

Uma Implementação Distribuída em Névoa do Algoritmo de Detecção de Novidade em Fluxos de Dados MINAS

Luís Henrique Puhl de Souza

Orientador: Prof. Dr. Hermes Senger

Fevereiro 2020

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

1. Introdução
2. Fundamentos
3. Estado da Arte e Trabalhos Relacionados
4. Proposta
5. Resultados Preliminares
6. Considerações Finais

Introdução

- Crescimento do número de dispositivos IoT e riscos associados;

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;
 - Exemplo: *Botnet* mirai, infectando cameras e roteadores, gerou 620 Gb/s (KAMBOURAKIS; KOLIAS; STAVROU, 2017).

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;
 - Exemplo: *Botnet* mirai, infectando cameras e roteadores, gerou 620 Gb/s (KAMBOURAKIS; KOLIAS; STAVROU, 2017).
- Detecção de intrusão em redes:

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;
 - Exemplo: *Botnet* mirai, infectando cameras e roteadores, gerou 620 Gb/s (KAMBOURAKIS; KOLIAS; STAVROU, 2017).
- Detecção de intrusão em redes:
 - detecção por assinatura;

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;
 - Exemplo: *Botnet* mirai, infectando cameras e roteadores, gerou 620 Gb/s (KAMBOURAKIS; KOLIAS; STAVROU, 2017).
- Detecção de intrusão em redes:
 - detecção por assinatura;
 - detecção por anomalia;

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;
 - Exemplo: *Botnet* mirai, infectando cameras e roteadores, gerou 620 Gb/s (KAMBOURAKIS; KOLIAS; STAVROU, 2017).
- Detecção de intrusão em redes:
 - detecção por assinatura;
 - detecção por anomalia;
 - ambiente de névoa e redes IoT.

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;
 - Exemplo: *Botnet* mirai, infectando cameras e roteadores, gerou 620 Gb/s (KAMBOURAKIS; KOLIAS; STAVROU, 2017).
- Detecção de intrusão em redes:
 - detecção por assinatura;
 - detecção por anomalia;
 - ambiente de névoa e redes IoT.
- Um sistema para detecção de intrusão em Redes IoT implementando em névoa;

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;
 - Exemplo: *Botnet* mirai, infectando cameras e roteadores, gerou 620 Gb/s (KAMBOURAKIS; KOLIAS; STAVROU, 2017).
- Detecção de intrusão em redes:
 - detecção por assinatura;
 - detecção por anomalia;
 - ambiente de névoa e redes IoT.
- Um sistema para detecção de intrusão em Redes IoT implementando em névoa;
- A hipótese do trabalho é que o algoritmo MINAS pode ser distribuído em nós de nuvem e névoa reduzindo a latência e com pouco comprometimento na qualidade de detecção.

Fundamentos

- Métodos Detecção de Novidade;
- Ambientes de computação Distribuída;
- Plataformas de processamento distribuído de fluxos.

Definição de Fluxo de Dados

Um fluxo de dados (*Data Stream*) é uma sequência massiva possivelmente ilimitada de exemplos multi-dimensionais

$x_1, x_2, \dots, x_n, \dots$ recebidos em instantes associados $t_1, t_2, \dots, t_n, \dots$

Métodos Detecção de Novidade

Métodos Detecção de Novidade (*Novelty Detection*) lidam com o reconhecimento e classificação de exemplos em padrões que diferem de padrões anteriores (PERNER, 2007; GAMA; RODRIGUES, 2010).

Conforme Gama e Rodrigues (2010), são características de fluxos de dados contínuos:

- Evolução de Conceito (*Concept Evolution*): surgimento de um conceito durante o fluxo;

Métodos Detecção de Novidade

Métodos Detecção de Novidade (*Novelty Detection*) lidam com o reconhecimento e classificação de exemplos em padrões que diferem de padrões anteriores (PERNER, 2007; GAMA; RODRIGUES, 2010).

Conforme Gama e Rodrigues (2010), são características de fluxos de dados contínuos:

- Evolução de Conceito (*Concept Evolution*): surgimento de um conceito durante o fluxo;
- Mudança de Conceito (*Concept Drift*, deriva ou desvio): modificação da distribuição de um padrão conhecido. A modificação pode ser repentina, incremental ou recorrente;

Métodos Detecção de Novidade

Métodos Detecção de Novidade (*Novelty Detection*) lidam com o reconhecimento e classificação de exemplos em padrões que diferem de padrões anteriores (PERNER, 2007; GAMA; RODRIGUES, 2010).

Conforme Gama e Rodrigues (2010), são características de fluxos de dados contínuos:

- Evolução de Conceito (*Concept Evolution*): surgimento de um conceito durante o fluxo;
- Mudança de Conceito (*Concept Drift*, deriva ou desvio): modificação da distribuição de um padrão conhecido. A modificação pode ser repentina, incremental ou recorrente;
- Ruído e *Outliers*: que não pertencem a um conceito ou pertencem a um conceito porém estão fora da distribuição conhecida.

Algoritmo MINAS

Algoritmo e suas estratégias:

- Modelo de aprendizado *Offline-Online*;

Algoritmo MINAS

Algoritmo e suas estratégias:

- Modelo de aprendizado *Offline-Online*;
- Transformação dos dados analisados para o espaço \mathbb{R}^d ;

Algoritmo MINAS

Algoritmo e suas estratégias:

- Modelo de aprendizado *Offline-Online*;
- Transformação dos dados analisados para o espaço \mathbb{R}^d ;
- Modelo de classificação com *Clusters*;

Algoritmo MINAS

Algoritmo e suas estratégias:

- Modelo de aprendizado *Offline-Online*;
- Transformação dos dados analisados para o espaço \mathbb{R}^d ;
- Modelo de classificação com *Clusters*;
- Função de classificação baseada em distância euclideana;

Algoritmo MINAS

Algoritmo e suas estratégias:

- Modelo de aprendizado *Offline-Online*;
- Transformação dos dados analisados para o espaço \mathbb{R}^d ;
- Modelo de classificação com *Clusters*;
- Função de classificação baseada em distância euclideana;
- Algoritmo de agrupamento para identificação de novos padrões;

Algoritmo MINAS

Algoritmo e suas estratégias:

- Modelo de aprendizado *Offline-Online*;
- Transformação dos dados analisados para o espaço \mathbb{R}^d ;
- Modelo de classificação com *Clusters*;
- Função de classificação baseada em distância euclideana;
- Algoritmo de agrupamento para identificação de novos padrões;
- Classificação de novos padrões entre recorrência, extensão e novidade;

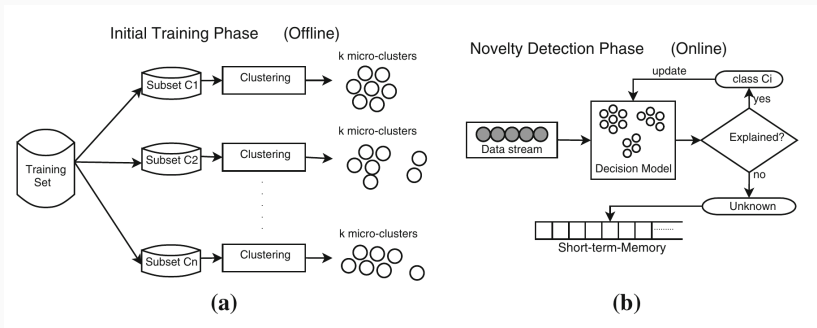


Figura 1: Visão geral do algoritmo MINAS com fases *Offline* (a) e *Online* (b) (FARIA; CARVALHO; GAMA, 2015)

Algoritmo 1: MINAS, trecho de classificação

Entrada: *Modelo, FCD, params, MemTmp, MemSleep*

```
1: for all exemplo  $\in$  FCD do
2:   (Dist, micro)  $\leftarrow$  micro-mais-proximo(exemplo, Modelo)
3:   if Dist < raio(micro) then
4:     exemplo.classe  $\leftarrow$  micro.rotulo
5:     atualizar-micro(micro, exemplo)
6:   else
7:     exemplo.classe  $\leftarrow$  desconhecido
8:     MemTmp  $\leftarrow$  MemTmp  $\cup$  exemplo
9:     if  $|MemTmp| \geq params.NumMinExemplos$  then
10:      Modelo  $\leftarrow$  deteccao-novidade(Modelo, MemTmp, params)
11:    end if
12:  end if
13:  gerenciamento-memoria(...)
14: end for
```

Extensões do Algoritmo MINAS

- FuzzyND: extensão do algoritmo original para classificação com conjunto de etiquetas *fuzzy* (Da Silva et al., 2018; SILVA, 2018);
- MINAS-LC e MINAS-BR: extensão do algoritmo original tratando classificação multi-etiquetas (COSTA et al., 2019; COSTA, 2019);

Ambientes de computação Distribuída

- Computação em Nuvem (*Cloud Computing*) é um modelo que permite acesso conveniente a recursos computacionais compartilhados (MELL; GRANCE, 2012)
- **Características Essenciais:**
 - Auto-serviço sob demanda,
 - Amplo acesso à rede,
 - Agrupamento de recursos,
 - Rápida elasticidade,
 - Serviço mensurado;
- **Modelo de Serviço:**
 - *Software* (SaaS),
 - Plataforma (PaaS),
 - Infraestrutura (IaaS),
- **Implementações:**
 - Nuvem privada,
 - Nuvem comunitária,
 - Nuvem pública,
 - Nuvem híbrida.

Ambientes de computação Distribuída

- Computação de Borda (*Edge Computing*):
Refere-se a qualquer recurso computacional ou de rede entre os dispositivos de borda e centro de dados hospedados em nuvem (SHI et al., 2016).

Ambientes de computação Distribuída

- Computação em Névoa (*Fog Computing*)

Uma arquitetura horizontal a nível de sistema que distribui funções de computação, armazenamento, controle e rede próximos aos usuários no espaço contínuo nuvem-coisa (IEEE Communications Society, 2018).

Características:

- Mobilidade,
- Heterogeneidade,
- Baixa Latência,
- Distribuição geográfica,
- Alto número de nós,
- Interoperabilidade e federação,
- Uso de fluxo de dados e aplicações em tempo real.

Plataformas de processamento distribuído

- Mineração de Dados:

Plataformas de processamento distribuído

- Mineração de Dados:
 - *MapReduce e Apache Hadoop;*

Plataformas de processamento distribuído

- Mineração de Dados:
 - *MapReduce* e *Apache Hadoop*;
 - *Apache Spark* com *Resilient Distributed Dataset - RDD*;

Plataformas de processamento distribuído

- Mineração de Dados:
 - *MapReduce* e *Apache Hadoop*;
 - *Apache Spark* com *Resilient Distributed Dataset - RDD*;
- Mineração de Fluxo de Dados:

Plataformas de processamento distribuído

- Mineração de Dados:
 - *MapReduce* e *Apache Hadoop*;
 - *Apache Spark* com *Resilient Distributed Dataset - RDD*;
- Mineração de Fluxo de Dados:
 - *Apache Spark Streaming* com estratégia de *micro-batching*;

Plataformas de processamento distribuído

- Mineração de Dados:
 - *MapReduce* e *Apache Hadoop*;
 - *Apache Spark* com *Resilient Distributed Dataset* - *RDD*;
- Mineração de Fluxo de Dados:
 - *Apache Spark Streaming* com estratégia de *micro-batching*;
 - *Apache Storm*;

Plataformas de processamento distribuído

- Mineração de Dados:
 - *MapReduce* e *Apache Hadoop*;
 - *Apache Spark* com *Resilient Distributed Dataset - RDD*;
- Mineração de Fluxo de Dados:
 - *Apache Spark Streaming* com estratégia de *micro-batching*;
 - *Apache Storm*;
 - *Apache Flink*;

Plataformas de processamento distribuído

- Mineração de Dados:
 - *MapReduce* e *Apache Hadoop*;
 - *Apache Spark* com *Resilient Distributed Dataset - RDD*;
- Mineração de Fluxo de Dados:
 - *Apache Spark Streaming* com estratégia de *micro-batching*;
 - *Apache Storm*;
 - *Apache Flink*;
- Arquiteturas *Lambda* e *Kappa*;

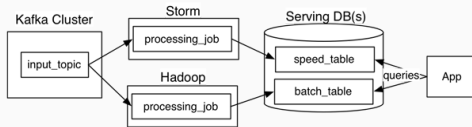


Figura 2: Arquitetura *Lambda* com detalhes práticos (KREPS, 2014).

Fundamentos

```
DataStream<String> lines = env.addSource(  
    new FlinkKafkaConsumer<> (...));  
  
DataStream<Event> events = lines.map((line) -> parse(line));  
  
DataStream<Statistics> stats = events  
    .keyBy("id")  
    .timeWindow(Time.seconds(10))  
    .apply(new MyWindowAggregationFunction());  
  
stats.addSink(new BucketingSink(path));
```

Source

Transformation

Transformation

Sink

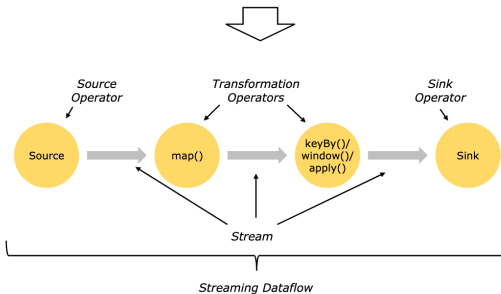


Figura 3: Exemplo de código e *data flow* do Apache Flink (Apache Flink, 2020)

Estado da Arte e Trabalhos Relacionados

Sistemas de detecção de intrusão em redes

- Ferramenta BigFlow (VIEGAS et al., 2019):
 - Capacidade de tratamento de grandes volumes;
 - Atualização semanal com avaliação de um especialista;

Estado da Arte e Trabalhos Relacionados

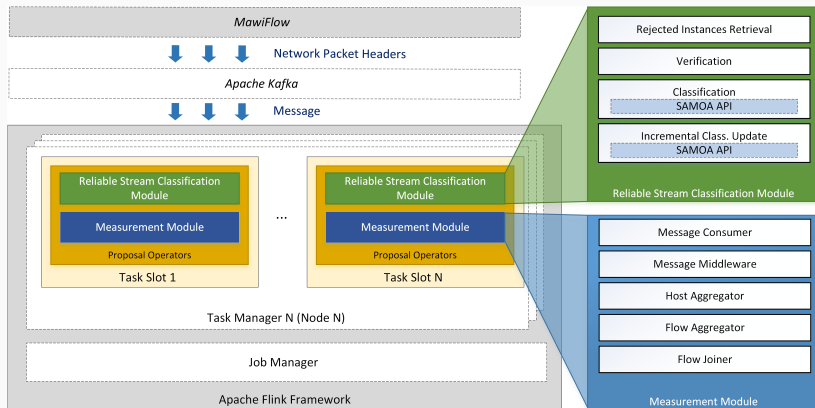


Figura 4: Visão geral da arquitetura e distribuição da ferramenta BigFlow (VIEGAS et al., 2019).

Sistemas de detecção de intrusão em redes

- Ferramenta CATRACA (LOPEZ, 2018):
 - Divisão em camadas alocadas em nuvem e névoa;
 - Modelo de decisão baseado em árvore de decisão;

Estado da Arte e Trabalhos Relacionados

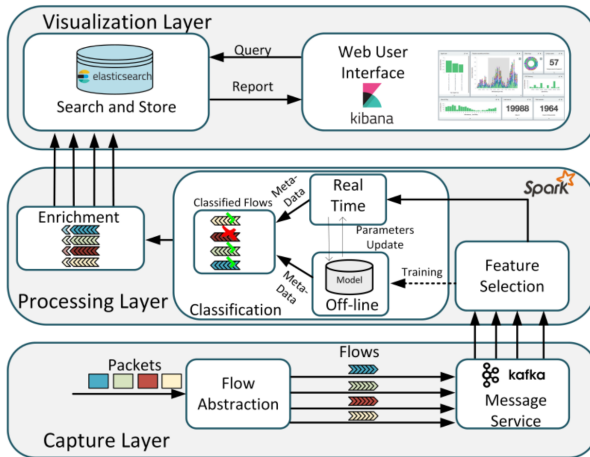


Figura 5: Arquitetura em camadas da ferramenta CATRACA (LOPEZ, 2018).

Sistemas de detecção de intrusão em redes

- Arquitetura IDSA-IoT (CASSALES et al., 2019):
 - Avaliação do algoritmo MINAS, ECSMiner e AnyNovel;
 - Distribuição das tarefas em nuvem e névoa focada em IoT;

Estado da Arte e Trabalhos Relacionados

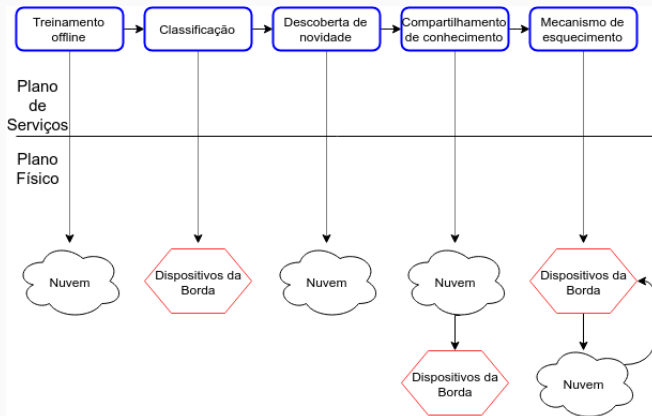


Figura 6: Distribuição de Serviços da Arquitetura IDSA-IoT. Produzida e traduzida por Cassales et al. (2019).

Proposta

Proposta da Pesquisa

- Implementar a distribuição do algoritmo MINAS em nuvem e névoa conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS.

Métodologia

- Plataforma de processamento distribuído;

Proposta da Pesquisa

- Implementar a distribuição do algoritmo MINAS em nuvem e névoa conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS.

Métodologia

- Plataforma de processamento distribuído;
- Estratégias de implementação da arquitetura IDSA-IoT;

Proposta da Pesquisa

- Implementar a distribuição do algoritmo MINAS em nuvem e névoa conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS.

Métodologia

- Plataforma de processamento distribuído;
- Estratégias de implementação da arquitetura IDSA-IoT;
- Experimentação com a distribuição do algoritmo MINAS em ambientes;

Proposta da Pesquisa

- Implementar a distribuição do algoritmo MINAS em nuvem e névoa conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS.

Métodologia

- Plataforma de processamento distribuído;
- Estratégias de implementação da arquitetura IDSA-IoT;
- Experimentação com a distribuição do algoritmo MINAS em ambientes;
- Métricas de qualidade de classificação para validação da implementação;

Proposta da Pesquisa

- Implementar a distribuição do algoritmo MINAS em nuvem e névoa conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS.

Métodologia

- Plataforma de processamento distribuído;
- Estratégias de implementação da arquitetura IDSA-IoT;
- Experimentação com a distribuição do algoritmo MINAS em ambientes;
- Métricas de qualidade de classificação para validação da implementação;
- Métricas de escalabilidade.

O sistema M-FOG é dividido em 5 módulos subdivididos em 2 grupos.

Módulos principais implementam o algoritmo MINAS

- módulo treinamento (*Training Module*);
- módulo classificador (*Classification Module*);
- módulo detector de novidades (*Novelty Detection Module*).

Módulos auxiliares, utilizados para avaliação

- módulo auxiliar *source* (fonte);
- módulo auxiliar *sink* (sorvedouro, consumidor final).

Proposta

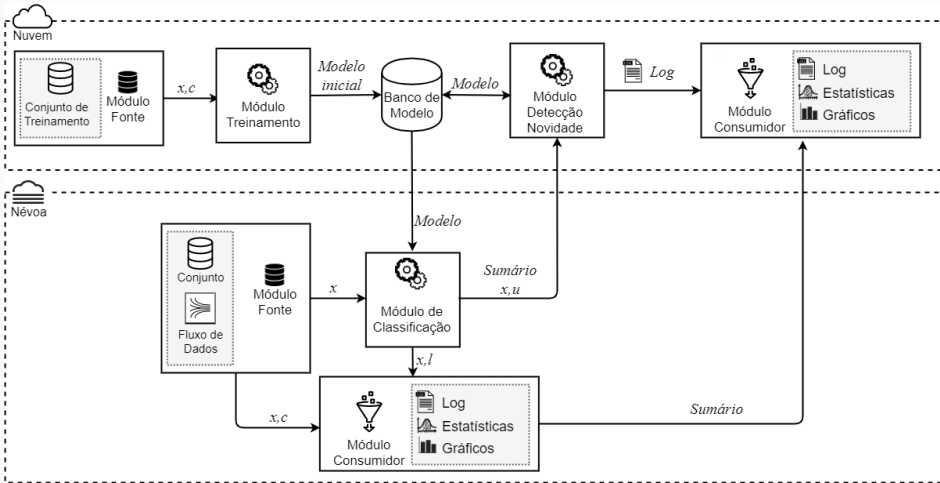


Figura 7: Arquitetura e fluxos de dados do sistema M-FOG.

Resultados Preliminares

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;
- *Apache Kafka* é um sistema de mensagens distribuído;

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;
- *Apache Kafka* é um sistema de mensagens distribuído;
 - Interface de programação com cliente produtor e consumidor;

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;
- *Apache Kafka* é um sistema de mensagens distribuído;
 - Interface de programação com cliente produtor e consumidor;
 - Mensagens organizadas em tópicos que são distribuídos em partições;

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;
- *Apache Kafka* é um sistema de mensagens distribuído;
 - Interface de programação com cliente produtor e consumidor;
 - Mensagens organizadas em tópicos que são distribuídos em partições;
- A hipótese de que a carga seria distribuída entre os consumidores, uma vez que o consumidor pode selecionar uma partição para leitura;

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;
- *Apache Kafka* é um sistema de mensagens distribuído;
 - Interface de programação com cliente produtor e consumidor;
 - Mensagens organizadas em tópicos que são distribuídos em partições;
- A hipótese de que a carga seria distribuída entre os consumidores, uma vez que o consumidor pode selecionar uma partição para leitura;
- Em experimento com um produtor, 8 partições e 8 consumidores, observou-se que um consumidor processava a maior parte das mensagens, poucos consumidores recebiam algumas mensagens e a maioria dos consumidores não recebia mensagem alguma.

Segunda Implementação com Apache Flink

- Implementação escrita em Scala ou Java;

Segunda Implementação com Apache Flink

- Implementação escrita em Scala ou Java;
- Processamento de fluxos *Stateful*;

Segunda Implementação com Apache Flink

- Implementação escrita em Scala ou Java;
- Processamento de fluxos *Stateful*;
- Falta de bibliotecas que distribuam algoritmos base como *K-means*;

Segunda Implementação com Apache Flink

- Implementação escrita em Scala ou Java;
- Processamento de fluxos *Stateful*;
- Falta de bibliotecas que distribuam algoritmos base como *K-means*;
- Sistema *M-FOG* em desenvolvimento, atualmente na fase de validação através das métricas de qualidade de classificação.

Métricas e Ambientes

- Métricas de qualidade de classificação:
 - Avaliação do fluxo de saída do classificador;
 - Uso de uma matriz de confusão ou erro;
 - Taxa de desconhecidos;
 - Macro F-score;

$$\mathbf{E}_n = \begin{pmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,J} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ e_{M,1} & e_{M,2} & \cdots & e_{M,J} \end{pmatrix}$$

$$UnkR_n = \frac{1}{M} \sum_{i=1}^M \frac{\#Unk_i}{\#ExC_i}$$

$$Fscore1_n = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Métricas e Ambientes

- Métricas de escalabilidade:
 - Número e tipo de processadores;
 - Uso de memória;
 - Tempo de processamento;
 - Taxa de eventos;
 - Latência entre a produção e classificação.
- Ambientes de teste:
 - Computador Pessoal (para desenvolvimento);
 - Nuvem UFSCar;
 - Nevoa composta de SBC (*Single Board Computer*) ARM 4 núcleos;


Considerações Finais


Trabalho continua com a finalização da implementação e validação do MFOG com MINAS.


Cronograma


2020				
03	04	05	06	07
Qualificação				
Desenvol.				
Validação				
Experimentos				
Submissão				
			Defesa	


Obrigado!


 Apache Flink. *Apache Flink*. 2020. Disponível em: <https://flink.apache.org/>.


 CASSALES, G. W. et al. IDSA-IoT: An Intrusion Detection System Architecture for IoT Networks. In: *2019 IEEE Symposium on Computers and Communications (ISCC)*. [s.n.], 2019. p. 1–7. ISBN 978-1-7281-2999-0. ISSN 1530-1346. Disponível em: <https://ieeexplore.ieee.org/document/8969609/>.


 COSTA, J. D. *Detecção De Novidade Em Fluxos Contínuos De Dados Multirrótulo*. 127 p. Tese (Master) — UNIVERSIDADE FEDERAL DE SÃO CARLOS, 2019. Disponível em: <https://repositorio.ufscar.br/handle/ufscar/12197>.


 COSTA, J. D. et al. Novelty detection for multi-label stream classification. *Proceedings - 2019 Brazilian Conference on Intelligent Systems, BRACIS 2019*, n. 8, p. 144–149, 2019.


 Da Silva, T. P. et al. A fuzzy multiclass novelty detector for data streams. *IEEE International Conference on Fuzzy Systems*, IEEE, v. 2018-July, p. 1–8, 2018. ISSN 10987584.


 FARIA, E. R. d.; CARVALHO, A. C. Ponce de L. F.; GAMA, J. Minas: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery*, v. 30, n. 3, p. 640–680, May 2015. ISSN 1573-756X. Disponível em: <https://doi.org/10.1007/s10618-015-0433-y>.


 GAMA, J.; RODRIGUES, P. P. *Knowledge Discovery from Data Streams*. [S.l.]: Chapman and Hall/CRC, 2010. ISBN 9781439826119.


 IEEE Communications Society. *IEEE Std 1934-2018: IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*. IEEE, 2018. 176 p. ISBN 9781504450171. Disponível em: <https://ieeexplore.ieee.org/document/8423800>.


 KAMBOURAKIS, G.; KOLIAS, C.; STAVROU, A. The Mirai botnet and the IoT Zombie Armies. In: *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*. IEEE, 2017. v. 2017-Octob, p. 267–272. ISBN 978-1-5386-0595-0. Disponível em: <http://ieeexplore.ieee.org/document/8170867/>.


 KREPS, J. *Questioning the Lambda Architecture* – O'Reilly. 2014. 10 p. Disponível em: <https://www.oreilly.com/radar/questioning-the-lambda-architecture/>.


 LOPEZ, M. E. A. *A monitoring and threat detection system using stream processing as a virtual function for Big Data*. Tese (Theses) — Sorbonne Université ; Universidade federal do Rio de Janeiro, Jun 2018. Disponível em: <https://tel.archives-ouvertes.fr/tel-02111017>.

 MELL, P.; GRANCE, T. The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology. In: NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Public Cloud Computing: Security and Privacy Guidelines*. 2012. p. 97–101. ISBN 9781620819821. Disponível em: <http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>.

 PERNER, P. Concepts for novelty detection and handling based on a case-based reasoning process scheme. In: *Advances in Data Mining. Theoretical Aspects and Applications*. [S.l.]: Springer, 2007. p. 21–33. ISBN 978-3-540-73435-2.

 SHI, W. et al. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, Institute of Electrical and Electronics Engineers Inc., v. 3, n. 5, p. 637–646, oct 2016. ISSN 23274662. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7488250>.

 SILVA, T. P. da. *Abordagem Fuzzy para Detecção de Novidade em Fluxo Contínuo de Dados*. 89 p. Tese (Master) — Universidade Federal de São Carlos, 2018. Disponível em: [〈https://repositorio.ufscar.br/handle/ufscar/10544〉](https://repositorio.ufscar.br/handle/ufscar/10544).

 VIEGAS, E. et al. Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. *Future Generation Computer Systems*, v. 93, p. 473 – 485, 2019. ISSN 0167-739X. Disponível em: [〈http://www.sciencedirect.com/science/article/pii/S0167739X18307635〉](http://www.sciencedirect.com/science/article/pii/S0167739X18307635).

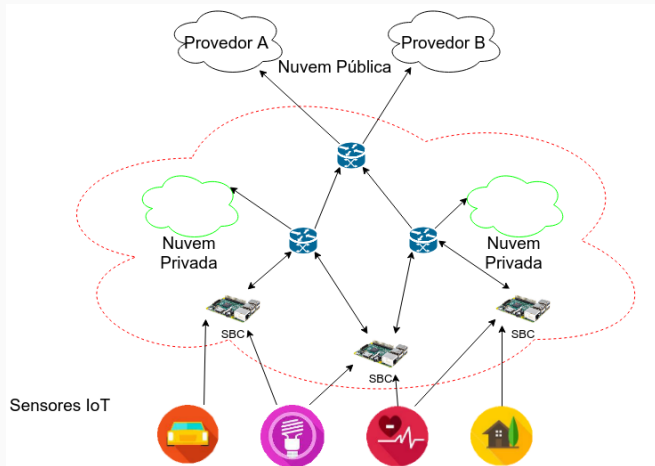


Figura 8: Estrutura Física da Arquitetura IDSA-IoT. Produzida e traduzida por Cassales et al. (2019).