MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Use of Transactions within a Reactive Microservices Environment

MASTER'S THESIS

Martin Štefanko

Brno, Fall 2017

MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Use of Transactions within a Reactive Microservices Environment

MASTER'S THESIS

Martin Štefanko

Brno, Fall 2017

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Martin Štefanko

Advisor: Bruno Rossi, PhD

Acknowledgement

thanks

Abstract

abstract

Keywords

Contents

1 Introduction

2 Transactions

This chapter introduces the basic concepts of transactions, their properties and common problems of the management of transactions across multiple nodes in the distributed systems.

2.1 Atomic transaction

An atomic transaction is an unit of processing that provides all-or-nothing property to the work that is conducted within its scope, also ensuring that shared resources are protected from multiple users [1]. It represents an unified and inseparable sequence of operations that are either all provided or none of them take effect.

From

Every transaction is associated with a transaction coordinator or transaction manager which is responsible for the control and supervision of the participants performing individual operations. Applications are commonly required only to contact the transaction manager about the start of the transaction. Transaction managers will be described in more detail in the following section.

The transaction can end in two forms: it can be either *commited* or *aborted*. The commit determines the successful outcome - all operations within the transaction have been performed and their results are permanently stored in a durable storage. The abort means that all performed operations have been undone and the system is in the same state as if the transaction have not been started.

2.2 ACID properties

A transaction can be viewed as a group of business logic statements with certain shared properties [2]. Generally considered properties are one or more of atomicity, consistency, isolation and durability. These four properties are often referenced as ACID properties [3] and they describe the major points important for the transaction concepts.

2.2.1 Atomicity

The transaction consists of a sequence of operations performed on different resources or by different participants. Atomicity means that all operation in the transaction are performed as if they were a single operation. When the transaction commits successfully all of its participants are also required to perform a valid commit. Conversely, if the transaction fails and is aborted all performed operations and effects are forced to be undone. This defines a possibility to abort at any point so that all changes done by the transaction will be reverted to the state before the transaction start.

Atomicity is generally achieved by the usage of the consensus multi-phase protocols. Standardized protocol is the two phase commit protocol which is used by the majority of modern transaction systems including Narayana. Consensus protocols together with other approaches for obtaining the atomicity property will be investigated in the following section.

2.2.2 Consistency

2.2.3 Isolation

2.2.4 Durability

2.3 Distributed transactions

A distributed transaction is the transaction performed in a distributed system. The distributed system consists of a number of independent devices connected through a communication network. Such systems are liable to the frequent failures of individual participants or communication channels between them.

The transaction manager can be implemented as a separate service or being placed with some participant or the client. **TODO**

2.4 Transaction manager

A transaction manager is a component liable for coordinating transactions in the sequential or parallel execution across one or more re-

sources. It provides proper and complete execution and it administers the comprehensive result of the transaction.

The main responsibilities of transaction manager are starting and ending (commit or abort) of the transaction, management of the transaction context, supervision of transactions scoped across multiple resources and the recovery from failure.

2.4.1 Local transaction manager

A local transaction manager or a resource manager is responsible for the coordination of transactions concerning only a single resource. Because of its scope it is often build in directly to the resource. The span of the resource is defined by its managing platform.

The resource manager is required to provide a support for the participation in the global transactions that span over several resources. This means that it is effectively capable to handle complete transaction processing to the different transaction manager.

2.5 Failure handling

CA CP theorem - CAP

3 Microservices architecture pattern

4 Communication patterns

4.1 Consensus protocols

4.1.1 2PC

4.1.2 3PC raft paxos

4.2 Event based protocols

Eventual consistency

4.2.1 CQRS

5 Transactions in distributed environment

5.1 Saga pattern

A saga, as described in the original publication [4], is a long lived transaction that can be written as a sequence of transactions that can be interleaved with other transactions. Each operation that is a part of the saga represents an unit of work that can be undone by the compensation action. The saga guarantees that either all operations complete successfully, or the corresponding compensation actions are run for all executed operations to cancel the partial processing.

5.1.1 Participants

6 Conclusion

Bibliography

[1] M. Little, J. Maron, and G. Pavlik, *Java transaction processing*. Prentice Hall, 2004.

[2] M. Musgrove, "Narayana + wildfly," 2015. [Online]. Available: https://developer.jboss.org/servlet/JiveServlet/download/53044-3-129391/jbug-brno-transactions.pdf

[3] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Computing Surveys*, vol. 15, no. 4, pp. 287–317, 1983.

[4] H. Garcia-Molina and K. Salem, "Sagas," *ACM SIGMOD Record*, vol. 16, no. 3, pp. 249–259, 1987.