Luís Clício Carvalho Sá
Orientadora: Juliana Oliveira de Carvalho

ArionStream: uma arquitetura distribuída para análise de vídeos em tempo real em aplicações de IoMT na borda

Picos - PI
Julho de 2025

Luís Clício Carvalho Sá
Orientadora: Juliana Oliveira de Carvalho

ArionStream: uma arquitetura distribuída para análise de vídeos em tempo real em aplicações de IoMT na borda

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Sistemas de Informação da Universidade Federal do Piauí, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Bacharelado em Sistemas de Informação

Picos - PI
Julho de 2025

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

S111a Sá, Luís Clício Carvalho.
ArionStream: uma arquitetura distribuída para análise de vídeos em tempo real em aplicações de IoMT na borda / Luís Clício Carvalho Sá – 2025.
62 f.

1 Arquivo em PDF.
Indexado no catálogo *online* da biblioteca José Albano de Macêdo-CSHNB
Aberto a pesquisadores, com restrições da Biblioteca.

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal do Piauí, Curso de Bacharelado em Sistemas de Informação, Picos, 2025.
"Orientadora: Juliana Oliveira de Carvalho".

1. Sistemas informacionais. 2. Arquitetura computacional. 3. Tecnologia da informação – ArionStream. I. Sá, Luís Clício Carvalho. II. Carvalho, Juliana Oliveira de. III. Título.

CDD 005.7

Elaborada por Maria Letícia Cristina Alcântara Gomes
Bibliotecária CRB n° 03/1835

ARIONSTREAM: UMA ARQUITETURA DISTRIBUÍDA PARA ANÁLISE DE VÍDEOS EM TEMPO REAL EM APLICAÇÕES DE IOMT NA BORDA

LUÍS CLÍCIO CARVALHO SÁ

Monografia aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 27 de junho de 2025

gov.br
Documento assinado digitalmente
JULIANA OLIVEIRA DE CARVALHO
Data: 01/07/2025 19:05:54-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dra. Juliana Oliveira de Carvalho

gov.br
Documento assinado digitalmente
FRANCISCO AIRTON PEREIRA DA SILVA
Data: 01/07/2025 14:16:14-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dr. Francisco Airton Pereira da Silva

gov.br
Documento assinado digitalmente
RAYNER GOMES SOUSA
Data: 01/07/2025 18:08:36-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dr. Rayner Gomes Sousa

Agradecimentos

Agradeço, primeiramente, a Deus, pelo dom da vida e por tudo o que tem providenciado em minha caminhada. Aos meus pais, Climério e Sílvia, que sempre fizeram o possível — e muitas vezes o impossível — para nos guiar pelos caminhos do bem. Sou profundamente grato por todo o esforço, pelos ensinamentos e pelos valores que transmitiram a mim e aos meus irmãos, sempre com amor, fé e dedicação. Aos meus irmãos, Clísnei, Silmara e Clisandro, por estarem ao meu lado como apoio incondicional, fonte de inspiração e amizade verdadeira. À minha esposa, Mikaele, por ser minha companheira de todos os dias, dividindo os desafios e as conquistas dessa jornada com amor, paciência e parceria. A todos os meus familiares, professores e amigos que, de diferentes formas, fizeram e fazem parte dessa trajetória, meu sincero agradecimento. Um agradecimento especial à minha orientadora, professora Dr^{a}. Juliana, pela orientação atenciosa, pelo suporte técnico e pelos valiosos conselhos ao longo deste trabalho. E, ao meu querido pai, que já não está entre nós, expresso aqui minha gratidão mais profunda e sincera. Que, de onde estiver, possa ver que todo o seu esforço, dedicação e exemplo continuam dando frutos. Este trabalho também é para o senhor.

A todos, o meu muito obrigado.

E, na verdade, toda a correção, ao presente, não parece ser de gozo, senão de tristeza, mas depois produz um fruto pacífico de justiça nos exercitados por ela.

Bíblia Sagrada CCB, Hebreus 12:11

Resumo

A crescente complexidade de aplicações baseadas em vídeo, como monitoramento urbano inteligente, segurança automatizada e análise de processos industriais, tem elevado a demanda por arquiteturas capazes de realizar análise de vídeo em tempo real com baixa latência e alta resiliência, mesmo em cenários de conectividade limitada. Esses desafios são ainda mais acentuados no contexto da Internet das Coisas Multimídia (IoMT), onde sensores visuais operam em grande escala e geram fluxos contínuos e volumosos de dados que exigem processamento distribuído e rápido. Diante desse cenário, este trabalho propõe a ArionStream: uma arquitetura distribuída e conteinerizada, voltada à análise de vídeo em tempo real na borda da rede. A ArionStream adota o paradigma de microsserviços e uma estratégia de comunicação assíncrona híbrida — com e sem intermediação por serviços de mensageria — para promover desacoplamento entre componentes, escalabilidade horizontal e tolerância a falhas. O projeto foi fundamentado a partir de conceitos consolidados de sistemas distribuídos, computação em borda e arquitetura orientada a serviços, e implementado com foco na flexibilidade de implantação e na adaptabilidade a diferentes requisitos. A validação experimental da arquitetura foi conduzida por meio de testes de desempenho em múltiplos cenários, variando a complexidade do fluxo de dados e a quantidade de nós computacionais. Foram coletadas e analisadas métricas como latência entre serviços e escalabilidade. Os resultados demonstraram que a ArionStream é capaz de manter desempenho consistente, com latências estáveis e comportamento previsível mesmo sob diferentes configurações e cargas de trabalho. Observou-se, ainda, que a arquitetura foi eficaz em conter o acúmulo de latência nos serviços terminais, mesmo quando serviços iniciais estavam sob alta carga. A principal hipótese deste trabalho é que, em aplicações de IoMT com análise de vídeo em tempo real, arquiteturas distribuídas construídas com comunicação assíncrona e processamento na borda representam uma solução eficaz para superar os gargalos típicos da computação centralizada em nuvem. Assim, a ArionStream oferece um modelo conceitual e prático que pode ser adaptado a diferentes domínios críticos, contribuindo para o avanço de sistemas multimídia resilientes e escaláveis.

Palavras-chaves: Computação em borda; IoMT; Análise de vídeo; Arquitetura distribuída; Microsserviços; Comunicação assíncrona.

Abstract

The increasing complexity of video-based applications — such as smart urban monitoring, automated surveillance, and industrial process analysis — has intensified the demand for architectures capable of performing real-time video analysis with low latency and high resilience, even in limited-connectivity environments. These challenges become even more critical in the context of the Internet of Multimedia Things (IoMT), where visual sensors operate on a large scale and generate continuous, high-volume data streams that require distributed and fast processing. In this context, this work proposes ArionStream: a distributed and containerized architecture designed for real-time video analysis at the network edge. ArionStream adopts the microservices paradigm and a hybrid asynchronous communication strategy — with and without intermediary messaging services — to promote component decoupling, horizontal scalability, and fault tolerance. The architecture was designed based on consolidated concepts from distributed systems, edge computing, and service-oriented architecture, with emphasis on flexible deployment and adaptability to diverse operational requirements. The architecture was experimentally validated through performance tests in multiple scenarios, varying both the dataflow complexity and the number of computing nodes. Metrics such as service-to-service latency and scalability were collected and analyzed. The results demonstrated that ArionStream is capable of maintaining consistent performance, with stable latencies and predictable behavior under different workloads and configurations. Furthermore, it was observed that the architecture effectively limited the propagation of latency to downstream services, even when upstream services were under heavy load. The central hypothesis of this work is that in IoMT applications involving real-time video analysis, distributed architectures that combine asynchronous communication with edge processing represent an effective strategy to overcome typical bottlenecks of cloud-centric computing. Thus, ArionStream contributes both as a conceptual model and a practical solution for building resilient and scalable multimedia systems across critical domains.

Keywords: Edge computing; IoMT; Video analysis; Distributed architecture; Microservices; Asynchronous communication.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AMQP	*Advanced Message Queuing Protocol*
API	*Application Programming Interface*
CPU	*Central Processing Unit*
FPS	*Frames Per Second*
HTTP	*Hypertext Transfer Protocol*
IoMT	*Internet of Multimedia Things*
IoT	*Internet of Things*
RAM	*Random Access Memory*
RGB	*Red, Green, Blue*
RTSP	*Real Time Streaming Protocol*
Wi-Fi	*Wireless Fidelity*

Sumário

1 Introdução

A análise de vídeo em tempo real tem se consolidado como uma área estratégica para diversas aplicações críticas, como segurança pública, monitoramento urbano, supervisão industrial e gestão de tráfego (HASSAN; HASSAN, 2022). Esses sistemas são impulsionados pela crescente demanda por decisões rápidas e precisas, baseadas na extração automática de informações relevantes a partir de fluxos contínuos de vídeo (LEE; CAMERON; HASSALL, 2022). Com o avanço da visão computacional e da inteligência artificial, tornou-se viável substituir a vigilância humana por mecanismos automatizados capazes de detectar eventos, comportamentos ou anomalias com maior precisão e consistência (GUBBI et al., 2013). No entanto, essa evolução tecnológica trouxe consigo um conjunto de desafios significativos relacionados à latência, ao volume massivo de dados gerados e à escalabilidade das soluções existentes (MONTAZEROLGHAEM, 2021).

De acordo com o relatório da Ericsson Mobility Report, aproximadamente 76% do tráfego móvel global será composto por vídeos até 2025 (ORSOLIC; SKORIN-KAPOV, 2020), o que evidencia o impacto crescente desse tipo de dado sobre as infraestruturas de rede. Esse cenário é especialmente crítico no contexto da Internet das Coisas Multimídia (*Internet of Multimedia Things*, IoMT), que integra dispositivos inteligentes com capacidades sensoriais visuais — como câmeras — em ambientes conectados. A proliferação desses dispositivos intensifica a geração de dados e impõe requisitos rigorosos de processamento em tempo real (TU; XU; CHEN, 2021; ZHANG et al., 2021). Além disso, aplicações de IoMT frequentemente operam em ambientes com recursos computacionais restritos, conectividade intermitente ou latência elevada, o que compromete a eficácia dos sistemas que dependem exclusivamente de processamento em nuvem (ZIELINSKI et al., 2019; ZEN et al., 2022).

A necessidade de respostas rápidas e autônomas em tempo real torna inviável, em muitos casos, a transferência de todos os dados para centros remotos de processamento. Nesse contexto, o paradigma de *Edge Computing* desponta como uma alternativa estratégica, ao aproximar os recursos computacionais das fontes de dados (SHI et al., 2016). O processamento na borda permite que inferências e ações sejam realizadas localmente, reduzindo a dependência de conectividade constante com a nuvem, minimizando a latência e aumentando a resiliência do sistema (LU et al., 2023). Ainda assim, desenvolver arquiteturas distribuídas que conciliem desempenho, escalabilidade e robustez em ambientes de IoMT continua sendo um desafio em aberto, especialmente quando se considera a heterogeneidade dos dispositivos, a variabilidade do tráfego de dados e a necessidade de adaptação dinâmica (DEBAUCHE; MAHMOUDI; GUTTADAURIA, 2022).

Diante dessas necessidades, este trabalho propõe a ArionStream, uma arquitetura distribuída projetada para a análise de vídeo em tempo real em aplicações de IoMT na borda

da rede. A proposta se fundamenta em uma abordagem assíncrona e modular, baseada em serviços conteinerizados que permitem a orquestração flexível de diferentes etapas do processamento de vídeo. Com a adoção de comunicação desacoplada e distribuição de tarefas, a ArionStream visa atender aos requisitos de latência, escalabilidade e tolerância a falhas, mesmo em ambientes com conectividade limitada. Além disso, a arquitetura foi projetada para permitir o compartilhamento eficiente de dados entre serviços e adaptar-se dinamicamente a mudanças na carga de trabalho e nas condições de rede.

A viabilidade da proposta foi avaliada por meio de experimentos controlados, nos quais a arquitetura foi submetida a diferentes cenários de carga e distribuição computacional. Os testes permitiram medir a latência de transferência de dados entre os serviços, a escalabilidade em função do número de dispositivos e a robustez diante da sobrecarga de fluxos de vídeo. Os resultados demonstram que a ArionStream é capaz de processar dados de vídeo em tempo real de forma eficiente, mantendo latência estável e sem perdas de dados, mesmo sob condições adversas. Tais evidências indicam o potencial da arquitetura para compor sistemas de monitoramento resilientes, adaptáveis e alinhados às exigências da computação na borda.

Este trabalho está estruturado da seguinte forma: o Capítulo 2 apresenta o referencial teórico, abordando os principais conceitos envolvidos. O Capítulo 3 apresenta os trabalhos relacionados, discutindo abordagens relevantes da literatura. O Capítulo 4 descreve a arquitetura ArionStream em detalhes, abordando seus fundamentos, topologia e componentes. Em seguida, o Capítulo 5 apresenta os experimentos realizados e os resultados obtidos. Por fim, o Capítulo 6 apresenta as conclusões do trabalho, destacando suas contribuições, limitações e possibilidades de continuidade futura.

1.1 Objetivos

Este trabalho tem como objetivo geral propor, implementar e validar uma arquitetura distribuída voltada para aplicações de Internet das Coisas Multimídia (IoMT) com execução na borda da rede. A motivação central reside na necessidade de aprimorar o desempenho, a flexibilidade e a tolerância a falhas em cenários caracterizados por conectividade limitada e alta demanda por processamento de vídeo em tempo real.

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- Investigar e comparar tecnologias e abordagens existentes para o processamento de vídeo em tempo real na borda, com o intuito de embasar tecnicamente a implementação da arquitetura proposta no contexto de aplicações IoMT.
- Realizar um levantamento bibliográfico sobre técnicas de processamento distribuído aplicáveis a tarefas de visão computacional, com o objetivo de consolidar o conhe-

cimento existente e apoiar pesquisas futuras nas áreas de sistemas distribuídos e processamento de imagens.

- Oferecer uma base técnica para o desenvolvimento de aplicações robustas de IoMT que exijam processamento de vídeo na borda, auxiliando profissionais na construção de soluções capazes de atender a requisitos rigorosos de desempenho e resiliência.

2 Referencial teórico

A arquitetura ArionStream, proposta neste trabalho, fundamenta-se em um conjunto interdisciplinar de conceitos e tecnologias oriundos de diferentes áreas da computação, tais como sistemas distribuídos, microsserviços, comunicação assíncrona, *streaming* de vídeo, computação em borda (*Edge Computing*) e Internet das Coisas Multimídia (IoMT). A compreensão aprofundada desses fundamentos é essencial para contextualizar as decisões de projeto da arquitetura, bem como para interpretar os resultados obtidos nos experimentos realizados.

Portanto, este capítulo tem o propósito de apresentar os principais fundamentos teóricos que embasam a proposta, fornecendo ao leitor os subsídios conceituais necessários para a análise crítica da ArionStream. A abordagem dos tópicos selecionados busca evidenciar não apenas a relevância das tecnologias envolvidas, mas também sua integração coerente em uma solução voltada à análise de vídeo em tempo real em ambientes distribuídos e com restrições de conectividade. Ao consolidar esse embasamento, pretende-se estabelecer uma base sólida para a compreensão da arquitetura proposta e de suas contribuições.

2.1 Sistemas distribuídos

A arquitetura de sistemas distribuídos constitui um paradigma essencial na computação moderna, caracterizado pela cooperação entre múltiplos computadores autônomos interconectados, que atuam de forma coordenada para a execução de tarefas complexas e o compartilhamento de recursos (BAZGIR; HAQUE; UDDIN, 2024). Essa abordagem descentralizada oferece vantagens significativas em termos de escalabilidade, desempenho e tolerância a falhas, superando limitações típicas de arquiteturas centralizadas. A distribuição das tarefas entre os nós permite o processamento paralelo, o que contribui diretamente para o aumento da capacidade de resposta — aspectos críticos em aplicações que exigem alta disponibilidade e eficiência operacional (TYTARCHUK et al., 2024).

Um dos pilares dos sistemas distribuídos é o modelo de comunicação entre seus componentes, tradicionalmente realizado por meio da troca de mensagens. Essa comunicação pode ocorrer de forma síncrona, com espera pela resposta, ou de maneira assíncrona, na qual o envio de mensagens não bloqueia a execução do remetente (BUDAU; BERNARD, 2002). A escolha entre esses modelos depende das exigências específicas da aplicação. Em cenários que lidam com grandes volumes de dados ou demandam processamento contínuo em tempo real, a comunicação assíncrona destaca-se por oferecer maior desacoplamento entre os componentes, permitindo maior paralelismo e resiliência frente à variação de carga e possíveis falhas de rede.

Apesar de suas vantagens, sistemas distribuídos apresentam desafios significativos relacionados à consistência de dados, à coordenação entre componentes e à segurança das comunicações. A manutenção da consistência exige mecanismos robustos de sincronização e replicação de dados, que assegurem que todos os nós operem com informações atualizadas e confiáveis (BURCKHARDT, 2013). A coordenação dos elementos distribuídos, por sua vez, requer protocolos eficientes para a orquestração das tarefas, a resolução de conflitos e a manutenção da ordem de execução (DAVE et al., 2024).

Mesmo diante desses desafios, os sistemas distribuídos permanecem como uma solução robusta e versátil, amplamente adotada em aplicações que exigem elasticidade, alta disponibilidade e desempenho escalável (TYTARCHUK et al., 2024). A possibilidade de distribuir a carga de trabalho entre diferentes nós, aliada à capacidade de continuar operando mesmo diante de falhas parciais, torna essa arquitetura especialmente apropriada para ambientes dinâmicos, que demandam processamento contínuo e eficiente.

2.2 Microsserviços

A arquitetura de microsserviços configura-se como uma abordagem moderna para o desenvolvimento de aplicações distribuídas, baseada na decomposição de sistemas monolíticos em um conjunto de serviços pequenos, independentes e especializados. Cada microsserviço é responsável por uma funcionalidade específica e se comunica com os demais por meio de interfaces bem definidas, geralmente expostas por APIs (ISHA et al., 2023). Essa organização favorece a modularidade, promovendo maior flexibilidade, reutilização e independência no desenvolvimento, implantação e escalonamento dos componentes.

A adoção de microsserviços oferece uma variedade de vantagens em comparação com a arquitetura monolítica tradicional. A autonomia dos serviços simplifica a manutenção e a atualização do sistema, uma vez que as modificações em um serviço não exercem influência sobre os demais (ISHA et al., 2023). A escalabilidade granular possibilita que cada serviço seja dimensionado de acordo com suas necessidades particulares, otimizando o uso de recursos (RAFFIN et al., 2021). A adaptabilidade da arquitetura permite a utilização de diversas tecnologias e linguagens de programação para cada serviço, possibilitando a seleção das ferramentas mais adequadas para cada funcionalidade.

Contudo, o uso de microsserviços também impõe desafios, especialmente no que diz respeito à coordenação entre os serviços, à consistência de dados e à complexidade operacional (DAVE et al., 2024). A comunicação por meio de APIs requer atenção especial a aspectos como versionamento, latência e tolerância a falhas. A descentralização introduz dificuldades adicionais no monitoramento, na rastreabilidade e na gestão de dependências entre serviços. Para lidar com esses aspectos, faz-se necessário o uso de soluções robustas de orquestração, observabilidade e automação de infraestrutura (DAVE et al., 2024).

Apesar dessas complexidades, a arquitetura de microsserviços tem se consolidado como

uma escolha apropriada para sistemas que demandam escalabilidade, resiliência e agilidade no desenvolvimento. A capacidade de decompor um sistema complexo em serviços autônomos possibilita a construção de aplicações mais robustas e adaptáveis às mudanças nos requisitos de negócio (JOHNSON et al., 2024).

2.3 Comunicação assíncrona

A comunicação assíncrona configura-se como um paradigma de troca de informações no qual o emissor e o receptor não precisam estar simultaneamente ativos para a mensagem ser transmitida e processada. Diferentemente da comunicação síncrona, que exige resposta imediata entre as partes envolvidas, a comunicação assíncrona permite que mensagens sejam enviadas e recebidas em momentos distintos, promovendo maior flexibilidade, escalabilidade e tolerância a falhas em sistemas distribuídos (BUDAU; BERNARD, 2002).

Esse modelo de comunicação é especialmente vantajoso em ambientes nos quais a latência, a variabilidade da carga e a disponibilidade dos serviços são fatores críticos. Ao desacoplar os componentes do sistema, possibilita que cada serviço opere de forma independente, sem bloqueios, o que favorece o paralelismo e a continuidade do processamento (ZHANG; CHEN, 2024). Em aplicações como análise de vídeo em tempo real, onde o fluxo de dados é constante e o tempo de resposta é sensível, esse tipo de comunicação torna-se essencial para garantir desempenho e resiliência (KHAN, 2024).

A implementação de comunicação assíncrona em arquiteturas distribuídas geralmente envolve o uso de intermediários, como filas de mensagens ou sistemas de publicação/assinatura, que armazenam temporariamente as mensagens até que o destinatário esteja disponível. Essa estratégia assegura a entrega confiável mesmo em cenários de indisponibilidade momentânea de algum serviço, além de facilitar a expansão e a manutenção do sistema. Novos serviços podem ser adicionados ou removidos com mínima interferência, promovendo maior adaptabilidade e desacoplamento funcional (VOLODYMYR; DENYS, 2023).

Entretanto, o uso de comunicação assíncrona também impõe desafios. O gerenciamento das mensagens exige controle rigoroso sobre o fluxo, a ordenação e a persistência, especialmente em situações com múltiplos produtores e consumidores. A latência variável da rede, combinada à eventual ausência de garantias temporais rígidas, pode impactar a consistência dos dados e a previsibilidade do sistema (DAMYANOV; VARBANOV, 2024). Adicionalmente, a observabilidade e o rastreamento de mensagens tornam-se mais complexos, exigindo ferramentas adequadas de monitoramento distribuído.

Apesar dessas dificuldades, a comunicação assíncrona tem se mostrado uma escolha estratégica para aplicações que exigem escalabilidade, robustez e resiliência, como é o caso das soluções voltadas à IoMT. Quando corretamente implementada, permite construir sistemas modulares e responsivos, capazes de se adaptar a variações na carga de trabalho e a falhas temporárias de infraestrutura, sem comprometer a continuidade do serviço

(DAMYANOV; VARBANOV, 2024).

2.4 Streaming de vídeo

O *streaming* de vídeo é uma tecnologia essencial para a transmissão e fluxo contínuo de conteúdo visual por meio de redes de dados, permitindo que vídeos sejam reproduzidos em tempo real sem a necessidade de *download* completo do arquivo (RAMADHA; YOVITA; WIBOWO, 2022). Essa técnica baseia-se na fragmentação do conteúdo em pacotes, que são enviados sequencialmente ao dispositivo receptor, onde são decodificados e exibidos conforme chegam. O modelo de *streaming* tornou-se predominante em aplicações que demandam entrega instantânea de vídeo, como plataformas de entretenimento, videoconferência, monitoramento remoto e vigilância em tempo real.

A eficiência e a qualidade da transmissão por *streaming* dependem de fatores como a taxa de *bits* do vídeo, a largura de banda disponível e a capacidade de processamento dos dispositivos envolvidos (PATEL et al., 2024). Taxas de *bits* mais elevadas proporcionam vídeos de maior qualidade visual, mas exigem redes com maior capacidade de transmissão. A largura de banda, por sua vez, determina a velocidade de entrega dos pacotes, sendo um fator crítico para evitar *buffering* ou interrupções. A capacidade de processamento do dispositivo receptor afeta diretamente a fluidez da reprodução, especialmente em vídeos de alta resolução ou com altas taxas de quadros (TAN; WANG, 2023).

No contexto de transmissão em tempo real, como ocorre em aplicações de videomonitoramento, os desafios tornam-se mais evidentes. A latência — ou seja, o atraso entre a captura e a exibição do vídeo — deve ser minimizada para garantir a atualidade das informações, especialmente em aplicações que exigem resposta imediata (JIANG; WANG, 2024). Além disso, a confiabilidade na entrega dos pacotes é essencial para evitar perdas de dados, distorções visuais e falhas na reprodução contínua. Em redes públicas, esses desafios são ampliados por fatores como congestionamento e variação de largura de banda (JIANG; WANG, 2024).

Em aplicações de IoMT, o uso de *streaming* de vídeo adquire um papel ainda mais crítico. A combinação de sensores visuais, conectividade variável e necessidade de análise em tempo real impõe requisitos rigorosos sobre a infraestrutura de transmissão. Nesses cenários, a adoção de arquiteturas distribuídas e técnicas de processamento na borda torna-se indispensável para reduzir a latência, garantir continuidade do serviço e permitir respostas rápidas com base no conteúdo transmitido (AMINIYEGANEH; COUTINHO; BOUKERCHE, 2024).

2.5 Computação de borda

A computação de borda (*Edge Computing*) constitui um paradigma que visa aproximar os recursos de processamento e armazenamento dos pontos de geração de dados, ou seja, posicionando-os na extremidade — ou "borda" — da rede (ROHITH; SUNIL et al., 2021). Em contraste com o modelo tradicional de computação em nuvem, no qual os dados são enviados para centros de dados remotos, a computação de borda propõe uma arquitetura descentralizada, capaz de realizar operações localmente. Essa abordagem é especialmente vantajosa em cenários que exigem baixa latência, alto volume de dados, conectividade limitada e maior controle sobre a privacidade da informação.

Entre os principais benefícios associados à computação de borda está a redução significativa da latência, já que os dados são processados próximos à sua origem, evitando o tempo de ida e volta até a nuvem (ROHITH; SUNIL et al., 2021). Esse aspecto é crucial para aplicações em tempo real, como a análise de vídeo contínuo, em que atrasos na tomada de decisão podem comprometer a eficácia do sistema (JAMIL et al., 2023). Adicionalmente, o processamento local contribui para a diminuição do consumo de largura de banda, uma vez que somente informações relevantes ou agregadas precisam ser transmitidas para servidores centrais (EL-SAYED et al., 2017). Outro fator importante é a preservação da privacidade, visto que dados sensíveis podem ser analisados sem deixar o ambiente de origem.

Por outro lado, a computação na borda impõe desafios próprios. A gestão de múltiplos dispositivos heterogêneos, muitas vezes distribuídos em vários locais, exige mecanismos de orquestração e automação eficientes. A segurança também se torna uma preocupação, já que os dispositivos de borda estão mais expostos a ameaças físicas e cibernéticas, requerendo estratégias robustas de proteção, autenticação e controle de acesso (TU; YANG; CAO, 2025). Além disso, garantir a consistência e a sincronização dos dados entre os diferentes nós da borda e a nuvem demanda protocolos adequados de replicação e atualização.

Mesmo com desafios, a computação de borda tem se mostrado particularmente adequada para aplicações distribuídas que exigem autonomia, resposta rápida e escalabilidade (SUSATYONO; SUASANA; ROZIKIN, 2024). No contexto da IoMT, onde sensores visuais geram grandes volumes de dados em ambientes muitas vezes sujeitos a restrições de rede, a *Edge Computing* oferece uma alternativa viável e estratégica para viabilizar sistemas mais responsivos, resilientes e seguros.

2.6 Internet das Coisas Multimídia (IoMT)

A Internet das Coisas Multimídia (IoMT) representa uma evolução da Internet das Coisas (IoT), com foco na integração de dispositivos capazes de capturar, processar e transmitir conteúdo audiovisual em tempo real, como câmeras, microfones e sensores de

imagem (BATTISTI; MUCHALUAT-SAADE; DELICATO, 2021). Essa vertente amplia significativamente o escopo das aplicações da IoT, ao incorporar dados complexos e de alta dimensionalidade, permitindo o desenvolvimento de sistemas mais sensíveis ao contexto e capazes de tomar decisões baseadas em informações visuais e sonoras (ALVI et al., 2015).

Ao contrário da IoT tradicional, que lida predominantemente com dados estruturados e de baixa taxa de geração (como leituras de temperatura ou umidade), a IoMT lida com fluxos contínuos e volumosos de dados multimídia, exigindo algoritmos eficientes de compressão, transmissão e análise (OVEYSIKIAN, 2024). O processamento de vídeo e áudio em tempo real requer arquiteturas com baixa latência, alta disponibilidade e significativa capacidade computacional (PATEL et al., 2024). Tais requisitos são particularmente críticos em aplicações voltadas à segurança, monitoramento urbano, saúde e automação industrial, onde a agilidade na identificação e resposta a eventos é determinante para o sucesso do sistema.

Entre as principais vantagens da IoMT está a possibilidade de construir sistemas distribuídos e escaláveis, nos quais dispositivos podem ser adicionados ou removidos com relativa facilidade. A combinação de dados visuais com outros sensores contextuais — como temperatura, movimento ou geolocalização — potencializa a criação de aplicações inteligentes e autônomas, com maior capacidade de interpretação do ambiente. Além disso, a automação de tarefas complexas, como reconhecimento facial, contagem de pessoas ou detecção de objetos, permite reduzir a dependência da supervisão humana e aumentar a eficiência operacional (EL-SAYED et al., 2017).

Entretanto, a IoMT também impõe desafios significativos. A transmissão contínua de conteúdo multimídia exige infraestrutura de rede robusta e confiável, com capacidade para lidar com picos de tráfego e variações na qualidade da conexão (MONTAZEROLGHAEM, 2021). A segurança dos dados torna-se ainda mais sensível, considerando que imagens e áudios podem conter informações pessoais ou críticas, exigindo políticas rígidas de criptografia, autenticação e controle de acesso. Além disso, a heterogeneidade de dispositivos e plataformas torna essencial a adoção de padrões abertos e interoperáveis para garantir a integração e o funcionamento correto do ecossistema.

Diante de suas características, a IoMT se apresenta como uma tecnologia estratégica para uma ampla gama de aplicações que requerem sensoriamento multimodal em tempo real. Sua adoção em ambientes inteligentes pode transformar a maneira como os sistemas monitoram, analisam e interagem com o mundo físico, sendo especialmente relevante em contextos nos quais o tempo de resposta, a autonomia e a escalabilidade são requisitos fundamentais (ZIKRIA; AFZAL; KIM, 2020).

3 Trabalhos relacionados

O desenvolvimento de arquiteturas para análise de vídeo em tempo real em ambientes distribuídos de IoMT tem sido foco recorrente na literatura científica, refletindo os desafios crescentes de latência, escalabilidade e autonomia em contextos de borda da rede. Diversos trabalhos propõem abordagens com diferentes graus de modularidade, desacoplamento e orquestração, utilizando estratégias variadas de comunicação e organização funcional. O presente capítulo discute os principais trabalhos relacionados, a partir de uma análise crítica fundamentada em cinco critérios técnicos: (i) presença de processamento multinível, com a separação de tarefas entre diferentes camadas de nós na borda; (ii) uso combinado de comunicação assíncrona com e sem intermediário, ou seja, troca de mensagens tanto diretamente entre serviços quanto por meio de mediadores; (iii) projeto arquitetônico voltado explicitamente à execução na borda da rede, sem dependência de infraestrutura centralizada; (iv) realização de análises de desempenho em diferentes cenários de execução; e (v) adoção do modelo de microsserviços como base para a estrutura modular da aplicação.

O trabalho de Neto et al. (2020) propõe uma arquitetura organizada em três níveis funcionais, com cada tarefa atribuída a diferentes nós de borda, caracterizando um modelo de processamento distribuído hierárquico. A comunicação entre os serviços é realizada exclusivamente por meio de um mediador assíncrono, sem uso de canais diretos. A proposta é modular e voltada à borda, mas limita-se a modelagens teóricas, sem experimentos práticos variados. Rohith, Sunil et al. (2021), por sua vez, apresenta um sistema embarcado de vigilância, com análise de vídeo em tempo real, estruturado de forma monolítica, sem divisão em múltiplas camadas funcionais nem adoção de microsserviços. A comunicação entre módulos não é especificada, e os testes se restringem a um único ambiente operacional.

Curry et al. (2022) introduzem uma arquitetura baseada em processamento neurossimbólico de dados multimodais, com serviços especializados para diferentes tipos de entrada. A implementação ocorre com componentes independentes que trocam dados por meio de eventos e filas, ou seja, comunicação assíncrona com intermediário. Não há, no entanto, uso de canais diretos entre os serviços. A solução é orientada à borda e apresenta uma avaliação detalhada em múltiplos cenários com diferentes configurações de entrada. De forma semelhante, Giao et al. (2022) propõem um *framework* para integração de aplicações IoT em ambientes industriais. Embora permita o desacoplamento de componentes e utilize contêineres, o modelo não adota microsserviços explicitamente, e a comunicação ocorre apenas por meio de mediadores.

Silva et al. (2022) exploram uma extensão de uma arquitetura preexistente, incorporando mecanismos de autoescalonamento baseados em aprendizado de máquina *on-line*,

com foco em adaptação dinâmica conforme a carga de trabalho. O sistema opera em múltiplos níveis e é modularizado, utilizando filas para comunicação assíncrona, mas sem canais diretos entre serviços. A proposta é totalmente voltada à execução na borda e apresenta testes com diferentes conjuntos de dados e estratégias de escalonamento. Por fim, Wang et al. (2023) descrevem uma arquitetura distribuída baseada em computação orientada a funções, com múltiplos níveis de execução em sub-redes e troca de eventos assíncronos com intermediários. Entretanto, a comunicação entre o plano de controle e os módulos é realizada por meio de chamadas síncronas, inviabilizando a caracterização de comunicação assíncrona híbrida.

Diante desse panorama, destaca-se a proposta apresentada neste trabalho, que reúne todas as características avaliadas, com um diferencial importante: adota um modelo de comunicação assíncrona híbrida, com canais diretos entre alguns componentes e troca de mensagens por meio de mediadores entre os demais. Essa combinação estratégica visa otimizar a latência inicial sem comprometer o desacoplamento entre os módulos seguintes. Além disso, a arquitetura foi projetada especificamente para a borda, com ênfase em modularidade, resiliência e facilidade de implantação. A avaliação experimental foi conduzida em múltiplos cenários com diferentes topologias de execução, fornecendo uma análise comparativa realista do comportamento da solução frente ao aumento de carga e complexidade.

A seguir, apresenta-se a Tabela 1, que sintetiza os resultados da comparação entre os trabalhos analisados com base nos cinco critérios descritos anteriormente. A tabela tem como objetivo fornecer uma visão estruturada e objetiva das características observadas em cada proposta, facilitando a identificação dos diferenciais técnicos da arquitetura aqui proposta. As colunas indicam, respectivamente: (i) se o trabalho realiza *processamento multinível* na borda, ou seja, com divisão de tarefas entre diferentes camadas funcionais; (ii) se adota *comunicação assíncrona híbrida*, combinando canais diretos e intermediados; (iii) se foi *projetado para execução na borda*, sem dependência de nuvem; (iv) se contempla *análise de desempenho em diferentes cenários*, com variações de carga ou topologia; e (v) se adota *microsserviços* como modelo de organização arquitetural.

Os dados da Tabela 1 revelam que, embora algumas propostas cumpram individualmente a maioria dos critérios, a arquitetura apresentada neste trabalho tem destaque para sua comunicação assíncrona híbrida, processamento distribuído e validação experimental sob diferentes cenários.

Tabela 1 – Comparativo entre trabalhos relacionados.

Trabalho	**I**	**II**	**III**	**IV**	**V**
Neto et al. (2020)	✓	✗	✓	✗	✓
Rohith, Sunil et al. (2021)	✗	✗	✓	▲	✗
Curry et al. (2022)	✓	✗	✓	✓	✓
Giao et al. (2022)	✓	✗	✓	▲	✗
Silva et al. (2022)	✓	✗	✓	✓	✓
Wang et al. (2023)	✓	✗	✓	✓	✓
Este trabalho	✓	✓	✓	✓	✓

Legenda: ✓= Presente; ✗= Ausente; ▲= Parcial.
I - Processamento multinível na borda; **II** - Comunicação assíncrona (com e sem intermediário);
III - Projetado para uso na borda; **IV** - Análise de desempenho em diferentes cenários;
V - Utiliza microsserviços.

4 ArionStream

Este capítulo apresenta a arquitetura ArionStream, uma solução distribuída projetada para viabilizar a análise de vídeo em tempo real em aplicações de Internet das Coisas Multimídia (IoMT), operando na borda da rede. Motivada pelas crescentes demandas por sistemas capazes de processar fluxos multimídia em ambientes com recursos computacionais restritos e conectividade potencialmente intermitente, a ArionStream foi concebida para oferecer desempenho robusto, flexibilidade e tolerância a falhas.

A proposta adota uma abordagem modular e distribuída, na qual serviços especializados atuam de forma assíncrona para realizar o processamento contínuo de fluxos de vídeo. Essa estrutura permite escalabilidade horizontal, balanceamento dinâmico de carga e desacoplamento entre as etapas do fluxo de processamento, características fundamentais para aplicações multimídia sensíveis à latência e à disponibilidade.

Ao longo deste capítulo, são detalhados os principais elementos que sustentam a arquitetura. Inicia-se com a descrição das funcionalidades essenciais voltadas ao processamento distribuído de vídeo. Em seguida, cada serviço é apresentado individualmente, com ênfase em suas responsabilidades e nas interações com os demais componentes. A topologia geral da arquitetura é então explorada, evidenciando o fluxo de dados e os mecanismos de comunicação adotados. Por fim, discute-se o conjunto de tecnologias de *software* empregadas na implementação da ArionStream, bem como os critérios técnicos que orientaram sua escolha.

4.1 Funcionalidades

A arquitetura ArionStream foi projetada com um conjunto de funcionalidades essenciais para otimizar o processamento e a análise de fluxos de vídeo em aplicações de IoMT na borda da rede. Estas funcionalidades visam atender aos desafios de ambientes com recursos computacionais limitados e requisitos de processamento em tempo real, promovendo a eficiência, a flexibilidade e a escalabilidade do sistema.

Inicialmente, a ArionStream oferece suporte à integração de múltiplas fontes de *stream* de vídeo. Esta capacidade permite que a arquitetura processe fluxos de vídeo provenientes de diversas câmeras ou dispositivos de captura simultaneamente, ampliando o escopo de monitoramento e análise em aplicações complexas. A flexibilidade na integração de diferentes fontes é crucial para aplicações de IoMT que podem envolver uma variedade de sensores e dispositivos multimídia.

Uma funcionalidade chave da arquitetura é o compartilhamento de vídeo entre vários serviços de processamento diferentes. Ao permitir que múltiplos serviços de processamento acessem o mesmo fluxo de vídeo, a arquitetura evita a necessidade de cada serviço

se conectar diretamente à fonte, otimizando o uso da largura de banda e dos recursos do dispositivo de origem. Este compartilhamento de dados é fundamental para o encadeamento de diferentes etapas de processamento e análise.

Visando a eficiência no consumo de recursos dos dispositivos fonte, a ArionStream implementa a redução da sobrecarga no consumo das *streams* de vídeo. Através da abstração da comunicação com as fontes de vídeo por meio de serviços adaptadores, a arquitetura minimiza o impacto no desempenho dos dispositivos de origem, permitindo que eles concentrem sua função principal de captura e transmissão de vídeo para apenas um tipo de serviço.

A arquitetura também suporta o encadeamento de serviços para permitir processamento em múltiplos níveis com a segregação de responsabilidades. Esta funcionalidade possibilita a construção de fluxos de processamento complexos, onde diferentes serviços especializados realizam etapas específicas de análise. A segregação de responsabilidades facilita o desenvolvimento, a manutenção e a reutilização de serviços, promovendo a modularidade e a flexibilidade no desenvolvimento e implantação.

Para otimizar o tempo de transmissão e reduzir a latência, a arquitetura emprega o uso de conexões diretas e assíncronas entre serviços de vídeo. A comunicação direta entre esses serviços utiliza protocolos eficientes e comunicação assíncrona, para minimizar a sobrecarga e agilizar o fluxo de dados de vídeo, sem o uso de serviços intermediários.

Finalmente, após o processamento dos vídeos, os dados resultantes são organizados no formato de tópicos, permitindo que os serviços posteriores possam filtrar somente o que for de interesse. Esta abordagem baseada em tópicos no sistema de mensageria facilita a comunicação seletiva e eficiente entre os serviços de análise e atuação, otimizando o consumo de recursos e a tomada de decisões baseada nos dados processados.

4.2 Serviços

A arquitetura é composta por um conjunto de serviços especializados e interdependentes, cada um com uma responsabilidade bem definida dentro do fluxo de processamento e análise de vídeo. A modularidade da arquitetura, baseada nesses serviços distintos, contribui para a flexibilidade, a escalabilidade e a manutenibilidade do sistema. Os principais serviços que integram a ArionStream são descritos a seguir:

1. **Adaptador de *Stream* de Vídeo (ASV):** atua como um serviço intermediário fundamental, responsável por abstrair a comunicação com as diversas fontes de *stream* de vídeo. Dada a variedade de protocolos e formatos utilizados pelas diferentes câmeras e dispositivos de captura (como RTSP e HTTP), o ASV tem a função de padronizar a interface de comunicação para os demais serviços da arquitetura. Além disso, o ASV é responsável por realizar o *broadcast* do fluxo de vídeo para os Processadores de *Stream* de Vídeo (PSVs) conectados, evitando sobrecarregar o

dispositivo fonte com múltiplas conexões e prevenindo a necessidade de reimplementação da lógica de comunicação com a fonte em cada serviço de processamento.

2. **Processador de *Stream* de Vídeo (PSV):** é o serviço encarregado de realizar o processamento primário no fluxo de vídeo recebido. Um PSV pode se conectar a um ASV, ou até mesmo a outro PSV, permitindo a criação de fluxos de processamento em múltiplos estágios. Após a execução de suas tarefas de processamento (que podem incluir desde a aplicação de filtros e algoritmos de visão computacional até a detecção de eventos específicos), o PSV encaminha os dados resultantes para um serviço intermediário de mensagens, garantindo a comunicação assíncrona com os serviços subsequentes.

3. **Agente/*broker* de mensagens, ou serviço de mensageria:** desempenha um papel crucial na arquitetura ArionStream, atuando como um serviço auxiliar intermediário que facilita a troca de mensagens entre os diversos serviços de maneira assíncrona e desacoplada. Ao invés de conexões diretas ponto a ponto, os serviços comunicam-se através do agente de mensagens, que é responsável por rotear as mensagens para as filas ou tópicos apropriados. Essa abordagem promove a resiliência e a escalabilidade do sistema, permitindo que os serviços operem de forma independente e que novos serviços sejam adicionados ou removidos sem afetar a comunicação dos demais.

4. **Classificador:** é o serviço responsável por atribuir categorias ou rótulos aos dados processados pelos PSVs, com base em regras de negócio definidas conforme as necessidades da aplicação. Essas regras podem ser simples ou complexas, permitindo flexibilidade para que diferentes critérios de análise sejam adotados, conforme o contexto em que a arquitetura estiver sendo utilizada. Ao receber os dados do serviço de mensageria, o Classificador aplica a lógica estabelecida e publica os resultados em novos tópicos, que serão consumidos por serviços subsequentes. Essa etapa é essencial para a filtragem e roteamento das informações relevantes no fluxo de processamento.

5. **Atuador:** é um serviço especializado responsável por executar ações específicas com base nos dados categorizados pelos Classificadores. Um Atuador pode se inscrever em tópicos específicos do serviço de mensageria, filtrando apenas os dados de seu interesse. Essa especialização permite a criação de agentes autônomos capazes de reagir a eventos detectados no fluxo de vídeo, automatizando respostas e ações em tempo real.

4.3 Topologia

A topologia da arquitetura define a organização e o fluxo de comunicação entre os diversos serviços, desde a captura do vídeo nas fontes até a execução de ações pelos atuadores. A arquitetura foi projetada para ser flexível e adaptável a diferentes cenários de implantação, mantendo a eficiência e a baixa latência como prioridades. A Figura 1 ilustra as possíveis distribuições dos serviços e fluxos de dados entre os componentes.

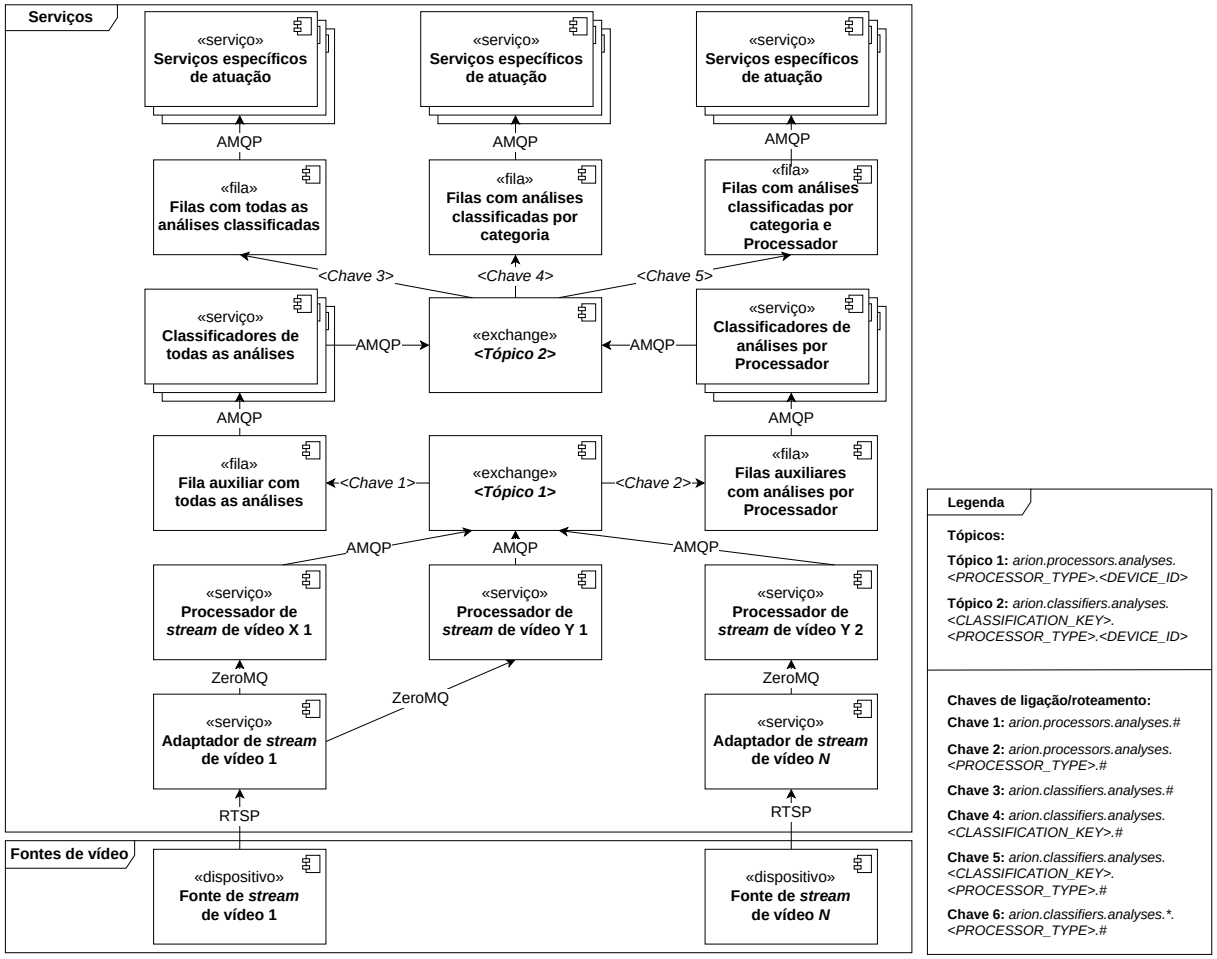

Figura 1 – Topologia da arquitetura ArionStream, ilustrando possibilidades para distribuição dos serviços e fluxos de dados entre os componentes.

As fontes de *stream* de vídeo podem utilizar diferentes protocolos para fornecer o fluxo de dados, como o RTSP e o HTTP. Para lidar com essa heterogeneidade e fornecer uma interface de comunicação unificada para os demais serviços, a arquitetura introduz a camada dos Adaptadores de *Stream* de Vídeo (ASVs). Os ASVs atuam como pontos de entrada para os fluxos de vídeo, abstraindo as particularidades de cada protocolo e fornecendo um padrão de comunicação consistente para os Processadores de *Stream* de Vídeo (PSVs). Essa padronização simplifica o desenvolvimento e a integração de novos serviços de processamento.

A comunicação entre os ASVs e os PSVs é estabelecida de maneira assíncrona, empregando um modelo de publicação/subscrição (*publish/subscribe*). Nessa interação, os PSVs

se conectam diretamente aos ASVs, manifestando seu interesse em receber os fluxos de vídeo. Os ASVs, por sua vez, captam as imagens das fontes de vídeo e realizam *broadcast* dessas imagens para todos os PSVs conectados. Essa comunicação direta e sem intermediários para o fluxo de vídeo visa minimizar a latência na etapa inicial de processamento. Para otimizar o uso da rede, as imagens são comprimidas antes do envio pelo ASV e armazenadas em um *buffer* de memória no PSV ao serem recebidas. A descompressão ocorre apenas no momento em que a imagem é efetivamente necessária para o processamento, reduzindo o consumo de largura de banda e a utilização de memória pelo *buffer* do PSV.

Uma vez processadas pelos PSVs, as informações resultantes (metadados da análise, eventos detectados, etc.) são encaminhadas para o serviço de mensageria através de tópicos (referenciado como Tópico 1 na Figura 1). O serviço de mensageria atua como um ponto central para a comunicação assíncrona entre os serviços que não precisam de interação direta, também sendo responsável por rotear as mensagens para filas, para que possam ser consumidas por serviços interessados. Algumas filas podem receber todos os dados encaminhados pelos PSVs, enquanto outras podem ser especializadas para receber dados de PSVs específicos, oferecendo flexibilidade no consumo da informação.

Os serviços de Classificação consomem os dados das filas do serviço de mensageria. Esses serviços podem aplicar regras de negócio para categorizar, enriquecer, modificar ou até mesmo descartar os dados recebidos. Após a classificação, os novos dados são publicados em outro tópico do serviço de mensageria (Tópico 2 na Figura 1), tornando-se disponíveis para outros serviços tomarem decisões ou executarem ações.

De forma semelhante aos Classificadores, os Atuadores também se conectam ao serviço de mensageria para receber dados. Eles podem se inscrever em todos os dados publicados em um determinado tópico ou filtrar para consumir apenas as informações que lhes são relevantes. Essa capacidade de filtragem permite a criação de atuadores altamente especializados e a separação clara da lógica de ação com base nos resultados da análise de vídeo.

4.4 Tecnologias utilizadas

A implementação da arquitetura ArionStream envolveu a seleção de um conjunto de tecnologias de *software* que se mostraram adequadas para atender aos requisitos de desempenho, flexibilidade e escalabilidade do sistema. A escolha de cada tecnologia foi motivada por suas características específicas, seu ecossistema de bibliotecas e documentação, e sua adoção pela comunidade de desenvolvedores.

Para a implementação dos Adaptadores de *Stream* de Vídeo (ASVs) e dos Processadores de *Stream* de Vídeo (PSVs), a linguagem de programação Python foi escolhida. Python se destaca por seu rico ecossistema de bibliotecas para processamento de vídeo e computação numérica, além de sua sintaxe clara e facilidade de desenvolvimento. Especificamente,

os ASVs utilizam as bibliotecas OpenCV e VidGear para abstrair a comunicação com as diversas fontes de *stream* de vídeo, oferecendo suporte a diferentes protocolos e formatos de vídeo de maneira eficiente.

A comunicação direta e assíncrona entre os serviços de ASVs e PSVs foi implementada utilizando as bibliotecas ZeroMQ e ImageZMQ. ZeroMQ é uma biblioteca de mensagens de alto desempenho que permite a criação de padrões de mensagens flexíveis, incluindo o modelo de *publish/subscribe* utilizado para a transmissão de vídeo dos ASVs para os PSVs. ImageZMQ é uma extensão do ZeroMQ otimizada para o transporte eficiente de imagens, o que é crucial para minimizar a latência na transferência dos quadros de vídeo. Essa comunicação direta evita a sobrecarga de um serviço intermediário de mensagens para o fluxo de vídeo bruto.

Para o serviço de mensageria da arquitetura ArionStream foi selecionado o RabbitMQ, um *broker* de mensagens amplamente utilizado e que oferece suporte ao protocolo AMQP. O RabbitMQ foi escolhido por sua robustez, escalabilidade e suporte a funcionalidades avançadas de roteamento de mensagens, como tópicos e chaves de roteamento, que permitem a implementação de um sistema de mensagens flexível e granular para a comunicação entre os PSVs, Classificadores e Atuadores.

A comunicação para o envio de dados processados pelos PSVs e Classificadores para o serviço de mensageria, assim como o recebimento desses dados pelos Classificadores e Atuadores, é realizada através do protocolo AMQP, utilizando a biblioteca cliente correspondente para interagir com o RabbitMQ. O uso de tópicos e chaves de roteamento permite que os serviços consumidores (Classificadores e Atuadores) filtrem apenas as mensagens de seu interesse, otimizando o processamento e evitando a sobrecarga com dados irrelevantes.

Finalmente, o Docker foi utilizado para isolar e empacotar todos os serviços da arquitetura ArionStream em contêineres. A contêinerização facilita a implantação, o gerenciamento e a escalabilidade dos serviços, além de garantir a portabilidade da arquitetura para diferentes ambientes de execução. O Docker, juntamente com seu ferramental, permite a criação de uma infraestrutura de implantação simples e consistente, essencial para a prototipagem e a eventual implantação em ambientes de produção.

5 Experimentos e análise dos resultados

Este capítulo apresenta a metodologia experimental adotada para avaliar o desempenho e as características da arquitetura ArionStream em diferentes cenários de uso. O objetivo principal dos experimentos é validar a capacidade da arquitetura em processar fluxos de vídeo em tempo real, analisar sua latência sob diferentes configurações de carga e verificar sua escalabilidade em um ambiente de rede local.

Para atingir esse objetivo, foram definidos um conjunto de métricas de avaliação relevantes para aplicações de análise de vídeo distribuída, incluindo a latência entre os serviços, escalabilidade e taxa de perda de dados. Além disso, foram criados cinco cenários de teste distintos, cada um representando uma configuração específica da arquitetura com diferentes quantidades de serviços, simulando assim diversas demandas de processamento.

Os experimentos foram conduzidos em um ambiente de execução controlado, utilizando um *cluster* com Docker Swarm composto por duas e três máquinas conectadas em rede local via *Wi-Fi*. A coleta e a análise dos resultados obtidos nesses experimentos permitirão uma avaliação quantitativa do desempenho da arquitetura ArionStream e fornecerão entendimentos valiosos sobre suas potencialidades e limitações em diferentes condições operacionais. As seções subsequentes detalharão as métricas avaliadas, os cenários de teste implementados, o ambiente de execução utilizado e, finalmente, a apresentação e análise dos resultados obtidos.

5.1 Métricas avaliadas

Para avaliar o desempenho e as características da arquitetura ArionStream nos experimentos realizados, foram selecionadas as seguintes métricas quantitativas, que fornecem uma base para a análise da sua eficiência, capacidade de resposta e robustez em diferentes condições operacionais:

1. **Latência entre serviços ($L_{\textbf{serviços}}$):** Esta métrica quantifica o intervalo de tempo decorrido desde o momento em que um dado ou imagem é enviado de um serviço até o momento em que é recebido pelo serviço subsequente. A avaliação dessa latência em cada estágio da arquitetura permite identificar possíveis gargalos de comunicação e a eficiência da transferência de dados entre os componentes. As fórmulas específicas para cada medição são:

    - **Latência de recebimento de imagem pelo PSV ($L_{\textbf{ASV}\rightarrow\textbf{PSV}}$):** Representa o tempo que a imagem leva para ir do Adaptador de *Stream* de Vídeo (ASV)

até o Processador de *Stream* de Vídeo (PSV):

$$L_{\text{ASV}\rightarrow\text{PSV}} = T_{\text{recebimento\_PSV}} - T_{\text{envio\_ASV}} \tag{5.1}$$

Onde:

- $T_{\text{envio\_ASV}}$: Horário em que a imagem foi enviada pelo Adaptador de *Stream* de Vídeo.
- $T_{\text{recebimento\_PSV}}$: Horário em que a imagem foi recebida pelo Processador de *Stream* de Vídeo.

- **Latência de recebimento de dado pelo Classificador ($L_{\textbf{PSV}\rightarrow\textbf{Classificador}}$):** Mede o tempo que o dado processado leva para ir do PSV até o Classificador:

$$L_{\text{PSV}\rightarrow\text{Classificador}} = T_{\text{recebimento\_Classificador}} - T_{\text{envio\_PSV}} \tag{5.2}$$

Onde:

- $T_{\text{envio\_PSV}}$: Horário em que o dado processado foi enviado pelo PSV.
- $T_{\text{recebimento\_Classificador}}$: Horário em que o dado foi recebido pelo Classificador.

- **Latência de recebimento de dado pelo Atuador ($L_{\textbf{Classificador}\rightarrow\textbf{Atuador}}$):** Indica o tempo que o dado classificado leva para ir do Classificador até o Atuador:

$$L_{\text{Classificador}\rightarrow\text{Atuador}} = T_{\text{recebimento\_Atuador}} - T_{\text{envio\_Classificador}} \tag{5.3}$$

Onde:

- $T_{\text{envio\_Classificador}}$: Horário em que o dado classificado foi enviado pelo Classificador.
- $T_{\text{recebimento\_Atuador}}$: Horário em que o dado classificado foi recebido pelo Atuador.

2. **Escalabilidade:** A escalabilidade da arquitetura ArionStream foi avaliada por meio da observação do comportamento das métricas de latência sob diferentes cenários de teste e variações no ambiente de execução (com diferentes quantidades de máquinas). O objetivo é verificar a capacidade da arquitetura de manter um desempenho aceitável ou de escalar seus recursos de forma eficiente à medida que a carga de trabalho e o número de componentes do sistema aumentam.

3. **Taxa de perda de dados ($P_{\textbf{perda}}$):** Esta métrica quantifica a proporção de dados (imagens ou ocorrências) que são perdidos durante a transmissão entre os componentes da arquitetura. É valiosa para verificar a robustez do sistema, especialmente sob condições de rede instáveis ou em cenários de alta demanda:

$$P_{\text{perda}}(\%) = \left(\frac{N_{\text{enviados}} - N_{\text{recebidos}}}{N_{\text{enviados}}}\right) \times 100 \tag{5.4}$$

Onde:

- N_{enviados}: Número total de dados enviados pelo serviço emissor.
- $N_{\text{recebidos}}$: Número total de dados recebidos pelo serviço receptor.

Nos cenários de teste conduzidos em ambiente laboratorial controlado, espera-se que essa métrica apresente valores favoráveis para a arquitetura, indicando uma baixa taxa de perda de dados.

5.2 Cenários de testes

Para uma avaliação abrangente da arquitetura ArionStream, foram concebidos cinco cenários de testes distintos, cada um configurado com diferentes quantidades de serviços (Adaptadores de *Stream* de Vídeo, Processadores de *Stream* de Vídeo, Classificadores e Atuadores). O objetivo desta variedade de configurações é simular ambientes de aplicação reais com diferentes demandas de processamento e analisar o comportamento da arquitetura sob essas distintas condições. A Tabela 2 apresenta um resumo das configurações de cada cenário:

Tabela 2 – Configurações dos cenários de testes com as respectivas quantidades de serviços por tipo.

Cenário	ASVs	PSVs	Classificadores	Atuadores	Relação ASV-PSV
Cenário 1	1	1	1	1	1 ASV para 1 PSV
Cenário 2	3	3	3	3	1 ASV para 1 PSV
Cenário 3	5	5	5	5	1 ASV para 1 PSV
Cenário 4	5	10	5	5	1 ASV para 2 PSVs
Cenário 5	5	15	5	5	1 ASV para 3 PSVs

Em cada cenário de teste, cada serviço de Adaptador de *Stream* de Vídeo (ASV) foi configurado para fornecer um fluxo de vídeo simulado contendo 1000 imagens com resolução de 1080x720 *pixels*. As imagens utilizadas eram sintéticas, geradas programaticamente, do tipo RGB e inteiramente preenchidas com a cor preta. Essa abordagem visou isolar variáveis relacionadas ao conteúdo visual e garantir a repetibilidade dos experimentos, focando exclusivamente na avaliação do desempenho arquitetural e comportamento dos serviços sob carga de dados constante.

A taxa de envio das imagens pelos serviços de ASV foi limitada a 10 quadros por segundo (*FPS*). Essa limitação foi imposta para controlar a taxa de geração de dados e permitir uma análise mais precisa do desempenho da arquitetura sob uma carga de trabalho definida.

É importante ressaltar que os serviços de Processador de *Stream* de Vídeo não realizaram nenhum processamento computacionalmente intensivo sobre o fluxo de vídeo. Em vez disso, cada PSV foi configurado para simular um intervalo de processamento aleatório

entre 50 milissegundos e 150 milissegundos por quadro recebido. Essa simulação tem como objetivo emular a latência introduzida por um algoritmo de processamento de vídeo real, sem focar na complexidade específica desse algoritmo, que foge do escopo deste trabalho. O foco principal dos experimentos é avaliar a latência de transferência e a escalabilidade da arquitetura em si, sob diferentes configurações de serviços.

5.3 Ambiente de execução

A fim de conduzir os experimentos de avaliação da arquitetura ArionStream de maneira controlada e replicável, foi estabelecido um ambiente de execução específico, composto por recursos de *hardware* e *software* dedicados.

Para a correta operação dos experimentos e a coleta precisa das métricas, foi necessária a criação e utilização de serviços auxiliares: um serviço de tempo centralizado e um serviço de banco de dados. O serviço de tempo foi implementado para fornecer uma referência temporal para todos os serviços da arquitetura, garantindo a precisão na medição das latências. O serviço de banco de dados foi utilizado para o armazenamento dos dados coletados durante as execuções dos testes, facilitando a posterior análise dos resultados.

Os experimentos foram realizados utilizando um total de três máquinas físicas interconectadas por meio de uma rede local com conexão *Wi-Fi*. As especificações de *hardware* e *software* de cada uma dessas máquinas são detalhadas na Tabela 3. Para investigar o impacto da quantidade de recursos computacionais no desempenho da arquitetura, foram conduzidas duas execuções completas dos cinco cenários de teste descritos na seção anterior. Na primeira execução, foram utilizadas somente duas máquinas, enquanto na segunda execução, todas as três máquinas foram empregadas. Essa abordagem permitiu verificar as possíveis vantagens ou desvantagens decorrentes da adição de mais recursos de *hardware* ao ambiente de execução.

Tabela 3 – Especificações de *hardware* e *software* das máquinas de execução.

Máquina	Processador	Núcleos lógicos	Memória RAM	Sistema operacional	Versão do Docker	Nó de gerenciamento
Máquina 1	AMD Ryzen 5 5500U	12	12 GB DDR4	ArchLinux	28.1.2001	Sim
Máquina 2	AMD Ryzen 5 5500U	12	8 GB DDR4	Kubuntu 22.04 LTS	28.1.2001	Não
Máquina 3	Intel Core i3-7020U	4	16 GB DDR4	Kubuntu 24.04 LTS	28.1.2001	Não

As máquinas foram configuradas para formar um *cluster* utilizando o Docker Swarm. O Docker Swarm orquestrou a distribuição e o gerenciamento dos contêineres dos serviços da arquitetura ArionStream entre as máquinas disponíveis. Em ambas as execuções, a Máquina 1 (conforme especificado na Tabela 3) foi designada como o nó de gerenciamento (*manager*) do *cluster* com Docker Swarm. Os serviços auxiliares (serviço de tempo e serviço de banco de dados) foram intencionalmente restringidos para serem executados exclusivamente na Máquina 1. Essa decisão visou centralizar a aquisição do horário e o armazenamento dos dados das execuções, mantendo um ambiente mais controlado para a coleta das métricas.

Durante a configuração do *cluster* Docker Swarm, não foram definidos critérios específicos de alocação (*placement*) para os contêineres dos serviços da arquitetura ArionStream. Essa escolha deliberada teve como objetivo observar o comportamento da arquitetura em um contexto onde a distribuição dos serviços entre os nós do *cluster* é realizada de forma automática e equilibrada pelo Docker Swarm, simulando cenários com distribuições de carga potencialmente imprevisíveis. A intenção foi avaliar a resiliência e a adaptabilidade da arquitetura em condições onde o controle granular sobre a alocação de recursos não é explicitamente definido.

5.4 Resultados

Esta seção apresenta os resultados obtidos a partir das execuções experimentais da arquitetura ArionStream nos diferentes cenários propostos. As análises foram organizadas em duas subseções, conforme a configuração do ambiente de execução: com duas máquinas (Subseção 5.4.1) e com três máquinas (Subseção 5.4.2). Em cada caso, são exploradas as latências dos serviços em diferentes estágios do pipeline (Processadores de *Stream* de Vídeo, Classificadores e Atuadores), considerando medidas estatísticas relevantes, histogramas e séries temporais. Os resultados servem de base para a interpretação do comportamento da arquitetura sob diferentes níveis de carga, e fundamentam as discussões realizadas na análise crítica subsequente.

5.4.1 Execução com 2 máquinas

A Figura 2 apresenta os histogramas das latências registradas para os três tipos de serviço da arquitetura ArionStream — Processadores de *Stream* de Vídeo, Classificadores e Atuadores — agregando os dados de todos os cenários na execução com duas máquinas. Os Processadores de *Stream* de Vídeo exibem uma distribuição assimétrica acentuada à direita, com forte concentração de latências abaixo de 20.000 ms, mas com uma cauda longa que se estende além de 70.000 ms, indicando a ocorrência de valores extremos e alta variabilidade, típicas de picos ocasionais de sobrecarga. Os Classificadores, por sua vez, apresentam uma distribuição mais concentrada e quase simétrica, com maior densidade entre 70 ms e 160 ms, refletindo estabilidade no tempo de resposta, ainda que com pequenas flutuações periódicas visíveis no padrão de contagem. Já os Atuadores demonstram latências significativamente menores, com concentração elevada de ocorrências abaixo de 30 ms e ausência de caudas longas, o que evidencia sua natureza leve e a baixa carga de processamento associada à etapa final do fluxo de execução.

A Figura 3 apresenta os histogramas das latências por tipo de serviço — Processadores de *Stream* de Vídeo, Classificadores e Atuadores — segmentados por cenário, na execução com duas máquinas. Nos Processadores de *Stream* de Vídeo (PSVs), observa-se que a distribuição das latências se desloca gradualmente para valores mais altos à medida que o

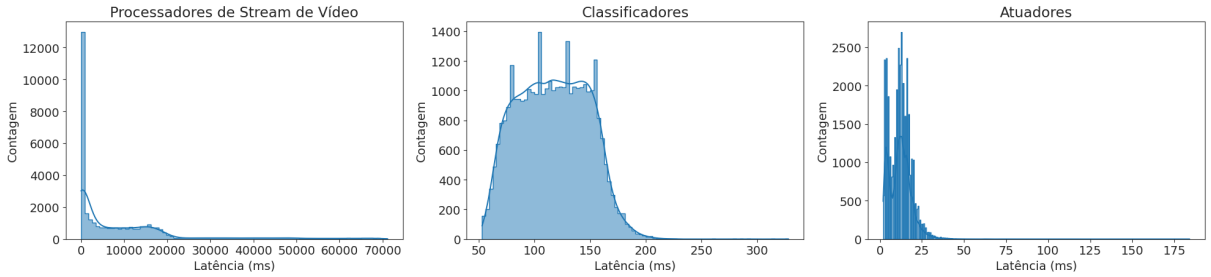

Figura 2 – Histogramas com as latências de todos os serviços por tipo (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) em todos os cenários na execução com 2 máquinas.

cenário se torna mais complexo. O Cenário 5, em particular, apresenta uma concentração de latências muito acima dos demais, com forte assimetria à direita e cauda longa, o que indica que a carga crescente afeta diretamente este estágio do fluxo de execução. Nos Classificadores, o padrão é mais regular: as distribuições mantêm-se concentradas entre 70 ms e 160 ms, com um leve deslocamento e alargamento da curva conforme a complexidade do cenário aumenta, o que sugere impacto moderado e previsível da carga. Por fim, os Atuadores exibem histogramas bastante similares entre os cenários, com a maioria das latências concentrada abaixo de 30 ms, evidenciando que este serviço permanece estável mesmo sob maior volume de dados. A figura reforça que, durante a execução, os efeitos da carga se manifestam principalmente nos PSVs, enquanto os demais serviços preservam desempenho consistente e controlado.

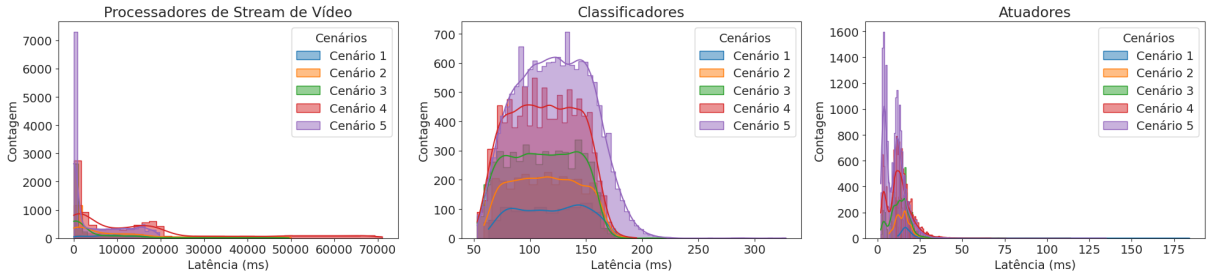

Figura 3 – Histogramas com as latências de todos os serviços por tipo (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) e cenário na execução com 2 máquinas.

A Figura 4 apresenta a evolução das latências máximas, médias e medianas por imagem processada nos serviços da arquitetura — Processadores de *Stream* de Vídeo, Classificadores e Atuadores — durante o Cenário 1 com duas máquinas. Nos Processadores de *Stream* de Vídeo, observa-se um crescimento quase linear em todas as métricas, sugerindo acúmulo gradual de latência ao longo da execução, possivelmente devido à formação de filas de processamento sob carga contínua. As latências ultrapassam 17.000 ms mesmo neste cenário de baixa complexidade, evidenciando a sensibilidade desse serviço ao tempo acumulado. Nos Classificadores, o comportamento é substancialmente mais estável: tanto as latências máximas quanto as médias e medianas permanecem dentro de um intervalo restrito, com pequenas oscilações e sem tendência de crescimento ao longo das imagens, o que demonstra uma execução equilibrada neste estágio intermediário. Já nos Atuadores,

os valores de latência são os mais baixos entre os três serviços e se mantêm praticamente constantes, com exceção de alguns picos iniciais que rapidamente se estabilizam. A figura demonstra que, no Cenário 1 com duas máquinas, a arquitetura consegue manter baixa variabilidade e responsividade nos serviços intermediários e finais, enquanto os PSVs já apresentam indícios de acúmulo progressivo, mesmo sob baixa carga.

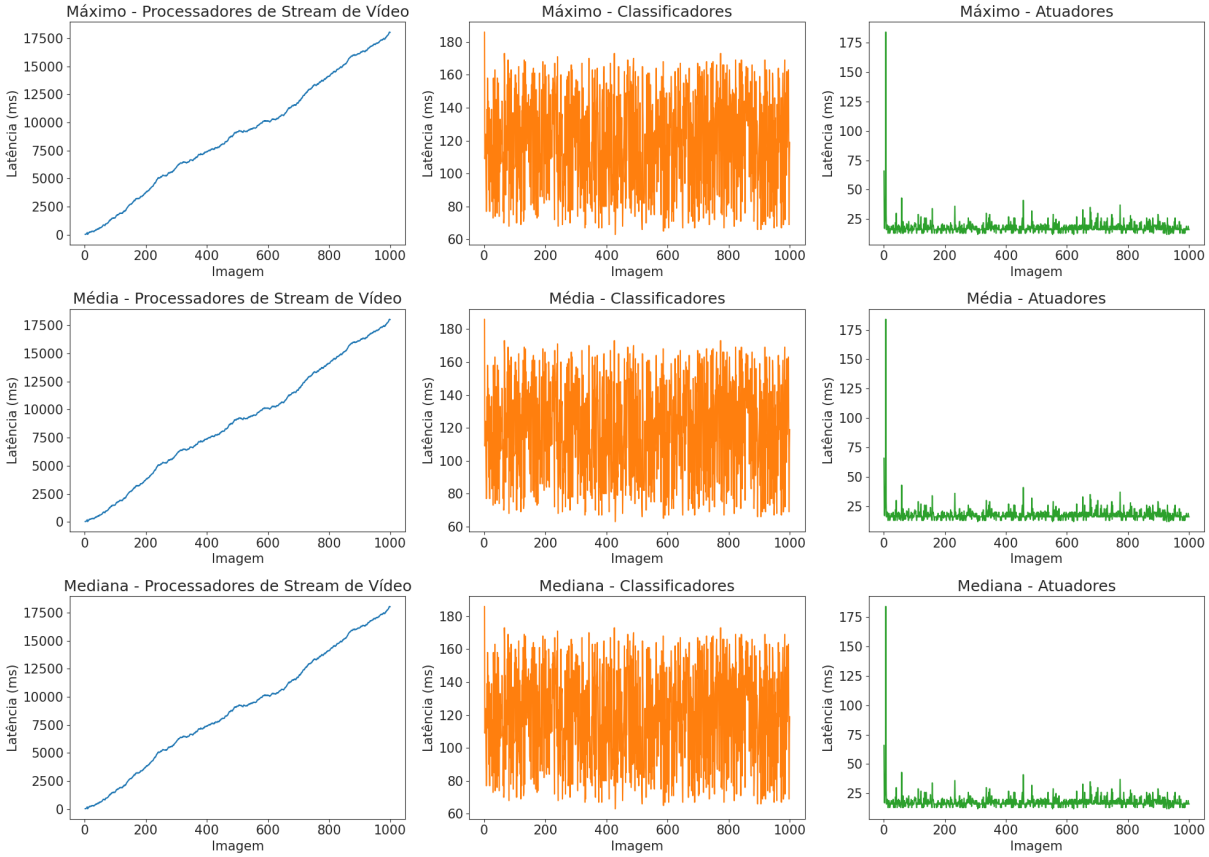

Figura 4 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 1 da execução com 2 máquinas.

A Figura 5 ilustra a evolução das latências máximas, médias e medianas por imagem para os três tipos de serviço da arquitetura no Cenário 2 com duas máquinas. Nos Processadores de *Stream* de Vídeo, observa-se um crescimento acentuado e praticamente linear nas três métricas, atingindo mais de 50.000 ms nas latências máximas. Este padrão evidencia um acúmulo progressivo de carga, com aumento significativo da latência à medida que mais imagens são processadas, mesmo com a execução distribuída entre dois nós. Nos Classificadores, o comportamento permanece estável em comparação ao cenário anterior, com latências medianas entre 80 ms e 160 ms, e oscilações frequentes, porém em um intervalo previsível. Não há tendência de crescimento, o que indica que a sobrecarga gerada pelos PSVs não está se propagando de forma significativa para este serviço intermediário. Por fim, os Atuadores mantêm latências médias e medianas abaixo de 25 ms, ainda que com maior dispersão em comparação ao Cenário 1, especialmente nas latências máximas, que apresentam picos esporádicos acima de 70 ms. Esses resultados reforçam que o Cená-

rio 2 impõe uma carga mais relevante aos PSVs, mas os demais serviços seguem operando com desempenho consistente.

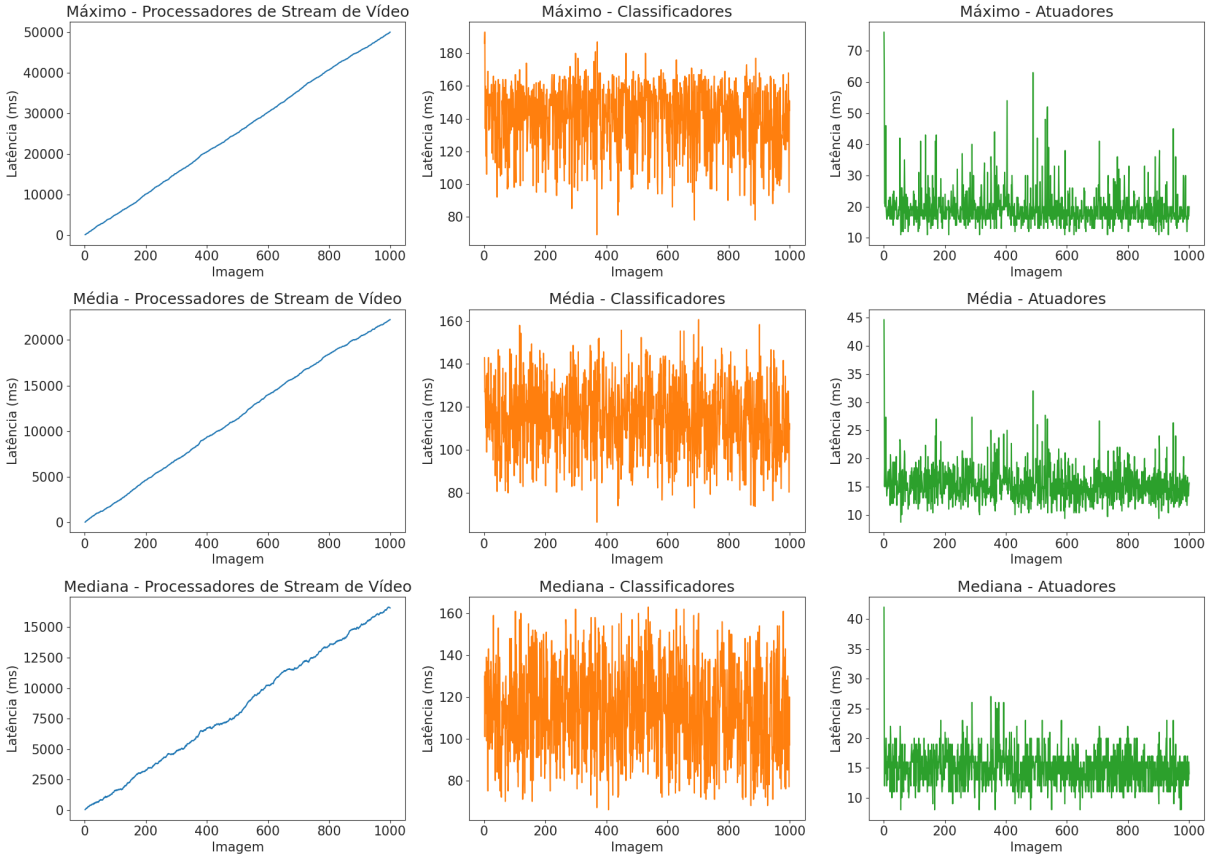

Figura 5 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 2 da execução com 2 máquinas.

A Figura 6 apresenta a evolução das latências máximas, médias e medianas por imagem trafegada nos serviços da arquitetura durante o Cenário 3, com execução em duas máquinas. Nos Processadores de *Stream* de Vídeo, observa-se uma melhoria significativa em comparação ao Cenário 2, especialmente na latência mediana, que se mantém abaixo de 2.500 ms e com crescimento oscilatório moderado. Ainda que as latências máximas e médias apresentem crescimento contínuo, seus valores são consideravelmente inferiores aos observados no cenário anterior, sugerindo uma redistribuição de carga mais eficiente neste arranjo. Isso pode indicar que a maneira como o Docker Swarm distribuiu as cargas pode ter resultado em uma otimização temporária, proporcionando melhor balanceamento de tarefas ou reduzindo gargalos internos. Nos Classificadores, os padrões continuam estáveis: tanto máximos quanto médios e medianos oscilam em torno de 100 a 160 ms, com ausência de tendência de acúmulo ao longo do tempo. Os Atuadores, por sua vez, mantêm desempenho consistente, com latências majoritariamente abaixo de 30 ms e pequenas variações pontuais. Em conjunto, os resultados sugerem que o Cenário 3 apresentou ganhos operacionais importantes nos PSVs, tornando-se mais eficiente que o cenário anterior mesmo com maior número de componentes, destacando o papel da orquestração no desempenho

geral da arquitetura.

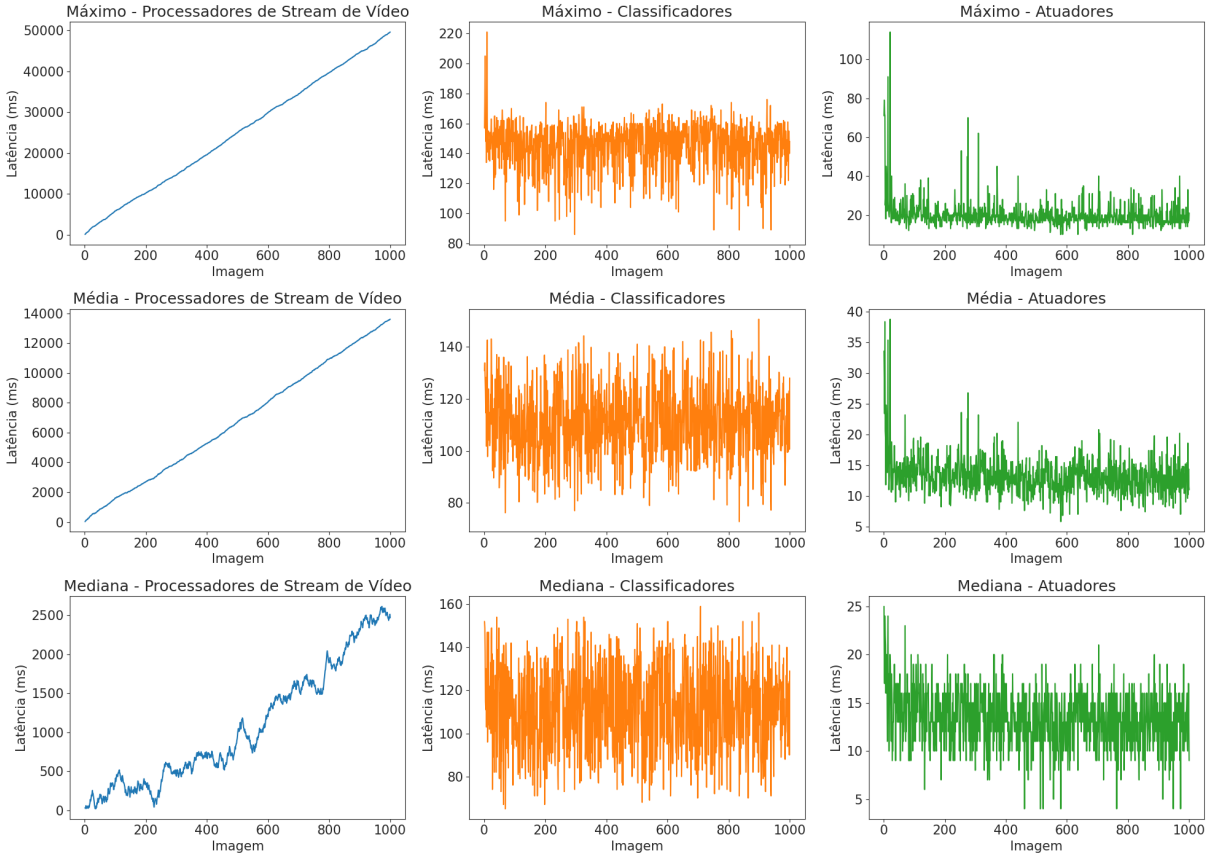

Figura 6 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 3 da execução com 2 máquinas.

A Figura 7 apresenta a evolução das latências máximas, médias e medianas por imagem para os três tipos de serviço da arquitetura durante o Cenário 4 com duas máquinas. Nos Processadores de *Stream* de Vídeo, observa-se um retorno ao padrão de crescimento acentuado nas três métricas, com as latências máximas ultrapassando 70.000 ms e a mediana atingindo 20.000 ms, indicando que o aumento no número de serviços neste cenário sobrecarregou novamente este estágio. Esse comportamento representa um contraste direto com o Cenário 3, onde havia sido registrada uma melhora relativa nos PSVs; aqui, o crescimento progressivo de carga volta a exercer impacto negativo. Nos Classificadores, as latências mantêm estabilidade, com oscilações regulares entre 100 ms e 160 ms e sem tendência de crescimento, reforçando a robustez deste serviço intermediário. Já os Atuadores seguem apresentando baixas latências, com médias e medianas abaixo de 25 ms e picos máximos inferiores a 70 ms, mesmo sob maior carga.

A Figura 8 apresenta a evolução das latências máximas, médias e medianas por imagem para os três tipos de serviço da arquitetura no Cenário 5, o mais complexo da execução com duas máquinas. Surpreendentemente, os Processadores de *Stream* de Vídeo apresentam desempenho consideravelmente melhor em comparação ao Cenário 4, com latências máximas limitadas a aproximadamente 20.000 ms e medianas abaixo de 15.000 ms —

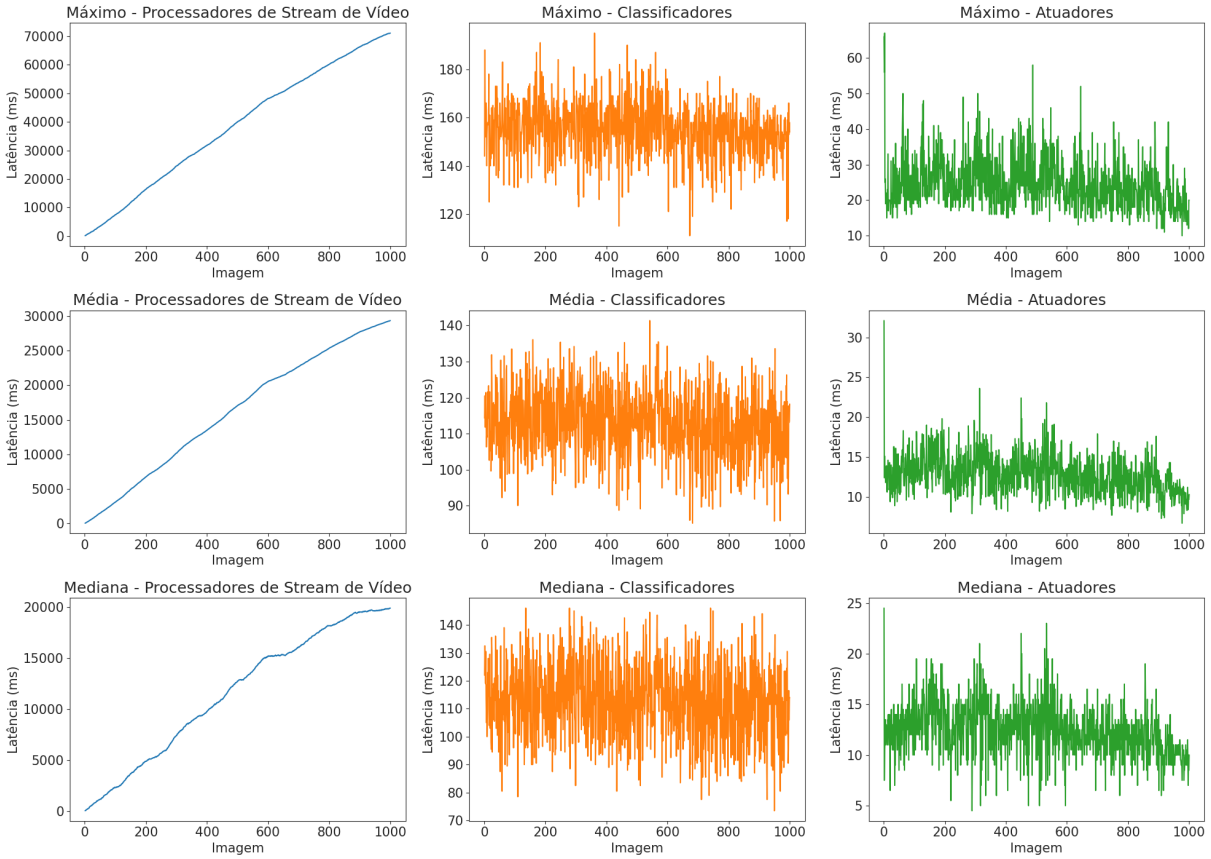

Figura 7 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 4 da execução com 2 máquinas.

valores inferiores aos observados em cenários anteriores com menor complexidade. Esse comportamento quebra a tendência esperada de degradação progressiva, sugerindo que o Docker Swarm, ao distribuir uma maior quantidade de serviços, pode ter alcançado um equilíbrio mais eficiente na utilização dos recursos disponíveis. Nos Classificadores, o padrão permanece estável, com latências variando entre 120 ms e 160 ms, sem sinais de sobrecarga, embora o cenário registre alguns picos esporádicos, inclusive acima de 300 ms. Os Atuadores mantêm comportamento previsível e eficiente, com latências medianas próximas de 15 ms e baixos níveis de variação. A figura reforça que, apesar da carga máxima neste cenário, o sistema conseguiu atingir uma melhor eficiência operacional nos PSVs, evidenciando que o desempenho da arquitetura não está vinculado unicamente à quantidade de serviços, mas também à forma como eles são alocados e distribuídos entre os nós disponíveis.

A Tabela 4 apresenta os dados estatísticos das latências agregadas pelo valor máximo de cada imagem trafegada entre os serviços da arquitetura (PSVs, Classificadores e Atuadores), durante os testes com duas máquinas. As métricas descritas incluem valores mínimos, máximos, média, desvio padrão e percentis (P25, P50, P75, P90, P95 e P99), permitindo uma análise detalhada da variabilidade e dos picos de latência. Observa-se que os Processadores de *Stream* de Vídeo (PSVs) concentram os maiores valores médios e

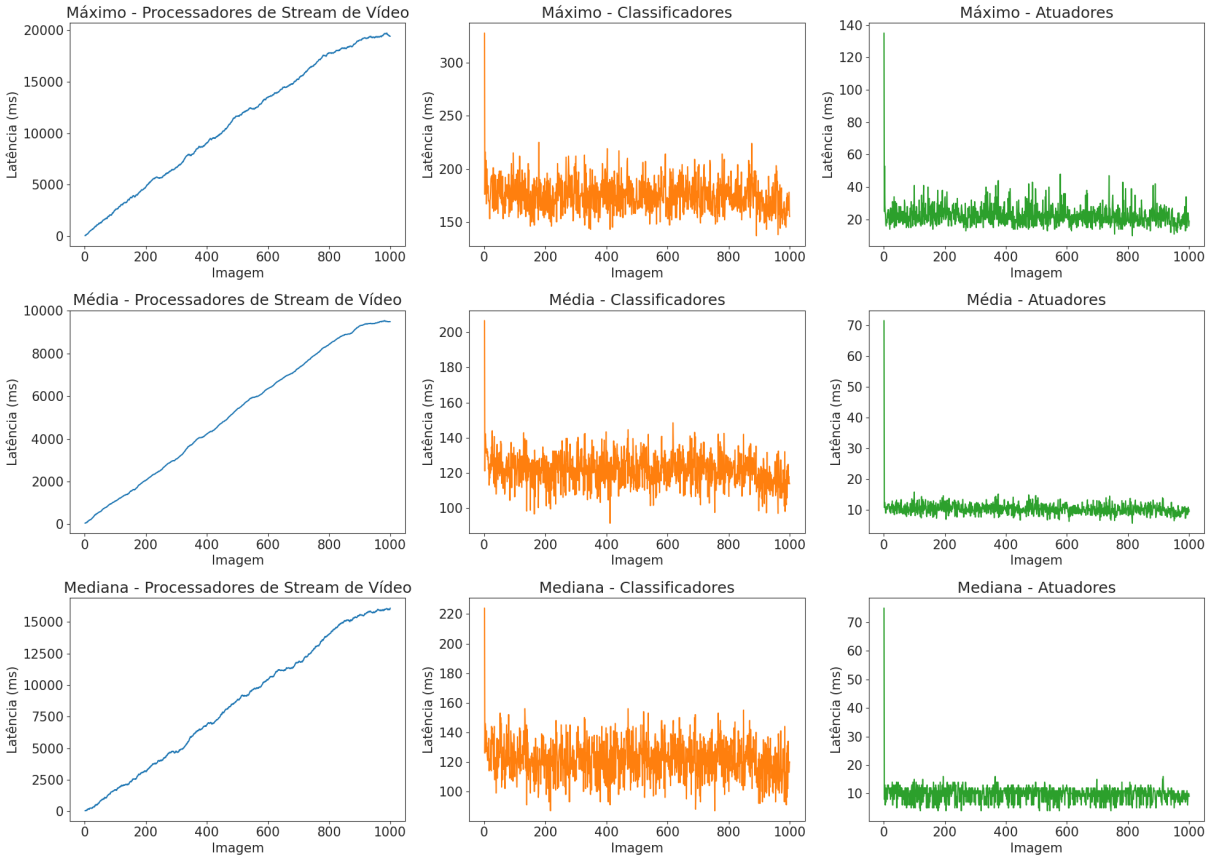

Figura 8 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 5 da execução com 2 máquinas.

dispersões, com destaque para os Cenários 3 e 4, que alcançam médias superiores a 24.000 ms e 38.000 ms, respectivamente, refletindo a sobrecarga em situações de maior complexidade. Já os Classificadores demonstram crescimento progressivo das latências conforme o número de serviços aumenta, chegando a uma média de 175,03 ms no Cenário 5. Por fim, os Atuadores mantêm latências significativamente menores, mesmo sob maior carga, evidenciando sua leveza computacional.

A Tabela 5 complementa a análise do desempenho da arquitetura ao apresentar as estatísticas descritivas das latências agregadas pela média de cada imagem trafegada, nos diferentes tipos de serviço e nos cinco cenários testados com duas máquinas. A análise evidencia um comportamento semelhante ao observado na Tabela 4: os PSVs apresentam os maiores tempos médios de latência, variando de 5.210,65 ms (Cenário 5) a 16.152,66 ms (Cenário 4), com destaque para os altos desvios padrão, indicando variabilidade significativa nas transmissões. Por outro lado, Classificadores e Atuadores mantêm valores mais estáveis, com médias que oscilam em faixas mais estreitas, mesmo com o aumento da complexidade dos cenários. Um ponto de interesse é o comportamento do Cenário 5, que, apesar de possuir mais serviços que os anteriores, apresenta as menores latências médias nos três tipos de serviços, sugerindo que a distribuição automática dos contêineres pelo Docker Swarm pode ter resultado em uma alocação mais equilibrada dos recursos

Tabela 4 – Dados estatísticos descritivos das latências agregadas pelo máximo de cada imagem trafegada pelo tipo de serviço (PSVs, Classificadores e Atuadores) na execução com 2 máquinas, valores em milissegundos.

Processadores de *Stream* de Vídeo										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	28,27	18.051,64	8.903,27	5.074,24	5.168,64	9.156,37	13.300,58	16.144,54	16.972,34	17.705,63
Cenário 2	100,79	50.002,89	25.304,10	14.591,93	12.698,11	25.203,12	38.325,95	45.297,92	47.543,75	49.500,40
Cenário 3	51,66	49.630,98	24.907,86	14.145,22	12.502,80	25.022,00	37.290,77	44.629,47	46.775,80	49.043,65
Cenário 4	139,64	71.117,49	38.391,31	21.155,57	20.428,45	40.075,02	56.979,71	66.278,35	68.920,04	70.819,04
Cenário 5	56,48	19.733,57	11.021,92	5.973,15	5.727,14	11.663,85	16.622,23	19.049,60	19.359,58	19.630,53
Classificadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	63,00	186,00	118,26	29,73	91,75	119,00	144,00	159,00	164,00	169,00
Cenário 2	69,00	193,00	140,30	20,22	127,00	144,00	156,00	163,00	166,00	175,00
Cenário 3	86,00	221,00	144,85	15,35	137,00	148,00	156,00	160,00	164,00	171,00
Cenário 4	111,00	195,00	155,18	11,20	149,00	156,00	162,00	168,00	172,00	182,00
Cenário 5	137,00	328,00	175,03	15,04	164,00	174,00	184,00	194,00	200,00	212,00
Atuadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	12,00	184,00	17,84	6,57	16,00	17,00	19,00	21,00	24,00	32,00
Cenário 2	11,00	76,00	19,43	5,94	16,00	18,00	20,00	25,00	30,00	43,00
Cenário 3	10,00	114,00	19,97	7,23	17,00	19,00	21,00	25,00	30,00	45,00
Cenário 4	10,00	67,00	24,53	7,10	20,00	23,00	28,00	34,00	38,00	46,00
Cenário 5	10,00	135,00	22,43	6,61	18,00	22,00	25,00	29,00	33,00	41,00

computacionais, atenuando os efeitos da sobrecarga.

Tabela 5 – Dados estatísticos descritivos das latências agregadas pela média de cada imagem trafegada pelo tipo de serviço (PSVs, Classificadores e Atuadores) na execução com 2 máquinas, valores em milissegundos.

Processadores de *Stream* de Vídeo										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	28,27	18.051,64	8.903,27	5.074,24	5.168,64	9.156,37	13.300,58	16.144,54	16.972,34	17.705,62
Cenário 2	49,01	22.225,39	11.454,47	6.552,18	5.748,15	11.373,68	17.279,10	20.302,33	21.234,95	22.007,93
Cenário 3	33,38	13.604,18	6.777,13	3.905,34	3.334,54	6.715,27	10.201,35	12.277,96	12.874,23	13.488,87
Cenário 4	57,25	29.348,21	16.152,66	8.867,31	8.326,40	17.082,09	24.027,91	27.720,85	28.557,32	29.196,88
Cenário 5	45,19	9.543,06	5.210,65	2.941,55	2.553,66	5.413,27	7.898,60	9.293,92	9.415,90	9.511,92
Classificadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	63,00	186,00	118,26	29,73	91,75	119,00	144,00	159,00	164,00	169,00
Cenário 2	66,33	160,67	114,46	16,84	102,33	114,33	126,67	137,33	142,02	151,67
Cenário 3	72,80	150,60	110,78	13,30	101,60	111,00	119,60	128,20	133,61	141,60
Cenário 4	85,10	141,40	112,60	9,34	106,20	112,55	119,03	124,91	127,61	133,50
Cenário 5	91,40	206,60	121,69	9,18	115,72	121,73	127,48	132,40	136,20	142,07
Atuadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	12,00	184,00	17,84	6,57	16,00	17,00	19,00	21,00	24,05	32,01
Cenário 2	8,67	44,67	15,41	3,11	13,33	15,00	16,67	19,00	20,67	26,00
Cenário 3	5,80	38,80	13,28	3,05	11,40	13,00	14,65	16,40	17,80	23,20
Cenário 4	6,70	32,10	12,87	2,47	11,10	12,70	14,40	16,00	17,10	19,00
Cenário 5	5,60	71,47	10,27	2,46	9,20	10,13	11,20	12,20	12,93	14,20

A Tabela 6 apresenta os dados estatísticos descritivos das latências agregadas pela mediana de cada imagem trafegada nos Processadores de *Stream* de Vídeo (PSVs), Classificadores e Atuadores durante a execução com duas máquinas. Observa-se que os PSVs apresentam as maiores variações de latência, com destaque para o Cenário 4, que possui a maior média (11.539,66 ms) e também a maior dispersão, evidenciada pelo desvio padrão de 6.208,36 ms. O Cenário 3 contrasta significativamente, apresentando a menor média (1.072,44 ms), o que pode estar relacionado à menor carga de trabalho ou a uma distri-

buição mais eficiente entre os serviços. Para os Classificadores, as latências se mantêm mais estáveis ao longo dos cenários, com médias variando entre 111,09 ms e 121,53 ms, sugerindo um impacto mais controlado das mudanças na topologia da arquitetura. Já os Atuadores apresentam as menores latências entre os três tipos de serviço, com médias entre 9,85 ms e 17,84 ms, reforçando que este estágio final do pipeline é o menos impactado pelo aumento da carga.

Tabela 6 – Dados estatísticos descritivos das latências agregadas pela mediana de cada imagem trafegada pelo tipo de serviço (PSVs, Classificadores e Atuadores) na execução com 2 máquinas, valores em milissegundos.

Processadores de *Stream* de Vídeo										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	28,27	18.051,64	8.903,27	5.074,24	5.168,64	9.156,37	13.300,58	16.144,54	16.972,34	17.705,63
Cenário 2	32,26	16.638,92	8.317,06	4.893,71	4.099,82	7.799,81	12.529,17	15.010,32	15.883,23	16.515,40
Cenário 3	19,33	2.614,40	1.072,44	760,36	445,62	801,47	1.628,40	2.325,74	2.460,07	2.564,63
Cenário 4	37,31	19.865,28	11.539,66	6.208,36	5.707,01	12.727,14	17.193,74	19.471,06	19.630,60	19.802,84
Cenário 5	48,91	16.100,24	8.527,99	4.985,33	4.039,62	8.823,55	12.880,74	15.512,48	15.840,33	16.013,70
Classificadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	63,00	186,00	118,26	29,73	91,75	119,00	144,00	159,00	164,00	169,00
Cenário 2	66,00	163,00	113,66	22,07	96,00	114,00	131,00	143,00	150,00	159,00
Cenário 3	65,00	159,00	111,09	19,33	96,00	110,00	126,00	137,00	142,00	152,00
Cenário 4	73,50	146,00	112,57	13,40	103,00	113,00	122,00	130,00	135,00	142,50
Cenário 5	87,00	224,00	121,53	12,97	113,00	122,00	129,00	138,00	142,00	150,00
Atuadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	12,00	184,00	17,84	6,57	16,00	17,00	19,00	21,00	24,00	32,00
Cenário 2	8,00	42,00	15,07	3,08	13,00	15,00	17,00	19,00	20,00	23,00
Cenário 3	4,00	25,00	13,30	2,98	11,00	13,00	16,00	17,00	18,00	20,00
Cenário 4	4,50	24,00	12,46	2,57	11,00	12,50	14,00	15,50	16,50	19,50
Cenário 5	4,00	75,00	9,85	3,14	8,00	10,00	11,00	12,00	13,00	14,00

A Tabela 7 apresenta a taxa de perda de dados trafegados nos cinco cenários da execução com duas máquinas, comparando a quantidade enviada com a quantidade efetivamente recebida. Em todos os cenários, observa-se uma taxa de perda igual a 0,00%, indicando que, no contexto específico do ambiente controlado de testes, não ocorreram perdas durante a transmissão dos dados entre os serviços da arquitetura. Esse resultado sugere que, sob condições ideais de rede e carga, a arquitetura foi capaz de preservar integralmente o tráfego de dados mesmo em cenários mais exigentes, como o Cenário 5, com 45.000 transmissões.

Tabela 7 – Taxa de perda de dados trafegados em cada cenário na execução com 2 máquinas.

	Qtd. de dados enviados	**Qtd. de dados recebidos**	**Taxa de perda de dados**
Cenário 1	3.000	3.000	0,00%
Cenário 2	9.000	9.000	0,00%
Cenário 3	15.000	15.000	0,00%
Cenário 4	30.000	30.000	0,00%
Cenário 5	45.000	45.000	0,00%

5.4.2 Execução com 3 máquinas

A Figura 9 apresenta os histogramas das latências registradas para os três tipos de serviço da arquitetura ArionStream — Processadores de *Stream* de Vídeo, Classificadores e Atuadores — considerando todos os cenários executados com três máquinas. Observa-se que os PSVs exibem uma distribuição fortemente assimétrica à direita, com grande concentração de latências abaixo de 20.000 ms, mas com presença de valores extremos que se estendem até cerca de 140.000 ms, indicando ocorrência de *outliers* e alta variabilidade em determinadas circunstâncias. Por outro lado, os Classificadores apresentam uma distribuição mais concentrada, com a maioria das latências situando-se entre 100 ms e 300 ms, denotando comportamento mais estável e previsível. Já os Atuadores demonstram latências extremamente baixas e com mínima dispersão, concentradas em valores entre 0 ms e 100 ms, confirmando sua baixa carga computacional e papel terminal no pipeline de processamento. A comparação entre os três histogramas evidencia que a adição de uma terceira máquina contribuiu para a redução da pressão sobre os serviços intermediários, embora o impacto da carga ainda se manifeste de forma mais acentuada nos PSVs, reforçando a necessidade de balanceamento adequado nesse estágio da arquitetura.

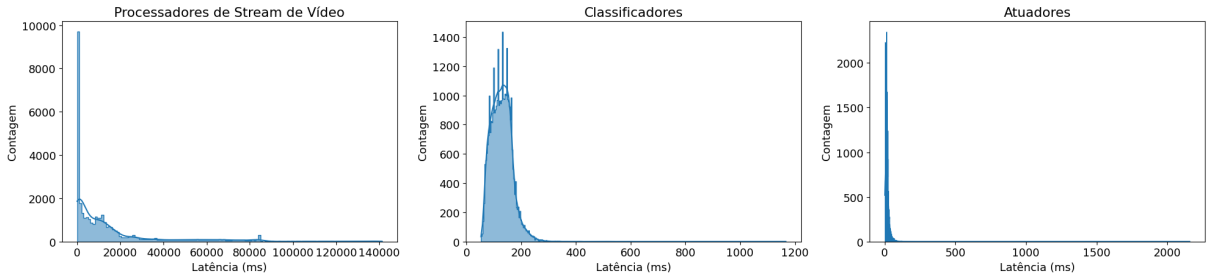

Figura 9 – Histogramas com as latências de todos os serviços por tipo (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) em todos os cenários na execução com 3 máquinas.

A Figura 10 apresenta os histogramas das latências por tipo de serviço — Processadores de *Stream* de Vídeo, Classificadores e Atuadores — desagregados por cenário, durante a execução com três máquinas. Nos PSVs, nota-se que os cenários com maior número de serviços (especialmente o Cenário 5, em roxo) exibem caudas longas e maior frequência de latências elevadas, indicando que o aumento na complexidade da topologia contribui para a ocorrência de picos de latência mais pronunciados. Já nos Classificadores, as distribuições mantêm-se mais concentradas e assimétricas à direita, com maior densidade entre 100 e 200 ms, e com distinções claras entre cenários: cenários mais simples, como o 1 e 2, mantêm latências mais contidas, enquanto cenários mais complexos, como o 4 e 5, deslocam a distribuição para valores levemente superiores. Por fim, os Atuadores apresentam histogramas estreitos e centrados próximos de 0 ms em todos os cenários, demonstrando comportamento altamente consistente e estável. De forma geral, a figura reforça que, mesmo com o acréscimo de uma terceira máquina, a complexidade do cenário ainda impacta de forma relevante os PSVs, enquanto os demais serviços mantêm

estabilidade, evidenciando o bom isolamento de carga entre camadas da arquitetura.

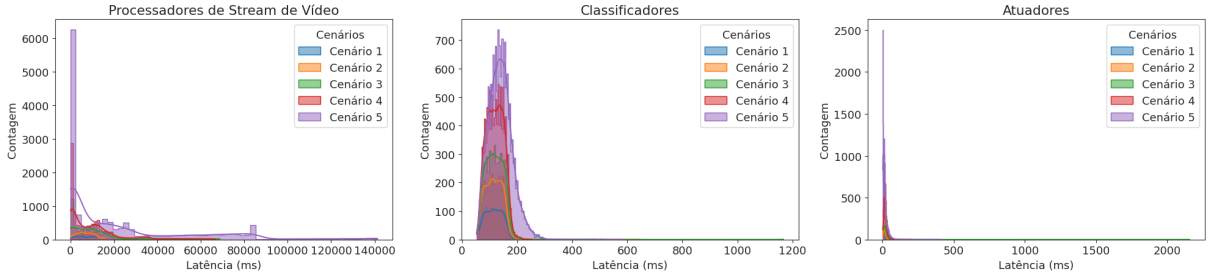

Figura 10 – Histogramas com as latências de todos os serviços por tipo (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) e cenário na execução com 3 máquinas.

A Figura 11 apresenta a evolução das latências máximas, médias e medianas por imagem trafegada para cada tipo de serviço — Processadores de *Stream* de Vídeo, Classificadores e Atuadores — durante a execução do Cenário 1 com três máquinas. Observa-se que os Processadores de *Stream* de Vídeo apresentam um crescimento quase linear e constante em todas as três métricas, refletindo acúmulo de latência ao longo do tempo, mesmo em um cenário com baixa carga. Esse comportamento sugere que há retenção progressiva nos buffers ou filas internas, o que pode ser atribuído à natureza sequencial do processamento e ao volume de dados acumulado. Em contraste, os Classificadores exibem latências significativamente mais estáveis, com variações pontuais e picos esporádicos, o que indica comportamento mais resiliente a flutuações momentâneas. Já os Atuadores mantêm latências baixas e pouco dispersas em todas as métricas, concentrando-se majoritariamente abaixo de 50 ms, com apenas alguns picos isolados. A figura evidencia a eficiência do escalonamento com três máquinas para os serviços intermediários e finais, embora os PSVs continuem sensíveis ao acúmulo de carga, mesmo em um cenário inicial de baixa complexidade.

A Figura 12 mostra a evolução das latências máximas, médias e medianas por imagem para os três tipos de serviço da arquitetura ArionStream no Cenário 2, com execução em três máquinas. Nos Processadores de *Stream* de Vídeo, o padrão de crescimento linear observado nas três métricas se intensifica em comparação ao Cenário 1, alcançando valores superiores a 16.000 ms nas latências máximas. Esse crescimento sugere acúmulo mais pronunciado de carga mesmo com a distribuição em três nós, possivelmente reflexo da elevação do número de PSVs no cenário. Nos Classificadores, há um aumento perceptível na variabilidade, principalmente nas métricas máxima e média, com ocorrência de picos de latência superiores a 600 ms. Ainda assim, a latência mediana se mantém relativamente estável, em torno de 150 ms, indicando que a maioria das amostras permanece dentro de um intervalo aceitável, apesar dos *outliers*. Os Atuadores, por sua vez, seguem demonstrando um comportamento controlado, com latências médias e medianas entre 10 ms e 30 ms, e picos ocasionais que, embora mais frequentes que no Cenário 1, não ultrapassam 200 ms. Em conjunto, os dados revelam que a complexidade adicional do

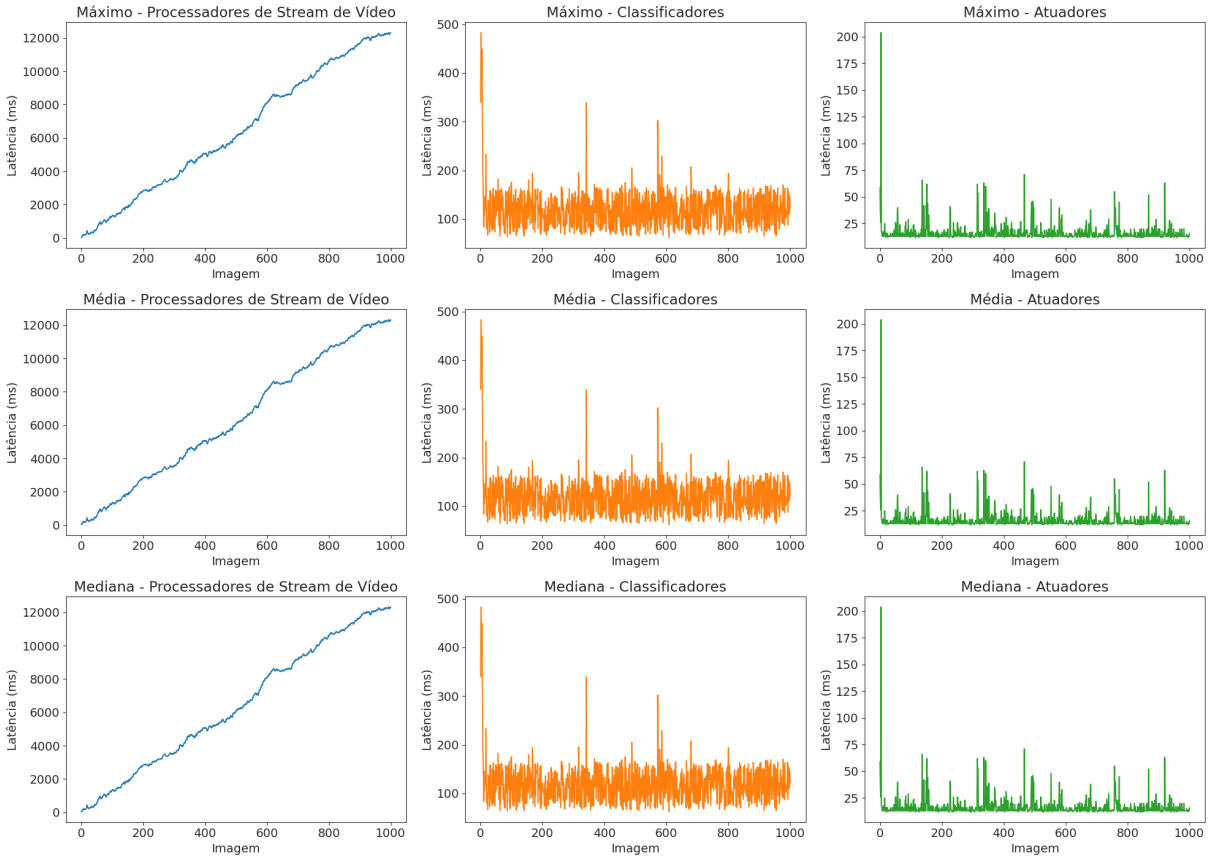

Figura 11 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 1 da execução com 3 máquinas.

Cenário 2 impacta principalmente os PSVs e, em menor grau, os Classificadores, enquanto os Atuadores continuam apresentando estabilidade mesmo sob carga crescente.

A Figura 13 apresenta a evolução das latências máximas, médias e medianas para os Processadores de *Stream* de Vídeo, Classificadores e Atuadores durante o Cenário 3 da execução com três máquinas. Nos PSVs, o padrão crescente das três métricas permanece evidente, mas com a inclusão de degraus abruptos entre as imagens 300 e 400, sugerindo acúmulo repentino de carga ou contenção de recursos em instantes específicos. As latências máximas ultrapassam 70.000 ms, enquanto as medianas e médias mantêm crescimento contínuo, ainda que com menor intensidade. Nos Classificadores, o comportamento é relativamente estável, mas registra-se um pico extremo na latência máxima próximo à imagem 300, que é próximo de 1.200 ms, indicando uma possível sobrecarga momentânea ou atraso de comunicação fora do padrão. Nos Atuadores, embora as latências em geral permaneçam baixas, observa-se um pico anômalo significativo tanto nas latências máximas quanto médias — acima de 2.000 ms e 400 ms, respectivamente — entre as imagens 300 e 350, o que destoa dos demais pontos e indica uma possível propagação transitória de latência vinda dos serviços anteriores. Essa figura evidencia que, no Cenário 3, apesar da distribuição dos serviços entre três máquinas, o sistema ainda está sujeito a flutuações severas em momentos específicos, reforçando a necessidade de monitoramento contínuo e

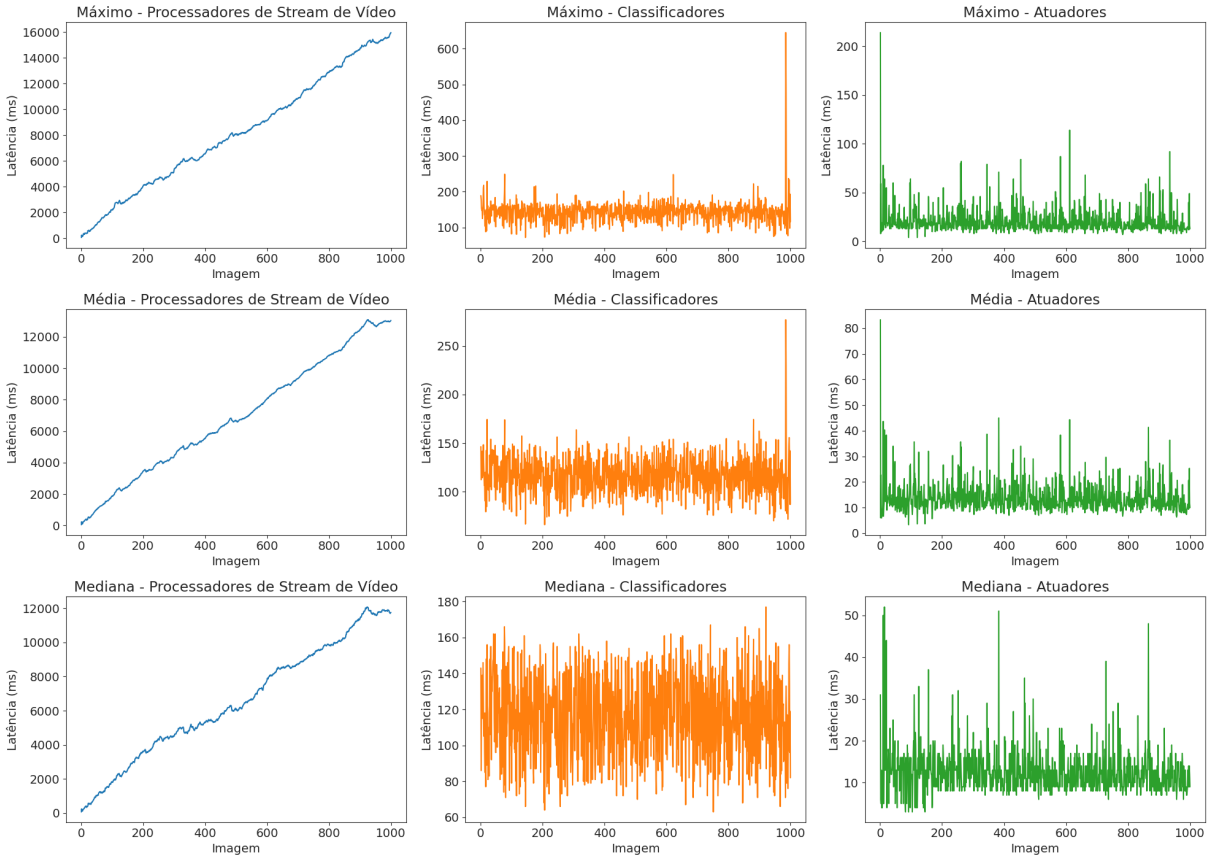

Figura 12 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 2 da execução com 3 máquinas.

estratégias de balanceamento dinâmico.

A Figura 14 exibe a evolução das latências máximas, médias e medianas para os três tipos de serviço — Processadores de *Stream* de Vídeo, Classificadores e Atuadores — durante o Cenário 4 com três máquinas. Nos Processadores de *Stream* de Vídeo, nota-se uma continuidade no padrão de crescimento linear já observado em cenários anteriores, com latências máximas ultrapassando os 70.000 ms e crescimento constante também nas médias e medianas, o que evidencia o impacto do aumento da carga nesse estágio do pipeline, mesmo com a presença de três nós. Nos Classificadores, há maior estabilidade estatística em comparação aos cenários anteriores: apesar da presença de picos, os valores permanecem contidos entre 120 ms e 260 ms, sem grandes desvios ao longo da execução, indicando bom aproveitamento dos recursos alocados e menor sensibilidade à carga adicional. Nos Atuadores, embora se observem oscilações mais visíveis que nos cenários anteriores, as latências médias e medianas permanecem predominantemente abaixo de 30 ms, e os picos máximos não ultrapassam 140 ms, sugerindo que, mesmo com o aumento na complexidade do cenário, o desempenho deste serviço terminal manteve-se dentro de limites operacionais adequados. De forma geral, a figura mostra que, no Cenário 4, a sobrecarga se intensifica principalmente nos PSVs, enquanto os demais serviços apresentam resiliência estável, beneficiados pela distribuição equilibrada promovida pela terceira

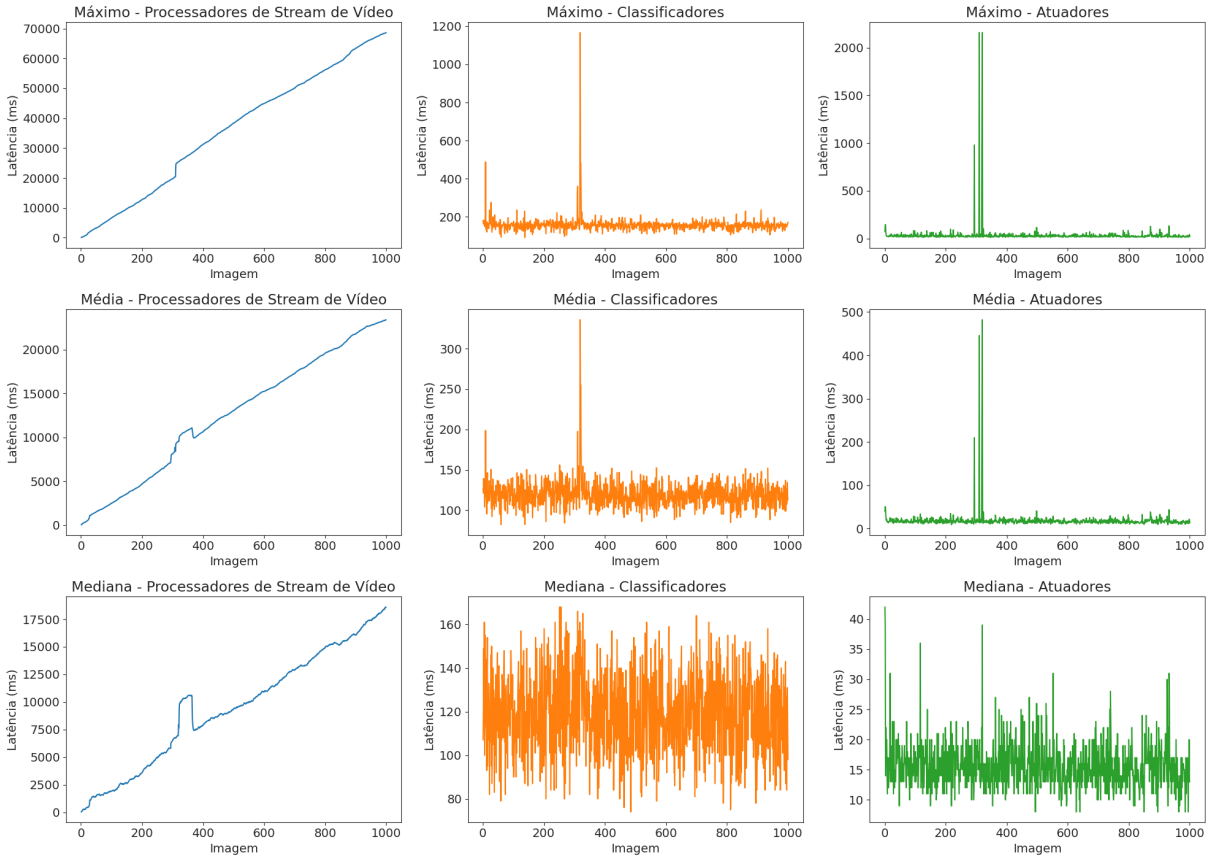

Figura 13 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 3 da execução com 3 máquinas.

máquina.

A Figura 15 apresenta as curvas de latência máxima, média e mediana para os serviços da arquitetura ArionStream no Cenário 5, o mais complexo da execução com três máquinas. Os Processadores de *Stream* de Vídeo (PSVs) atingem as maiores latências de toda a série, com picos máximos superiores a 140.000 ms e médias que ultrapassam 35.000 ms, mesmo com a distribuição entre múltiplos nós. O crescimento exponencial dessas curvas evidencia um acúmulo expressivo de carga, indicando que os PSVs são altamente sensíveis à escalabilidade horizontal em ambientes de alta densidade de serviços. Nos Classificadores, observa-se um comportamento peculiar: as latências apresentam um padrão de arco, com valores mais elevados nas imagens intermediárias (por volta da imagem 400) e redução gradual nas etapas finais. Essa dinâmica pode estar relacionada a uma redistribuição adaptativa dos contêineres pelo Docker Swarm ao longo da execução, resultando em melhora progressiva na alocação de recursos. Por fim, os Atuadores continuam apresentando latências baixas e estáveis, com a maioria das amostras abaixo de 30 ms, embora com picos isolados que ultrapassam 400 ms, sugerindo que, mesmo sob alta carga global, esse serviço terminal mantém desempenho satisfatório. Em conjunto, a figura evidencia que, embora a arquitetura se beneficie da adição de nós, o volume extremo de serviços no Cenário 5 pressiona severamente os PSVs, reforçando a importância de estratégias de

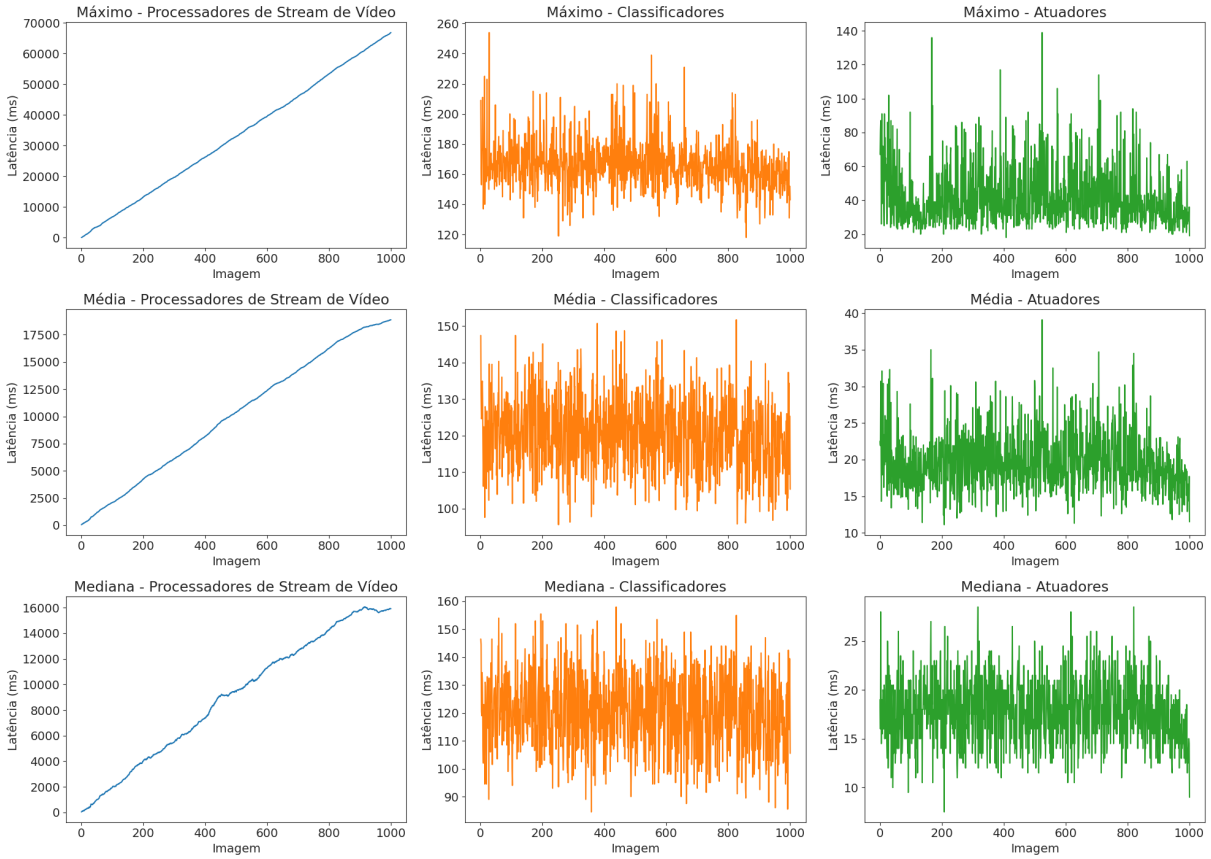

Figura 14 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 4 da execução com 3 máquinas.

balanceamento mais refinadas para esse componente.

A Tabela 8 consolida os dados estatísticos das latências máximas registradas por imagem trafegada, discriminadas por tipo de serviço (PSVs, Classificadores e Atuadores) nos cinco cenários da execução com três máquinas. Os Processadores de Stream de Vídeo (PSVs) continuam a apresentar os maiores valores absolutos e maior dispersão entre os cenários, com destaque para o Cenário 5, que atinge média de 80.947,80 ms e pico de 141.209,19 ms, acompanhado de um desvio padrão elevado (42.058,98 ms), evidenciando forte sensibilidade à sobrecarga. Nos Classificadores, observa-se um aumento progressivo nas latências médias à medida que a complexidade do cenário cresce, alcançando 210,74 ms no Cenário 5, com percentil P99 de 387,00 ms, o que indica que, apesar de sua relativa estabilidade, este serviço também sofre com picos de latência em situações de maior carga. Por sua vez, os Atuadores registram os menores valores em todos os cenários, mas com variação crescente: enquanto no Cenário 1 a média é de 16,37 ms, no Cenário 5 esse valor sobe para 55,74 ms, com um máximo de 411,00 ms.

A Tabela 9 apresenta as estatísticas descritivas das latências agregadas pela média de cada imagem trafegada entre os serviços da arquitetura ArionStream (PSVs, Classificadores e Atuadores) nos cinco cenários da execução com três máquinas. Os Processadores de *Stream* de Vídeo mantêm as maiores latências médias entre os serviços, com destaque para

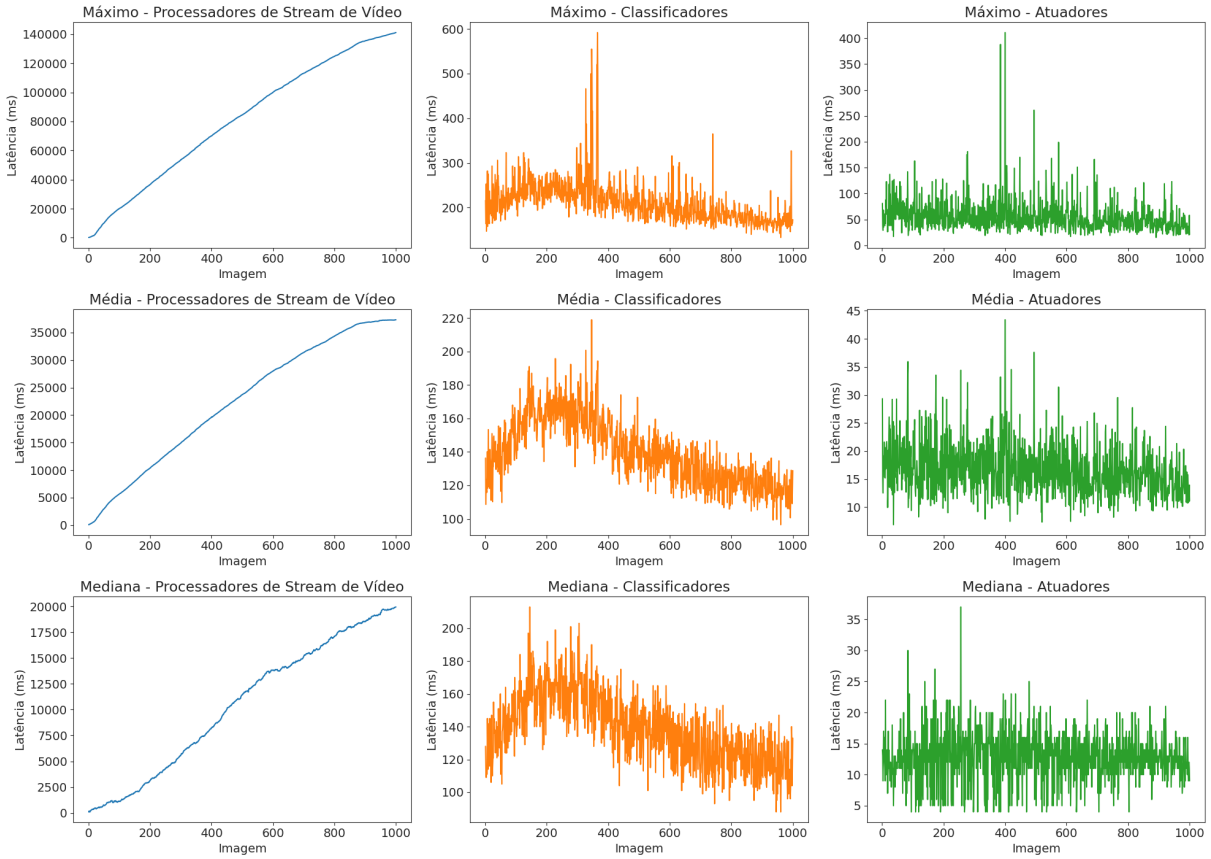

Figura 15 – Latências agrupadas por máximo, média e mediana para cada tipo de serviço (Processadores de *Stream* de Vídeo, Classificadores e Atuadores) no Cenário 5 da execução com 3 máquinas.

Tabela 8 – Dados estatísticos descritivos das latências agregadas pelo máximo de cada imagem trafegada pelo tipo de serviço (PSVs, Classificadores e Atuadores) na execução com 3 máquinas, valores em milissegundos.

Processadores de *Stream* de Vídeo										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	26,21	12.322,75	6.406,94	3.755,30	3.186,56	6.077,71	9.714,58	11.689,15	12.085,74	12.249,09
Cenário 2	67,97	15.956,46	8.180,39	4.418,41	4.596,56	8.087,42	11.826,48	14.695,97	15.263,41	15.567,98
Cenário 3	76,57	68.678,13	35.925,42	20.491,42	16.378,27	38.352,78	53.129,00	63.392,51	66.294,52	68.278,15
Cenário 4	92,21	66.805,70	33.145,71	19.243,15	16.508,79	32.960,72	49.557,85	60.082,79	63.438,15	66.071,57
Cenário 5	171,16	141.209,19	80.947,80	42.058,98	45.158,64	84.552,08	118.823,80	135.454,83	138.230,22	140.525,15
Classificadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	62,00	483,00	119,59	38,99	91,00	117,00	143,00	158,00	163,00	229,00
Cenário 2	72,00	645,00	142,96	29,98	129,00	145,00	156,00	165,00	174,00	215,00
Cenário 3	92,00	1.167,00	158,20	42,17	146,00	157,00	165,00	174,00	187,00	236,00
Cenário 4	118,00	254,00	166,19	15,58	157,00	165,00	173,00	184,00	195,00	214,00
Cenário 5	133,00	592,00	210,74	47,21	178,00	203,00	232,00	260,00	278,00	387,00
Atuadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	12,00	204,00	16,37	9,24	13,00	13,00	16,00	20,00	28,00	53,00
Cenário 2	4,00	214,00	20,46	13,03	13,00	17,00	23,00	34,00	43,00	68,00
Cenário 3	11,00	2.158,00	34,75	108,13	20,00	24,00	32,00	44,00	58,00	102,00
Cenário 4	18,00	139,00	41,68	17,42	29,00	36,00	49,00	66,00	79,00	92,00
Cenário 5	15,00	411,00	55,74	30,96	37,00	49,00	66,00	88,00	112,00	154,00

o Cenário 5, que apresenta uma média de 22.385,96 ms e desvio padrão de 11.382,92 ms, o que evidencia não apenas a carga elevada nesse estágio, mas também a alta variabilidade. A tendência de crescimento contínuo conforme o cenário se torna mais complexo é clara, com o P99 superando os 37.000 ms nesse último cenário. Nos Classificadores, observa-se

Tabela 9 – Dados estatísticos descritivos das latências agregadas pela média de cada imagem trafegada pelo tipo de serviço (PSVs, Classificadores e Atuadores) na execução com 3 máquinas, valores em milissegundos.

Processadores de *Stream* de Vídeo										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	26,21	12.322,75	6.406,94	3.755,30	3.186,56	6.077,70	9.714,58	11.689,15	12.085,74	12.249,09
Cenário 2	56,27	13.089,87	6.925,74	3.729,94	3.982,80	6.695,94	10.088,60	12.451,51	12.895,23	13.012,05
Cenário 3	44,72	23.391,97	12.516,90	6.902,88	5.998,39	13.061,86	18.408,72	21.774,91	22.733,86	23.276,29
Cenário 4	50,58	18.875,59	10.134,26	5.704,83	5.130,23	10.372,10	15.203,68	17.974,96	18.429,17	18.775,81
Cenário 5	84,58	37.335,36	22.385,96	11.382,92	12.739,44	23.722,94	32.780,43	36.823,52	37.200,29	37.288,08
Classificadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	62,00	483,00	119,59	38,99	91,00	117,00	143,00	158,00	163,05	229,04
Cenário 2	66,00	277,00	116,37	18,95	103,33	116,33	128,42	140,03	145,35	155,67
Cenário 3	82,00	335,80	119,54	16,44	109,60	119,20	128,20	136,40	142,61	152,00
Cenário 4	95,60	151,70	120,41	9,74	113,20	120,35	127,00	133,11	136,11	143,71
Cenário 5	96,53	218,93	139,89	19,27	124,85	137,73	154,30	166,5	172,60	186,80
Atuadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	12,00	204,00	16,37	9,24	13,00	13,00	16,00	20,00	28,05	53,02
Cenário 2	3,33	83,33	13,77	5,70	10,33	12,33	15,33	19,70	23,33	35,67
Cenário 3	9,00	482,20	18,43	22,75	14,20	16,20	18,85	22,22	25,40	38,40
Cenário 4	11,1	39,10	19,63	3,95	16,90	19,00	21,80	24,81	27,21	30,80
Cenário 5	6,87	43,40	16,73	4,53	13,47	16,13	19,40	22,53	24,60	29,62

um comportamento mais estável: as latências médias variam de 116,37 ms (Cenário 2) a 139,89 ms (Cenário 5), com crescimento gradual e previsível. Apesar dos aumentos, os desvios padrões se mantêm baixos em comparação aos PSVs, e os percentis superiores indicam que a maior parte das amostras se encontra abaixo de 200 ms, mesmo no cenário mais exigente. Por fim, os Atuadores continuam com as menores latências do sistema, apresentando médias abaixo de 20 ms em todos os cenários, e valores de P99 que não ultrapassam 54 ms. Esses resultados indicam que, embora a complexidade crescente impacte mais severamente os PSVs, a arquitetura mantém bom controle de latência nos serviços intermediários e finais, mesmo sob alta demanda e com maior distribuição entre nós.

A Tabela 10 traz os valores estatísticos das latências agregadas com base na mediana por imagem trafegada, organizados por tipo de serviço e por cenário, durante a execução com três máquinas. Nos Processadores de *Stream* de Vídeo, verifica-se um aumento consistente da mediana conforme a complexidade dos cenários cresce, partindo de 6.077,71 ms no Cenário 1 para 11.444,07 ms no Cenário 5. Embora os valores médios de latência já fossem altos nas tabelas anteriores, a análise da mediana reforça que não apenas picos isolados, mas a maioria das imagens já está sendo processada com alta latência, especialmente nos cenários mais avançados. Os Classificadores, por sua vez, mantêm relativa estabilidade ao longo dos cenários, com medianas variando entre 116 ms e 137,5 ms, e percentis superiores que demonstram controle aceitável de latência mesmo sob maior carga. Já os Atuadores continuam a exibir o melhor desempenho entre os três serviços, com medianas entre 13 ms e 15 ms e baixa dispersão, mesmo no Cenário 5, cujo valor de P99 é de apenas 22 ms. A comparação entre os serviços mostra que os PSVs são o principal gargalo de latência, enquanto Classificadores e Atuadores permanecem eficientes, o que confirma a importância da orquestração e da escalabilidade vertical ou segmentada para

Tabela 10 – Dados estatísticos descritivos das latências agregadas pela mediana de cada imagem trafegada pelo tipo de serviço (PSVs, Classificadores e Atuadores) na execução com 3 máquinas, valores em milissegundos.

Processadores de *Stream* de Vídeo										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	26,21	12.322,75	6.406,94	3.755,30	3.186,56	6.077,71	9.714,58	11.689,15	12.085,74	12.249,09
Cenário 2	67,61	12.081,19	6.524,99	3.334,41	4.257,05	6.145,56	9.318,85	11.397,42	11.798,82	11.921,06
Cenário 3	34,71	18.611,02	9.493,43	5.145,57	4.977,88	9.888,26	13.711,99	16.129,07	17.410,92	18.334,32
Cenário 4	45,69	16.048,69	9.052,66	4.928,81	4.692,55	9.496,53	13.364,68	15.700,39	15.842,34	15.989,60
Cenário 5	95,95	19.952,60	10.423,45	6.380,07	4.304,50	11.444,07	15.887,98	18.559,49	19.383,32	19.813,04
Classificadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	62,00	483,00	119,59	38,99	91,00	117,00	143,00	158,00	163,05	229,04
Cenário 2	63,00	177,00	115,57	22,67	98,00	116,00	133,00	146,00	152,00	161,00
Cenário 3	74,00	168,00	118,78	18,55	105,00	118,50	132,00	143,00	150,00	161,00
Cenário 4	84,50	158,00	120,38	13,34	111,00	120,00	130,50	137,50	141,50	152,00
Cenário 5	88,00	213,00	138,65	20,62	123,75	137,50	153,00	166,00	173,00	186,00
Atuadores										
	Mín.	**Máx.**	**Média**	**Desv. Pad.**	**P25**	**P50**	**P75**	**P90**	**P95**	**P99**
Cenário 1	12,00	204,00	16,37	9,24	13,00	13,00	16,00	20,00	28,00	53,00
Cenário 2	3,00	52,00	12,57	5,09	9,00	12,00	14,00	17,00	20,00	31,00
Cenário 3	8,00	42,00	15,57	3,74	13,00	15,00	17,00	20,00	22,00	27,00
Cenário 4	7,50	28,50	17,80	3,28	15,50	18,00	20,00	22,00	23,00	25,50
Cenário 5	4,00	37,00	12,80	4,00	11,00	13,00	15,00	18,00	19,00	22,00

mitigar os efeitos da carga sobre o primeiro estágio do fluxo.

A Tabela 11 apresenta os dados relativos à quantidade enviada e efetivamente recebida de dados trafegados em cada um dos cinco cenários da execução com três máquinas, bem como a taxa correspondente de perda. Em todos os cenários — incluindo o mais exigente, com 45.000 dados esperados no Cenário 5 — a taxa de perda foi de 0,00%, evidenciando que nenhum dado foi perdido durante a execução sob essas condições experimentais. Esses resultados indicam que, no ambiente controlado de teste e com suporte de orquestração via Docker Swarm, a arquitetura ArionStream conseguiu garantir entregas completas e confiáveis, mesmo com aumento substancial da carga de trabalho.

Tabela 11 – Taxa de perda de dados trafegados em cada cenário na execução com 3 máquinas.

	Qtd. de dados enviados	**Qtd. de dados recebidos**	**Taxa de perda de dados**
Cenário 1	3.000	3.000	0,00%
Cenário 2	9.000	9.000	0,00%
Cenário 3	15.000	15.000	0,00%
Cenário 4	30.000	30.000	0,00%
Cenário 5	45.000	45.000	0,00%

5.5 Análise dos resultados

Os experimentos realizados revelaram comportamentos consistentes e, na maioria, previsíveis ao longo dos diferentes cenários testados, tanto na execução com duas quanto com três máquinas. A análise dos dados demonstrou que, em geral, o aumento no número de máquinas disponíveis contribuiu para a redução da latência, evidenciando o impacto positivo da distribuição dos serviços em um ambiente de execução paralelo. Contudo, foi

possível identificar exceções relevantes: em alguns cenários mais complexos, a execução com três máquinas resultou em latências superiores às observadas com apenas duas. Esse comportamento se explica pela heterogeneidade dos recursos de *hardware* entre os nós do *cluster*, uma vez que uma das máquinas possuía menor quantidade de unidades de processamento (CPUs). Como o Docker Swarm adota por padrão uma estratégia de balanceamento uniforme na alocação de serviços, essa limitação ocasionou uma competição desigual por recursos computacionais, levando à formação de gargalos locais e ao consequente impacto no desempenho global.

Outro aspecto notável refere-se à confiabilidade da comunicação entre os serviços. Em todos os cenários e execuções, não foram observadas perdas de dados durante o tráfego de mensagens entre os módulos, demonstrando robustez na transferência ponto a ponto no ambiente controlado adotado. No entanto, é importante destacar que esse resultado não pode ser generalizado para ambientes reais, mais sujeitos a variações imprevisíveis de rede, interferências externas e restrições de infraestrutura. A ausência de perdas, portanto, indica um comportamento promissor, mas que requer validações adicionais sob condições operacionais adversas para confirmar a resiliência da arquitetura ArionStream.

Durante as execuções, o sistema foi monitorado continuamente por meio de mecanismos de geração de logs e análise de mensagens. Nenhum travamento, falha inesperada ou interrupção crítica foi identificado. Também não foram detectados erros de comunicação entre os serviços, nem mensagens descartadas ou malformadas nos tópicos de mensageria. Esse controle foi fundamental para garantir a consistência dos dados coletados e a validade das métricas apresentadas. Cabe ressaltar, no entanto, que o ambiente de testes foi estável e livre de fatores externos, o que contribuiu para esses bons resultados. Cenários de produção, com maior complexidade e imprevisibilidade, poderão demandar estratégias adicionais de detecção e recuperação de falhas.

Em relação ao desempenho por tipo de serviço, os Processadores de *Stream* de Vídeo foram, de forma consistente, o principal ponto de impacto da carga computacional. Isso ocorre, principalmente, pelo fato de todos os PSVs serem obrigados, nos experimentos, a processar todas as imagens geradas, sem qualquer mecanismo de filtragem ou amostragem. Essa abordagem acarreta acúmulo progressivo de latência nos PSVs, especialmente nos cenários de maior complexidade. Em contraponto, nos cenários reais, é possível que os PSVs operem com lógica de descarte adaptativo, processando apenas os quadros mais recentes ou ajustando sua carga com base em sua capacidade atual, o que tende a reduzir significativamente a pressão sobre esses serviços.

Adicionalmente, observou-se uma relação indireta entre os serviços: quando os PSVs apresentaram maior latência, os Classificadores e Atuadores tenderam a exibir menores tempos de resposta, e vice-versa. Essa dinâmica é coerente com o funcionamento da arquitetura: ao acumular latência nos PSVs, menos dados são entregues aos serviços seguintes, aliviando sua carga. Por outro lado, quando os PSVs operam com menor latência, entre-

gam um volume maior de dados, aumentando a demanda sobre os demais módulos. Esse comportamento confirma que os efeitos da carga se distribuem de forma interdependente entre as camadas do fluxo de execução.

Em termos de latência total de transferência, os resultados indicam uma tendência clara de crescimento conforme os cenários se tornam mais complexos, ainda que esse crescimento não ocorra de forma estritamente linear. Tal comportamento evidencia que a complexidade do cenário — entendida como o número e a combinação de serviços ativos — interage com múltiplos fatores, como paralelismo, orquestração e contenção de recursos, moldando a latência final observada.

Destaca-se, ainda, que algumas variações inesperadas nos resultados — como a melhora no desempenho em determinados cenários mais carregados — apontam para otimizações oportunas realizadas pelo Docker Swarm na alocação dos serviços entre os nós. Essas situações indicam que, mesmo sob cargas elevadas, é possível atingir uma alocação eficiente de recursos quando a distribuição se ajusta bem à topologia e à capacidade dos nós. Assim, os dados sugerem que a latência total de transferência é determinada por uma combinação complexa de fatores, que incluem não apenas o número de serviços, mas também a forma como se relacionam, o padrão de tráfego entre eles e o comportamento do orquestrador.

Por fim, a análise conjunta dos resultados evidencia que, embora os PSVs sejam os mais impactados pela sobrecarga, a arquitetura como um todo consegue conter a propagação da latência para os serviços subsequentes. Classificadores e Atuadores mantêm, na maior parte dos cenários, estabilidade e baixa variabilidade nos tempos de resposta, reforçando a capacidade da arquitetura ArionStream de preservar sua responsividade global, mesmo sob condições crescentes de pressão computacional.

6 Conclusão

Este trabalho teve como objetivo propor, implementar e validar a ArionStream, uma arquitetura distribuída voltada para aplicações de Internet das Coisas Multimídia (IoMT), com execução na borda da rede. A proposta surgiu da necessidade de enfrentar os desafios típicos de cenários onde há conectividade limitada, sensoriamento multimídia intensivo e demandas por processamento contínuo e em tempo real — características recorrentes em ambientes críticos de monitoramento automatizado e resposta rápida.

O percurso metodológico adotado possibilitou o alcance dos objetivos específicos de forma coerente e fundamentada. Inicialmente, foi conduzido um levantamento bibliográfico que permitiu identificar os principais conceitos e tecnologias associados ao desenvolvimento de sistemas distribuídos, com destaque para microsserviços, comunicação assíncrona e computação de borda. Esses conceitos foram sistematizados e integrados no desenho da arquitetura ArionStream, que combina modularidade, escalabilidade e tolerância a falhas por meio de uma abordagem conteinerizada e distribuída.

A validação prática da arquitetura foi conduzida por meio de experimentos controlados, com variação no número de máquinas e na complexidade dos cenários de execução. Os resultados obtidos demonstraram que a ArionStream é capaz de manter a estabilidade da latência, mesmo em cenários com maior carga e distribuição de serviços. Observou-se ainda que a arquitetura se adapta bem a diferentes condições de alocação, evidenciando sua flexibilidade. A avaliação empírica contribuiu para confirmar o potencial da abordagem proposta, bem como para identificar oportunidades de otimização e aprimoramento.

Ainda assim, algumas limitações foram identificadas e merecem atenção. A alocação automática dos serviços, embora funcional, não leva em conta o relacionamento lógico entre eles, o que pode resultar em alocações subótimas e impacto no desempenho. Além disso, a ausência de mecanismos de segurança, como criptografia e autenticação, representa uma lacuna importante para aplicações reais. Os testes foram realizados em um ambiente de rede local estável, o que limita a extrapolação dos resultados para contextos mais sujeitos a falhas e interferências.

A experiência adquirida ao longo do desenvolvimento proporcionou importantes lições técnicas e conceituais. Destaca-se a importância de uma comunicação assíncrona híbrida — combinando mensagens com e sem intermediação por serviços de mensageria — como forma eficaz de equilibrar desacoplamento, flexibilidade e desempenho. Também ficou evidente que a adoção de microsserviços, embora vantajosa, impõe exigências elevadas de orquestração e observabilidade, sendo necessário planejar cuidadosamente o monitoramento e a gestão de dependências. Por fim, a abordagem de processamento na borda mostrou-se estratégica, não apenas como alternativa à nuvem, mas como componente essencial para garantir autonomia e responsividade em sistemas críticos.

As principais contribuições deste trabalho podem ser resumidas como:

- Proposição de uma arquitetura distribuída baseada em microsserviços para análise de vídeo em tempo real em aplicações de IoMT executadas na borda da rede, com comunicação assíncrona híbrida.
- Implementação de uma abordagem modular, flexível e escalável, capaz de operar sob restrições de conectividade, heterogeneidade de dispositivos e cargas variáveis.
- Validação experimental da arquitetura em diferentes configurações de execução, com análise quantitativa dos impactos sobre latência, escalabilidade e robustez.
- Sistematização de um conjunto de boas práticas e diretrizes arquiteturais que podem orientar o desenvolvimento de soluções semelhantes voltadas ao processamento distribuído de fluxos multimídia.

6.1 Trabalhos futuros

Como desdobramento deste trabalho, propõe-se a introdução de critérios explícitos de alocação de serviços durante a orquestração, permitindo que componentes logicamente relacionados sejam implantados em nós mais próximos. Essa otimização pode reduzir a latência e o volume de dados trafegados entre os serviços, além de contribuir para um melhor aproveitamento dos recursos computacionais disponíveis em cada máquina.

Também se recomenda a realização de experimentos em ambientes mais realistas e adversos, com injeção de ruído na rede, simulação de perda de pacotes, variações de latência e limitações de largura de banda. Esses testes podem evidenciar o comportamento da arquitetura sob condições operacionais críticas, mais próximas da realidade de aplicações de campo, além de verificar sua resiliência e estabilidade frente a falhas.

Outra frente importante é o aprimoramento da avaliação de desempenho. Embora este trabalho tenha focado na análise de latência, outras métricas, como vazão (volume de dados processados por unidade de tempo), devem ser consideradas para uma visão mais abrangente do comportamento da arquitetura. Além disso, é necessário coletar e analisar dados de uso de CPU, memória e rede durante os testes, a fim de mensurar com maior precisão o custo computacional da solução proposta e permitir estimativas mais realistas de escalabilidade e viabilidade de implantação.

No tocante à escalabilidade, futuros estudos devem ir além da abordagem descritiva adotada neste trabalho, incorporando análises quantitativas mais robustas, como gráficos de tendência de desempenho em relação ao aumento do número de máquinas ou serviços. Esse tipo de análise pode indicar limites de eficiência e ajudar a prever o comportamento da arquitetura em implantações de maior escala.

Adicionalmente, a proposta aqui desenvolvida poderia ser comparada com arquiteturas alternativas, como modelos monolíticos ou baseados em nuvem, possibilitando uma avaliação mais clara das vantagens e desvantagens de uma abordagem distribuída na borda da rede. Essa comparação experimental traria contribuições para a tomada de decisão de arquitetos de sistemas em cenários de projeto.

Outro aspecto fundamental a ser aprimorado envolve a segurança. A arquitetura atual não implementa mecanismos de criptografia nas comunicações nem autenticação entre serviços, o que pode representar um risco em ambientes que tratam dados sensíveis ou expostos à internet. Trabalhos futuros devem contemplar a integração de camadas de segurança, como criptografia de ponta a ponta e autenticação mútua, mantendo o equilíbrio entre proteção e desempenho.

Por fim, a adaptação da arquitetura para suportar balanceamento dinâmico de carga com base em métricas em tempo real — como utilização de CPU, memória e rede — é uma linha de evolução promissora, que pode ampliar significativamente a eficiência e a capacidade de resposta do sistema em cenários com variação dinâmica de carga.

Referências

ALVI, S. A. et al. Internet of multimedia things: Vision and challenges. *Ad Hoc Networks*, Elsevier, v. 33, p. 87–111, 2015. Citado na página 21.

AMINIYEGANEH, K.; COUTINHO, R. W.; BOUKERCHE, A. Iot video analytics for surveillance-based systems in smart cities. *Computer Communications*, Elsevier, v. 224, p. 95–105, 2024. Citado na página 19.

BATTISTI, A. L. É.; MUCHALUAT-SAADE, D. C.; DELICATO, F. C. Enabling internet of media things with edge-based virtual multimedia sensors. *IEEE Access*, IEEE, v. 9, p. 59255–59269, 2021. Citado na página 21.

BAZGIR, E.; HAQUE, E.; UDDIN, M. S. Analysis of distributed systems. *International Journal of Computer Applications*, v. 975, p. 8887, 2024. Citado na página 16.

BUDAU, V.; BERNARD, G. Synchronous/asynchronous switch for a dynamic choice of communication model in distributed systems. In: IEEE. *Ninth International Conference on Parallel and Distributed Systems, 2002. Proceedings.* [S.l.], 2002. p. 97–102. Citado 2 vezes nas páginas 16 e 18.

BURCKHARDT, S. Consistency in distributed systems. In: *LASER Summer School on Software Engineering.* [S.l.]: Springer, 2013. p. 84–120. Citado na página 17.

CURRY, E. et al. Multimodal event processing: A neural-symbolic paradigm for the internet of multimedia things. *IEEE Internet of Things Journal*, IEEE, v. 9, n. 15, p. 13705–13724, 2022. Citado 2 vezes nas páginas 22 e 24.

DAMYANOV, D.; VARBANOV, Z. Using asynchronous programming in c#-problems, practical tips and scenarios: A short review. In: IEEE. *2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES).* [S.l.], 2024. p. 1–6. Citado 2 vezes nas páginas 18 e 19.

DAVE, S. A. et al. Scalable microservices for cloud based distributed systems. *DIRA*, v. 12, n. 3, p. 776–809, 2024. Citado na página 17.

DEBAUCHE, O.; MAHMOUDI, S.; GUTTADAURIA, A. A new edge computing architecture for iot and multimedia data management. information 2022, 13, 89. *mdpi. com*, 2022. Citado na página 13.

EL-SAYED, H. et al. Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment. *ieee access*, IEEE, v. 6, p. 1706–1717, 2017. Citado 2 vezes nas páginas 20 e 21.

GIAO, J. et al. A framework for service-oriented architecture (soa)-based iot application development. *Processes*, MDPI, v. 10, n. 9, p. 1782, 2022. Citado 2 vezes nas páginas 22 e 24.

GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, Elsevier, v. 29, n. 7, p. 1645–1660, 2013. Citado na página 13.

HASSAN, A. A.; HASSAN, T. M. Real-time big data analytics for data stream challenges: An overview. *European Journal of Information Technologies and Computer Science*, v. 2, n. 4, p. 1–6, 2022. Citado na página 13.

ISHA, V. et al. An approach to clean architecture for microservices using python. In: IEEE. *2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*. [S.l.], 2023. p. 1–5. Citado na página 17.

JAMIL, M. N. et al. Workload orchestration in multi-access edge computing using belief rule-based approach. *IEEE Access*, IEEE, 2023. Citado na página 20.

JIANG, H.; WANG, J. Real-time streaming media processing and optimization technology in intelligent video surveillance. In: IEEE. *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*. [S.l.], 2024. p. 1–5. Citado na página 19.

JOHNSON, O. et al. Building a microservices architecture model for enhanced software delivery, business continuity and operational e< iciency. *International Journal of Frontiers in Engineering and Technology Research*, v. 7, n. 02, p. 070–081, 2024. Citado na página 18.

KHAN, M. Mastering real-time data processing applications : Optimization strategies for peak performance. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, v. 10, p. 895–904, 10 2024. Citado na página 18.

LEE, J.; CAMERON, I.; HASSALL, M. Managing process safety and operational risks with industry 4.0 technologies. In: *Handbook of Smart Materials, Technologies, and Devices: Applications of Industry 4.0*. [S.l.]: Springer, 2022. p. 1501–1527. Citado na página 13.

LU, S. et al. Edge computing on iot for machine signal processing and fault diagnosis: A review. *IEEE Internet of Things Journal*, IEEE, v. 10, n. 13, p. 11093–11116, 2023. Citado na página 13.

MONTAZEROLGHAEM, A. Software-defined internet of multimedia things: Energy-efficient and load-balanced resource management. *IEEE Internet of Things Journal*, IEEE, v. 9, n. 3, p. 2432–2442, 2021. Citado 2 vezes nas páginas 13 e 21.

NETO, A. R. et al. An architecture for distributed video stream processing in iomt systems. *Open Journal of Internet Of Things (OJIOT)*, RonPub, v. 6, n. 1, p. 89–104, 2020. Citado 2 vezes nas páginas 22 e 24.

ORSOLIC, I.; SKORIN-KAPOV, L. A framework for in-network qoe monitoring of encrypted video streaming. *IEEE access*, IEEE, v. 8, p. 74691–74706, 2020. Citado na página 13.

OVEYSIKIAN, A. Data compression algorithms for improving real-time monitoring and automation in iot-enabled smart homes. *International journal of Modern Achievement in Science, Engineering and Technology*, v. 2, n. 1, p. 17–30, 2024. Citado na página 21.

PATEL, S. et al. Practically high performant neural adaptive video streaming. *Proceedings of the ACM on Networking*, ACM New York, NY, USA, v. 2, n. CoNEXT4, p. 1–23, 2024. Citado 2 vezes nas páginas 19 e 21.

RAFFIN, T. et al. A microservice-based architecture for flexible data acquisition at the edge in the context of hairpin stator production. In: IEEE. *2021 11th International Electric Drives Production Conference (EDPC)*. [S.l.], 2021. p. 1–8. Citado na página 17.

RAMADHA, A. A.; YOVITA, L. V.; WIBOWO, T. A. Design and implementation named data networking-based video streaming system. In: *2022 5th International Conference on Information and Communications Technology (ICOIACT)*. [S.l.: s.n.], 2022. p. 66–70. Citado na página 19.

ROHITH, M.; SUNIL, A. et al. Comparative analysis of edge computing and edge devices: key technology in iot and computer vision applications. In: IEEE. *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*. [S.l.], 2021. p. 722–727. Citado 3 vezes nas páginas 20, 22 e 24.

SHI, W. et al. Edge computing: Vision and challenges. *IEEE internet of things journal*, Ieee, v. 3, n. 5, p. 637–646, 2016. Citado na página 13.

SILVA, T. P. da et al. Online machine learning for auto-scaling in the edge computing. *Pervasive and Mobile Computing*, Elsevier, v. 87, p. 101722, 2022. Citado 2 vezes nas páginas 22 e 24.

SUSATYONO, J. D.; SUASANA, I. S.; ROZIKIN, K. Integrating big data and edge computing for enhancing ai efficiency in real-time applications. *Journal of Technology Informatics and Engineering*, v. 3, n. 3, p. 337–349, 2024. Citado na página 20.

TAN, X.; WANG, C. Temporal difference enhancement for high-resolution video frame interpolation. In: *Proceedings of the 2023 15th International Conference on Machine Learning and Computing*. [S.l.: s.n.], 2023. p. 433–437. Citado na página 19.

TU, J.; XU, Q.; CHEN, C. Cans: Communication limited camera network self-configuration for intelligent industrial surveillance. In: IEEE. *IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society*. [S.l.], 2021. p. 1–6. Citado na página 13.

TU, J.; YANG, L.; CAO, J. Distributed machine learning in edge computing: Challenges, solutions and future directions. *ACM Computing Surveys*, ACM New York, NY, v. 57, n. 5, p. 1–37, 2025. Citado na página 20.

TYTARCHUK, Y. et al. The impact of distributed systems on the architecture and design of computer systems: advantages and challenges. *Data and Metadata. 2024. Vol. 3,№ 225. DOI: 10.56294/dm2024. 225 URL: https://dm. ageditor. ar/index. php/dm/article/view/225/916*, 2024. Citado 2 vezes nas páginas 16 e 17.

VOLODYMYR, G.; DENYS, D. Asynchronous communication of microservices. *System technologies*, v. 1, p. 108–118, 05 2023. Citado na página 18.

WANG, Y. et al. A design of edge distributed video analysis system based on serverless computing service. In: IEEE. *2023 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. [S.l.], 2023. p. 1–6. Citado 2 vezes nas páginas 23 e 24.

ZEN, K. et al. Latency analysis of cloud infrastructure for time-critical iot use cases. In: IEEE. *2022 Applied Informatics International Conference (AiIC)*. [S.l.], 2022. p. 111–116. Citado na página 13.

ZHANG, W.; CHEN, L. Design and optimization of a high availability, low latency messaging broker using zookeeper and kafka for asynchronous processing. *Asian American Research Letters Journal*, v. 1, n. 3, 2024. Citado na página 18.

ZHANG, X. et al. A data trading scheme with efficient data usage control for industrial iot. *IEEE Transactions on Industrial Informatics*, IEEE, v. 18, n. 7, p. 4456–4465, 2021. Citado na página 13.

ZIELINSKI, E. et al. Secure real-time communication and computing infrastructure for industry 4.0—challenges and opportunities. In: IEEE. *2019 International Conference on Networked Systems (NetSys)*. [S.l.], 2019. p. 1–6. Citado na página 13.

ZIKRIA, Y. B.; AFZAL, M. K.; KIM, S. W. Internet of multimedia things (iomt): Opportunities, challenges and solutions. *Sensors*, MDPI, v. 20, n. 8, p. 2334, 2020. Citado na página 21.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, **Luís Clício Carvalho Sá**, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação "ArionStream: uma arquitetura distribuída para análise de vídeos em tempo real em aplicações de IoMT na borda" de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI, 4 de julho de 2025.

gov.br Documento assinado digitalmente
LUIS CLICIO CARVALHO SA
Data: 04/07/2025 07:45:32-0300
Verifique em https://validar.iti.gov.br

_____

Assinatura