PARADIGMA, No.4, 2025, səh.142-146
https://doi.org/10.30546/30015.2025.01.4.1076

MİCROSERVİS MÜHİTİNDƏ MƏLUMATLARIN ÖTÜRÜLMƏSİNİN EFFEKTİV YOLLARI

T.A. Əsədli, A.X. Xanməmmədov
Azərbaycan Universiteti, Ceyhun Hacıbəyli 71, Bakı, Azərbaycan
e-mail: tural.asadli@student.au.edu.az

Xülasə: Mikroservis arxitekturasında məlumat ötürülməsi sistemin performansına, etibarlılığına və miqyaslanmasına birbaşa təsir göstərir. Servislər arasında effektiv kommunikasiya təmin etmək üçün sinxron və asinxron üsullar, mesaj növbələri, API Gateway həlləri tətbiq edilir. Bu yanaşma sistemin çevikliyini artırır və komponentlər arasında asılılığı azaldır.

Açar sözlər: Mikroservislər, mesaj növbələri, RESTful API, gRPC, event-driven arxitektura, data consistency.

Giriş

Microservis arxitekturası monolit sistemlərdən fərqli olaraq tətbiqin bir-birindən müstəqil işləyən kiçik xidmətlərə bölünməsi prinsipinə əsaslanır. Hər bir microservis öz verilənlər bazasına və biznes məntiqinə malik olur. Bu struktur çeviklik, miqyaslana bilmə və davamlılıq baxımından mühüm üstünlüklər yaradır. Lakin bu mühitdə əsas çağırışlardan biri xidmətlər arasında məlumatların düzgün, təhlükəsiz və səmərəli ötürülməsidir [1]. Xidmətlərarası düzgün kommunikasiya olmazsa, sistemdə gecikmələr, məlumat uyğunsuzluğu və təhlükəsizlik riskləri yarana bilər. Mikroservis arxitekturası müasir proqram təminatı sistemlərinin qurulmasında geniş tətbiq olunan yanaşmadır. Bu arxitekturada böyük tətbiqlər kiçik, müstəqil servislərə bölünür və hər bir servis öz məsuliyyət sahəsində fəaliyyət göstərir. Monolit sistemlərdən fərqli olaraq, mikroservislər ayrı-ayrı deploy edilə, miqyaslana və idarə edilə bilir. Lakin bu üstünlüklər servislər arasında məlumat mübadiləsinin düzgün təşkili məsələsini ortaya çıxarır. Mikroservis mühitində məlumatların effektiv ötürülməsi sistemin ümumi performansını, etibarlılığını və dayanıqlığını müəyyən edir. Servislər arasında düzgün kommunikasiya strategiyası seçilmədiyi təqdirdə gecikmələr, məlumat itkisi və sistem çökməsi kimi problemlər yarana bilər. Buna görə də mikroservis arxitekturasında işləyən təşkilatlar məlumat ötürülməsi üçün ən uyğun texnologiya və metodları seçməli, sistemin ehtiyaclarına uyğun həllər tətbiq etməlidirlər.

Məlumat ötürülməsi üsullarının əsas növləri

Mikroservis arxitekturasında məlumat mübadiləsi üçün müxtəlif yanaşmalar mövcuddur. Hər bir üsulun özünəməxsus üstünlükləri və tətbiq sahələri vardır. Təşkilatlar sistemin tələblərinə, performans göstəricilərinə və biznes məqsədlərinə uyğun olaraq bu üsullardan birini

və ya bir neçəsini eyni vaxtda istifadə edə bilərlər. Düzgün kommunikasiya modelinin seçilməsi sistemin gələcək inkişafına və miqyaslanmasına əhəmiyyətli təsir göstərir.

1. Sinxron Kommunikasiya (Synchronous Communication)

Sinxron kommunikasiya modelində bir servis digər servisə sorğu göndərir və cavab gözləyir. Bu zaman sorğu göndərən servis cavab alınana qədər öz işini davam etdirə bilmir. Sinxron yanaşma real vaxt tələb edən əməliyyatlar üçün uyğundur.

RESTful API

REST (Representational State Transfer) mikroservislər arasında ən geniş istifadə olunan protokoldur. HTTP əsaslı olan bu yanaşma standart metodlar (GET, POST, PUT, DELETE) vasitəsilə resurslarla əməliyyatlar aparır. RESTful API anlaşıqlı, sadə və müxtəlif platformalarda dəstəklənir. JSON formatında məlumat mübadiləsi həyata keçirilir və sərhədləri aydın müəyyən olunmuş endpoint-lər vasitəsilə servislər bir-biri ilə əlaqə qurur. Bu üsul kiçik və orta həcmli sorğular üçün effektivdir, lakin yüksək yüklü sistemlərdə performans problemləri yarana bilər.

gRPC

gRPC Google tərəfindən hazırlanmış yüksək performanslı RPC (Remote Procedure Call) çərçivəsidir. HTTP/2 protokolu üzərində işləyir və Protocol Buffers (protobuf) formatında məlumat serializasiyası həyata keçirir. REST API-dan fərqli olaraq, gRPC daha az həcmli məlumat paketləri göndərir və binary format istifadə etdiyi üçün daha sürətlidir. Mikroservislər arasında yüksək tezlikli məlumat mübadiləsi tələb edən sistemlər üçün gRPC ideal həlldir. Eyni zamanda bidirectional streaming dəstəyi və type safety təmin edir.

2. Asinxron Kommunikasiya (Asynchronous Communication)

Asinxron kommunikasiyada servis sorğu göndərdikdən sonra cavab gözləmir və öz işini davam etdirir. Bu model servislər arasında asılılığı azaldır və sistemin ümumi dayanıqlığını artırır. Mesaj əsaslı kommunikasiya vasitəsilə servislər bir-birindən müstəqil şəkildə fəaliyyət göstərə bilir.

Mesaj Növbələri (Message Queues)

Mesaj növbələri asinxron kommunikasiyanın əsas tərkib hissəsidir. RabbitMQ, Apache Kafka, Amazon SQS kimi platformalar servislərin mesajları növbələrə göndərməsinə və digər servislər tərəfindən emal edilməsinə imkan verir. Bu yanaşma sistemin yükünü bərabər paylamağa, mesajların itməməsinə və servislər arasında boş əlaqəni (loose coupling) təmin etməyə kömək edir. Məsələn, bir servis sifariş yaradıb mesajı növbəyə göndərə bilər, ödəniş servisi isə müstəqil olaraq həmin mesajı oxuyub emal edə bilər.

Event-Driven Arxitektura

Hadisə əsaslı arxitektura mikroservislər arasında məlumat mübadiləsinin daha mürəkkəb formasıdır. Bu modeldə servislər hadisələri (events) publish edir və digər servislər maraq göstərdikləri hadisələrə subscribe olurlar. Apache Kafka, AWS EventBridge, Azure Event Grid kimi platformalar event streaming və event sourcing üçün istifadə olunur. Hadisə əsaslı yanaşma sistemin miqyaslanmasını asanlaşdırır, real vaxt məlumat emalını təmin edir və servislər arasında tam müstəqillik yaradır.

3. API Gateway

API Gateway mikroservis arxitekturasında mərkəzi giriş nöqtəsi rolunu oynayır. Bütün xarici sorğular əvvəlcə API Gateway-ə daxil olur, sonra uyğun servislərə yönləndirilir. Bu həll authentication, rate limiting, request routing, load balancing və monitoring kimi funksiyaları mərkəzləşdirərək sistemin idarəolunmasını sadələşdirir. Kong, AWS API Gateway, Azure API Management kimi platformalar bu məqsədlə istifadə olunur. API Gateway həmçinin müxtəlif servislərdən gələn məlumatları birləşdirərək (aggregation) klienta vahid cavab qaytara bilir.

4. Service Mesh

Service Mesh mikroservislər arasında şəbəkə səviyyəsində idarəetməni təmin edən infrastruktur layıdır. Istio, Linkerd, Consul kimi həllər servislərin bir-biri ilə necə əlaqə qurduğunu, trafikin necə yönləndirildiyini, təhlükəsizliyin necə təmin edildiyini idarə edir. Service Mesh load balancing, circuit breaking, retry logic, distributed tracing, mTLS (mutual TLS) şifrələməsi kimi xüsusiyyətləri avtomatik həyata keçirir. Bu yanaşma development komandalarını şəbəkə səviyyəsindəki mürəkkəbliyə qapılmaqdan azad edir və servislər arasında etibarlı kommunikasiya təmin edir.

5. Database Per Service Pattern

Mikroservis arxitekturasının əsas prinsiplərindən biri hər servisin öz məlumat bazasına sahib olmasıdır. Bu yanaşma servislər arasında tam müstəqillik yaradır və hər bir servis öz məlumatlarını istədiyi texnologiya ilə saxlaya bilər (polyglot persistence). Lakin bu model servislər arasında məlumat ardıcıllığını (data consistency) təmin etmək məsələsini ortaya çıxarır. Bu problemin həlli üçün Saga Pattern, Event Sourcing və CQRS (Command Query Responsibility Segregation) kimi yanaşmalar tətbiq edilir.

6. Data Consistency və Distributed Transactions

Mikroservis mühitində məlumat ardıcıllığını təmin etmək ən mürəkkəb məsələlərdən biridir. Monolit sistemlərdə ACID (Atomicity, Consistency, Isolation, Durability) tranzaksiyalar asanlıqla idarə olunurdu, lakin paylanmış sistemlərdə bu mümkün deyil. Bunun əvəzinə eventual consistency modeli tətbiq edilir. Saga Pattern uzunmüddətli tranzaksiyaları kiçik addımlara bölərək hər addımın kompensasiya mexanizmini təmin edir. Two-Phase

Commit (2PC) protokolu isə bütün servislərə tranzaksiya əməliyyatını commit etmək və ya rollback etmək barədə siqnal göndərir.

7. Caching Strategiyaları

Mikroservislər arasında məlumat mübadiləsini optimallaşdırmaq üçün caching mexanizmləri mühüm rol oynayır. Redis, Memcached kimi in-memory cache həlləri tez-tez istifadə olunan məlumatları yaddaşda saxlayaraq sorğu sürətini əhəmiyyətli dərəcədə artırır. Cache-aside, write-through, write-behind kimi strategiyalar məlumatların necə və nə vaxt cache-ə yazılacağını müəyyən edir. Distributed caching sistemləri isə birdən çox serverin yaddaşını birləşdirərək böyük həcmli məlumatların sürətli şəkildə əldə edilməsini təmin edir.

8. Circuit Breaker Pattern

Mikroservis arxitekturasında bir servisin çökməsi digər servislərə zəncir reaksiyası yarada bilər. Circuit Breaker pattern bu problemin qarşısını alır. Əgər bir servis müəyyən sayda uğursuz sorğudan sonra cavab vermirsə, Circuit Breaker "açılır" və həmin servisə sorğular müvəqqəti dayandırılır. Bu zaman fallback mexanizmi işə düşür və ya keşdən məlumat qaytarılır. Müəyyən müddətdən sonra Circuit Breaker yenidən cəhd edir və əgər servis bərpa olunubsa, normal iş rejiminə qayıdır. Resilience4j, Hystrix kimi kitabxanalar bu pattern-i həyata keçirir.

9. Monitoring və Distributed Tracing

Mikroservislər arasında məlumat axınını izləmək üçün monitoring və distributed tracing alətləri vacibdir. Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana) sistemin performansını real vaxtda izləməyə imkan verir. Jaeger, Zipkin kimi distributed tracing platformaları isə sorğunun bütün servislər arasında keçdiyi yolu vizuallaşdırır və darboğazları müəyyənləşdirməyə kömək edir. Bu alətlər latency, error rate, throughput kimi metrikləri qeydə alır və sistem administratorlarına operativ qərar vermək imkanı yaradır.

10. Komandanın hazırlığı və best practices

Mikroservis arxitekturasında işləyən development komandaları məlumat ötürülməsi üsulları, kommunikasiya pattern-ləri və distributed sistemlərin özünəməxsus çətinlikləri ilə tanış olmalıdırlar. API versiyalaşdırması, backward compatibility, idempotency, timeout və retry strategiyaları kimi mövzularda bilik əldə etməlidirlər. Komanda üzvləri containerization (Docker), orchestration (Kubernetes), CI/CD pipeline qurulması və infrastructure as code (Terraform, Ansible) kimi texnologiyalara yiyələnməlidirlər. Mütəmadi olaraq kod review, architecture review və performans testləri həyata keçirilməlidir.

Nəticə

Mikroservis arxitekturasında məlumatların effektiv ötürülməsi sistemin uğurlu fəaliyyətinin əsasını təşkil edir. Sinxron və asinxron kommunikasiya modelləri, mesaj növbələri, event-driven arxitektura, API Gateway, Service Mesh kimi yanaşmalar sistemin tələblərinə uyğun olaraq seçilməli və tətbiq edilməlidir. Məlumat ardıcıllığı, distributed tracing, caching strategiyaları və circuit breaker pattern-i mikroservis mühitinin kompleks məsələlərini həll etməyə kömək edir. Düzgün texnologiya seçimi və komandanın bu sahədə bilikli olması sistemin performansını, miqyaslanmasını və etibarlılığını artırır. Mikroservis arxitekturasının üstünlüklərindən tam istifadə etmək üçün məlumat ötürülməsi strategiyaları diqqətlə planlaşdırılmalı və sistemin inkişafı ilə paralel olaraq təkmilləşdirilməlidir.

Ədəbiyyat

1. Newman S., (2021), Building Microservices: Designing Fine-Grained Systems (2nd Edition), O'Reilly Media, p.344.
2. Richardson C., (2018), Microservices Patterns: With Examples in Java, Manning Publications, p.416.
3. Fowler M., Lewis J., (2014), Microservices: A Definition of This New Architectural Term: [Elektron resurs] / – 25 mart, 2014. URL: https://martinfowler.com/articles/microservices.html
4. Indrasiri K., Siriwardena P., (2021), Microservices Communication Patterns. In: Microservices for the Enterprise, Apress, Berkeley, CA, p. 245.

EFFECTIVE WAYS OF DATA TRANSFER IN MICROSERVICE ENVIRONMENT

T.A. Asadli, A.Kh. Khanmammadov
Azerbaijan University, Jeyhun Hajibeyli 71, Baku, Azerbaijan

Abstract: Data transfer in microservice architecture directly impacts system performance, reliability, and scalability. Synchronous and asynchronous methods, message queues, and API Gateway solutions are implemented to ensure effective communication between services. This approach enhances system flexibility and reduces dependencies between components.

Keywords: Microservices, message queues, RESTful API, gRPC, event-driven architecture, data consistency.