Simulation Modelling Practice and Theory 60 (2016) 90–107

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Transparent three-phase Byzantine fault tolerance for parallel and distributed simulations

Zengxiang Li [a,*], Wentong Cai [b], Stephen John Turner [b], Zheng Qin [a], Rick Siow Mong Goh [a]

[a] Institute of High Performance Computing, 1 Fusionopolis Way, Singapore 138632, Singapore
[b] Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 17 April 2015
Received in revised form 16 August 2015
Accepted 22 September 2015
Available online 26 October 2015

Keywords:
Parallel and distributed simulation
Byzantine fault tolerance
Replication
Checkpoint
Epidemic effect
Time synchronization

ABSTRACT

A parallel and distributed simulation (federation) is composed of a number of simulation components (federates). Since the federates may be developed by different participants and executed on different platforms, they are subject to Byzantine failures. Moreover, the failure may propagate in the federation, resulting in epidemic effect. In this article, a three-phase (i.e., detection, location, and recovery) Byzantine Fault Tolerance (BFT) mechanism is proposed based on a transparent middleware approach. The replication, checkpointing and message logging techniques are integrated in the mechanism for the purpose of enhancing simulation performance and reducing fault tolerance cost. In addition, mechanisms are provided to remove the epidemic effects of Byzantine failures. Our experiments have verified the correctness of the three-phase BFT mechanism and illustrated its high efficiency and good scalability. For some simulation executions, the BFT mechanism may even achieve performance enhancement and Byzantine fault tolerance simultaneously.

1. Introduction

A parallel and distributed simulation [1,2] is usually developed to study the problems of a complex system, e.g., city transportation system [3] and a global supply-chain system [4]. In order to study the complex system in desired fidelity or to encourage cooperations in the system, the parallel and distributed simulation (federation) is usually composed of a large number of simulation components (federates), which might be developed by different participants and executed on parallel and distributed platforms. Hence, the federates are subject to various failures. Previously, we have proposed mechanisms [5,6] to tolerate crash-stop failures of federates. In this article, we are focusing on Byzantine failures which are expected to be of increasing concern [7]. Byzantine failures refer to arbitrary behaviors, such as outputs and/or state updates, of a computer program, which are different from its assumed behavior based on the developed algorithms and received inputs. Byzantine failures might happen because of hardware faults or software bugs. According to Microsoft survey [8], the failure rate of computers including CPU, DRAM and disk subsystems is non-trivial. Besides computers, the simulation may also involve some embedded devices whose failure rate may increase due to the adverse environmental conditions (e.g., weather and temperature). Software bugs can be easily introduced by those participants with poor programming skills. What is worse, some participants might play as traitors and plot malicious attacks. For instance, some roadside severs in the distributed traffic simulation [3] may be attacked and then output wrong simulation results.

* Corresponding author.
E-mail addresses: liz@ihpc.a-star.edu.sg (Z. Li), aswtcai@ntu.edu.sg (W. Cai), steve@pmail.ntu.edu.sg (S.J. Turner), qinz@ihpc.a-star.edu.sg (Z. Qin), gohsm@ihpc.a-star.edu.sg (R.S.M. Goh).

http://dx.doi.org/10.1016/j.simpat.2015.09.012
1569-190X/

Since the Byzantine generals' problem [9] was proposed at the beginning of 1980s, a number of *Byzantine Fault Tolerance* (BFT) algorithms [10,11] have been proposed for distributed systems. Recently, some frameworks [12] have also been proposed to tolerate Byzantine failures in Cloud-computing environments. BFT algorithms [10–12] typically require $3f + 1$ replicas to tolerate f Byzantine replicas. Recently, BFT algorithms requiring just $2f + 1$ replicas are proposed based on a trusted distributed component [13] or a trusted queue service [14]. $2f + 1$ replicas are the minimum for BFT algorithms, since a majority of replica must be no-faulty for majority voting agreement protocol to detect faulty replicas. The replication-based fault tolerance mechanisms proposed in [15,16] enable to change the number of replicas dynamically, taking the tradeoff between degree of fault tolerance and performance into account. However, they work on crash-stop failures, rather than Byzantine failures.

It is challenging to support Byzantine fault tolerance in parallel and distributed simulations. This is because that the federates may communicate with each other very frequently and they need to coordinate their time advancements using complex synchronization approaches. In the case that federates encounter Byzantine failures, they are denoted as *faulty federates*. Otherwise, they are denoted as *non-faulty federates*. The executions of non-faulty federates are normal. In contrast, the execution of a faulty federate is arbitrary. It might update local execution state arbitrarily and produce arbitrary outputs, e.g., send arbitrary messages. Hence, the failure may quickly propagate in the federation execution due to the frequent interactions among federates. We refer to this phenomenon as epidemic effects caused by the Byzantine failure. The Byzantine failure may result in incorrect time synchronization and inconsistent state in the federation execution. Furthermore, it may also propagate in different manners due to the characteristics of different time synchronization approaches (Refer to Section 2). A Byzantine failure happens in a single federate will result in wrong simulation execution results. From this point of view, a federation execution may have an increased risk of failure with the increased federation scale. In order to support robust simulation executions, we propose a novel BFT mechanism with following advantages.

Firstly, the BFT mechanism is developed based on a transparent middleware approach. Similar to the BFT algorithm proposed in [13], each federate requires $2f + 1$ replicas, which are connected to the federation through a trusted middleware. The middleware is composed of a replication module and a BFT module. The former masks the presence of multiple replicas of the same federate without increasing federation scale; while the latter detects, locates and recovers Byzantine failures without interrupting the simulation execution.

Secondly, the BFT mechanism is executed in three phases in order to reduce the redundant executions of federate replicas. Only $f + 1$ replicas (Group1) of the same federate are executed during failure free executions. Byzantine failures are detected if Group1 replicas produce different outputs. If so, additional f replicas (Group2) are started using the checkpoint and message logging of Group1 replicas. By comparing the outputs of Group1 and Group2 replicas, the faulty replicas are identified. Finally, the Byzantine failures are recovered by excluding the faulty replicas from the federation execution.

Thirdly, the BFT mechanism is facilitated with epidemic effect removal mechanisms. Both *Barrier* and *No-Barrier* replication structures are supported by the replication module. Barrier replication prevents federates from propagating their Byzantine failures to other federates. But it may degrade simulation performance significantly, as the federate execution is always blocked by its slowest replica. In contrast, No-Barrier replication can enhance the simulation performance, as the federate execution is always pushed forward by its fastest replica. However, if the fastest replica is the faulty replica, an epidemic effect would happen. For different time synchronization approaches, the Byzantine failure may propagate in different manners. To solve this problem, different epidemic effect removal mechanisms are proposed respectively. In this way, the three-phase BFT mechanism using No-Barrier replication is expected to achieve both performance enhancement and Byzantine fault tolerance in the same simulation execution.

The three-phase BFT mechanism is a general solution to various Byzantine failures. The Byzantine failures caused by hardware faults can be tolerated by running replicas on different execution platforms. The Byzantine failures caused by software bugs can be tolerated by developing replicas in software diversity manner [17,18]. The Byzantine failure caused by traitors and malicious attacks can be tolerated by involving different participants. For instance, multiple roadside servers using different security protection mechanisms should be involved in the distributed traffic simulation scenario [3].

As an extension to our previous work [19], this article (i) revises the three-phase BFT mechanism and the epidemic effect removal mechanisms for conservative and optimistic synchronization approaches; (ii) provides the correctness proof of the epidemic effect removal mechanisms; and (iii) designs the experiments using a Bordless Fab simulation model to evaluate the efficiency and scalability of the proposed BFT mechanism.

This article is structured as follows: HLA-based simulation is introduced in Section 2. Section 3 illustrates the three-phase BFT mechanism using either Barrier or No-Barrier replication structure. Section 4 describes the epidemic effect removal mechanisms. Section 5 introduces Borderless Fab simulation model and reports the experimental results. The related work on simulation fault tolerance is discussed in Section 6. Finally, Section 7 concludes the article. The correctness proof of the epidemic effect removal mechanisms are provided in the Appendix A.

2. HLA-based simulations

The *High Level Architecture* (HLA), IEEE 1516 standard [20] provides a framework to build a parallel and distributed simulation (federation) by re-using and inter-operating a group of simulation components (federates). For generality, the three-phase BFT mechanism is developed for HLA-based simulations.

In an HLA-based simulation, federates participate in a federation execution through the underlying Runtime Infrastructure (RTI), as shown in Fig. 1. The communication between federates and the RTI is bidirectional. Federates invoke services