MINISTERUL EDUCAŢIEI ŞI CERCETĂRII AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Admis la susținere
Șef de departament: conf. univ., dr. Fiodorov Ion

„____" _____ 2022

Analiza și cercetarea tranzacțiilor în cadrul arhitecturii distribuite de Microservicii

Research and analysis of transactions within a distributed microservices architecture

Teză de master

Student: **Denis Balan, gr. TI201M**
Conducător: **lect. univ., Catruc Mariana**

Chișinău 2022

REZUMAT

Cuvinte-cheie: tranzacții, consensus, paxos, teorema cap, saga, microservice, event bus, broker, sistem informatic, 2pc, 3pc, masstransit, nservicebus, opensleigh, orleans.

Problema care stă la baza acestei teze de analiză și cercetare constă în lipsa unui astfel de raport ce ar prezenta avantajele și dezavantajele, ar compara și juxtapune framework-urile existente pe piață, de asemenea ce ar arăta contrastul în vederea utilizării tranzacțiilor și conceptelor tranzacționale în proiectele enterprise din cadrul companiilor it în soluțiile lor.

Soluția presupune un raport de cercetare ce exemplifică fiecare framework în parte în vedere implementării în cadrul lui a pattern-urilor de proiectare tranzacționale prin intermediul microserviciilor. O valoare adăugată a acestei cercetări este modelarea a unor sisteme reale, monitorizarea lor, testarea performanței, compararea rezultatelor inter-implementări, alegerea unor framework-uri competitive pentru a corespunde cerințelor stipulate în cadrul sistemelor enterprise.

Funcționalul tranzacțiilor și implementării lui constă în simularea unui sistem real de cumpărare și procesare a comenzilor online, care implică în sine conexiunea mai multor microservicii ce delimitează contexte diferite cum ar fi catalogul, procesul de achiziționare, depozit, achitarea, expedierea, și altele.

Un instrument important în vederea implementării acestor concepte reprezintă saga pattern, propusă de Hector Garcia-Molin în 1987. Această cercetare să bazează pe ideile și conceptele expuse și o acordă la implementările reale la momentul de față de pe piață.

Lucrarea dată își propune ca scop cercetarea tranzacțiilor în cadrul arhitecturii distribuite, și analiza implementării lor în contextul framework-urilor ce se utilizează vast în sisteme enterprise, ulterior cu propunerea și sugerarea unor recomandări ample privind folosirea lor în diferite contexte și condiții a rulării sistemelor distribuite cu utilizarea saga tranzacții.

ABSTRACT

Keywords: transactions, consensus, paxos, cap theorem, saga, microservice, event bus, broker, computer system, 2pc, 3pc, masstransit, nservicebus, opensleigh, orleans.

The problem underlying this analysis and research thesis is the lack of such a report that would present the advantages and disadvantages, compare and juxtapose the existing frameworks on the market, as well as show the contrast in transactions and transactional concepts usage within enterprise projects in various it companies and their solutions.

The solution involves a research report that exemplifies each framework in view of implementing transactional design patterns through microservices. An added value of this research is the modelling of real systems, their monitoring, performance testing, comparison of results between implementations, selection of competitive frameworks to match the requirements stipulated in the enterprise systems.

The functionality of transactions and its implementation consists in simulating a real world, online purchasing and order processing system, which itself involves the connection of several microservices delimiting different contexts such as catalog, purchasing processes, warehouse, checkout, shipping, and others.

An important tool for implementing these concepts is the saga pattern, proposed by Hector Garcia-Molin in 1987. This research builds on the ideas and concepts presented by the paper, and relates them to actual implementations in the market today.

The given work aims at investigating transactions in distributed architecture, and analyzing their implementation in the context of frameworks that are widely used in enterprise systems, subsequently proposing and suggesting broad recommendations for their usage in different contexts and conditions of distributed systems that are running and widely using saga transactions.

CUPRINS

INTRODUCERE **8**

1 CERCETAREA DOMENIULUI **10**

1.1 Analiza cererii pe piață 10

1.2 Definirea problemei cercetate 11

1.3 Taxonomia și termenii utilizați 13

1.4 Obiectivele cercetării 13

1.5 Scopul analizei 14

1.6 Cerințe tehnice 15

1.7 Structura tezei de master 15

2 CONCEPTELE TRANZACȚIILOR **16**

2.1 Tranzacție 16

2.2 Proprietățile ACID 16

2.3 Nivelurile de izolare 17

2.3.1 Read uncommitted 18

2.3.2 Read committed 19

2.3.3 Repeatable read 19

2.3.4 Serializable 20

2.4 Managerul de tranzacții 21

2.4.1 Tipuri de manageri 21

2.4.2 Protocolul X/Open XA 22

2.5 Protocoale de consens 23

2.5.1 2PC 23

2.5.2 3PC 25

2.5.3 Paxos 26

2.5.4 Rezumat algoritmi de consens 28

3 PARADIGME ARHITECTURALE **29**

3.1 Model arhitectural 29

3.1.1 Arhitectura monolitică 29

3.1.2 Arhitectura microserviciilor 31

3.2 Principii ale microserviciilor 33

3.3 Provocări 36

3.3.1 Sistemele distribuite 36

3.3.2 Consistența eventuală 37

3.3.3 Teorema CAP 39

3.3.4 Operaționale 40
3.3.5 Factorul uman 41
4 MODELUL SAGA 42
4.1 Operațiuni 42
4.2 Acțiuni compensatorii 43
4.3 Tranzacția BASE 44
4.4 Componenta de execuție 44
4.5 Saga distribuite 45
5 IMPLEMENTAREA SAGA ȘI REZULTATELE SISTEMULUI 47
5.1 Scenariul cercetat 47
5.2 NServiceBus 48
5.3 MassTransit 49
5.4 OpenSleigh 50
5.5 Orleans Sagas 50
5.6 Tehnologii utilizate 51
5.6.1 Docker 51
5.6.2 MongoDb 52
5.6.3 RabbitMQ 53
5.7 Descrierea la nivel de cod 54
5.8 Testarea și sinteza implementărilor 59
CONCLUZII 64
BIBLIOGRAFIE 65
ANEXA A 67
ANEXA B 68
ANEXA C 69
ANEXA D 70

INTRODUCERE

În fiecare zi intrăm în conexiune cu noile tehnologii, însă de puține ori ne punem întrebarea, ce reprezintă ele. Un element de procesare a tranzacțiilor nu este altceva, decât un instrument, o unealtă care este folosită cu foarte multă inteligenta în rezolvarea multor probleme.

Această lucrare propune să facem o descriere aprofundată asupra procesării tranzacțiilor în cadrul microserviciilor, bazându-ne pe particularitățile specifice din domeniu, fiind alcătuită din scriere de software sau proiectare și analiză arhitecturală, întregul calculator constă din utilizarea unor unelte, și orchestrarea nivelelor mai joase, de către compartimentele de nivel mai sus.

Potrivit Market Research Future, se așteaptă ca piața arhitecturii microserviciilor să crească la aproximativ 33 de miliarde de dolari până în 2023, cu un CAGR (rata anuală combinată de creștere) de 17% între 2017 și 2023 respectiv [1].

Unele probleme sunt de nerezolvat fără o proiectare potrivită, de astfel în cadrul lucrării vom cerceta și analiza o suită de framework-uri pentru a rezolva problemele apărute în contextul elaborării și proiectării aplicațiilor software, din ramura procesării tranzacțiilor, dar care au fost aplicate în relativ puține contexte.

Operațiunile din cadrul sistemelor distribuite complexe necesită mai mulți pași care trebuie finalizați pentru ca operațiunea să aibă succes. Ne referim la aceste operațiuni ca tranzacții. Definiția tradițională a unei tranzacții este o singură unitate de lucru compusă din două sau mai multe operații [2].

Dacă oricare dintre aceste operații componente nu poate fi finalizată, întreaga tranzacție eșuează, lăsând datele în starea în care se afla înainte de inițierea tranzacției. Cu alte cuvinte, o tranzacție este o colecție de operații care fie reușesc, fie toate eșuează.

Obținerea acestui sistem consistent din cadrul unei unități de lucru este scopul unui monitor tradițional de procesare a tranzacțiilor, care este un software pe care îl vom cerceta și analiza în cadrul lucrării date.

Un exemplu de tranzacție simplă este transferul de fonduri dintr-un cont în altul în cadrul aceleiași bănci. Unitatea de lucru a tranzacției constă din două operații:

- debitarea dintr-un cont;
- creditarea unui alt cont.

În mod ideal, ambele operații se vor executa reușit (commit), dar și mai important este că, dacă o operație nu poate fi realizată, nici una nu va fi executată (adică ambele vor reveni la starea lor inițială).

Este în regulă dacă ambele credite și debituri nu se schimbă - aplicația care inițiază tranzacția poate încerca întotdeauna din nou. Dar este o problemă gravă dacă creditul este executat fără debitul asociat sau invers.

O tranzacție este înțeleasă ca o unitate de lucru efectuată în cadrul unui sistem de procesare în raport cu o unitatea de executare și tratată într-un mod coerent și fiabil, independent de alte tranzacții. O tranzacție este o unealtă pusă la îndemână unui programator prin care acesta poate exprima anumite proprietăți operaționale ale unor programe [3].

Tranzacțiile sunt utile pentru un tratament simplificat al erorilor: dacă survine o condiție de eroare programatorul poate executa o comandă de a putea reveni la starea inițială corectă, fără a trebui să refacă "manual" toate valorile modificate.

Procesarea tranzacțiilor este recunoscută pe scară largă ca fiind o tehnologie critică pentru aplicațiile moderne. Capacitatea de a grupa o secvență de operații într-o unitate logică de lucru reprezintă un concept de garanție comună care se bazează pe anumite dovezi de corectitudine. În special în mediul distribuit [4], înțelegerea și realizarea acestor cerințe, pe baza librăriilor tehnologice disponibile în prezent, reprezintă o sarcină de programare considerabil de complexă.

Un moment important în cadrul cercetării este analiza necesității utilizării tranzacțiilor, care se axează pe următoarele argumente: existența unei resurse partajate, accesate de mai multe utilizatori și procese în mod simultan, apariția erorilor la diverse niveluri (transport, procesare, stocare, etc..).

Dacă gestionarea are loc fără a ține cont de particularitățile tranzacționale, acest acces simultan la o resursă partajată de procesare va cauza probleme din mai multe considerente [5]:

- probleme rezultate din cauza **concurenței (concurrency)**;
- probleme cauzate de **defecțiuni (failures)**.

Lucrarea este structurată în două parți: teoretică și analitică. În partea teoretică este definit procesul de lucru a tranzacțiilor, propagarea și procesarea lor, inclusiv sistemele (Framework-uri) ce permit efectuarea acestor acțiuni. În partea analitică a fost realizată compararea sistemelor existente, analiza lor, și concluzionarea caracteristicilor specifice în vederea alegerii celui mai potrivit pachet existent.

BIBLIOGRAFIE

[1] „Microservices Architecture Market, By Deployment (Cloud, On-Premise),” Market Research Future, 01 02 2021. [Interactiv]. Available: https://www.marketresearchfuture.com/reports/microservices-architecture-market-3149. [Accesat 11 10 2021].

[2] D. Kaye, Loosely Coupled - The Missing Pieces of Web Services, Shroff Publishers & Distributors, 2003.

[3] M. Schofield, „What is a Transaction?,” Microsoft, 31 5 2018. [Interactiv]. Available: https://docs.microsoft.com/en-us/windows/win32/ktm/what-is-a-transaction. [Accesat 11 10 2021].

[4] W. Jia și Wanlei Zhou, Distributed Network Systems: From Concepts to Implementations, Springer, 1990.

[5] V. Gottfried și W. Gerhard, Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery, Morgan Kaufmann, 2002.

[6] C. Richardson, „Pattern: Microservice Architecture,” 20 02 2018. [Interactiv]. Available: https://microservices.io/patterns/microservices.html. [Accesat 11 10 2021].

[7] J. Grace și G. Peter, Reactive Systems Explained, OReilly, 2020.

[8] H. Garcia-Molina, „Sagas - Long lived transactions,” *ACM SIGMOD Record,* vol. 16, nr. 3, p. 259, 1987.

[9] Microsoft, „What is .NET?,” Microsoft, 2003. [Interactiv]. Available: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet. [Accesat 11 10 2021].

[10] MassTransit, „A free, open-source distributed application framework for .NET,” 2012. [Interactiv]. Available: http://masstransit-project.com/. [Accesat 12 10 2021].

[11] NServiceBus, „The most developer-friendly service bus for .NET,” 2015. [Interactiv]. Available: https://particular.net/nservicebus. [Accesat 12 10 2021].

[12] D. Guida, „Yet another Saga management library for .NET Core,” 2017. [Interactiv]. Available: https://www.opensleigh.net/. [Accesat 12 10 2021].

[13] Orleans, „Orleans is a cross-platform framework for building robust, scalable distributed applications,” 2018. [Interactiv]. Available: https://dotnet.github.io/orleans/. [Accesat 12 10 2021].

[14] D. Pine, „What's new in .NET 5,” Microsoft, 30 11 2020. [Interactiv]. Available: https://docs.microsoft.com/en-us/dotnet/core/dotnet-five. [Accesat 10 11 2021].

[15] Nuget.Org, „What is NuGet?,” Microsoft, 01 01 2010. [Interactiv]. Available: https://www.nuget.org/. [Accesat 10 11 2021].

[16] Docker, „Docker - Orientation and setup,” Docker, 1 1 2018. [Interactiv]. Available: https://docs.docker.com/get-started/. [Accesat 10 11 2021].

[17] B. Matthew și C. James, Professional .NET Transactions, Willey, 2002.

[18] H. Darwen, A Guide to the SQL standard : a users guide to the standard database language SQL, Addison Wesley, 1997.

[19] R. R, Database management systems, Osborne/McGraw-Hill, 2000.

[20] S. Loop, Designing Data-Intensive Applications, OReilly, 2015.

[21] N. Suri, „Resilient Atomic Commit Protocols for Mobile Environments,” *Researchgate,* vol. I, p. 80, 2006.

[22] B. A, „Three-phase commit protocol - 3PC,” 10 5 2017. [Interactiv]. Available: https://bohutskyi.com/3pc.html. [Accesat 3 11 2021].

[23] I. Keidar, „Increasing the Resilience of Distributed and Replicated,” The Hebrew University of Jerusalem, Jerusalim, 1998.