

Dynasa: Adapting Grid Applications to Safety using Fault-Tolerant Methods

Xuanhua Shi
CGCL/SCTS, School Comput.
Huazhong Univ. Sci. & Tech.,
Wuhan, 430074, China
xhshi@hust.edu.cn

Jean-Louis Pizat
IRISA/INSA de Rennes,
Campus de Beaulieu,
35042 Rennes, France
jean-louis.pizat@irisa.fr

Eric Rodriguez
CEDRAT, 15 Chemin de
Malacher-Inovallée,
38246 MEYLAN, France
eric.rodriguez@cedrat.com

Hai Jin
CGCL/SCTS, School Comput.
Huazhong Univ. Sci. & Tech.,
Wuhan, 430074, China
hjin@hust.edu.cn

Hongbo Jiang
EIE Department
Huazhong Univ. Sci. & Tech.,
Wuhan, 430074, China
hongbojiang2004@gmail.com

ABSTRACT

Grid applications have been prone to encountering problems such as failures or malicious attacks during execution, due to their distributed and large-scale features. The application itself, however, has limited power to address these problems. This paper presents the design, and implementation of an adaptive framework - Dynasa, which strives to handle security problems using adaptive fault-tolerance (i.e., checkpointing and replication) during the execution of applications according to the status of the grid environments.

Categories and Subject Descriptors

D.4.5 [Reliability]: Fault-tolerance; D.4.6 [Security and Protection]: Invasive software

General Terms

Reliability, Security

Keywords

Adaptive, Fault tolerance, Safety

1. INTRODUCTION

Grids make possible high performance computing from the resources in one institute controlled by one group of users to resources over the Internet controlled by groups of users. On the downside, grids also pose many problems which make dependable grid computing a great challenge [5]. Since the virtual organization inherently is a dynamic environment, grid applications communicate with un-trusted resource or users. Although most grid infrastructures protect themselves against malicious attacks with portal systems, grid services are typically maintained behind network

firewalls with holes punched through firewalls in order to allow services to communicate with others [4]; These permanently opening ports potentially allows potential intruders to determine what services are listening on those ports, thus increasing the chance of the system getting attacked.

Most grid applications are HPC applications, which easily expose them to malicious attacks – one successful attack may lose days' or weeks' results. In view of the dynamic nature of the grid and the long running times of applications, we present an adaptive framework, *Dynasa*, that handles malicious attacks during runtime. Dynasa enables applications to execute in an adaptive way in order to handle the security problems using fault-tolerance methods. This feature is desirable for grid applications because grid users often have limited control over the computing environment, and system administrators cannot always meet all the requirements from a large number of users and applications. Dynasa is based on the Dynaco component [1], which is able to modify its behavior during runtime. Dynasa makes adaptive decisions according to changes of security levels, and the adaptive actions are based on fault-tolerance techniques (i.e., checkpointing and replication).

2. FRAMEWORK

The Dynasa framework is composed of two parts, the *safety sensor*, and the *adaptive controller*, as illustrated in Fig. 1. The adaptive controller is a component. It includes *Decider*, *Planner*, *Executor*, *Action* and *Service*, which are also components.

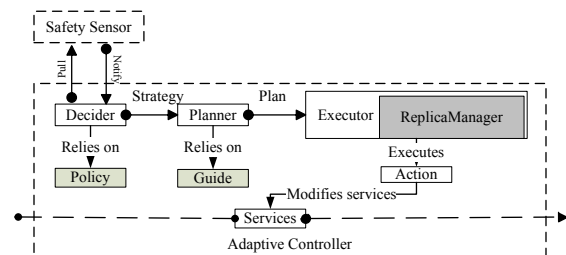


Figure 1: Dynasa's framework

sensor. The decider chooses strategies based on **Policy**, which describes component-specific information required to make the decisions. The decider has already been implemented in Dynaco; we add new policies to handle the security problems. The **Planner** is a program that makes adaptation plans, which instruct the component to adopt a strategy given by the decider, and the plan-making is based on the **Guide**. The planner has been implemented in Dynaco; we add new guides to target the security problems. The **Executor** is a virtual machine that interprets the plan provided by the planner by means of **Actions**. For example, checkpointing is an action. In addition, the executor handles the replica management semantic for secure applications; the **ReplicaManager** fulfills this function. The **Service** is the program of a grid application wrapped as a component. The **Safety Sensor** is a component deployed outside the adaptive controller, which provides the subscription and notification interface for the adaptive controller to obtain information about environmental status. These status include potential Denial of Service (DoS) attack incoming, malicious content attack incoming, node failure, just to name a few.

There are three specific types of security problems in grid applications described in [3]. Accordingly, we propose three protocols to address these problems. These protocols are implemented as policies and guides in Dynasa: 1) when there are some potential attacks presented, the replication manager should create a passive replica on a remote site; 2) when there are some DoS attacks on the running processes, the replication manager should create two replicas, and switch the primary replica site to the new replica site; 3) when there are some malicious content attacks which attack the running processes, the replication manager creates a new replica from the last correct checkpoints, switches the primary to the new replica, and creates another replica based on the new primary one. When the security level of the environment is increased, the adaptive controller will kill one passive replica in order to reduce the overhead of replication management. The desirable behavior of Dynasa is that the adaptive controller can make adaptive decisions during the execution of the applications according to any security events introduced above.

3. IMPLEMENTATION

Dynasa consists of several components, the safety sensor obtains the safety information with the snort tool (snort is a widely-used open-source tool for intrusion detection); the adaptive controller gives replication instruction to the replication manager. The replication manager consists of group communication and synchronization manager.

The adaptive controller provides a framework to input policies and guides, which allow the programmer fully control of the application. We add policies and guides to deal with security problems in grid environments. As shown in Fig. 2, replication manager is composed of two parts: the reliable group communication, which is based on Spread [6], and the replica synchronization manager. First, the reliable group communication provides three functions: joining/leaving groups, detecting the failures of replica, and reliable multicast. When one new replica is created, it joins the replication management group, and sends its states to the group of replicas. Based on the group message, the replication manager knows whether the replica is running in good states or not. If one replica fails, the replication manager will take some action to correct the faults, such as creating one new replica or making synchronization actions to change the status. Second, the synchronization manager is a service which synchronizes the status of every replica. For the MPICH2 application, the synchronization manager is a program which updates the latest checkpoints images from the primary

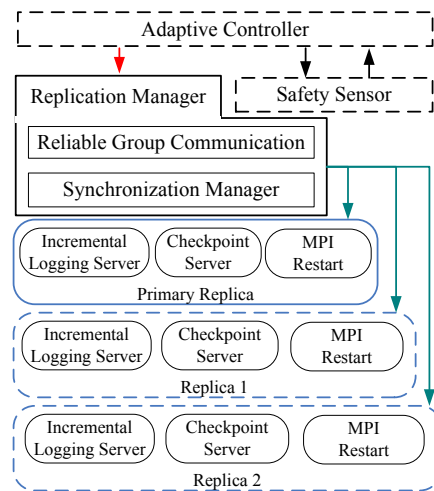


Figure 2: Implementation

replica site to other replica sites, and the copying of checkpoints files is performed in parallel to reduce transferring time.

We implement the creation of the replica of the HPC application based on the MPICH-V project. On every replica site, there are three services deployed: incremental logging service, checkpoint service and restart service. The incremental logging service logs the new state of the application and the nodes, which is used by the synchronization service. The checkpoint service is based on an abstract checkpointing mechanism, which provides a unified API to address system-level task checkpointing. We use a coordinated checkpoint protocol, called Pcl [2], to coordinate the checkpoint operation of every MPI process. To be able to restart from a specific checkpoint state of the MPI application, we modify the mpiexec and FTPM to support recovering from the crash of the whole MPI program (The modified MPICH-Pcl is available on the website <http://mpich-v.lri.fr/index.php>). Based on this restarting service, the application can be restarted on any replica site with checkpoint files.

Acknowledgments

This work is partly funded by the ARA SSIA SafeScale project of ANR, France and the National Science Foundation of China under Grant No. 60603058. Experiments presented in this paper were carried out using the Grid'5000 testbed. The authors would like to thank Guang Tan from INRIA for helpful comments and suggestions.

4. REFERENCES

- [1] J. Buisson, F. André, and J. Pazat, A Framework for Dynamic Adaptation of Parallel Components, *Proceedings of ParCo 2005*, Sep. 2005
- [2] C. Coti, T. Herault, P. Lemarinier, and etc, Blocking vs. Non-Blocking Coordinated Checkpointing for Large-Scale Fault Tolerant MPI, *Proceedings of Int. Conf. for High Performance Networking Computing, Networking, Storage and Analysis*, Nov. 2006
- [3] Y. Demchenko, L. Gommans, C. de Laat, and B. Oudenaarde, Web Services and Grid Security Vulnerabilities and Threats Analysis and Model, *Proceedings of the Sixth IEEE/ACM International Workshop on Grid Computing*, 2005
- [4] M. L. Green, S. M. Gallo, and R. Miller, Grid-Enabled Virtual Organization Based Dynamic Firewall, *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, 2004
- [5] S. Hwang, and C. Kesselman, Grid Workflow: a Flexible Failure Handling Framework for the Grid, *Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing*, 2003
- [6] Spread, <http://www.spread.org/>