Предлог мастер рада: Компаративна анализа за дистрибуиране трансакције у микросервисној архитектури

Иван Ђукановић (R2 31/2024)

1. Увод

Савремени развој софтвера фаворизује микросервисну архитектуру због њене скалабилности и независности сервиса, али тај прелаз доноси фундаменталне изазове у домену конзистентности података. Проблем атомске обраде података у дистрибуираним системима је директно условљен *CAP (Consistency, Availability, Partition tolerance)* теоремом, која диктириа да се у условима мрежних партиција мора бирати између тренутне конзистентности и високе доступности система. У свијету гдје сваки сервис посједује изоловану базу података, класични *ACID (Atomicity, Consistency, Isolation, Durability)* принципи постају тешко одрживи, што захтјева примјену напредних трансакционих образаца.

Предмет истраживања овог рада је компаративна анализа кључних образаца: *Two-Phase Commit (2PC), Try-Confirm-Cancel (TCC)* и *Saga* кроз практичну имплементацију, док ће *Three-Phase Commit (3PC)* бити анализиран са теоријског аспекта као надоградња *2PC* модела. Посебан фокус ће бити стављен на њиховиу отпорност на мрежне прекиде и падове чворова, истражујући како сваки од њих балансира између перформанси и интегритета података у критичним сценаријима.

2. Теоријска основа и технолошки стек

Дистрибуирани трансакциони модели представљају скуп протокола који осигуравају да се сложене операције над више сервиса заврше или потпуним успјехом или безбједним повратком у претходно стање.

- ***2PC/3PC*** модели представљају „тешку“ артиљерију конзистентности; док се *2PC* ослања на централног координатора и стриктно закључавање ресурса, *3PC*, уз већ постојеће фазе које нуди *2PC* образац, уводи фазу прије потврде како би ублажио проблем блокирања ресурса у случају отказа координатора.
- ***TCC*** образац нуди модернију алтернативу кроз апликативну резервацциjу, чиме се избјегавају дуготрајна закључавања ресурса на нивоу базе.
- ***Saga*** модел, за разлику од претходних, не тежи тренутној, већ евентуалној конзистентности. Он трансакцију разбија на низ асинхроних локалних трансакција

повезаних путем порука. У случају неуспјеха било ког корака, Сага покреће компензационе трансакције које логички поништавају претходно извршене акције.

Технолошки оквир рада се ослања на Кваркус (*eng. Quarkus*) верзију 3.30.3 (Јава 21) који омогућава развој ресурсно ефикасних микросервиса. Кључне компоненте инфраструктуре су:

- ***PostgreSQL**:* За персистенцију података сваког сервиса уз подршку за *ACID* својства на локалном нивоу.
- ***MicroProfile LRA (Long Running Actions):*** Користи се за симулацију *2PC* протокола у дистрибуираном окружењу. Путем *LRA* координатора и анотација, систем реплицира фазе гласања (*Prepare)* и потврде (*Commit)* на апликативном нивоу, омогућавајући анализу без употребе тешких *XA* трансакција.
- ***Custom TCC Framework:*** Имплементација *TCC* образца је изведена без екстерних зависности, директно кроз пословну логику сервиса. Овај приступ служи као компаративни модел који демонстрира мануелно управљање резервисаним ресурсима и идемпотентношћу.
- ***Apache Kafka & SmallRye Messaging:*** Основа за асинхрони Сага образац. Омогућава лабаву спрегу сервиса и отпорност на отказе кроз комуникацију засновану на догађајима (*event-driven)* и механизме компензације.
- ***Nginx & Keycloak:*** Обезбјеђују сигорносни слој штитећи интерне трансакционе процесе и пружајући јединствену улазну тачку клијентима.

3. Архитектура система

За потребе истраживања биће имплементиран систем за електронску трговину који се састоји од три кључна домена. ***Order service*** дјелује као мозак операције и оркестратор трансакција. ***Inventory service*** је задужен за управљање стањем залиха и њихову резервацију, док ***Payment service*** симулира валидацију средстава и процесирање наплате.

Комуникација унутар система је хибридна. Синхрони модел (*REST)* се примјењује за *2PC* и *TCC,* гдје је неопходна директна повратна информација од учесника. Насупрот томе, асинхрони модел путем Кафке користи се за Сага образац, омогућавајући лабаву спрегу и рад заснован на догађајима. Сваки сервис посједује сопствену, изоловану инстанцу базе података, чиме се симулирају реални услови дистрибуираног окружења.

4. Имплементација у Кваркус окружењу

Имплементација ће максимално искористити напредне Кваркус екстензије. За асихрону комуникацију ће се користити *quarkus-messaging-kafka* док ће симулацију трансакционе стабилности *2PC* модела осигурати *quarkus-narayana-lra. TCC* модел ће бити реализован кроз *quarkus-rest-client,* омогућавајући високе перформансе чак и у сихнроним позивима. Посебан фокус ће бити стављен на идемпотентност потрошача, гдје ће се примјеном *Inbox/Outbox* обрасца осигурати да систем остане стабилан чак и у случају дуплираних порука унутар Кафке.

5. Методологија тестирања и прикупљања података

Како би се добили релевантни подаци, систем ће бити подвргнут ригорозним тестовима:

- ***Load* тестирање *(k6):*** Симулираће се постепени раст оптерећења како би се идентификовала тачка пуцања сваког обрасца.
- **Опсервабилност (*Jaeger & Prometheus):*** *Jaeger* ће визуелизовати путању сваке трансакције, док ће *Prometheus* пратити *CPU* и *RAM* ресурсе у реалном времену.
- **Инжењеринг хаоса (*Chaos Mesh):*** Ово је кључна фаза гдје ће се намјерно изазивати падови мрежних рута и база података како би се провјерило који образац најбоље чува интегритет података у „немогућим“ условима.

6. Анализа резултата и дискусија

Рад ће кулминирати детаљном анализом гдје ће се кроз табеларне и графичке приказе упоредити перформансе свих модела. Дискусија ће се фокусирати на тзв. трговину (*eng. trade-offs)* – колика је цијена сложености имплементације Сага образца у односу на перформансе и да ли је спорост *2PC* модела оправдана у системима гдје је конзистентност апсолутни приоритет.

7. Пријетње валидности

У циљу осигурања објективности и прецизности резултата истраживања, идентификоване су следеће пријетње валидности које су узете у обзир приликом планирања експеримента.

7.1. Interna validnost

- **Мрежна латенција у локалној инфраструктури:** Тестирање се спроводи у контролисаној локалној мрежи (*LAN*) између два физичка уређаја (клијент и сервер) како би се избјегао непредвидив утицај јавног интернета и сервиса попут тунеловања. Ипак, мрежни интерфејси и рутер уводе минималну латенцију која ће бити документована и третирана као константа у свим тестним сценаријима.
- **Ресурсно надметање:** Будући да се сви Докер (*eng. Docker*) контејнери (сервиси, базе, Кафка) покрећу на једној машини, постоји ризик да интензивно коришћење ресурса од стране једног сервиса утиче на перформансе другог. Ово ће се ублажити ограничавањем ресурса (*CPU/RAM*) по сваком контејнеру унутар Докер компоуз (*eng. Compose*) конфигурације.

7.2. Екстерна валидност

- **Специфичност окружења:** Резултати су директно везани за Кваркус верзију 3.30.3 и Јава 21. Иако ови подаци дају јасну сликку за овај екосистем, они не морају нужно бити идентични у другим технолошким стаковима (попут *.NET* или *Go* језика), што ограничава универзалну примјењивост закључка.

- **Скалабилност базе података:** Тестирање користи једну *PostgreSQL* инстанцу по сервису. У реалним продукционим системима са кластерованим базама података, кашњења у репликацији података би могла додатно утицати на вријеме трајања трансакције.

7.3. Валидност конструкта

- **Симулација отказа:** Инжењеринг хаоса ће се користити за симулацију прекида рада, али се фокусира на бинарне исходе (сервис ради/не ради). Постоји пријетња да суптилнији проблеми, попут постепеног успоравања мреже или парцијалних грешака у меморији, неће бити у потпуности обухваћени овим истраживањем.

7.4. Валидност закључка

- ***JVM Warm-up effect:*** Јава апликације показују слабије перформансе непосредно након покретања док *JIT* компајлер не оптимизује код. Да би се избјегла ова пријетња, сваки *k6* тест ће почети фазом „загријавања“ у трајању од 60 секунди прије почетка званичног мјерења метрика.
- **Статистичка значајност:** Резултати ће бити базирани на просјечним вриједностима *P95 и P99* перцентила добијених из више узастопних тестова, како би се елиминисали случајни екстремни резултати узроковани позадинским процесима оперативног система.

8. Закључак

У закључку ће бити изведена препорука за инжењерску праксу. На основу емпиријских доказа, биће дефинисано у којим сценаријима (на примјер: финансијске трансакције, ритејл системи са великим прометом) треба одабрати који од анализираних образаца.

9. Литература и референце

1. **Richardson, C. (2018).** *Microservices Patterns: With examples in Java*. Manning Publications.
2. **Quarkus Documentation (2024).** *Using LRA in Quarkus*. https://quarkus.io/guides/lra
3. **Quarkus Documentation (2024).** *Using Kafka in Quarkus*. https://quarkus.io/guides/kafka#receiving-messages-from-kafka