Apache Kafka VS Apache Pulsar VS Rabbit MQ: un'analisi tecnica

Andrea Acampora
Giacomo Accursi

Alma Mater Studiorum · Università di Bologna
Campus di Cesena
Dipartimento di Informatica, Scienza e Ingegneria

Febbraio 2024

Obiettivi del progetto

L'obiettivo di questo progetto è effettuare un confronto tecnico fra i più popolari message broker presenti sul mercato: *Apache Kafka*, *Apache Pulsar* e *RabbitMQ*.

Analisi teorica

- Caratteristiche e funzionalità
- Componenti e protocolli
- Modello dei dati
- Modello di consegna
- Schema dei messaggi

Confronto pratico

- Facilità d'uso
- Performance
  - Latenza
  - Throughput
- Tolleranza ai guasti

Introduzione alle tecnologie

Apache Kafka

Apache Kafka è una piattaforma distribuita per lo streaming di eventi sviluppata da Linkedin, utilizzata per la creazione di pipeline di dati in tempo reale e applicazioni di streaming.

Apache Pulsar

Apache Pulsar è una piattaforma di messaggistica e streaming publish-subscribe sviluppata da *Yahoo* ed assorbita poi da *Apache Foundation*.

RabbitMQ

RabbitMQ è un sistema di messaggistica open-source creato in Erlang e basato sul modello di messaggistica *"Message Queue"*.

Tecnologie a confronto - Componenti

Ad alto livello l'architettura di tutti i message broker consiste di tre elementi principali: produttore, consumatore e broker.

Componenti cluster

- **Apache Kafka**: Broker, Apache Zookeeper (Kraft), Schema Registry
- **Apache Pulsar**: Broker, Pulsar Proxy, Apache Zookeeper, Apache Bookkeeper
- **RabbitMQ**: Broker

Broker

- **Apache Kafka**: topic e partizioni
- **Apache Pulsar**: topic e partizioni
- **RabbitMQ**: message queue, exchange

Tecnologie a confronto - Protocolli

Apache Kafka

Utilizza un protocollo binario su *TCP*. Il protocollo definisce tutte le API come coppie di messaggi richiesta/risposta.

Apache Pulsar

Utilizza un protocollo binario personalizzato per le comunicazioni tra produttori/consumatori e broker.

RabbitMQ

Utilizza il protocollo di comunicazione AMQP (Advanced Message Queuing Protocol). È un protocollo aperto, standardizzato e indipendente dal fornitore, progettato per la messaggistica avanzata.

Tecnologie a confronto - Sicurezza

Autenticazione

- **Apache Kafka**: utilizza *SASL (Simple Authentication and Security Layer)* ma supporta molti meccanismi, tra cui *SCRAM*, *OAUTHBEARER* e *GSSAPI*.
- **Apache Pulsar**: Le configurazioni di sicurezza di Pulsar includono opzioni per abilitare l'autenticazione tramite *Token JWT (JSON Web Token)* o *TLS (Transport Layer Security)*.
- **RabbitMQ**: *plain authentication*, *LDAP authentication*, *SSL/TLS authentication*.

Autorizzazione

- **Apache Kafka**: *ACL (Access Control Lists)*.
- **Apache Pulsar**: *RBAC (Role Based Access Control)*.
- **RabbitMQ**: *RBAC (Role Based Access Control)*.

Tecnologie a confronto - Persistenza

Apache Kafka

In Apache Kafka, trattandosi di un sistema di commit-log, i record vengono aggiunti alla fine di ogni partizione e ogni partizione è divisa in segmenti.

Apache Pulsar

Il livello di persistenza in Apache Pulsar è affidato ad Apache BookKeeper. I dati sono organizzati in "ledger", ovvero una sequenza di write-ahead log (WAL) associata a un topic specifico.

RabbitMQ

La gestione dello storage è principalmente orientata alla conservazione temporanea dei messaggi fino a quando non vengono elaborati o consegnati ai consumatori. Le code "durable" garantiscono la persistenza dei messaggi anche a seguito del riavvio del server, scrivendo i dati su disco.

Tecnologie a confronto - Tolleranza ai guasti

Apache Kafka

- Replicazione delle partizioni dei topic
- Elezione nuovo leader in seguito al fallimento di un nodo

Apache Pulsar

- Multi-tenancy e isolamento
- Ripartizione dei dati sui boookies
- Ripristino automatico dei fallimenti

RabbitMQ

- Quorum Queues
- Modalità cluster per distribuzione su più nodi

Tecnologie a confronto - Modello di consegna

Pattern di comunicazione

Apache Kafka si basa su un modello di comunicazione **Publish-Subscribe**, RabbitMQ segue il modello classico di **Message Queue**, mentre Pulsar offre un modello di comunicazione flessibile, combinando gli aspetti di Publish-Subscribe e Message Queue.

Modello di consumo dei messaggi

- **Apache Kafka**: pull-based
- **Apache Pulsar**: push-based, con un API che simula il pull dei consumatori
- **RabbitMQ**: push-based

Tecnologie a confronto - Modello di consegna

Semantica e garanzie di consegna

Sia RabbitMQ che Apache Kafka e Apache Pulsar offrono garanzie di messaggistica. Tutti offrono garanzie *at-most-once* e *at-least-once*, ma Apache Kafka offre anche garanzie *exactly-once* in alcuni scenari molto limitati. Apache Kafka e Apache Pulsar supportano le scritture idempotenti, il che significa che i messaggi duplicati non vengono memorizzati nel livello di memorizzazione.

Progettazione dei test

Metriche di valutazione

- **Facilità d'uso**
- **Latenza**
- **Throughput**
- **Tolleranza ai guasti**

Il progetto è hostato in una repository *GitLab* e come linguaggio di programmazione è stato scelto *Kotlin* per l'interoperabilità con *Java*, in quanto viene fornito il supporto da tutti e tre i Message Broker.

Progettazione dei test

Design e implementazione

- **Kafka-Driver**
- **Pulsar-Driver**
- **RabbitMQ-Driver**
- **Benchmark**

Deploy

Per agevolare la conduzione dei test i message broker sono stati eseguiti all'interno di container *Docker* tramite l'utilizzo del tool *Docker Compose*.

Testing

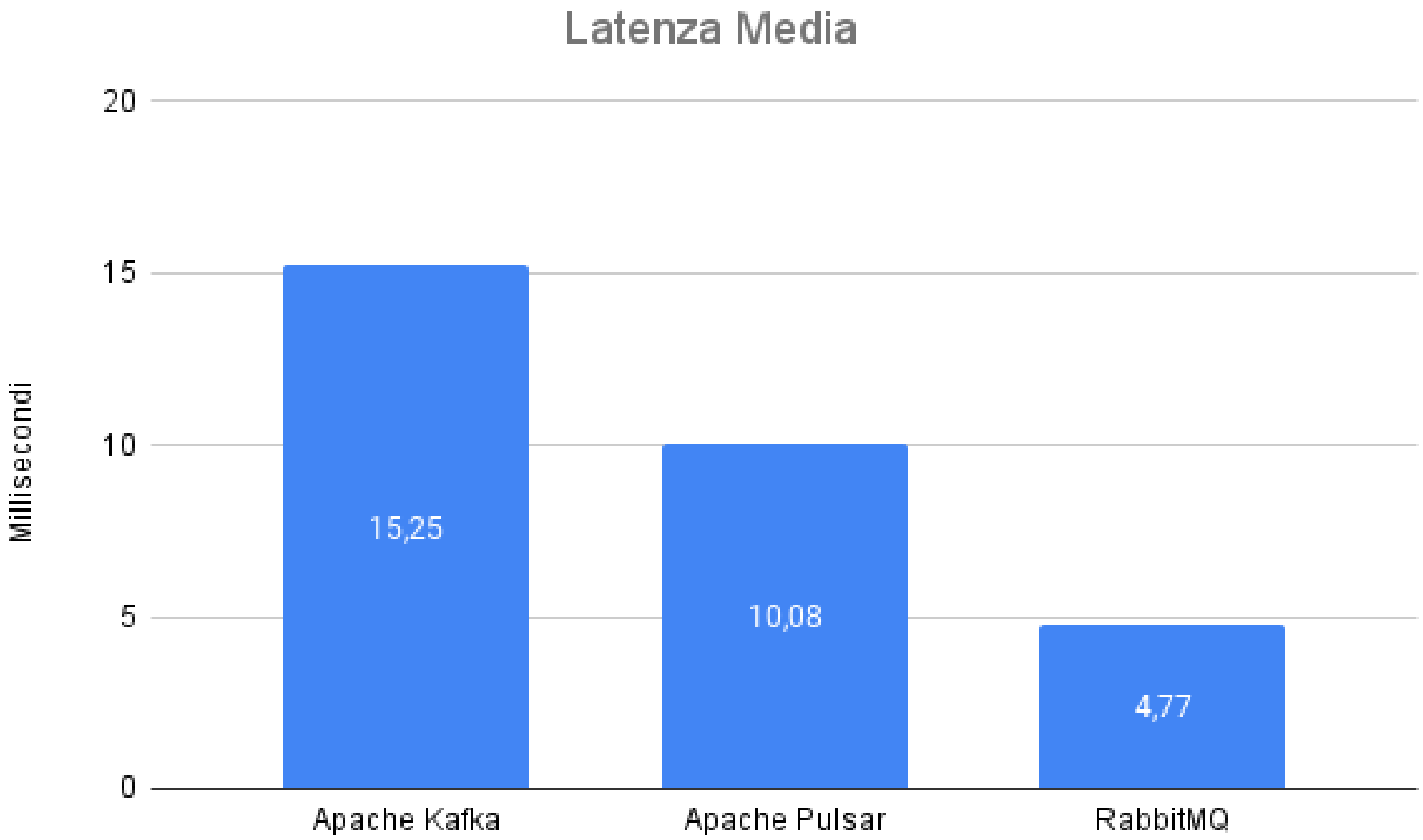
Al fine di testare i driver creati sono stati eseguiti alcuni Unit Tests utilizzando la suite *TestContainer*, necessaria per l'esecuzione dei message broker all'interno di container *Docker*.

Analisi dei risultati - Facilità d'uso

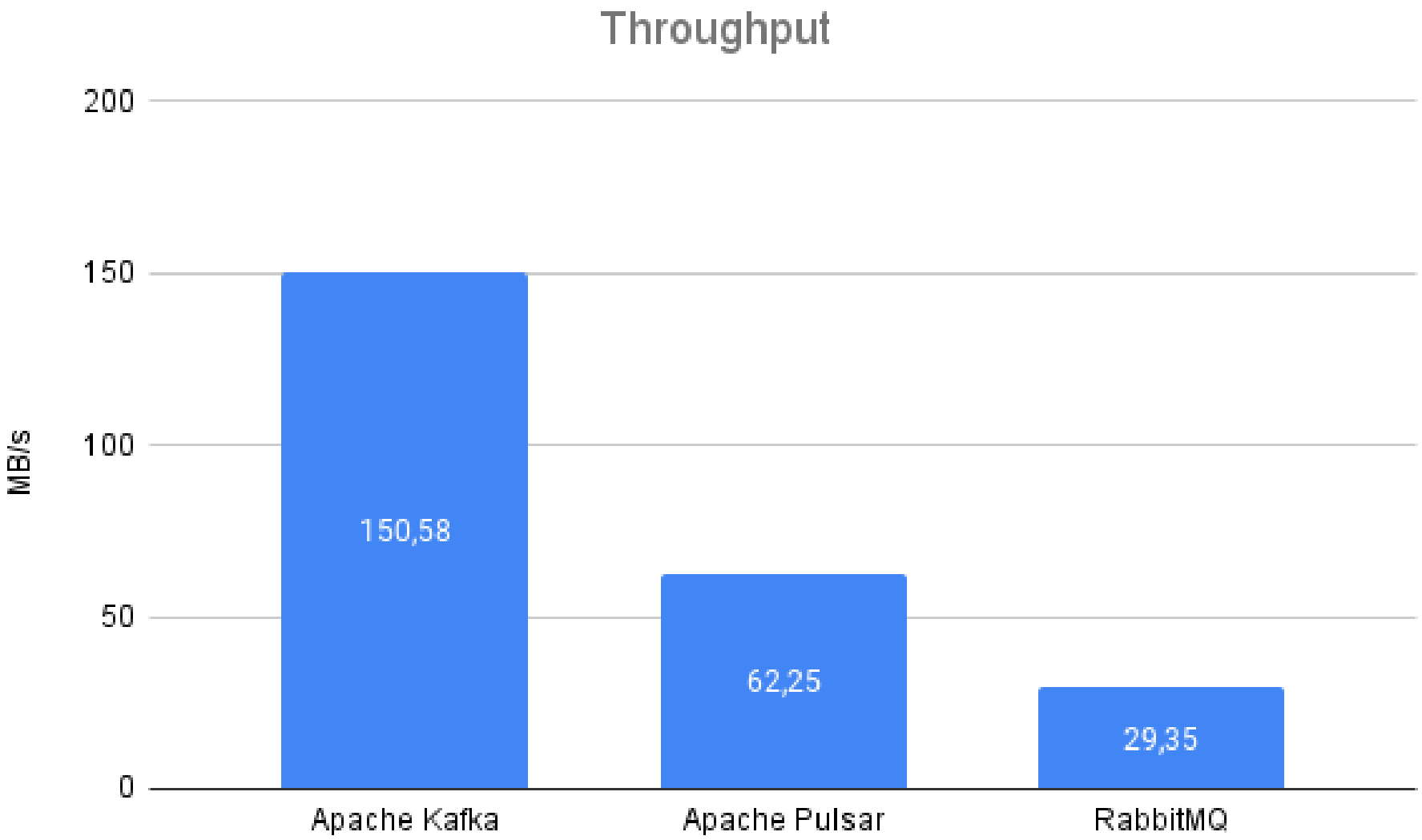
Facilità d'uso

RabbitMQ risulta essere il message broker che richiede meno configurazioni per l'avvio in quanto è sufficiente mandare in esecuzione solo il servizio del broker stesso. Apache Pulsar e Apache Kafka, invece, richiedono la presenza di ulteriori componenti quali ad esempio Apache Bookkeeper, per la gestione della persistenza e Apache Zookeeper per la gestione dei metadati del cluster.

Analisi dei risultati - Latenza



Analisi dei risultati - Throughput



Analisi dei risultati - Tolleranza ai guasti

In questo ultimo test è stato simulato il fallimento di un nodo del cluster al fine di valutare la capacità di esso di continuare a funzionare e mantenere la continuità del servizio.

Risultato

I test hanno dimostrato che tutte e tre le piattaforme sono state in grado di gestire con successo il fallimento di uno o più nodi del cluster.

Conclusioni

Possiamo affermare che Apache Kafka si distingue per il suo modello di log distribuito, Pulsar per l'architettura multi-layer, e RabbitMQ per la velocità e la flessibilità nel supportare vari protocolli di messaggistica. Apache Kafka e Apache Pulsar offrono strumenti e configurazioni avanzate, mentre RabbitMQ si distingue per la sua semplicità e facilità d'uso.

La scelta tra Apache Kafka, Apache Pulsar e RabbitMQ dipende fortemente dai requisiti specifici del sistema, dalle preferenze di implementazione e dalle priorità degli sviluppatori.