Simulation Framework for Evaluation of Fault Tolerant Large Dynamic Distributed System

Sanjay Bansal  
Acropolis Institute of Technology & Research  
Indore, India

Sanjeev Sharma  
Rajiv Gandhi Prodyogiki Vishwavidya  
Bhopal, India

Ishita Trivedi  
Rajiv Gandhi Prodyogiki Vishwavidya  
Bhopal, India

Ankita Verma  
Acropolis Institute of Technology & Research  
Indore, India

Mirnalika Ghosh  
Acropolis Institute of Technology & Research  
Indore, India

ABSTRACT
The use of Java based simulators in the design and development of distributed system for evaluating the dependability on algorithms is appreciable due to their efficiency and scalability. It allows in designing the realistic simulation scenarios. In this work, we have proposed a Saturn, a multithreaded process oriented over simulation framework which is designed for modeling large scale distributed system. Realistic simulation is provided by it to provide a wide-range of distributed system technologies. It is an innovative solution to the problem of evaluating dependability characteristics of distributed system. Our solution is based on several proposed extensions to the simulation model of the MONARC simulation framework. These extensions refer to fault tolerance and system orchestration mechanisms in order to access the reliability and availability of distributed systems. The extended simulation model includes the necessary components to describe various actual failure situations and provides the mechanism to evaluate different strategies for replication and redundancy procedure as well as security enforcement mechanism. It is a simulator which also evaluates major QoS of the heartbeat based adaptive failure detection mechanism.

Keywords
Simulation; Fault Tolerance; Large and Dynamic Distributed System; Modeling.

1. INTRODUCTION
Fault tolerance in distributed system has emerged as an important and essential requirement in today’s large and dynamic distributed system. Fault tolerance is one of the most challenging parts in research domain. Solving critical issues as per the new trend which includes performance orientation and capability to handle multiple faults i.e. permanent failure of multiple nodes, process and links are emerging day by day[QL,12]. These new issues require consideration for several aspects like fast and accurate detection of permanent failure of multiple nodes and process and hence this improves recovery algorithms to recover the system as early as possible with better resource utilization. Besides this, improved strategies for replica generation and placement can result in better dependability rather than just adding the redundancies. Redundancies must be added when it is required rather than adding it equally for all components. More redundancies can be added for more critical and important component as compared to less critical and less important data. This results in performance oriented dependable approach which can tolerate more number of permanent faults too[AA, 11].

Modeling and simulation are the viable solutions to evaluate the new algorithms which are developed for enhancing the performance and fault tolerance in large and distributed systems. Use of discrete-event simulators is appealing due to their efficiency and scalability [AB, 11]. Modeling and simulations are effective tools for development of new algorithms for fault tolerance distributed computing. The strength of discrete event simulation are flexibility less time and efforts that it take to evaluate a newly design algorithm. MONARC (Models Of Networked Analysis at Regional Centers) is a discrete event simulator designed to evaluate algorithms in the domain of large and complex distributed system [AB PAPER KA, 5]. This tool is used to simulate network protocols, scheduling algorithms and also for large and complex distributed systems. MONARC can be extended to perform analysis of the new algorithms developed for dependability. In this paper, we present an extension to the MONARC simulation model that can be used to evaluate the performance of fault tolerance algorithms. This model simulates various failure scenarios in a realistic way and also provides dynamic configuration possibility to map a real situation which usually occurs in real distributed systems. It can also perform evaluation of failure detection mechanism when multiple faults occur like failure of multiple nodes, processes and links. Various replication locations, placement and generation can be simulating for evaluation of dynamic situations. Recovery process with various mode and approaches can be simulated and evaluated by the proposed simulator. Since MONARC can easily be extended and adapted [AD, 11 PAPER KA 6-10]. Hence, this model with very less efforts and time can be extended to simulate various failures like failure detection mechanism, recovery strategies, replication, check pointing replication and consistency scenario. The proposed framework includes a wide range of failures like permanent failure of nodes, processes, links, storage device as well as transient faults. Simulation of check pointing replication which is widely used in multiple fault tolerance technique is also possible in a flexible and reconfigurable way. The important components like fault injection; fault detection and recovery are included as independent entities.
In section 2 we present related work. Section 3 presents an overview of MONARC which is presented as a formal and informal architecture of proposed extension of MORAC.

2. RELATED WORK

SimGrid provides core functionality for the evaluation of scheduling algorithms in a heterogeneous and largely distributed system known as Grid [HC, 08]. GridSim is a grid simulation which investigates effective resource allocation techniques based on Grid [RB, 02] containing computational economy. OptorSim [WV, 07] is a Data Grid simulator for testing various optimization technologies. ChicagoSim [KR, 02] is a simulator designed to investigate scheduling strategies in conjunction with data location.

Discrete Event System Specification (DEVS) is a mathematical formalism [BP, 00] to describe real-world system behavior in an abstract and rigorous manner. Object oriented language, such as Java, are used as a framework for modeling and simulation based on DEVS [HTTP1]. It has reached to a mature stage and is widely used to simulate many real world applications. DEVSJAVA [HTTP2] is an implementation in Java of such a framework. None of these projects present general solutions to modeling fault tolerance technologies for large scale distributed systems. They tend to focus on providing evaluation methods for the traditional research in this domain, which has recently targeted the development of functional infrastructures. For fault tolerance in case of large scale distributed systems our model aims to provide the means to evaluate a wide-range of solution. The simulation model provided by MONARC is more generic than others, as demonstrated in [CD, 08]. It is able to describe various actual distributed system technologies and provides the mechanism to describe concurrent network traffic, to evaluate different strategies in data replication, and to analyze job scheduling procedures. MONARC offers ample customization possibilities, thus enabling us to integrate our model while preserving the interface. Because of this feature, our own model can incorporate custom failure, recovery and rescheduling algorithms that the user may need for a particular scenario.

3. MONARCH SIMULATION

A process oriented approach is suitable for discrete event simulation over dependable computing in a distributed environment and MONARC is supported for the same. Due to this it is most convenient to simulate concurrent tasks to be executed on various computing nodes [IL, 03]. In real situation a large distributed system consists of large number of nodes and millions of concurrent tasks scheduled to execute on these large number of nodes with data transfer in Giga bytes or even more. MONARC is suitable for simulation of such real situations. The major components of MONARC are processing unit, storage devices and other important networking components such as hubs, switches etc.

Such components can be grouped in regional centers (Figure 1). A regional center is used to abstract a group of resources that are under the control of a single organization. Several regional centers can be linked to simulate cooperation with other resources, similar to how clusters are used in larger Grids. The simulation model also includes components to model the behavior of the applications and their interaction with users. Such components are used to generate data processing. One of the strengths of MONARC is that it can be easily extended. This is made possible by its layered structure. The first two layers contain the core of the simulator (called the “simulation engine”) and the second layer contains the models for the basic components of a distributed system (processing units, jobs, databases, networks, job schedulers etc). The particular components can be different types of jobs, job schedulers with specific scheduling algorithms or database servers that support data replication. Using this structure it is possible to build a wide range of models, from the very centralized to the distributed system models, with an almost arbitrary level of complexity.

![Fig. 1. The Regional center model being adopted in MONARC, OM [16]](image1)

![Fig 2: The layers of MONARC [CD, 08]](image2)

4. Formal and Informal Architecture of Extended MONRAC (XMONARC)

The extended model consists of additional important and essential components required to simulate the fault tolerance related aspects in large and dynamic
distributed system. Thus, model includes various modules to inject faults, failure detection, recovery, statistics, consistency, redundancies, resource management and allocation, fault tolerance capability variance and configuration; etc. The proposed framework is shown in figure 1.

The basic function of this module is to model faults in various parts and to find location of the distributed system. Various failures can be simulated in any component of the system. It includes failure of permanent nodes, processes, failure if links, transient faults, etc. In order to implement this java RMI based client program is developed. To stop the client program manually or automatically simulation of permanent faults is done. Whereas, to simulate transient faults client programs are stopped and started at different stages of time interval. This allows us to simulate wide range of fault injection which may be of any characteristics. Simulation of multiple node failures separately or simultaneously can be obtained very accurately. Failure of another node while recovery of first node is on progress can also be simulated easily and independently. Fault injection can be configured by any mechanism like mean time to failure (MTTF), regular mathematic mechanism i.e. Binomial, Poisson, Guassian, etc., and random and abruptly. Fault injection module works in coordination with fault tolerance module.

4.2 Fault Tolerance Module

It provides mechanism for failure detection and recovery protocols. Failure detection is implemented using heartbeat mechanism. After detection of failure of nodes, rescheduling can be done on other healthy processing units. In order to model a permanent fault, the client program which is written using java RMI, stopped based on simulation requirement. It also allows simulation of hybrid and includes check pointing and message logging mechanism. Various check pointing and replication strategies can be implemented easily. Failure detection mechanism is based on heartbeat strategies. A trigger system is implemented to inform recovery module to start recovery after failure is detected. Static, dynamic and adaptive dynamic heartbeat strategies can be implemented easily in this module.

4.3 Fault Capability Variance and Configuration Module

The fault tolerance capability module can configure a specified fault tolerance capability as well as can be used to evaluate the maximum, minimum and average fault tolerance capabilities by simulating a wide range of scenario. Specified fault tolerance capability can be set by this module. It can also set different fault tolerance capabilities of components based on components importance and criticality. Run time enhancement of capability can also be configured for some specified components that can also be simulated by this module. In order to implement these scenarios, check pointing replication are increased or decreased by this module. A multithreaded check pointing replication is triggered to check point of an application node or process. Degree of redundancies is main core fundamental decision factor for configuration of a specified fault tolerance capability for application nodes or processes.

4.4 Check pointing Replication Tracking Module

It works as Replica Location Service (RLS) of proposed architecture. An RLS provides a mechanism for discovering and registering the existence of replicas. Master Index module keeps the Meta data of all stored replicas at different storage nodes. The Meta data of replica consists of replica id which includes the information that is of which application node the replica is, replica number, replica size, storage node where it is stored and it also gives when it was last updated and created i.e. timings etc. Each replica file is described by the attributes of a predefined metadata schema. The search mechanism in the Metadata aims to return replica file by fulfilling the search criteria. The metadata file also includes the Logical Replica Name (LRN) of the corresponding replica file. This LRN, is used as a key in our distributed catalogue which is implemented in the Distributed Replica Location Service (hence DRLS) overlay. In this overlay, mappings of LRNs to the physical location of data files is represented by Physical Filenames (PFNs) stored. This module stores and provides information of all replicas stored at any location. It also provides all locations, where identical replicas are stored and location of a particular replica. In order to prevent it from single point failure, more than one copy is stored in different locations. This information is also used by
consistency module to maintain the consistency among all replicas of same application nodes.

4.5 Consistency Module

This module is used to maintain consistencies among all replicas generated for the purpose of fault tolerance as a redundancy. It can use any algorithm developed to maintain the consistency among all replicas stored at different locations in given distributed computing environment. In order to tolerate n number of faults, n+1 replica are stored at different locations. Consistency module ensures the consistencies among the replicas of same application nodes or processes. In order to tolerate more number of faults there is need to generate replica or replicas on run time. In such situations, consistency module creates the replica with updates in order to prevent the inconsistency situation.

5. COMPARISION

We compare our proposed simulation framework with other recent simulation framework which was anticipated by two eminent researchers on certain important features. The comparison of our proposed work with the work of Mr. Adrian Botenau is shown in following table 1.

<table>
<thead>
<tr>
<th>Feature</th>
<th>XMONARC</th>
<th>MONARC by Cilprian Dorbe[CD.08]</th>
<th>MONARC by Adrian [AB.11]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check pointing Replication</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Fusion</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Injection of Faults</td>
<td>Software as well as hardware Induced</td>
<td>Hardware Induced</td>
<td>Hardware Induced</td>
</tr>
<tr>
<td>Simulation of Multiple Faults</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Simulation of Performance oriented Multiple failure</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Simulation of performance and variable Redundancies based on Importance and Criticality of Application</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Simulation of Pure Adaptive Strategies of Failure Detection</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Simulation of Automatic &amp; Self leaning Fault Tolerance</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Simulation of Transparent Recovery</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Generation of heterogeneous task in large amount</td>
<td>Fully</td>
<td>Partially</td>
<td>Partially</td>
</tr>
<tr>
<td>Checkpoint as a object</td>
<td>Supported</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Discrete Event Simulator</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Type of Framework for simulation</td>
<td>Multithreaded Process Oriented</td>
<td>Multithreaded, Process Oriented</td>
<td>Multithreaded, Process Oriented</td>
</tr>
<tr>
<td>Generic</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support dynamically and instantiate a set of users, resources, jobs and checkpoints</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Architecture</td>
<td>Layered</td>
<td>Layered</td>
<td>Layered</td>
</tr>
<tr>
<td>Use of Real World Fault Tolerance</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
6. CONCLUSION

XMONARC framework for dependability simulation and evaluation is capable to simulate multiple fault failures such as failure of nodes by hardware induced and software induced. The new trend in fault tolerance computing is performance oriented towards multiple fault simulation and is supported by proposed work compared to the simulation proposed by Adrian Boteanu and Ciprian Dobre. The self configuration and automatic fault tolerance can be simulated as well as can be evaluated by the proposed framework i.e. XMONARC.

7. REFERENCES


