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*

05 Julho 2021

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Computação
Programa de Pós-Graduação em Ciência da Computação

Obrigado CNPq pelo suporte financiero (contrato 167345/2018-4).

Índice

Introdução

Introdução

Contexto

- 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 $\mathrm{Gb/s}$ (KAMBOURAKIS; KOLIAS; STAVROU, 2017).
- Detecção de intrusão em redes:
  - detecção por assinatura *versus* anomalia;
  - ambiente de névoa e redes IoT.

Proposta

- 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 névoa reduzindo a latência sem redução na qualidade de classificação.

Introdução - Cenário

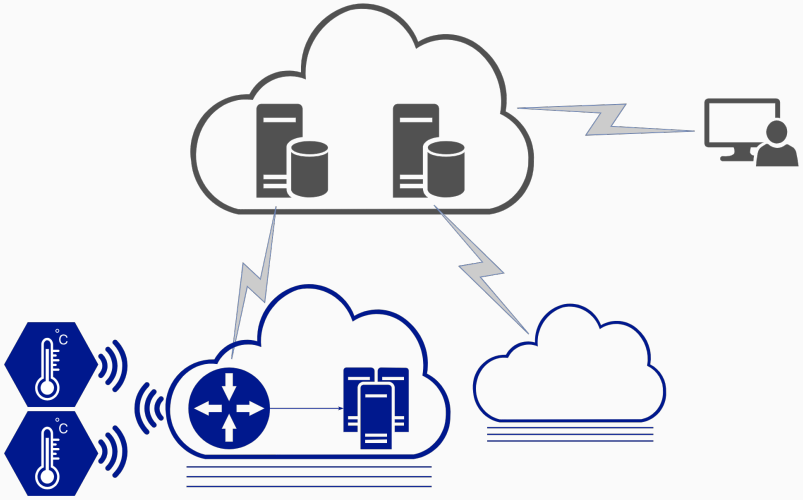


Figura 1: Visão geral de IoT, Névoa e Nuvem.

Fonte: O autor.

Ambientes de computação Distribuída

- **Computação em Nuvem** (*Cloud Computing*) é um modelo que permite acesso conveniente a recursos computacionais compartilhados (MELL; GRANCE, 2012).
  Características: Auto-serviço sob demanda, Amplo acesso à rede, Agrupamento de recursos, Rápida elasticidade, Serviço mensurado;

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

- **Computação em Névoa** (*Fog Computing*) é 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.

Definição de Fluxo de Dados

Um fluxo de dados (*Data Stream*) é uma sequência massiva possivelmente ilimitada de exemplos multi-dimensionais $x_1, x_2, \ldots, x_n, \ldots$ recebidos em instantes associados $t_1, t_2, \ldots, t_n, \ldots$ (AGGARWAL et al., 2003).

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 (GAMA; RODRIGUES, 2010).

- 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.

Estado da Arte e Trabalhos Relacionados

Estado da Arte e Trabalhos Relacionados

Sistemas de detecção de intrusão em redes

- Ferramenta BigFlow (VIEGAS et al., 2019):
  - \+ Integração da extração dos descritores de fluxo à emissão de alarmes;
  - \+ Capacidade de tratamento de grandes volumes;
  - − Atualização semanal com avaliação de um especialista;
  - − Execução somente em nuvem.
- Ferramenta CATRACA (LOPEZ, 2018; SANZ; LOPEZ, 2018):
  - \+ Divisão em camadas alocadas em nuvem e névoa;
  - \+ Modelo de decisão baseado em árvore de decisão;
  - − Extração dos descritores de fluxo é feita em névoa, classificação e detecção é feita em nuvem.
- Arquitetura IDSA-IoT (CASSALES et al., 2019):
  - \+ Avaliação do algoritmo MINAS, ECSMiner e AnyNovel;
  - \+ Distribuição das tarefas em nuvem e névoa focada em IoT;
  - − Implementação e detalhamento da arquitetura em aberto.

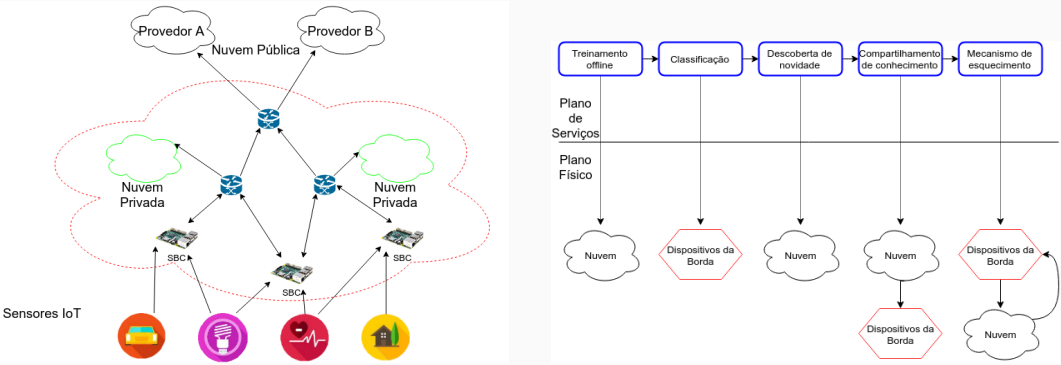


Figura 2: Distribuição de serviços da arquitetura IDSA-IoT.

Fonte: Cassales et al. (2019).

Proposta

Fundamentos - MINAS

- 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;

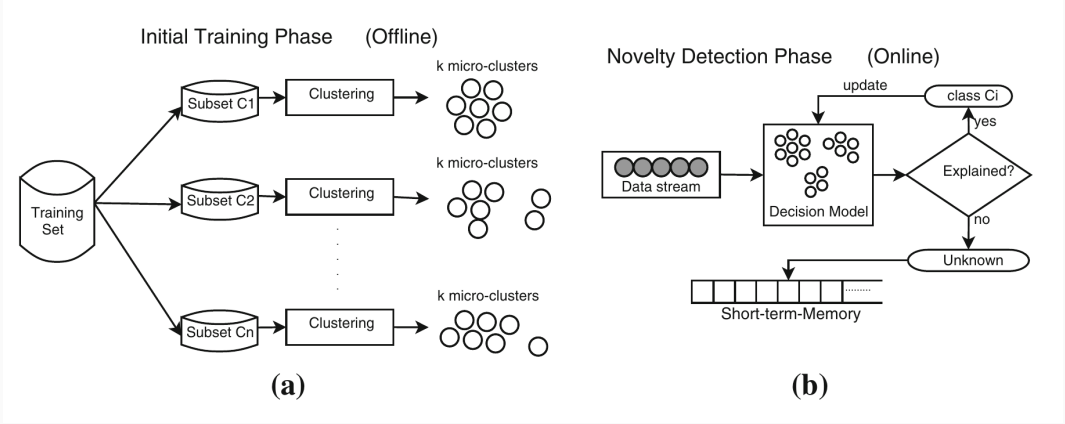


Figura 3: Visão geral do algoritmo MINAS com fases *Offline* (a) e *Online* (b).

Fonte: Faria, Carvalho e Gama (2016).

Fundamentos - MINAS

```
1  Função MinasOnline(Modelo, fluxoEntrada, fluxoSaída, janelaLimpeza, gatilhoDetecçãoNov):
2      Desconhecidos ← ∅; ModeloAntigo ← ∅; últimaLimpeza ← 0; proximaNovidade ← 0;
3      para cada exemplo_i ∈ fluxoEntrada faça
4          maisPróximo ← clusterMaisPróximo (exemplo, Modelo);
5          se maisPróximo.distância < maisPróximo.cluster.raio então
6              exemplo.rótulo ← maisPróximo.cluster.rótulo;
7              maisPróximo.cluster.últimoUso ← i;
8          senão
9              exemplo.rótulo ← "desconhecido";
10             Desconhecidos ← Desconhecidos ∪ exemplo;
11             se | Desconhecidos | ≥ gatilhoDetecçãoNov então
12                 Modelo ← Modelo ∪ DetecçãoNovidade (Modelo ∪ ModeloAntigo, *Desconhecidos);
13             se i > ( últimaLimpeza + janelaLimpeza) então
14                 Modelo ← moveModeloAntigo (Modelo, *ModeloAntigo, últimaLimpeza);
15                 Desconhecidos ← removeExemplosAntigos (Desconhecidos, últimaLimpeza);
16                 últimaLimpeza ← i;
17         fluxoSaída.adicione(exemplo);
```

Algoritmo 1: Interpretação do algoritmo MINAS *online* (FARIA; CARVALHO; GAMA, 2016).

- Arquiteturas *Lambda* e *Kappa*;
- 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*;
- Não especializadas em fluxo de dados:
  - Não-plataforma (construção dos mecanismos de envio e recebimento);
  - Interface de Troca de Mensagens - MPI;

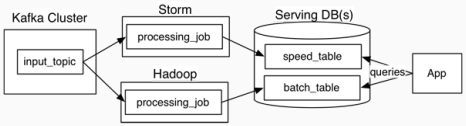


Figura 4: Arquitetura *Lambda* com Kafka, Storm, Hadoop, SGBD tradicional e aplicação consumidora.

Fonte: Kreps (2014).

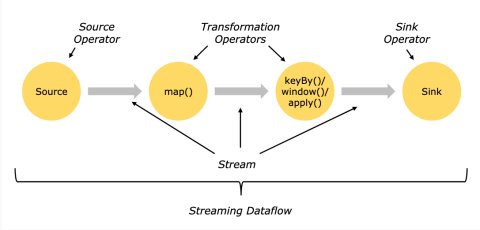


Figura 5: Arquitetura Apache Flink.

Fonte: Apache Flink (2020).

Proposta

Pergunta de Pesquisa

- É viável paralelizar e distribuir o algoritmo MINAS seguindo a arquitetura IDSA-IoT?
- Quais são os efeitos na qualidade de classificação se distribuir o algoritmo MINAS?

Proposta

Proposta da Pesquisa

- Implementar uma versão distribuída do algoritmo MINAS conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS;
- Avaliar a implementação quanto à viabilidade e qualidade.

Método

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

Proposta - Avaliações Preliminares

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;
- Processamento de fluxos *Stateful*;
- Falta de bibliotecas que distribuam algoritmos base como *K-means*;
- Gerenciador de trabalhos (*job manager*) e gerenciador de tarefas (*job manager*) ocupam mais de 1 *GB* em execuções consecutivas, portanto não é confiável para dispositivos pequenos.

Proposta - Implementação MPI

Terceira Implementação com MPI

- Implementado em linguagem C, OpenMPI 4.0.4, seguindo a técnica SPMD;
- Dividido em 2 módulos e 4 tarefas.
- `mpirun` cria processos, o processo de *0* executa o módulo raiz e os demais processos executam o módulo folha;
- Módulo raiz, com as tarefas `Fonte` e `Detector`, trata dos fluxos de entrada e saída além de gerenciar o conjunto de desconhecidos e a detecção de novidade;
- Módulo folha, com as tarefas `Classificador` e `Atualiza Modelo`, trata da classificação de cada exemplo e manutenção do modelo local de cada instância.

Proposta - Implementação MPI

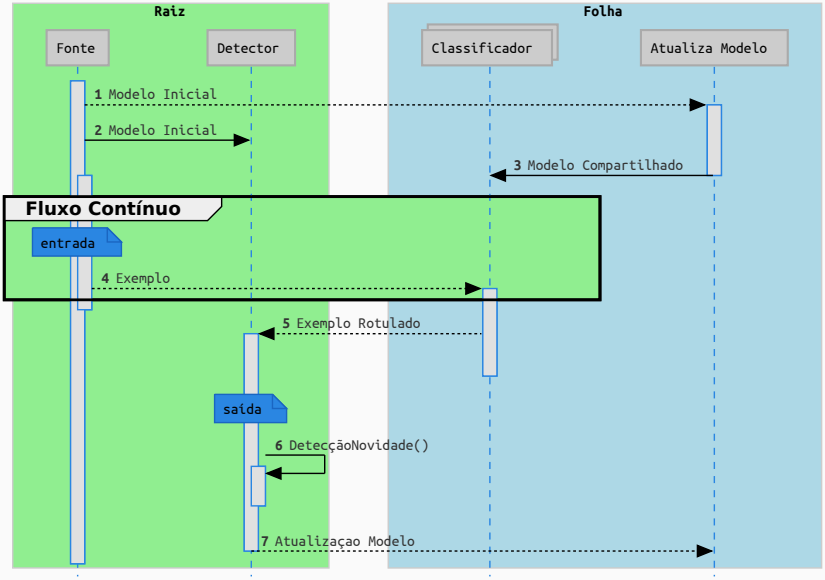


Figura 6: Arquitetura e fluxos de dados do M-FOG.
Fonte: O autor.

Métricas e Ambientes

- Métricas de qualidade de classificação:
  - Avaliação do fluxo de saída do classificador em uma matriz de confusão própria;
  - Taxa de desconhecidos, acurácia e erro por classe.

$$\mathbf{C} = \{c_1, c_2, \cdots, c_m\} \tag{1}$$

$$\mathbf{Y} = \{y_1, y_2, \cdots, y_k\} \tag{2}$$

$$\mathbf{L} = \{l_1, l_2, \cdots, l_n\} = \mathbf{C}' \cup \{\text{“-”}\} \cup \mathbf{Y} \tag{3}$$

$$\mathbf{E}_x = (e_{ij}) \in \mathbb{N}^{m \times n} \tag{4}$$

$$A(l_j) = \begin{cases} \nexists & \text{se } l_j = \text{“-”} \\ c_i & \text{se } \exists c_i = l_j \ : c_i \in \mathbf{C}' \\ c_i & \text{se } e_{ij} = max\{e_{aj} : j \in [0, m]\} \end{cases} \tag{5}$$

$$UnkR_{x,i} = \frac{e_{ij} : l_j = \text{“-”}}{\sum_{j=1}^{n} e_{ij}} \tag{6}$$

$$tp_i = \sum_{j=1}^{n} e_{ij} \ \text{ se } l_j \neq \text{“-”} \text{ e } A(l_j) = c_i \tag{7}$$

$$fn_i = \sum_{j=1}^{n} e_{ij} \ \text{ se } l_j \neq \text{“-”} \text{ e } A(l_j) \neq c_i \tag{8}$$

$$acc_x = \frac{1}{m} \sum_{i=1}^{m} \frac{tp_i}{fn_i + tp_i} \tag{9}$$

$$err_x = \frac{1}{m} \sum_{i=1}^{m} \frac{fn_i}{fn_i + tp_i} \tag{10}$$

Proposta - Método de Avaliação

Métricas e Ambientes

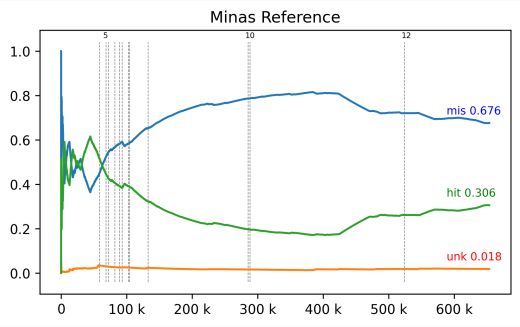
- Métricas de escalabilidade:
  - Número e tipo de processadores;
  - Uso de memória;
  - Tempo de processamento;
  - Latência, tempo entre a entrada e saída de cada descritor de fluxo.
- Ambientes de teste:
  - Computador Pessoal (para desenvolvimento);
  - Nevoa composta de SBC (*Sigle Board Computer*) ARM 4 núcleos;
  - Conjunto de dados para IDS, Kyoto 2006+, segmento dezembro de 2015 como estabelecido por Cassales et al. (2019).

Resultados

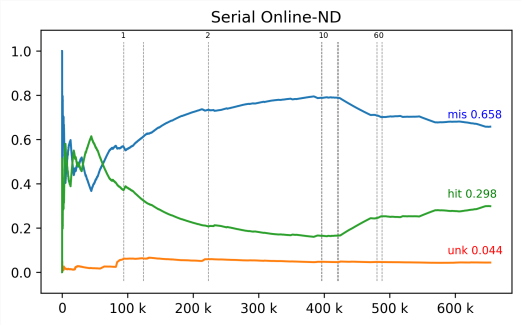
Tabela 1: Listagem dos principais experimentos.

Experimento	Programa	Características
a-Referência	MINAS referência 2013	Raio é a distância máxima.
b-Sequencial	MINAS sequencial para validação	Raio é o desvio padrão das distâncias; Modelo único; Remoção de desconhecidos mais agressivo.
c-Paralelo	M-FOG 1 nó, 4 processadores	Classificadores paralelos; Detecção de novidade assíncrona.
d-Distribuído	M-FOG 3 nós, 12 processadores	Mais processadores; Comunicação em rede.

Resultados - Validação



(a) Experimento *a-Referência*, implementação de referência do algoritmo MINAS.

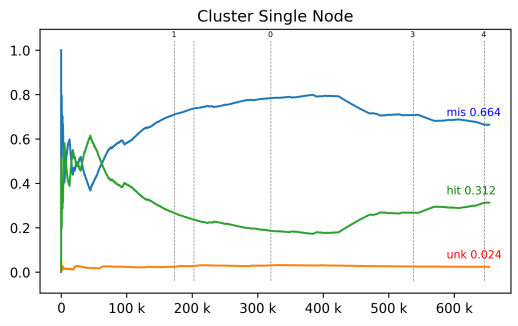


(b) Experimento *b-Sequencial*, M-FOG sequencial.

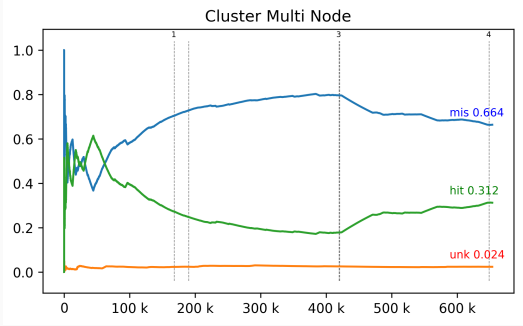
Figura 7: Visualização de fluxo do conjunto *Kyoto* Dez. 2015.

Fonte: O autor.

Resultados - Efeitos Distribuição



(a) Experimento *c-Paralelo*, M-FOG com 1 nó e 4 núcleos.



(b) Experimento *d-Distribuído*, M-FOG com 3 nós de 4 núcleos cada.

Figura 8: Visualização de fluxo do conjunto *Kyoto* Dez. 2015.

Fonte: O autor.

Resultados - Experimentos Principais

Tabela 2: Sumário das métricas extraídas dos experimentos principais.

Experimento Métrica	*a-Referência*	*Offline*	*b-Sequencial*	*c-Paralelo*	*d-Distribuído*
`unk`	0.018333		0.043717	0.023521	0.023718
`hit`	0.305618		0.298438	0.312416	0.312478
`err`	0.676049		0.657843	0.664061	0.663802
Novidades	12		9	5	5
Tempo (s)	2761.83	194.12	80.79	522.10	207.14
Sistema (s)	7.15	0.075	11.51	47.77	157.61
Decorrido (s)	2772.07	194.27	93.03	145.04	95.38
Latência (s)	$4.24 \cdot 10^{-3}$		$1.42 \cdot 10^{-4}$	$2.22 \cdot 10^{-4}$	$1.46 \cdot 10^{-4}$
Processadores	1	1	1	4	12
Speedup				0.6414092	0.9753617
Eficiência				0.1603523	0.0812801

Resultados - Variação Processadores

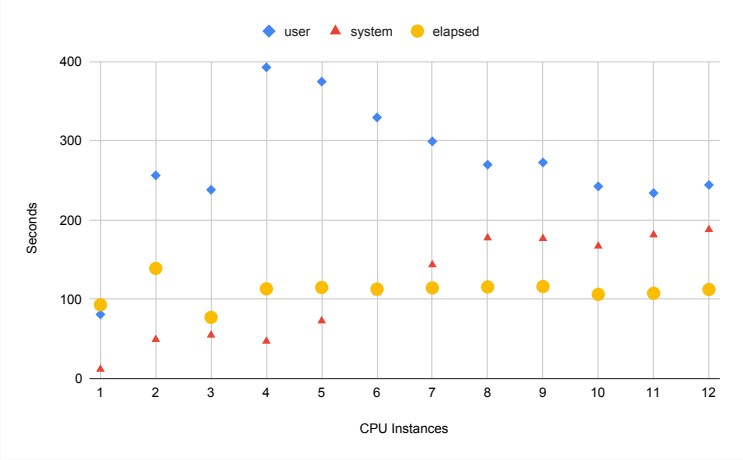
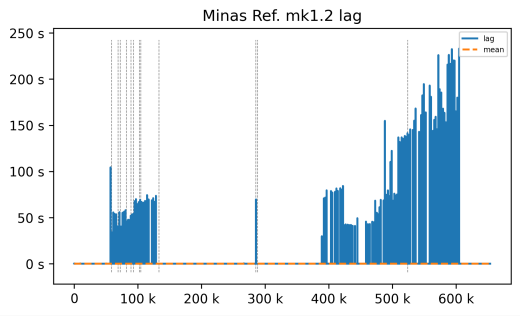


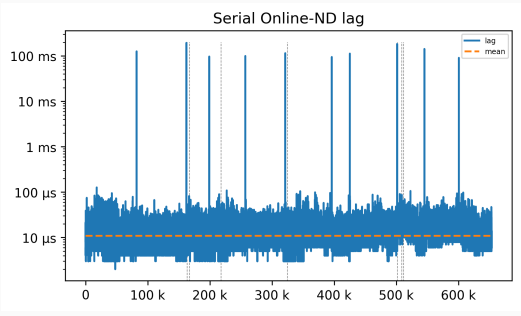
Figura 9: Métricas de tempo para execuções do M-FOG com variação no número de processadores.

Fonte: O autor.

Resultados - Latência (i)



(a) Implementação de referência.

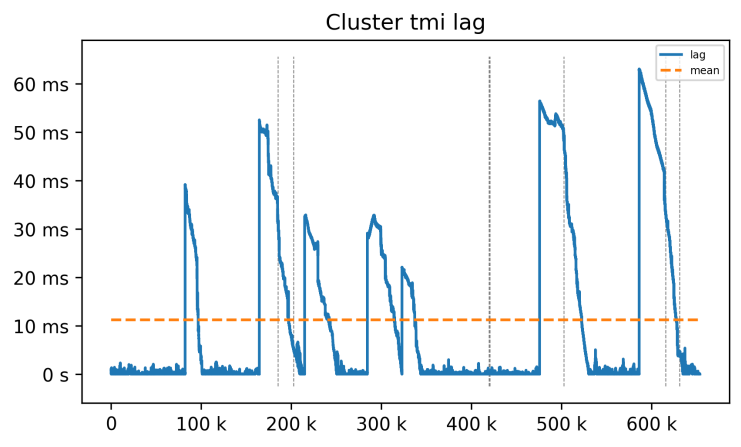


(b) Implementação sequencial.

Figura 10: Visualização de Latência.

Fonte: O autor.

Resultados - Latência (ii)



(c) Implementação paralela.

Figura 10: Visualização de Latência.

Fonte: O autor.

Conclusão

Conclusão

Resultados obtidos:

- Algoritmo MINAS distribuído e a arquitetura IDSA-IoT implementada com modificações;
- Distribuição tem pequeno efeito sobre as métricas de qualidade;
  - Maior efeito é a redução de etiquetas novidade na versão distribuída;
- Resultados mostram que a implementação M-FOG não atinge escala pelo CCR e eficiência;

Trabalhos futuros:

- Da arquitetura: Distribuição do modelo entre redes distintas (conjuntos aditivos);
- Na implementação:
  - Outros algoritmos de agrupamento (CluStream);
  - Estratégia de otimização da distribuição de carga (micro ou mini batching);
  - Outras plataformas de processamento otimizadas para o ambiente névoa;
- No algoritmo:
  - Explorar distribuição espacial dos clusters (polígonos sem sobreposição, árvore de busca);
  - Algoritmo com modelo de tamanho fixo (máxima precisão com recursos disponíveis);

Contribuições e Publicações

- Artigo aceito na trilha principal da 21ª Conferência Internacional em Computação Científica e suas Aplicações (ICCSA 2021, ⟨https://iccsa.org/⟩) em Cagliari, Itália, Setembro 13-16 2021 (PUHL et al., Em via de publicação);
- Código fonte com experimentos e métodos publicamente disponíveis em ⟨https://github.com/luis-puhl/minas-flink⟩.