Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 22 (2013) 502 – 511

Procedia Computer Science

17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013

A Mass Data Update Method in Distributed Systems

Tsukasa Kudo[a,*], Yui Takeda[b], Masahiko Ishino[c], Kenji Saotome[d], Nobuhiro Kataoka[e]

[a]*Shizuoka Institute of Science and Technology, 2200-2, Toyosawa, Fukuroi-shi 437-8555, Japa*
[b]*Mitsubishi Electric Information Systems Corp., 325, Kamimachiya, Kamakura-shi 247-8520, Japan*
[c]*Fukui University of Technology, 3-6-1, Gakuen, Fukui City 910-8505, Japan*
[d]*Hosei University, 2-17-1 Fujimi, Chiyoda-ku, Tokyo 102-8160, Japan*
[e]*Interprise Laboratory, 1-6-2 Tateishi Fujisawa, Kanagawa 251-0872, Japan*

Abstract

Today, distributed business systems are used widely, and the consistency of their databases are maintained by the distributed transactions. On the other hand, a great deal of data is often updated in a lump-sum in the business systems. And, since the non-stop service has become general, it is necessary to perform this lump-sum update concurrently with the online transactions that service to users. So, some methods are utilized to avoid the influences on the online transactions, like the mini-batch that splits a lump-sum update into small transactions and executes them one after another. However, in the distributed systems, since it has to be executed by the distributed transactions, there is a problem on the efficiency to update a great deal of data by this method. For this problem, we propose to apply an update method to the distributed systems, which utilizes the records of data about the time to avoid conflicts between the lump-sum update and the online transactions. Moreover, through experiments using a prototype, we confirmed that it can update data more efficiently than the chain of small transactions even in the distributed systems.

Keywords: database; distributed system; batch processing; mini-batch; distributed transaction

1. Introduction

With the spread of the Internet and distributed processing technologies, the decentralization of the business systems in corporations is developing [7, 15, 13]. At present, it has also spread to the various Web services, such as net shops [1, 11]. For example, a corporation having branch offices in each region introduces a server to each office, and the system operations adapted to the business of the individual office is performed. In this case, each server has the database for each office business to reduce its communication cost, and the databases of other offices are accessed in order to perform comprehensive operations only when it is needed. In such a decentralized system environment (hereinafter, "distributed system"), some distributed processing technologies are introduced: the distributed transaction to control the updates across multiple databases and maintain their consistency; the

[*]Corresponding author. Tel.: +81-538-45-0201 ; fax: +81-538-45-0110.
E-mail address: kudo@cs.sist.ac.jp.

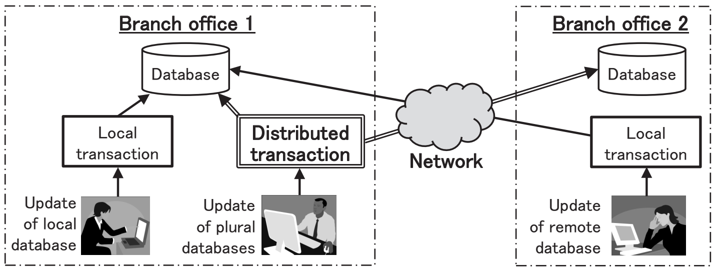

Fig. 1. An example of distributed system.

replication to improve the efficiency of the query at each office by placing replicated databases of the master database in each server [10].

On the other hand, often a great deal of data is updated by the lump-sum update. However, today, nonstop services have become to be used widely because of the spread of the internet business, the improvement of the convenience for users and so on. For example, in banking systems, ATM (Automatic Teller Machine) is provided for users as a nonstop service, and the entry data is reflected in the database by the transaction. That is, it is necessary to execute the lump-sum update processing without affecting the transactions of nonstop services [14]. So, for example, it is executed as the mini-batch that performs a great deal of update as a chain of small update transactions [2]. However, the distributed transaction is necessary for the distributed system. So, there is a problem that it is inefficient, because it is accompanied by access control over the network [13]. Similarly, the replication has a problem that it takes a long while to reflect the data, when a great deal of data is updated [5, 6].

Here, we had proposed a method to update a great deal of data in a lump-sum without stopping online services for centralized systems (hereinafter, "temporal update") [3]. This method uses an extended transaction time database [8] to avoid the conflict between the lump-sum update processing and the transactions that process the entries of users from online terminals (hereinafter, "online entry"). Concretely, in this method, the former updates the data at the future time, though the later updates the data at the present time. Therefore, since it can commit plural updated data collectively in the lump-sum update processing, we expect that this method can be executed efficiently even in the distributed systems.

Our goal in this paper is to examine the efficiency of this method to update a great deal of data in a lump-sum in the distributed systems. In this study we composed a prototype to evaluate it. And, our empirical results show that our method can execute a lump-sum update much more efficiently than the mini-batch even in the distributed systems. The rest of this paper is organized as follows. In section 2, we show the problem of the lump-sum update in the distributed systems, and the related study. In section 3, we show an implementation of the temporal update in a distributed system. In section 4, we show the evaluations of its efficiency based on experiments, and we show our considerations in section 5.

2. Conventional Update Methods and Related Study

2.1. Problem of conventional update method in distributed systems

In Fig. 1, we show an example of the business system that is constructed as a distributed system. The database of each branch office stores the data of the office, and users update the database of their office or another office depending on their business. Such an update is processed by the update transaction for the single database (hereinafter, "local transaction"). On the other hand, the processing to update the plural databases of the branch offices simultaneously has to be processed by the distributed transaction, which has the function such as two-phase commit, to maintain the integrity of these databases. For example, in the inventory management system constructed as the distributed system as shown in Fig. 1, there is the processing to move the stock from some office to the other one. In this case, a single distributed transaction executes both of the processing for this stock