Exploring the issues on Security and Fault Tolerance of Adaptive Distributive Systems

*Vinesh Jain**

Abstract

The extension of access control, authentication mechanism and cryptography has been well addressed in different research communities during recent times. Consequently, the fault tolerance has also been emerged as a prime parameter in designing different web enabled operating system and server management. These both contemplations of distributed system are appealing enough to implement the algorithm of fault tolerance and security issues.

Keywords: Adaptive Security Manager, Distributed System, Fault Tolerance.

Introduction

1.1 Overview

Recent researches have been accomplished in the broader paradigm of distributed system. Primarily the examples of distributed system have been categorized into Internet, Intranet and even in the form of mobile and ubiquitous computing [1] [2]. Several interesting phenomena have been evolved. Load balancing, enhancing of system throughput etc also became operational to improve the resource utilization and reduce the completion time of the process. The significant works [3] [4] exhibit the different forms of load balancing either be static and dynamic in distributed computing. The data in our system consists of a collection of items that we call active objects. An object could be a file say or a Java object. But each such logical object is implemented by a collection of physical copies called as replicas. The replicas are physical objects each stored at a single computer with data and behavior that are tied to some degree of consistency by the system operation. Therefore there must be well defined algorithms to tackle the issues of replicas and group communication especially under crash. Besides the group coordination under crisis, the general form of all distributed system also suffers serious security threats, a pattern for usage monitoring across the distributed system network and files. Since the fault tolerance and security maintenance is co-relational issue, and the sudden alterations could also prove the level of system adoption, so it would be appreciable, if any adaptation manager with a security metrics can be solicited. The application of software metrics has proven to be smart and penetrating technique for improving the quality of software design and implementation [5]. This proposal would like to incorporate these all issues and effectively proposes an innovative model of adaptive distributed system backed up by the Adaptation Manager, which significantly reciprocates as per the sudden alterations made in system modeling, file architecture, group communication etc.

Program Coordinator – IT, Institute of Management Studies, Dehradun

1.2 Preliminaries on Adaptive Distributed System

The following section describes the basics of conventional adaptive distributed system [6]. Adaptive software – that is, software that can change its behavior at runtime – has a number of potential benefits, ranging from the ability to respond rapidly to security threats to the opportunity to optimize performance as characteristics of the underlying execution environment change. The value is especially pronounced in distributed systems, which are arguably more prone to the type of situations that would benefit from an adaptive response. Unfortunately, implementing adaptive software is also more challenging in such systems for a variety of reasons, including the need to coordinate adaptation among multiple machines. A number of issues have to be addressed to construct such adaptive software, especially software that supports graceful adaptation in a distributed system. These include:

- Change detection: It involves detecting changes and determining if they are significant enough to warrant adaptation. It is generally not feasible to centralize detection into one component even on a single host, so any number of different components may be involved. In a distributed system, global agreement is also often needed to determine whether a change has occurred and an adaptation should be made.
- Adaptation policy: Involves deciding which layers and which components within a layer should make an adaptation once a change has been detected. There are typically multiple options involving various tradeoffs.
- Coordination: It involves doing an adaptation in a coordinated manner so that both component and system functionality are preserved while the adaptation is in progress. Two types of coordination may be needed *inter-component coordination* that coordinates adaptations across adaptive components and layers on a given host, and *inter-host coordination* that coordinates adaptations across hosts in a distributed systems.

1.3 Problem Statement

Mathematically, I would like to present a scenario of distributed system, which has to be made adaptive through the proposed model of algorithm and also to ensure the security of the distributed system. Considering the sequence of modifications to a file F in a volume that is replicated at 3 servers, S_1, S_2, and S_3, the volume storage group for F is { S_1, S_2, S_3 }. F is modified about the same time by two clients C_1, C_2. On account of network fault C_1 can access only S_1 and S_2 (C_1's available volume storage is $\{S_1, S_2\}$) and C_2 can access only S_3 (C_2's available volume storage is $\{S_3\}$).

The following observations are made:

- Initially the vector associated with the file F for all 3 servers are the same; let us consider it as [1, 1, 1].
- C_1 runs a process that opens F, modifies it and then closes it. The main server process of C_1 broadcasts an upgrade message to its available volume storage $\{S_1, S_2\}$, finally resulting a new version of F and the vector [2,2,1] at S_1 and S_2 but no change in S_3.
- Meanwhile C_2 runs two processes each of which opens F modifies it and then closes it. The main server process at C_2 broadcasts an update message to its available volume storage $\{S_3\}$ after each modification. Finally resulting a new version of F and the file vector [1, 1, 3] at S_3.
- At some time later, the network fault is repaired and C_2 makes a routine check to see whether the inaccessible members of volume storage group have become accessible and discover that S_1 and S_2 are now accessible. It modifies its available volume storage to $\{S_1, S_2, S_3\}$ for the volume containing F and requests the file vector for F from all members of the new available volume storage. When they arrive, C_2 discovers that

S_1 and S_2 each have file vectors [2, 2, 1] whereas S_3 has [1, 1, 3]. This represents a conflict requiring a manual intervention to bring F up to date in a manner that minimizes the loss of update information.

The above scenario clearly depicts the importance of smart adaptive manager in monitoring the distributed system. One major observation has been made out of it is the process lifetime which gives an indication about the longevity of the task on processor node irrespective of client and server. The presence of a longer process lifetime in a node doesn't allow it to share the task sets. This creates a disbalancing faulty situation where security of the system can be challenged. Certain new generation proposed algorithms are presented in this proposal encompassing the preemptive migration theory, which allows the transfer of the task to the lightly loaded node at lighter communication overhead thus makes the system self adaptable and secured.

The threshold load level (Lthi) of processor 1 in the non directed graph G (V,E) at t=0 is given as

$$Lthi = \phi$$

Here the average life time j, of a process on processor i is

$$\phi = 1/m \, \Sigma \, \tau_i$$

if,

$\tau_i > \phi_i^t$ then for preemptive migration,

$\Psi_i = 1$

where $\Psi_i = 1$ is preemptive transfer eligibility indicator which can be 1 or 0.

2 Implementation Model

High Level Description of Fault Tolerance and Adjustment

```
Step 1. Calculate Lthi
Step 2. Check for prolonging task t.
        //to check whether the process life time t is more than the average life time.
Step 3. refer the trace of fault in section 1.1
Step 4. import call_back_faulty();
Step 5. Calculate overhead
        If overhead observed < overhead max. then transfer the task and update call_back_faulty()
        Endif
        Else
        Continue loop on Step 1.
        Initialize a metric Alert (n x m)
        Check for sparse and strum sequence
        // Sparse indicates the faulty traces in immediate past
        and strum produces comparison.
        Record and update metric Alert for each process call.
        Exit all processes
Stop.
```

2.1 Metric Proposition

Let A be the Alert Matrix, assuming a real symmetric tri-diagonal matrix of size n x m given below

$$A = \begin{pmatrix} a_1 & b_2 & 0\ldots0 \\ b_2 & a_2 & b_3\ldots0 \\ \cdots & \cdots & \cdots \\ 0\ldots b_{n-1} & a_{n-1} & b_n \\ 0\ldots0 & b_n & a_n \end{pmatrix}$$

The particular method that has been adopted here to fabricate the security matrix is to generate a sequence called as strum sequence. This property states that number of sign changes in the successive members of the sequence (in the form of process and resources in distributed systems) is equal to the number eigen value of distributed components which are less than λ.

Therefore, the sequence will be like:

$f_0(\lambda)=1$

$f_1(\lambda)=a_1-\lambda$

$f_{i+1}(\lambda)=(a_{i+1}-\lambda)\, f_i(\lambda)-b^2_{i+1}$

$1<=i<=n-1$

Conclusion and Contribution of Research

Massive parallel distributed system is gaining popularity due to the high level scaling and reduced support file vector and access parameters. Similarly, the recent trend of server configuration and network intrusion detection also put forward the idea of innovative algorithm which takes care of process part in adaptive system, at the same time it covers up the intrusion if any occurs in the distributed resources. This proposal is an effort to implement the overview of this algorithm considering the distributed systems. The web component and web programs are also in the purview of this proposal.

References

1. Abadi et al. *Secure Web Tunneling,* in proceedings of 7th International www conference, pp 531-9, Elsevier In Computer Networks and ISDN Systems, Vol. 30, Nos. 1-7.

2. R.Blakley, *CORBA Security – An Introduction to Safe Computing with Objects.* Reading Mass: Addison-Wesley.

3. Thomas Kunz, *The influence of different workload descriptions on heuristic load balancing scheme,* IEEE Transaction on Software Engineering, Vol. 17, No. 07, July 1991.

4. M.H.Balter and A.B.Downey, *Exploiting process life time distribution for dynamic load balancing,* ACM Transaction on Computer System, Vol. 15, No. 03, August 1997, pp 253-255.

5. Hilda B. Klasky, *A Study of Software Metrics,* M.Sc. Dissertation Report submitted in The State University of New Jersy, May 2003.

6. Wen-Ke Chen, et al. *Constructing Adaptive Software in Distributed Systems.* Tech. Report, The Defense Advance Research Projects Agency under grant N66001-97-C-8518 and the National Science Foundation under grants ANI-9979438 and CDA-9500991.