Uma Arquitetura de Microsserviços para Detecção de Anomalias em Dados de Mobilidade Urbana Heterogêneos

André N. Prestes, Marco A. B. Thomé, Roberta L. Gomes, Vinícius F. S. Mota

[1]Departamento de Informática – Universidade Federal do Espírito Santo
Vitoria – ES – Brasil

andre.prestes@edu.ufes.br, {mabthome,rgomes,vinicius.mota}@inf.ufes.br

Abstract. *The adoption of collaborative platforms has grown in a way that public agents are increasingly seeking partnerships with these information providers. Data from cameras, social networks, and applications can contribute to the management of smart cities, such as detecting unusual traffic events, for example. This work presents a framework that uses heterogeneous sources to detect anomalous traffic events. The framework is responsible for collecting data, filtering and clustering them, detecting and visualizing anomalies in real-time based on these clusters. In this work, we propose the use of microservices to execute each component of the framework. As a case study, the proposed architecture detects anomalies in urban mobility data from Vitória-ES, based on city hall and Twitter data.*

Resumo. *A adoção de plataformas colaborativas tem crescido de forma que agentes públicos buscam cada vez mais parcerias com estes provedores de informação. Dados de câmeras, redes sociais e aplicativos podem contribuir para o gerenciamento de cidades inteligentes, como na detecção de eventos atípicos no trânsito, por exemplo. Este trabalho apresenta um arcabouço que utiliza fontes heterogêneas para detecção de eventos anômalos de tráfego. O arcabouço é responsável por coletar dados, filtrar e agrupá-los, e a partir destes grupos, detectar e visualizar anomalias em tempo real. Neste trabalho, propomos a utilização de microsserviços para executar cada componente do framework. Como estudo de caso, a arquitetura proposta é utilizada para detectar anomalia nos dados de mobilidade urbana, da cidade Vitória-ES, baseado em dados da Prefeitura e do Twitter.*

1. Introdução

Os dados coletados por sensores, câmeras, redes sociais e aplicativos podem contribuir para a detecção automática de eventos atípicos no trânsito. Além disso, os dados gerados por plataformas colaborativas podem enriquecer a massa de dados para monitoramento em uma cidade e diminuir a dependência de infraestruturas públicas legadas, colaborando para a detecção de eventos atípicos [Sidauruk and Ikmah 2018]. Um exemplo é a cidade de Vitória-ES, que ocupou o primeiro lugar do *Ranking Connected Smart Cities*[1] (Cidades Inteligentes e Conectadas) de 2018, entre os municípios brasileiros com até 500 mil habitantes. A Prefeitura possui uma central de videomonitoramento, conectada a várias câmeras na cidade, onde agentes públicos supervisionam as vias da cidade.

[1] www.connectedsmartcities.com.br/resultados-downloads-connected-smart-cities/

Em paralelo, essa central também recebe informações sobre a situação do trânsito em tempo real por meio de uma parceria com o aplicativo *Waze*[2], que agrega informações providas pelos motoristas que utilizam o aplicativo. Interpretar, analisar e gerar novos serviços a partir de toda essa massa de informações se torna uma tarefa árdua, fazendo-se necessária a implantação de soluções para análise integrada e eficaz dos dados de forma a melhorar o monitoramento de eventos e incidentes nas cidades [Montori et al. 2017]. A utilização de mecanismos de alerta automáticos possibilitaria, então, uma melhor gestão da cidade, acelerando as tomadas de decisões em relação a eventos inesperados [de Souza et al. 2018]. Essa grande quantidade e variedade de informações demanda uma arquitetura que permita a integração dos diversos tipos de dados, e ao mesmo tempo seja escalável [Pan et al. 2013].

Em [Thome et al. 2020] é proposto um arcabouço de referência para detecção de anomalias a partir de eventos heterogêneos. O arcabouço coleta dados de fontes heterogêneas para construir um histórico de dados e, a partir deste, identificar as anomalias. O arcabouço foi utilizado para identificar situações anômalas no trânsito de Vitória, utilizando os dados da Prefeitura e de contas públicas voltadas ao trânsito no *Twitter* como entrada para os algoritmos e métodos estatísticos citados em [Thome et al. 2020]. Deste modo, aumentou-se a confiabilidade dos eventos considerados anômalos, pois os mesmos eventos poderiam estar presentes nas diversas fontes simultaneamente. No entanto, Thomé *et al.* desenvolveram um protótipo centralizado do arcabouço proposto com todas as funções em um só programa, o que dificulta a criação e adição de novos serviços, além de comprometer a escalabilidade do sistema.

Este trabalho estende o trabalho proposto em [Thome et al. 2020], ao propor o desacoplamento de todos os módulos e a implementação de uma arquitetura distribuída, baseada em microsserviços, para o arcabouço de detecção de anomalias. Microsserviços têm sido propostos como uma abordagem capaz de isolar as funcionalidades de um sistema. Por este motivo, foi utilizada a ferramenta *Docker*[3] para desacoplar a coleta de dados de fontes heterogêneas, do armazenamento destes dados, dos algoritmos de detecção de anomalia, da visualização de dados e de um sistema de alerta. Dessa forma, utilizando os algoritmos e scripts apresentados e criados em [Thome et al. 2020], é possível emitir um alerta quando o congestionamento em uma via é pior do que o esperado em um determinado horário.

Além disso, este trabalho apresenta uma avaliação paramétrica do arcabouço, permitindo o melhor entendimento do efeito de cada parâmetro de entrada do algoritmo na detecção de anomalias em dados de mobilidade urbana.

O restante deste artigo é organizado como segue: A Seção 2 detalha uma visão geral do arcabouço para detecção de anomalias proposto em [Thome et al. 2020]. A Seção 3 apresenta a arquitetura de implementação proposta neste trabalho. Os resultados são discutidos na Seção 4. Os trabalhos relacionados são apresentados na Seção 5. Por fim, a Seção 6 conclui o trabalho.

[2] https://www.waze.com/
[3] https://www.docker.com/

2. Arcabouço para Detecção de Anomalias

Em [Thome et al. 2020] foi proposto um arcabouço para detecção e alerta de anomalias baseado em fontes heterogêneas de dados que pudesse ser modularizado. As anomalias são observações que fogem do comportamento esperado dos dados e que, neste trabalho, podem representar eventos atípicos de trânsito como acidentes e obras. A solução proposta tem como objetivo detectar anomalias em tempo real, baseando-se no histórico de eventos coletados de fontes heterogêneas.

A Figura 1 apresenta o arcabouço proposto, sendo que cada caixa representa um módulo e suas funcionalidades e as setas representam a comunicação entre os módulos. A arquitetura do arcabouço pode ser dividida em cinco módulos, descritos a seguir:

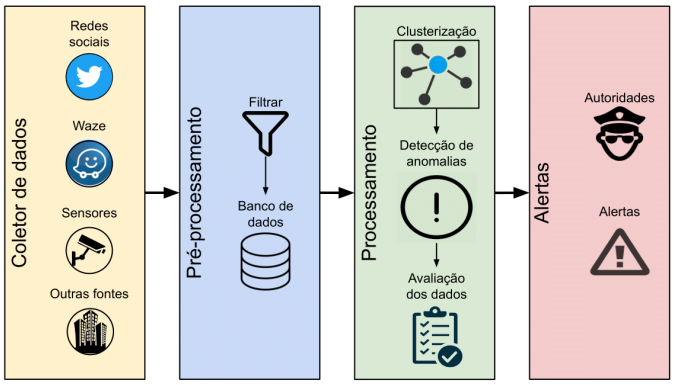


Figura 1. Arcabouço para detecção e alerta de anomalia [Thome et al. 2020].

- Coleta de dados: Representa cada fonte de coleta de dados em tempo real, sendo que estas podem ser modificadas conforme a necessidade [Silva et al. 2016].
- Pré-processamento: Responsável por filtrar e armazenar os dados recebidos do Coletor de Dados. Um filtro deve ser criado para cada fonte de dado, armazenando-a corretamente.
- Processamento: Responsável por classificar observações recebidas do Pré-Processamento, identificar as anomalias e enviar notificações ao Sistema de Alertas, em tempo real. O processo é dividido em três etapas:
  - Gerar série temporal padrão: calcula o maior grupo por período do dia, baseado nas séries temporais de um período de D dias de observações, obtendo uma única série temporal;
  - Detectar anomalias: compara observações em tempo real com a série padrão para as identificar como anomalia; e
  - Avaliar anomalias: analisa o desempenho da detecção de forma automática, checando se existe alguma ocorrência sobre a anomalia em outras fontes de dados.
- Distribuidor de alertas: Recebe os alertas gerados pelo processamento e encaminha para uma autoridade registrada.
- Visualizador de dados: Disponibiliza uma *dashboard* para visualização dos dados coletados.

O arcabouço considera o seguinte modelo de dados:

$$< timestamp, fonte, [E_1, E_2, ..., E_n] >$$

sendo $timestamp$ a data e hora da observação, $fonte$ a descrição da fonte e $E_{1..n}$ um conjunto de atributos descritivos, por exemplo a localização e tipo de evento.

Para detectar uma anomalia em tempo real, o *Processamento* compara um novo evento com um padrão baseado no histórico de observações. Por exemplo, ao receber uma velocidade em uma via, o *Processamento* identifica se ela está na faixa de valores esperados para o horário ou não.

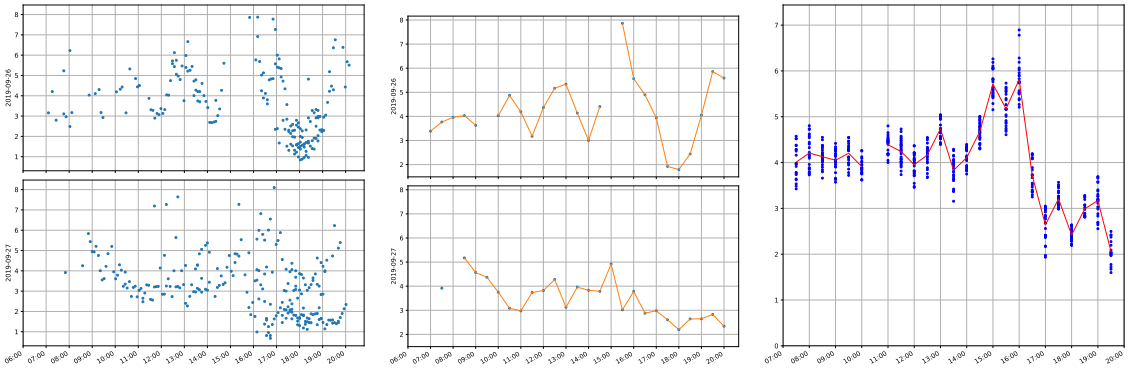
Este padrão do histórico de observações é obtido por meio de uma série temporal padrão, que identifica o padrão de uma via em cada período do dia, usando uma técnica de agrupamento. O algoritmo proposto recebe como parâmetros: *i*) Dados de uma via; *ii*) *D* dias considerados como histórico em uma janela deslizante; *iii*) J_{min} janela em minutos para divisão de dia em períodos; *iv*) Função *Reduction*, para sumarizar com um valor único todas as observações dentro de uma janela J_{min}; *v*) uma Função de agrupamento (*clustering*) das *D* séries temporais diárias. O resultado do algoritmo é um agrupamento dos eventos por janela de tempo, representando uma série temporal padrão. Em [Thome et al. 2020] foram utilizados média e mediana para a função *Reduction* e os algoritmos BIRCH, DBSCAN, K-Means e Optics para função de agrupamento.

Algoritmos de agrupamento até podem ser utilizados para detectar anomalias. Contudo, possuem alto custo computacional. Para permitir que a anomalia seja detectada em tempo real, cada novo evento é comparado com a série temporal por meio de uma técnica de detecção de anomalia por distância. No trabalho, foram utilizados o Z-Score, que calcula a distância normalizada entre uma amostra e a média, e o IQR, que calcula a distância interquartil.

Os autores implementaram um protótipo que coleta dados da Prefeitura de Vitória que, por sua vez, são adquiridos pela plataforma do *Waze*, divididos em três tipos: *Engarrafamento*, *alertas* e *irregularidades*. Além disso, as informações são comparadas com textos extraídos do *Twitter* de contas oficiais que noticiam problemas no tráfego de Vitória. A partir dos dados coletados da Prefeitura e da rede social *Twitter*, o arcabouço identifica quando ocorre uma anomalia. A Figura 2 apresenta as fases do algoritmo descrito acima. O histórico da coleta de dados diários, Fig. 2(a). Cada janela do dia é então sumarizado por uma função de média das observações na janela J_{min}, Fig. 2(b). Por fim, os dados dos D dias sumarizados são agrupados, por meio de algum algoritmo de *clustering*, para formar uma série temporal padrão, Fig. 2(c).

A sazonalidade de eventos urbanos possui certa variância. Por exemplo, engarrafamentos podem acontecer um pouco antes ou depois do horário usual. Assim, a detecção de anomalias utiliza também as janelas imediatamente anterior e posterior ao evento para caracterizar um evento anômalo. No exemplo do engarrafamento, se um evento ocorrer às 9:20, considerando uma janela de 30 minutos, o evento será comparado com a janela de 8:30, de 9:00 e de 9:30, e será considerado anômalo se identificado como uma anomalia nas três janelas.

O protótipo implementado em [Thome et al. 2020] é totalmente centralizado, o que dificulta a criação de novos serviços, como *Dashboards* de visualização, novos algoritmos de detecção de anomalia, etc. Além disso, os autores não apresentaram uma avaliação paramétrica do efeito dos parâmetros sobre a detecção de anomalias.



(a) Dados de velocidade de uma via coletados em dias distintos

(b) Séries Temporais após função Reduction de média a cada janela de 30 minutos ($J_{min} = 30$).

(c) Série temporal resultante do agrupamento dos D dias. Para cada novo evento é calculada a distância ao centro da respectiva janela.

Figura 2. Exemplo de Execução do Algoritmo proposto para o arcabouço.

O presente trabalho estende o trabalho proposto em [Thome et al. 2020] implementando-o como uma arquitetura de microsserviços. Deste modo, é possível aumentar a escalabilidade do sistema, em especial para o monitoramento de grandes cidades. Além disso, este trabalho apresenta uma avaliação paramétrica do arcabouço, permitindo o melhor entendimento do efeito de cada parâmetro de entrada do algoritmo na detecção de eventos anômalos.

3. Arquitetura Distribuída para Coleta de Dados e Detecção de Anomalias

O objetivo principal deste trabalho é estender o arcabouço proposto em [Thome et al. 2020] utilizando uma arquitetura distribuída. Para isso, primeiramente, foi necessário pesquisar uma ferramenta que possibilitasse a implementação do arcabouço de modo distribuído. O *Docker* foi a ferramenta escolhida devido ao fato desta prover uma arquitetura de criação de microsserviços, por meio de contêineres. Adicionalmente, os contêineres podem ser vistos como máquinas virtuais modulares e extremamente leves, que possibilitam uma maior flexibilidade para criar, implantar, copiar e migrar contêineres [Saha et al. 2018]. Em resumo, a conteinerização tem como vantagens isolar cada módulo; garantir a portabilidade; e facilitar o gerenciamento.

A Figura 3 apresenta a arquitetura distribuída proposta, dividida em contêineres. Os nomes nas caixas representam as aplicações, enquanto as cores correspondem aos módulos da Figura 1.

As linhas tracejadas representam contêineres, a superposição de caixas representa a possibilidade de múltiplos contêineres de um módulo, as setas representam o sentido das informações e os rótulos delas o tipo dos dados passados. As ferramentas usadas em cada um dos contêineres para que se adequassem às necessidades do arcabouço são detalhadas a seguir:

- Coleta de dados: Foram utilizadas duas fontes de dados, como contêineres independentes que executam continuamente:
  - *Twitter*: Para criação do contêiner de coleta de tweets foi utilizada a API

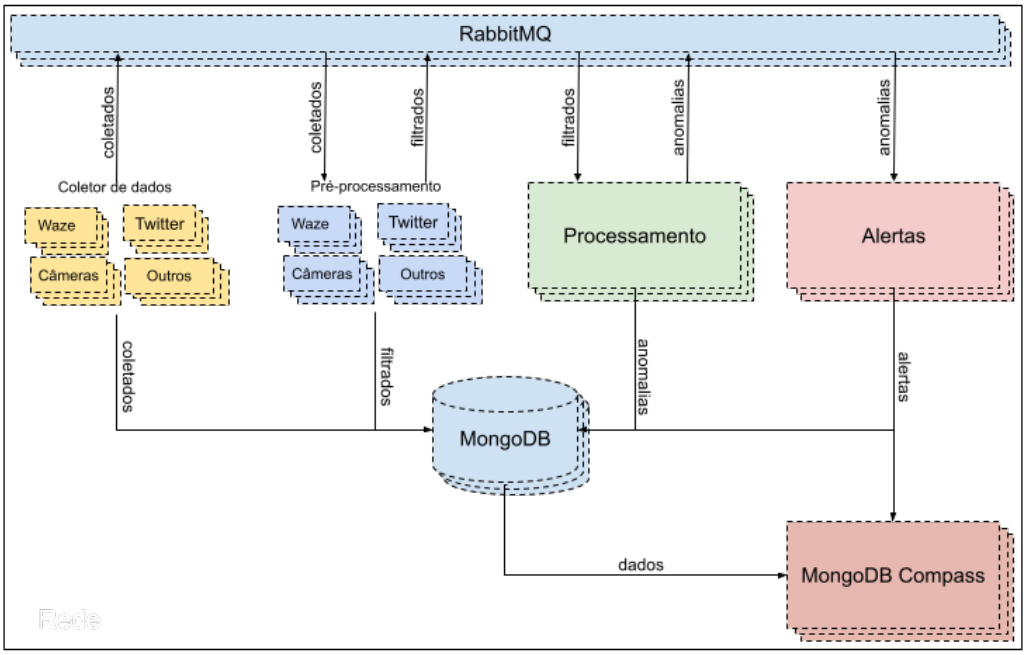


Figura 3. Arquitetura distribuída do arcabouço de detecção de anomalias: as linhas tracejadas representam contêineres e as setas representam o sentido das informações (rotuladas de acordo com o tipo dos dados).

tweepy[4], que possibilita encontrar tweets em contas e hashtags específicas.

  - *Waze* (API da Prefeitura): A API, disponibilizada pela Prefeitura de Vitória, é utilizada para coleta dos dados Waze. Esses dados contêm informações sobre acidentes, alertas e irregularidades fornecidos à prefeitura pelo aplicativo por meio de um convênio. Cada um destes encontra-se em um contêiner diferente.

- Pré-processamento: Contêineres criados para padronizar os dados de acordo com as necessidades do usuário e do contêiner de processamento. Dividido em quatro contêineres: um para o *Twitter*, e três para a API da Prefeitura, que fornece três fontes de dados: *Alertas*, *Acidentes* e *Irregularidades*.
- Processamento: Contêineres criados para detectar anomalias de trânsito e informar o sistema de alertas em tempo real. Pode haver um número variável de contêineres, que são baseados nas N ruas com maior número de dados.
- *Broker* de mensagens: Responsável pela comunicação entre os contêineres. Utilizamos o RabbitMQ[5], que possui uma imagem pré-existente para Docker.
- Banco de dados: Para armazenamento do banco de dados foi utilizado o MongoDB [6], um Banco de dados NoSQL que permite dividir os bancos em *collections*.
- Distribuidor de alertas: Interface web que recebe os alertas gerados pelos contêineres de processamento e envia mensagens para os números de celulares e e-mail das autoridades cadastradas. Tal cadastro também ocorre por esta interface web.

[4] https://www.tweepy.org/
[5] https://www.rabbitmq.com/
[6] https://www.mongodb.com/

- Visualizador de dados: O MongoDB Compass[7], um visualizador de dados integrado ao MongoDB, foi utilizado como Dashboard do novo arcabouço.

A seguir, detalhamos as ferramentas utilizadas para a composição da arquitetura apresentada na Figura 3.

- Docker:
  - *DockerFile* é um arquivo utilizado para criar uma imagem personalizada para contêineres, que especifica o que será instalado, qual imagem será usada como base, o comando básico que o contêiner irá executar, quais arquivos serão copiados para dentro do contêiner, entre outros parâmetros.
  - *docker-compose* é o arquivo em que está especificado todo o ambiente, apresentando quais são os contêineres a serem criados, a quantidade destes, suas dependências, seus volumes, a qual rede eles estão conectados, se o contêiner irá subir sozinho após terminar de executar o comando básico, a quais portas o contêiner está conectado, entre outras informações.
- RabbitMQ:
  - *Fila*: Onde as mensagens ficam armazenadas e de onde são retiradas. Apresenta um nome específico facilitando assim a divisão e o acesso por parte dos *consumers*.
  - *Publisher*: Responsável por incluir cada nova mensagem na fila, podendo ser qualquer contêiner que esteja conectado na mesma rede que o contêiner do RabbitMQ.
  - *Consumer*: Responsável por consumir (retirar) a informação da fila, podendo ser qualquer contêiner que esteja conectado na mesma rede que o contêiner do RabbitMQ.

4. Avaliação e Discussão

O protótipo do arcabouço desenvolvido foi avaliado utilizando os dados de engarrafamento, disponibilizados na API da Prefeitura de Vitória, e os dados coletados do *Twitter*.

A arquitetura, apresentada na Figura 3, foi implementada e configurada em um data center do Laboratório de Núcleo de Estudos em Redes Definidas por *software* (LabNerds)[8]. O datacenter oferece serviço de alocação dinâmica de recursos. As especificações internas dos contêineres foram realizadas utilizando diversos DockerFiles, facilitando a criação personalizada de cada tipo de contêiner. Já as características externas dos contêineres foram explicitadas no *docker-compose*, em que cada contêiner foi conectado a uma mesma rede, foi habilitada a função de subir automaticamente após cair, qual DockerFile utilizar e, para o banco de dados, em qual pasta os Volumes serão conectados.

Os contêineres foram configurados para coletar dados de trânsito fornecidos pela Prefeitura a cada 5 minutos. Além disso, há o monitoramento contínuo de contas oficiais com notícia sobre trânsito da cidade de Vitória. Os dados são armazenados em um banco de dados MongoDB.

[7] https://www.mongodb.com/products/compass
[8] http://nerds.ufes.br

4.1. Avaliação Paramétrica do Arcabouço

Como discutido anteriormente, [Thome et al. 2020] apresentou um protótipo centralizado do arcabouço e comparou os algoritmos de agrupamento. Neste trabalho, apresentamos uma análise dos parâmetros do arcabouço. Para isto, foram realizados experimentos permutando os parâmetros do arcabouço, como explicitado na Tabela 1. A função *reduction* utilizou a média simples para sumarizar os dados a cada janela.

A fim de avaliar o desempenho do protótipo e encontrar a melhor configuração, utilizando os dados de engarrafamento da cidade de Vitória entre outubro/2018 e novembro/2019, foi calculada a taxa de acerto de anomalia. Uma anomalia é considerada certa se existe um *tweet*, no espaço próximo de tempo de 30 minutos, que mencione a rua relativa à anomalia. Assim, tomando k como um dia, a taxa média de acerto de anomalia R_k é calculada como:

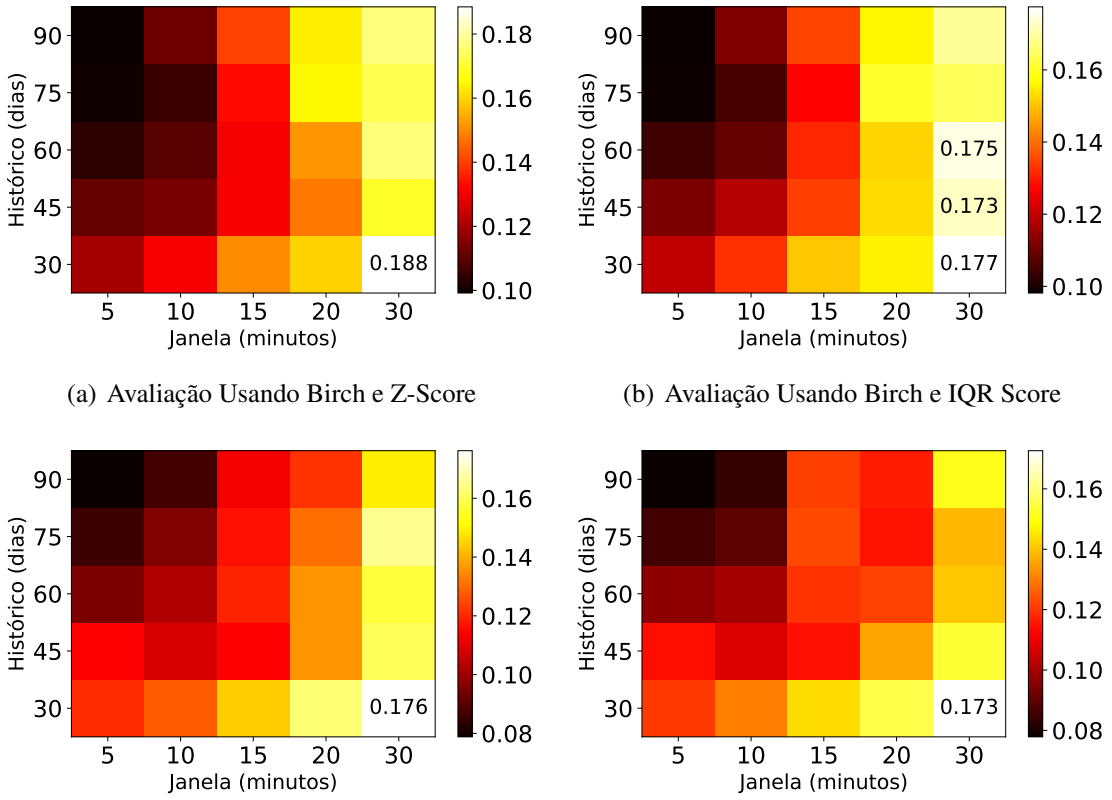
$$\text{seja } T_k \text{ o número de anomalias em } k \text{ que estão relacionadas a um evento no Twitter}$$
$$\text{seja } S_k \text{ a soma de anomalias detectadas em } k$$
$$\text{a taxa de acerto em } k \text{ é } R_k = \begin{cases} T_k/S_k, & \text{se } S_k > 0 \\ 0, & \text{se } S_k = 0 \end{cases}$$
$$\text{a taxa média de acerto de anomalias é } R = \textstyle\sum_k R_k/k$$

Parâmetro	Valores utilizados
Algoritmo de clusterização, $Clustering$	K-Means com 3 clusters Birch, DBSCAN, OPTICS
Método de classificação de anomalia	Z-Score, IQR
Tamanho do histórico em dias, D	30, 45, 60, 75, 90
Tamanho da janela em minutos, $Jmin$	5, 10, 15, 20, 30

Tabela 1. Parâmetros de Entrada para Avaliar o Protótipo do Arcabouço.

As taxas médias de anomalias certas R são apresentadas como mapas de calor para cada configuração do experimento, sendo que as células correspondentes aos 10% melhores resultados explicitam o valor R. As Figuras 4 e 5 apresentam os resultados das avaliações do protótipo, R, por meio de mapas de calor. Primeiramente, nota-se que um histórico de 30 dias e uma janela de 30 minutos gerou, na maior parte dos casos, os melhores resultados.

Essa informação sugere que eventos mais recentes devem possuir maior valor nas detecções de anomalias que os mais antigos, o que poderia ser justificado pelo fato de o trânsito apresentar comportamento sazonal em diferentes períodos, semanalmente, por conta da movimentação característica de dias úteis, e mensalmente, por conta de períodos de férias escolares, feriados ou obras na infraestrutura. Além disso, de acordo com os resultados, os efeitos de eventos de mobilidade urbana aparentam possuir uma certa inércia, de forma que as consequências aparentam durar até uma hora. Isto pode refletir o fato de incidentes de trânsito possivelmente causarem um efeito dominó, reduzindo a velocidade dos veículos próximos, gerando engarrafamento, que somente retorna à normalidade algum tempo após a solução do incidente. Entre os algoritmos de clusterização e técnicas de detecção de anomalias, a configuração K-Means, com 3 clusters, e Z-Score apresentou melhores valores da média das taxas diárias.



(a) Avaliação Usando Birch e Z-Score

(b) Avaliação Usando Birch e IQR Score

(c) Avaliação Usando DBSCAN e Z-Score

(d) Avaliação Usando DBSCAN e IQR Score

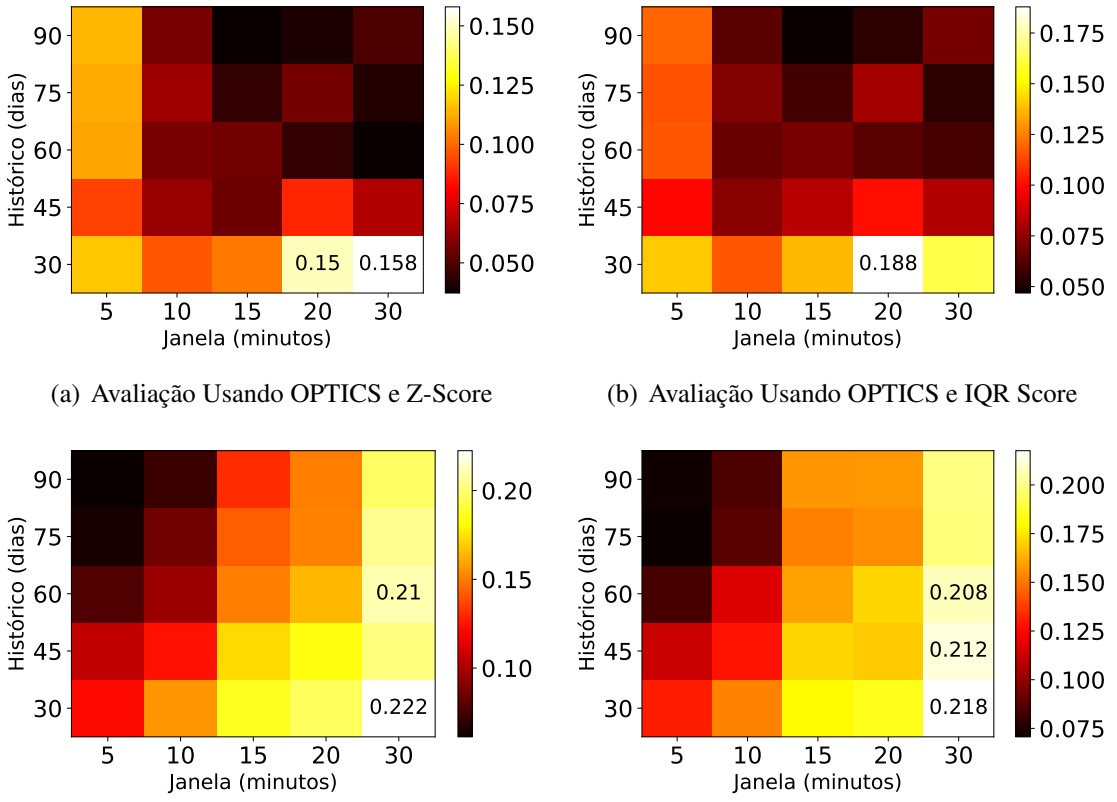
Figura 4. Avaliação do Protótipo com Birch e DBSCAN.

4.2. Visualização de dados

A Figura 6 ilustra dois exemplos de visualização de dados fornecidas pelo *Mongo Compass*. A Fig. 6(a) apresenta a distribuição de eventos por rua disponibilizados pela Prefeitura. Pode-se observar que um conjunto de poucas ruas (18) acumulam 2% ou mais dos eventos coletados. Considerando que cada rua é monitorada por um contêiner de processamento, esse resultado indica que para uma cidade de porte médio, o sistema proposto suporta a quantidade de ruas da cidade que possuem eventos significativos para serem monitoradas.

A Fig. 6(b) apresenta a distribuição das velocidades nos eventos de congestionamento informados para a Av. Rio Branco, a avenida com maior número de eventos. Pode-se observar que esta avenida apresenta uma distribuição quase-normal para a velocidade média nos eventos de congestionamento.

A modularização por microsserviços proposta neste projeto permite maior flexibilidade, manutenibilidade, além de permitir sua extensão, como para adicionar novos algoritmos e/ou fontes de dados, em relação ao proposto em [Thome et al. 2020]. Ressalta-se que este é um projeto em andamento e está em desenvolvimento contínuo. Com isto, a criação do contêiner de alertas e o método de decisão do número de contêineres de processamento a serem criados ainda serão incluídos.



(a) Avaliação Usando OPTICS e Z-Score

(b) Avaliação Usando OPTICS e IQR Score

(c) Avaliação Usando K-Means com 3 Clusters e Z-Score

(d) Avaliação Usando K-Means com 3 Clusters e IQR Score

Figura 5. Avaliação do Protótipo com OPTICS e K-Means com 3 Clusters.

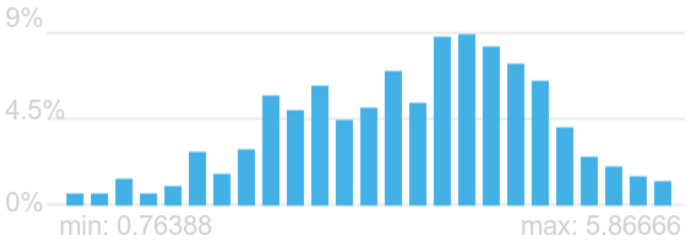
5. Trabalhos relacionados

Em [Calikus et al. 2020] é apresentado um *framework* generalista para detecção de anomalias, compatível com uma abstração das etapas usadas como base em [Thome et al. 2020], sendo estas: adaptação da entrada, busca por um padrão de dados comuns, cálculo de distância de observações ao padrão e classificação das observações. O artigo prossegue com a escolha de algumas tecnologias para cada passo, avaliando diversas combinações. Na segunda etapa, no entanto, os métodos apresentados não consideram comportamentos cíclicos, sendo os mesmos considerados neste trabalho.

Neste trabalho as fontes de dados utilizadas foram o *Twitter* e dados de trânsito da Prefeitura de Vitória, obtidos por meio de convênio com o aplicativo *Waze*, desta forma sendo muito semelhante a [Sidauruk and Ikmah 2018] no quesito de coleta de dados.

Ressalta-se que este trabalho se diferencia das demais propostas por prover uma arquitetura modularizada, com cada módulo implementado em contêineres independentes. Dessa forma, facilitando a manutenção e diminuindo o risco de falha do sistema causado por um erro em uma parte específica da arquitetura. Isso ocorre pois em ambos os casos, como o sistema é conteinerizado, é possível subir novos contêineres das partes que apresentaram falhas para substituí-las, impedindo assim que o sistema inteiro pare de funcionar.

(a) Distribuição de eventos por rua coletados pela API da Prefeitura(*Waze*) até o dia 09/10/2020.



(b) Distribuição da velocidade (em m/s) na Av. Rio Branco, coletados utilizando a API da Prefeitura(*Waze*).

Figura 6. Dados coletados utilizando a API da Prefeitura(*Waze*) pelo contêiner que coleta em tempo real.

6. Conclusões

Este trabalho apresentou uma implementação em uma arquitetura distribuída e escalável de um arcabouço para gerenciamento de dados de mobilidade urbana baseado em fontes de dados heterogêneas. Este arcabouço implementa um modelo de série temporal que sumariza os dados em intervalos de tempo ao longo das 24hs do dia, detectando anomalias e gerando alertas. Além disso, foi realizado um estudo paramétrico para identificar as melhores configurações para o arcabouço.

Os resultados sugerem o uso de K-Means com 3 clusters para encontrar a série padrão, representando as faixas de velocidades mais comuns em cada período do dia, e Z-Score para detectar as anomalias a partir dos maiores agrupamentos obtidos. Essa configuração foi utilizada, então, para o desenvolvimento do sistema distribuído de detecção de anomalia. A partir dos resultados apresentados, notou-se baixos valores para as taxas de acerto, possivelmente consequência do baixo engajamento social na rede utilizada na métrica de avaliação dentro da cidade.

A extensão proposta por este trabalho desacoplou o arcabouço em microsserviços independentes para coleta de dados, processamento, detecção de anomalias e alertas. Além disso, esta versão conteinerizada permite que novas tecnologias, algoritmos ou serviços sejam adicionados de modo fácil e independente de outras partes do arcabouço.

Na versão corrente, o número de contêineres do módulo de processamento é fixado a partir do número de ruas monitoradas. Como trabalhos futuros pretende-se implementar um algoritmo de decisão que defina o número de contêineres do módulo de processamento dinamicamente. Além disso, a *dashboard* será integrada ao sistema de alerta, que ainda está em desenvolvimento.

Por fim, serão realizados testes de cargas e escalabilidade, comparando-o à versão centralizada (original) do arcabouço. Desta forma, pretende-se chegar a um arcabouço distribuído capaz de garantir o monitoramento de cidades que possuem informações de tráfego em maiores quantidades.

Agradecimentos

O presente trabalho foi realizado com apoio do Programa Institucional de Iniciação Científica da UFES, da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001, do CNPq, da Fundação de Amparo à Pesquisa do Espírito Santo (FAPES) e FAPESP (Grants #2018/23011-1 e #2020/05182-3). Adicionalmente, este trabalho foi viabilizado por meio do termo de cooperação técnica 004/2018, entre a Secretaria Municipal de Segurança Pública de Vitória-Espírito Santo e a UFES. Os autores agradecem o esforço da Secretaria pela disponibilização dos dados.

Referências

Calikus, E., Nowaczyk, S., Sant'Anna, A., and Dikmen, O. (2020). No free lunch but a cheaper supper: A general framework for streaming anomaly detection. *Expert Systems with Applications*, 155:113453.

de Souza, A. M., Botega, L. C., Garcia, I. C., and Villas, L. A. (2018). Por aqui é mais seguro: Melhorando a mobilidade e a segurança nas vias urbanas. In *XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, Porto Alegre, RS, Brasil. SBC.

Montori, F., Bedogni, L., and Bononi, L. (2017). A collaborative internet of things architecture for smart cities and environmental monitoring. *IEEE Internet of Things Journal*, 5(2):592–605.

Pan, B., Zheng, Y., Wilkie, D., and Shahabi, C. (2013). Crowd sensing of traffic anomalies based on human mobility and social media. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 334–343.

Saha, P., Beltre, A., Uminski, P., and Govindaraju, M. (2018). Evaluation of docker containers for scientific workloads in the cloud. In *Proceedings of the Practice and Experience on Advanced Research Computing*, pages 1–8.

Sidauruk, A. and Ikmah (2018). Congestion correlation and classification from twitter and waze map using artificial neural network. In *International Conference on Information Technology, Information System and Electrical Engineering*, pages 224–229.

Silva, T. H., Celes, C., Neto, J., Mota, V., Cunha, F., Ferreira, A., Ribeiro, A., Vaz de Melo, P., Almeida, J., and Loureiro, A. (2016). *Users in the urban sensing process: Challenges and research opportunities*. Academic Press.

Thome, M., Neves, A., Gomes, R., and Mota, V. (2020). Um arcabouço para detecção e alerta de anomalias de mobilidade urbana em tempo real. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, volume XXXVIII, pages 1–14.