Proactive Fault Tolerance Technique for a Mobile Grid Environment

Abhishek Bichhawat[1] and R. C. Joshi[2]

Department of Electronics and Computer Engineering,
Indian Institute of Technology Roorkee,
Roorkee-247667, India

[1]abhibpec@iitr.ernet.in
[2]rcjosfec@iitr.ernet.in

***Abstract*— Mobile grid is a promising paradigm in the field of distributed computing in which mobile devices along with wired devices are part of the grid and participate in the activities of the grid as the wired devices do. One of the major challenges which hurdles the growth of mobile grid computing is the issue of faults and failures. The network is subject to intermittent connections and limited bandwidth which leads to frequent disconnections amongst mobile devices. Apart from the network, the device's own resource pool is limited by battery and power. Fault-tolerant execution is necessary for extracting maximum benefits from the mobile devices. In this paper, we present a task migration based approach to support fault tolerant execution of jobs on mobile devices. The proposed technique provides better utilization of resources compared to other fault tolerance techniques.**

Keywords- Task migration; fault-tolerance; mobile grid

I. INTRODUCTION

A Grid is defined as a system that coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service [1]. Mobile computing is another computing paradigm of distributed systems, considering mobility, portability and wireless communications [2]. The extension of the grid to mobile computing by making it available to the users even when they are mobile forms the basis of Mobile Grid. Mobile Grid, in relevance to both grid and mobile computing, is a full inheritor of grid with the additional feature of supporting mobile users and resources in a seamless, transparent, secure and efficient way [3]. The mobile grid can provide higher computational power and resources than the existing grid technology.

Mobile grid could help in the utilization of any unutilized resources on the devices. Apart from utilizing resources on the mobile devices, the mobile grid can provide mobile devices with an opportunity to use the resources on the grid, thereby saving their own resources considerably and overcoming the physical shortcoming of the device. Mobile devices having access to the grid as users would thus be able to perform certain tasks on the run which otherwise would have been done only when the user could access a wired device. Certain situations consent the contribution of resources from mobile devices to the grid resource warehouse.

As a combination of both mobile and grid computing, mobile grid raises a combined set of issues involving factors from both the computing areas. Mobility of devices raises much more issues and aggravates the existing issues. One of the major issues the mobile grid faces is that of fault-tolerance. It is necessary for a mobile device to execute a job in a fault-tolerant fashion to avoid wastage of resources. Mobile devices have various problems such as intermittent connection, limited power supply, and low communication bandwidth. These problems result in unpredictable network behavior and unreliable communication. Due to this, mobile grids also suffer the problem of frequent communication failure. Communication failure has an adverse effect on the grid as there might be a data transfer in progress or some message passing amongst processes which gets interrupted and shall result in degradation of performance. Apart from communication failure, the shortage of battery or power resources on-device is another reason for faults and failures of mobile nodes.

Some of the ways for ensuring fault tolerance include checkpointing and task replication. Checkpointing is the process of saving the current state of the process. In case of a failure, the task can be restarted from the last saved state. Task replication provides fault tolerant performance by executing the task on several devices by creating replicas of the task. Thus, if a node performing the task fails, the result can be obtained from some other node to which the task has been allocated.

Task migration is a proactive fault tolerance technique which can be used for fault-tolerant performance in a mobile grid environment. In this technique, the job under execution can be migrated to some other potential resource, the base station or the grid server in case a fault occurs at the mobile node. Another device can then start the execution of the job, from the point where it stopped executing at the previous resource, providing its resources for execution.

In this paper, we provide a fault tolerance technique based on migration of tasks by predicting occurrences of failures on a mobile device. Section 2 defines briefly about task migration and its relevance as a fault-tolerance technique in mobile grids. We also present some previous work in the field of fault tolerance using task migration. Section 3 describes the architecture for implementation of task migration. Section 4 provides the scheme for migration in case of faults. In section 5, we discuss the existing techniques and works in the field of fault-tolerance in mobile grids and compare our scheme with

those approaches. We conclude with some future research proposals.

II. BACKGROUND

A. Task Migration

Task migration can be defined as the process of dynamically relocating a process from the node on which it is executing to another node which is capable of executing it [4]. Migration can be achieved at various levels in the system, either user or kernel level. Most implementations for migration support both migration styles. As we are concerned with the migration of a job and not the level of operation, we shall use the terms process and task migration interchangeably.

Task migration allows the system to take advantage of changes in process execution or network utilization during the execution of a process. The migration of a task is a complex process. It requires the current state of the task to be saved and transferred from one system to another. This state capture is with the help of a local process manager which keeps a record of all the processes running on the system.

Some of the notable advantages of task migration include [5, 6]:

- Load balancing can be performed by migrating processes from heavily loaded nodes to lightly loaded nodes.
- Fault resilience can be achieved by migrating processes from partially failed nodes or in response to an anticipated node failure.
- Migration of processes closer to the source of some data required. This reduces the access time for the application.
- Without disrupting the service to the clients, a server can be shut down or made unavailable for maintenance by migrating tasks running on the server to some other system.
- Users using complex applications would be able to work on another computer at another location without closing the application, by migrating applications.

B. Fault Tolerance by Task Migration

Occurrence of failures on systems is a common phenomenon. The failures can be classified as hardware or software and can be a result of some erroneous program running on the system, or due to overheating, or no power availability etc. Apart from system failures, in networks, communication failures also result. This could be an outcome of no network availability or fault in the communication channel etc. Many schemes for fault tolerance have been proposed based on task migration. Systems have also been designed to facilitate fault tolerance by task migration.

Task migration for fault tolerance has already been proposed for various systems. In [12], the authors present a fault tolerance technique based on task migration for MPI applications in high performance computing environments. A similar scheme for fault tolerance using process-level live migration was predicted by Wang et al. in [14] for high performance environment. Nagarajan et al. presented a fault tolerance technique based on operating system virtualization techniques as in Xen [13]. They used Xen's live migration mechanism for a failing node to migrate an MPI task to a prospective node for continuing the application. A proactive fault recovery using process migration was suggested in [15] for distributed CORBA applications. Recently, a proactive fault tolerance methodology using preemptive migration was suggested in [17]. The proposed schemes and methods are all based on fault prediction based on health-monitoring. The schemes initiate process migration by monitoring temperature and other parameters which provide an indication of occurrence of a failure. The methods described have been proposed for high performance computing systems, especially the ones running MPI applications.

Apart from these, task migration has been studied for distributed computing systems for providing load balancing [16]. A method for task migration in grid environment has been proposed by Frechette and Avresky [18]. The method instigates task migration in response to network failures and denial of service (DoS) attacks. But, there has been no study of a fault tolerance approach based on task migration for a mobile grid environment.

In a mobile grid environment, most of the failures are understood to occur on mobile devices and are caused due to power shortage or disconnections. Failures of the nodes at hardware level are less common in mobile devices. The task migration scheme proposed in this paper tackles faults occurring due to battery power shortage or due to intermittent connectivity.

III. SYSTEM MODEL

Fig. 1 shows the system model of the infrastructure which has been considered while developing the scheme for migration of tasks. The grid server is a wired node that receives requests for job executions from various nodes. The resource discoverer discovers the different nodes connected to the server at a given instant and identifies the various resources available on the device. It also differentiates between mobile devices and wired devices so that the jobs can be scheduled accordingly. The resource manager takes care of the resources on different nodes by monitoring the jobs and resources. The log is where the usage record of job and user is maintained. The security service takes care of securing the authentication, authorization and managing identity amongst others like single sign-on.

The schedulers for grid and mobile grid environment are separated in order to allow efficient and optimal scheduling in both the environments. The grid scheduler uses optimal scheduling algorithms while the mobile grid scheduler uses algorithms based on availability of the devices. The separation allows computationally intensive and critical real time jobs to be assigned to the wired part of the grid and computationally less intensive jobs not bound by any time constraints can be assigned to the mobile part of the grid.

The mobile device runs a process known as *monitor* process. The process keeps monitoring the battery levels and

the connection strength on the mobile device and stores it. The job runs on the mobile device while the *monitor* process runs in the background. In case a migration interrupt is generated, the job is sent back to the server which finds another device to allocate the job.

IV. Migration Scheme

Fig. 2 describes the migration scheme which a mobile device shall follow while executing a job. The job is allocated to a mobile device by the grid server. The *monitor* process runs in the background and is responsible for initiation of task migration. The mobile device is concerned only with the execution of jobs. When the battery level falls below a threshold or the signal strength is very low, the *monitor* process interrupts the job and directs it to be migrated. The mobile device establishes a communication with the server and transfers the current state of the job to the server. The server, then, searches for another potential device to begin the execution of the job from the saved state and guides the job towards that device.

The scheme ensures that the mobile device does not run out of battery resource at any point of time while its resources are used for computational purposes or for providing services to the grid. Alongside, the scheme takes care that the device does not fail due to power shortage. This also ensures that the device owner has some minimal resources available for his usage at any time. Apart from the power factor, the scheme also considers the network availability on the device. Considering the signal strength, the scheme prepares the mobile device for migration of the job if the signal strength is low and constantly reducing. This ensures a fault resilient performance because the period of disconnection is not known accurately and the job results might not be available for quite some time.

The threshold value for battery is set such that the user always has some minimal power available on the mobile device for his personal use. It is set such that the mobile device is able to provide service to the user until the user can recharge it again. The network threshold value is set such that the mobile device has enough time to complete the migration of task to the grid server. The threshold value is an indicative value which predicts the occurrence of failure to a fair accuracy.

The *monitor* process can be implemented in two different ways: (1) the process can be part of the job under execution. It continuously polls the system to obtain the power and signal information. If the need for migration arises, it migrates from the mobile device to the grid server, else it continues with the job; (2) the process can be run in the background independent of the job under execution. When the job arrives on the mobile device, it initiates the execution of *monitor* process. This is to ensure that the *monitor* process runs only when a job is being executed on the mobile device and does not use the resources when there is no job to be executed by the mobile device. The process requires nominal processing power but ensures that the migration be initiated at the required time to prevent the occurrence of failure.

V. Implementation

The migration scheme described in section IV has been implemented using mobile agents over mobile devices, mainly laptops. The mobile agents were used to transfer the job from the central scheduler to the prospective mobile host. The mobile agents retracted in case a fault was anticipated in form of resource shortage on the mobile device.

The monitor process starts as soon as the mobile agent begins execution on the mobile device. The resources are continuously monitored by the monitor process. In other words, polling is done to determine any significant reduction in the battery power or the signal strength. When the battery power or the signal strength falls below a certain predetermined threshold, the mobile agent retracts to the central scheduler, saving all the work done on the mobile device.

VI. Comparison with Other Fault-Tolerance Approaches

Two common approaches to ensure a fault tolerant performance in mobile grid include checkpointing and task replication. Checkpointing is the process of saving the current state of the process. In case of a failure, the task can be restarted from the last saved state. Methods in [7-9] provide checkpointing approach for achieving fault resilience. These algorithms take into account various parameters for initiating and taking checkpoints. The main overhead associated with checkpointing is the saving of state information at regular intervals. These techniques result in frequent checkpointing in certain situations. The transfer of checkpoints to the base station is another overhead for the mobile device. A decentralized approach to checkpointing was suggested in [9] where a mobile host may choose to take a checkpoint and save it on its neighboring hosts rather than saving it on a base station. This results in an overhead on the mobile host as requests and breaks keep occurring continuously. The message traffic also increases in the network which is undesirable. Another consideration would be that of memory which is limited in mobile devices where the checkpoints are stored.

Task replication provides fault tolerant performance by executing the task on several devices by creating replicas of the task. Authors suggest the use of replication as a fault tolerance mechanism in [10-11]. When a fault occurs, results from the other device can be used. This requires proper determination of the number of replicas to be run on different devices because if the number is too large, it would result in worthless redundant information, unless all other devices fail, and if too small, there might be a possibility that the results are not obtained and the resources were wasted on the devices. The main issue with task replication is the fact of the redundant information. The resources on certain devices might have done a job which is not useful in actual computation if all the devices have produced the result without any faults. The technique reduces the effective utilization of the resources because multiple resources execute the same job and in scenarios with fewer failures, redundant data is available. This is not desirable in a grid

computing environment where we look to maximize the resource utilization for better computation.

Task migration on the other hand requires a process to run in the background which consumes minimal resources. The job needs to be transferred only when the *monitor* process discovers that it is running out of battery or the signal strength is very low. This consumes fewer resources when compared to those in checkpointing and the resources can be used more efficiently. Task migration also ensures that all the productive work done by the mobile device is preserved. In any case, there is no loss of work done by the mobile device which results in efficient utilization of resources. However, in case the device suffers a hardware failure, the presented migration technique is unable to recover from the failure. As the mobile devices suffer faults due to resource shortage more commonly, the technique shall be advantageous when compared to the other two techniques in a mobile grid environment.

All the three techniques ensure that the unused resources are utilized to the maximum, but if the utilization efficiency, the efficiency of utilization of the resources, i.e., the amount of useful work done without any loss, is considered, the technique of task migration proves better than the other two techniques. Task migration incurs an overhead with the monitor process running in the background. This overhead along with the overhead of task migration is higher than what is incurred due to the technique of checkpointing. But, the technique of task migration ensures that no useful work done by the device is lost due to any failure, which could have been predicted and avoided.

Task replication uses twice or more resources to perform the same task and hence, the amount of useful work done reduces to half or less than that. This technique shall only be useful in case of critical real time jobs. In normal functioning of the grid, this only results in wastage of resources by redundancy.

Checkpointing ensures that the useful work done by the device or the host is saved, so there is no need to perform the task from beginning, if a fault occurs in the future. Although this technique ensures that useful work done is not lost, still some of the useful work done might be lost due to the period between two checkpoints. Checkpointing is costly and hence, cannot be performed very frequently to ensure saving of all the work. And if performed over longer intervals, there might be a possibility of losing a small but considerable amount of work done by the device.

Using task migration and certain fault predictive techniques, fault tolerance in mobile devices can be achieved at a higher level as compared to checkpointing and task replication.

VII. Conclusion

The task migration scheme presented in this paper tackles the issue of fault tolerance in a mobile grid environment by incurring very less overhead to the mobile device. The scheme presents a simple technique for preventing failures by monitoring the resources continuously and migrating tasks in case a failure is anticipated. The technique shall provide better and efficient utilization of resources as compared to the techniques of checkpointing and task replication in a mobile grid environment. In future, we plan to implement the migration scheme in a mobile grid environment along with other fault tolerance approaches and to develop a hybrid technique comprising of the fault tolerance techniques to ensure maximum utilization of resources. We also plan to extend the scheme to handle hardware faults by identifying various changes in the device which might result in a failure.

References

[1] I. Foster, “What is the Grid? A three point checklist,” Argonne National Laboratory, fp.mcs.anl.gov/~foster/ Articles/WhatIsTheGrid.pdf, 2002.

[2] M. Satyanarayanan, “Fundamental challenges in mobile computing,” Proceedings of the fifteenth annual ACM Symposium on Principles of Distributed Computing, Philadelphia, Pennsylvania, pp. 1-7, 1996.

[3] A. Litke, D. Skoutas, and T. Varvarigou, “Mobile grid computing: Changes and challenges of resource management in a mobile grid environment,” 5th International Conference on Practical Aspects of Knowledge Management, 2004.

[4] D. S. Milojicic, and Y. Paindaveine, “Process vs. Task migration,” Proceedings of the 29th Hawaii International Conference on System Sciences (HICSS'96) Volume 1: Software Technology and Architecture, pp. 636-645, January 03-06, 1996.

[5] S. Zhang, M. Khambatti, and P. Dasgupta, “Process Migration through Virtualization in a Computing Community,” 13th IASTED International Conference on Parallel and Distributed Computing Systems (PDCS2001), Anaheim, CA, August 2001.

[6] D. S. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, “Process migration,” ACM Computing Surveys (CSUR), vol. 32, no. 3, pp. 241-299, Sept. 2000.

[7] I. Rao, and T. Chung, “A proxy based efficient checkpointing scheme for fault recovery in mobile grid system,” High Performance Computing - HiPC 2006, pp. 448-459, 2006.

[8] J. Park, and H. Yu, “Context information based fault tolerant technique in mobile grid,” Whitepapers, Mar. 5, 2008.

[9] P. J. Darby III, and N. Tzeng, "Decentralized QoS-aware checkpointing arrangement in mobile grid computing," IEEE Transactions on Mobile Computing, vol. 9, no. 8, pp. 1173-1186, Apr. 2010.

[10] A. Litke, D. Skoutas, K. Tserpes, and T. Varvarigou, “Efficient task replication and management for adaptive fault tolerance in mobile grid environments,” Future Generation Computer Systems, vol. 23, no. 2, pp. 163-178, February 2007.

[11] S. Choi, I. Cho, K. Chung, B. Song, and H. Yu, “Group-based resource selection algorithm supporting fault-tolerance in mobile grid,” Proceedings of the Third International Conference on Semantics, Knowledge and Grid, pp. 426-429, 2007.

[12] S. Chakravorty, C. L. Mendes, and L. V. Kale, “Proactive fault tolerance in MPI applications via task migration,” In Lecture Notes in Computer Science: Proceedings of the International Conference on High Performance Computing (HiPC), vol. 4297, pp. 485–496, 2006.

[13] A. B. Nagarajan, and F. Mueller, “Proactive fault tolerance for HPC with Xen virtualization,” Proceedings of the 21st Annual International Conference on Supercomputing (ICS’07), pp. 23-32, 2007.

[14] C. Wang, F. Mueller, C. Engelmann, and S. L. Scott, “Proactive process-level live migration in HPC environments,” Proceedings of the 2008 ACM/IEEE conference on Supercomputing, Austin, Texas, 2008.

[15] S. Pertet, and P. Narasimhan, **“**Proactive recovery in distributed CORBA applications,” Proceedings of the 2004 International Conference on Dependable Systems and Networks, pp. 357-366, 2004.

[16] T. T. Y. Suen, and J. S. K. Wong, “Efficient task migration algorithm for distributed systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 4, pp. 488-499, July 1992.

[17] C. Engelmann, G. R. Vallee, T. Naughton, and S. L. Scott, “Proactive fault tolerance using preemptive migration,” Proceedings of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 252-257, 2009.

[18] Stephen Frechette, and D.R. Avresky, "Method for task migration in grid environments," Fourth IEEE International Symposium on Network Computing and Applications, pp.49-58, 2005.

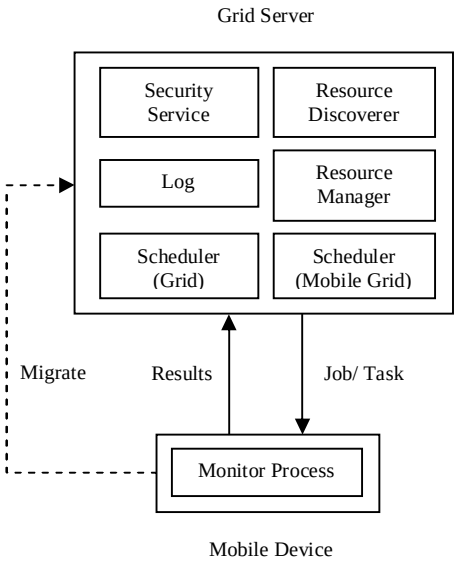

Figure 1. System Model

```
getcurrentstatus() {
        Assign battery_status with the available
        system battery;
        Assign network_status with the system
        network connectivity, i.e., the % of
        connectivity available;
}

monitor(){
do {
        getcurrentstatus();
        if (status_battery < battery_threshold) or
        (status_network < network_threshold) then
                job_migrate();
        end if
} while(true);
}

job_migrate() {
        Save the current state of the job;
        Establish communication with the server;
        Transfer the current state of the job to the
        server;
}

do_work(){
        Perform the task assigned to the host
}
```

Figure 2. Migration Scheme