A Reservation-Based Extended Transaction Protocol for Coordination of Web Services

Wenbing Zhao, Cleveland State University, USA
Firat Kart, University of California-Santa Barbara, USA
L. E. Moser, University of California-Santa Barbara, USA
P. M. Melliar-Smith, University of California-Santa Barbara, USA

ABSTRACT

Web services can be used to automate business activities that span multiple enterprises over the Internet. Such business activities require a coordination protocol to reach consistent results among the participants in the business activity. In the current state of the art, either classical distributed transactions or extended transactions with compensating transactions are used. However, classical distributed transactions lock data in the databases of different enterprises for unacceptable durations or involve repeated retries, and compensating transactions can lead to inconsistencies in the databases of the different enterprises. In this article, we describe a novel reservation protocol that can be used to coordinate the tasks of a business activity. Instead of resorting to compensating transactions, the reservation protocol employs an explicit reservation phase and an explicit confirmation and cancellation phase. We show how our reservation protocol maps to the Web services coordination specification, and describe our implementation of the reservation protocol. We compare the performance of the reservation protocol with that of the two-phase commit protocol and optimistic two-phase commit protocol. We also compare the probability of inconsistency for the reservation protocol with that for compensating transactions.

Keywords: business activity; continuous availability; extended transaction model; relaxed atomicity; reservation protocol; transaction processing systems; Web services

INTRODUCTION

Business activities often involve related tasks that are carried out over a long period of time in a loosely coupled distributed environment. Web services (Champion, Ferris, Newcomer, & Orchard, 2002) make it possible to automate business activities across multiple enterprises over the Internet. Such direct computer-to-computer interactions, without human supervision or intervention, provide speed improvements and cost reductions for distributed enterprise
computing. However, such enterprise applications must operate with a high degree of availability, reliability, scalability, and performance. Problems in the operation of the Web services can adversely affect the relationships between an enterprise and its customers, suppliers, and partners. Resolving inconsistencies among the databases of multiple enterprises is difficult, expensive, time consuming, and error prone, much more so than within a database of a single enterprise.

Many enterprise applications are programmed using the transaction-processing programming paradigm. A transaction is a set of operations in the application state that exhibits atomicity, consistency, isolation, and durability (ACID) properties. (Gray & Reuter, 1993)

For distributed transactions, commercial transaction processing systems have used the two-phase commit protocol, which involves a coordinator and multiple participants (Gray & Reuter, 1993). Although the two-phase commit protocol might work well for the coordination of operations within a single enterprise, the use of the protocol in distributed transactions that span multiple enterprises unavoidably involves the locking of a data record of one enterprise by another enterprise. Even for successful transactions, data records are locked for significant periods of time, adversely affecting performance. If the transaction coordinator fails, this locking period might be too long for an enterprise to tolerate.

Optimistic two-phase commit protocols (Herlihy, 1986; Kung & Robinson, 1981; Thomasian, 1998) do not lock database records but rather search for conflicts between operations on database records when transactions are committed, with one or more transactions being aborted when a conflict is detected. In principle, optimistic two-phase commit protocols achieve the same degree of transaction isolation as the two-phase commit protocol: Even though the record is not locked physically, it is locked logically. Optimistic two-phase commit protocols gamble with the fact that conflicts are rare; they achieve lower overheads when no conflicts occur at the expense of higher overheads when a conflict is detected.

Instead of implementing a business activity as a distributed transaction using the two-phase commit protocol or an optimistic two-phase commit protocol, a business activity might be implemented as an extended transaction, where one or more localized transactions are executed and committed individually at each site, as in the sagas strategy (Garcia-Molina & Salem, 1987). When a business activity must be rolled back, compensating transactions are applied to reverse the committed local transactions. Although useful in many cases, compensating transactions have their limitations. One problem is the possibility of cascading compensations that result from the relaxation of the isolation property; that is, before the compensating transaction is applied, other transactions might see and depend on the results of the committed transaction and, thus, must also be compensated. Identifying such transactions is difficult because there is no way to find them a priori. Furthermore, it might be difficult or impossible to compensate a committed transaction. For example, if an end-of-quarter audit transaction is executed immediately after a sales task is committed in a publicly traded company, the sales are included in the total revenue reported to the public. The completion of the audit transaction, followed by the compensation of the sales task, can result in inconsistencies. In general, the programming of compensating transactions is difficult and prone to error.

In this article, we present a novel reservation-based extended transaction protocol that avoids the use of compensating transactions while achieving atomicity and consistency similar to or better than other existing extended transaction protocols. Each task within a business activity is executed as two steps. The first step involves an explicit reservation of resources according to the business logic. For example, if the task involves reserving two seats out of 200 available seats on an airline flight, those two seats are explicitly reserved in a separate step. In the interests of the airline, a fee that is proportional to the duration of the reservation can...