The Saga Pattern in a Reactive Microservices Environment (temp title)

Martin Štefanko[a], Ondřej Chaloupka[b], Bruno Rossi[a]

[a] *Masaryk University, Faculty of Informatics, Brno, Czech Republic*
[b] *Red Hat, Brno, Czech Republic*

Abstract

Keywords: transactions, saga, compensating transactions, reactive, microservices, distributed system

1. Introduction

Transaction processing is widely recognized as a critical technology for modern applications [1]. Particularly in the distributed environment, the comprehension and the achievement of these requirements, based on the currently available technology stack, represents a considerably complex programming task.

Microservices architectural pattern defines a sophisticated development technique which separates the application domain into a set of isolated services that collaborate together to model various business concepts [2]. Due to their distributed character, microservices applications are often inclined to many issues associated with the failure processing and the isolated development model. To ensure that the system is able to function even in a degraded state, these applications commonly provide a design that guarantees certain quality properties which are defined in the Reactive Manifesto [3] as the reactive systems. The four recognized attributes of reactive microservices systems are responsiveness, resilience, elasticity and the asynchronous message passing.

These characteristics naturally extend to the application of the transaction processing in a reactive environment. As transactions may presumably span multiple services while still providing ACID (Atomicity, Consistency, Isolation, Durability) guarantees, it is required that all impacted participants reach a shared uniform consensus on the transaction result. This consensus is achieved through the utilization of consensus protocols represented conventionally by the Two-phase commit protocol (2PC). However, due to their commonly locking nature, these protocols may present difficulties with the achievement of properties reactive systems need to provide.

The saga pattern [4] represents an alternative approach to transaction processing applicable for long living transactions. In a long running transaction, traditional consensus protocols may hold locks on resources for long time periods which is not acceptable in a reactive environment. The saga addresses this problem by allowing participants to commit their intermediate states as local operations. Each operation is required to define a compensation action that can semantically undo the performed operation. The pattern guarantees that either all operations are completed successfully, or the compensation actions are executed for each performed operation to amend the partial processing.

2. Background

A transaction is a 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 a unified and inseparable sequence of operations that are either all performed or none of them is applied.

A transaction can also be viewed as a group of business logic statements that share certain common properties. Generally considered properties are one or more of atomicity, consistency, isolation, and durability. These four properties are often referenced as ACID properties [5] which describe the major points important for the transaction concepts.

3. Conclusion

References

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

[2] S. Newman, Building Microservices, 1st Edition, O'Reilly Media, Inc., 2015.

[3] J. Bonér, D. Farley, R. Kuhn, M. Thompson, Reactive manifesto (2018).
URL `https://www.reactivemanifesto.org/`

[4] H. Garcia-Molina, K. Salem, Sagas, ACM SIGMOD Record 16 (3) (1987) 249–259. `doi:10.1145/38714.38742`.

[5] T. Haerder, A. Reuter, Principles of transaction-oriented database recovery, ACM Computing Surveys 15 (4) (1983) 287–317. `doi:10.1145/289.291`.