Apache Pulsar

Vinicius Gomes Gualberto
dept. Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU)
Monte Carmelo, Brasil
viniciusgualberto@ufu.br

***Index Terms*—**

I. Introdução

[**?**] ApacheLaTeX. Os sistemas distribuídos são formados por um conjunto de computadores independentes que se apresentam ao usuário como um sistema único e coerente. Essa abordagem proporciona benefícios como escalabilidade, tolerância a falhas e maior disponibilidade. No entanto, também introduz complexidades como comunicação, sincronização e gerenciamento de estado entre os diversos nós.

Para lidar com esses desafios, utiliza-se um Middleware, um tipo de software que atua como intermediário entre aplicações e recursos distribuídos. Ele abstrai a complexidade do ambiente subjacente, permitindo que desenvolvedores foquem na lógica da aplicação.

Dentre os diversos tipos de Middleware, destaca-se o Apache Pulsar, um sistema de mensagens em tempo real, projetado para oferecer alto desempenho, escalabilidade e suporte nativo a múltiplos modelos de comunicação. Ele é utilizado em arquiteturas orientadas a eventos e em pipelines de dados modernos, sendo uma alternativa robusta ao Apache Kafka.

II. Facilidade de uso

[**?**]

Apesar de sua arquitetura robusta e altamente escalável, o Apache Pulsar se destaca também pela sua facilidade de uso. A ferramenta oferece uma interface de linha de comando (*CLI*) simples e intuitiva para operações básicas como produção e consumo de mensagens. Além disso, o Pulsar possui APIs para diversas linguagens de programação, como Java, Python, Go e C++, o que facilita sua integração em diferentes tipos de aplicações.

A instalação em modo standalone é direta e ideal para testes e desenvolvimento local, sem exigir uma configuração complexa de cluster. Além disso, a documentação oficial é bem estruturada e repleta de exemplos práticos, o que acelera a curva de aprendizado mesmo para usuários iniciantes em sistemas de mensageria distribuída.

Outro ponto importante é a criação dinâmica de tópicos e o suporte automático a múltiplos padrões de assinatura (exclusive, shared, failover), que permitem ao desenvolvedor adaptar o sistema às necessidades da aplicação sem grandes modificações na infraestrutura.

Em suma, o Apache Pulsar combina potência com simplicidade, oferecendo recursos avançados sem comprometer a usabilidade.

III. Fundamentação

Para compreender o funcionamento do Apache Pulsar como Middleware de mensageria distribuída, é necessário dominar uma série de conceitos técnicos que sustentam sua arquitetura e operação. A seguir, detalharemos os principais fundamentos necessários para compreender o Pulsar em profundidade.

A. Mensageria Assíncrona

A mensageria assíncrona é um modelo de comunicação amplamente adotado em sistemas distribuídos modernos, especialmente em arquiteturas orientadas a eventos e microsserviços. Nesse modelo, o produtor de mensagens envia dados sem precisar aguardar que o consumidor processe a informação naquele exato momento. Essa característica oferece desacoplamento temporal entre os componentes, promovendo maior tolerância a falhas, escalabilidade e desempenho.

Por exemplo, em uma plataforma de e-commerce, o serviço de pagamento pode publicar uma mensagem de ”pedido confirmado” sem se preocupar com qual serviço a consumirá ou quando isso ocorrerá. O serviço de estoque ou o de emissão de nota fiscal pode consumir essa mensagem em momentos distintos, sem interferir no fluxo principal.

Esse paradigma contrasta com a comunicação síncrona, como chamadas HTTP entre APIs, onde o remetente bloqueia a execução até receber uma resposta. No Pulsar, a comunicação assíncrona é implementada de forma nativa com garantia de entrega, persistência e modelos avançados de assinatura.

B. Broker

O broker é o componente central responsável por intermediar a comunicação entre produtores e consumidores. Ele atua como uma ponte lógica que:

- Recebe as mensagens dos produtores.

- Gerencia a distribuição dessas mensagens para os consumidores.
- Roteia mensagens conforme a assinatura configurada (exclusive, shared, failover).
- Aplica políticas de retenção, compressão, quotas e autenticação.

No Apache Pulsar, os brokers não armazenam diretamente as mensagens em disco. Em vez disso, eles transferem os dados recebidos para o Apache BookKeeper, especializado no armazenamento durável e replicado. Essa separação entre roteamento e armazenamento melhora drasticamente a escalabilidade horizontal: é possível adicionar mais brokers para lidar com picos de tráfego sem afetar a persistência dos dados.

Além disso, os brokers mantêm o controle de sessões dos consumidores, detectam falhas e reatribuem tópicos quando necessário, colaborando com o Zookeeper.

C. Producer, Consumer e Topic

Esses três elementos constituem o núcleo da lógica de mensagens no Apache Pulsar. Sua interação define o fluxo de informações no sistema:

- **Producer (Produtor)**: é uma aplicação ou componente que publica mensagens em um tópico. Um mesmo tópico pode ter múltiplos produtores, e os dados podem ser enviados com diferentes níveis de prioridade, compressão ou criptografia.
- **Consumer (Consumidor)**: é quem se inscreve em um tópico e recebe mensagens conforme as regras da assinatura. No Pulsar, os consumidores podem pertencer a diferentes grupos de assinatura, com comportamento personalizado, como balanceamento de carga (shared) ou tolerância a falhas (failover).
- **Topic (Tópico)**: é o canal lógico onde as mensagens são publicadas e lidas. Ele pode ser:
  - *Non-partitioned (não particionado)*: um único fluxo de mensagens.
  - *Partitioned*: dividido em múltiplas partições, cada uma tratada como um subfluxo independente, permitindo paralelismo e alto throughput.

Além disso, o Apache Pulsar permite a criação dinâmica de tópicos, o que facilita o desenvolvimento em ambientes altamente dinâmicos, como IoT e microsserviços.

D. Zookeeper e BookKeeper

Esses dois sistemas trabalham em conjunto com o Apache Pulsar para fornecer persistência, coordenação e tolerância a falhas:

- **Apache Zookeeper**: atua como serviço de coordenação distribuída. Ele armazena metadados sobre o cluster, como tópicos, sessões ativas, status dos brokers e informações de particionamento. Quando um broker falha, o Zookeeper notifica os demais componentes, permitindo a redistribuição automática de tópicos.
- **Apache BookKeeper**: é o sistema de log distribuído responsável por armazenar todas as mensagens publicadas. Ele organiza os dados em estruturas chamadas *ledgers*, que são replicadas em múltiplos nós conhecidos como *bookies*. Cada mensagem pode ter diferentes níveis de replicação, dependendo das configurações do tópico (ex: fator de replicação 3).
  Esse modelo de armazenamento distribuído oferece alta durabilidade, throughput elevado e baixa latência. Além disso, BookKeeper permite leitura assíncrona de mensagens antigas para reprocessamento, o que é útil em casos de falhas, auditorias ou treinamento de modelos de IA.

A arquitetura baseada nesses dois serviços permite ao Pulsar operar com alta disponibilidade e consistência forte (CAP: CP).

E. Assinaturas

O Apache Pulsar fornece modelos de assinatura flexíveis para lidar com diferentes padrões de consumo:

- **Exclusive**: apenas um consumidor pode estar conectado ao tópico com determinada assinatura. As mensagens são entregues exatamente uma vez (at-most-once). É o modelo mais simples e útil para aplicações que exigem controle total sobre o processamento.
- **Shared**: múltiplos consumidores compartilham o consumo das mensagens. O Pulsar realiza o balanceamento automático de carga, atribuindo mensagens a consumidores de forma distribuída. Esse modelo é ideal para processamentos paralelos de alto volume, como em sistemas de recomendação ou análise em tempo real.
- **Failover**: define uma ordem de prioridade entre consumidores. Apenas o consumidor com maior prioridade recebe as mensagens; os demais permanecem inativos. Em caso de falha, um consumidor de backup assume automaticamente. Esse modelo garante alta disponibilidade sem perda ou duplicação de dados.

Esses modelos permitem que o Apache Pulsar atenda a diferentes requisitos de negócio, desde sistemas com entrega garantida até ambientes com balanceamento de carga dinâmico.

IV. Operação

O Apache Pulsar foi projetado com uma arquitetura moderna e modular, voltada para alta performance, escalabilidade e confiabilidade. Nesta seção, exploramos detalhadamente seus principais recursos, componentes e mecanismos operacionais, oferecendo uma visão técnica aprofundada sobre como o middleware funciona na prática.

A. Arquitetura Modular

O Apache Pulsar possui uma arquitetura distribuída baseada em três componentes principais:

- **Broker**: responsável por receber mensagens de produtores, aplicar políticas de roteamento e distribuí-las aos consumidores.
- **BookKeeper (Storage Layer)**: encarregado do armazenamento persistente e replicado das mensagens.

- **Zookeeper (Coordination Layer)**: fornece gerenciamento de configuração, descoberta de serviços e coordenação do cluster.

Diferentemente de outras plataformas de mensageria que combinam roteamento e armazenamento no mesmo componente, o Pulsar separa claramente essas responsabilidades. Isso traz vantagens como:

- Escalabilidade horizontal independente (é possível escalar brokers e bookies separadamente).
- Alta disponibilidade nativa, com failover automático.
- Maior throughput e menor latência, já que os brokers podem operar em memória e delegar persistência ao BookKeeper.

B. Modelo de Publicação e Assinatura (Pub/Sub)

O Pulsar adota o modelo de comunicação *Publish/Subscribe*, em que produtores publicam mensagens em tópicos e consumidores assinam esses tópicos para receber mensagens. Essa abordagem desacopla os componentes e facilita a evolução do sistema ao longo do tempo.

Os tópicos podem ser:

- **Normais (non-partitioned)**: possuem um único fluxo de mensagens.
- **Particionados (partitioned)**: divididos logicamente em subfluxos, cada um com sua própria sequência de mensagens. Isso permite paralelismo no consumo e aumento no throughput.

Essa estrutura é ideal para cargas de trabalho que requerem alto desempenho, como processamento de logs, eventos de IoT e dados de transações em tempo real.

C. Gerenciamento de Tenants, Namespaces e Tópicos

O Apache Pulsar suporta **multi-tenancy**, ou seja, permite que múltiplos usuários ou aplicações compartilhem a mesma infraestrutura com isolamento lógico. Essa capacidade é essencial em ambientes de nuvem e plataformas de dados compartilhadas.

A estrutura hierárquica é:

- **Tenant**: representa uma organização ou cliente distinto.
- **Namespace**: subdivisão dentro de um tenant, usada para aplicar políticas de retenção, replicação e quotas.
- **Topic**: canal lógico onde mensagens são trocadas.

Essa organização permite configurar diferentes políticas de forma granular e segura, como retenção de mensagens, tempo de expiração, número de assinantes, entre outros.

D. Persistência e Armazenamento com BookKeeper

Ao contrário de soluções que armazenam dados apenas em memória ou de forma local, o Pulsar usa o Apache BookKeeper como camada de persistência. As mensagens são armazenadas em estruturas chamadas *ledgers*, que são replicadas automaticamente entre múltiplos *bookies* para garantir tolerância a falhas.

Características importantes:

- Escrita sequencial otimizada em disco.
- Replicação síncrona para garantir durabilidade.
- Separação entre armazenamento e processamento, permitindo elasticidade.

Essa arquitetura garante que mesmo em caso de falha de um ou mais nós, os dados permaneçam íntegros e acessíveis, o que é essencial em sistemas críticos.

E. Entrega de Mensagens

O Pulsar oferece diferentes modos de entrega de mensagens, ajustando-se a diversos requisitos de negócios:

- **At-most-once**: as mensagens são entregues no máximo uma vez, sem garantias de reentrega em caso de falha.
- **At-least-once**: as mensagens podem ser reentregues se necessário, garantindo que nenhuma mensagem seja perdida.
- **Effectively-once**: modo avançado que combina exatamente uma entrega com consistência de estado, usado em cenários com integração de transações.

Além disso, o Pulsar permite **acknowledgment manual** ou automático, controle de fluxo, bufferização inteligente e configuração de *dead letter topics*, nos quais mensagens problemáticas são redirecionadas para análise posterior.

F. Controle de Fluxo, Compressão e Segurança

O Apache Pulsar é altamente configurável e possui suporte a diversas funcionalidades avançadas:

- **Controle de fluxo**: os consumidores podem regular a taxa de recebimento de mensagens com base em sua capacidade de processamento.
- **Compressão**: suporte a vários algoritmos (LZ4, Zlib, ZSTD, Snappy) para reduzir o volume de dados trafegados e armazenados.
- **Segurança**: suporte a autenticação (TLS, OAuth2, JWT), autorização baseada em permissões e comunicação criptografada.

Esses recursos tornam o Pulsar adequado tanto para ambientes de produção em larga escala quanto para aplicações com requisitos rígidos de segurança e desempenho.

G. APIs, SDKs e Integrações

O Apache Pulsar fornece APIs nativas para diversas linguagens, incluindo:

- Java (cliente oficial e mais completo)
- Python (usado amplamente em aplicações de dados)
- Go, C++, Node.js e WebSocket (para casos específicos)

Além disso, o Pulsar se integra com uma vasta gama de ferramentas e ecossistemas, como:

- Apache Flink, Spark, NiFi
- Kafka (via Pulsar Kafka Wrapper)
- Debezium e Confluent Connectors
- Prometheus, Grafana (monitoramento)

Essas integrações reforçam o uso do Pulsar em arquiteturas de Big Data, IA, ETL e microsserviços, promovendo interoperabilidade e facilidade de adoção.

V. Desmostração do código

A. Objetivo

Este estudo de caso visa demonstrar a aplicação do middleware Apache Pulsar em um sistema bancário distribuído, utilizando Java com Maven. O sistema é composto por um cliente que envia transações e um servidor que processa essas transações e as registra em uma estrutura simplificada de blockchain.

B. Estrutura do Projeto

Pacote: banco

Dependências: ¡!– pom.xml –¿ ¡dependency¿ ¡groupId¿org.apache.pulsar¡groupId¿ ¡artifactId¿pulsar-client¡artifactId¿ ¡version¿2.11.0¡/version¿ ¡/dependency¿

C. Código do Servidor

package banco; import org.apache.pulsar.client.api.*; public class ServidorBancario public static void main(String[] args) throws PulsarClientException Blockchain blockchain = new Blockchain(); pgsql Copiar Editar PulsarClient cliente = PulsarClient.builder() .serviceUrl(”pulsar://localhost:6650”) .build(); Consumer¡byte[]¿ consumidor = cliente.newConsumer() .topic(”transacoes-bancarias”) .subscriptionName(”sub-banco”) .subscriptionType(SubscriptionType.Shared) .subscribe(); System.out.println(”Servidor iniciado. Aguardando transações...”); while (true) Message¡byte[]¿ mensagem = consumidor.receive(); String transacao = new String(mensagem.getData()); System.out.println(”Transação recebida: ” + transacao); blockchain.adicionarTransacao(transacao); consumidor.acknowledge(mensagem); blockchain.imprimirCadeia(); System.out.println(”——————————”);

D. Código do Cliente

package banco; import java.util.Scanner; import org.apache.pulsar.client.api.*; public class ClienteBancario public static void main(String[] args) throws PulsarClientException PulsarClient cliente = PulsarClient.builder() .serviceUrl(”pulsar://localhost:6650”) .build(); pgsql Copiar Editar Producer¡byte[]¿ produtor = cliente.newProducer() .topic(”transacoes-bancarias”) .create(); Scanner scanner = new Scanner(System.in); System.out.println(”Cliente bancário iniciado.”); while (true) System.out.print(”Digite a transação (ou ’sair’): ”); String entrada = scanner.nextLine(); if (entrada.equalsIgnoreCase(”sair”)) break; produtor.send(entrada.getBytes()); System.out.println(”Transação enviada com sucesso.”); cliente.close();

E. Execução no Terminal

Para rodar o sistema:

1) Abra o terminal na pasta onde está o Apache Pulsar.
2) Inicie o servidor Pulsar:

```
bin\pulsar standalone
```

3) Em outro terminal, vá para o diretório do projeto Maven:

```
cd C:\Users\vinic\OneDrive\Área de Trabalho\UF
mvn clean compile exec:java -Dexec.mainClass="
```

4) Em um terceiro terminal, execute o cliente:

```
mvn exec:java -Dexec.mainClass="banco.ClienteB
```

TABLE I
Exemplo de Saída do Servidor

Evento	Saída do Terminal
Servidor iniciado	Aguardando transações...
Transação 1	Bloco 1: Transferência de R$200 recebida
Transação 2	Bloco 2: Pagamento de boleto R$120 recebido

F. Resultados

Após executar o cliente e enviar transações, o terminal do servidor exibe:

G. Análise

O Apache Pulsar demonstrou ser eficaz no tratamento de mensagens em tempo real, provendo baixa latência e alto throughput. A arquitetura baseada em tópicos e assinaturas facilita o desacoplamento entre produtor e consumidor, característica essencial em sistemas distribuídos.

VI. LaTeX-Considerações Finais

Além da compreensão técnica, este trabalho proporcionou uma visão estratégica sobre o papel de middlewares como o Apache Pulsar em arquiteturas modernas. Foi possível identificar como sua proposta de separação entre processamento e armazenamento traz vantagens significativas em termos de desempenho e escalabilidade, especialmente em aplicações que exigem alta taxa de transferência de mensagens e confiabilidade.

A implementação prática com código Java reforçou o aprendizado dos conceitos teóricos e demonstrou a aplicabilidade do Pulsar em cenários reais, como sistemas bancários e aplicações de missão crítica. A simplicidade da API, combinada com uma arquitetura sólida, torna o Pulsar acessível tanto para desenvolvedores iniciantes quanto para arquitetos de sistemas experientes.

Por fim, este estudo reforça a importância do uso consciente de tecnologias de middleware em sistemas distribuídos, destacando o Apache Pulsar como uma ferramenta promissora que alia flexibilidade, desempenho e robustez. Seu domínio técnico pode representar um diferencial competitivo para desenvolvedores e empresas que desejam construir soluções escaláveis e resilientes.

References

[1] Kjerrumgaard, David. Apache Pulsar in action. Simon and Schuster, 2021. KJERRUMGAARD, David. Apache Pulsar in action. Simon and Schuster, 2021. Kjerrumgaard, D. (2021). Apache Pulsar in action. Simon and Schuster.

[2] Sharma, Rahul, and Mohammad Atyab. ”Introduction to apache pulsar.” Cloud-Native Microservices with Apache Pulsar: Build Distributed Messaging Microservices. Berkeley, CA: Apress, 2021. 1-22.