Uma Arquitetura de Microsserviços para Análise de Dados de Mobilidade Urbana baseada em Dados 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. *Data gathered by sensors, cameras, social networks and apps can contribute to the automatic detection of unusual traffic events. Furthermore, the heterogeneous nature of data sources brings the advantage of information redundancy, which can increase the degree of reliability of a detected event. This work extends the implementation of a framework for detecting anomalous traffic events by using a microservices architecture. For this, the framework was decomposed in microservices to collect data, filter and group them as time series, detect anomalies and issue real-time alerts. As a case study, the proposed architecture is used to detect anomalies in real-time urban mobility data from the city of Vitória-ES, based on data from the city hall and Twitter.*

Resumo. *Os dados coletados por sensores, câmeras, redes sociais e aplicativos podem contribuir para a detecção automática de eventos atípicos no trânsito. Além disso, a variedade das fontes de dados traz como vantagem a redundância de informações, o que pode aumentar o grau de confiabilidade sobre um evento detectado. Este trabalho estende a implementação de um arcabouço para detecção de eventos anômalos de tráfego por meio de uma arquitetura de microsserviços. Para isto, o arcabouço foi decomposto em microsserviços para coletar dados, filtrar e agrupá-los como séries temporais, detectar e visualizar anomalias em tempo real. Como estudo de caso, a arquitetura proposta é utilizada para detectar anomalia nos dados de mobilidade urbana em tempo real, da cidade Vitória-ES, baseado em dados da prefeitura e do Twitter.*

1. Introdução

A adoção de plataformas colaborativas por parte dos cidadãos tem crescido de forma que agentes públicos buscam cada vez mais o estabelecimento de parcerias com estes provedores de informação. Os dados gerados por essas plataformas 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 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 e com todas as funções em um só programa do arcabouço proposto, o que dificulta a criação e adição de novos serviços, além de aspectos de 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 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.

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. 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. A Figura 1(a) 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:

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

- 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: Tem como tarefa a filtragem e armazenamento de dados.
- Processamento: Recebe os dados padronizados no pré-processamento, agrupa-os em séries temporais e analisa os dados utilizando os algoritmos que determinam as possíveis anomalias, assim gerando eventuais alertas [Fahad et al. 2014].
- Distribuidor de alertas: Recebe os alertas gerados pelo processamento e encaminha para uma autoridade registrada.
- Visualizador de dados: É responsável pela visualização dos banco de dados adquiridos.

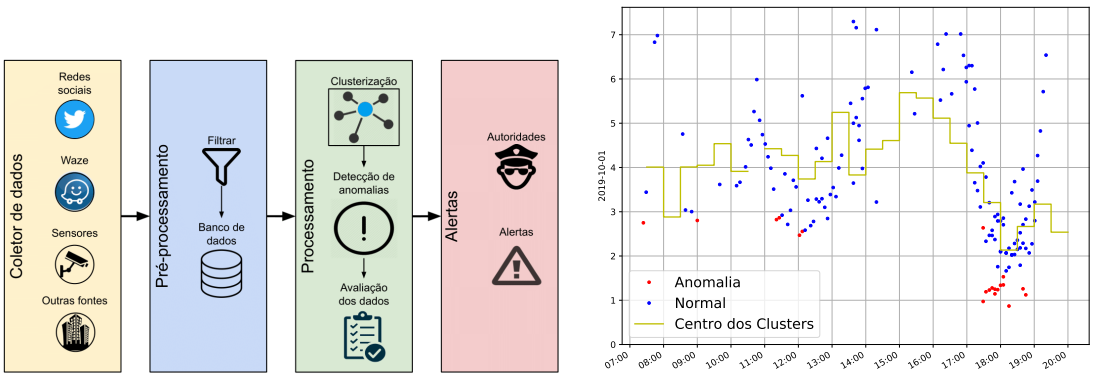
O arcabouço considera o seguinte modelo de dados:

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

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.

As anomalias são detectadas a partir do cálculo do maior *cluster* baseado em séries temporais de um período de X dias. Deste modo, novos eventos são comparados com o *cluster* do histórico para classificá-los como normais ou anomalias.

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 1(b) apresenta os eventos e anomalias detectados no dia 1º de Outubro de 2019. O protótipo implementado inicialmente é totalmente centralizado e como um programa único, o que dificulta a criação de novos serviços, como *Dashboards* de visualização, novos algoritmos de detecção de anomalia, etc. O objetivo do presente trabalho é estender o trabalho proposto em [Thome et al. 2020] e implementá-lo como uma arquitetura de microsserviços.



(a) Arcabouço (b) Eventos no dia 01/10/2019

Figura 1. Arcabouço para detecção e alerta de anomalia proposto em [Thome et al. 2020] e resultado da observação ao longo de um dia.

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

O objetivo principal deste trabalho é implementar 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.

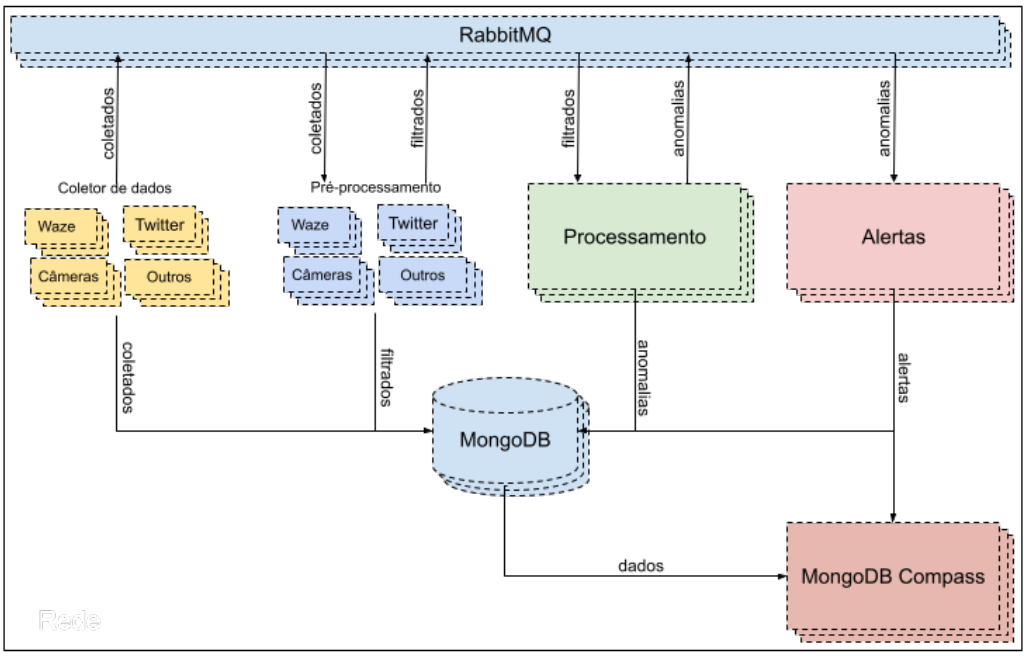


Figura 2. Arquitetura distribuída do arcabouço de detecção de anomalias: as linhas tracejadas representam contêineres e as setas representam o sentido das informações (rotuladas de acordo com o tipo dos dados).

A Figura 2 apresenta a arquitetura distribuída proposta, dividida em contêineres. Os nomes nas caixas representam as aplicações, enquanto as cores correspondem aos módulos da Figura 1(a) 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 tweepy[4], que possibilita encontrar tweets em contas e hashtags específicas.

[4] `https://www.tweepy.org/`

  - *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 baseado nas N ruas com maior número de dados.
- *Broker* de mensagens: Responsável pela comunicação entre os contêineres. Utilizamos o RabbitMQ[3], que possui uma imagem pré-existente para Docker.
- Banco de dados: Para armazenamento do banco de dados foi utilizado o MongoDB[4], 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.
- Visualizador de dados: O MongoDB Compass[5], 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 2.

- 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 um 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 especifico 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.

[3]https://www.rabbitmq.com/
[4]https://www.mongodb.com/
[5]`https://www.mongodb.com/products/compass`

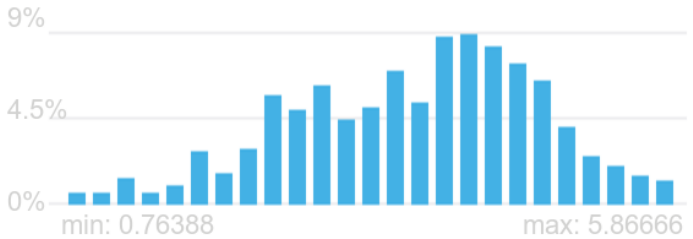
4. Resultados e Discussão

A arquitetura, representada na Figura 2, foi implementada e configurada em um data center do Laboratório de Núcleo de Estudos em Redes Definidas por *software* (LabNerds)[6], que permite 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. Continuamente eles monitoram contas oficiais com notícia sobre trânsito da cidade de Vitória. Os dados são armazenados no MongoDB.

A Figura 3 ilustra dois exemplos de visualização de dados fornecidas pelo *Mongo Compass*. A Fig. 3(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. 3(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) 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 3. Dados coletados utilizando a API da prefeitura(*Waze*) pelo contêiner que coleta em tempo real.

[6]http://nerds.ufes.br

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.

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.

Este trabalho se diferencia das demais propostas por prover uma arquitetura modularizada, com cada módulo implementado em contêineres independentes. Desta 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.

6. Conclusões

O objetivo principal deste trabalho é a 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, e detectando anomalias e gerando alertas.

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 e da Fundação de Amparo à Pesquisa do Espírito Santo (FAPES). 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 em tempo real.

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.

Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S., and Bouras, A. (2014). A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE transactions on emerging topics in computing*, 2(3):267–279.

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.