RAPORT DE ANALIZA

Kafka vs RabbitMQ

Ioana-Carina Anton
Teodora Adriana Fariseu

Kafka vs RabbitMQ

Diferente functionale si de arhitectura

Kafka

Event streaming

Event streaming este procesul de a captura date in timp real de la surse de evenimente precum bazele de date, senzori, dispozitive mobile, servicii cloud si aplicatii software sub forma unor fluxuri de evenimente. Asadar, fluxurile de evenimente asigura un flow continuu si o interpretare a datelor in asa fel incat informatiile corecte sa fie in locurile potrivite la timpul potrivit.

Apache Kafka ca platforma de event streaming

Kafka combina 3 capabilitati cheie, a.i. sa putem implementa cazurile de utilizare pentru end-to-end event streaming cu ajutorul unei singure solutii battle-tested:

- *publish* (write) si *subscribe* (read) fluxuri de evenimente, inclusiv importul/exportul continuu al datelor din alte sisteme
- *stocare* a fluxurilor de evenimente in mod durabil si fiabil
- *procesarea* fluxurilor de evenimente, fie in timp real, fie intr-un mod retrospectiv

Livrarea mesajelor

Exista mai multe moduri de a garanta livrarea mesajelor:

- cel mult o data – mesajele se pot pierde, dar nu vor fi retrimise
- cel putin o data – mesajele nu sunt pierdute niciodata, dar exista posibilitatea de a fi retirmise
- exact o data – fiecare mesaj este trimis o data si doar o data

In Kafka, in momentul in care un mesaj a fost publicat, acesta va fi inscris intr-un log. Odata ce un mesaj publicat a fost inscris in log, acesta nu va mai fi pierdut cat timp un broker care reproduce partitia in care a fost scris acest mesaj ramane in viata.

In cazul in care un producer icnearca sa publice un mesaj si intampina o eroarea de retea, nu poate fi sigur daca aceasta eroare s-a intamplat inainte sau dupa ce mesajul a fost inregistrat.

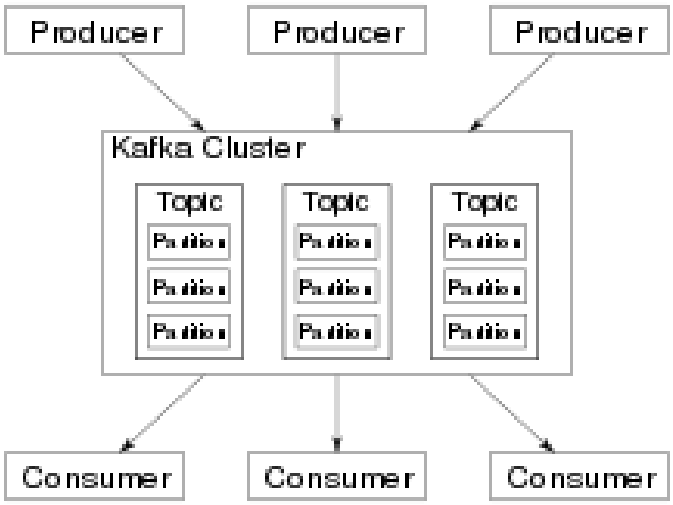
De la ver. 0.11.0.0, producer-ul Kafka suporta si o optiune de livrare idempotenta, care asigura faptul ca relivrarea nu va rezulta in duplicarea intrarii in log. Pentru a realiza acest lucru, broker-ul atribuie fiecarui producer un ID si deduplica mesajele folosind un numar de secventa care este tirmis de catre producer impreuna cu fiecare mesaj.

Pentru cazurile de utilizare senzitive la latenta, exista optiunea de a lasa producer-ul sa aleaga nivelul de durabilitate dorit. Totusi, producerul poate opta si sa transmita mesajele in mod complet asincron sau ca doreste pana cand liderul (nu neaparat urmaritorii) sa aiba mesajul.

Suport tranzactional

De la versiunea 0.11.0.0, producerul Kafka suporta si abilitatea de a trimite mesaje la mai multe partitii ale unui topic folosind semantici tranzactionale. Fie toate mesajele sunt trimise cu succes sau nici unul. Cazul de utilizare pentru acesta este pentru procesarea de tipul “o singura data” intre topicuri Kafka.

Arhitectura / Logica de rutare



Kafka stocheaza mesaje de tipul cheie-valoare care provin in mod arbitrat de la diferite procese numite producer. Datele pot fi impartite in diferite partitions in cadrul diferitelor topics. In cadrul unei partitii, mesajele sunt ordonate strict in functie de offset-ul lor (pozitia unui mesaj in cadrul unei partitii), indexate si stocate impreuna cu un timestamp. Alte procese, numite consumers, pot sa citeasca mesajele din partitii. Pentru procesarea stream-urilor, Kafka pune la dispozitie diferite librarii API, 5 cele mai importante fiind:

- **Producer API –** permite unei aplicatii sa publice streamuri de inregistrari
- **Consumer API –** permite unei aplicatii sa se aboneze la topic-uri si sa proceseze stream-urile
- **Connector API –** executa api-urile reutilizabile de producer si consumer care pot sa faca legatura dintre topics cu aplicatiile existente
- **Streams API –** converteste stream-urile de intrare in iesire si produce rezultatul
- **Admin API –** administrarea Kafka topics, brokers si alte obiecte Kafka

Securitate

Metode de securitate suportate:

- Autentificare de conexiuni la brokers de la clienti, alte brokers si instrumente, utilizand SSL sau SASL
- Autentificarea conexiunilor de la brokers la ZooKeeper
- Criptarea datelor transferate intre brokers si clienti, intre brokers sau intre brokers si instrumente, utilizand SSL
- Autorizarea operatiilor de read si write de clienti
- Autorizarea este conectabila si este acceptata integrarea cu serviciile de autorizare externe

Securitatea este optionala – clientii non-securizati sunt suportati, de asemenea si o combinatie de clienti autentificati, neautentificati, criptati si necriptati.

RabbitMQ

Livrarea mesajelor

In cazul unui eveniment in care o conexiune esueaza intre un client si un nod RabbitMQ, clientul va fi nevoi sa stabileasca o conexiune noua catre broker. Orice canal deschis in conexiunea precedenta va fi inchisa in mod automat si va trebui redeschisa.

In general, cand o conexiunea esueaza, clientul va fi informat de conexiune prin aruncarea unei exceptii.

Cand o conexiune esueaza, mesajele s-ar putea afla inca in transit intre client si server – ar putea fi in mijlocul procesului de a fi decodificate sau codificate de orice parte, sa fie in bufferele stivei TCP. In cazul acestor evenimente, mesajele in tranzit nu vor fi livrate – ele vor trebui sa fie retransmise. Clientii si serverul stiu cand sa faca acest lucru prin utilizarea de acknowledgements.

Acknowledgement-urile pot fi utilizate in ambele directii – sa permita unui client sa indice serverului ca a primit si/sau procesat un mesaj si sa permita serverului sa indice acelasi lucru publisher-ului. Acestea sunt cunoscute sub forma de *consumer acknowledgement* si *publisher confirms*.

O aplicatie consumatoare nu ar trebui sa marcheze mesaje ca acknowledged numai in cazul in care a terminat procesarea completa a lor. O data ce a fost marcat, broker-ul este liber sa marcheze mesajele pentru stergere.

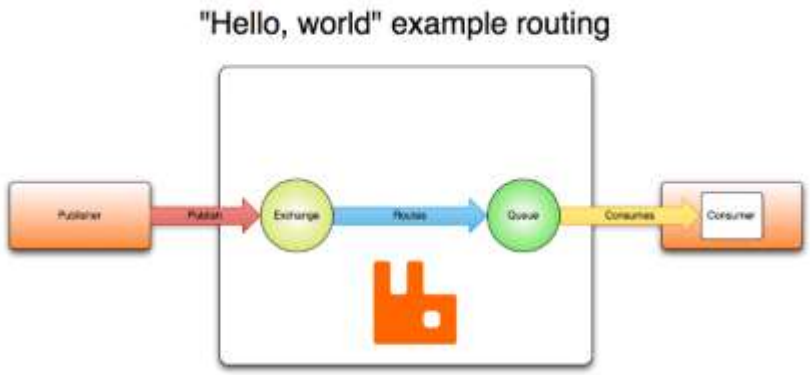
Utilizarea de aknowledgements asigura o livrare *cel putin o data*. Fara acknowledgements, pierderea de mesaje este posibila in timpul operatiilor de publicare si consumare si este garantata doar o livrare de tipul *cel mult o data*.

AMQP 0-9-1

AMQP 0-9-1 (Advanced Message Queuing Protocol) este un protocol de mesaje care permite aplicatiilor client sa comunice cu brokers de middleware de mesagerie.

Arhitectura/Logica de rutare

Brokerii de mesaje primesti mesaje de la *publishers* (/producers) si le ruteaza catre *consumers*. Deoarece este un protocol de retea, producatorii, consumatorii si broker-ul pot sta pe masini diferite.



In modelul AMQP 0-9-1, mesajele sunt publicate la *exchanges*. Exchange-urile distribuie copii de mesaje la *queues* utilizant reguli numite *bindings*. Dupa aceea, broker-ul poate fie sa livreze mesajul la consumatorii subscribed la coada, sau consumatorii isi aduc mesajele din coada.

- Producer – trimite un mesaj la coada de mesaje prin intermediul unui exchange
- Exchange – este responsabil de rutarea mesajelor cu linkuri si rutarea cheilor la diferite cozi. Un binding este o relatie dintre un exchange si un queue
- Queue – un buffer de mesaje
- Consumer – consuma mesaje dintr-o coada

Securitate

Fiecare conexiune RabbitMQ are asociat un utilizator care este autentificat. De asemenea, tinteste si un host virtual pentru care utilizatorul trebuie sa aiba un anumite set de permisiuni. Credentialele utilizatorului, host-ul virtual si (optional) certificatul clientului sunt specificate la initializarea conexiunii.

Calitatea serviciului oferit

	Kafka	RabbitMQ
Disponibilitate	- replicare	- replicare - poate fi configurat sa replice informatiile

		legate de exchange si bindings - in cazul replicarii cozilor, trebuie facuta o setare manual
Performanta/Debit	1 million messages/sec	4K-10K messages/sec
Scalabilitate	- in momentul adaugarii unui nod nou in cluster, utilizatorul poate sa aleaga sa mute partitii deja existente in noul nod creat	- suport bun pentru adaugarea sau stergerea unor noduri noi in cluster
Latenta	- 50 percentile: 1ms - - 99.9 percentile fara replicare: 15ms - - 99.9 percentile cu replicare: 30ms	- media: 1-4ms - max: 2-17ms

Concluzii

Tehnologia RabbitMQ este mai potrivita pentru cazurile de utilizare simple. Exista anumite avantaje in trafic de date mai scazut, precum priority queue si posibilitati mai flexibile de rutare.

Kafka se potriveste pe date masice si debit inalt, la care RabbitMQ nu poate face fata, sau in cazul in care ai nevoie de un commit log sau consumatori multiplii.

Bibliografie

[1]https://en.wikipedia.org/wiki/Apache_Kafka

[2]https://kafka.apache.org/documentation/

[3]https://www.rabbitmq.com/documentation.html

[4] https://www.rabbitmq.com/tutorials/amqp-concepts.html

[5] https://www.educba.com/rabbitmq-vs-kafka/