

Упоредна евалуација приступа 2PC, TCC и Saga за управљање дистрибуираним трансакцијама у микросервисној архитектури

Comparative evaluation of 2PC, TCC, and Saga approaches for distributed transaction management in microservice architecture

Иван Ђукановић, Факултет техничких наука, Нови Сад

Студијски програм – СОФТВЕРСКО ИНЖЕЊЕРСТВО И ИНФОРМАЦИОНЕ ТЕХНОЛОГИЈЕ

Кратак садржај – У овом раду приказана је имплементација и емпијска компаративна анализа три софтверска приступа за управљање дистрибуираним трансакцијама у микросервисној архитектури: Двофазно потврђивање глобалне трансакције (2PC), Try-Confirm-Cancel (TCC) и Saga. Приступи су развијени у екосистему Java коришћењем радног окружења Quarkus. Квантитативно испитивање обухватило је мјерење перформанси (пропусност и капацитет) под високим оптерећењем помоћу алата k6, као и евалуацију отпорности на отказе убризгавањем мрежних грешака преко алата Toxiproxy. Резултати показују супериорност образаца Saga и TCC у погледу скалабилности и пропусности у односу на блокирајући протокол 2PC, чиме дају јасне смјернице за избор оптималног трансакционог модела у дистрибуираним системима.

Кључне речи: дистрибуиране трансакције, микросервиси, 2PC, TCC, Saga.

Abstract – This paper presents the implementation and empirical comparative analysis of three software approaches for managing distributed transactions in microservice architectures: Two-Phase Commit (2PC), Try-Confirm-Cancel (TCC), and Saga. The approaches were developed within the Java ecosystem using the Quarkus framework. Quantitative evaluation included performance testing (throughput and latency) under high load using the k6 tool, as well as fault-tolerance testing through network fault injection via Toxiproxy. The results demonstrate the superiority of the Saga and TCC patterns regarding scalability and throughput compared to the blocking 2PC protocol, providing clear guidelines for selecting the optimal transaction model in distributed systems.

Keywords: distributed transactions, microservices, 2PC, TCC, Saga.

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Владимир Димитријески, ванр. проф.

1. УВОД

Развој савремених софтверских система донио је широку примјену дистрибуираних микросервисних архитектура као алтернативну опцију монолитном приступу. У оваквим системима, функционалности су подијељене на мање, независне јединице које међусобно комуницирају путем HTTP/REST или асинхроних порука [1]. Иако овакав приступ доноси предности у погледу скалабилности и флексибилности, он суштински мијења начин на који се управља подацима. Када се операције протежу кроз више сервиса од којих сваки управља сопственом базом података, традиционалне Atomicity, Consistency, Isolation, Durability (ACID) трансакције нису примјениве преко тих граница. Због тога је потребна примјена протокола за управљање дистрибуираним трансакцијама који балансирају између конзистентности, поузданости и перформанси.

У овом раду анализирана су три приступа управљању дистрибуираним трансакцијама: двофазно потврђивање глобалне трансакције (енгл. Two-Phase Commit – 2PC), Try-Confirm-Cancel (TCC) и Saga. Циљ рада је извршити упоредну анализу ова три протокола кроз практичну имплементацију у окружењу Quarkus [3]. Анализа обухвата мјерење кључних параметара система у контролисаним условима симулираних оптерећења и отказа, са посебним освртом на вријеме одзива, пропусну моћ и очување интегритета података. На тај начин, рад тежи да пружи емпијске податке о томе како се различити протоколи понашају у погледу перформанси и конзистентности у микросервисним системима.

2. ПРЕГЛЕД ТРАНСАКЦИЈСКИХ ПРОТОКОЛА И ИМПЛЕМЕНТАЦИЈЕ

У оквиру истраживања имплементирана су три тестна дистрибуирана система заснована на микросервисној архитектури, при чему сваки користи различите приступе управљању трансакцијама. Системи су реализовани у софтверском екосистему Java коришћењем радног окружења Quarkus.

2.1. Протокол двофазног потврђивања (енгл. *Two-phase commit 2PC*)

Протокол двофазног потврђивања представља централизовани атомски механизам који осигурава строгу конзистентност кроз двије фазе: гласање и извршавање [1]. У имплементираном моделу користи се трансакциони координатор заснован на компоненти *Narayana JTA*. Главна карактеристика овог приступа јесте закључавање ресурса на нивоу подтрансакција све док координатор не донесе коначну одлуку. Иако спречава појаву „прљавих читања”, ово закључавање директно утиче на пропусност система и повећава ризик од мртве петље (енгл. *deadlock*) у условима високог оптерећења.

2.2. Try-Confirm-Cancel (TCC)

Образац *TCC* је апликативни трансакциони образац намијењен за системе са високим захтјевима по питању перформанси, гдје се изолација постиже резервацијом ресурса умјесто условним закључавањем на нивоу базе података [4]. Имплементација се састоји из три експлицитне фазе

Фаза *Try* провјерава доступност ресурса и врши се њихова привремена резервација (нпр. уписивањем у посебно стање или табелу резервација).

Ако су сви микросервиси успјешно извршили фазу *Try*, Фаза *Confirm* покреће коначну потврду и ресурси се трајно заузимају.

У случају отказа Фаза *Cancel* извршава експлицитно отказивање било ког учесника чиме се резервисани ресурси ослобађају.

2.3. Saga

Образац *Saga* [2] редифинише глобалну трансакцију као низ локалних трансакција које се извршавају секвенцијално. Сваки корак одмах трајно уписује измјене у своју локалну базу података, чиме се у потпуности елиминише закључавање. Уколико нека од локалних трансакција откаже, покрећу се компензационе трансакције (енгл. *compensating transactions*) у обрнутом редослиједу како би поништио претходно успјешне кораке и вратио систем у конзистентно стање.

У раду је имплементиран шаблон оркестратор (енгл. *orchestrator*). Централна компонента (оркестратор) управља током извршавања и експлицитно позива наредне сервисе или покреће компензације у случају грешке. Комуникација између оркестратора и сервиса дешава се преко платформе за асинхроно слање порука попут платформе *Apache Kafka*. Овај модел прихвата концепт конвергентне конзистентности (енгл. *eventual consistency*), што значи да систем током извршавања може бити у привремено неусклађеном стању.

2.4. Верификација квалитета кода

Прије приступања експерименту мјерења перформанси, спроведено је мутационо тестирање помоћу алата *Pitest*. Циљ овог корака био је верификација ригорозности постављених интеграционих тестова убризгавањем вјештачких

отказа у код током рада апликације. Постигнути високи индекс ”уклоњених мутација” (енгл. *mutation score*) је потврдио да је логика покривености тестова исправна, те да су имплементирани дистрибуирани протоколи спремни за валидно експериментално испитивање.

3. ПОСТАВКА ЕКСПЕРИМЕНТА

Како би се извршила објективна квантитативна упоредна анализа имплементираних протокола, формирано је јединствено експериментално окружење унутар контролираних хардверских и софтверских оквира. Сва три приступа (*2PC*, *TCC* и *Saga*) су тестирана под идентичним условима симулираног оптерећења и мрежних аномалија.

3.1. Експериментално окружење

Све компоненте архитектуре покренуте су у контејнеризованом окружењу помоћу алата *Docker* и *Docker Compose* на рачунару са процесором *AMD Ryzen 7* (8 физичких језгара, 16 нити) и 16 *GB RAM* меморије, док је прикупљање података покренуто на другом рачунару, чиме је издвојено оптерећење прикупљања података за експеримент од саме покренуте инстанце апликације. За перзистенцију података сваког појединачног микросервиса су коришћене независне инстанце базе података *PostgreSQL*.

3.2. Тестирање перформанси

За генерисање виртуелног оптерећења и симулацију реалних корисничких захтјева коришћен је алат отвореног кода *k6*. Сценарио тестирања обухватао је постепено подизање броја истовремених виртуелних корисника (енгл. *Virtual Users – VUs*) са почетних 10 на максималних 150 у врху оптерећења, током дефинисаног временског интервала.

Примарне метрике које су праћене и биљежене током извршавања *k6* скрипти обухватају: укупан број успјешних и неуспјешних трансакција у секунди (енгл. *Throughput*), крајње кашњење извршавања трансакције (енгл. *End-to-End Latency*), укључујући средњу вредност, као и 90. и 95. перцентил и проценат успјешности *HTTP* одговора (успјешне трансакције са статусом 200 *OK* наспрот грешкама 4xx и 5xx).

3.3. Убризавање грешака

Да би се испитала отпорност на отказе и способност очувања исправности података у кризним сценаријима, између микросервиса и база података је интегрисан алат *Toxiproxy*. Спроведена су три карактеристична сценарија убризгавања мрежних аномалија (енгл. *toxic scenarios*).

Вјештачко кашњење (енгл. *Latency Toxic*) подразумијева увођење симулираног мрежног кашњења од 2000 милисекунди на комуникационим каналима ка бази или између сервиса, чиме се тестира понашање тајмаута (енгл. *timeout*) приступа.

Прекид везе (енгл. *Timeout/Close Toxic*) обухвата нагли и потпуни прекид активних *TCP* конекција у кључним

фазама трансакције (нпр. након фазе гласања у *2PC* или током извршавања средњег корака у *Saga*).

Ограничење пропусног опсега (енгл. *Bandwidth Toxic*) подразумијева редукцију брзине преноса података на мрежи како би се симулирала деградација инфраструктуре.

Кроз ове мрежне манипулације је директно верификовано да ли приступи успјешно покрећу механизме компензације, операције отказивања или логику враћања стања ради спријечавања појаве неконзистентних записа у базама података.

У Табели 1 се могу видјети резултати мутационих тестова.

Сервис	Покривеност линија кода	Покривеност мутација	Снага тестова
<i>2PC</i>	90% (288/319)	78% (87/111)	96% (87/91)
<i>Saga - inventory</i>	95% (251/265)	95% (74/78)	100% (74/74)
<i>Saga - order</i>	100% (152/152)	97% (31/32)	97% (31/32)
<i>Saga - payment</i>	96% (208/217)	97% (57/59)	98% (57/58)
<i>TCC - inventory</i>	95% (253/265)	96% (77/80)	100% (77/77)
<i>TCC - order</i>	99% (151/152)	97% (28/29)	97% (28/29)
<i>TCC - payment</i>	100% (205/205)	100% (53/53)	100% (53/53)

Табела 1: Приказ резултата мутационих тестова

4. РЕЗУЛТАТИ И АНАЛИЗА ПЕРФОРМАНСИ

У овом поглављу приказани су квантитативни резултати тестирања перформанси имплементираних трансакционих модела под оптерећењем од 150 истовремених виртуелних корисника (*VUs*). Кључни параметар поређења обухвата средњу вриједност кашњења (енгл. *end-to-end latency*) која говори колико су трансакције трајале у просјеку.

У Табели 2 се могу видјети добијене перформансе сва три приступа у милисекундама. Експеримент је покренут по три пута за сваки приступ, а у табели су издвојене итерације са најбољим перформансама за сваки приступ.

Приступ	Просјечно	Медијана	p(90)	p(95)	Максимално
<i>2PC</i>	1178,40	531	2543	2571	58247
<i>TCC</i>	1069,96	542	2069	2563	10626
<i>Saga</i>	819,13	522	1547	2045	5501

Табела 2: Перформансе трансакција

4.1. Анализа пропусне моћи и кашњења

На основу резултата, образац *Saga* остварује најбоље перформансе у свим тестираним сценаријима. Током три извршавања теста је остварено између 39929 и 42689 трансакција, што представља највећи број обрађених трансакција међу свим тестираним приступима, са средњим кашњењем од 819,13 милисекунди. Овакав резултат је директна последица асинхроне природе овог обрасца и одсуства било каквог дуготрајног закључавања дистрибуираних ресурса – свака локална трансакција одмах потврђује своје измјене на нивоу локалне базе података.

Образец *TCC* је кроз три инстанце експеримента одрадио између 34225 и 36354 трансакције са просјечним кашњењем од 1069,96 милисекунди које је умјерено веће у односу на образац *Saga*. Разлог лежи у чињеници да образац *TCC* захтијева експлицитну двофазну комуникацију на апликативном нивоу (фаза *Try* праћена фазом *Confirm*). Ипак, задржавањем

високе успјешности од 100%, образац *TCC* се потврђује као одличан избор за системе који захтијевају високу пропусност уз обавезну резервацију ресурса.

4.2. Архитектонски утицај на деградацију протокола *2PC*

Протокол двофазног потврђивања (*2PC*) је показао најслабије резултате. Током три извршавања остварено је између 25016 и 26801 трансакција, што је најмањи број обрађених трансакција од три тестирана приступа, док средње кашњење износи 1178,40 милисекунди. Поред тога, протокол *2PC* је једини приступ код кога је забиљежен пад успјешности извршавања трансакција на између 98,9% и 99,7%, док су остали имали 100,0% успјешно извршних трансакција.

Главни узрок оваквог понашања јесте песимистичко закључавање ресурса (енгл. *pessimistic locking*) које намеће *JTA* трансакциони координатор. Како сви учесници морају да држе закључане редове у базама података од почетка фазе гласања па све до коначног потврђивања, мрежни нивои захтјева почињу брзо да се акумулирају. Ово доводи до засићења базе података (исцрпљивање *Docker* контејнерских ресурса и базена конекција), као и појаве трансакционих тајмаута, што директно резултује падом перформанси и генерисањем грешака.

5. АНАЛИЗА ОТПОРНОСТИ НА ГРЕШКЕ

Поред перформанси под високим оптерећењем, кључан аспект евалуације протокола јесте њихова способност очувања конзистентности података у условима инфраструктурних отказа. Понашање система при убризгавању мрежних аномалија анализирано је кроз три карактеристична сценарија користећи алат *Toxiproxy*.

5.1. Утицај вјештачког кашњења

Увођење мрежног кашњења од 2000 милисекунди на каналима комуникације је показало суштинске архитектонске разлике у управљању тајмаутима.

Код протокола *2PC* песимистичко закључавање, комбиновано са кашњењем, довело је до брзог засићења базена конекција. Када би трансакција прекорачила дефинисани тајмаут, *JTA* координатор би успјешно покренуо операцију *rollback*, спријечавајући неконзистентност, док су перформансе система биле слабије.

Образец *TCC* При појави кашњења у фази *Try*, апликативни тајмаут је прекидао извршавање и активирао фазу *Cancel*. Сви резервисани ресурси су успјешно ослобођени, чиме је очувана исправност.

Код обрасца *Saga*, будући да кораци одмах уписују промене, рад се наставио све до истека тајмаута оркестратора. У случају прекорачења, оркестратор је успјешно иницирао асинхроне компензационе трансакције.

5.2. Ефекти прекида везе

Нагли прекид активних *TCP* конекција је представљао највећи изазов за исправност система.

Обрасци *Saga* и *TCC* pokazala su izuzetnu otpornost zahvaljujući implementiranim mehanizmima oporavka i idempotentnosti *HTTP* захтјева. Након успостављања везе, *Saga* оркестратор је успјешно довршио компензације, док је *TCC* координатор успјешно извршио фазе *Cancel*, враћајући систем у конзистентно стање.

Највећа рањивост протокола *2PC* уочена је у сценарију отказа везе након фазе гласања, а прије фазе потврде. Ако трансакциони координатор изгуби везу са микросервисом у овом критичном прозору, ресурси у бази података остају закључани све до ручне интервенције или истека дубоких системских тајмаута базе. Ово потврђује теоријски недостатак протокола *2PC* као блокирајућег протокола.

5.3. Збирна евалуација коректности

Сprovedени *Fault-Injection* експерименти су потврдили да апликативни трансакциони обрасци (*TCC* и *Saga*), уз правилну имплементацију идемпотентности, пружају знатно већу флексибилност и доступност у дистрибуираним срединама склоне грешкама. Док протокол *2PC* фаворизује строгу конзистентност по цијену блокирања цијелог система, образац *Saga* успјешно примјењује концепт конвергентне конзистентности, чинећи архитектуру отпорнијом на парцијалне мрежне отказе. Ово је прије свега последица структуралне рањивости протокола *2PC* — отказ централизованог координатора након фазе гласања оставља све учеснике блокираним на неодређено вријеме, што директно угрожава доступност система у цјелини.

6. ЗАКЉУЧАК И ПРЕПОРУКЕ

На основу спроведене емпиријске евалуације три кључна концепта за управљање дистрибуираним трансакцијама, дефинисане су јасне препоруке за развој софтвера у микросервисним архитектурама.

Образак *Saga* се препоручује за системе високе доступности и велике пропусне моћи (нпр. е-трговина, стриминг платформе) гдје је конвергентна конзистентност прихватљива. Елиминација закључавања омогућава линеарну скалабилност, али захтијева пажљиво пројектовање идемпотентних компензација.

Образак *TCC* је оптималан за домене гдје је неопходна моментална апликативна резервација ресурса (нпр. резервација авио-карата или хотелских соба), али гдје се жели избјећи песимистичко блокирање базе података какво намеће протокол *2PC*. Образак *TCC* нуди баланс између добрих перформанси и заштите од прекорачења капацитета.

Протокол *2PC* треба ограничити искључиво системе са ниским степеном конкурентности гдје је строга конзистентност апсолутни захтјев (нпр. основни банкарски трансфери). У дистрибуираним системима већег обима, његова блокирајућа природа представља озбиљан ризик по стабилност система.

Приликом тумачења резултата експеримента треба узети у обзир неколико ограничења: окружење

тестирања, иако распоређено на два умрежена рачунара, није еквивалентно продукцијском услед дијелених системских ресурса међу свим сервисима, што значи да оптерећење једног сервиса може утицати на перформансе осталих; симулација отказа путем алата *Toxiproxy* је извршена кроз контролисане, унапријед дефинисане сценарије, а не кроз насумичне или истовремене вишеструке отказе карактеристичне за стварна производна окружења; и коначно, имплементације приступа *Saga*, *2PC* и *TCC*, иако засноване на теоријским поставкама из литературе, представљају само једну од могућих конкретизација сваког приступа, па добијене резултате не треба тумачити као апсолутну и коначну мјеру перформанси самих приступа, већ као показатељ понашања конкретних имплементација у датим условима тестирања.

7. ЛИТЕРАТУРА

- [1] C. Richardson, *Microservices Patterns: With Examples in Java*. Shelter Island, NY, USA: Manning Publications, 2018.
- [2] K. Dürr, R. Lichtenthäler and G. Wirtz, "An Evaluation of Saga Pattern Implementation Technologies," in *Proceedings of the 13th European Workshop on Services and their Composition (ZEUS 2021)*, CEUR Workshop Proceedings, vol. 2839, pp. 74–82, 2021.
- [3] Quarkus, "Quarkus Guides – Transactions, Kafka and REST," 2025. [Online]. Available: <https://quarkus.io/guides/>
- [4] Apache Seata, "TCC Mode Documentation," 2024. [Online]. Available: <https://seata.apache.org/docs/user/mode/tcc/>

Кратка биографија:



Иван Ђукановић је рођен у Подгорици 7. новембра 2001. године. Након завршених основних студија, уписао се на мастер академске студије на Факултету техничких наука, студијски програм Софтверско инжењерство и информационе технологије.

Контакт:
ivan.djukanovic07@gmail.com