Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/25BE33Tomic

KOMPARATIVNA ANALIZA PERFORMANSI SISTEMA APACHE KAFKA I RABBITMQ

COMPARATIVE PERFORMANCE ANALYSIS OF APACHE KAFKA AND RABBITMQ SYSTEMS

Nikolina Tomić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu će biti izvršena komparativna analiza Apache Kafke, moćne platforme za obradu i upravljanje tokovima podataka u realnom vremenu sa RabbitMq-om, koji sa svojom pouzdanom i skalabilnom infrastrukturom za razmenu poruka igra ključnu ulogu u olakšavanju efikasne komunikacije između aplikaciju u kompleksnim sistemima. Rad uključuje i opis implementacije oba ova alata na projektima napisanim u programskom jeziku Java i izvršenog eksperimenta na kom je bazirana analiza.*

Ključne reči: *Distribuirani sistemi, razmena poruka, pub-sub, Apache Kafka, RabbitMq*

Abstract – *This work will present a comparative analysis of Apache Kafka, a powerful platform for processing and managing data streams in real time, with RabbitMq, which with its reliable and scalable messaging infrastructure plays a key role in facilitating efficient communication between applications in complex systems. The paper includes a description of the implementation of both of these tools on projects written in the Java programming language and the performed experiment on which the analysis is based.*

Keywords: *Distributed systems, messaging, pub-sub, Apache Kafka, RabbitMq*

1. UVOD

Velike količine novca i intelektualne energije uložene su u tradicionalne sisteme za upravljanje podacima. Ovo upravljanje podacima svodilo se na skladištenje podataka na fajl sistemima i u bazama podataka. Vremenom sve veći problem postaje da se sve spoji u jednu celinu tako da funkcioniše u realnom vremenu [1]. Ovo je problem upravljanja podacima u pokretu. Ovaj problem rešavaju sistemi za razmenu poruka, kao što su Apache Kafka i RabbitMq.

2.1 Pub-sub

Publish subscribe je takođe arhitekturalni obrazac koji omogućava asinhronu komunikaciju između različitih komponenti ili servisa u aplikaciji.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dušan Gajić, vanr. prof.

U ovom modelu, publisheri/pošaljioci generišu poruke ili događaje i šalju ih bez direktnog znanja o tome ko će ih primiti. Subscriber-i se registruju za primanje određenih vrsta poruka (jedne ili više) i reaguju na njih kada stignu. Ovaj obrazac se često koristi u distribuiranim sistemima, IoT uređajima i aplikacijama koje zahtevaju efikasnu i skalabilnu komunikaciju [2].

2.2 Broker zasnovan na logovima

Broker zasnovan na logovima je softverski sistem gde producer poruke šalje poruku koju broker smešta u log, dok consumer čita poruke sekvencijalno iz loga [3].

3. APACHE KAFKA

Apache Kafka je distribuirani sistem za razmenu poruka zasnovan na pub-sub mehanizmu koji poseduje broker zasnovan na logovima. Služi za procesiranje strimova, podataka u realnom vremenu, kao i za metrike i logovanje. Jedinica podataka unutar Kafke je poruka. Sve poruke su organizovane u teme. Producer smešta poruku u temu, dok se consumeri pretplaćuju na teme i čitaju poruke iz njih [1].

4. RABBITMQ

RabbitMq je softver za razmenu poruka koji se bazira na pub-sub modelu. Koristi se za komunikaciju između visokopropusnih aplikacija i kada se vrši složeno rutiranje do consumera [4].

5. POREĐENJE APACHE KAFKA I RABBITMQ

I Apache Kafka i RabbitMq su sistemi redova poruka koji se koriste u obradi tokova podataka. Oba ova alata pristupaju razmeni poruka iz potpuno različitih uglova.

5.1 Arhitektura

Kafka se oslanja na distribuirani model podataka. Podaci su organizovani u teme, a svaka tema je podeljena u particije. Ovo omogućava horizontalno skaliranje i paralelnu obradu podataka. Takođe Kafka klaster se obično sastoji od više brokera , što doprinosi visokoj dostupnosti i otpornosti na kvarove.

RabbitMq koristi centralizovaniji model obrade podataka. Poruke se šalju na exchage, a zatim se usmeravaju prema redovima na osnovu određenih pravila rutiranja. RabbitMq je tipično organizovan u formi jednog

centralizovanog servera, što ga čini manje složenim u odnosu na Kafku.

5.2 Performanse

I Kafka i RabbitMq nude prenos poruka visokih performansi. Međutim, Kafka je ipak poznata po svojoj niskoj latenciji i sposobnosti za obradu velikih količina poruka u realnom vremenu i samim tim nadmašuje RabbitMq u kapacitetu prenosa poruka.

Kafka može da šalje milione poruka u sekundi jer koristi sekvencijalni disk I/O da omogući razmenu poruka velike propusnosti.

RabbitMq takođe nudi dobre performanse ali latencija može biti nešto veća, posebno u situacijama sa velikim opterećenjem. I RabbitMq se može konfigurisati da šalje milione poruka u sekundi, ali je za to potrebno više brokera. U proseku, performanse su mu hiljade poruka u sekundi.

5.3 Brisanje poruka

Kafka dodaje poruke u log datoteku, koja ostaje dok ne istekne retention period (koji se može konfigurisati). Zbog ovoga, consumeri mogu ponovo obraditi poruke u bilo kom trenutku u predviđenom roku.

RabbitMq broker usmerava poruku do odredišnog reda. Consumer kada pročita poruku šalje potvrdni odgovor brokeru, koji zatim briše poruku iz reda.

5.4 Vrste podataka koji se koriste

Kafka najbolje funkcioniše sa operativnim podacima kao što su procesne operacije i aktivnosti sistema, dok je RabbitMq najbolji za transakcijske podatke, kao što su zahtevi korisnika.

5.5 Preuzimanje poruka

Kafka broker služi samo za slanje poruka u red. Produceri objavljuju poruke u red i nisu svesni preuzimanja poruke od strane consumera. Na consumer strani se nalazi sva logika. Aktivni su, prate i čitaju informacije.

Kod RabbitMq se svo ovo rutiranje vrši na brokeru. Producer prati da li je poruka stigla do željenog consumera. Consumeri imaju pasivnu ulogu i samo čekaju da broker stavi poruku u red [5].

5.6 Tok podataka

Kafka koristi neograničen tok podataka, dok RabbitMq koristi ograničeni tok podataka.

5.7 Prioritet poruke

RabbitMq brokeri dozvoljavaju dodavanje poruka u prioritetni red. Umesto slanja i čitanja redovnih poruka po principu “first in first out“, broker obrađuje poruke većeg prioriteta pre ostalih.

Za razliku od RabbitMq, Kafka ne podržava prioritetne redove i sve poruke tretira isto.

5.8 Redosled poruka

Kafka koristi teme i particije za stavljanje poruka u red. Kada producer pošalje poruku, one idu u određenu temu i particiju. Pošto Kafka ne podržava direktnu razmenu između producera i consumera, consumeri izvlače poruke sa particije drugačijim redosledom.

RabbitMq šalje i stavlja poruke u red određenim redosledom. Osim ako se poruka višeg prioriteta ne stavi u red, consumeri primaju poruke onim redosledom kojim su one poslate.

5.9 Skalabilnost

RabbitMq može proširiti svoj kapacitet za rukovanje porukama i horizontalno i vertikalno. Uvek se može dodeliti više računarsih resursa RabbitMq serveru da bi se povećala efikasnost razmene poruka.

Isto tako, Kafkina arhitektura dozvoljava dodavanje više particija određenoj temi kako bi se opterećenje poruka ravnomerno raspodelilo.

5.10 Tolerancija na greške

I Kafka i RabbitMq su robusne arhitekture otporne na sistemski otkaz.

Može se grupisati više RabbitMq brokera u klastere i rasporediti se na različite servere. RabbitMq takođe replicira poruke u redu na distribuiranim čvorovima. Ovo omogućava sistemu da se oporavi od kvara koji utiče na bilo koji server.

Kao i RabbitMq, Kafka deli sličnu mogućnost obnavljanja i redudantnost tako što hostuje Kafka klastere na različitim serverima. Svaki klaster se sastoji od replika log datoteke koje se mogu oporaviti u slučaju kvara.

5.11 Logika rutiranja

Kod RabbitMq mogu postojati dva tipa događaja koji se nalaze u istom redu i ne mora se naglašavati consumeru koju poruku prima u svojoj aplikaciji. Ovo odlučuje logika rutiranja koja je dodata u samom RabbitMq brokeru. Napravi se red za svakog consumera. Ukoliko aplikacija treba da obrađuje dva različita događaja, red se kofiguriše da koristi oba ova događaja za tu aplikaciju. Ovako se razdvoji logika rutiranja unutar platforme i ona ne mora da se nalazi u consumeru.

Dok se kod Kafke mora unapred dobro isplanirati koliko se pravi particija u zavisnosti od broja različitih događaja koji se dodaju u red. Sva logika se nalazi u consumeru. Kod Kafke će svaki događaj imati svoju particiju. Ukoliko consumer želi da obrađuje dva događaja on mora da se pretplati na dve particije da bi dobijao ove poruke.

Još jedan način bi mogao biti da se napravi jedna particija sa dva različita događaja, da se na tu particiju pretplate dva consumera: 1 koji želi da prima oba događaja i 1 koji treba da prima samo jedan događaj. U consumeru koji želi da prima samo jedan događaj mogao bi se napraviti filter koji će odbacivati one događaje koje consumer ne treba da obrađuje.

6. KOMPARATIVNA ANALIZA PREFORMANSI

Jako je teško kvantifikovati performanse sistema jer one mogu zavisiti od puno parametara: konfiguracija, hardverski resursi, memorija…

U svrhu ovog istraživanja, sprovedena je detaljna komparativna analiza između dva popularna alata - Apache Kafka i RabbitMq. Za potrebe analize, kreirana su

dva projekta, pri čemu je svaki od njih koristio jedan od navedenih alata. U oba projekta, implementirana je osnovna konfiguracija i dodat je isti Java kod. Jedina razlika između projekata jeste sistem za razmenu poruka koji je integrisan u njih. Na producer strani nalaze se 4 filea iz kojih se podaci čitaju redom, a za svaki podatak se šalju dve poruke. Na consumer strani vrši se agregacija na osnovu pristiglih poruka, zatim se rezultati upisuju u file i meri se vreme potrebno za slanje svih ovih poruka.

U slučaju prvog file-a šalje se 2.000 poruka. Kafka je uspešno obradila ove poruke za 0.2 sec, dok je RabbitMq za to trebalo 2 sec. Razlika je skoro neosetna ali se čak i na ovako malom broju poruka već vidi razlika u performansama, što jasno ukazuje na prednost Kafke. Kafkina log bazirana arhitektura omogućava brzu i efikasnu obradu poruka čak i u manjim scenarijima. Iz ovoga, a takođe i iz truda i vremena uloženog za konfigurisanje sistema u projekte se može doneti zaključak da u nekim situacijama, gde se ne šalje veliki broj poruka ili brzina obrade poruka nije bitna, Kafka nije uvek najbolji izbor zbog njene složenosti. Kafka zahteva određeni nivo konfiguracije i infrastrukture da bi pravilno funkcionisala. Međutim, u manjim scenarijima ovakva kompleksnost može biti nepotrebna i dodatno otežati implementaciju i održavanje. Sa druge strane, RabbitMq se veoma jendostavno konfiguriše, što olakšava rad u manjim okruženjima. Takođe Kafka klaster i ako može biti horizontalno skaliran, i dalje zahteva odredjene resurse za svoje funkcionisanje. U scenarijima malog broja poruka, često je javlja pitanje da li su troškovi i resursi potrebni za održavanje Kafke opravdani, s obzirom na njihovu potencijalno visoku vrednost. Iz navedenih razloga je u ovakvim situacijama bolje koristiti neki jednostavniji alat za razmenu poruka, kao što je RabbitMq. Ovakvi alati će imati skoro iste performanse a mnogo su lakši za implementaciju i održavanje.

Sledeći file nad kojim su se pokretali projekti jeste file posle čijeg čitanja se šalje 20.000 poruka. Kafka je završila obradu svih poruka za 6 sec, dok je RabbitMq trebalo 10 sec. Kao što se može videti, i u ovom slučaju razlike u performansama i dalje nisu previše izražene, ali Kafkina log bazirana arhitektura omogućava brže pisanje i čitanje poruka čak i kada se koristi jedan broker.

Posle trećeg file-a šalje se 200.000 poruka. U poređenju, Kafka je završila slanje i obradu svih ovih poruka za samo 14 sec, dok je RabbitMq za istu operaciju trebalo čak 48 sec. U ovom slučaju je prvi put prisutna veća razlika. Kafka ima prednost svoje sposobnosti za horizontalno skaliranje, a samim tim povećanje broja poruka nije veliki izazov za Kafka klaster. Sa druge strane, RabbitMq sa svojom centralizovanom arhitekturom pokazuje slabije performanse u scenarijima većeg broja poruka, jer centralni sistem može postati usko grlo.

Poslednji file jeste file nakon čijeg čitanja se šalje 2.000.000 poruka. Kafka je završila slanje i obradu svih ovih poruka za 89 sec, a RabbitMq za 337 sec. Kao što se može primetiti, sa povećanjem broja poruka koje se šalju, razlika između Apache Kafke i RabbitMq je sve veća. U ovako zahtevnim scenarijima Kafka pokazuje svoju punu snagu.

Uz dodatno skaliranje i optimizaciju Kafka bi pokazala još bolje performanse. Kafka je sposobna da obradi ogromnu količinu podataka uz minimalni gubitak performansi. Nasuprot tome, RabbitMq je ograničen svojom centralizovanom prirodom, koja ograničava njegovu sposobnost brze obrade, i zbog koje u ovakvim situacijama zaostaje za Kafkom.

Iz priloženog se može zaključiti da Kafka pruža bolje performanse u poređenju sa RabbitMq, i da ukoliko je brzina slanja i obrade poruka ono što je krucijalno za projekat, svakako treba izabrati Kafku. Takođe važno je napomenuti da izbor između ova 2 alata zavisi od konkretnih zahteva projekta. RabbitMq takođe ima svoje prednosti, posebno u slučajevima gde je potrebna jednostavna i brza implementacija ili zahtevno rutiranje.

U oba ova slučaja korišćena je osnovna konfiguracija, tako da se performanse u oba slučaja mogu znatno poboljšati podešavanjem parametara.

7. ZAKLJUČAK

Kafka je posebno dizajnirana da bi se nosila sa zahtevima visokih performansi, skalabilnosti i izdržljivosti. Njena sposobnost horizontalnog skaliranja, kao i sposobnost efikasnog upravljanja velikom količinom podataka, čini je izuzetno pogodnom za scenarije koji zahtevaju obradu u realnom vremenu i sisteme zasnovane na događajima.

Međutim i ako Kafka ima brojne prednosti, uspeh zavisi od pažljivog planiranja, projektovanja i konfigurisanja. Integracija sa postojećim sistemima zahteva duboko razumevanje kako Kafka funkcioniše i kako je najbolje iskoristiti njene performanse.

Sa druge strane, RabbitMq nudi širok spektar mogućnosti za upravljanje redovima. Osim toga, pruža podršku za različite tipove razmene poruka, što omogućava fleksibilnost u komunikaciji. Takođe, RabbitMq podržava prioritet poruka, i osim standardnog AMQP protokola podržava i druge protokole komunikacije kao što su MQTT i STOMP, pružajući tako više opcija za razmenu poruka. Zahvaljujući raznovrsnosti alata i bibliloteka koje podržavaju RabbitMq, moguće je lako integrisati ovaj sistem sa raznim aplikacijama i tehnologijama, što ga čini prilagodljivim i pogodnim za različite scenarije.

Na kraju krajeva, najbolji alat zavisi od potreba. Iz ličnog iskustva mogu reći da je Apache Kafka mnogo korišćeniji alat na komercijalnim/klijentskim projektima, i da je njene performanse, skalabilnost, izdržljivost i sposobnost za rad sa događajima čine jako moćnim resursom u modernom svetu podataka. Primena Apache Kafke u raznim domenima otvara mnoge mogućnosti za buduća istraživanja i inovacije u oblasti obrade podataka.

8. LITERATURA

[1] Gwen Shapira, Todd Palino, Rajini Sivaram, Krit Petty, “Kafka The Definitive Guide - Real-Time Data and Stream Processing at Scale”, O'Reilly Media, 2021

[2] Martin Kleppmann, “Designing data-intensive applications”, O’Reilly Media

[3] “Traditional Message Queues vs. Log-based Message Brokers”, https://rkenmi.com/posts/traditional-message-queues-vs-log-based-message-brokers/
[4] Linkedin, “RabbitMq Features & Architecture”, https://www.linkedin.com/pulse/rabbitmq-features-architecture-huzaifa-asif/, poslednji pristup 23.9.2023.
[5] aws, https://aws.amazon.com/compare/the-difference-between-rabbitmq-and-kafka/, poslednji pristup 23.9.2023.

Kratka biografija:

Nikolina Tomić rođena je u Novom Sadu 24.12.1996. godine. Upisala fakultet 2015. godine, smer Elektrotehnika i računarstvo. Diplomirala 2020. godine sa temom „Bajesove mreže“.