XII CONGRESSO DE INICIAÇÃO CIENTÍFICA DO IFSP ITAPETININGA
Itapetininga, 19, 20 e 21 de maio de 2026
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Itapetininga

CONCEITOS E PRÁTICAS DE COMUNICAÇÃO ASSÍNCRONA ENTRE APLICAÇÕES COM APACHE KAFKA

Marco Antônio Z. Montagna – Bolsista PACTec/IFSP[1]
Profa. Dra. Janaina C. Abib - IFSP[2]
Prof. Me. Ednilson G. Rossi - IFSP[3]

Introdução

A comunicação entre aplicações constitui um elemento central nos sistemas distribuídos modernos, especialmente em cenários que demandam elevada escalabilidade, alta disponibilidade e processamento contínuo de dados. Nesse contexto, a adoção de arquiteturas baseadas em microsserviços intensificou a necessidade de mecanismos mais eficientes para a troca de informações entre componentes distribuídos, de modo a reduzir o acoplamento entre serviços e ampliar a resiliência operacional. Conforme Paiva e Santos (2025), arquiteturas orientadas a microsserviços, quando associadas a práticas adequadas de comunicação entre sistemas, favorecem maior escalabilidade, melhor desempenho e facilidade de manutenção, sendo especialmente indicadas para aplicações corporativas de larga escala. Na arquitetura de microsserviços a comunicação síncrona, embora amplamente utilizada, apresenta limitações relacionadas ao bloqueio de requisições e à dependência direta entre serviços, o que pode comprometer o desempenho e a disponibilidade da aplicação. Como alternativa, a comunicação assíncrona apresenta-se como uma abordagem mais robusta, pois permite que mensagens sejam enviadas e processadas de forma independente, sem a necessidade de espera imediata por respostas, ou seja, não caracteriza uma chamada bloqueante. Esse modelo de comunicação reduz o acoplamento entre os serviços, aumenta a tolerância a falhas e favorece a escalabilidade do sistema, uma vez que os componentes podem operar de forma independente, mesmo diante de indisponibilidades momentâneas de outros serviços. A literatura recomenda que sistemas baseados em microsserviços, sempre que possível, utilizem mensagens assíncronas entre serviços internos, reservando a comunicação síncrona para interações que exigem resposta imediata ao usuário final (MICROSOFT, 2026). Entre as tecnologias que viabilizam esse modelo de comunicação, destacam-se as plataformas de streaming de eventos, como o Apache Kafka, que possibilita o processamento contínuo de grandes volumes de dados em tempo real, com alta tolerância a falhas e escalabilidade horizontal. O Apache Kafka é amplamente utilizado em sistemas modernos devido à sua capacidade de atuar como um barramento de eventos distribuído, permitindo o desacoplamento entre

[1] Estudante do curso de Bacharelado em Engenharia de Software, Instituto Federal de São Paulo (IFSP) – São Carlos/SP. E-mail: m.zanchettta@aluno.ifsp.edu.br.
[2] Docente da Área de Computação. Instituto Federal de São Paulo (IFSP) – Araraquara/SP. E-mail: janaina@ifsp.edu.br.
[3] Docente da Área de Computação. Instituto Federal de São Paulo (IFSP) – Araraquara/SP. E-mail: ednilsonrossi@ifsp.edu.br.

produtores e consumidores das mensagens e, dessa forma, garantindo persistência e o reprocessamento de mensagens, aumentando a resiliência de todo o sistema (APACHE SOFTWARE FOUNDATION, 2026).

Objetivo

Conhecer os conceitos fundamentais da comunicação assíncrona entre aplicações utilizando o Apache Kafka, compreendendo sua arquitetura, funcionamento e aplicabilidade em sistemas distribuídos modernos, além de evidenciar suas vantagens em relação à comunicação síncrona.

Metodologia

Este trabalho adotou uma abordagem exploratória e aplicada, combinando levantamento bibliográfico e desenvolvimento experimental. Inicialmente, realizou-se uma revisão da literatura sobre sistemas distribuídos, comunicação assíncrona, serviços de mensageria e plataformas de *event streaming*. Em seguida, foi desenvolvida uma prova de conceito utilizando JavaScript na implementação dos serviços e Docker para conteinerização do ambiente, com o objetivo de validar os conceitos estudados e demonstrar o funcionamento do Apache Kafka em um cenário controlado.

Resultados

De acordo com Tanenbaum e van Steen (2007), sistemas distribuídos consistem em componentes independentes que cooperam por meio de rede, exigindo mecanismos eficientes de comunicação, escalabilidade e tolerância a falhas. Inicialmente, foram analisadas obras acadêmicas e documentações técnicas relacionadas à comunicação assíncrona e ao funcionamento do Apache Kafka, possibilitando a compreensão de seus principais componentes, tais como produtores (*producers*), responsáveis pelo envio de mensagens; consumidores (*consumers*), encarregados da leitura e processamento; *brokers*, que armazenam e distribuem os dados; tópicos, canais lógicos de publicação; e partições, utilizadas para escalabilidade e paralelismo (NARKHEDE; SHAPIRA; PALINO, 2017). Para a experimentação, foi empregado um ambiente conteinerizado com Docker, permitindo a simulação de um sistema distribuído de forma controlada, portátil e reprodutível. A aplicação prática https://github.com/marcozmz/hello-world-apache-kafka/ foi desenvolvida em JavaScript, no qual produtores enviaram mensagens estruturadas em formato JSON para tópicos específicos, enquanto consumidores realizaram a leitura e o processamento assíncrono dessas mensagens. Durante os testes, foram explorados conceitos como particionamento de tópicos, paralelismo no consumo, persistência de mensagens e controle de *offsets*, mecanismo que registra a posição de leitura de cada consumidor. Esses elementos permitiram observar o comportamento do sistema em diferentes cenários de carga e processamento, evidenciando características como escalabilidade, desacoplamento e resiliência, atributos essenciais em arquiteturas orientadas a eventos (KLEPPMANN, 2017). A partir dos estudos realizados, percebe-se que o Apache Kafka apresenta elevada capacidade de processamento de dados, suportando o envio e consumo de mensagens de forma eficiente, até mesmo em cenários com grande volume de informações. A utilização de tópicos e partições possibilitou a distribuição de carga entre múltiplos consumidores, promovendo escalabilidade horizontal e melhor aproveitamento de recursos computacionais. Observou-se que a persistência das

mensagens em disco garante maior confiabilidade, permitindo o reprocessamento de eventos em caso de falhas ou indisponibilidade temporária dos consumidores. Além disso, o modelo de controle de *offsets* possibilita maior flexibilidade no consumo das mensagens, permitindo que aplicações retomem o processamento a partir de pontos específicos. Em comparação com a comunicação síncrona, o uso do Apache Kafka apresenta significativa redução no acoplamento entre serviços, além de maior tolerância a falhas e resiliência do sistema como um todo. A arquitetura baseada em eventos se beneficia dos registros distribuídos (*logs*), um recurso nativo do Apache Kafka. Esse recurso favorece a observabilidade e rastreabilidade dos eventos, requisitos amplamente necessários em aplicações modernas.

Conclusão

A partir dos estudos realizados e da implementação prática desenvolvida, o uso do Apache Kafka indica ser uma solução robusta, escalável e eficiente para a implementação de comunicação assíncrona em sistemas distribuídos, especialmente em cenários que demandam processamento em tempo real e alta disponibilidade. A utilização de registros distribuídos, aliada ao uso de tópicos e particionamento, permite o desenvolvimento de aplicações desacopladas, resilientes e altamente escaláveis. A integração com tecnologias como JavaScript e Docker facilita sua adoção em ambientes modernos de desenvolvimento, contribuindo para a construção de arquiteturas baseadas em microsserviços e eventos. Como continuidade deste trabalho, propõe-se a realização de estudos comparativos entre o Apache Kafka e outras soluções de mensageria, como o RabbitMQ, visando analisar suas diferenças arquiteturais, desempenho e aplicabilidade em diferentes cenários, contribuindo para a escolha adequada de tecnologias em projetos de sistemas distribuídos.

Referências

APACHE SOFTWARE FOUNDATION. Apache Kafka. [S. l.], 2024. Disponível em: https://kafka.apache.org/. Acesso em: 25 abr. 2026.

KLEPPMANN, Martin. Designing data-intensive applications: the big ideas behind reliable, scalable, and maintainable systems. 1. ed. Sebastopol: O'Reilly Media, 2017.

MICROSOFT. Comunicação assíncrona baseada em mensagens. Microsoft Learn, 2026. Disponível em: https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/architect-microservice-container-applications/asynchronous-message-based-communication. Acesso em: 25 abr. 2026.

NARKHEDE, Neha; SHAPIRA, Gwen; PALINO, Todd. Kafka: the definitive guide: real-time data and stream processing at scale. 1. ed. Sebastopol, CA: O'Reilly Media, 2017.

PAIVA, G. R. C.; SANTOS, P. C. Arquitetura de software: microsserviços, escalabilidade e práticas eficientes na comunicação entre sistemas. In: JORNADA CIENTÍFICA E TECNOLÓGICA DO IFSULDEMINAS, 17.; SIMPÓSIO DE PÓS-GRADUAÇÃO DO IFSULDEMINAS, 14., 2025. Anais [...]. Disponível em: https://josif.ifsuldeminas.edu.br/ojs/index.php/anais/article/view/2753. Acesso em: 20 mar. 2026.

TANENBAUM, Andrew S.; STEEN, Maarten van. Sistemas distribuídos: princípios e paradigmas. 2. ed. São Paulo: Pearson Prentice Hall, 2007.