Falut-Tolerant Heterogeneous Clusters in Distributed Systems *

4 0 － 4

Kenji Shima, Hiroaki Higaki, and Makoto Takizawa †
Tokyo Denki University ‡
e-mail{sima,hig,taki}@takilab.k.dendai.ac.jp

1 Introduction

Kinds of distributed applications like teleconferences and telemedicines have been developed and widely available now. In the applications, multiple autonomous application processes are cooperated to achieve some objectives by communicating with each other through communication networks.

The processes in the distributed system may suffer from faults, i.e. *Byzantine faults*[2] and *stop-faults*[4]. An approach towards making the system fault-tolerant is to replicate the processes in the system. There are two schemes to replicate the processes, i.e. *state machine*[3] and *primary-backup*[1] approaches. In the state machine approach, the replicas can continuously service in the presence of the faults while it takes time to recover from the fault of the primary replica in the primary-backup one. However, it is expensive to replicate all processes in the system using the state machine approach. The primary-backup approach implies less over-head than the state machine one. Kinds of applications have different requirements on reliability and availability to the system. For example, all the faults have to be invisible in some applications while the processes can be rebooted and restarted in other applications if some faults occur.

In this paper, each process is replicated into a collection of multiple *replicas*, which is named a *cluster*. We would like to propose a system composed of *dynamic* clusters in each of which the replicas are replicated in kinds of replication schemes so that the requirement from the application are satisfied. Moreover, the replication scheme of the cluster changes dynamically according to the change of the system environment and requirements.

In section 2, we present the replication schemes. In section 3, we present the system model. In section 4, we define a dynamic cluster. In section 5, we present inter-cluster communication.

2 Replication Schemes

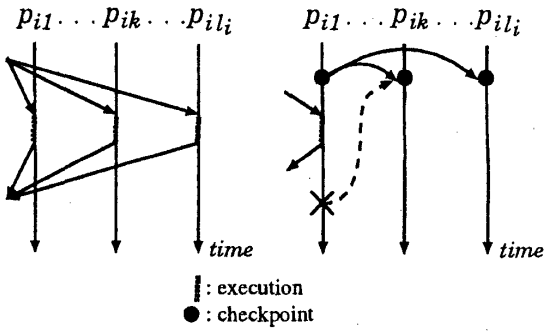
We would like to discuss how to replicate a process p_i into replicas p_{i1}, ..., p_{il_i} ($l_i \geq 1$). There are two kinds of approaches towards replicating p_i [1, 3] [Figure 1]:

(1) *state-machine approach*, and
(2) *primary-backup approach*.

In the state-machine approach (*active replication*) [3], every replica p_{ij} is modeled as a deterministic finite state machine. That is, every p_{ij} does the same computation by receiving and sending the same messages ($j = 1, \ldots, l_i$). If some replica p_{ij} is faulty, the computation of p_i is continued without stopping.

In the primary-backup approach (*passive replication*) [1], there is one *primary* replica p_{i1}. The other replicas $p_{i2}, \ldots, p_{il_i}$ are named *backup* ones. p_{i1} receives and sends messages and computes while no backup replica computes. p_{i1} takes a checkpoint and sends the local state information to all the backup ones. Then, every backup replica changes the local state. If p_{i1} is faulty, one of the backup replicas is selected and then starts to compute from the checkpoint taken most recently as a new primary replica.

The state machine approach implies more redundant processing and communication than the primary backup one because all the replicas do the same computation by sending and receiving the same messages. However, it requires less time-overhead for recovering from faults, that is, the computation can be taken over immediately by the other replicas if some replica is faulty. In addition, it tolerates the Byzantine faults by comparing messages sent by the replicas. On the other hand, the primary-backup approach implies less over-head but it requires more time-overhead for recovering from the faults because the backup replicas have to be rolled back to the checkpoint if the primary replica is faulty. Therefore, the primary-backup approach is adopted to the cluster whose process is required to be not expensive.



(1) State-machine approach (2) Primary-backup approach

Figure 1: replication schemes

3 System Model

A distributed system is composed of multiple processes interconnected by the communication network. A distributed application program is executed by the cooperation of multiple processes communicating with each other through the communication network. A collection of processes is referred to as *group* $G = \{p_1, \ldots, p_n\}$ ($n \geq 2$) [Figure.2]. G is composed of multiple autonomous processes. In order to tolerate process faults, each process is replicated to a collection of replicas, i.e. *cluster*. In order to tolerate the process faults, G is composed of multiple *clusters* where each cluster is a collection of more than one replica of each process. We assume that the faults occur independently, i.e. each replica is located in different computers. The replication scheme of the cluster depends on the environment in which the replicas are executed and on the application requirements.

*異種クラスタの混在する分散型システム
†島 健司、桧垣 博章、滝沢 誠
‡東京電機大学

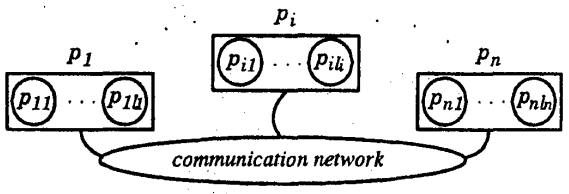


Figure 2: Group G

4 Dynamic Clusters

There are kinds of replication schemes in the clusters;

(1) passive replication by the primary-backup approach,

(2) active replication by the state machine approach, and

(3) dynamic replication, where the replicas are passively or actively replicated if the environment of the cluster is changed.

The replications (1) and (2) are discussed already in section 2. The passive replication tolerates the stop-faults but not the Byzantine faults of the replicas, and requires lower overhead because only the primary is computing. The active replication tolerates both stop-faults and Byzantine faults, and requires more computations and resources, i.e. more expensive because the computations are done by all the replicas.

In the distributed applications, the requirements from the applications and the environments of the system are changed. For example, a cluster which has been implemented on the less-reliable workstations is changed to be realized on the high-available workstations. An application which has required the system to be highly available is changed to require it to be realized cheaply even if it is less available. Thus, the distributed system has to be so *flexible* [?] that the changes could be absorbed automatically and autonomously in the system. In this paper, we would like to consider the following points on the application requirements:

(1) reliability,

(2) availability,

(3) cost, and

(3) performance, i.e. response time and throughput.

We would like to consider the following points of the system environments:

(1) faults, i.e. Byzantine faults and stop-fault, which the processes suffer from.

(2) performance of the processes e.g. processing speed.

In order to absorb the changes in the application requirements and the environments of the system, we would like to newly propose a *dynamic* replication of the cluster. A process p_i is replicated to be a cluster of replicas $p_{i1}, \ldots p_{i_{i_i}}$ in the same as the active and passive replications. In the dynamic replication, the replication scheme is changed among the active and passive replications according to the changes of the requirements and environments. A cluster in the dynamic replication is named *dynamic* one.

In the dynamic replication, there are two kinds of changes, i.e. the passive to active and the active to passive changes. For example, a cluster which has been actively replicated is changed to be passively replicated. Here, one replica continues the computation as the coordinator and the others stop the computation as the participants. In Table 1, the characteristics of the active and passive replications are summarized. In the former passive-to-active case, the cluster has to have enough processing capacity to execute multiple replicas and is required to tolerate the Byzantine faults. In the latter active-to-passive case, the cluster cannot be changed until all the replicas get the same state as the coordinator.

Replication	active	passive
Requirements		
availability	higher	lower
cost	higher	lower
Environment	Byzantine fault	stop-fault

Table 1: Dynamic replication

5 Concluding Remarks

This paper has presented a cluster of replicas which are dynamically replicated, i.o. *dynamic* cluster. In the dynamic cluster, the replication scheme is changed so that the requirements for the application and system environments are satisfied. By the dynamic replication, the distributed systems can support to the application with the *flexible* service.

References

[1] Budhiraja, N., Marzullo, K., Schneider, B. F., and Toueg, S., "The Primary-Backup Approach," *Distributed Computing Systems, ACM Press*, 1994, pp.199-221.

[2] Lamport, L., Shostak, R., and Pease, M., "The Byzantine Generals Problem," *ACM Trans. Programming Languages and Systems*, Vol.4, No.3, 1982, pp.382–401.

[3] Schneider, B. F., "Replication Management using the State-Machine Approach," *Distributed Computing Systems, ACM Press*, 1993, pp.169-197.

[4] Schneider, B. F., "Byzantine Generals in Action: Implementing Fail-Stop Processors," *ACM Tran. on Computing Systems*, Vol.2, No.2, 1993, pp.145-154.

[5] Shima, K., Higaki, H., and Takizawa, M., "Fault-Tolerant Causal Delivery in Group Communication," *Proc. of IEEE ICPADS'96*, 1996, pp.302-309.

[6] Shiratori, N., Sugawara, K., and Kinoshita, T., "Flexible Networks: Basic Concepts and Architecture," *IEICE Tran. Commun.*, Vol.E77-B, No.11, 1994, pp.1287-1293.