WS-BusinessActivity Overview

by Simon Zambrovski

Table of contents

1. Transactions

Note:
This content is based on Arjuna Technology Report "Introducing WS-Coordination" (see Resources)

Distributed systems pose reliability problems not frequently encountered in centralized systems. A distributed system consisting of a number of computers connected by a network can be subject to independent failure of any of its components, such as the computers themselves, network links, operating systems, or individual applications. Decentralization allows parts of the system to fail while other parts remain functioning, which leads to the possibility of abnormal behavior of executing applications. Consider the case of a distributed system where the individual computers provide a selection of useful services, which can be utilized by an application. It is natural that an application that uses a collection of these services requires that they behave consistently, even in the presence of failures. A very simple consistency requirement is that of failure atomicity: the application either terminates normally, producing the intended results, or is aborted, producing no results at all. This failure atomicity property is supported by Atomic transactions, which have the following familiar ACID properties:

- Atomicity: The transaction completes successfully (commits) or if it fails (aborts) all of its effects are undone (rolled back)
- Consistency: Transactions produce consistent results and preserve application specific invariants
- Isolation: Intermediate states produced while a transaction is executing are not visible to other transactions. Furthermore transactions appear to execute serially, even if they are actually executed concurrently. This is typically achieved by locking resources for the duration of the transaction so that they cannot be acquired in a conflicting manner by another transaction
- Durability: The effects of a committed transaction are never lost (except by a catastrophic failure)

A transaction can be terminated in two ways: committed or aborted (rolled back). When a transaction is committed, all changes made within it are made durable (forced on to stable storage such as disk). When a transaction is aborted, all changes made during the lifetime of the transaction are undone. In addition it is possible to nest atomic transactions; where the effects of a nested action are provisional upon the commit/abort of the outermost (top-level) atomic transaction.

2. ACID Transactions may be too strong

Traditional transaction processing systems are sufficient to meet requirements if an application function can be represented as a single top-level transaction. However, this is frequently not the case. Top-level transactions are most suitably viewed as short-lived entities, performing stable state changes to the system; they are less well suited for structuring long-lived application functions that run for minutes, hours, days, or longer. Long-lived top-level transactions may reduce the concurrency in the system to an unacceptable level by holding on to resources (usually by locking) for a long time. Furthermore, if such a transaction aborts much valuable work already performed will be undone. ACID transactions have another criterion, that must be met of the participants. This is a criterion of highly trust. Given that the industry is moving towards loosely-coupled, coarse-grained interaction model supported by Web services, it has become clear that the semantics of traditional ACID transactions are unsuitable for Web scale deployment. Web services-based transactions differ from traditional transactions in that they execute over long periods between parties which share no trust relationship, they require commitments to the transaction to be negotiated at runtime, and isolation levels have to be relaxed.

3. WS-Coordination Foundations

An important aspect of WS-Transaction that differentiates it from traditional transaction protocols is that a synchronous request/response model is not assumed. This model derives from the fact that WS-Transaction is layered upon the WS-Coordination protocol whose own communication patterns are asynchronous by default.

Protocol View

4. Resources

- WS-Coordination specification
- Arjuna Technology Report : Introducing WS-BusinessActivity