

Also reliable scheduling and increase in performance of system cannot be ensured at the same time. When it comes to WDS, task scheduling with heterogeneous systems further becomes complex since number of other parameters related to wireless networks have to be taken into account. To present a more realistic and precise scheduling result, scheduling and dispatching times have also been included into the proposed scheduling approach. Heterogeneous systems are dependable computing platforms. Many times heterogeneous distributed computing (HDC) systems can achieve higher performance with lesser cost than single super-systems due to resource optimization.

However, in these systems, processors and networks are not free from failure and thus may increase the criticality of the running applications. To deal with such failures, a reliable scheduling algorithm is required. Unfortunately, most scheduling algorithms for scheduling tasks in HDC systems do not sufficiently consider inter-dependent reliability requirements of tasks. Fig. 1 shows a hierarchical structure of task scheduling classifications in distributed systems.

With the use of directed acyclic graph (DAG) a reliability-based scheduling algorithm for tasks is proposed, which can achieve a better reliability for applications. A comparative study using both randomly generated graphs and the graphs of a few real applications, it can be shown that proposed scheduling algorithm gives a better performance over the existing scheduling algorithms in terms of make-span, scheduling length ratio, and reliability[1]. A hierarchical structure of how task scheduling can be classified in distributed systems is shown in Fig.1. At the topmost level, local and global scheduling have been distinguished. Thus local scheduling consists of assigning of processor time of a single processor to multiple processes whereas global scheduling consists of deciding where to execute a process. Therefore local scheduling is left to the local operating system of the processor to which the process is actually allocated. This gives the processors (nodes) increased authority and reduces the responsibility (resulting overhead) of the global scheduling strategy. However it doesn’t mean that global scheduling is the job of a single central authority, but rather that the problems of local and global scheduling can be considered as separate issues, and (at least logically) separate mechanisms are required to solve each. The second level in the hierarchy (below global scheduling) provides a choice between static scheduling and dynamic scheduling. This
division is based on the time of making scheduling decisions. Each stochastic task is characterized by its deadline and its
time cost distribution, which is represented in terms of mean
and variance. Here, static scheduling means assigning of
processes to processors before compilation (compile time)
whereas dynamic scheduling includes assigning of processes
to a processor during execution (run time), and thus can be
reassigned while they are running. The other difference is that
for static scheduling, decisions are made based only on
information regarding the processes and the static system
while a dynamic scheduler also takes into account the current
state of the system [34].

Wireless Distributed Systems (WDS) are heterogeneous
computing systems consisting of a varied set of machines,
varied set of communication protocols and programming
environments. They offer a range of capabilities in terms
of architecture for different execution requirements. One of the
challenging areas in such systems is the task scheduling
problem that deals with the distribution of a set of limited
resources to different tasks to optimize certain performance
criterions, like the completion time, communication delay, etc
[7][32].

2. RELATED WORK
Heterogeneous distributed systems have become widely used
for scientific and commercial applications. They are a cluster of
normal machines, programmable digital machines, and
application oriented ICs [26]. A heterogeneous distributed
system involves a number of heterogeneous modules which
communicate with each other to solve a problem [27]. The
applications running in such systems consist of multiple
subtasks that have varied execution requirements. These
subtasks need to be ordered for execution and assigned to
the various machines in the DS in such a way that the overall
execution time is reduced [14].

Nowadays, distributed systems are also being employed in
real-time applications, where the systems depend not only on
results of computation, but also on the instant at which these
results are available. The results of missing deadlines of hard
real-time systems may be disastrous, though such

consequences for soft real-time systems are comparatively
less risky. Some of the hard real-time applications include
aircraft control, radar for tracking missiles, and medical
electronics, railway crossings, etc. On-line transaction
processing systems are soft real-time applications. When it
comes to real-time applications, reliability plays an important
role and is one of the most important issues. With growing
needs of creating reliable real-time applications along with the
added advantage of high-speed networks and high-
performance machines, wireless distributed systems are being
increasingly used for many real-time applications in which the
output of the systems depend not only on the results of a
computation but also on the time at which these results are
produced.

A systematic execution of applications in such environments
requires good scheduling strategies that take into account both
logical and architectural features to achieve a good
assignment of tasks to processors so as to maximize some
performance criterion.

Also most of the jobs executed in many real-time systems are
critical in nature and therefore high reliability becomes a
compulsory requirement of these systems, and the case is
especially true for hard real-time applications.

Scheduling plays an important role in obtaining high
reliability and performance in distributed computing.
Moreover, a task may have different execution times for
different inputs. The objective of scheduling is to map tasks
onto machines and order their execution so that task
precedence requisites are satisfied with a minimum schedule
length. Apart from this aim, dynamic scheduling also provides
high reliability for non-preemptive, a-periodic, real-time jobs
without any additional hardware cost. Here, a framework
has been developed that is useful for simulated as well as real-
time scheduling by which the jobs are scheduled dynamically,
as and when they arrive in a distributed system. In this
framework, an identified machine, called scheduler, is made
responsible for initiating tasks for execution. The proposed
methodology also takes into account the dispatching and
scheduling times in addition to reliability costs. Most
scheduling algorithms neglect these parameters when dealing
with real-time computing. This approach is used in the

Figure 1: Hierarchical Structure Of Task Scheduling Classifications In Distributed Systems
simulation studies to make real-time jobs more predictable and reliable, and the scheduling more realistic.

A lot of scheduling algorithms have been applied earlier in the literature to support real-time scheduling. RT scheduling algorithms are classified into two categories: static or off-line scheduling [1][12][16][18][20][22][25] and dynamic or real-time scheduling [10][13][15][17][24][29]. Palis addressed task-scheduling problems with the context of reservation-based real-time systems that guarantee quality of service. The real-time tasks considered in Palis’s scheduling framework are preemptive tasks [17], while here it is assumed in proposed scheduling model that real-time tasks are non-preemptive. Various algorithms mentioned in [16][29] were designed for independent real-time tasks. However the algorithms like those described in [12][20][22], could not schedule tasks with precedence constraints, whereas proposed algorithm which represents tasks by directed acyclic graphs (DAG) can do so. The first testing of the algorithm has been done on non-real-time DAGs and will further be extended it with real-time DAGs to study the real-time scheduling of tasks [18]. However, most of the real-time scheduling algorithms have considered homogeneous systems as the base for implementation, making them unsuitable for use in heterogeneous systems.

In the literature, almost all parallel jobs have been represented by Directed Acyclic Graphs [2][7][12]. Wu et al. in his work, proposed a runtime parallel incremental DAG scheduling approach whereas Cosnard et al. presented a parameterized DAG scheduling algorithm, which first extracts symbolic linear clusters and then assigns these task clusters to various machines [7]. In distributed computing, a typical fork-join paradigm model [23] is used where the main program runs on one processor and spawns a number of tasks from time-to-time. Sahni and Vairaktarakis used this paradigm and developed efficient heuristics to obtain reduced finish time schedules [23]. However the scheduling algorithms here were also designed for homogeneous systems only.

Task scheduling in wireless distributed systems with heterogeneous machines has a number of challenges. To name a few, load balancing resource management [8] and reliable scheduling [3][6][31]. Scheduling with heterogeneous systems has been referred in many papers [4][6][11][21][28][30]. It can be found that minimizing the earlier task’s completion time further leads to a minimal start time of the next task [14][30]. Topcuoglu et al. in his work referred two efficient as well as low-complexity heuristics for DAGs: the heterogeneous Earliest-Finish-Time (HEFT) algorithm and the other one, the Critical-Path-on-a-Machine (CPOP) algorithm [30]. Özgüner proposed a matching and scheduling framework [9] whereas Maheswaran and Siegel designed a real-time matching and scheduling algorithm for heterogeneous system [14]. Beaumont worked with a static scheduling algorithm but for heterogeneous workstations.

In order that reliability of different resources be also given importance in a system while making scheduling decisions, Özgüner introduced two cost functions that were included into a matching and scheduling algorithm for tasks with precedence constraints [6]. Unfortunately, all the scheduling algorithms assumed that tasks are non-real-time. Hence non-real-time scheduling algorithms are inefficient in scheduling real-time jobs efficiently, because they are not designed to be capable enough to meet the predictability requirement of real-time jobs.

A lot of work has been done in real-time computing with heterogeneous systems [8][22][24][31]. Tracy et al. considered RT scheduling in distributed systems [31], Huh, Welch, Shirazi et al. designed an approach for dynamic resource management in real-time heterogeneous systems [8]. Ranaweera and Agrawal worked on scheduling scheme to reduce the number of pipeline stages [22]. Though the above algorithms considered both the real-time and heterogeneous systems issues into consideration, they did not focus on the issue of reliability. The uncertain nature of the task execution times and data transfer rates is also neglected by most traditional scheduling heuristics. Here, real-time scheduling has been proposed in heterogeneous distributed systems, to minimize the reliability cost of the systems. The scheduling algorithms developed in [18] were static in nature, whereas in this paper, the algorithms were dynamic.

Considerations for scheduling and dispatching times have been ignored by most dynamic algorithms whether they belong to non-real-time or real-time scheduling. To have reliable real-time scheduling results more precise, scheduling and dispatching times need to be incorporated in scheduling algorithms.

In this paper, only few of the parameters have been focused that are necessary for reliable scheduling for real-time systems. Further development in the work will focus on communication link failure and available processor power as the main parameters for reliable scheduling in wireless distributed systems. Hence, a variety of scheduling strategies with reference to non-real-time applications have not been discussed.

3. PROPOSED SYSTEM AND R-MODEL

Here a generalized system model for parallel applications running on a heterogeneous distributed system is described. The various parameters that affect the reliability cost in the reliability model are also discussed in this paper. Computation time, communication delay, etc. are also important for reliable scheduling of real-time applications in wireless heterogeneous systems.

A. System Model

The following figure-fig. 2 shows the scheduler model commonly used in scheduling processes. The model is similar to the one described in [10][11][15][28], wherein a global scheduler works in association with a Resource Manager.

The tasks are randomly generated using a graph representation language. A schedule queue (SQ) for arriving jobs is maintained by the scheduler. According to the dependency of tasks in a job, Directed Acyclic Graphs are constructed. Depending on the precedence constraints of the tasks, they are accordingly sorted. The scheduler schedules real-time tasks of each job in SQ and places an accepted job in a dispatch queue (DQ). Using the information conveyed through these DAGs, the tasks are scheduled on different processors. Thus from the dispatch queue, tasks of each accepted job are transmitted to designated machines, also called processing elements (PES), for execution. Assuming there are n tasks and m processors an n×m matrix is generated showing the execution costs of available tasks on the m processors.
The scheduler runs in parallel with processors and is not responsible for execution of tasks. Each scheduler maintains its own local queue (LQ) where real-time tasks are transmitted from Dispatch Queue. If all the tasks of an incoming job can be scheduled and completed within deadlines, then the job is considered as acceptable else, the job is rejected.

With distributed scheduling, jobs arrive continuously at each local scheduler and this scheduler produces schedules in parallel with other schedulers. The centralized scheduling scheme however has better approach as compared to distributed scheme. The implementation of a centralized scheduling model is simple and easier than that of a distributed scheduling model. Reliability requires fault tolerance which can be easily provided by the use of a backup scheduler that executes in parallel with the main scheduler. The backup scheduler works independently for identification of schedulable jobs but does not dispatch them to the processors until a failure of the primary scheduler is detected. The disadvantage of using dedicated schedulers is that they remain unutilized, but if when the scheduler is used as a scheduler and processor both, then a prediction mechanism will be required to identify the idle slot of the scheduler. Therefore, the centralized scheduler has been used as the scheduler model.

A real-time job is represented by a directed acyclic graph (DAG) \( G = \{V, E\} \), where \( V = \{v_1, v_2, ..., v_n\} \) indicates a set of real-time tasks, and \( E \) represents a set of directed edges with weights. The weights indicate the amount of data transmitted among real-time tasks through the edge connecting them. If \( e_{ij} = (v_i, v_j) \in E \), then \( e_{ij} \) denotes a message transmitted from task \( v_i \) to \( v_j \), and \( |e_{ij}| \) is the amount of data transmitted between these tasks.

In the proposed model, all tasks arrive from a task generator through a general task queue (Q) and get fed into a central processor called the system processor or the scheduler from where they are distributed to the dispatch queues, Q1, Q2, Qm, for all of the available processors in the system for execution, as shown in Figure 1. The wireless heterogeneous system is represented by a set \( P = \{p_1, p_2, ..., p_m\} \) of machines, where \( p \) is a processor with its own memory. Machines in the wireless distributed system communicate with each other through message passing and are interconnected by a high-speed network. The communication delay between two tasks assigned to the same machine is assumed to be zero [19][20][30].

### B. Reliability Model

The underlying model for computing reliability of WDS is similar as the one defined in [18][19][27], that presumes that permanent failures occur according to a Poisson probability distribution and that the failures are mutually independent. Let \( X \) be an \( m \) by \( n \) binary matrix corresponding to a schedule, where \( n \) tasks of a job are assigned to \( m \) processors in the system. The entry for element \( x \) equals 1 if and only if \( v \) has been assigned to \( p_j \) else \( x_{ij} = 0 \).

Machine failures during an idle time, are not considered since machine’s failure during an idle period can be overcome by replacing the failed machine with a spare unit, in proposed reliability model[18][27]. However power required for computation of a task with a wireless distributed system is required to be taken into account [9]. The reliability cost of a task \( t_i \) on \( p_j \) is a product of \( p_j \)’s failure rate \( \lambda_i \) and \( t_i \)'s execution time on \( p_j \). Thus, the total reliability cost of an individual machine is the summation of the reliability costs of all tasks assigned to that machine. Given a vector of failure rates \( \Lambda = (\lambda_1, \lambda_2, ..., \lambda_m) \), a specific schedule \( X \), and a job \( J \), the reliability cost of the distributed system is defined as below,

\[
RC_{p_j}(\Lambda, X, J) = \sum_{j=1}^{m} \sum_{i=1}^{n} (-\lambda_i x_{ij} c_{ij})
\]

To achieve better overall reliability, scheduling a task with larger execution time to a more reliable machine is a better choice. However, scheduling in WDS is a NP hard problem and no defined solutions exist to solve it. The methods that have been used can be divided into 2 categories: exact algorithms and approximation algorithms. The exact algorithms commonly use the branch and bound technique and are applicable to small-sized problems. On the other hand, the approximation algorithms derive sub-optimal task allocations within reasonable times. Heuristic and Genetic algorithms (GAs) have also been adopted for solving problems and obtained promising results. GAs (Goldberg, 1989) belongs to a branch of computational intelligence called metaheuristic. The other fact is that exact algorithms search for optimal solutions and are thus computationally intensive, while metaheuristic algorithms giving near-optimal solutions within reasonable times are more suitable for real-time applications [33].
4. EXPERIMENTS & RESULTS

It is assumed that all of these tasks are synchronous, i.e. their first request arrives simultaneously at the time zero. The scheduler will run in parallel with the applications processors, scheduling the new ready tasks, from the task queue (Q), and periodically updating the dispatch queues. This organization ensures that the processors will always find some tasks in the dispatch queues [33][34][35] when they finished their current tasks. The objective of proposed work is to present a simple heuristic scheduling algorithm which accounts for maximizing both the reliability of the application and yielding the required probability for each task in this application to meet its deadline, at the same time. This algorithm exploits parallelism in tasks whenever needed to satisfy the required objectives and improve the previous results,[36].

A. Algorithm:

```plaintext
{ 
Till there are tasks in the queue
Send them one by one to the central processor or scheduler
Processor=1
While scheduler not empty 
{ 
Check for each processor in the system
While (processor load !=max && there is enough processing power)
{ 
    Assign task to the processor break
}
Processor = processor +1
If processor= max then processor =1
}
One processor is the master processor or scheduler. Following are few of the parameters that were tested for reliability

a. Computation time: time required by each task to execute on each processor.
b. Communication failure rate: the failure rate between two processors during communication.
c. Communication delay: the time required to transfer data between two tasks of two different processors.
d. Intertask communication on same processor is assumed to be zero.
e. Failure rate: probability that the processor will fail.
f. Weighted matrix: amount of data transferred between various communicating tasks.

The graph for CPU utilization is shown in fig. 3.
The task graph considered is shown in fig. 4 and values for various parameters related to computing are as follows

Failure Rate Matrix:

<table>
<thead>
<tr>
<th></th>
<th>0.000000</th>
<th>0.000000</th>
<th>0.000000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td></td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Figure 3: CPU utilization

Figure 4. Task graph considered for scheduling

Communication Delay Matrix:

<table>
<thead>
<tr>
<th></th>
<th>0.000006</th>
<th>0.000003</th>
<th>0.000002</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.000002</td>
<td>0.000005</td>
<td>0.000002</td>
</tr>
<tr>
<td></td>
<td>0.000002</td>
<td>0.000003</td>
<td>0.000003</td>
</tr>
</tbody>
</table>

Computation Time Matrix:

<table>
<thead>
<tr>
<th></th>
<th>0.000000</th>
<th>0.000000</th>
<th>0.000000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.000002</td>
<td>0.000003</td>
<td>0.000000</td>
</tr>
<tr>
<td></td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000006</td>
</tr>
<tr>
<td></td>
<td>0.000003</td>
<td>0.000002</td>
<td>0.000002</td>
</tr>
<tr>
<td></td>
<td>0.000005</td>
<td>0.000002</td>
<td>0.000002</td>
</tr>
</tbody>
</table>

Failure Rate Array:

<table>
<thead>
<tr>
<th></th>
<th>0.000000</th>
<th>0.000000</th>
<th>0.000000</th>
</tr>
</thead>
</table>

Weighted Matrix

<table>
<thead>
<tr>
<th></th>
<th>-0.000</th>
<th>1.000</th>
<th>0.000</th>
<th>0.000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.000</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>3.000</td>
</tr>
<tr>
<td></td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>-0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

5. CONCLUSION
In this paper, a methodology has been proposed, which maximizes the wireless distributed system reliability for the purpose of successful execution of a task. The scheduler takes care to schedule the tasks onto the appropriate processor so that the task is also completed and there is no overloading of processors. Testing is being carried out for maximizing reliability of scheduling tasks. Once it is done, reliability of real time tasks will also be taken care of for the wireless distributed systems.

6. REFERENCES
[7] Xiao Qin , Hong Jiang , “A Dynamic and Reliability-Driven Scheduling Algorithm for Parallel Real-time Jobs on Heterogeneous Clusters” at Department of Computer Science, New Mexico Institute of Mining and Technology.


[34] He, Zili Shao, Bin Xiao, Qingfeng Zhuge, Edwin Sha, “Reliability Driven Task Scheduling for Heterogeneous Systems”.
