Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 31 (2008) 2005–2015
computer communications
www.elsevier.com/locate/comcom

Design and analysis of an adaptive object replication algorithm in distributed network systems

Lin Wujuan [a,*], Bharadwaj Veeravalli [b]

[a] *Hitachi Asia Ltd., Singapore*
[b] *National University of Singapore, Singapore*

Received 13 July 2006; received in revised form 28 December 2007; accepted 6 January 2008
Available online 18 January 2008

Abstract

In this paper, we propose an adaptive object replication algorithm for distributed network systems, analyze its performance from both theoretical and experimental standpoints. We first present a mathematical cost model that considers all the costs associated with servicing a request, i.e., *I/O* cost, control-message transferring cost, and data-message transferring cost. Using this cost model, we develop an adaptive object replication algorithm, referred to as *Adaptive Distributed Request Window* (ADRW) algorithm. Our objective is to dynamically adjust the *allocation schemes* of objects based on the decision of ADRW algorithm, i.e., whether the system is *read-intensive* or *write-intensive*, so as to minimize the total servicing cost of the arriving requests. *Competitive analysis* is carried out to study the performance of ADRW algorithm theoretically. We then implement our proposed algorithm in a PC based network system. The experimental results convincingly demonstrate that ADRW algorithm is adaptive and is superior to several related algorithms in the literature in terms of the average request servicing cost.

Keywords: Object replication; Allocation scheme; Competitive analysis; I/O cost; Communication cost

1. Introduction

In a distributed network system, users at different nodes may issue transactions to access the objects in the system. Transferring an object from one node to another may be required by some application which will consume a varying network bandwidth. In turn, designing efficient and auto-adaptive object (data) dissemination and management schemes for applications always offers considerable challenges to the system designers. Generally, the requests issued by users are either *read* requests or *write* requests. A read request is serviced with a replica of the requested object, while a write request actually modifies the requested object. Particularly, in order to guarantee the *consistency* among multiple replicas of an object, every change (a *write* request) to an object must be transferred to all the other available replicas elsewhere (or in a *majority consensus* approach [26] for *weak consistency*). In other words, a write request for an object must be propagated to all the processors that have replicas of the object in their respective local memories. This will incur considerable communication cost.

In this paper, we consider three types of costs when servicing requests. The first one is the I/O cost, i.e., the cost of fetching an object from the local memory to the processor or saving an object from a processor to its local memory. The other two types of cost are due to communication in the underlying interconnection network, i.e., control-message transferring cost and data-message transferring cost. A control-message transfer is needed when a processor requests for an object which is not in its local memory, whereas a data-message transfer is just the transferring of an object between the processors via the interconnection network. The objective of this paper is to design an efficient

* Corresponding author. Tel.: +65 62312154.
E-mail addresses: wjlin@has.hitachi.com.sg (L. Wujuan), elebv@nus.edu.sg (B. Veeravalli).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.01.005

object replication algorithm to handle *on-line* requests arriving at the system with a minimum cost and maintain the consistency of multiple replicas of objects in the system. In a distributed network system, objects are usually replicated in several nodes for improving certain system performance metrics such as the *response time* of transactions, *bandwidth utilization*, *object availability*, and *system reliability* [7–9,13]. However, it should be noted that the system performance is very sensitive to the distribution of the replicas among the nodes. This is due to the fact that the cost of servicing a request associated with a local operation is different from the cost of servicing a request associated with a remote operation. When more replicas are allocated, the average cost of servicing a read request will be smaller, whereas the average cost of servicing a write request will be higher. Therefore, a crucial decision while designing an object replication algorithm lies in determining *how many replicas* of each object should be present in the network, and *where these replicas should be located*, often referred to as the object *allocation scheme* [15,16]. In other words, an object allocation scheme identifies the nodes at which the copies of the object are stored. Obviously, in a *read-intensive* network more replicas are beneficial whereas, in a *write-intensive* network fewer copies will be recommended.

The example in Fig. 1 further illustrates our motivation, with a system consisting of 8 nodes. In Fig. 1(a), Nodes 1, 2, and 3 read an object from Node A through Node B. In this case, if the system can generate a replica of the object in Node B, it will improve the request response time, and the network bandwidth utilization between Node A and B. After the replica in Node B is created, however, there is no more read requests from Nodes 1, 2, and 3, but only write requests from Nodes 4, 5, and 6 that want to update the object, as shown in Fig. 1(b). In this case, it would be better for the system to remove the replica in Node B (for the same reason to create replica in Node B), as long as the object reliability requirement (such as number of replicas) can be satisfied. Nevertheless, the request pattern in a real system is generally random and unpredictable. It is therefore necessary to design a system that can dynamically adjust the number and locations of object replicas, based on the current and history request information, so as to maximize the system performance.

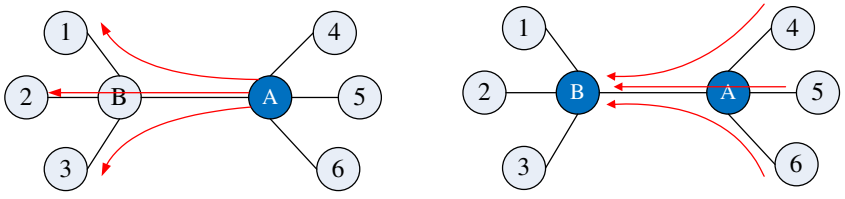
In this paper, we first propose a mathematical cost model that considers the cumulative cost of all the operations involved in servicing read or write requests, i.e., control-messages passing cost, data-messages passing cost, and I/O cost. Using this model, we design an efficient algorithm, referred to as *Adaptive Distributed Request Window* (ADRW) algorithm. The ADRW algorithm uses request windows to track the read/write on-line requests and dynamically adjust the object *allocation scheme*. Based on the decision of the request window mechanism, our objective is to minimize the total servicing cost of arriving read/write requests. The focus of this paper is to provide a rigorous theoretical framework and analysis of our proposed adaptive algorithm, from both the theoretical results derived and an actual PC-based experimental set-up. From theoretical standpoint, we use competitive analysis to quantify the performance of the ADRW algorithm. From the practical perspective, we carry out experiments to quantify the performance of ADRW algorithm under several influencing conditions, such as the request window size and the mean probability of read/write request in the system. The experimental results show that the ADRW algorithm is more adaptive and is superior in terms of the average cost of servicing a request. Further, based on the mean probability of read/write request, the experimental requests give more insights on designing object replication strategies for distributed network systems.

The rest of this paper is organized as follows. In Section 2, we present our ADRW algorithm, including the network model, cost model, and request window mechanism considered in this paper. In Section 3, using the competitive analysis, we evaluate the performance of ADRW algorithm from the theoretical standpoint. In Section 4, we implement the ADRW algorithm and study its performance under several influencing conditions by comparing with two other related algorithms in the literature. In Section 5, we introduce the related works and lastly, in Section 6, we conclude by summarizing this paper and discuss some possible future extensions.

2. ADRW Algorithm

2.1. Network model and notations

In this paper, we consider a distributed network system with n nodes, denoted as $p_1, p_2, \ldots, p_n$, interconnected via a message-passing communication network. Each node comprises a processor and a local memory (or disk drives). All the local memories are private and accessible only by their



(a) Nodes 1, 2 and 3 read object from Node A (b) Nodes 4, 5 and 6 update object in Nodes A and B

Fig. 1. Illustration of our motivation.