Adaptação Automática de Pipelines de Video Analytics com Suporte de Observabilidade

Luan I. F. Santos[3], Francisco A. de A. Gomes[1], Michel S. Bonfim[2], José G. R. Maia[4], Fernando A. M. Trinta[3] e Paulo A. L. Rego[3]

[1] Campus Crateús
[2] Campus Quixadá
[3] Departamento de Computação
[4] Instituto Universidade Virtual
Universidade Federal do Ceará – Brasil

luanicaro10@alu.ufc.br, almada@crateus.ufc.br, michelsb@ufc.br

gilvanmaia@virtual.ufc.br, {fernando.trinta, paulo}@dc.ufc.br

Abstract. *The expansion of urban cameras has boosted traffic monitoring, enabling real-time event detection. However, the dynamic nature of cities requires Video Analytics (VA) systems to be adaptable to unexpected changes. In Edge Computing, where resources are limited, processing efficiency is crucial. This work proposes the automatic adaptation of VA pipelines with observability support. The approach adjusts resolution, FPS, and inference models to balance accuracy and performance. A MAPE-K cycle analyzes metrics and applies real-time adjustments, optimizing CPU and GPU usage and reducing bottlenecks. Experiments show that adaptation improves system efficiency and stability.*

Resumo. *A expansão das câmeras urbanas impulsionou o monitoramento de tráfego, permitindo a detecção de eventos em tempo real. No entanto, a dinâmica das cidades exige que os sistemas de Video Analytics (VA) sejam adaptáveis a mudanças inesperadas. Em Edge Computing, onde os recursos são limitados, a eficiência no processamento é crucial. Este trabalho propõe a adaptação automática de pipelines de VA com suporte de observabilidade. A abordagem ajusta resolução, FPS e modelo de inferência para equilibrar precisão e desempenho. Um ciclo MAPE-K analisa métricas e aplica ajustes em tempo real, otimizando CPU e GPU e reduzindo gargalos. Experimentos mostram que a adaptação melhora a eficiência e estabilidade do sistema.*

1. Introdução

A instalação de câmeras em ambientes urbanos tem crescido exponencialmente nos últimos anos, refletindo uma crescente necessidade de garantir a segurança pública, prevenir crimes e monitorar atividades em tempo real [Zhang et al. 2019]. Cidades ao redor do mundo estão adotando tecnologias de vigilância para criar ambientes mais seguros, onde a presença de câmeras se tornou uma norma, permitindo que as autoridades respondam rapidamente a incidentes e melhorem a qualidade de vida dos cidadãos [Xu et al. 2023].

Cenários importantes em cidades inteligentes, como o monitoramento de tráfego e a segurança pública, exemplificam a relevância das câmeras de vídeo [Thomé et al. 2020,

Oliveira et al. 2024]. Por exemplo, no monitoramento de tráfego, sistemas inteligentes podem analisar imagens em tempo real para identificar congestionamentos, detectar acidentes, monitorar o cumprimento de regras de trânsito e prever picos de movimento. Isso permite que as autoridades tomem ações imediatas, como reprogramar semáforos ou redirecionar o tráfego para rotas alternativas [Ravindran 2023]. Além disso, a vigilância em áreas públicas, como praças e estações de transporte, permite a detecção de comportamentos suspeitos e a resposta imediata a emergências, contribuindo para a segurança coletiva e a prevenção de crimes. Esses cenários ilustram a aplicação do Video Analytics (VA), uma abordagem que permite a análise de dados de vídeo em tempo real à medida que são capturados. Esse processo de análise instantânea possibilita a tomada de decisões rápidas e a execução de ações imediatas com base nas informações extraídas do fluxo de vídeo, sendo especialmente útil em situações onde a rapidez da resposta é crucial. A Figura 1 ilustra uma representação visual de um *pipeline* típico de VA.

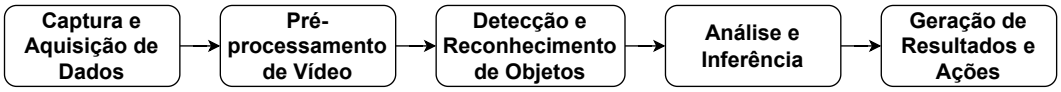

Figura 1. *Pipeline* de *Video Analytics*.

De acordo com a Figura 1, o *pipeline* de VA pode ser descrito em cinco etapas principais, cada uma com um papel importante no processamento e análise de vídeos. A primeira etapa é a **Captura e Aquisição de Dados (*Input*)**, em que o objetivo é coletar o vídeo a ser analisado. Isso pode ser feito por câmeras em tempo real, arquivos de vídeo ou fluxos ao vivo. A segunda etapa é o **Pré-processamento de Vídeo**, que tem o objetivo de melhorar a qualidade do vídeo para facilitar a análise (e.g., redução de ruído, ajustes de brilho, contraste e saturação). A terceira etapa é a **Detecção e Reconhecimento de Objetos**, em que o objetivo é identificar e localizar objetos ou pessoas no vídeo. A quarta etapa é **Análise e Inferência**, que tem o objetivo de processar e interpretar as informações extraídas do vídeo. Por fim, a **Geração de Resultados e Ações**, que tem por objetivo fornecer os resultados da análise ou acionar alertas e sistemas de controle.

Entretanto, a dinâmica dos ambientes urbanos pode mudar rapidamente com o surgimento de novos objetos e de eventos inesperados. Essa variabilidade exige que os sistemas de VA sejam capazes de processar informações de forma eficiente e adaptativa. Em ambientes de *Edge Computing* (EC), onde os recursos computacionais são limitados, a necessidade de um processamento ágil e eficaz se torna ainda mais crítica, pois a capacidade de resposta a essas mudanças pode impactar diretamente a eficácia das operações de monitoramento [Wong et al. 2024].

Diante desse cenário, a adaptação em tempo real se torna uma necessidade premente em ambientes de EC. A carga de trabalho pode variar significativamente, dependendo do número de câmeras ativas, da complexidade das cenas capturadas e da quantidade de eventos a serem analisados. Portanto, é fundamental que os sistemas de VA sejam projetados para se ajustar dinamicamente a essas variações, garantindo que os recursos disponíveis sejam utilizados de maneira otimizada [Wang et al. 2023].

Neste contexto, o conceito de observabilidade emerge como uma ferramenta poderosa para monitorar métricas e ajustar o *pipeline* de VA de forma proativa [Padmanabhan et al. 2023]. A observabilidade permite que os sistemas coletem dados

em tempo real sobre seu desempenho e comportamento, possibilitando a identificação de gargalos e a implementação de ajustes necessários para manter a eficiência do processamento, mesmo em face de mudanças rápidas no ambiente [Usman et al. 2022, Kosińska et al. 2023]. Por exemplo, é possível monitorar a taxa de quadros processados por segundo (FPS) ou a utilização da CPU enquanto o *pipeline* de vídeo está em execução, e se o FPS cair abaixo de um limiar ou a utilização da CPU estiver muito alta, ajustes automáticos podem ser feitos, como a distribuição de carga entre servidores ou a redução da qualidade do vídeo processado para manter o desempenho.

Este trabalho objetiva: (i) avaliar o uso de ferramentas de observabilidade em *pipelines* de *video analytics* e (ii) propor um esquema de adaptação para o *pipeline*, visando aprimorar a eficiência do processamento. Experimentos foram conduzidos para demonstrar a capacidade do sistema em responder a mudanças dinâmicas, e os resultados confirmam a eficácia do esquema de adaptação proposto, evidenciando seu potencial para transformar a operação de sistemas de *Video Analytics* (VA) em ambientes urbanos inteligentes.

O restante do artigo está organizado como segue. A Seção 2 apresenta a fundamentação teórica. A Seção 3 discute os trabalhos relacionados. A metodologia, algoritmos e mecanismos da adaptação automática de *pipelines* de VA com suporte de observabilidade são explicados na Seção 4. A Seção 5 detalha os experimentos e os resultados. Finalmente, a Seção 6 conclui o estudo e sugere temas para pesquisas futuras.

2. Fundamentação Teórica

VA em tempo real refere-se ao processo de analisar dados de vídeo à medida que são capturados, permitindo a tomada de decisões e ações imediatas com base nas informações derivadas do fluxo de vídeo dentro de um limite de tempo. As próximas subseções apresentam respectivamente as principais tecnologias e métodos para suporte à VA, bem como aspectos importantes do emprego de Observabilidade para suporte à VA.

2.1. Tecnologias e Métodos Utilizados em VA

A Inteligência Artificial (IA) e o *Machine Learning* (ML) são tecnologias fundamentais no contexto de VA. A IA desempenha um papel central ao permitir que os sistemas de VA reconheçam padrões e tomem decisões com base em dados visuais. Algumas técnicas relevantes incluem *Redes Neurais Convolucionais (CNNs)*, essenciais para detecção de objetos, reconhecimento facial e classificação de imagens; *Modelos de Deep Learning*, como YOLO (*You Only Look Once*) [Redmon 2016], SSD (*Single Shot MultiBox Detector*)e Faster R-CNN, que oferecem alta precisão na detecção e rastreamento em tempo real; e *Aprendizado por Reforço*, utilizado para adaptação contínua a ambientes dinâmicos, como o monitoramento de tráfego urbano.

Além da IA e ML, o Processamento Digital de Imagem (PDI) tem um papel importante em VA, sendo aplicado na preparação dos *frames* do vídeo antes da execução dos modelos de IA. Entre os mecanismos de PDI mais utilizados estão as técnicas de filtragem, responsáveis pela redução de ruídos para melhorar a qualidade; a segmentação de imagens, que identifica regiões de interesse; e a detecção de bordas, que destaca objetos ou contornos.

Com relação aos ambientes, VA ocorre tanto na *Edge*, em que ocorre próxima à fonte de captura, como em câmeras inteligentes ou dispositivos IoT, com benefícios de redução da latência, permitindo respostas mais rápidas, menor consumo de banda, pois apenas dados processados mais qualificados precisam ser enviados, como também na *Cloud*, que é usada para armazenar e processar grandes volumes de dados, especialmente em aplicações escaláveis. Ela também permite a implantação centralizada de modelos de IA e a integração com sistemas de *Business Intelligence*. Com relação aos Protocolos de Transmissão de Vídeo, os mais utilizados são RTSP (*Real-Time Streaming Protocol*), que é amplamente usado em sistemas de vigilância, e o WebRTC, que é ideal para aplicações de comunicação e *streaming* de baixa latência. Por fim, os componentes de software mais utilizados são: OpenCV, que é uma biblioteca *open-source* para tarefas de visão computacional; TensorFlow e PyTorch, que são *frameworks* populares para treinamento e implementação de modelos de *deep learning*; por fim, Kafka e RabbitMQ, que são ferramentas para processamento e transmissão de dados em tempo real.

2.2. Observabilidade em *Video Analytics*

A observabilidade é um elemento crucial para a eficácia dos sistemas de VA, especialmente devido à complexidade e ao volume de dados envolvidos nesses sistemas. A observabilidade refere-se à capacidade de monitorar e obter informações detalhadas sobre o funcionamento interno de um sistema, permitindo uma compreensão clara de seu desempenho e comportamento. Essa característica é indispensável para garantir que o processamento e a análise de vídeos ocorram de forma eficiente, precisa e confiável.

Um dos aspectos mais importantes da observabilidade é a identificação de gargalos no *pipeline*. Sistemas de VA são compostos por diversas etapas, como captura de vídeo, pré-processamento, inferência de modelos de ML e visualização de resultados. A observabilidade permite monitorar cada uma dessas fases, identificando onde ocorrem atrasos ou uso excessivo de recursos (e.g., CPU, GPU ou memória). Por exemplo, métricas de tempo de execução podem indicar que a etapa de inferência de modelos está demorando mais do que o esperado, possibilitando ajustes em tempo real para otimizar o desempenho. Além disso, a observabilidade é essencial para o diagnóstico e a solução de problemas. Falhas podem surgir devido a diversas causas em sistemas de VA, como perda de quadros, erros de modelos de detecção ou restrições de largura de banda. A capacidade de capturar logs detalhados, métricas e rastreamentos permite que os desenvolvedores identifiquem rapidamente a raiz do problema e implementem correções eficazes. Isso é particularmente importante em aplicações críticas, como segurança e vigilância, onde atrasos ou falhas podem ter consequências significativas.

Outro ponto relevante é a adaptabilidade do sistema. A observabilidade possibilita o ajuste dinâmico de parâmetros do *pipeline* com base nas condições do ambiente ou na carga de trabalho. Por exemplo, se o sistema detectar um aumento na densidade de objetos em uma cena, ele pode ajustar automaticamente a resolução do vídeo ou os parâmetros do modelo de inferência para manter o equilíbrio entre desempenho e precisão. Isso garante que o sistema permaneça eficiente mesmo sob condições variáveis. Por fim, a observabilidade é fundamental para garantir a qualidade dos resultados.

É necessário avaliar continuamente a precisão dos modelos de ML e a consistência dos dados processados em sistemas de VA, eventualmente recorrendo à aprendizagem

federada. Métricas como taxa de falsos positivos, acurácia e tempo médio de resposta podem ser monitoradas em tempo real para assegurar que o sistema atenda aos requisitos de desempenho e confiabilidade.

3. Trabalhos Relacionados

A adaptação dinâmica de *pipelines* de *video analytics* tem sido um tema de crescente interesse na literatura, com abordagens variando desde estratégias de ajuste em cenários de borda até mecanismos para gerenciar recursos em fluxos múltiplos. Esta seção discute os principais trabalhos relacionados, comparando-os com a solução proposta neste artigo.

[Zhang et al. 2022] propõe um sistema de transmissão adaptativa por lotes para equilibrar a largura de banda, a latência e a precisão da análise de vídeo. A abordagem utiliza aprendizado por reforço para ajustar dinamicamente o tamanho dos lotes, garantindo um desempenho otimizado em cenários de recursos limitados. Os autores de [Wang et al. 2024], por sua vez, introduzem um *framework* baseado em amostragem adaptativa e rastreamento orientado por detecção, empregando aprendizado por reforço para adaptar configurações em tempo real e balancear precisão e tempo de processamento. Ambos os trabalhos contribuem significativamente para o processamento eficiente em borda, mas não integram ferramentas de observabilidade como parte de sua abordagem.

[Sun et al. 2024] apresenta o BiSwift, um orquestrador de largura de banda para análise de vídeo em tempo real em cenários com fluxos múltiplos. O sistema utiliza um codec híbrido e um controlador de largura de banda para alocar recursos de forma justa, maximizando a precisão e a utilização da rede. De maneira semelhante, [Mi et al. 2024] introduz o AccDecoder, um sistema que combina super-resolução e aprendizado por reforço para otimizar a latência e a precisão em *pipelines* de vídeo. Embora ambos os trabalhos se concentrem em cenários distribuídos e enfrentem limitações de largura de banda, eles não consideram a possibilidade de adaptações mais amplas, como ajustes de FPS ou troca de modelos de inferência, exploradas neste estudo.

[Faye et al. 2024] propõe o VideoJam, uma arquitetura descentralizada para balanceamento de carga em sistemas de *video analytics* ao vivo, que prevê a carga futura com modelos leves de aprendizado de máquina. Os autores de [Zhang et al. 2023] apresentam um sistema que adapta os *pipelines* utilizando compressão híbrida e particionamento de modelos através de *offloading* adaptativo de tarefas, permitindo a execução colaborativa entre dispositivos e servidores. Esses trabalhos compartilham o objetivo de otimizar o desempenho sob cargas variáveis, mas sua aplicação é restrita a estratégias específicas de balanceamento de carga ou compressão. Em contraste, este trabalho adota uma abordagem mais ampla, integrando ferramentas de observabilidade e um algoritmo que pode ajustar dinamicamente diferentes parâmetros do *pipeline*. A Tabela 1 ilustra as semelhanças e diferenças entre os trabalhos relacionados e a nossa solução.

Enquanto os trabalhos existentes exploram abordagens como colaboração em ambientes de borda, compressão híbrida, particionamento de modelos, gerenciamento de largura de banda e balanceamento de carga, o nosso trabalho inova ao integrar a observabilidade como um componente central por meio de um *loop MAPE-K*. Além disso, a nossa solução propõe um ajuste adaptativo em múltiplas dimensões, incluindo resolução, tamanho do *batch*, taxa de quadros (FPS) e modelos utilizados, o que não é abordado de forma abrangente nos trabalhos analisados. Essa abordagem holística não apenas melhora

Tabela 1. Comparação entre os trabalhos relacionados e este trabalho.

Critério	Contexto da aplicação de video analytics	Métricas utilizadas para adaptação	Adaptações realizadas no *pipeline*
[Zhang et al. 2022]	Análise geral de vídeo, incluindo vigilância e aplicações de aprendizado profundo em servidores de borda ou na nuvem.	Tamanho do lote, latência de transmissão, largura de banda consumida, precisão da inferência.	Ajuste dinâmico do tamanho dos lotes na transmissão e no processamento.
[Zhang et al. 2023]	Análise geral de vídeo, incluindo vigilância e drones, com foco em transmissão adaptativa entre dispositivos locais e servidores.	Largura de banda disponível, latência total, utilização de recursos no dispositivo local e no servidor, e precisão de inferência.	Particionamento do modelo, compressão de dados, razão de compressão e balanceamento entre execução local e no servidor.
[Wang et al. 2024]	Análise de vídeo colaborativa com foco em cenários de múltiplos dispositivos móveis enviando requisições simultâneas.	Taxa de transferência, latência de ponta a ponta, largura de banda consumida, prioridade das requisições.	Corte de regiões-chave e algoritmos de priorização.
[Sun et al. 2024]	Análise de vídeo distribuída com múltiplos fluxos em servidores de borda, incluindo aplicações de vigilância e monitoramento de tráfego.	Largura de banda disponível, densidade e tamanho dos objetos, latência de processamento e precisão da inferência.	Alocação de largura de banda, seleção de quadros-chave e ajustes de parâmetros de codificação (resolução e bitrate).
[Mi et al. 2024]	*Pipelines* distribuídos com foco em vídeos de baixa resolução e limitações de recursos em câmeras e redes.	Correlação temporal entre quadros, latência de processamento, ganhos de super-resolução e precisão da inferência.	Seleção de quadros, reutilização de inferências anteriores e balanceamento entre precisão e latência.
[Faye et al. 2024]	Câmeras fixas e móveis em ambientes urbanos, com foco no balanceamento de carga para sistemas ao vivo.	Taxa de chegada de carga, tamanho das filas, taxa de processamento e previsão de carga futura.	Redistribuição de carga entre vizinhos; suporte a mudanças dinâmicas em configurações de câmeras.
Este Trabalho	Monitoramento de tráfego em ambientes urbanos inteligentes.	Uso de CPU e GPU, tamanho da fila, tempo de inferência, taxa de entrada e taxa de saída.	Ajustes no FPS e resolução do vídeo, tamanho do batch e modelo de inferência.

a eficiência do processamento, mas também garante que *video analytics* sejam executados com a máxima precisão e mínima latência, atendendo assim às exigências de aplicações em tempo real. Assim, apesar dos trabalhos relacionados oferecerem contribuições valiosas no contexto de *video analytics* em tempo real, a solução proposta se destaca ao integrar a observabilidade e a adaptação dinâmica, proporcionando uma solução mais robusta e eficiente para os desafios enfrentados na *Edge Computing*.

4. *Pipeline* de *Video Analytics*

Esta seção apresenta a modelagem do sistema de *pipeline* de VA e seu algoritmo de adaptação que incorpora suporte de observabilidade em cenários de *Edge Computing* para responder a mudanças dinâmicas e otimizar o uso dos recursos disponíveis.

4.1. Funcionamento do Sistema

O sistema proposto realiza o *video analytics* em tempo real com adaptações dinâmicas baseadas na carga de trabalho. Ele é composto por um *pipeline* de microsserviços especializados, como ilustrado na Figura 2, onde cada microsserviço desempenha uma tarefa específica, como detecção de objetos, rastreamento ou classificação. O fluxo de entrada é gerado por C câmeras, onde cada câmera c (1) transmite imagens a uma taxa de FPS_c quadros por segundo e (2) possui resolução ajustável $R_c \in \{480p, 720p, 1080p\}$.

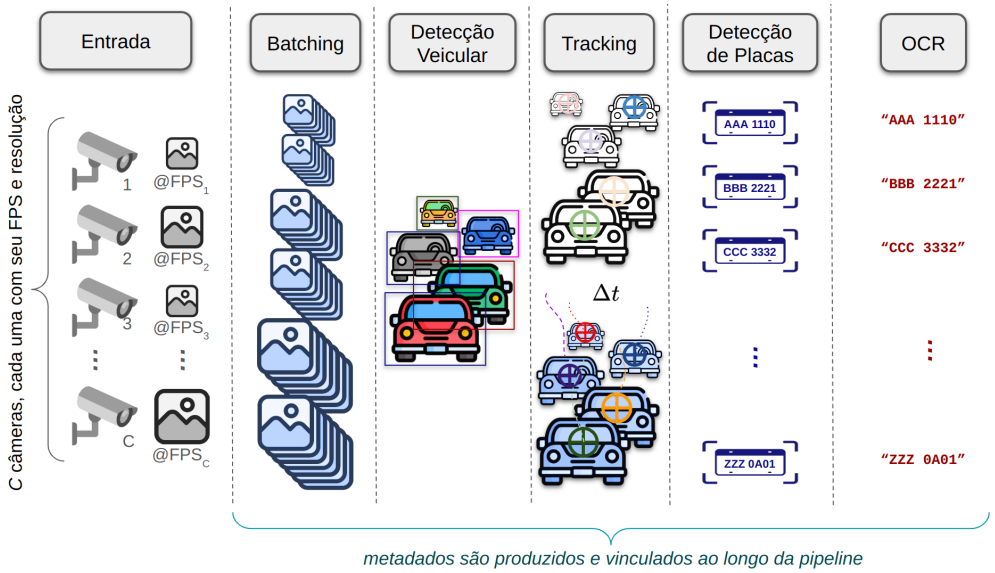

Figura 2. Visão geral do *Pipeline* de Processamento de Veículos.

Cada microsserviço m opera em uma arquitetura baseada em filas (Q_m), garantindo ordenamento e continuidade do fluxo de dados. O primeiro microsserviço (**Batching**) organiza os *frames* em lotes de tamanho B e os encaminha para processamento subsequente. O segundo microsserviço (**Detecção Veicular**) processa lotes para identificar a presença (ou ausência) de veículos em cada *frame*. O terceiro microsserviço (**Tracking**) tem como objetivo acompanhar o movimento dos veículos ao longo do tempo, evitando que as detecções de um mesmo veículo sejam tratadas como duplicatas ao atribuir um ID único a cada veículo. O quarto microsserviço (**Detecção de Placas**) realiza a detecção de placa sobre os veículos rastreados. Por fim, o último microsserviço (**OCR**) usa técnicas de *Optical Character Recognition* para extrair e converter o texto visualmente presente em imagens (ou vídeos) das placas em texto codificado digitalmente (*strings*).

O processamento do *pipeline* ocorre em GPUs ou CPUs, com a possibilidade de alternância dinâmica entre dispositivos para otimizar o desempenho. Parâmetros como resolução (R), tamanho do lote (B) e modelo de inferência (DL, variando entre modelos DL_L e DL_H, menos e mais pesados e precisos, respectivamente) são ajustados em tempo real para evitar sobrecargas e manter a eficiência.

Ferramentas de observabilidade monitoram continuamente métricas como tamanho das filas, taxas de entrada e saída, uso de CPU/GPU e tempo de inferência. Essa abordagem assegura que os *frames* sejam processados sem atrasos, mesmo em cenários de alta variabilidade.

4.2. Modelagem do Sistema

A modelagem do *pipeline* quantifica elementos-chave, como taxas de entrada e saída, estabilidade dos microsserviços e crescimento de filas. A carga de trabalho total gerada pelas câmeras é definida como $WL = \sum_{c=1}^{C} FPS_c$. Cada microsserviço m possui uma taxa de entrada λ_m e uma taxa de saída μ_m que dependem de variáveis como tamanho do lote (B), resolução (R), modelo de inferência (DL) e hardware ($Exec$).

O crescimento da fila $Q_m(t)$ no microsserviço m em um intervalo Δt é modelado

como:

$$Q_m(t + \Delta t) = Q_m(t) + (\lambda_m - \mu_m) \cdot \Delta t.$$

Note-se que μ_m é inversamente proporcional ao tempo de inferência $T_{infer}(DL, R, B, Exec)$, o qual depende de DL, R, B e $Exec$.

Para integrar esses elementos em uma tomada de decisão mais robusta, propõe-se um Índice de Estabilidade com base na taxa de entrada $\lambda_m(t)$, na taxa de saída $\mu_m(t)$ e em um valor crítico para o tamanho da fila, Qc_m. Nesse sistema, deseja-se manter a fila estável abaixo de Qc_m. A dinâmica é descrita pela equação:

$$IE_m(t) = \alpha_m \cdot \sigma\Big(\frac{\lambda_m(t)}{\mu_m(t)} - 1\Big) \; + \; (1 - \alpha_m) \cdot \sigma\Big(Q_m(t) - Qc_m\Big),$$

onde

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

O parâmetro α_m determina a importância relativa de cada termo na composição do IE. Assim, se α_m for alto, a relação $\frac{\lambda_m(t)}{\mu_m(t)} - 1$ terá maior peso; se α_m for baixo, a diferença $Q_m(t) - Qc_m$ prevalecerá. Ajustar α_m permite equilibrar a ênfase entre o uso de recursos de processamento e o controle do tamanho da fila, contribuindo para a escolha de estratégias de escalabilidade e alocação de recursos.

Em síntese, esse modelo possibilita mensurar e controlar de forma integrada a capacidade de processamento, a taxa de chegada de quadros e o tamanho das filas, constituindo uma base sólida para decisões sobre configuração e dimensionamento de recursos em sistemas de videomonitoramento em tempo real.

4.3. Algoritmo de Adaptação

Para garantir a operação em tempo real, foi implementado um ciclo MAPE-K. O sistema coleta métricas operacionais, como tamanho das filas (Q_m), taxas de entrada e saída (λ_m, μ_m), utilização de recursos, como CPU (U_{cpu}) e GPU (U_{gpu}), e tempo de inferência (T_{infer}). A análise dessas métricas identifica gargalos e oportunidades de otimização. Com base na análise, o sistema planeja ajustes como: reduzir resolução (R); alterar a quantidade de FPS (FPS); trocar para modelos mais leves (DL_L); e migrar o processamento entre GPU e CPU.

Essas ações são aplicadas de forma incremental, enquanto o componente de conhecimento armazena histórico de métricas e ajustes. Com o ciclo iterativo, o sistema busca manter a estabilidade em cenários com variabilidade. O Pseudocódigo 1 descreve o algoritmo de adaptação baseado no Índice de Estabilidade do Microsserviço (IE_m):

4.4. Ciclo MAPE-K e Integração com Observabilidade

O MAPE-K é o método central para gerenciar as adaptações dinâmicas no *pipeline*. A Figura 3 ilustra o ciclo utilizado neste trabalho, que se baseia em quatro etapas principais, complementadas pelo armazenamento e utilização do conhecimento acumulado. Mais detalhes sobre estas etapas são apresentadas a seguir.

Monitorar: Durante esta etapa, realiza-se a observação das métricas utilizadas para verificar a necessidade de adaptação nos microsserviços de detecção de veículos. As

Pseudocódigo 1: Algoritmo de Adaptação Baseado em IE_m

Inicializar parâmetros do *pipeline* (R, B, α_m, DL, $Exec$);
while *True* **do**
    Monitorar métricas: λ_m, Q_m, μ_m U_{cpu}, U_{gpu} e T_{infer};
    Calcular $IE_m[N]$;
    for *cada microsserviço* m **do**
        if $IE_m[N] >= 0.5$ *and* $IE_m[N] < 0.6$ **then**
            Diminuir R;
        end
        else if $IE_m[N] >= 0.6$ *and* $IE_m[N] < 0.7$ **then**
            Trocar para DL_L;
        end
        else if $IE_m[N] >= 0.7$ *and* $IE_m[N] < 0.8$ **then**
            Diminuir FPS;
        end
        else if $IE_m[N] < 0,3$ **then**
            Aumentar R, FPS ou trocar para DL_H;
        end
    end
    Atualizar configurações do *pipeline*;
    Armazenar métricas e ajustes no componente de conhecimento;
end

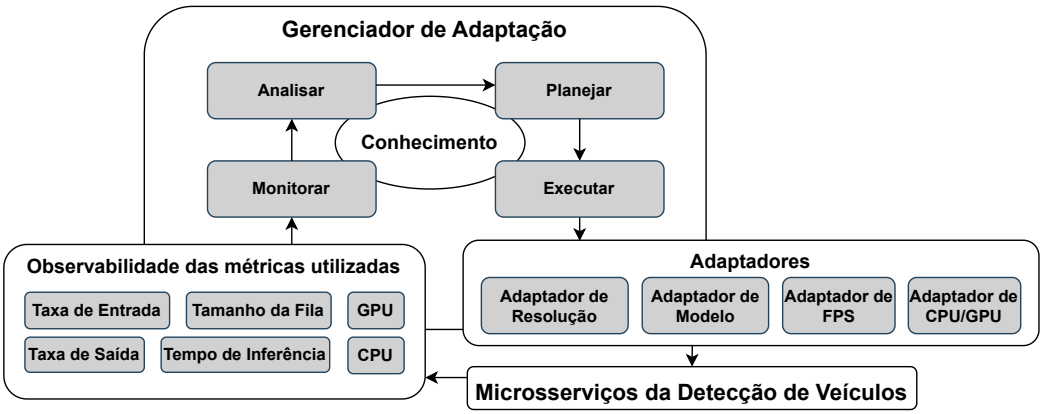

Figura 3. MAPE-K para detecção de veículos.

seguintes métricas são coletadas: tamanho das filas (Q_m), taxas de entrada (λ_m) e saída (μ_m), uso de CPU (U_{cpu}) e GPU (U_{gpu}), e tempo de inferência (T_{infer}).

As ferramentas de observabilidade como Prometheus[1] e Grafana[2] são utilizadas para agregar e visualizar essas métricas, permitindo uma compreensão detalhada do estado atual do sistema. O Prometheus é um projeto da *Cloud Native Computing Founda-*

[1] `https://prometheus.io/`
[2] `https://grafana.com/`

tion, que funciona como um sistema de monitoramento de serviços e sistemas com coleta métricas de alvos configurados em intervalos definidos e pode disparar alertas quando condições específicas são observadas. O Grafana é uma plataforma para visualizar e analisar métricas por meio de gráficos em tempo real.

Analisar: Nesta etapa, os dados coletados na etapa anterior são processados para identificar gargalos e instabilidades, como: (i) crescimento excessivo de filas, (ii) subutilização da capacidade dos microsserviços, (iii) sobrecarga de recursos computacionais (CPU e GPU); e aumento da latência no *pipeline*. A análise foca em determinar quais partes do *pipeline* necessitam de ajustes para estabilizar o sistema ou melhorar sua eficiência. Os dados armazenados no Prometheus são recuperados para realizar essa análise e entender o comportamento do *pipeline* para uma possível adaptação.

Planejar: Com base na análise, são definidos ajustes nos parâmetros do *pipeline*, tais como: (i) reduzir a resolução (R) dos *frames* da câmera para aliviar a carga, diminuir a taxa de *frames* por segundo (FPS_{eff}), trocar temporariamente para modelos de inferência mais leves (DL_L); e redistribuir a carga de processamento entre GPU e CPU. Esses ajustes são repassados para a próxima etapa, onde são executadas as modificações a fim de mitigar os problemas identificados.

Executar: As mudanças planejadas são implementadas automaticamente utilizando ferramentas de orquestração, como, por exemplo, o Kubernetes[3]. A aplicação incremental das ações garante que as adaptações não causem impactos negativos no processamento em tempo real.

Conhecimento: Todas as métricas e ajustes realizados são armazenados em um repositório de conhecimento. Isso permitirá ao sistema aprender com experiências passadas, refinando suas estratégias de adaptação para cenários futuros. A versão atual da solução ainda não usa algoritmos inteligentes para se beneficiar completamente da base de conhecimento.

5. Avaliação Experimental

Foram realizados experimentos com 2, 4, 8 e 16 câmeras/vídeos, considerando quatro cenários distintos de adaptação em VA: (*i*) Adaptação de modelo; (*ii*) Adaptação de resolução; (*iii*) Adaptação de resolução e modelo; e (*iv*) Adaptação de resolução, modelo e FPS. O objetivo foi verificar se as adaptações recomendadas pelo suporte de observabilidade resultaram em melhorias no processamento das requisições, capazes de reduzir os gargalos na detecção de veículos.

5.1. Configuração Experimental

A máquina utilizada nos experimentos possui a seguinte configuração: CPU Intel Core i9-12900F de 12ª geração e 24 núcleos, 128 GB de RAM e sistema operacional Ubuntu; e GPU NVIDIA GeForce RTX 3080 Ti, com chip Ampere GA102, 10.240 CUDA cores, 1365 MHz, GDDR6X, 12 GB de VRAM e potência de 350 W. O vídeo utilizado para simular o tráfego de veículos cruzando um semáforo tem duração de 260 s e foi replicado até 16 vezes a fim de simular o processamento de até 16 câmeras, elevando o estresse nos modelos de detecção devido ao aumento no número de veículos em cena.

[3] `https://kubernetes.io/`

Apesar do *pipeline* de VA ser composto por 5 microsserviços, apenas o microsserviço de detecção de veículos precisou de adaptação para que o sistema mantivesse o processamento em tempo real. As adaptações realizadas foram as seguintes:

Adapt$_M$: Alteração na entrada do modelo Yolov9m de 640p para 480p;
Adapt$_R$: Alteração da resolução da câmera de 1080p para 720p;
Adapt$_{RM}$: Alteração da resolução e entrada do modelo (Adapt$_M$ + Adapt$_R$); e
Adapt$_{RMF}$: Alteração da resolução, modelo e FPS, com as mesmas mudanças do Adapt$_{RM}$ mais a alteração do FPS de 30 para 20.

5.2. Resultados

O *pipeline* apresentou funcionamento estável para o processamento de até 8 câmeras. Entretanto, ao utilizar 16 câmeras, foi necessário aplicar adaptações para reduzir o processamento das imagens e, assim, possibilitar a detecção de veículos na solução.

Os resultados relativos à comparação da adaptação da solução nos quatro cenários mencionados anteriormente com a abordagem sem adaptação são apresentados na Figura 4. A Figura 4(a) apresenta a adaptação Adapt$_R$. Observa-se que, utilizando apenas o modelo, o índice IE_m teve um comportamento semelhante ao da abordagem sem adaptação até os 40 s, mas logo conseguiu se estabilizar. O IE_m permanece acima de 0,6 por 130 s e, em seguida, apresenta uma diminuição, evidenciando que a adaptação se efetivou. Por sua vez, a Figura 4(d) apresenta a adaptação Adapt$_{RMF}$. Nesse caso, em que todas as adaptações são realizadas, praticamente em um período de 40 s, o IE_m permanece acima de 0,6. Esses resultados mostram que as adaptações conseguem melhorar o desempenho do sistema e que, quanto mais adaptações forem realizadas, maior será a estabilidade do sistema.

Além da IE_m, optamos por apresentar a utilização de recursos tanto da CPU quanto da GPU. A Figura 5 (a) mostra que a utilização da CPU sem adaptação chegou até 70%, sendo que em todos os casos com adaptação a CPU teve uma menor utilização, o que mostra que a adaptação contribuiu para diminuir o uso desse recurso da máquina, diminuindo a degradação do sistema. A Figura 5 (b) mostra que a utilização da GPU sem adaptação chegou até 33%, sendo que em todos os casos com adaptação a GPU teve uma maior utilização, o que mostra que a adaptação contribuiu para aumentar uso desse recursos da máquina, mostrando que o sistema teve uma melhoria da carga de trabalho oriundas das adaptações realizadas. Optamos por não apresentar os gráficos referentes ao tamanho da fila e ao tempo de inferência nas adaptações, devido à limitação de espaço no texto. No entanto, as métricas coletadas e os dados dos resultados estão disponíveis na seção Disponibilidade de Artefatos.

Com relação à precisão dos quatro cenários, foi calculada a média dos veículos detectados em um cenário sem gargalo, com 8 câmeras, e esse valor foi multiplicado por dois para simular o uso de 16 câmeras. A média de veículos detectados sem gargalo foi de 47,10, a mediana 48,0 e o Intervalo de Confiança (IC) 95% inferior de 42,9 e superior de 51,3. Em seguida, foi feita uma comparação com a média de veículos detectados em cada adaptação realizada. A Tabela 2 apresenta a precisão dos quatro cenários. Como era esperado, as adaptações economizam recursos, mas reduzem a quantidade de informações disponíveis, o que, consequentemente, diminui a quantidade de veículos detectados. O melhor caso é o que ocorre em Adapt$_R$, contendo apenas 1,2% de

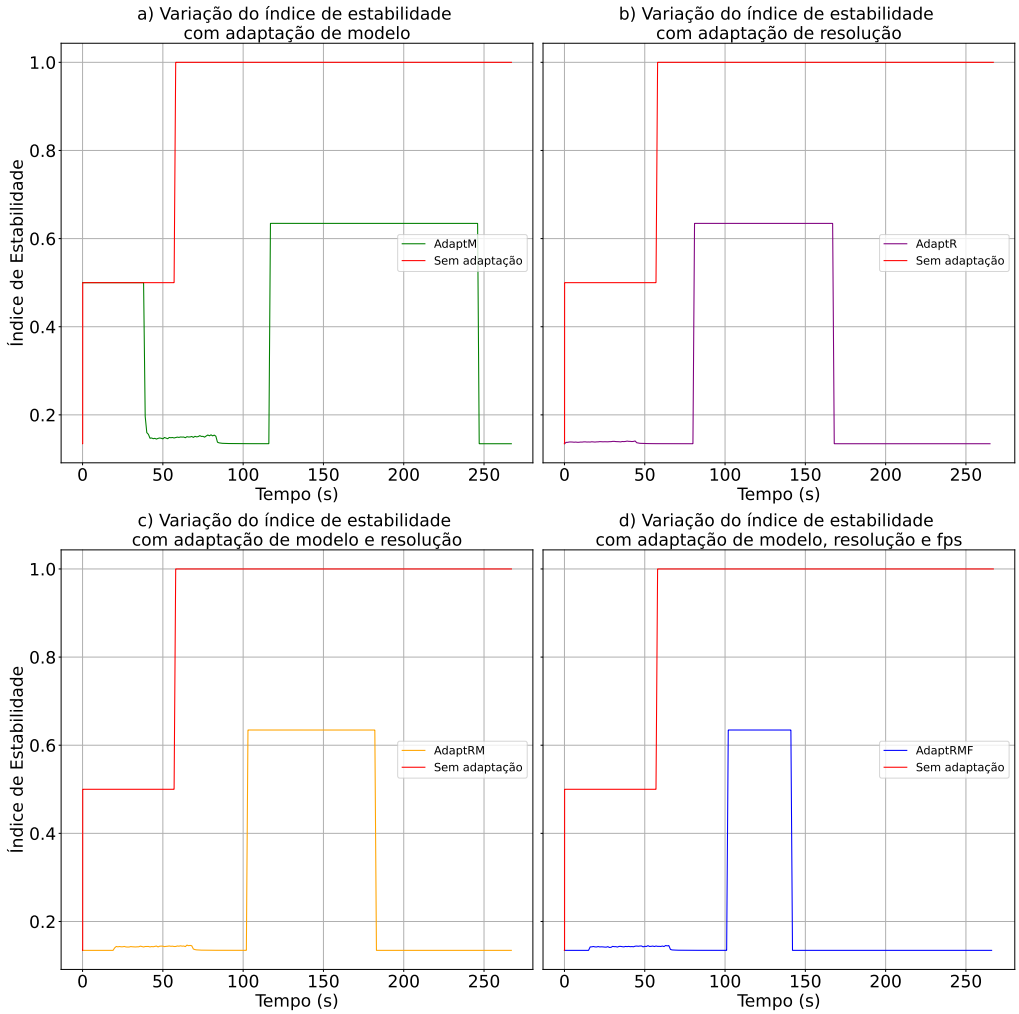

Figura 4. Variação do índice de estabilidade.

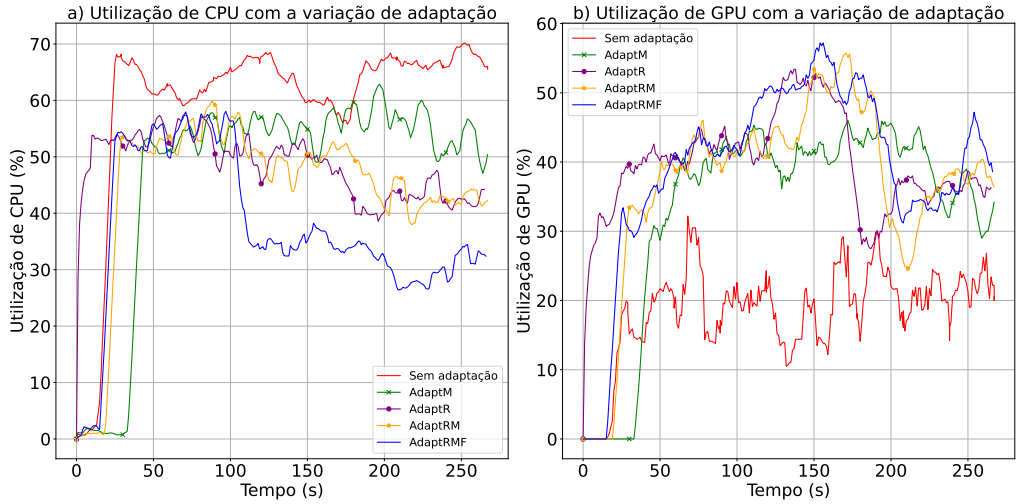

Figura 5. Utilização de CPU e GPU.

diferença em relação à quantidade de veículos detectados sem gargalo. Assim, apesar dos

benefícios das adaptações recomendadas pelo suporte de observabilidade, que melhoram o processamento das requisições e mitigam o gargalo da solução de detecção de veículos, há desvantagens em relação à precisão.

Tabela 2. Precisão da Detecções de Veículos

Adaptação	Média	Erro(%)	Mediana	IC 95% Inferior	IC 95% Superior
Adapt$_M$	43,8	- 7,6	32,0	39,3	48,2
Adapt$_R$	47,7	+ 1,2	40,0	43,5	51,9
Adapt$_{RM}$	44,6	- 5,5	32,0	40,2	49,0
Adapt$_{RMF}$	49,7	+ 5,2	48,0	45,3	54,1

6. Conclusão e Trabalhos Futuros

O mecanismo de adaptação automática para *pipelines* de VA proposto utiliza suporte de observabilidade para otimizar o processamento de vídeos em ambientes urbanos inteligentes e integra um ciclo MAPE-K para monitorar métricas e aplicar ajustes dinâmicos em tempo real, garantindo melhor balanceamento entre precisão e eficiência computacional. Os resultados experimentais demonstraram que a adaptação dinâmica melhora a estabilidade do sistema, reduzindo gargalos e otimizando a utilização da CPU e GPU. Além disso, verificou-se que, embora as adaptações reduzam a carga computacional, há uma pequena perda de acurácia, evidenciando um *trade-off* entre desempenho e precisão. Como trabalhos futuros, pretende-se: (i) incorporar técnicas de ML para prever variações de carga e otimizar ajustes de forma automática e proativa; (ii) explorar novas estratégias de balanceamento entre precisão e desempenho, considerando métricas mais avançadas de qualidade de inferência; e (iii) validar a robustez da solução em cenários reais com diferentes tipos de câmeras e condições ambientais.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Disponibilidade de Artefatos

As métricas utilizadas e os resultados encontrados estão disponíveis neste link[4].

Referências

Faye, Y., Faticanti, F., Jain, S., and Bronzino, F. (2024). Videojam: Self-balancing architecture for live video analytics. In *2024 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 149–163. IEEE.

Kosińska, J., Baliś, B., Konieczny, M., Malawski, M., and Zieliński, S. (2023). Towards the observability of cloud-native applications: The overview of the state-of-the-art. *IEEE Access*.

Mi, L., Yuan, T., Wang, W., Dai, H., Sun, L., Zheng, J., Chen, G., and Fu, X. (2024). Accelerated neural enhancement for video analytics with video quality adaptation. *IEEE/ACM Transactions on Networking*.

[4]https://drive.google.com/file/d/121oaOoC_AJCTvFWwI3dp1vQAlppRTMK8/view?usp=sharing

Oliveira, D., Bhering, F., Obraczka, K., Passos, D., and Albuquerque, C. (2024). Uma arquitetura para roteamento dinâmico de vídeos por multicaminhos em iot. In *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 545–558, Porto Alegre, RS, Brasil. SBC.

Padmanabhan, A., Agarwal, N., Iyer, A., Ananthanarayanan, G., Shu, Y., Karianakis, N., Xu, G. H., and Netravali, R. (2023). Gemel: Model merging for {Memory-Efficient},{Real-Time} video analytics at the edge. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 973–994.

Ravindran, A. A. (2023). Internet-of-things edge computing systems for streaming video analytics: Trails behind and the paths ahead. *IoT*, 4(4):486–513.

Redmon, J. (2016). You only look once: Unified, real-time object detection. In *CVPR Proceedings*. IEEE.

Sun, L., Wang, W., Yuan, T., Mi, L., Dai, H., Liu, Y., and Fu, X. (2024). Biswift: Bandwidth orchestrator for multi-stream video analytics on edge. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 1181–1190. IEEE.

Thomé, M., Prestes, A., Gomes, R., and Mota, V. (2020). Um arcabouço para detecção e alerta de anomalias de mobilidade urbana em tempo real. In *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 784–797, Porto Alegre, RS, Brasil. SBC.

Usman, M., Ferlin, S., Brunstrom, A., and Taheri, J. (2022). A survey on observability of distributed edge & container-based microservices. *IEEE Access*, 10:86904–86919.

Wang, X., Shen, M., and Yang, K. (2024). On-edge high-throughput collaborative inference for real-time video analytics. *IEEE Internet of Things Journal*.

Wang, Y., Liu, Z., Zhao, Y., Wang, X., and Qiu, C. (2023). Enabling real-time video analytics with adaptive sampling and detection-based tracking in edge computing. In *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, pages 3554–3559.

Wong, M., Ramanujam, M., Balakrishnan, G., and Netravali, R. (2024). {MadEye}: Boosting live video analytics accuracy with adaptive camera configurations. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 549–568.

Xu, R., Razavi, S., and Zheng, R. (2023). Edge video analytics: A survey on applications, systems and enabling techniques. *IEEE Communications Surveys & Tutorials*.

Zhang, L., Zhang, Y., Wu, X., Wang, F., Cui, L., Wang, Z., and Liu, J. (2022). Batch adaptative streaming for video analytics. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 2158–2167. IEEE.

Zhang, L., Zhong, Y., Liu, J., and Cui, L. (2023). Resource and bandwidth-aware video analytics with adaptive offloading. In *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 107–115. IEEE.

Zhang, Q., Sun, H., Wu, X., and Zhong, H. (2019). Edge video analytics for public safety: A review. *Proceedings of the IEEE*, 107(8):1675–1696.