Pós-Graduação em Ciência da Computação

Thomas Cristanis Cabral Nogueira

Uma avaliação de desempenho de sistemas de processamento de fluxo de dados

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2017

Thomas Cristanis Cabral Nogueira

Uma avaliação de desempenho de sistemas de processamento de fluxo de dados

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia de Software
Orientador(a): Kiev Santos da Gama

Recife
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

N778a Nogueira, Thomas Cristanis Cabral
Uma avaliação de desempenho de sistemas de processamento de fluxo de dados / Thomas Cristanis Cabral Nogueira. – 2017.
71 f.: il., fig., tab.

Orientador: Kiev Santos da Gama.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Engenharia de software. 2. Internet das coisas. I. Gama, Kiev Santos da (orientador). II. Título.

005.1 CDD (23. ed) UFPE- MEI 2019-063

Thomas Cristanis Cabral Nogueira

Uma Avaliação de Desempenho de Sistemas de Processamento de Fluxo de Dados

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação

Aprovado em: 05/09/2017.

BANCA EXAMINADORA

_____

Prof. Dr. Carlos André Guimarães Ferraz
Centro de Informática/UFPE

_____

Prof. Dr. Vanilson André de Arruda Burégio
Departamento de Estatística e Informática / UFRPE

_____

Prof. Dr. Kiev Santos da Gama
Centro de Informática / UFPE
(Orientador)

Decido este trabalho a minha família que foi porto seguro perante as dificuldades durante este percurso.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por toda bondade, amor e providência durante toda a minha vida, mas de forma especial durante este tempo. Um tempo de muito crescimento, onde pude sentir o consolo de Deus por muitas vezes.

Aos meus pais, Severino Mota Nogueira e Raquel Cabral de Lira. A vocês agradeço imensamente, por todo o esforço de uma vida. Pela formação e educação que me proporcionaram ao longo dos anos, de forma especial agradeço ainda mais nesse tempo. Também agradeço por toda paciência exercida durante esse tempo. Ao meu irmão Thales Iury Cabral Nogueira, por todo incentivo e preocupação durante este tempo. Que este trabalho sirva de inspiração para suas atuais e futuras pretensões profissionais.

Aos meus inúmeros familiares, de forma especial agradeço a Tia Suzana, por todo apoio dado nesse tempo, não tenho palavras para expressar meu agradecimento.

Ao meu orientador Kiev Gama, por todo o incansável impulso dado para este trabalho, em todos os instantes. Do começo ao fim, nos momentos calmos e nos mais árduos. Agradeço pela paciência e perseverança, por acreditar em mim nos momentos onde nem eu mesmo estava acreditando.

Aos amigos de convivência intensa Renê Gadelha e Hilário Tomaz, que não mediram esforços para me dar todo apoio desde o primeiro momento de chegada em Recife até a partida. Agradeço pela amizade sempre presente e pelo companheirismo de todas as horas. Da mesma forma agradeço ao amigo Adriano Ferraz, por toda a estima expressada durante os tempos de intensa convivência. Aos que também contribuíram valorosamente com este trabalho expresso a minha gratidão: Tércio Medeiros, Filipe Bezerra, Jocelmo Leite, André Calixto, João Emanoel, Emanoel Carlos, Herbertt Diniz e Edmilson Mota.

Aos tantos colegas e amigos, que não seria possível nomeá-los todos aqui, o meu muito obrigado. A minha gratidão de forma especial, a todos os meus irmãos da Comunidade Nova Berith, por fazerem parte mais uma vez da minha história.

Aos professores que de forma indireta, mas não menos valiosa contribuíram com este trabalho com seus conhecimentos e experiências partilhadas ao longos das disciplinas, agradeço aos professores: Fabio Queda, Renata Maria, Fernando Castor, Nelson Rosa, Patrícia Tedesco.

Ao Centro de Informática, a todos os funcionários, que dos menores aos maiores serviços contribuem substancialmente para que o Cin seja relevante. A Facepe, por promover o fomento a pesquisa, especialmente pelo financiamento deste trabalho. Assim como, a APG, particularmente através seus representantes, por proporcionar um local para o desenvolvimento deste trabalho.

"Por vezes, a pobreza da inteligência humana se manifesta na abundância de palavras."
(AGOSTINHO, 2014)

RESUMO

O crescimento dos dados em uma proporção grandiosa é uma realidade em uma parcela dos softwares atuais, principalmente os que são executados no âmbito da computação distribuída. Esse crescimento verifica-se em diversificados aspectos dos dados, como: no volume, na velocidade e na variedade. Nesse quadro, desponta o conceito de Big Data, que propõe um conjunto de técnicas e soluções para apoiar esse cenário, onde as atuais ferramentas de software não conseguem ter um papel efetivo na coleta, armazenamento, processamento e análise de grande volume de dados. Um desses importantes aspectos e requisito crucial na nossa visão, é a velocidade, que exige respostas rápidas para consultas complexas em fluxos de dados em tempo real. Na literatura, dentre as várias propostas de arquitetura para Big Data, a arquitetura de referência Lambda é uma das que se destacam. Entretanto uma perspectiva simplificada da mesma também teve o nosso interesse, a arquitetura Kappa. Nesse contexto, dado o problema do processamento eficiente de fluxo de dados em tempo real e fundamentado nas arquiteturas Lambda e Kappa, esta dissertação realizou dois experimentos. O primeiro, se deteve em explorar o sistema de processamento de fluxo de dados distribuído, Apache Storm. A partir disso, em comparação, um segundo experimento direcionado ao processamento no cenário na Computação em Névoa foi realizado em dispositivo de borda, com o Apache Edgent. Ambos foram avaliados, tendo sido observadas métricas de desempenho relacionadas a velocidade, precisão e disponibilidade de serviços. Os resultados obtidos através de experimentos apontam a possibilidade de transpor parte do processamento de fluxos de dados para dispositivos na borda da rede.

Palavras-chaves: Processamento de Fluxo de Dados. Internet das Coisas. Big Data. Avaliação de Desempenho.

ABSTRACT

Data growth in a large proportion is a reality in a portion of current software, especially those that are executed within the scope of distributed computing. This growth occurs in diversified aspects of data, such as volume, velocity, and variety. In this context, the Big Data concept emerges. This concept proposes a set of techniques and solutions to support the scenario, in which current software tools can not play a useful role in collection, storage, processing and analysis of large data. Speed, one of these important aspects, is a crucial requirement because there is a need for rapid responses to complex queries in real-time data stream. Among the various architectural proposals for Big Data Lambda reference architecture is one of the highlights, however a simplified version of it has also been of interest to us, the Kappa architecture. Given the problem of efficient processing of data stream in real time and based on the Lambda and Kappa architectures, this dissertation stopped in a first experiment to analyze a system of distributed data stream processing system, the Apache Storm. In comparison, a second experiment directed to the processing of local data stream in the scenario in Fog Computing was performed using Apache Edgent. In both, an evaluation was made, in which performance metrics related to the speed, accuracy, and availability of services were observed. The results obtained through experiments point to the possibility of transposing part of the processing of data stream to devices at the edge of the network.

Key-words: Data Stream Processing. Internet of Things. Big Data. Performance Evaluation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMI	Advanced Metering Infrastructure
BDR	Big Data Repository
CEP	Complex Event Processing
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
IoE	Internet of Everything
IoT	Internet of Things
JSON	JavaScript Object Notation
RAM	Random Access Memory
RDF	Resource Description Framework
RFID	Radio-Frequency Identification
SCA	Sentir Computar Atuar
SPFD	Sistema de Processamento de Fluxo de Dados
SQL	Structured Query Language
TI	Tecnologia da Informação
WSN	Wireless Sensor Networks

SUMÁRIO

1 INTRODUÇÃO

Na era em que vivemos, a tecnologia está nitidamente presente em praticamente todos os lugares, tornando-se algo tão intrínseco ao cotidiano do homem que passou a ser por vezes pouco perceptível. Essa observação foi inicialmente relatada como Computação Ubíqua (WEISER, 1999), que indica que as tecnologias mais profundas serão aquelas que irão desaparecer, que se entrelaçam no tecido da vida cotidiana até que sejam indistinguíveis dela. A evolução dessa concepção sofreu grande contribuição da Computação Distribuída, introduzindo acesso fácil a recursos de informação remota, comunicação com tolerância a falhas, alta disponibilidade e segurança (SAHA; MUKHERJEE, 2003), e da Internet, que de forma gradual alcança quase tudo do cotidiano.

Sendo impulsionado por toda essa perspectiva do surgimento da Computação Ubíqua, aliada à evolução de uma quantidade significativa de tecnologias, o termo IoT (ASHTON, 1999) é definido. Dentre as diversas definições estabelecidas, (PERERA et al., 2014) forneceu uma ampla exposição sobre a temática, definindo que: a IoT possibilita que pessoas e coisas sejam conectadas a qualquer momento, em qualquer lugar, com qualquer coisa e a qualquer pessoa, idealmente usando qualquer caminho/rede e qualquer serviço (PATRICK, 2009). Com isso, uma miríade de sistemas e dispositivos pode ser integrada, lidando com enorme quantidade de dados, que usam padrões e formatos heterogêneos, por vezes sendo capturados em tempo real.

Devido ao grande número de dispositivos implantados, a IoT torna-se uma área emergente e de crescimento contínuo. Para que seja possível ter um breve panorama da dimensão da IoT; é relatado que no ano de 2017 já são aproximadamente 8,4 bilhões de coisas conectadas, sendo que em torno de 63% desse total representa aplicações consumidoras de informação, aproximadamente 5,2 bilhões de unidades (GARTNER, 2017). Além disso, por volta de 2 trilhões de dólares em 2017 serão gastos em dispositivos finais. Esse alto investimento ocorre principalmente nas regiões, como: China, América do Norte e Europa Ocidental.

A consequência do cenário descrito acima, é um enorme volume de dados sendo produzido. Esse atributo de volume distintamente evidencia característica pertinente ao cenário de Big Data, que traz consigo vastos desafios, pois incita as restrições de capacidade dos sistemas atuais. Na literatura, algumas arquiteturas são concebidas com o intuito de estar dedicadas ao gerenciamento de Big Data (FERNANDEZ et al., 2015) (KREPS, 2014). No entanto, uma proposta específica tem ganhado a maior adoção como uma arquitetura de referência no domínio Big Data, a arquitetura Lambda. Esta arquitetura (MARZ, 2013), unifica o processamento em tempo real e em lote em uma única arquitetura, ao mesmo tempo isola as complexidades, permitindo que diferentes requisitos sejam manipulados em camadas distintas.

Retomando o cenário da IoT, é possível também apontar algumas limitações, principalmente sob a perspectiva de suas capacidades, como: processamento e armazenamento (DÍAZ; MARTÍN; RUBIO, 2016). Diante da necessidade de mitigar estas limitações apontadas, a Computação em Nuvem emerge como uma camada capaz de abstrair os problemas denotados. De acordo com (DÍAZ; MARTÍN; RUBIO, 2016), a integração da Computação em Nuvem com IoT, também chamada de Nuvem das Coisas (do inglês, Cloud of Things) (AAZAM et al., 2014), pode resolver o problema com as limitações da IoT. A integração entre esses dois conceitos provê um ambiente promissor, que entrega Tecnologia da Informação (TI) como Serviço (do inglês, IT-as-a-service) para as indústrias e pesquisadores implantarem suas aplicações (KIRAN et al., 2015).

Com o objetivo de ultrapassar os obstáculos presentes em ambientes críticos em IoT e Computação em Nuvem, a Computação em Névoa (do inglês, Fog Computing) é apresentada como uma alternativa adequada para uma série de serviços e aplicações críticas da IoT. A Computação em Névoa é concebida para ter a capacidade de ser virtualizada, fornecendo serviços de computação, armazenamento e rede entre dispositivos finais e centros de dados de Computação em Nuvem. A Computação em Névoa complementa a Computação em Nuvem. Segundo (BONOMI et al., 2012), em vez de eliminar a Computação em Nuvem, a Computação em Névoa permite uma nova geração de aplicativos e serviços, propiciando uma interação frutuosa entre Nuvem e Névoa, especialmente quando se trata de gerenciamento e análise de dados. Esse é um paradigma que desponta como uma extensão do modelo de Computação em Nuvem.

Diante desse complexo cenário, um enorme volume de dados precisa ser processado e altas taxas de transferência são relatadas. Por isso, alternativas para redução do custo de comunicação são extremamente necessárias. Além disso, existem lacunas sobre relatos de avaliação de desempenho de Sistema de Processamento de Fluxo de Dados (SPFD). Os trabalhos encontrados e que são relatados no Capítulo 3 apresentam uma série de lacunas relacionadas a esse aspecto.

Perante o cenário exposto, como também diante de lacunas na literatura atual, esta dissertação apresenta uma avaliação de desempenho, retratada através de uma metodologia de avaliação de desempenho (JAIN, 1991) que extrai resultados referentes à SPFD em tempo real, no cenário de computação distribuída e Computação em Névoa. Foram realizados dois experimentos divididos em: avaliar o Apache Storm, direcionado ao ambiente de *cluster* e Computação em Nuvem; avaliar o Apache Edgent, direcionado à computação em névoa em uma abordagem de processamento em dispositivo de borda, onde os dados são processados em equipamentos de pequeno porte da IoT.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

A motivação para este trabalho, primeiramente, surge a partir da percepção de lacunas inerentes ao processamento de fluxo de dados em tempo real, principalmente ligados

ao cenário da IoT. Evidentemente, esta constatação, dar-se a partir do que vem sendo publicado na literatura.

Uma dessas lacunas está relacionada á escassez de relatos acerca de resultados referentes a métricas de desempenho, como: taxa de transferência, latência e aspectos de uso dos recursos computacionais. Os primeiros trabalhos que incluíram resultados relativos à avaliação de desempenho nesse contexto foram publicados a partir de 2015. Dentre estes trabalhos, parte deles pouco relata sobre os aspectos de avaliação de desempenho, sendo até inconclusivos, conforme poderá ser visto em maiores detalhes no Capítulo 3. Entretanto, uma outra parcela de trabalhos também relatados no mesmo capítulo, apresenta importantes resultados e com isso serviu como base e inspiração para construção desta dissertação. Uma outra constatação, é que na literatura, sistemas de processamento em lote (do inglês, *Batch Processing*), como o Hadoop, já foram bem mais exploradas que os de tempo real, reforçando a viabilidade de exploração dessa área de pesquisa.

1.1.1 Pergunta de Pesquisa

Para orientar a investigação deste trabalho a seguinte pergunta de pesquisa foi definida:

- *Qual é o comportamento, em termos de desempenho, em sistemas de processamento de fluxo de dados em distintos cenários baseados em computação distribuída?*

1.2 OBJETIVOS DO TRABALHO

O objetivo geral desta dissertação consiste em realizar uma avaliação de desempenho, que analise o desempenho no processamento de fluxo de dados, em distintos ambientes do cenário distribuído. Pretendemos que a partir dos resultados obtidos essa avaliação seja capaz de servir como referência para esse domínio, contribuindo com os elementos necessários para a tomada de decisão em relação a aspectos de processamento de fluxo de dados em tempo real.

A partir do objetivo geral, foram definidos os seguintes objetivos específicos:

1. Estabelecer a partir da literatura e descrever uma metodologia de avaliação de desempenho
2. Identificar os sistemas de processamento de fluxo de dados direcionados aos cenários distribuídos
3. Comparar o processamento de fluxo de dados em diferentes cenários.
4. Apontar as características fundamentais de cada um dos cenários a serem avaliados.

1.3 METODOLOGIA

Para que fossem atingidos os objetivos traçados para esta dissertação, foi definido um plano de pesquisa, planejado e executado em etapas. De acordo com (WAZLAWICK, 2014), o uso de métodos científicos de pesquisa é particularmente importante na computação porque, como ciência, ela não pode se ocupar apenas da coleta de dados, a explicação dos dados é muito mais importante. De forma sumarizada os passos da metodologia empregada nesta dissertação foram divididos da seguinte forma:

1. Definição da temática e revisão da literatura: são dois processos que se entrelaçam em uma única etapa, por entendermos que possuem uma forte dependência entre si. Então, primeiramente foi definida a temática a ser explorada, fixada em torno do processamento de fluxo de dados. Por conseguinte, foi iniciada uma revisão de literatura sobre essa temática, como também das demais envolvidas: IoT, Big Data, Computação em Nuvem, Computação em Névoa. Com base no conhecimento mais aprofundado extraído nessa fase, foi possível começar a identificar o que já existe na literatura em termos de pesquisa e as oportunidades a serem exploradas.

2. Definição do experimento: nessa fase foram identificados os aspectos, métricas e demais critérios relevantes a serem observados. Também nessa etapa foram foi definida a carga de trabalho usada na execução dos experimentos.

3. Planejamento da avaliação de desempenho: diversos trabalhos na literatura[1] retratam as etapas do processo de avaliação de desempenho que foi adotada para este trabalho (JAIN, 1991). Respeitar esse tipo de metodologia evita que erros sejam cometidos durante o planejamento e execução de experimentos. Por isso, foi necessário definirmos bem o objetivo pretendido, não buscando qualquer tipo de viés que alterasse os resultados. A escolha das métricas, parâmetros e fatores não foi arbitrária, houve um cuidado na relevância dos mesmos, assim como uma análise que levou em consideração os trabalhos referentes à avaliação de desempenho existentes na literatura. No Capítulo 4 serão abordados todos os aspectos referentes a essa metodologia.

4. Execução dos experimentos: após o planejamento dos experimentos, foram realizadas as execuções. Dois cenários foram empregados na execução dos experimentos, o primeiro a partir de um cenário distribuído, em que um *cluster* composto por algumas máquinas foi usado. Nesse passo, englobamos todo o processo, desde a geração dos dados até o processamento, foco de toda a pesquisa. De forma semelhante, o experimento também foi executado, mas em cenário de processamento de fluxo de dados em dispositivo de borda.

[1] https://goo.gl/ZJt4re

5. Análise dos resultados obtidos: nessa última etapa, foram apresentados os resultados atingidos a partir dos experimentos, nos quais os cenários baseados em computação distribuída foram confrontados.

1.4 ORGANIZAÇÃO DO TEXTO

Este trabalho de dissertação encontra-se organizado em cinco capítulos. No Capítulo 2 são apresentados os conceitos principais sobre a temática abordada na dissertação. São expostos conceitualmente os cenários da Computação Distribuída, Computação em Nuvem e Computação em Névoa. São identificadas as definições mais relevantes sobre IoT, assim como suas características e principais desafios envolvidos. De forma semelhante seguimos essa mesma trilha para descrever Big Data. Dentro desse contexto, detalhamos a arquitetura Lambda, direcionada ao processamento de grande volume de dados, especificando sua camada de processamento em tempo real e descrevemos o SPFD Apache Storm. O Capítulo 3 é relativo aos trabalhos que abrangem o estado da arte, compreendendo desde propostas e avaliação de desempenho de arquiteturas para cenário de processamento de fluxo de dados até trabalhos que realizaram avaliações de desempenho de SPFDs em ambiente distribuído, sem qualquer relato de arquitetura empregada. O Capítulo 4 discorre sobre a metodologia utilizada para avaliação de desempenho, a descrição do experimento e os resultados obtidos. Por fim, no Capítulo 5, as conclusões encontradas com base nos experimento realizados são apresentadas. Além disso, são apontadas as contribuições, os trabalhos futuros que podem melhorar a avaliação realizada.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo, apresenta os principais conceitos sobre a temática abordada nesta dissertação. São apresentadas as principais definições sobre IoT, assim como suas mais importantes características. Posteriormente, o cenário de Big Data é apresentado e algumas arquiteturas voltadas para gerenciamento desse tipo de ambiente são descritas. Por fim, uma descrição sobre os SPFDs Apache Storm e Apache Edgent é realizada.

2.1 INTERNET DAS COISAS

De forma rápida a IoT desponta em diversas linhas de pesquisa e assim como conceitualmente abrange um miríade de dispositivos, também compreende uma série de tecnologias, propostas e modelos. A IoT parece refletir a evolução de uma ideia proposta no período em que esse termo foi cunhado, a computação Ubíqua ou onipresente (WEISER, 1999), que propõe a detecção do ambiente e dos objetos em torno de nós. Diversas são as definições sobre IoT elucidadas na literatura, algumas delas serão a seguir apresentadas.

2.1.1 Definições

A IoT também é nomeada como Internet of Everything (IoE), já demonstra em sua terminologia que ambiciona compreender toda a Internet. Esse conceito emerge em um primeiro momento a partir de uma perspectiva da industria, relacionado há uma nova concepção de RFID aplicada na cadeia de suprimentos de uma grande manufatura do setor produtivo (KALMESHWAR; PRASAD, 2017). O primeiro trabalho a expressar esse termo (ASHTON, 1999), afirma que os computadores são quase totalmente dependentes dos humanos para obter informação, entretanto, os humanos não fazem parte do diagrama da internet, mesmo sendo os mais importantes roteadores de informação, pois possuem tempo, atenção e precisão limitadas para capturar dados.

De acordo com (ATZORI; IERA; MORABITO, 2010) que propõe uma das definições mais relatadas na literatura, a IoT essencialmente é a presença generalizada ao nosso redor de uma variedade de coisas ou objetos, como as tags de Radio-Frequency Identification (RFID), sensores, atuadores, telefones celular. Com isso, é possível que esses dispositivos possam interagir entre si formando uma grande rede de máquinas e dispositivos com capacidade de interação (LEE; LEE, 2015). Outra definição que também se apoia nesses dois pilares (coisas/objeto e rede), é concebida em (REED; LARUS; GANNON, 2012), que define a IoT como uma rede crescente e amplamente invisível de objetos inteligentes interligados que prometem transformar a maneira como interagimos com as coisas cotidianas.

O termo "coisa", está intrinsecamente ligado a circunstância da IoT ser mais do que uma rede de objetos. Por isso, (DÍAZ; MARTÍN; RUBIO, 2016) definem IoT como um con-

junto de coisas interligadas pela Internet, essas coisas são: os humanos, as tags e os sensores. Ressaltando que o humano faz parte, mas a tendencia é que sejam a menor parte na geração e consumo de dados.

É possível dizer que IoT é uma grande solução para todos os dispositivos inteligentes que se conectam à Internet, monitorando e controlando tudo (KALMESHWAR; PRASAD, 2017), usando toda sua capacidade com o propósito de medir, comunicar e agir. Essas ações são o plano de fundo e a ideia chave da IoT, que é obter informações sobre o nosso meio ambiente para entender, controlar e atuar sobre isso (DÍAZ; MARTÍN; RUBIO, 2016).

Uma perspectiva bastante ampla é apresentada em (MIORANDI et al., 2012), que decompôs o conceito em três partes principais: uma rede global que interconecta os objetos inteligentes, o conjunto de tecnologias que suportam a IoT e por fim o conjunto de aplicações e serviços que a alavancam.

2.1.2 Características

A IoT possui um série de características claramente perceptíveis, algumas delas foram recebidas da própria Internet, já que esse é um conceito totalmente alicerçado nesse pilar. De acordo com o estudo realizado em (PATEL et al., 2011), a IoT herda comportamentos comuns a Internet, como: acesso remoto a serviços, interações através de software e acesso a banco de dados. Além dessas características descritas, outras fazem parte da IoT segundo o mesmo autor, como: detecção intermitente, coleta de dados regulares e controles do tipo Sentir Computar Atuar (SCA).

A IoT é composta por uma grande quantidade de sensores, podendo chegar a bilhões. Diante desse número elevado de sensores e seguindo as características já anteriormente citadas, (ZASLAVSKY; PERERA; GEORGAKOPOULOS, 2013) também relatam a necessidade dessas mesmas capacidades: sentir, comunicar, calcular e se necessário atuar. Esses sensores normalmente possuem características de baixa potência, baixa memória e limitações de bateria e rede (DÍAZ; MARTÍN; RUBIO, 2016). Entretanto, nem sempre IoT significa apenas esse tipo de dispositivo, estamos falando de um grande ecossistemas de tecnologias, padrões, protocolos, plataformas e diversificados dispositivos podem fazer parte desse ambiente, desde o menor dos nós finais, há um gateway de alto desempenho ou plataforma de nuvem (KALMESHWAR; PRASAD, 2017). Pelo fato desse paradigma ser orientado para dispositivos inteligentes a capacidade de autoconfiguração é crucial, além disso IoT é caracterizada por uma rede amplamente distribuída, com capacidade limitada de armazenamento e processamento, que envolvem preocupações quanto à confiabilidade, desempenho, segurança e privacidade (BOTTA et al., 2016). A seguir na Tabela 2 apresentamos de forma resumida as principais características da IoT apontadas na literatura, salientando que, essas características se aplicam a dispositivos de pequeno porte, normalmente encontrados em ambientes de IoT. A IoT não se restringe apenas a essas características, como bem expomos nesta mesma seção.

Tabela 2 – Síntese das características de dispositivos de pequeno porte em ambientes da IoT

	Internet	Comportamental	Física*
Características	Acesso remoto	Detecção	Baixa Potência
	Interação	Coleta	Baixa Memória
	Banco de Dados	SCA	Limitação Bateria
	-	Autoconfiguração	Limitação Rede
	-	Confiabilidade	L. Armazenamento
	-	Desempenho	L. Processamento
	-	Segurança	-
	-	Privacidade	-

2.1.3 Tecnologias Envolvidas

A evolução da IoT ocorreu obviamente devido há uma série de fatores, que vão desde os altos investimentos, o interesse e empenho da academia em pesquisa, mas também em razão da junção de diversas tecnologias. Essas tecnologias vão desde a comunicação sem fio, a Internet, sistemas embarcados e micro eletrônica, sistemas de controle de rede e de sensores sem fio entre outras tecnologias que contribuem para permitir a IoT aconteça (KALMESHWAR; PRASAD, 2017). De acordo com (LEE; LEE, 2015), são cinco as tecnologias amplamente utilizadas para a implantação de produtos e serviços bem sucedidos baseados em IoT: RFID, Wireless Sensor Networks (WSN), *Middleware*, Aplicações de IoT e Computação em Nuvem. Esta última, provê soluções que oferecem suporte a cenários onde grandes fluxo de dados devem ser processados de forma eficiente.

2.2 BIG DATA

O conceito de Big Data foi originalmente cunhado na área de visualização científica (COX; ELLSWORTH, 1997) (BRYSON et al., 1999), para descrever cenários de grande volume de dados. É um campo de pesquisa emergente, por isso ainda existe dificuldade em obter uma definição amplamente aceita. Em (ZASLAVSKY; PERERA; GEORGAKOPOULOS, 2013) Big Data é definido como uma grande quantidade de dados, no quais os atributos desafiam as restrições de capacidade de sistemas ou necessidade de negócios. De forma semelhante (LIU et al., 2015) definem como o conjunto de dados cujo o tamanho é além da capacidade que as típicas ferramentas de banco de dados podem capturar, armazenar, gerenciar e analisar. Uma definição mais sucinta é apresentada em (IBM; ZIKOPOULOS; EATON, 2011), que definem que o termo Big Data aplica-se a informações que não podem ser processadas ou analisadas usando processos ou ferramentas tradicionais. Ainda em (IBM; ZIKOPOULOS; EATON, 2011), uma outra contribuição que caracteriza o cenário de Big Data é descrita,

são os três Vs: volume, variedade e velocidade, conforme pode ser visto na Figura 1.

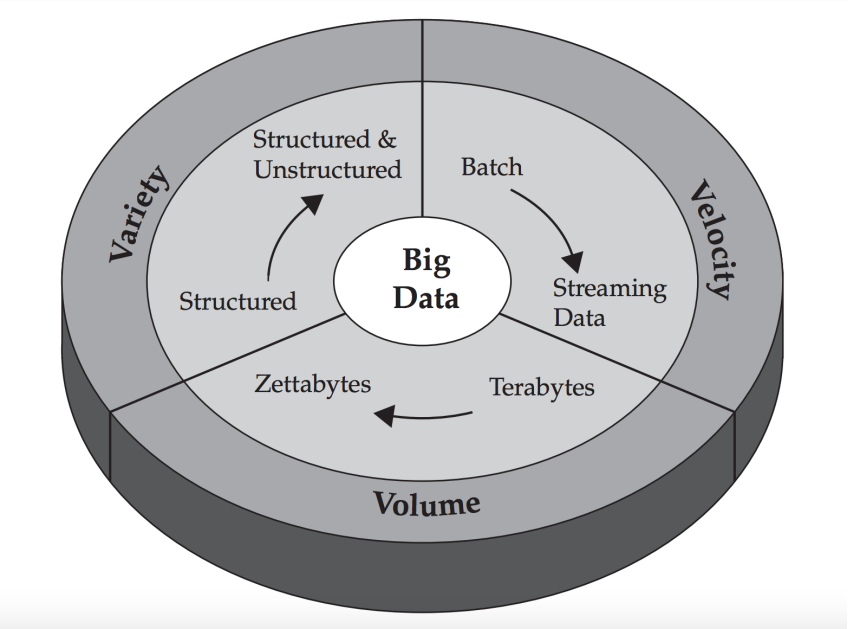

Figura 1 – Os três Vs do cenário de Big Data

Fonte: (IBM; ZIKOPOULOS; EATON, 2011)

2.3 PROCESSAMENTO DE DADOS EM TEMPO REAL NA IOT

Nesta seção, descreveremos a arquitetura Lambda, uma arquitetura de referência que tem como finalidade realizar o processamento de fluxo de dados em lote e em tempo real, uma solução completa voltada para gerenciamento de Big Data (MARZ, 2013).

2.3.1 Arquitetura Lambda

A construção de qualquer plataforma que aborde todos os cenários de IoT em tempo real é desafiadora (CHENG et al., 2015), no entanto a arquitetura Lambda se adapta aos requisitos de ambientes inteligentes em IoT (VILLARI et al., 2014). Essa proposta separa as complexidades de processamento de fluxo de dados em camadas distintas, onde cada camada satisfaz um subconjunto de propriedades e executa suas atribuições de forma coordenada (MARTÍNEZ-PRIETO et al., 2015). O aspecto da separação de complexidade é bastante oportuno, pois otimiza o custo de processamento, devido a compreensão que parte dos dados usa o processamento em lote e outra parte em tempo real. Por outro lado há também ponderações quanto ao custo com a manutenção de múltiplos projetos com propósitos diferentes (KIRAN et al., 2015). Em um ambiente onde um enorme conjunto de dados precisa de processamento e posteriormente armazenamento, torna-se inviável à cada consulta realizada fazer uma varredura em todos os dados. Por isso, essa abordagem propõe a geração de visões (do inglês, *views*) pre computadas e indexadas, para que

possam ser acessadas futuramente, chamada de visões em lote (do inglês, *batch view*), como pode ser visto na Figura 2. Com isso, não é necessário percorrer todos os dados durante um consulta, mas apenas nas visões em lote. De acordo com (MARZ, 2013), essa proposta pode ser formulada da seguinte maneira:

```
visão de lote = função(todos os dados)
consulta = funcão(visão de lote)
```

Aprofundando em uma visão geral sobre a arquitetura, é possível estabelecer que os dados são divididos de duas formas: dados históricos que contém todos os dados recebidos (processados na camada de lote) e o fluxo de dados recebido recentemente (processado na camada de velocidade)(PREUVENEERS; BERBERS; JOOSEN, 2016).

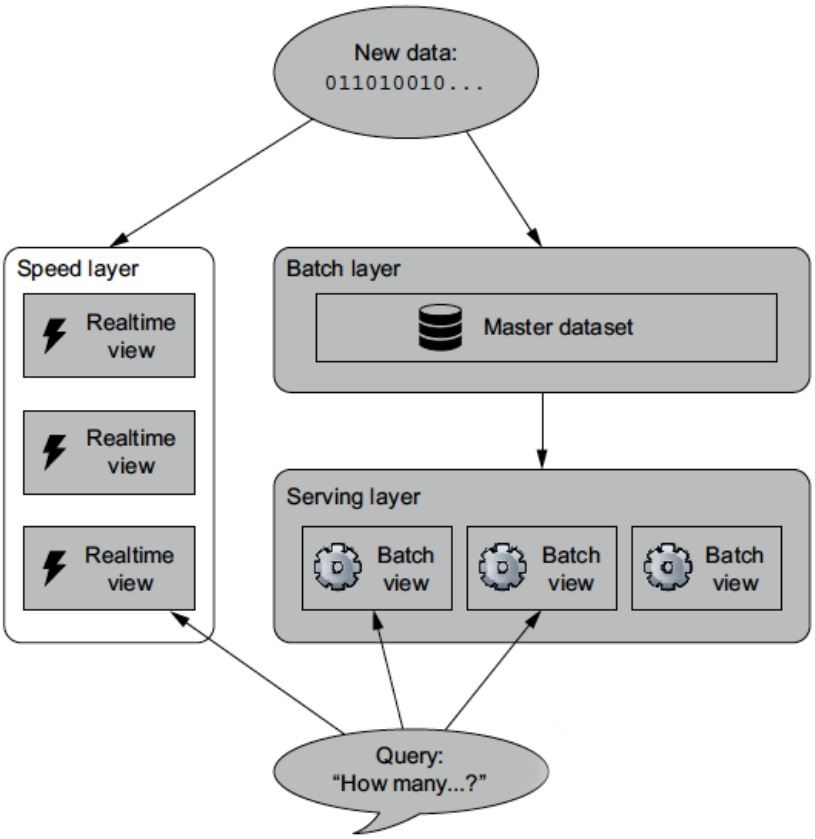

Figura 2 – Modelo proposto na Arquitetura Lambda

(MARZ, 2013)

Serão detalhadas nos seguintes itens abaixo as camadas que compõe essa arquitetura:

1. Batch Layer: É responsável por armazenar os dados, de forma imutável, dados que crescem constantemente e calcular funções arbitrárias [1] neste conjunto de dados. A camada é executada em um loop contínuo, recalculando continuamente as visualizações a partir do zero, gerando visualizações pré-computadas que são usadas para

[1] No contexto de Big Data essa definição remete a qualquer tipo de função

responder a consultas específicas. Dentro da batch layer a partir do processamento de todos os dados é possível encontrar padrões de comportamento. Além disso, esta camada quando comparada à speed layer tem maior latência, pois executa o processamento mais intenso (DÍAZ; MARTÍN; RUBIO, 2016). O Apache Spark e o Hadoop são fortemente usados nessa camada.

2. Serving Layer: Esta camada, em conjunto com a batch layer, é responsável pela maior parte do gerenciamento de Big Data (DÍAZ; MARTÍN; RUBIO, 2016), na qual a camada de lote emite as visualizações e a Serving Layer armazena possibilitando fazer leituras aleatórias. A batch layer serve como um banco de dados especializado e distribuído que carrega visões de lote(MARZ, 2013). Em (DÍAZ; MARTÍN; RUBIO, 2016) esta camada é referida como sendo a principal da arquitetura Lambda, uma vez que é responsável por servir e mesclar a visualização em lote e em tempo real.

3. Speed Layer: Os novos dados são armazenados na batch layer e também são enviados para a speed layer, como pode ser visto na parte mais acima da Figura 2. A camada tem a finalidade de produzir visões em tempo real com base nesses dados recebidos. As visões geradas nesta camada são combinadas com aquelas produzidas na serving layer para atender a consultas aleatórias. Diferente da batch layer, são observados apenas os dados mais recentes e de forma particionada, obtendo latências mais baixas. A latência é um ponto que deve ter maior atenção já que essa camada deve compensar as atualizações de alta latência da serving layer. Em resumo, as duas principais facetas da speed layer são: armazenar as visões em tempo real e processar o fluxo de dados recebido para atualizar essas visões (MARZ, 2013). Como já dito anteriormente, é importante notar que esta camada é o foco desta pesquisa.

2.3.2 Arquitetura Kappa

Na literatura, além da arquitetura Lambda outras proposta são citadas, com a função de serem dedicadas ao gerenciamento de grandes dados (do inglês, *Big Data Management*), como: Liquid (FERNANDEZ et al., 2015) e Kappa (KREPS, 2014). A arquitetura Kappa será descrita, pois esta de acordo com o foco da nossa pesquisa, o processamento e fluxo de dados em tempo real. Essa arquitetura é uma simplificação de Lambda, o conceito principal está em torno da ideia de deixar de lado a camada de processamento em lote (do inglês, Batch Layer) usando apenas um sistema de processamento de fluxo de dados (ZSCHöRNIG; WEHLITZ; FRANCZYK, 2017). Como é possível visualizar na Figura 3, a concepção sugere usar um mecanismo de fila de mensagem, por exemplo o Apache Kafka, para manter todos os registros dos dados e quando for necessário o reprocessamento, uma nova instância no SPFD inicia o processamento dos dados desde o inicio. Isso traz algumas vantagens, evita a duplicidade de armazenamento, traz maior flexibilidade, mas por outro lado aumenta ao volume de dados retidos no Apache Kafka, exigindo um maior poder

computacional (ZSCHöRNIG; WEHLITZ; FRANCZYK, 2017). Na literatura ainda são poucos os relatos sobre arquitetura Kappa.

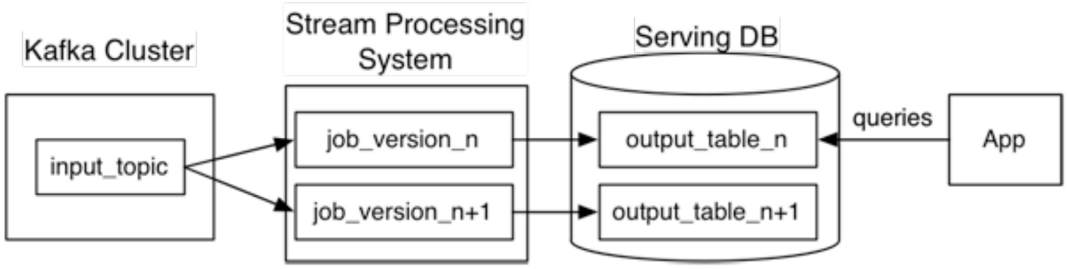

Figura 3 – Visão geral da arquitetura Kappa

Fonte: (KREPS, 2014)

2.3.3 Apache Storm

Os requisitos exigidos na cenário de processamento de grandes volumes de dados, levaram ao desenvolvimento de mecanismos capazes de processar o fluxo de dados contínuo de acordo com certas regras predefinidas, sendo processados dados em tempo real ou quase em tempo real. De acordo com (CUGOLA; MARGARA, 2012), a esta proposta é dado o nome de processamento de fluxo de dados (do inglês, *Data Stream Processing*).

Existem diversas plataformas com o propósito de realizar este processamento, mas o Apache Storm foi escolhido para este trabalho, porque possui características fundamentais para o cenário que iremos avaliar. O Apache Storm é escalável, resiliente, extensível, eficiente e gerenciável (TOSHNIWAL et al., 2014), também possui como característica a tolerância a falhas, que fornece garantias sobre os dados processados. Este recurso assegura que cada dado seja processado pelo menos uma vez e que seja transformado uma vez ou descartado em caso de falha. Além disso, conforme poderá ser visto no Capítulo 3 o Apache Storm alcançou melhores resultados em alguns cenários quando comparado a outros SPFDs, principalmente em relação a latência.

O Apache Storm é um mecanismo que atua diretamente para executar o processamento de fluxo de dados em tempo real. A execução no Apache Storm é baseada em grafo direcionado (DÍAZ; MARTÍN; RUBIO, 2016), chamado de topologia. Os dados de entrada a serem processados são a principal estrutura de dados do Apache Storm, esta estrutura é uma lista de elementos chamado de tupla, como pode ser visto no início da Figura 4. Os componentes essenciais do Apache Storm são: a topologia, o *spout* e o *bolt*. O *spout* é destinado a receber os dados consumidos de uma fonte de dados qualquer, normalmente

de filas de mensagens distribuídas (por exemplo, Apache Kafka), e os *bolts* executam o processamento de tupla.

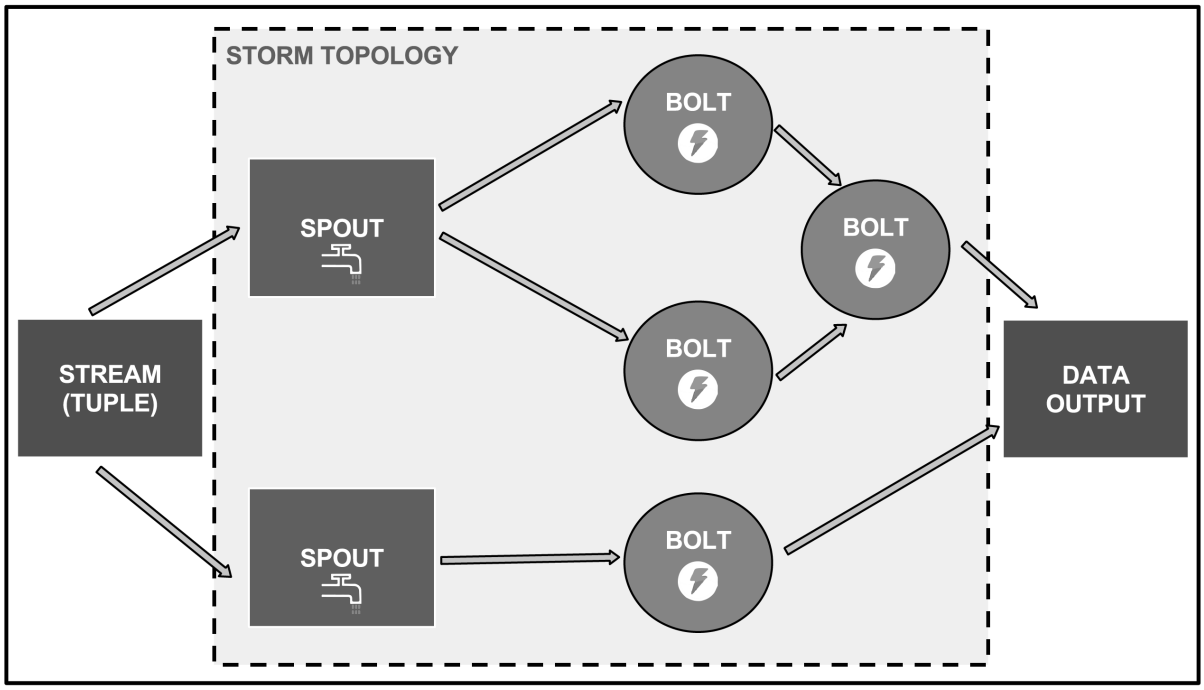

Figura 4 – Representação de uma topologia do Apache Storm

Fonte: Próprio Autor

O Apache Storm sob a perspectiva de execução em um *cluster* distribuído, possui alguns outros componentes fundamentais para sua operação, conforme pode ser visto na Figura 5. São dois os tipos de nós em um ambiente de *cluster*: mestre (do inglês, *master*) e trabalhador (do inglês, *worker*). O nó *master* executa um processo de plano de fundo (do inglês, *daemon*), chamado *nimbus*, que é responsável por distribuir tarefas aos nós *workers*. O *supervisor* recebe atribuições do *nimbus*, gera os processos *worker*, monitoriza a saúde (do inglês, health check) dos processos e assume as medidas necessárias. Conforme mostrado na Figura 5, um único *supervisor* pode ter mais de um processo *worker*, e cada um é mapeado para uma única topologia. Já a topologia pode ser atribuída a mais de um processo *worker*, onde as diferentes partes serão executadas por vários processos *worker*. Cada processo *worker* executa um ou mais *executors* e estes são compostos por uma ou mais *tasks* que executam o trabalho de *bolts* e *spout*, que de fato realizam a leitura e processamento dos dados.

Nesse cenário, todas as informações de coordenação e controle de estado entre *nimbus* e *supervisors* são realizadas por um componente chamado *zookeeper* [2]. O *zookeeper* é um serviço centralizado e de baixa carga que mantém informações de configuração em sistemas distribuídos. Além disso, para que o processamento aconteça, a troca de mensagens entre os componentes descritos é crucial. Essa comunicação pode ser dividida em dois tipos:

[2] https://zookeeper.apache.org

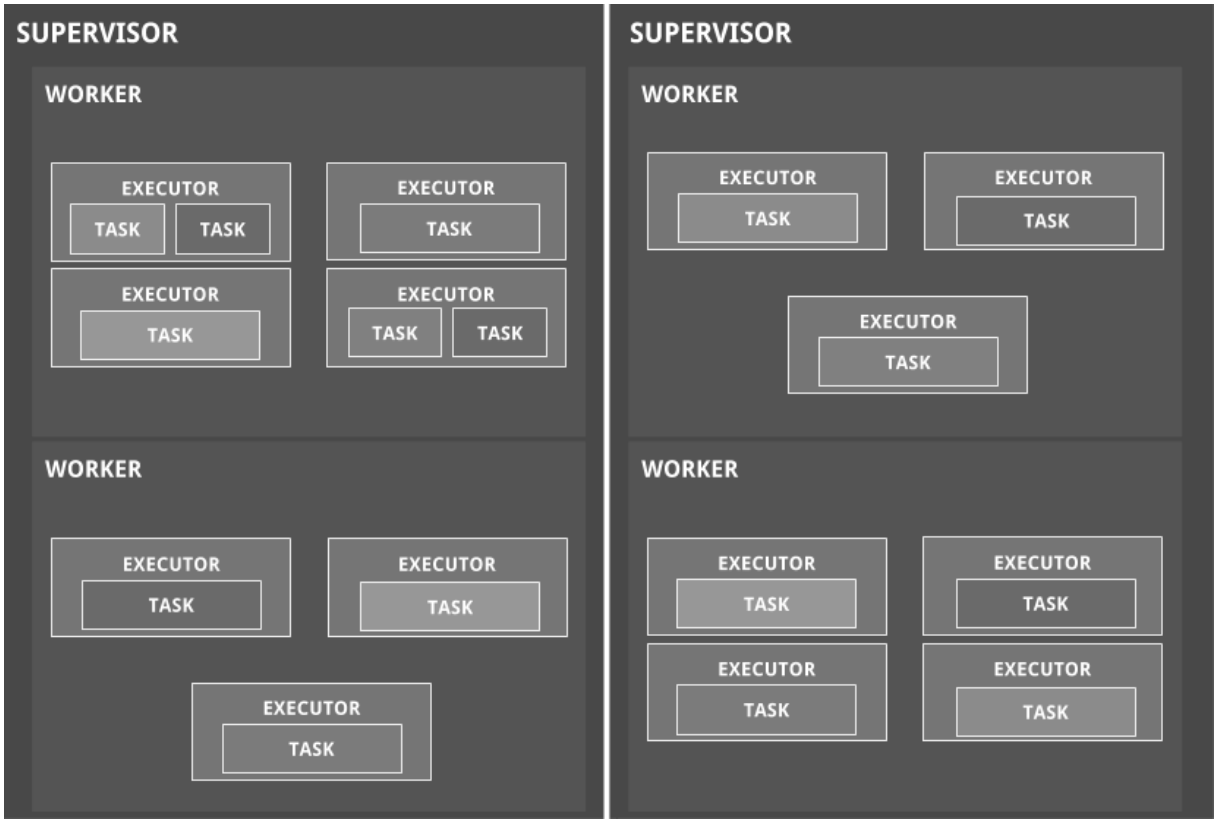

Figura 5 – Componentes Internos do Apache Storm

Fonte: Próprio Autor

intra-worker e inter-worker. A troca de mensagens intra-worker, que acontece dentro do processo do *worker*, na mesma máquina, é uma troca de mensagens entre *threads*, então o *LMAX Disruptor* [3], uma biblioteca de mensagens inter-thread é usada. A comunicação entre workers (inter-worker) ocorre entre processos de *worker* que estão em diferentes máquinas, através da rede. Então, essa comunicação é realizada por *ZeroMQ* [4] e Netty [5]. Por fim, destacamos que além do Apache Storm outras iniciativas de código aberto tem sido apresentada na literatura, visando o processamento de dados em tempo real, como: Spark Streaming (ZAHARIA et al., 2012), Twitter Heron (KULKARNI et al., 2015), Apache Flink (CARBONE et al., 2015)

Por fim, outra iniciativa que podemos destacar é o Apache Edgent, uma plataforma direcionada ao processamento de fluxo de dados na borda da rede. A plataforma possui um modelo de programação semelhante ao Apache Storm, podendo também ser usada em conjunto com o Apache Storm. Na seção a seguir, serão descritas demais indicações sobre esta iniciativa.

[3] https://lmax-exchange.github.io/disruptor/
[4] http://zeromq.org/
[5] http://netty.io/

2.3.4 Apache Edgent

O Apache Edgent inicialmente chamado de Apache Quarks é a iniciativa a IBM [6] para realizar processamento de fluxo de dados na borda da rede. Isso possibilita impulsionar o processamento em IoT dos sistemas centralizados para dispositivos de borda. A plataforma é de código aberto podendo ser incorporada a dispositivos de pequeno porte. Direcionada para a análise de dados, permite análises locais, em tempo real, sobre fluxo contínuos de dados provenientes de equipamentos, veículos, sistemas, eletrodomésticos, dispositivos e sensores de todos os tipos (por exemplo, *Raspberry Pi* ou *smartphones*) (EDGENT, 2017).

Na prática o Edgent tem um modelo de programação semelhante ao Apache Storm, baseado em topologias e já possui suporte a uma série de recursos importante comuns a SPFD, como: agregações (*aggregate*), mapeamento (*map*), filtro(*filter*), separação (*split*), união (*union*), janelas de tempo e de contagem [7]. Além desses recursos, já existe suporte há uma vasta opção de conectores, como o Apache Kafka, usado no experimento. Até a elaboração dessa dissertação não havíamos encontrado na literatura nenhum trabalho que relatasse qualquer avaliação sob o Apache Edgent.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada à fundamentação teórica sobre os conceitos em torno deste trabalho. Foram apresentadas e discutidas as principais definições sobre IoT existentes na literatura, passando até mesmo pelo conceito de Computação Ubíqua que a precedeu. Algumas das características presente em ambientes de IoT também foram expostas, sendo direcionadas principalmente aos dispositivos de pequeno porte. Seguindo pelo capítulo, outros conceitos importantes foram apresentados como: Big Data, Arquitetura Lambda, Processamento de fluxo de dados direcionado aos SPFDs. No capítulo subsequente é apresentado o estado da arte em relação as plataformas, arquiteturas, sistemas e avaliações de desempenho sobre SPFD.

[6] https://www.ibm.com/br-pt
[7] São espaços de tempo ou de quantidade onde os dados são processados

3 ESTADO DA ARTE

Este capítulo de estado da arte é direcionado a trabalhos que propõem arquiteturas ou plataformas baseadas na arquitetura Lambda e que apresentam avaliação em termos de performance. Também são apresentados trabalhos que realizaram avaliações de desempenho através de *benchmarking*, com o foco nas principais plataformas de processamento de fluxo de dados em tempo real para ambiente distribuído. Destacamos que, durante o processo de elaboração desta dissertação não encontramos na literatura trabalhos que façam a relação entre sistemas de processamento de fluxo de dados em distintos ambientes de computação distribuída, como: Computação em Nuvem e Computação em Névoa.

3.1 ARQUITETURA LAMBDA

No campo de Cidade Inteligentes (do inglês, Smart Cities) CiDAP (CHENG et al., 2015) é uma proposta baseada na arquitetura Lambda, mas com algumas adaptações. A proposta é orientada para ser uma plataforma de Big Data para Cidades Inteligentes que pretende preencher a lacuna existente entre o que uma plataforma de Big Data é, em uma visão de alto nível, e como deve funcionar adequadamente na sua execução. Basicamente, a proposta divide o processamento e o armazenamento em duas camadas, chamadas: Big Data Processing e Big Data Repository. A Big Data Repository (BDR) é a camada que mais se assemelha com a definição de camada de velocidade (do inglês, speed layer) da arquitetura Lambda. Nesse cenário, essa camada é responsável pelo processamento e armazenamento de todos os dados históricos de menor complexidade e em tempo real. Os dados são registrados em um formato JavaScript Object Notation (JSON) no Apache CouchDB, um sistema de armazenamento de dados distribuídos baseado em *NoSQL*, que suporta *Map-Reduce* para geração e incremento de visões (do inglês, *views*) em tempo real. Um aspecto que precisa ser ponderado nessa proposta é a capacidade de desempenhar tanto o processamento em lote como em tempo real. Partindo do ponto de vista da arquitetura Lambda, que implementa essa separação, esse pode ser um aspecto questionável nessa proposta, já que aproxima as complexidades de processamento em lote e em tempo real na mesma camada, sendo contrário à proposta de Lambda, que visa separar as complexidades. Para avaliar a camada BDR, um experimento foi conduzido para encontrar quantas atualizações de dados de sensores podem ser coletadas por uma instância do Apache CouchDB. O experimento atingiu seu limite com aproximadamente 300 atualizações por segundo em 32 instâncias do Apache CouchDB.

Uma arquitetura baseada na união da plataforma AllJoyn e arquitetura Lambda é proposta em (VILLARI et al., 2014), uma solução escalável capaz de gerenciar, armazenar e analisar um grande volume de dados, além de se adaptar aos requisitos de ambien-

tes inteligentes na IoT. Todas as aplicações são incorporadas em sistemas embarcados (por exemplo, eletrodomésticos, televisores, luzes, termostatos). Cada um dos ambientes domésticos está conectado ao barramento (*Daemon Bus*), que é responsável pelo gerenciamento de aplicativos ambientais (por exemplo, sala, escritório) e cada uma das aplicações ambientais está vinculada ao Apache Storm para o processamento de dados em tempo real. Para cada uma das aplicações ambientais ou cômodos do apartamento, há uma topologia dentro do Apache Storm. Essa definição é interessante porque permite um melhor controle e gerenciamento de cada ambiente individualmente, aplicando padrões de dados no processamento. Na avaliação experimental, o apartamento inteiro é definido como um *cluster* e são utilizados três *hosts*, mas não foram descritos os resultados obtidos, principalmente relacionados a métricas de desempenho. Seguindo ainda o contexto da IoT, a arquitetura λ-COAP (DÍAZ; MARTÍN; RUBIO, 2016) é apresentada propondo a integração entre ambientes de computação em nuvem e IoT através da arquitetura Lambda, além de um *middleware* para lidar com o Protocolo de Aplicação Restrita (do inglês, Constrained Application Protocol (CoAP)). A camada de velocidade desse projeto também é composta pela plataforma de processamento de fluxo de dados em tempo real, Apache Storm, que recebe o fluxo de dados através do sistema de filas de mensagem distribuída, nesse caso o Apache Kafka. A camada também é responsável pela geração de visões pré-computadas em tempo real.

Um dos sistemas mais complexos de criação humana são as redes de energia elétrica (LIU et al., 2015) e essa complexidade requer sistemas capazes de gerenciar grandes volumes de dados em alta velocidade e fazer uma análise avançada. Neste contexto, (LIU et al., 2015) propõe um sistema de processamento em Big Data para redes inteligentes com base na arquitetura Lambda e no padrão de processamento de fluxo de dados, o Processamento de Eventos Complexos (do inglês, Complex Event Processing (CEP)). Os dados mais recentes são recebidos da Infraestrutura de Medição Avançada (do inglês, Advanced Metering Infrastructure (AMI)), sendo enviados posteriormente para a arquitetura Lambda. A camada de velocidade desta proposta contém duas extensões do Apache Spark: (i) *Spark Streaming*, que suporta processamento de fluxo de dados em tempo real; (ii) *Spark SQL*, uma outra extensão, que executa consultas em fluxo de dados usando o Structured Query Language (SQL), mas não propriamente um motor de CEP. Entretanto, é uma alternativa viável, por permitir a criação de regras que podem ser executadas dentro dos fluxo de dados e tomar decisões, uma associação que pode ser altamente poderosa. O experimento foi auxiliado por uma aplicação que simulou 1.594 clientes por metro quadrado e testou o processamento de funções essenciais (por exemplo, máximo e mínimo diário, semanal, média mensal de consumo) em um conjunto de dados de 55 milhões de registros. Os resultados alcançados foram de menos de dez segundos para obtenção de resultados. Também foi exposto que o Spark e suas extensões utilizadas foram eficientes na execução do cenário.

Com o crescimento exponencial dos dados, o uso de métodos e técnicas convencionais de coleta, processamento, armazenamento e análise de dados tornou-se pouco eficiente. Por isso, SAMURAI (PREUVENEERS; BERBERS; JOOSEN, 2016) propõe uma arquitetura com consciência de contexto e recursos para extração de conhecimento (por exemplo: CEP, aprendizado de máquina e representação de conhecimento). A camada de velocidade foi implementada usando Spark Streaming, complementada com o motor de processamento de eventos complexos Esper e a extensão do Spark para aprendizagem de máquina chamada MLib. Uma consideração a ser feita é que o Apache Storm é usado em algumas estruturas preliminares do SAMURAI. É relatado que o Spark Streaming não obteve a mesma propriedade de baixa latência que o Apache Storm, embora nenhuma avaliação empírica tenha sido relatada para embasar a afirmação. O primeiro experimento apresenta resultados de processamento de aproximadamente 10.000 eventos por segundo e simula com isso o acesso de 250 usuários. Já o segundo experimento, analisou a escalabilidade horizontal com um número crescente de usuário produzindo dados. Foram usados até 12 nós e obtendo uma escalabilidade quase linear.

A arquitetura SOLID (MARTÍNEZ-PRIETO et al., 2015) concentra-se na capacidade de coletar, armazenar e expor um grande volume de dados em tempo real com sistemas de alto desempenho em tempo real e gerenciamento de grande volume de dados semânticos. As funções arbitrárias, em um conjunto de dados aleatórios em tempo real, são um problema grandioso e essa arquitetura quer abordar esses dois desafios. Essa proposta também se baseia na arquitetura Lambda, separando camadas e complexidades, armazenando e consumindo dados em tempo real, além de propor adaptações para suportar o *framework* de Descrição de Recursos (do inglês, *Resource Description Framework (RDF)*).

O paradigma de Computação em Nuvem é um ambiente promissor, entregando *TI* como serviço para as indústrias e pesquisadores implantarem suas aplicações (KIRAN et al., 2015). Usando os recursos do *RDF*, a arquitetura (KIRAN et al., 2015) apresenta uma proposta de baixo custo para o processamento de grande volume de dados em tempo real e em lote baseado na arquitetura Lambda. Os experimentos apontaram que é possível fornecer uma prova de conceito para o processamento de uma grande quantidade de dados na arquitetura Lambda, usando a infraestrutura de Computação em Nuvem. Como o cenário faz uso de produtos e soluções da *Amazon*, não foram inteiramente claros os detalhes de implementação da arquitetura, mas é feita referência baseada em relatório técnico do uso do Apache Storm na camada de velocidade. Os resultados obtidos no trabalho não estão focados na avaliação do desempenho, mas com base nos custos dos produtos oferecidos.

Tabela 3 – Relação de trabalhos baseados em Lambda no processamento de fluxo de dados

Autor	Ano	Contexto	Arquitetura	SPFD
VILLARI	2014	Smart Home	Lambda	Apache Storm
CHENG	2015	Smart Cities	Lambda	Apache CouchDB*
LIU	2015	Smart Grids	Lambda	Apache Spark
MARTÍNEZ	2015	Semantic data	Lambda	-
KIRAN	2015	Cloud	AWS	Apache Storm
DÍAZ	2016	Cloud/IoT	Lambda	Apache Storm
PREUVENEERS	2016	Context awareness	Própria	Apache Spark

3.2 BENCHMARKINGS DE SISTEMAS DE PROCESSAMENTO DE FLUXO DE DADOS

Nessa seção, são relatados trabalhos que trazem avaliações de desempenho realizadas através de *benchmarkings* e outros métodos de análise. São enfatizadas as principais plataformas de processamento de fluxo de dados em tempo real em ambiente distribuído, sendo algumas não avaliadas nesta dissertação. Nesses trabalhos são examinados aspectos de desempenho, como capacidade de computação e tolerância a falhas. Por fim, na literatura não foram encontrados trabalhos que abordassem a comparação entre sistemas de processamento de fluxo de dados aplicados a diferentes cenários, como por exemplo: processamento distribuído e em dispositivo de borda.

Os sistemas de processamento de fluxo de dados (do inglês, Data Stream Processing System) Apache Storm e Spark Streaming são comparados e avaliados em Stream Bench (LU et al., 2014) e as características de processamento foram identificadas, tomando como métricas de referência: latência e taxa de transferência. O trabalho também estabelece comparações entre medidas de desempenho e características distribuídas, tais como: tolerância a falhas, disponibilidade, escalabilidade. Os resultados apontam para um maior rendimento e menor impacto para a falha do Spark Streaming, enquanto o Apache Storm tem uma latência muito menor, exceto quando há um aumento no tamanho em escala dos dados. Em seguida, o Stream Bench (LU et al., 2014) foi expandido (QIAN et al., 2016), incluindo Apache Samza e Apache Storm com Trident[1]. Os experimentos verificaram que o Apache Storm com o Trident satura os recursos da rede, trazendo um maior rendimento. Por fim este trabalho aponta a capacidade limitada e os problemas de maturidade do Apache Samza.

Em (CHINTAPALLI et al., 2016), são avaliados Apache Storm, Spark Streaming e Flink. Destacando também que o Apache Storm possui a característica de baixa latência, também acompanhada por Flink, enquanto a Spark é capaz de ter um maior rendimento, já que possui a latência ligeiramente maior. É importante enfatizar que, nos trabalhos descritos nesta seção, o aspecto do desempenho foi o mais avaliado, mas esse é apenas um

[1] http://storm.apache.org/releases/2.0.0-SNAPSHOT/Trident-API-Overview.html

fator entre muitos outros, como segurança e integração, que não foram analisados e que também não são o foco desta dissertação.

Direcionado ao campo da IoT (LOPEZ; LOBATO; DUARTE, 2016), este trabalho de avaliação tem a característica de abordar dados reais dos domínios de transporte inteligente e monitoramento urbano da IoT. Semelhante ao trabalho anteriormente citado (CHINTAPALLI et al., 2016), também são avaliados o Apache Storm, Spark Streaming e Apache Flink. Fica evidente na literatura que essas plataformas são as mais relevantes; além disso, o Apache Kafka é empregado com a finalidade de ser o broker de mensagem, lidando com o serviço de publish/subscribe. Essas plataformas são avaliadas em dois experimentos, divididos essencialmente na avaliação de desempenho e de tolerância a falhas. Os resultados apresentados no primeiro experimento apontam uma maior taxa de transferência no Apache Storm, sendo usados ou não os recursos de paralelismo. Além disso, o Apache Storm apresenta um comportamento linear até uma determinada faixa de paralelismo. O experimento seguinte foi orientado para avaliar a tolerância a falhas, simulada a partir do desligamento de um dos nós. O Apache Storm leva algum tempo para se recuperar, havendo também uma perda considerável de mensagens. Dentre as plataformas avaliadas, o Spark Streaming consegue uma redistribuição mais rápida dos processos sem ocorrer a perda de mensagem e desempenho.

Os benckmarks (SHUKLA; SIMMHAN, 2016) (SHUKLA; CHATURVEDI; SIMMHAN, 2017) podem ser considerados os mais completos encontrados na literatura. São avaliadas 13 tarefas classificadas em diversificadas categorias e também relacionadas a IoT, sendo avaliado no SPFD Apache Storm. Devido à circunstância de diversos tipos de tarefas serem avaliadas e, com isso serem variados os custos computacionais, ocorreu uma alta variação nas taxas de desempenho. Por exemplo, a taxa de transferência variou entre 3.000 e 68.000 mensagens por segundo; a latência variando entre aproximadamente 0 e 2.500 milissegundos; entre 60% e 70% de uso da unidade central de processamento, (do inglês, Central Processing Unit (CPU)). Por fim, em todos os casos houve no máximo setenta por cento de uso da memória de acesso randômico (do inglês, Random Access Memory (RAM)).

Tabela 4 – Catálogo de trabalhos sobre benchmarkings no processamento de fluxo de dados

Autor	Ano	SPFD
LU	2014	Apache Storm/Spark Streaming
QIAN	2016	Apache Storm/Spark Streaming/Apache Samza/Trident
CHINTAPALLI	2016	Apache Storm/Spark Streaming/Flink
LOPEZ	2016	Apache Storm/Spark Streaming/Flink
SHUKLA	2017	Apache Storm

3.3 CONSIDERAÇÕES FINAIS

Neste capítulo, foram expostos trabalhos que relatam o mais alto nível de pesquisa nesta área. Para uma melhor estruturação e compreensão dividimos em duas seções. Na primeira, descrevemos os trabalhos que realizaram algum tipo de experimento avaliando arquiteturas de Big Data e que também traziam em seu escopo a avaliação de SPFD. Nesta seção ficou evidente que a arquitetura Lambda foi encontrada com maior frequência na literatura. Já a segunda seção apresenta trabalhos que avaliaram SPFD através da técnica do benchmarking. No capítulo seguinte serão descritos os passos realizados para execução do experimento, para por fim, apresentar e discutir os resultados obtidos.

4 AVALIAÇÃO DE DESEMPENHO

Neste capítulo é apresentada a descrição dos experimentos realizados. Os mesmos, foram conduzidos mediante múltiplas etapas, através de uma metodologia sistemática para avaliação de desempenho. Essa metodologia é largamente descrita na literatura e de forma objetiva auxilia no processo de construção da avaliação. Ainda no capítulo, são definidos os objetivos deste estudo e seus limites, para posteriormente serem listados os serviços disponibilizados, as métricas de desempenhos avaliadas, assim como os parâmetros e fatores. São acrescentados maiores detalhes sobre a técnica de avaliação de desempenho e a carga de trabalho, para a partir de uma descrição mais detalhada do *design* do experimento apresentar os resultados obtidos.

4.1 OBJETIVOS

O objetivo principal da avaliação de desempenho se entrelaça com o da pesquisa, que fundamenta-se em construir uma avaliação de desempenho para avaliar o comportamento em termos de desempenho em sistemas de processamento de fluxo de dados em distintos cenários baseados em computação distribuída. De forma ainda mais sucinta, objetivamente serão avaliados os SPFDs em dois distintos cenários distribuídos. Ressaltando que, são realizadas duas avaliações desacopladas, já que estamos em um conjunto diferente de hardware e software.

Pretendemos ainda que os resultados desta avaliação possa ter o papel de referência em avaliações de SPFD em tempo real. Além do mais, acreditamos que a partir dos resultados apresentados e suas conclusões, seja possível contribuir com os subsídios necessários para apoiar, se for oportuno, a possibilidade de transpor parte do processamento de fluxo de dados para dispositivos localizados na borda da rede.

4.2 LISTA DE SERVIÇOS

Os serviços[1] oferecidos pelo sistema[2] são essencialmente analise estatística, que em ambos os cenários são aplicadas a partir de função de agregação. Na função de agregação, são executados separadamente as operações estatísticas mais comuns: contagem, média, máximo e mínimo. Na listagem 4.1 é apresentada um trecho do código referente a topologia do Apache Storm, onde são invocados cada um dos serviços.

Listing 4.1 – Trecho de código contido na topologia do Apache Storm

[1] Pode ser definido como as funcionalidades implementadas a serem avaliadas.
[2] O nosso estudo de caso.

```
Stream stream = topologyTrident.newStream("lines", kafkaSpout)
.name("spout-kafka").parallelismHint(6)
.each(new Fields("line"), new ExtractSpeedJson(),
new Fields("id", "speed"))
.aggregate(new Fields("id", "speed"), new CounterAggregator(),
new Fields("key", "value")).name("operation")
.aggregate(new Fields("id", "speed"), new MaxAggregator(),
new Fields("key", "value")).name("operation")
.aggregate(new Fields("id", "speed"), new AverageAggregator(),
new Fields("key", "value")).name("operation")
.parallelismHint(2);
```

O Apache Edgent também tem como característica ser baseado em topologia, com isso o desenvolvimento se assemelha com o Apache Storm. Na listagem 4.2, é mostrado um trecho do código do Apache Edgent, da mesma forma, nessa parcela de código são invocados cada um dos serviços.

Listing 4.2 – Trecho de código contido na topologia do Apache Edgent

```
TStream<JsonObject> aggregateOP = JsonAnalytics.aggregate(
sensorWindow, id, valueAggregate, MEAN, MAX, MIN)
.map(SensorAnalytics.tupleCount(ID_COUNT_AGGREGATE)).tag("COUNT
   ")
.map(new KafkaConnectorEdgent().publish());
```

Além disso, para que fossem desenvolvidos os serviços de analise estatística, um outro tipo de serviço é executado previamente, o analisador (do inglês, *parse*). O analisador recebe uma mensagem com o conteúdo em formato *JSON* e transforma para um tipo Java, extraindo o campo de interesse. Só posteriormente a essa extração da informação é realizada a análise estatística. Não entraremos em outros detalhes sobre essa operação, a mesma não fez parte da avaliação de desempenho, sendo usada apenas para preparar os dados para a analise estatística.

4.3 MÉTRICAS DE DESEMPENHO, PARÂMETROS E FATORES

Seguindo a metodologia (JAIN, 1991), para cada um dos serviços acima listados foram definidas métricas que avaliam a capacidade quantitativa de execução dos serviços. Ressaltamos que essas medidas (métricas, parâmetros e fatores) não foram estabelecidas de forma arbitrária, mas definidas a partir de uma rigorosa análise.

Métricas

1. Taxa de transferência: essa métrica foi mensurada através da quantidade de dados processados pelo SPFD, em uma taxa medida em quantidade de tuplas por segundo.

2. Latência: foi medida a latência de execução de cada uma das operações.

3. Taxa de uso da CPU: medimos o percentual de uso da CPU em cada nó, no caso do cenário distribuído. Em virtude dos nós usados no experimento distribuído terem o mesmo modelo de processador, é garantido um mesmo padrão de medida.

4. Taxa de uso da memória RAM: da mesma forma também foi medido o percentual de uso de memoria RAM em cada nó.

5. Taxa de envio e recebimento de dados: é medida a taxa de transferência entre os nós responsáveis pelo processamento distribuído. Essa taxa foi mensurada em megabytes, sendo medida tanto a taxa de envio como a de recebimento dos dados.

Parâmetros Os parâmetros em uma avaliação de desempenho podem ser divididos em dois grupos: os parâmetros de sistema e os parâmetros de carga. Os parâmetros de sistema estão diretamente relacionado ao que afeta o desempenho do sistema, já os de carga são relativos aquilo que se diz sobre o desempenho da carga. Os parâmetros são cruciais para que sejam avaliados diversificadas condições (JAIN, 1991).

[**Parâmetros Sistema**]

1. Velocidade de processamento da CPU
2. Disponibilidade de memória RAM
3. Velocidade da rede
4. Tolerância a falhas
5. Número de brokers e partições na fila de mensagem

[**Parâmetros Carga**]

1. Número de mensagens lidas do conjunto de dados
2. Número de mensagens enviadas

Fatores Janelas de contagem (apenas no Apache Edgent)

4.4 TÉCNICA DE AVALIAÇÃO

De acordo com (JAIN, 1991) são três as técnicas de avaliação de desempenho: modelagem analítica, simulação e aferição. Para selecionar a técnica adequada alguns fatores são levados em consideração, tempo, recursos disponíveis e a precisão almejada. A partir disso e de acordo com nosso cenário, a aferição foi definida como técnica de avaliação, por ser focada em medições realizadas em protótipo.

4.5 CARGA DE TRABALHO

Com o intuito de medir desempenho em sistemas computacionais, é necessário haver um mecanismo de carga de trabalho (HOORN; ROHR; HASSELBRING, 2008). De acordo com (JAIN, 1991) a carga de trabalho consiste em uma lista de solicitações ao serviço do sistema. Essa definição é essencial, que de forma detalhada entendemos como um conjunto de todas as entradas que o cenário avaliado recebe ao longo de um determinado período de tempo. A carga de trabalho (do inglês, *Workload*) é o modelo de entrada de dados usado no cenário de *benchmarking*. Frequentemente, em estudos referentes a avaliação de desempenho, o termo "*workload*"é sempre empregado para qualquer carga de trabalho.

4.5.1 Caracterização

De acordo com o objetivo que apresentamos anteriormente, buscamos compreender, definir e caracterizar a carga de trabalho sintética. Esse elemento é o ponto de partida do experimento, onde estão inclusas funções para a leitura do conjunto de dados, agendamento da execução, extração de métricas e logs do comportamentais. Na Figura 6 é mostrada uma visão do funcionamento interno da carga de trabalho. No primeiro momento é realizada a leitura dos dados, que se encontram em formato de texto e que serão descritos na próxima seção. Para seguir um padrão comum em sistemas para o domínio da IoT, os mesmos são formatados no padrão JSON, conforme a Listagem 4.3. Os campos da Listagem 4.3 são os mesmos do *DataSet* (Seção 4.5.2). De forma paralela uma conexão por meio de Socket assíncrono é estabelecida, para por fim os dados serem enviados para o *middleware*.

Listing 4.3 – Exemplo de mensagem enviada

```
{
    "IdentificadorVeiculo": "55556",
    "tempo": "08:06:01",
    "placa": "BC7142",
    "latitude": "113.902962",
    "longitude": "22.557901",
    "velocidade": "19.000000"
}
```

A carga de trabalho é equivalente, tanto no cenário de processamento de fluxo de dados distribuídos como no cenário de processamento de fluxo de dados em dispositivo de borda., entretanto diferentes resultados são obtidos para o envio dos dados, pois estamos em cenários de hardware bastante distintos. No cenário distribuído, no qual o Apache Storm é avaliado são enviadas aproximadamente 7 milhões de mensagem por minuto e a execução é realizada em uma máquina do Tipo01, conforme a Tabela 5. No cenário em dispositivo de borda, onde o Apache Edgent é investigado, aproximadamente entre 200.000 e 300.000 mensagens são enviadas por minuto, sendo executado em um dispositivo descrito na Tabela 6. Em ambos a dimensão da mensagem segue a mesmo tamanho, em torno de 120 *bytes*.

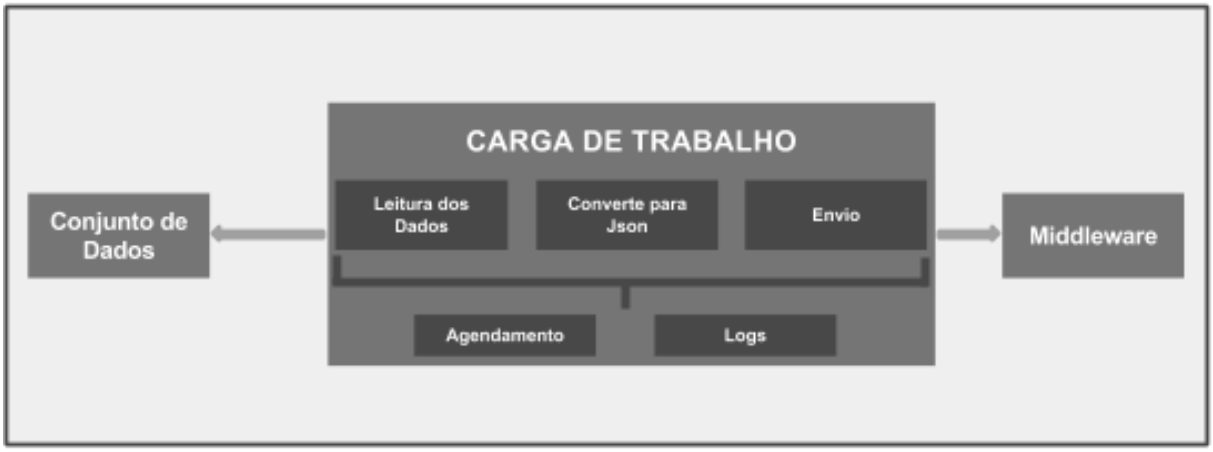

Figura 6 – Apresentação da Carga de Trabalho

Próprio Autor

4.5.2 DataSet

Um conjunto de dados reais e abertos da cidade chinesa de Shenzhen[3] foi usado nesse estudo. Este conjunto de dados, engloba vários tipos de informações relativas a cidade de Shenzhen, como: dados de celulares, dados de smartcards, informações relativas ao

[3] www-users.cs.umn.edu/zhang/data.html

deslocamento de táxi, ônibus e caminhões. No entanto, foram utilizados apenas os dados relativos a deslocamento, que essencialmente contêm as informações como: identificador, tempo, dados de localização (latitude e longitude) e velocidade. A velocidade do veiculo é o campo de interesse, o dado presente neste campo é usado no processamento, onde é realizada a análise estatística Este conjunto de dados foi escolhido por ter um tamanho considerável de dados, aproximadamente 50 milhões de registros. Podemos afirmar, que esse conjunto de dados é consideravelmente maior do que alguns relatados em trabalhos relacionados na literatura (LU et al., 2014) (QIAN et al., 2016).

4.6 EXPERIMENTOS

Na presente seção, iremos fornecer maiores detalhes sobre *design* dos experimentos, essa descrição auxilia a extração de informação com um menor esforço. Basicamente, foram conduzidos dois experimentos, cada um executado em um dos cenários propostos. O objetivo final desse experimento, como de qualquer um outro, é gerar um conhecimento (WAINER et al., 2007). No primeiro experimento se investigou as métricas de desempenho do Apache Storm no ambiente distribuído, já o segundo também explorou algumas dessas métricas, entretanto em um outro contexto, no processamento local em um dispositivo de borda. Ainda sobre os experimentos, ambos foram executados pelo mesmo período de tempo de 60 minutos. Para o cenário distribuído, esse é um valor razoável, pois é maior do que o tempo relatado em outros trabalhos (SHUKLA; SIMMHAN, 2016) (CHINTAPALLI et al., 2016). Nas subseções decorrentes, serão apresentados os detalhes referente ao *design* de cada experimento.

4.6.1 Experimento 01: Processamento Distribuído

Neste experimento, elaboramos um modelo arquitetural baseado na camada de velocidade da arquitetura Lambda, equivalente á arquitetura Kappa. A proposta é dividida em quatro partes, e cada uma delas possui sua responsabilidade. Na Figura 7, podem ser vistos cada um dos seguintes componentes: a carga de trabalho, *middleware*, *broker* de fila de mensagem e a parcela responsável pelo processamento do fluxo de dados, onde está Apache Storm.

O fluxo de dados inicia o caminho quando é produzido pela carga de trabalho, componente já relatado na seção 4.5. No segundo momento, o *middleware* recebe os dados enviados pela carga de trabalho por meio de conexão através de um *Socket* assíncrono, ideal para esse cenário, por prover uma comunicação mais eficiente e escalável. Após o recebimento dos dados, o *middleware* envia os dados recebidos da carga de trabalho para o *broker* do Apache Kafka, onde dois tópicos estão previamente criados, um para receber os dados e outro para saída dos dados processados.

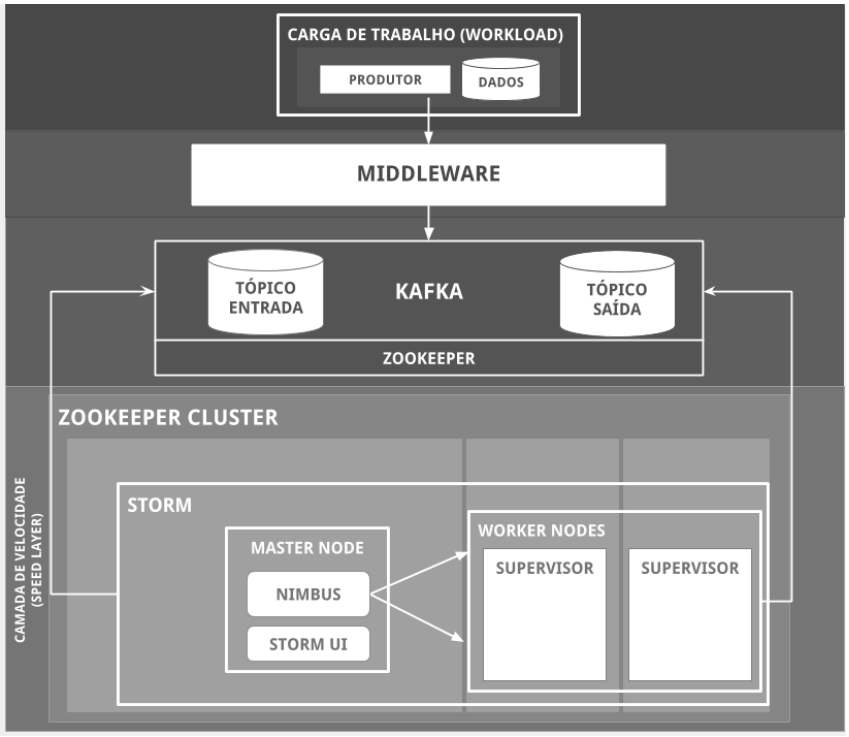

Figura 7 – Arquitetura projetada para o experimento no Apache Storm

Próprio Autor

Por fim, o Apache Storm consome os dados do tópico de entrada e realiza o processamento [4], distribuído entre três máquinas. Esse é um cenário distante da realidade do ambiente de computação em nuvem, entretanto foi o cenário possível de ser avaliado, devido a limitações de disponibilidade de recursos computacionais. Na Figura 7 as máquinas responsáveis pelo processamento estão identificadas pelos nomes: *Master Node*, *Worker Nodes*. Na última etapa desse processo, os dados depois de serem processados são inseridos em um tópico de saída, finalizando o ciclo de execução do experimento.

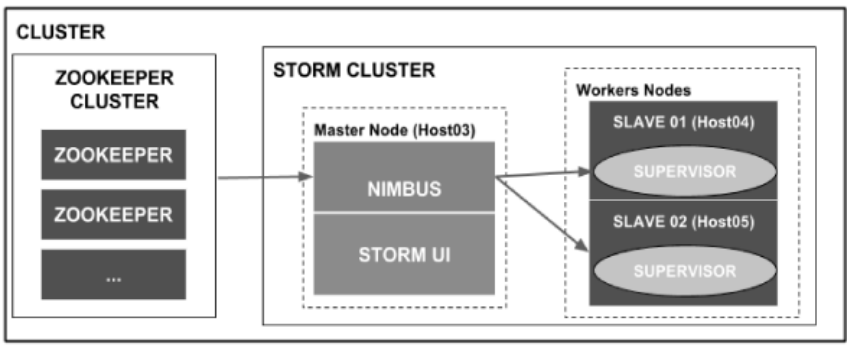

Figura 8 – Representação do Apache Storm em ambiente de Cluster

Próprio Autor

[4] São as funções descritas na seção Lista de Serviços

4.6.1.1 Recursos de Hardware

No experimento, foram empenhadas cinco máquinas, nas quais, três do *Tipo02* foram usadas para o processamento, formando um *cluster* do Apache Storm. As duas outras máquinas, uma foi usada para executar a carga de trabalho (Host01) e a outra para receber os dados (Host02), que são retidos em um *broker* de fila de mensagem com Apache Kafka. Na Tabela 5 é mostrado um resumo das configurações das máquinas usados no experimento distribuído.

Tabela 5 – Configurações das máquinas do experimento distribuído

	Host	RAM	CPU
Tipo01	Host 01	16 GB	Intel(R) Xeon(R) CPU ES-2660 v2 2.2GHz
Tipo02	Host 02	32 GB	
	Host 03		
	Host 04		
	Host 05		

4.6.2 Experimento 02: Processamento em dispositivo de borda

O experimento responsável pelo processamento local usando o Apache Edgent, é implantado em um único dispositivo de borda, entretanto usando a mesma carga de trabalho do experimento anterior. O dispositivo é a *Raspberry Pi*, um computador de baixo custo e alto desempenho, podendo exercer as mais diversas tarefas de um *desktop* convencional, como: navegar na internet, execução vídeos em alta definição, trabalhar com processadores de texto, planilhas (FOUNDATION, 2017) e até mesmo desempenhar processamento de dados no ecossistema da IoT.

Como podemos ver na Figura 13, o experimento pode ser dividido em duas partes, a primeira onde a carga de trabalho é executada e simultaneamente o Apache Edgent processa os dados, e uma segunda parte onde os dados são recebidos no *broker* do Apache Kafka. As operações ou serviços executados são os mesmos do experimento anterior e já relatados em detalhes na seção 4.1.

4.6.2.1 Recursos de Hardware

Semelhante à forma que descrevemos na subseção 4.6.1.1, descreveremos nesta seção os detalhes dos recursos usados nesse experimento. Por outro lado, diferente do que foi descrito anteriormente, nesse experimento um único dispositivo foi usado, sendo responsável pela carga de trabalho e o processamento dos dados. A Tabela 6 contém as informações dos recursos do dispositivo utilizado no experimento.

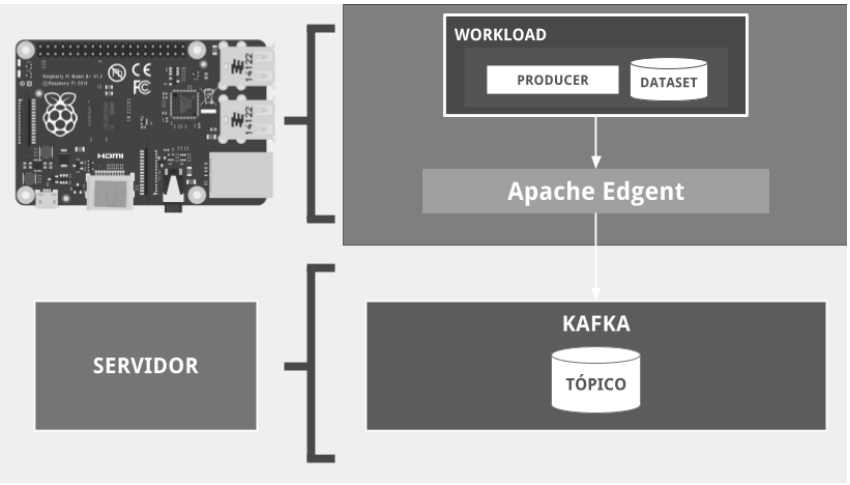

Figura 9 – Arquitetura projetada para o experimento em dispositivo de borda

Próprio Autor

Tabela 6 – Configurações do dispositivo *RaspberryPI*

Raspberry PI 3	
Modelo	B
Processador	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
Memória Ram	1GB
Armazenamento	32GB
Sistema Operacional	Raspbian (baseado no Debian)

4.7 RESULTADOS

Nesta seção, são apresentados os resultados obtidos na avaliação de desempenho em diferentes sistemas de processamento de fluxo de dados, em âmbito distribuído e local. A descrição dos resultados é baseada nas condições em que foram executados os experimentos. Isso é importante de ser destacado, pois não é justo que uma determinada tecnologia ou abordagem seja considerada totalmente correta ou errada, mas que foi apropriada ou não em certas circunstâncias (KITCHENHAM; CHARTERS, 2007). Entendemos que esse conceito descrito, que é oriundo da engenharia de software baseada em evidências nos auxiliou para uma condução mais sensata deste trabalho. Além disso, em áreas do conhecimento que se desenvolvem a partir de uma compreensão empírica onde nenhuma teoria cientifica é diretamente aplicável, experimentos e a observação dos dados resultantes são a única maneira de resolver determinados problemas. O nosso trabalho trata-se de um experimento planejado, de acordo com (MONTGOMERY, 2010), na perspectiva da estatística, os experimentos planejados são uma abordagem muito poderosa para estudar sistemas complexos. Na próxima subseção, serão apresentados os resultados obtidos através dos experimentos realizados.

4.7.1 Ambiente distribuído com Apache Storm

Os gráficos apresentados nas Figuras 10, 11, 12 e 13, trazem uma visão geral dos resultados recolhidos no experimento que avaliou o desempenho do Apache Storm. Cada Figura contém um conjunto de seis gráficos, que representam respectivamente: quantidade de tuplas emitidas, latência da execução, uso de RAM, quantidade de pacotes enviados, taxa de uso de CPU e quantidade de pacotes recebidos. Em todos os gráficos apresentados, a média foi usada como medida de representação, mas outras medidas estatísticas são apresentadas na tabela 7 contida no Apêndice D. Ressaltamos também que, os resultados referente à taxa de transferência a seguir apresentados, representam o total referente às duas máquinas (*Worker Nodes*) usadas para execução do processamento, conforme foi visto na Figura 8.

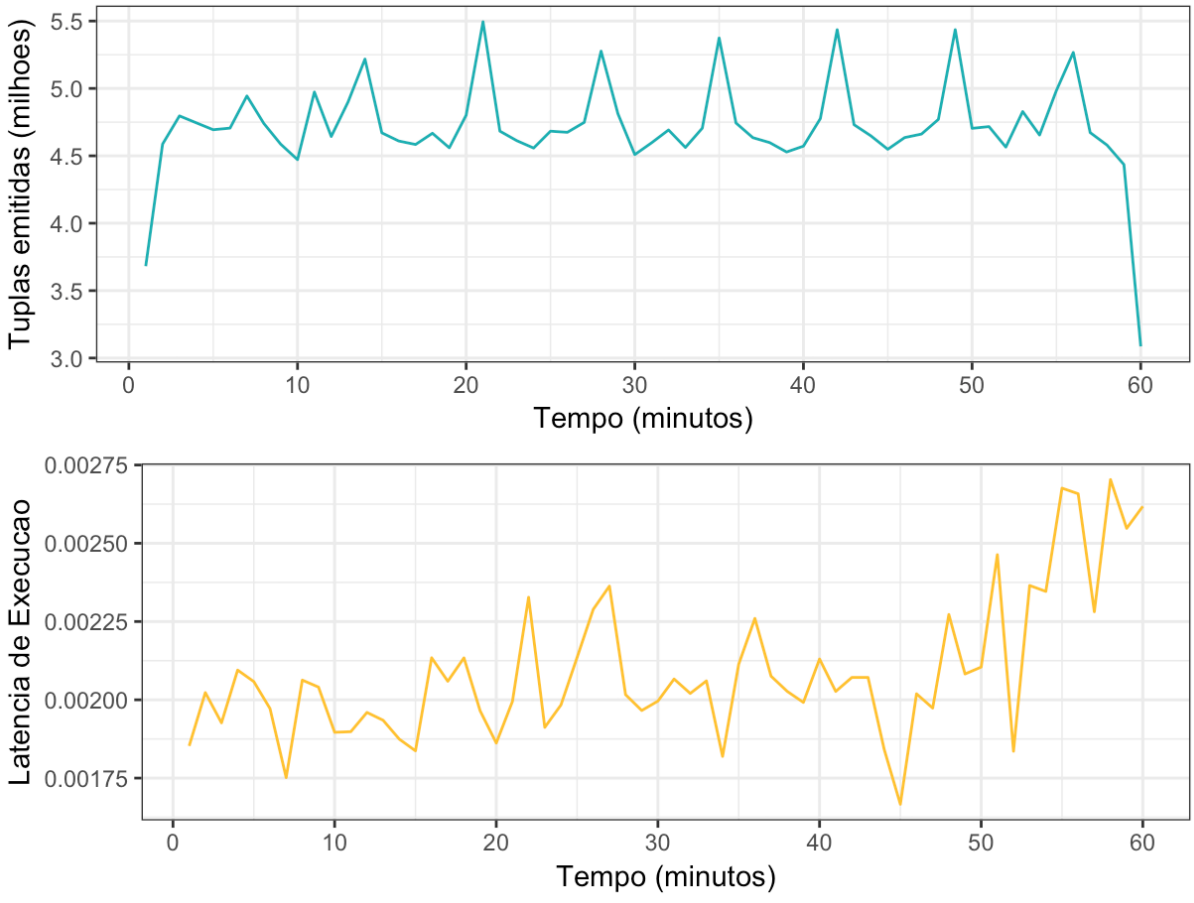

Figura 10 – Taxa de transferência e Latência no processamento do cálculo da média no Apache Storm

Fonte: Próprio Autor

Nos gráficos contidos nas Figuras 10, 11, 12 e 13, também são apresentados os resultados das métricas referentes ao serviços processados, abaixo um breve resumo dos resultados de cada um dos serviços avaliados.

Na Figura 10 são apresentados os resultados referentes ao processamento do serviço de

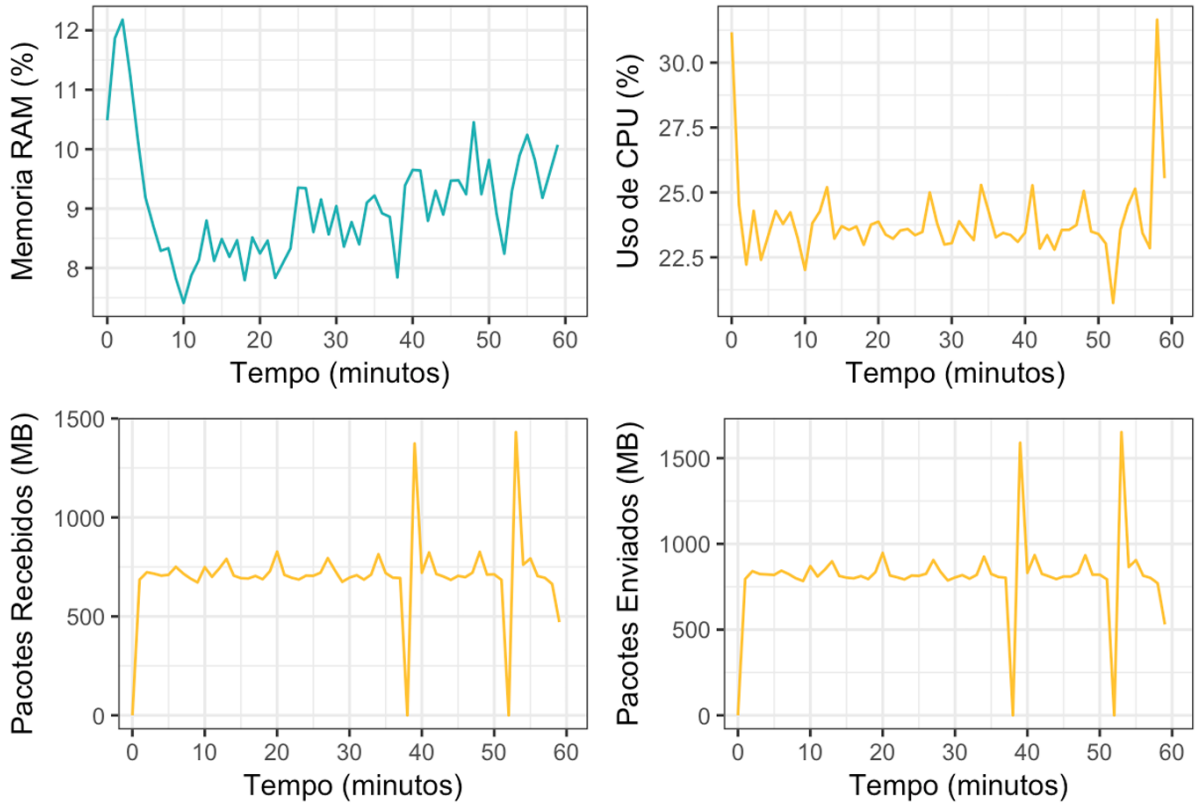

Figura 11 – Resultado das Métricas de no processamento do cálculo da média no Apache Storm

Fonte: Próprio Autor

cálculo da velocidade média do veiculo, extraídos do *DataSet* e enviado para os SPFDs. Para esse serviço o *Apache Storm* obteve uma taxa de transferência que variou entre entre 3.086.326 e 5.494.727 tuplas processadas por minuto, a latência: entre 0,0016 e 0,0027 milissegundos, o uso de Memória RAM: entre 7,41% e 12,17% (do total de 32 GB), o uso de CPU: entre 20,7% e 31,6% e a taxa de até 1.691 / 1.431 (enviados/recebidos) *megabytes* por minuto.

Na Figura 11 são exibidos os resultados referentes ao processamento do serviço de contagem de mensagens enviadas, a cada mensagem JSON recebida no Apache Storm essa informação é incrementada. Para esse serviço o Apache Storm obteve uma taxa de transferência que variou entre 4.328.383 e 5.659.412 de tuplas processadas por minuto, a latência: entre 0,0015 e 0,0021 milissegundos, o uso de Memória RAM: entre 7.3% e 12.2% (do total de 32 GB), o uso de CPU: entre 18.5% e 24.8% e a taxa de até 271 / 726 (enviados/recebidos) *megabytes* por minuto.

Dando continuidade a apresentação dos resultados, na Figura 12 são mostrados os resultados referentes ao processamento do serviço de analise estatística da velocidade máxima de cada veiculo, lembrando que as informações do veiculo são extraídas do DataSet. Para o processamento desse serviço o Apache Storm alcançou uma taxa de transferência que variou entre 3.498.518 e 5.596.479 de tuplas processadas por minuto, a latência: entre

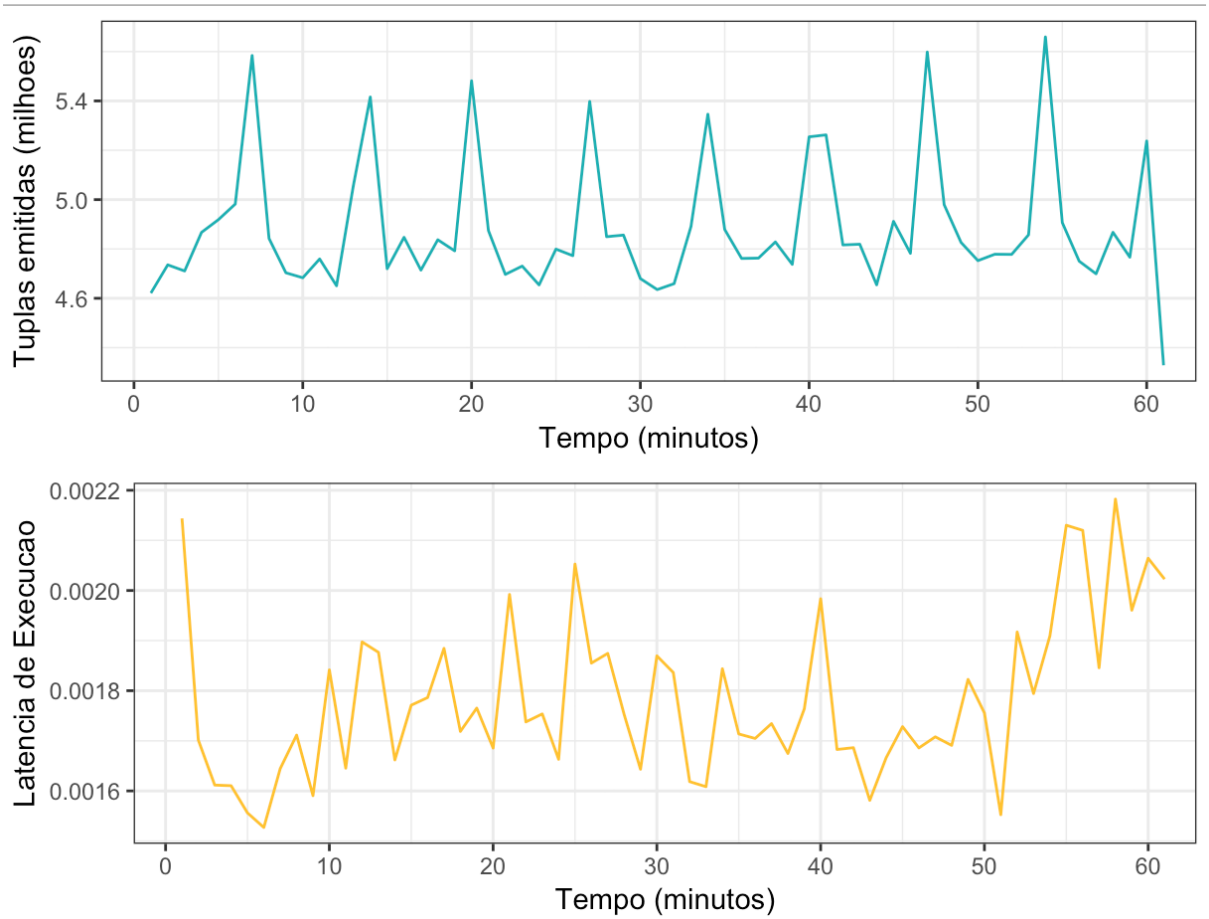

Figura 12 – Resultado do processamento de Contagem no Apache Storm: Taxa de transferência e latência

Fonte: Próprio Autor

0,0017 e 0,0023 milissegundos, o uso de Memória RAM: entre 7,04% e 12,14% (do total de 32 GB), o uso de CPU: entre 21,1% e 27,9% e a taxa de até 1.812 / 1.588 (enviados/-recebidos) *megabytes* por minuto.

Por fim, na Figura 13 os dados referentes ao processamento do serviço de analise estatística da velocidade mínima são retratados. Para o processamento desse serviço o Apache Storm alcançou uma taxa de transferência que variou entre 3.132.533 e 5.210.603 de tuplas processadas por minuto, a latência: entre 0,0017 e 0,0025 milissegundos, o uso de Memória RAM: entre 8,1% e 15,6% (do total de 32 GB), o uso de CPU: entre 6,4% 46,0% e a taxa de até 1.566 / 1.348 (enviados/recebidos) *megabytes* por minuto.

4.7.2 Dispositivo de borda com Apache Edgent

Os resultados das métricas se encontram expresso em gráficos nas Figuras 14, 15, 16 e 17, a seguir um resumo dos resultados alcançados por cada serviço é apresentado. Os gráficos abaixo apresentam as execuções ao longo de 60 minutos, onde foram avaliadas diferentes janelas de tempo, 5, 10 e 20 segundos. Também foram usados os mesmos serviços avaliados

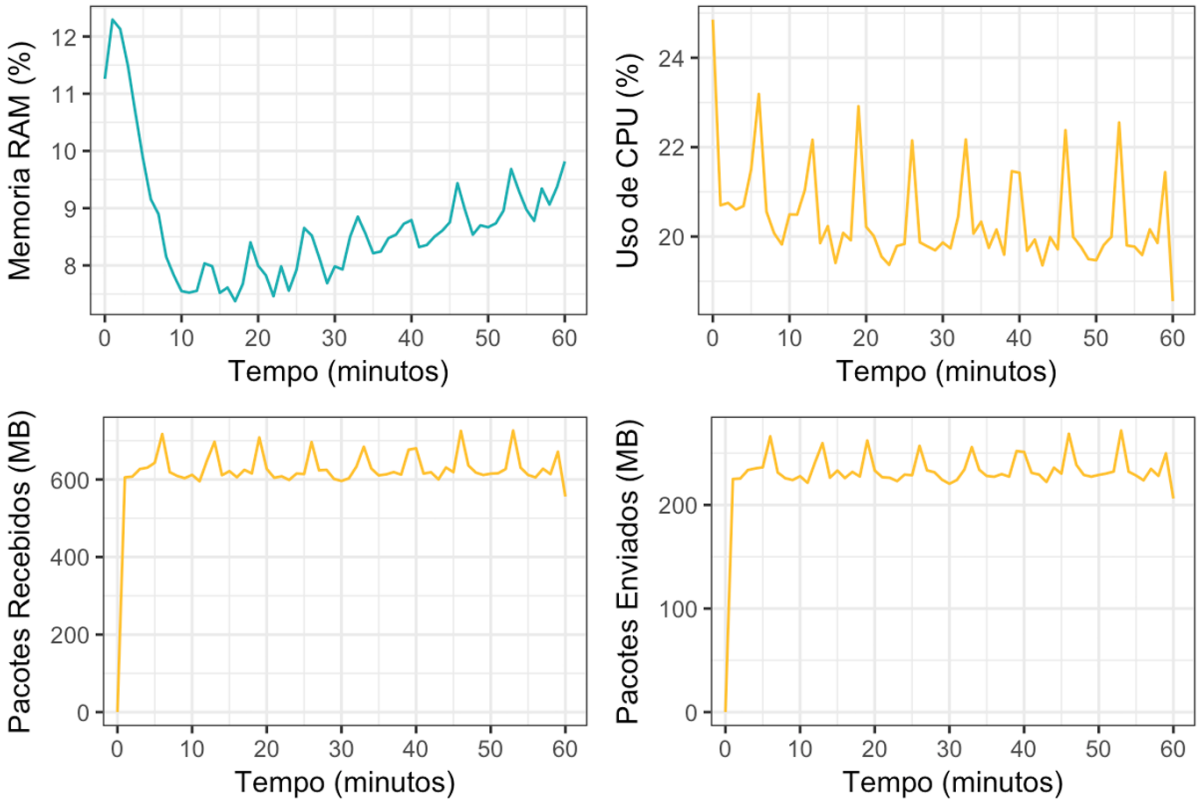

Figura 13 – Resultado do processamento de Contagem no Apache Storm: Taxa de transferência e latência

Fonte: Próprio Autor

no Apache Storm (contagem, média, máximo e mínimo), extraindo do dataset também a informação da velocidade do veículo.

Nas Figuras de 18 a 25 são apresentados os resultados referentes ao processamento de todos os serviços no Apache Edgent, em cada um dos gráficos são apresentados os valores relativos a três diferentes janelas de tempo: 5, 10, 20 segundos. Em relação a taxa de transferência a variação entre as diferentes janelas de tempo foi de aproximadamente 70%, com as menores janelas de tempo tendo melhor desempenho. Quanto maiores as janelas de tempo, mais informação precisou ser armazenada em cache, isso torna o processo mais lento e maior o consumo de memória, como será apresentado adiante. O consumo de memória nas menores janelas de tempo se estabeleceu entre 20% e 30% e nas maiores janelas alcançando acima dos 70%.

Embora estejamos nos referindo a um dispositivo de baixa capacidade, o mesmo possui um processador com quatro núcleos de processamento. Essa capacidade foi avaliada no experimento tendo o consumo variando entre 27% e 44%, o que demostra que esse dispositivo tinha a capacidade de suportar o processamento de um número de tuplas ainda maior. A seguir são apresentados os detalhes dos resultados das métricas relacionadas a cada uma das operações.

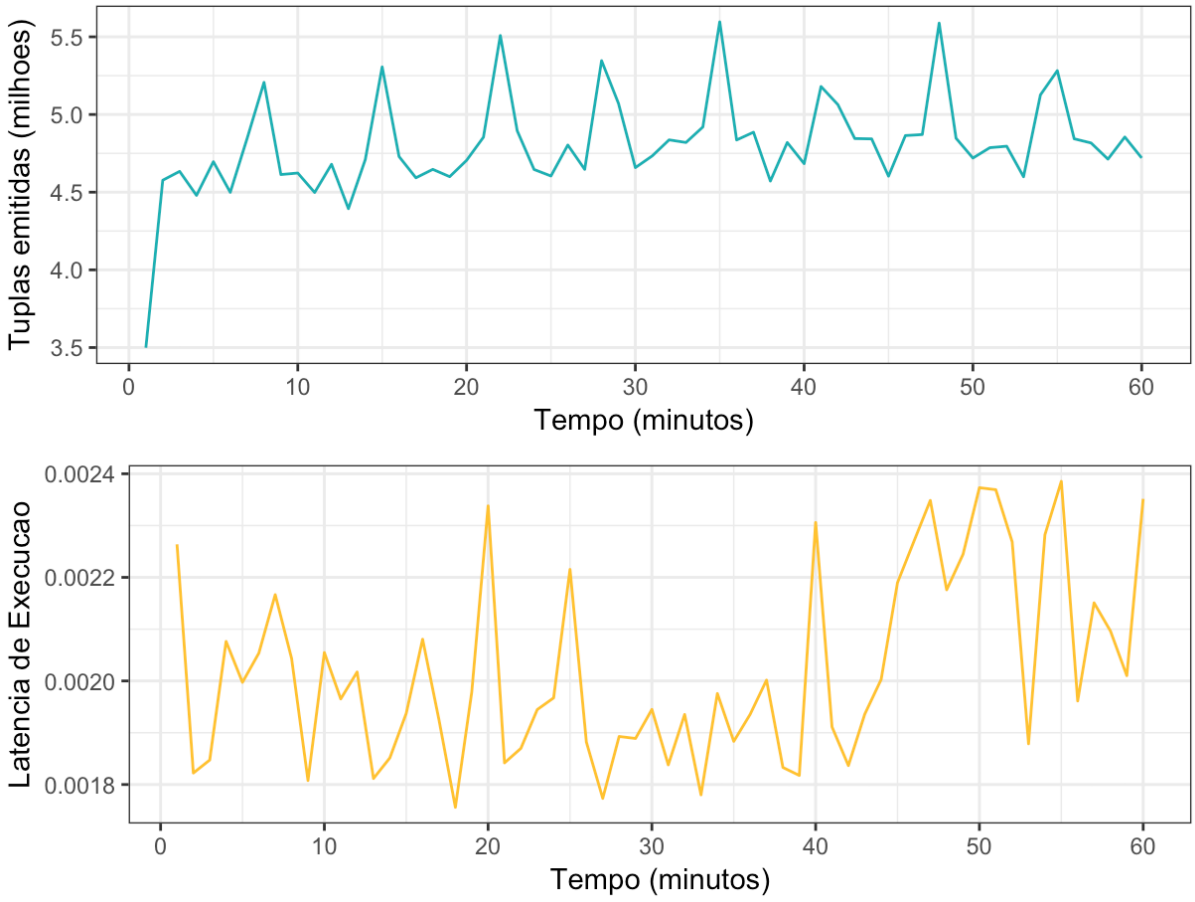

Figura 14 – Resultado do processamento de Máximo no Apache Storm: Taxa de transferência e latência

Fonte: Próprio Autor

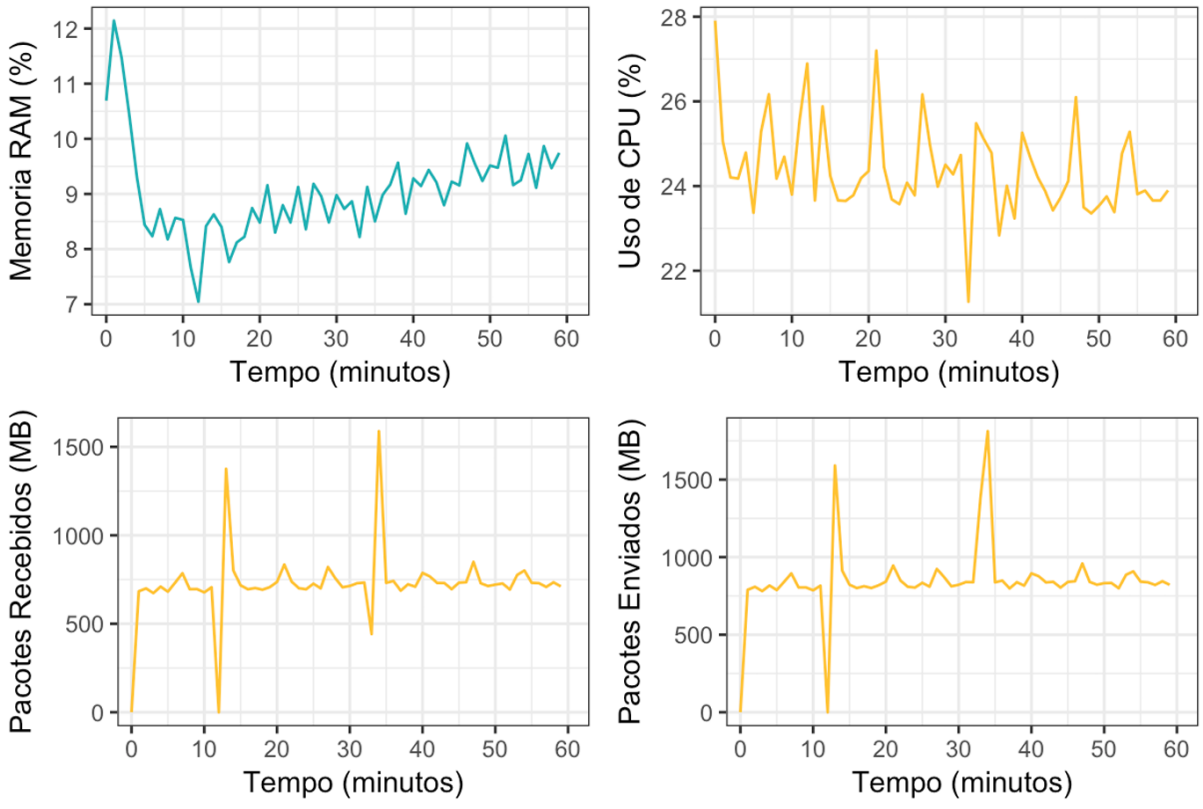

Figura 15 – Resultado do processamento de Máximo no Apache Storm: Taxa de transferência e latência

Fonte: Próprio Autor

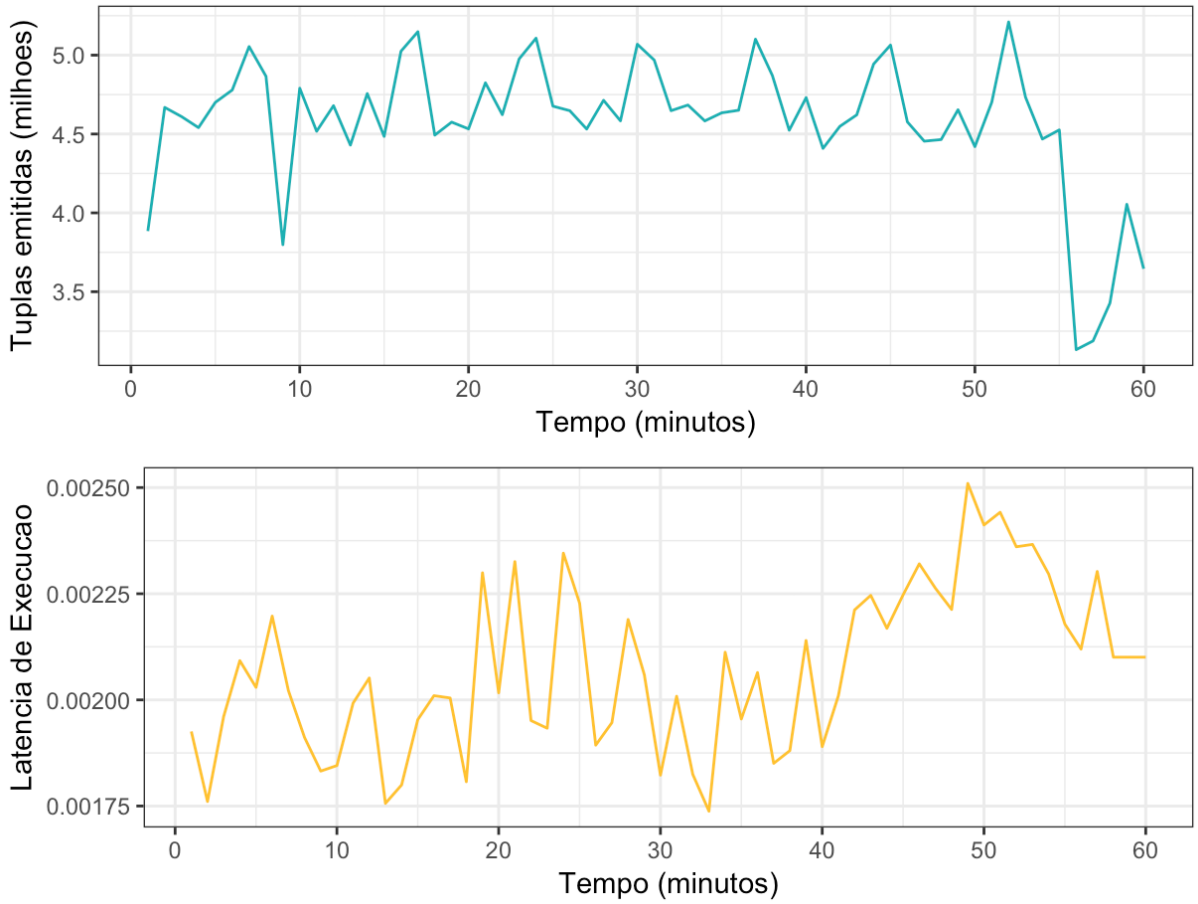

Figura 16 – Resultado do processamento de Mínimo no Apache Storm: Taxa de transferência e latência

Fonte: Próprio Autor

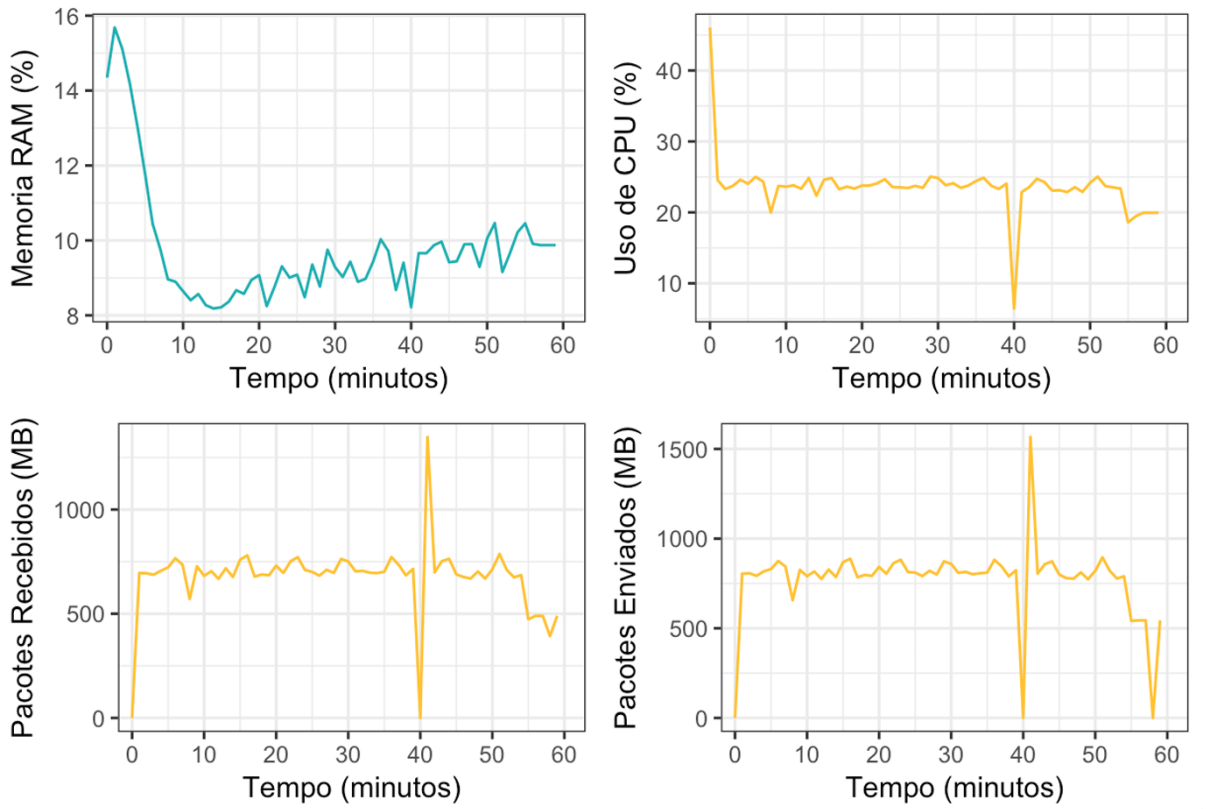

Figura 17 – Resultado do processamento de Mínimo no Apache Storm: Taxa de transferência e latência

Fonte: Próprio Autor

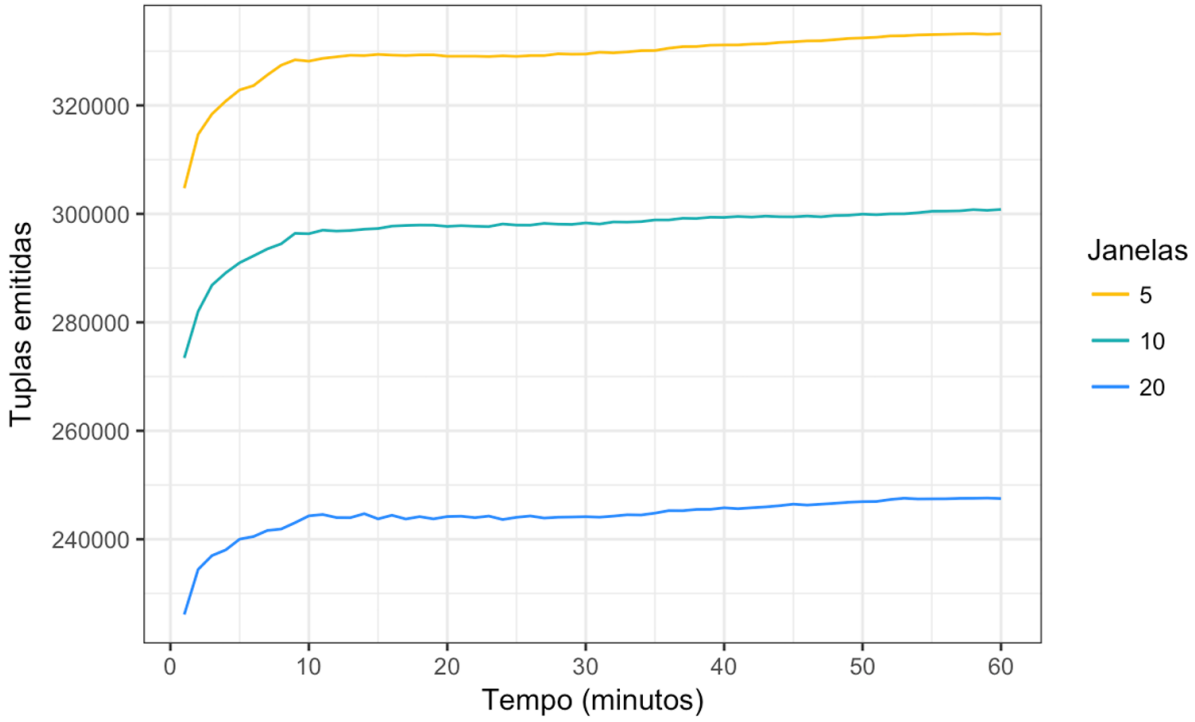

Figura 18 – Resultado do processamento de Média no Apache Edgent: Taxa de transferência e latência

Fonte: Próprio Autor

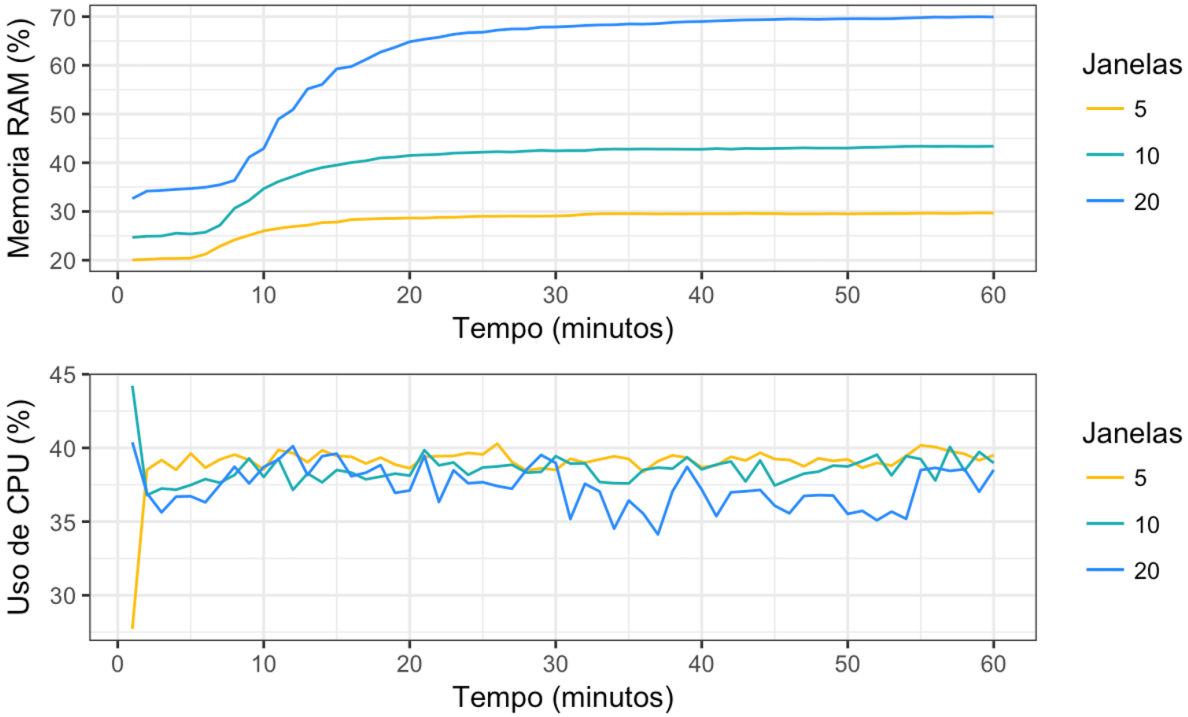

Figura 19 – Resultado do processamento de Média no Apache Edgent: Taxa de transferência e latência

Fonte: Próprio Autor

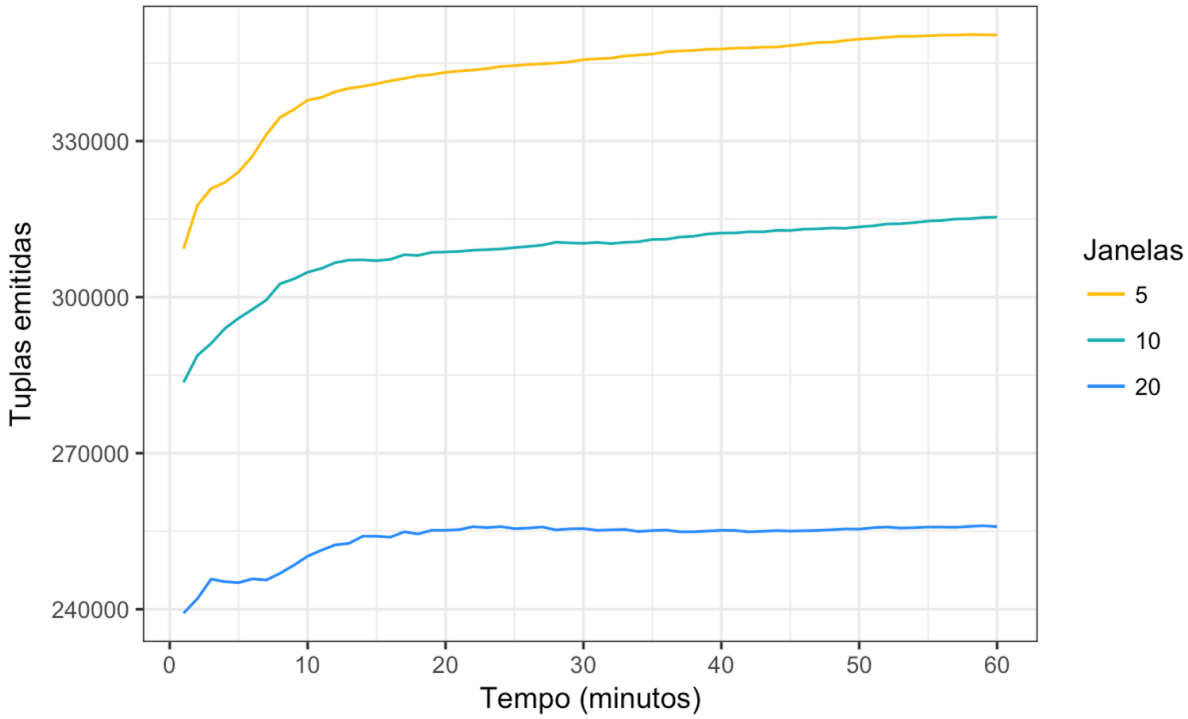

Figura 20 – Resultado do processamento de Contador no Apache Edgent: Taxa de transferência e latência

Fonte: Próprio Autor

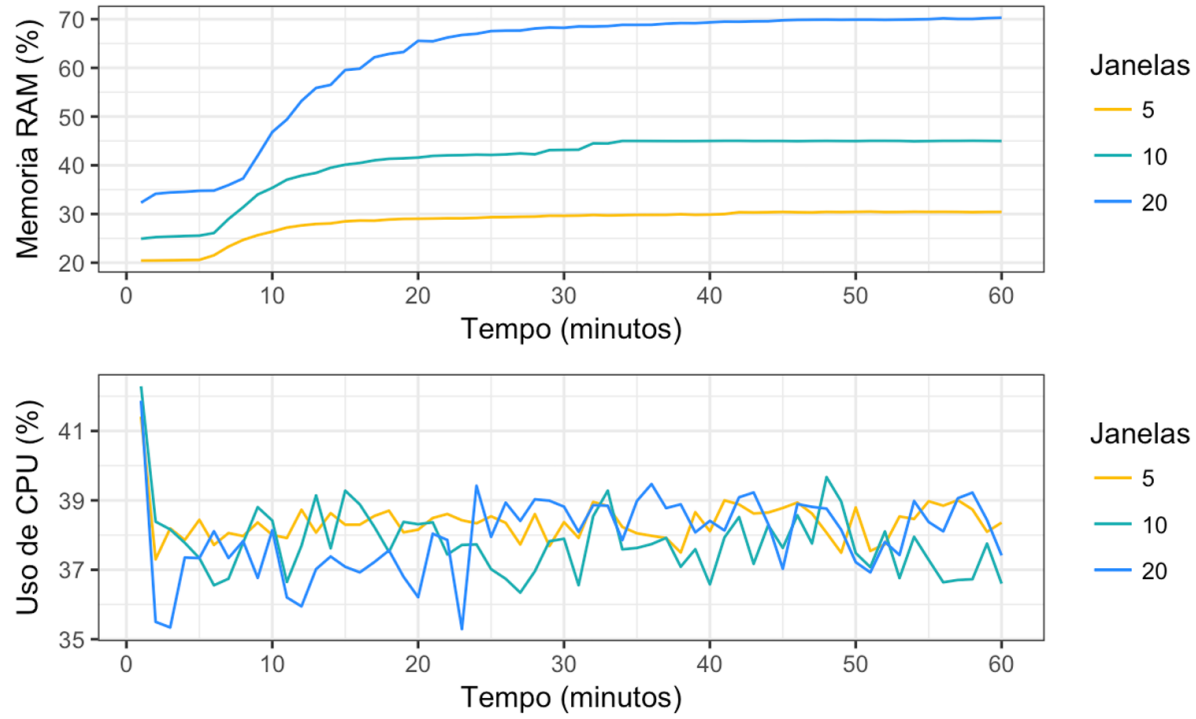

Figura 21 – Resultado do processamento de Contador no Apache Edgent: Taxa de transferência e latência

Fonte: Próprio Autor

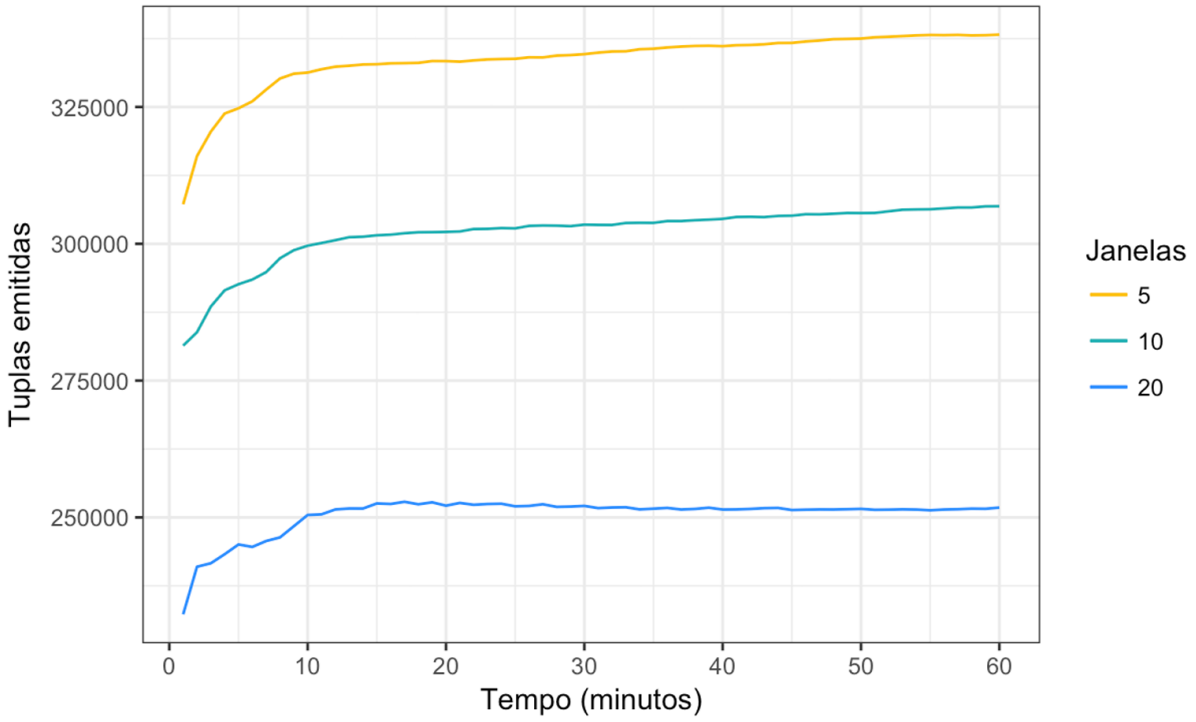

Figura 22 – Resultado do processamento de Máximo no Apache Edgent: Taxa de transferência e latência

Fonte: Próprio Autor

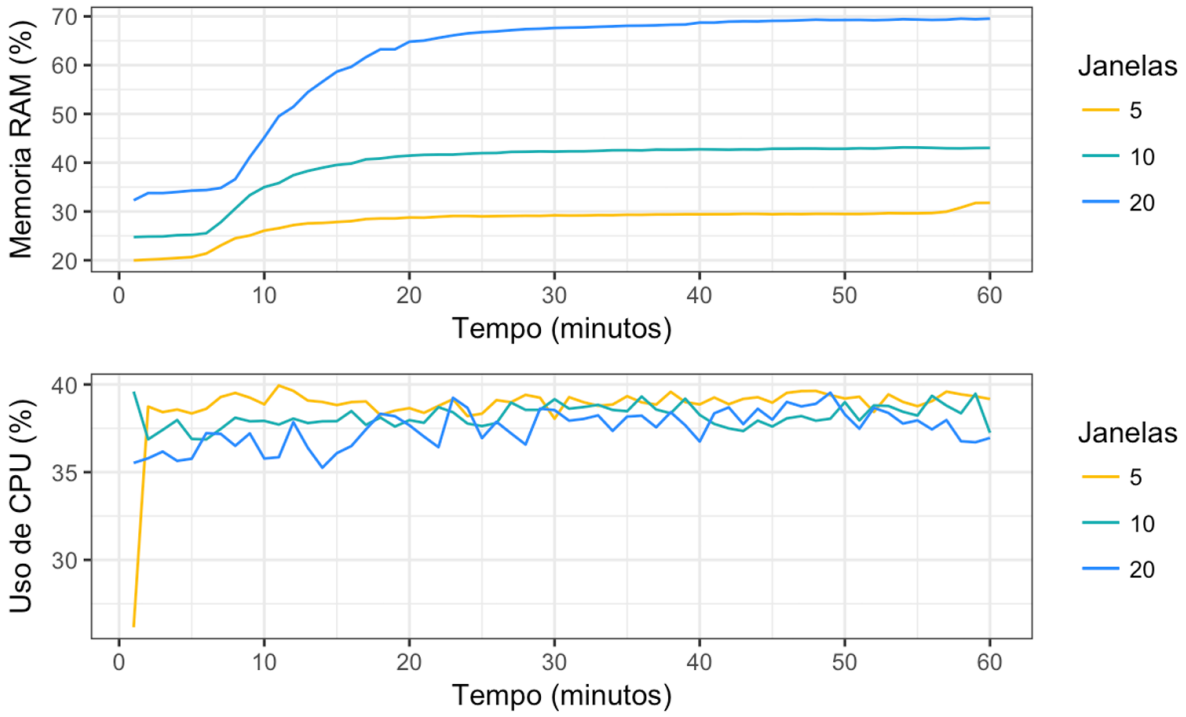

Figura 23 – Resultado do processamento de Máximo no Apache Edgent: Taxa de transferência e latência

Fonte: Próprio Autor

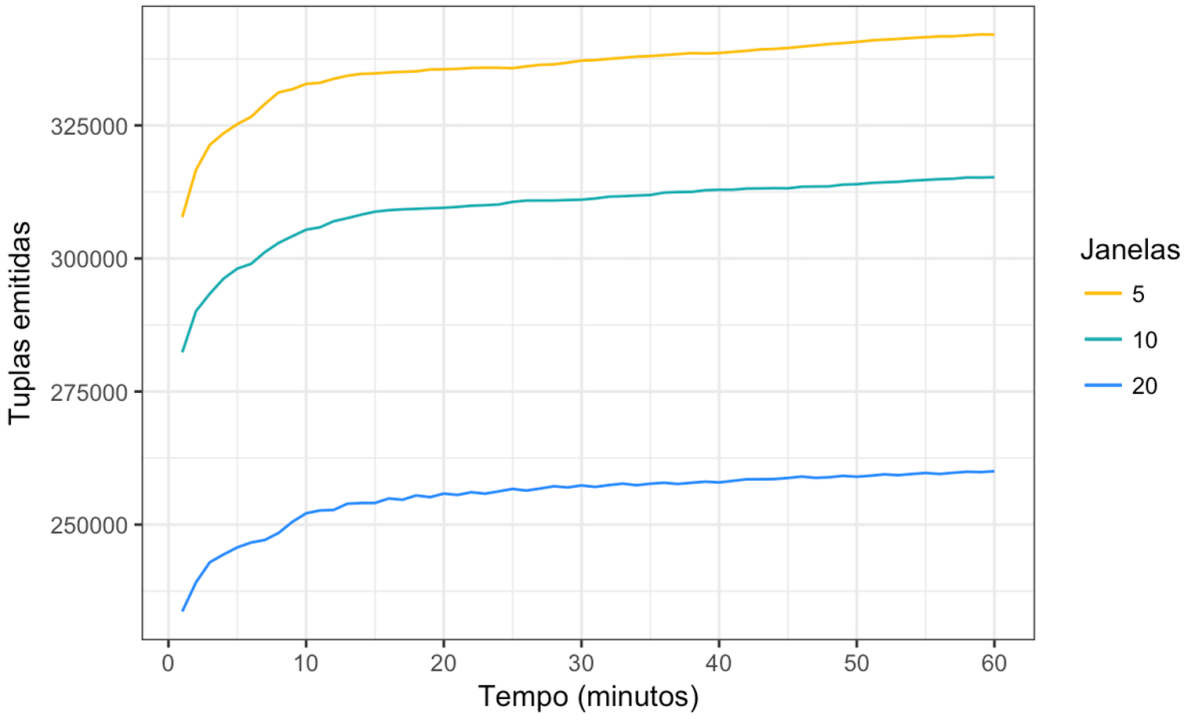

Figura 24 – Resultado do processamento do Mínimo no Apache Edgent: Taxa de transferência e latência

Fonte: Próprio Autor

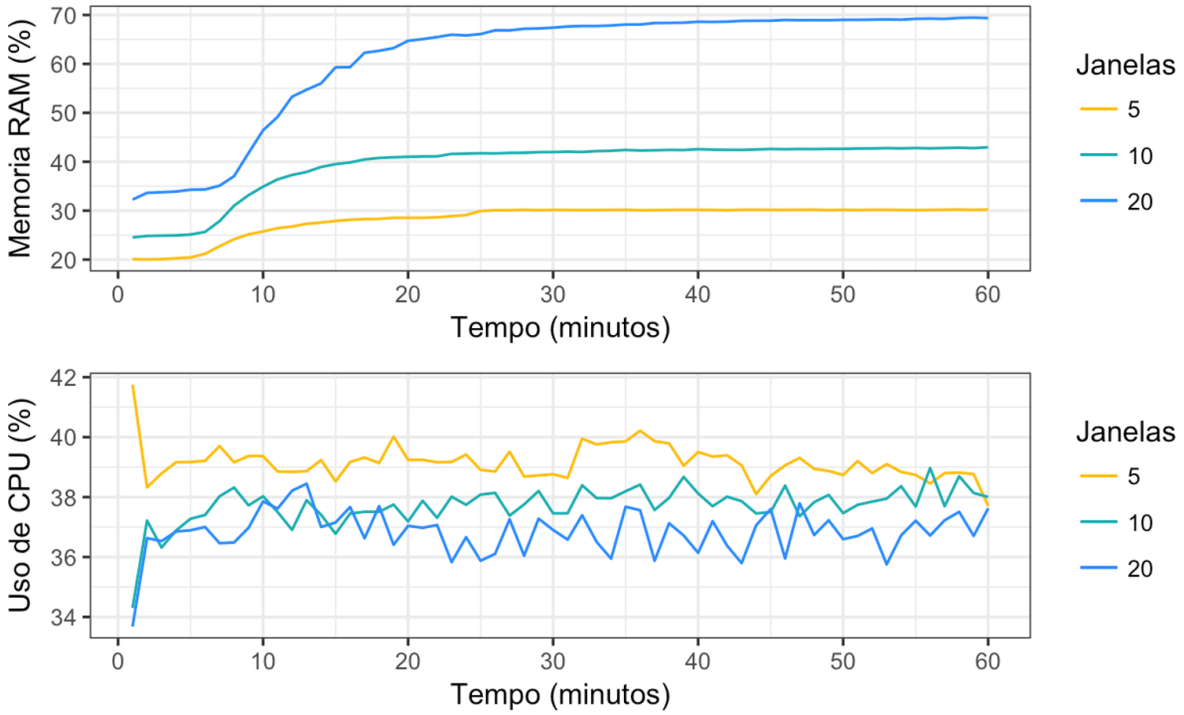

Figura 25 – Resultado do processamento do Mínimo no Apache Edgent: Taxa de transferência e latência

Fonte: Próprio Autor

4.8 DISCUSSÃO

Primeiramente cada caso do experimento foi replicado dez vezes, as informações apresentadas são um sumário dessas replicações. A partir de todos esses resultados acima listados, referentes a ambos os cenários do experimentos, algumas observações concluídas. Abaixo as observações serão listadas:

Apache Storm Diante do número médio de mensagens que a carga de trabalho envia durante a execução dos experimentos, que varia entre seis e sete milhões de mensagens por minuto, aliado os resultados obtidos referentes as métricas de sistema (Memória e CPU) é possível observar que os recursos computacionais empregados no *broker* de fila de mensagem do Apache Kafka foram insuficientes, causando um obstáculo no experimento. Essa informação foi confirmada ao extrair as informações de sistema da máquina onde o *broker* estava sendo executado.

A partir do item acima, também é possível observar que o Apache Storm suportaria ainda um maior volume de dados de entrada, já que parte dos recursos computacionais ficaram ociosos.

Os resultados apontam um baixo uso de memória em todos os serviços executados, variando sempre entre 8% e 15% do total disponível. Isso aponta que a carga trabalho poderia enviar uma maior quantidade de dados.

Em todos os serviços executados há uma alta taxa de transferência entre os nós, essa observação leva a considerar que realizar parte do processamento em um dispositivo de borda pode auxiliar na redução dos custos de comunicação.

A execução se manteve estável em todos os serviços e durante todo experimento, tendo o desvio padrão da taxa de transferência variando entre 268.986 e 429.137 mensagens e o coeficiente de variação entre 0,05 e 0,09. A baixa variação aponta uma estabilidade na execução do processamento.

Dificilmente será possível comparar os resultados obtidos com outros trabalhos da literatura, pois além das diferenças de configuração de ambiente a diversidade de versões do Apache Storm também é um fator. Por exemplo, em (TOSHNIWAL et al., 2014) a taxa de transferência que obtivemos em nosso experimento foi alcançada com aproximadamente entre sete e dez maquinas, pois é usado uma versão mais antiga do Apache Storm.

Apache Edgent O Apache Edgent funciona idealmente com janelas de tempo ou contagem mais curtas, já que estamos em um dispositivo de pequeno porte. Nos gráficos é perceptível que a cada aumento de janela todas as métricas de forma escalar sofrem alteração. Pois estamos lidando com um dispositivo com baixa capacidade, não suportando janelas de tempo maiores.

A carga de trabalho envia entre duzentas e trezentas mil mensagens por minuto, isso demostra que o Apache Edgent processou a maior parte dos dados recebidos, já que a taxa de transferência em todos os serviços ficou próximo a esse valor.

Fica perceptível que janelas de contagem maiores que 20 apresentaram problemas de desempenho, já que com essa configuração 70% da memória chegou a ser usada.

A execução também se manteve estável em todos os serviços e durante todo experimento, como demostrado abaixo nas seguintes janelas de contagem:

- 5: O desvio padrão da taxa de transferência variando entre 4.727 e 8.901 mensagens e o coeficiente de variação entre 0,014 e 0,018 .

- 10: O desvio padrão da taxa de transferência variando entre 4.602 e 6.693 mensagens e o coeficiente de variação entre 0,015 e 0,021.

- 20: O desvio padrão da taxa de transferência variando entre 3.490 e 5.391 mensagens e o coeficiente de variação entre 0,014 e 0,021.

4.9 CONSIDERAÇÕES FINAIS

Neste capítulo inicialmente foi definido o escopo do experimento. Baseado na proposta de (JAIN, 1991), foram descritos os objetivos a serem atingidos, métricas e demais parâmetros necessários para avaliação. Além disso, outros detalhes importantes, como a descrição da técnica de avaliação, caracterização da carga de trabalho e o projeto arquitetural do experimento. Por fim, foram apresentados os resultados obtidos. No próximo capítulo são apresentadas as considerações finais e contribuições acerca deste trabalho.

Por fim, é importante destacar que o cenário relatado no primeiro experimento, não foi um ambiente propriamente dito de computação em nuvem. Pois, algumas características existentes nesse tipo de ambiente não estavam presentes no cenário do experimento; a escalabilidade horizontal é uma dessas características, além de ser uma premissa em am-

biente de computação em nuvem. É possível dizer que um ambiente baseado em *cluster* foi utilizado e verificou-se como sendo satisfatório para a proposta, atendendo as exigências da avaliação de desempenho de SPFD. Ressaltamos que o conceito de computação em nuvem colaborou direcionando os caminhos escolhidos na construção deste trabalho, contribuindo fortemente para impulsionar os objetivos pretendidos.

5 CONCLUSÃO

Neste capítulo, são apresentadas as considerações finais sobre esta dissertação, assim como as contribuições que o trabalho pode trazer. Além disso, são relatadas as adequações que podem ser feitas para os trabalhos futuros.

O trabalho apresentado nessa dissertação, abordou os aspectos que envolvem o processamento de fluxo de dados, no qual foram identificados os conceitos que estão em volta dessa temática. A partir de toda contextualização e fundamentação teórica, seguimos para a elaboração de uma avaliação de desempenho, seguindo um roteiro bem planejado fornecido por (JAIN, 1991). Após esse processo de preparação da avaliação, uma série de experimentos foi executada, dados foram coletados, analisados e interpretados. Por fim, destacamos que o principal objetivo delineado para o trabalho foi alcançado, elaborar uma avaliação de desempenho voltada para mensurar o desempenho de SPFD em ambiente distribuído e local. Também a pergunta de pesquisa foi respondida durante a descrição dos resultados do Capítulo 4. A partir da disponibilização dos resultados, desejamos que essa avaliação seja capaz de servir como referência para esse domínio.

5.1 CONTRIBUIÇÕES

São apresentadas algumas contribuições desenvolvidas ao longo desde trabalho, como:

1. Uma avaliação de desempenho de SPFD em distintos cenários distribuídos.
2. Avanço no estado da arte a respeito do uso da Arquitetura Kappa.
3. A construção de uma carga de trabalho genérica que pode ser reutilizada em outros experimentos.
4. Concepção de um conjunto de dados referente a métricas, tanto do Apache Storm, como do Apache Edgent. Esses dados podem ser usados para ampliação da avaliação por outros trabalhos

5.2 TRABALHOS FUTUROS

A partir dos resultados obtidos e avaliando as limitações existentes neste trabalho, relataremos alguns itens para trabalhos futuros:

1. Usar a carga de trabalho para avaliação de outros SPFDs
2. Baseado no conceito de fatores de avaliação de desempenho, avaliar outros fatores pertinentes;

3. Realizar um novo experimento, no qual possa ser avaliado todo o cenário que foi dividido em dois experimentos. Desde a geração dos dados e pré processamento no dispositivo de borda até o processamento e armazenamento distribuído

REFERÊNCIAS

AAZAM, M.; KHAN, I.; ALSAFFAR, A. A.; HUH, E. N. Cloud of things: Integrating internet of things and cloud computing and the issues involved. In: *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014*. [S.l.: s.n.], 2014. p. 414–419. ISSN 2151-1403.

AGOSTINHO, S. *Confissões*. [S.l.: s.n.], 2014.

ASHTON, K. That "internet of things"thing. *RFiD Journal*, 1999.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, n. 15, p. 2787 – 2805, 2010. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610001568>.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. New York, NY, USA: ACM, 2012. (MCC '12), p. 13–16. ISBN 978-1-4503-1519-7. Disponível em: <http://doi.acm.org/10.1145/2342509.2342513>.

BOTTA, A.; DONATO, W. de; PERSICO, V.; PESCAPé, A. Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*, v. 56, p. 684 – 700, 2016. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X15003015>.

BRYSON, S.; KENWRIGHT, D.; COX, M.; ELLSWORTH, D.; HAIMES, R. Visually exploring gigabyte data sets in real time. *Commun. ACM*, ACM, New York, NY, USA, v. 42, n. 8, p. 82–90, ago. 1999. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/310930.310977>.

CARBONE, P.; KATSIFODIMOS, A.; EWEN, S.; MARKL, V.; HARIDI, S.; TZOUMAS, K. Apache flink™: Stream and batch processing in a single engine. *IEEE Data Eng. Bull.*, v. 38, p. 28–38, 2015.

CHENG, B.; LONGO, S.; CIRILLO, F.; BAUER, M.; KOVACS, E. Building a big data platform for smart cities: Experience and lessons from santander. In: *2015 IEEE International Congress on Big Data*. [S.l.: s.n.], 2015. p. 592–599. ISSN 2379-7703.

CHINTAPALLI, S.; DAGIT, D.; EVANS, B.; FARIVAR, R.; GRAVES, T.; HOLDERBAUGH, M.; LIU, Z.; NUSBAUM, K.; PATIL, K.; PENG, B. J.; POULOSKY, P. Benchmarking streaming computation engines: Storm, flink and spark streaming. In: *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. [S.l.: s.n.], 2016. p. 1789–1792.

COX, M.; ELLSWORTH, D. Application-controlled demand paging for out-of-core visualization. In: *Proceedings of the 8th Conference on Visualization '97*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997. (VIS '97), p. 235–ff. ISBN 1-58113-011-2. Disponível em: <http://dl.acm.org/citation.cfm?id=266989.267068>.

CUGOLA, G.; MARGARA, A. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 44, n. 3, p. 15:1–15:62, jun. 2012. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2187671.2187677>.

DÍAZ, M.; MARTÍN, C.; RUBIO, B. λ-coap: An internet of things and cloud computing integration based on the lambda architecture and coap. In: *Collaborative Computing: Networking, Applications, and Worksharing: 11th International Conference, CollaborateCom 2015, Wuhan, November 10-11, 2015, China. Proceedings*. Cham: Springer International Publishing, 2016. p. 195–206. ISBN 978-3-319-28910-6. Disponível em: <http://dx.doi.org/10.1007/978-3-319-28910-6_18>.

DÍAZ, M.; MARTÍN, C.; RUBIO, B. State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing. *Journal of Network and Computer Applications*, v. 67, p. 99 – 117, 2016. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S108480451600028X>.

EDGENT, A. *Apache Edgent: A Community for Accelerating Analytics at the Edge*. 2017. Disponível em: <http://edgent.apache.org/>.

FERNANDEZ, R. C.; PIETZUCH, P. R.; KREPS, J.; NARKHEDE, N.; RAO, J.; KOSHY, J.; LIN, D.; RICCOMINI, C.; WANG, G. Liquid: Unifying nearline and offline big data integration. In: *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. [s.n.], 2015. Disponível em: <http://www.cidrdb.org/cidr2015/Papers/CIDR15_Paper25u.pdf>.

FOUNDATION, R. P. *WHAT IS A RASPBERRY PI?* 2017. Disponível em: <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>.

GARTNER. *Gartner Says 8.4 Billion Connected "Things"Will Be in Use in 2017, Up 31 Percent From 2016*. 2017. Disponível em: <http://www.gartner.com/newsroom/id/3598917>.

HOORN, A. van; ROHR, M.; HASSELBRING, W. Generating probabilistic and intensity-varying workload for web-based software systems. In: *Proceedings of the SPEC International Workshop on Performance Evaluation: Metrics, Models and Benchmarks*. Berlin, Heidelberg: Springer-Verlag, 2008. (SIPEW '08), p. 124–143. ISBN 978-3-540-69813-5. Disponível em: <http://dx.doi.org/10.1007/978-3-540-69814-2_9>.

IBM; ZIKOPOULOS, P.; EATON, C. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. 1st. ed. [S.l.]: McGraw-Hill Osborne Media, 2011. ISBN 0071790535, 9780071790536.

JAIN, R. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: Wiley, 1991. I-XXVII, 1-685 p. (Wiley professional computing). ISBN 978-0-471-50336-1.

KALMESHWAR, M.; PRASAD, N. Internet of things: Architecture,issues and applications. *Journal of Engineering Research and Application*, v. 7, n. 6, p. 85–88, 2017.

KIRAN, M.; MURPHY, P.; MONGA, I.; DUGAN, J.; BAVEJA, S. S. Lambda architecture for cost-effective batch and speed big data processing. In: *Big Data (Big Data), 2015 IEEE International Conference on*. [S.l.: s.n.], 2015. p. 2785–2792.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007. Disponível em: <http://www.dur.ac.uk/ebse/resources/Systematic-reviews-5-8.pdf>.

KREPS, J. *Questioning the Lambda Architecture*. 2014. Disponível em: <https://www.oreilly.com/ideas/questioning-the-lambda-architecture>.

KULKARNI, S.; BHAGAT, N.; FU, M.; KEDIGEHALLI, V.; KELLOGG, C.; MITTAL, S.; PATEL, J. M.; RAMASAMY, K.; TANEJA, S. Twitter heron: Stream processing at scale. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2015. (SIGMOD '15), p. 239–250. ISBN 978-1-4503-2758-9. Disponível em: <http://doi.acm.org/10.1145/2723372.2742788>.

LEE, I.; LEE, K. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, v. 58, n. 4, p. 431–440, 2015. Disponível em: <http://EconPapers.repec.org/RePEc:eee:bushor:v:58:y:2015:i:4:p:431-440>.

LIU, G.; ZHU, W.; SAUNDERS, C.; GAO, F.; YU, Y. Complex adaptive systems san jose, ca november 2-4, 2015 real-time complex event processing and analytics for smart grid. *Procedia Computer Science*, v. 61, p. 113 – 119, 2015. ISSN 1877-0509. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050915029993>.

LOPEZ, M. A.; LOBATO, A. G. P.; DUARTE, O. C. M. B. A performance comparison of open-source stream processing platforms. In: *2016 IEEE Global Communications Conference (GLOBECOM)*. [S.l.: s.n.], 2016. p. 1–6.

LU, R.; WU, G.; XIE, B.; HU, J. Stream bench: Towards benchmarking modern distributed stream computing frameworks. In: *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. Washington, DC, USA: IEEE Computer Society, 2014. (UCC '14), p. 69–78. ISBN 978-1-4799-7881-6. Disponível em: <http://dx.doi.org/10.1109/UCC.2014.15>.

MARTÍNEZ-PRIETO, M. A.; CUESTA, C. E.; ARIAS, M.; FERNÁNDEZ, J. D. The solid architecture for real-time management of big semantic data. *Future Generation Computer Systems*, v. 47, p. 62 – 79, 2015. ISSN 0167-739X. Special Section: Advanced Architectures for the Future Generation of Software-Intensive Systems. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X1400209X>.

MARZ, J. W. N. *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications, aout, 2013. ISBN 978-1617290343. Disponível em: <http://opac.inria.fr/record=b1134858>.

MIORANDI, D.; SICARI, S.; PELLEGRINI, F. D.; CHLAMTAC, I. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, v. 10, n. 7, p. 1497 – 1516, 2012. ISSN 1570-8705. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1570870512000674>.

MONTGOMERY, D. C. *Applied Statistics and Probability for Engineers*. 5st. ed. [S.l.]: Wiley, 2010. ISBN 0470053046, 978-0470053041.

PATEL, P.; PATHAK, A.; TEIXEIRA, T.; ISSARNY, V. Towards application development for the internet of things. In: *Proceedings of the 8th Middleware Doctoral Symposium*. New York, NY, USA: ACM, 2011. (MDS '11), p. 5:1–5:6. ISBN 978-1-4503-1072-7. Disponível em: <http://doi.acm.org/10.1145/2093190.2093195>.

PATRICK, F. P. G. Internet of things strategic research roadmap. *The Cluster of European Research Projects*, 2009. Disponível em: <http://www.internet-of-things-research.eu/pdf/IoTClusterStrategicResearchAgenda2009.pdf>.

PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 414–454, First 2014. ISSN 1553-877X.

PREUVENEERS, D.; BERBERS, Y.; JOOSEN, W. Samurai: A batch and streaming context architecture for large-scale intelligent applications and environments. *Journal of Ambient Intelligence and Smart Environments*, v. 8, n. 1, p. 63–78, January 2016. ISSN 1876-1364. Disponível em: <https://lirias.kuleuven.be/handle/123456789/525543>.

QIAN, S.; WU, G.; HUANG, J.; DAS, T. Benchmarking modern distributed streaming platforms. In: *2016 IEEE International Conference on Industrial Technology (ICIT)*. [S.l.: s.n.], 2016. p. 592–598.

REED, D.; LARUS, J. R.; GANNON, D. Imagining the future: Thoughts on computing. *Computer*, v. 45, n. 1, p. 25–30, Jan 2012. ISSN 0018-9162.

SAHA, D.; MUKHERJEE, A. Pervasive computing: A paradigm for the 21st century. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 36, n. 3, p. 25–31, mar. 2003. ISSN 0018-9162. Disponível em: <http://dx.doi.org/10.1109/MC.2003.1185214>.

SHUKLA, A.; CHATURVEDI, S.; SIMMHAN, Y. Riotbench: A real-time iot benchmark for distributed stream processing platforms. *CoRR*, abs/1701.08530, 2017. Disponível em: <http://arxiv.org/abs/1701.08530>.

SHUKLA, A.; SIMMHAN, Y. Benchmarking distributed stream processing platforms for iot applications. *CoRR*, abs/1606.07621, 2016. Disponível em: <http://arxiv.org/abs/1606.07621>.

TOSHNIWAL, A.; TANEJA, S.; SHUKLA, A.; RAMASAMY, K.; PATEL, J. M.; KULKARNI, S.; JACKSON, J.; GADE, K.; FU, M.; DONHAM, J.; BHAGAT, N.; MITTAL, S.; RYABOY, D. Storm@twitter. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2014. (SIGMOD '14), p. 147–156. ISBN 978-1-4503-2376-5. Disponível em: <http://doi.acm.org/10.1145/2588555.2595641>.

VILLARI, M.; CELESTI, A.; FAZIO, M.; PULIAFITO, A. Alljoyn lambda: An architecture for the management of smart environments in iot. In: *Smart Computing Workshops (SMARTCOMP Workshops), 2014 International Conference on*. [S.l.: s.n.], 2014. p. 9–14.

WAINER, J. et al. Métodos de pesquisa quantitativa e qualitativa para a ciência da computação. *Atualização em informática*, v. 1, p. 221–262, 2007.

WAZLAWICK, R. S. *Metodologia de Pesquisa Para Ciência Da Computação*. 2st. ed. [S.l.]: Elsevier Editora Ltda., 2014. ISBN 978-85-352-7782-1.

WEISER, M. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 3, n. 3, p. 3–11, jul. 1999. ISSN 1559-1662. Disponível em: <http://doi.acm.org/10.1145/329124.329126>.

ZAHARIA, M.; DAS, T.; LI, H.; SHENKER, S.; STOICA, I. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. In: *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing*. Berkeley, CA, USA: USENIX Association, 2012. (HotCloud'12), p. 10–10. Disponível em: <http://dl.acm.org/citation.cfm?id=2342763.2342773>.

ZASLAVSKY, A. B.; PERERA, C.; GEORGAKOPOULOS, D. Sensing as a service and big data. *CoRR*, abs/1301.0159, 2013. Disponível em: <http://arxiv.org/abs/1301.0159>.

ZSCHöRNIG, T.; WEHLITZ, R.; FRANCZYK, B. A personal analytics platform for the internet of things - implementing kappa architecture with microservice-based stream processing. In: INSTICC. *Proceedings of the 19th International Conference on Enterprise Information Systems - Volume 2: ICEIS,*. [S.l.]: SciTePress, 2017. p. 733–738. ISBN 978-989-758-248-6.

APÊNDICE A – AMOSTRA DO CONJUNTO DE DADOS DE ENTRADA

Identificador	Hora	Placa	Latitude	Longitude	Velocidade
57899	17:53:21	BL0978	114.138008	22.552073	37.0
57899	12:24:46	BL0978	114.137032	22.551296	0.0
57899	21:00:17	BL0978	114.170593	22.566845	0.0
57899	14:30:48	BL0978	114.112213	22.585052	0.0
57899	08:02:28	BL0978	114.157463	22.564417	38.0
57899	18:52:56	BL0978	114.11277	22.584627	37.0
57899	16:52:33	BL0978	114.170502	22.566803	0.0
57899	09:23:23	BL0978	114.170502	22.566797	16.0
57899	08:01:29	BL0978	114.158577	22.560957	37.0
57899	08:04:52	BL0978	114.16362	22.566784	31.0
57899	09:37:26	BL0978	114.170868	22.566782	0.0
57899	14:54:26	BL0978	114.147308	22.555546	44.0
57899	18:56:01	BL0978	114.11763	22.569757	40.0
57899	20:52:58	BL0978	114.170219	22.566809	12.0
57899	22:13:58	BL0978	114.170403	22.566843	0.0
57899	09:24:19	BL0978	114.170959	22.562414	38.0
57899	15:33:53	BL0978	114.148491	22.559801	27.0

APÊNDICE B – AMOSTRA DOS DADOS DAS MÉTRICAS (APACHE EDGENT)

Time	Count	meanRate	m1Rate	m5Rate	m15Rate	rateUnit
1503699924	307254	5178.18	4062.06	2693.55	2364.30	events/second
1503699984	632078	5341.47	4934.84	3193.41	2563.45	events/second
1503700044	961501	5389.40	5322.37	3618.83	2755.38	events/second
1503700104	1295249	5427.86	5465.18	3965.46	2934.19	events/second
1503700164	1623816	5441.39	5499.20	4245.14	3100.114	events/second
1503700224	1956212	5458.23	5532.31	4478.21	3256.65	events/second

Time	Ram %	CPU %
1503699924	19.9	26.1
1503699984	20.1	38.7
1503700044	20.24	38.4
1503700104	20.4	38.5
1503700164	20.6	38.3
1503700224	21.3	38.6

APÊNDICE C – AMOSTRA DOS DADOS DAS MÉTRICAS (APACHE STORM)

TimeStamp	WorkerHost	TaskId	Bolt	Info	Valor	Ram %	CPU %	Recebido	Enviado
10:02:18,144	1489150939	host:6701	spout-kafka-operation	execute count	2347324	13.2	25.2	679787.8	301958.22
10:02:18,145	1489150939	host:6701	spout-kafka-operation	execute-latency	0.001627387829524	-	-	-	-
10:03:18,145	1489150939	host:6701	spout-kafka-operation	execute count	2339214	13.1	24.9	675486.06	304853.44
10:03:18,146	1489150939	host:6701	spout-kafka-operation	execute-latency	0.0833333333333	-	-	-	-
10:04:18,149	1489150939	host:6701	spout-kafka-operation	execute count	2425811	12.4	25.5	701046.56	314800.44

APÊNDICE D – SUMÁRIO DOS RESULTADOS COLETADOS

Tabela 7 – Sumário das métricas de desempenho do Apache Storm

Apache Storm				
		Transferência	L. Exec (ms)	L. Comp (ms)
Count	Max	5.659.412	0,0021	760,75
	Min	4.328.383	0,0015	573
	Mean	4.885.641	0,0017	678,993
	Sum	298.024.103	0,1088	41418
	SD	268.986	0,0001	39,653
Avg	Max	5.494.727	0,0027	761
	Min	3.086.326	0,0016	444
	Mean	4.712.941	0,0020	666,244
	Sum	282.776.460	0,1255	39974
	SD	355.650	0,0002	52,58
Max	Max	5.596.479	0,0023	752,5
	Min	3.498.518	0,0017	554,5
	Mean	4.803.990	0,0020	683,70
	Sum	288.239.425	0,1213	41022
	SD	315.688	0,0001	43,975
Min	Max	5.210.603	0,0025	965,5
	Min	3.132.533	0,0017	477
	Mean	4.578.726	0,0020	698.98
	Sum	274.723.595	0,1243	41939
	SD	429.137	0,0001	59,70

Tabela 8 – Sumário dos dados referente a métricas de sistema do Apache Storm

Apache Storm				
		CPU %	Ram %	Enviados/Recebidos
Count	Max	24.8%	12.2%	271 / 726 (MB)
	Min	18.5%	7.3%	0 / 0 (MB)
	Mean	20.4%	8.6%	230 / 618 (MB)
	Sum	-	-	14.039 / 37.751 (MB)
	SD	1.12%	1,0%	32,5/ 87,6 (MB)
Avg	Max	31,6%	12,17%	1.691 / 1.431 (MB)
	Min	20,7%	7,41%	0 / 0 (MB)
	Mean	23,8%	9,0%	808.6 / 792.1 (MB)
	Sum	-	-	48.521 / 42.130 (MB)
	SD	1,63%	0,90%	242,5 / 210,5 (MB)
Max	Max	27,9%	12,14%	1.812 / 1.588 (MB)
	Min	21,1%	7,04%	0 / 0 (MB)
	Mean	24,3%	9,0%	846,9 / 724,3 (MB)
	Sum	-	-	50.814 / 43.458 (MB)
	SD	1,1%	0,86%	236,3 / 200,9 (MB)
Min	Max	46,0%	15,6%	1.566 / 1.348 (MB)
	Min	6,4%	8,1%	0 / 0 (MB)
	Mean	23,6%	9,73%	770,7 / 676,5 (MB)
	Sum	-	-	46.242 / 40.591 (MB)
	SD	3,9%	1,64%	218,5 / 170,8 (MB)