Journal of Network and Computer Applications 32 (2009) 423–432

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

A novel fault-tolerant execution model by using of mobile agents[☆]

Wenyu Qu[a,b,*], Masaru Kitsuregawa[b], Hong Shen[c], Zhiguang Shan[d]

[a] School of Computer Science and Technology, Dalian Maritime University, 1 Linghai Road, Ganjingzi, Dalian 116026, China
[b] Institute of Industrial Science, The University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
[c] School of Computer Science, University of Adelaide, SA 5005, Australia
[d] Department of Informatization Research, State Information Center, Beijing 100045, China

ARTICLE INFO

Article history:
Accepted 20 February 2008

Keywords:
Mobile agents
Migration time
Life expectancy
Population distribution

ABSTRACT

The exponential expansion of the Internet and the widespread popularity of the World Wide Web give a challenge to experts on reliable and secure system design, e.g., e-economy applications. New paradigms are on demand and mobile agent technology is one of the features. In this paper, we propose a fault-tolerance execution model by using of mobile agents, for the purpose of consistent and correct performance with a required function under stated conditions for a specified period of time. Failures are classified into two classes based on their intrinsic different effects on mobile agents. For each kind of failure, a specified handling method is adopted. The introduction of exceptional handling method allows performance improvements during mobile agents' execution. The behaviors of mobile agents are statistically analyzed through several key parameters, including the migration time from node to node, the life expectancy of mobile agents, and the population distribution of mobile agents, to evaluate the performance of our model. The analytical results give new theoretical insights to the fault-tolerant execution of mobile agents and show that our model outperforms the existing fault-tolerant models. Our model provides an effective way to improve the reliability of computer systems.

1. Introduction

With the far-reaching significance of the Internet and dramatic advances in computer technology, computers are no longer isolated computational machines. People communicate with the outer world through wireless networks, LANs, and the Internet. The exponential expansion of the Internet and the widespread popularity of the World Wide Web increase the difficulty of designing reliable and secure systems. New techniques are demanded to facilitate this trend and many researches have been conducted. For example, Li and Shen (2004a, b, 2005) and Li et al. (2005) paid efforts to improve network service efficiency for multimedia applications. As we know, any component (hardware or software) in a network may fail (malicious or not), thus preventing the system from consistently and correctly performing with a required function under stated conditions for a specified period of time. This paper focuses on crash failures (i.e., processes prematurely halt). Benign and malicious failures (e.e., Byzantine failures) (Pleisch and Schiper, 2003) are not discussed.

Fault tolerance is the property that enables a system (often computer-based) to continue operating properly in the event of the failure of (or one or more faults within) some of its components (http://en.wikipedia.org/wiki/Fault-tolerant_system, 2007). The basic characteristics of fault tolerance require no single point of failure or repair, fault isolation to the failing component, fault containment to prevent propagation of the failure, and availability of reversion modes. Existing fault-tolerant techniques can be roughly classified into two kinds: replication and checkpointing. Replication approach (Lyu and Wong, 2003) uses replicated servers to mask the failures. When one server is down, the computation still continues by using the results from other servers. However, this approach requires multiple servers at each stage, even when no failure occur; these redundant servers add an overhead to the communication between servers to guarantee the consistent execution, especially when they are widely separated. Checkpointing (Park et al., 2002), on the other hand, saves the intermediate execution into a stable storage so that the execution can be resumed in case of a failure. Although the communication cost for replication is avoided, the execution may be prevented even by a single failure. Besides, the occupied storage spaces are locked until the user requirement has been entirely completed at the destination node. Unlocking storages requires additional messages sent to all nodes of the itinerary.

In this paper, we propose a new fault-tolerant execution model based on a combination of the replication approach and the checkpointing approach by using of mobile agents. Our model reduced the redundant communications between nodes where there is no failure occurred. Theoretical analysis of migration

☆ This work is supported by the National Natural Science Foundation of China (90718030 and 60673054).

* Corresponding author at: School of Computer Science and Technology, Dalian Maritime University, 1 Linghai Road, Ganjingzi, Dalian 116026, China.
E-mail address: kequi@dlut.edu.cn (W. Qu).

1084-8045/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2008.02.008

time,[1] life expectancy,[2] and population distribution[3] of mobile agents are given for our model. Our approach exploits a new way to design cost-effective fault-tolerant agent-driven systems. Our analysis reveals new theoretical insights into the statistical behaviors of mobile agents and provides useful tools for effectively estimating the performance of agent-driven systems.

The main contributions of this paper are summarized as follows:

- We proposed a new fault-tolerant execution model which effectively combines available techniques. In our model, failures are classified into two classes based on their intrinsic different effects on mobile agents. For each kind of failure, an exceptional handling method is adopted. Our model greatly decreases network resources consumption and improves the overall network performance. It achieves better cost-effectiveness than existing models.
- We applied stochastic process to analyze the behaviors of mobile agents in fault-tolerant execution, which exploits a new approach to assessing the performance of agent-based systems. For the first time, to the best of our knowledge, the behaviors of mobile agents in networks that may contain faults are statistically analyzed in a quantitative way.
- We analyzed several key parameters which dominate the behaviors of mobile agents, including the migration time, the life expectancy, and the population distribution, in great theoretical depth. Our analysis provides a useful way of controlling mobile agents' behaviors by tuning relevant parameters according to various system characteristics.

The remainder of this paper is structured as follows. Section 2 reviews fault-tolerant approaches that utilize either replication or checkpointing. Section 3 presents our model. Section 4 analyzes the migration time, the life expectancy, and the population distribution. presents some discussion on the case that the network is reliable. Finally, Section 5 presents our experimental results, and Section 6 gives our conclusion remarks.

2. Related work

Mobile agent is one such paradigm for building distributed systems which has drawn a lot of attention in both academia and industry. The use of mobile agents can be found in various areas (Kotz and Gray, 1999), such as electronic commerce (He et al., 2003; Maes et al., 1999), network management (Bieszczad et al., 1998; Du et al., 2003), and information retrieval (Bergadano et al., 1999; Theilmann and Rothermel, 2000). As defined in Milojicic (1999), mobile agents are executing software entities that are capable of migrating from node to node in heterogeneous networks on behalf of network users. The key idea underlying mobile agents is to bring the computation to the data rather than the data to the computation (Schoder and Eymann, 2000). Lange and Oshima (1999) concluded that mobile agents can reduce the network load, overcome network latency, encapsulate protocols, execute asynchronously and autonomously, and dynamically adapt to changes. Although some of these strengths can be realized with combinations of many traditional distributed-computing techniques, no competing technique shares all of them (Gray et al., 2000). The merits have led a number of leading companies and research institutions to develop mobile agent systems. The existing systems include Ara, D'Agents, Aglets, Concordia, Gypsy, Mole, JatLite, Voyager and others (Paulino, 2002).

Mobile agents' ability to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems Lange and Oshima (1999). Many mobile agent-based fault-tolerant approaches have been proposed so far. A brief introduction to proposed approaches following the same classification of replication vs. checkpointing is given as follows.

In a replication scheme, an agent is replicated and migrated to several sites. The replicas must agree on one execution site for each stage and the redundant execution of other sites must be undone. For the agreement procedure, a distributed migration proposed in Vogler et al. (1997) injects an agent replica into stable storage upon arriving at an agent server. However, if an agent server crashes, the replica remains unavailable for an unknown time period. In Johansen et al. (1999), Rothermel and Strasser (1998) and Straßer and Rothermel (1998), a protocol was presented that provides the exactly once property in the migration of mobile agents. Every time an agent wants to migrate, it is replicated to a set of nodes. In every group of replicas there is one worker node, which is responsible for the execution of the agent. The other replica nodes receive a copy of the agent and act as observers of the worker node. When the worker fails, the observers will detect it and elect a new worker by running an election protocol. The elected worker will try to provide the same service or, if that is not possible, will execute an exception handling mechanism. In Assis Silva and Popescu-Zeletin (1998) a protocol was proposed which can be seen as a variation of that proposed in Rothermel and Strasser (1998). This scheme includes the distributed storage of recovery information and it used a three-phase commit protocol. However, the execution of a 3PC protocol is more expensive and introduces a higher latency. In Pleisch and Schiper (2000, 2001), the consensus protocol is used; among the replicas, one, called primary, is initially responsible for the execution. The priorities of the replicas are predetermined, hence, a replica takes over the execution, only when all of the higher priority replicas fail. To detect a possible failure, the time-out is used in Pals et al. (2000). When a primary does not respond within a certain time-out period, the first replica broadcasts the failure of the primary and starts up the agent execution. However, it is possible that the primary was too slow to respond within the time-out period, in which case, two agents may have performed the same execution stage. In Pleisch and Schiper (2003), a replication model is modeled as a sequence of agreement problems. This model ensures both non-blocking and exactly once properties, but it generates multiple replica agents and introduces a large amount of communication for each hop in the network, which will certainly consume a certain amount of network resources.

In the checkpointing scheme, the intermediate states of an agent are saved into a stable storage periodically, so that the agent can resume the execution from the most recent saved state after a crash (Elnozahy et al., 2002; Nicola, 1995). In Walsh et al. (1998), checkpointing is provided to deal with node failures, but a mobile agent in this system cannot proceed with its computation until the failed node recovers and restores the state of the mobile agent from persistent storage. In Dalmeijer et al. (1998), a checkpoint manager monitors all the agents inside a cluster of machines and responds to restart the agents when there is a node failure. In addition to communication cost, the checkpoint manager can also be a point of failure. In Johansen et al. (1999), a rear-guard agent is associated with each agent, to respond by launching a new agent when a failure causes the first agent to vanish. Again, along with the communication cost, the checkpoint manager is also a

[1] Migration time refers to the period that a mobile agent migrates from one node to another.

[2] Life expectancy refers to the average life-span of mobile agents.

[3] Population refers to the number of mobile agents running in the network.