Uma plataforma de IoT para integração de dispositivos baseada em nuvem com Apache Kafka

Adriano B. de Sousa, José R. Torres Neto, Geraldo P. R. Filho, Jó Ueyama

[1]Instituto de Ciência Matemáticas e de Computação – ICMC
Universidade de São Paulo – USP
13566 – 590 – São Carlos – SP – Brasil

{adriano.belfort.sousa, jrtorresneto}@usp.br
{geraldop, joueyama}@icmc.usp.br

Abstract. *Billions of devices connected to the Internet generate a large amount of data every day, which they serve as the basis for applications on a global scale. The increasing demand for these applications creates some challenges in the Internet of Things ecosystem, e.g. data integration and devices heterogeneity. Solutions on cloud-based platforms have been increasingly used to address such issues. However, a robust infrastructure for storing and processing large amounts of run-time data is required. This work presents a cloud IoT platform capable of handling with the integration of a large amounts of data and heterogeneity of devices based on Apache Kafka. The main contributions of this work are: i) the specification, implementation and validation of an IoT platform in the cloud and; ii) design and development of a generic interface based on webservice for abstraction of details regarding the platform with the devices. With integration of the platform and the interface developed, it was possible to obtain satisfactory results in the standardization of the communication and in the flow and storage of reliable data, offering rates of fast transfers in Mb/s.*

Resumo. *Bilhões de dispositivos conectados à Internet geram um grande volume de dados diversos diariamente, os quais servem como base para aplicações em escala global. A demanda crescente dessas aplicações geram alguns desafios no ecossistema da Internet das Coisas, e.g. integração de dados e heterogeneidade de dispositivos. Soluções em plataformas baseadas na nuvem têm sido cada vez mais utilizadas para sanar tais desafios. Entretanto, uma infraestrutura robusta para o armazenamento e o processamento de grandes quantidades de dados em tempo de execução é requerida. Este trabalho apresenta uma plataforma de IoT em nuvem capaz de lidar com a integração de grandes quantidades de dados e heterogeneidade dos dispositivos com base no Apache Kafka. As principais contribuições deste trabalho são: i) especificação, implantação e validação de uma plataforma IoT em nuvem e; ii) design e desenvolvimento de uma interface genérica baseada em webservice para abstração de detalhes a respeito da plataforma com os dispositivos. Com integração da plataforma e a interface desenvolvida foi possível obter resultados satisfatórios na padronização da comunicação e no fluxo e armazenamento de dados de maneira confiável, oferecendo taxas de transferências rápidas em MB/s.*

1. Introdução

Nos últimos anos, os avanços na área de micro sistemas eletro-mecânicos auxiliaram na disseminação do paradigma da *Internet of Thing* (IoT). Na IoT, as “coisas" são dispositivos capazes de monitorar e disseminar dados do ambiente para, em seguida processá-los e interconectá-los nas “coisas" por meio de uma estrutura de rede dinâmica e global [Botta et al. 2014]. Segundo a consultoria [Gartner 2017], a projeção para o ano de 2017 foi de 8,4 bilhões de dispositivos IoT conectados e em uso no mundo. Em razão disso, diversos serviços estão sendo desenvolvidos no contexto de IoT, tais como automação residencial, e-health e monitoramento urbano.

Mesmo com um crescimento quantitativo dos dispositivos, as soluções na IoT não conseguem ter a capacidade computacional necessária para serviços modernos ubíquos e pervasivos. Isso ocorre pois tais dispositivos são caracterizados por possuir recursos computacionais limitados (i.e., processamento, armazenamento e comunicação). Nesse sentido, a computação em nuvem [Zhou and Buyya 2018], que disponibiliza recursos em *data centers*, acessíveis a partir de qualquer lugar, compensa tais restrições, fornecendo não somente recursos virtualmente ilimitados de processamento, rede e armazenamento, como também mecanismos e serviços de gerência para os dispositivos e para os dados que eles geram. Adicionalmente, a inclusão de dispositivos reais às plataformas na nuvem aumentam o escopo e alcance no desenvolvimento de novas soluções para os problemas do mundo real [Botta et al. 2014].

Vale frisar que devido as restrições computacionais apresentadas nos dispositivos, o desenvolvimento de serviços inteligentes na IoT que utilizam a intercomunicação entre os dispositivos não é uma tarefa trivial. Além disso, a heterogeneidade dos dispositivos dificulta consideravelmente a comunicação e a integração entre eles, tornando um desafio mais complexo no desenvolvimento de serviços na IoT de acordo com a escala do problema a se resolver. Para sanar tais problemas, várias soluções [Kreps et al. 2011a, Triawan et al. 2016, Wiska et al. 2016] utilizam *middlewares* para lidar com a integração dos dados e com a heterogeneidade dos dispositivos. Apesar da utilização de *middlewares* ser promissora, ainda há diversos desafios para serem resolvidos, sendo um deles a especificação de uma interface padrão para a conexão de dispositivos, sensores e aplicações heterogêneos para o desenvolvimento de serviços inteligentes.

Para preencher a lacuna mencionada anteriormente, este trabalho apresenta uma plataforma de IoT em nuvem capaz de lidar com a integração dos dados e heterogeneidade dos dispositivos com base no Apache Kafka [Kreps et al. 2011b]. Para isso, a comunicação entre os dispositivos e o Kafka é intermediada e padronizada por uma interface de comunicação genérica, a qual é baseada em *webservice* REST que abstrai os detalhes a respeito da plataforma com os dispositivos. A interface implementada permite a comunicação de variados tipos e cargas de dados de maneira confiável. A plataforma, por sua vez, é responsável por armazenar as mensagens enviadas, disseminando-as para as aplicações consumidoras. Com isso, a plataforma permite o uso de aplicações em tempo real para um grande volume de dados, garantindo a redução de latência com alta transmissão de. Vale salientar que como prova de conceito, a plataforma foi implantada e validada na plataforma de nuvem da Google.

O restante deste artigo está organizado da seguinte maneira. A Seção 2 apresenta

os trabalhos da literatura relacionados com esta proposta. A Seção 3 descreve a solução proposta. Uma avaliação de desempenho da plataforma proposta é apresentada na Seção 4. Finalmente, a Seção 5 apresenta as considerações finais e os trabalhos futuros.

2. Trabalhos Relacionados

O trabalho original sobre Kafka proposto por Kreps et al. [Kreps et al. 2011a] usa um agregador de *logs* gerados por várias instâncias de aplicações utilizadas por usuários. Esses *logs* são armazenados no Kafka e encaminhados para serviços de análise de tempo real. *Clusters* Kafka auxiliares são utilizados para o *offload* dos *logs* para o Apache Hadoop. Diferente da proposta anterior, este trabalho estende o Apache Kafka com uma interface em integração com a nuvem.

Os autores em [Triawan et al. 2016] propuseram o uso do *middleware publish/subscribe* Mosquitto, que implementa o protocolo *Message Queue Telemetry Transport* (MQTT) hospedado na nuvem para uma aplicação de sistema hidropônico controlável pela Internet com sensores e atuadores. As taxas de transmissão conseguidas, especialmente para o recebimento de dados do *middleware*, são de ordem inferior a 100KBps, tal velocidade pode ser um empecilho para aplicações de tempo real. Wiska et al. [Wiska et al. 2016] usa uma rede de sensores sem fio ou WSN é amparada pelo Kafka para a ingestão e armazenamento grandes quantidades de dados de monitoramento de CO_2. Os testes conduzidos pelo estudo mostraram que a integração do Kafka à rede de sensores permite o processamento em tempo quase real com baixa latência. Vale frisar entretanto que não há uma padroniza na comunicação, tendo que configurar cada dispositivo com a especificação do Kafka.

Este trabalho se diferencia dos trabalhos apresentados na especificação de uma interface padrão para a conexão de dispositivos, sensores e aplicações heterogêneos à plataforma e no desenvolvimento de uma plataforma robusta o suficiente para permitir o seu uso por aplicações de tempo real com grande volumes de dados, garantindo a latência e taxas de transmissão, requisitos cruciais em ambientes IoT, os quais são compostos por dispositivos de computação escassa.

3. Uma plataforma de IoT para integração de dispositivos

Esta seção apresenta a especificação, implementação e implantação de uma plataforma IoT em nuvem capaz de lidar com a integração dos dados e heterogeneidade dos dispositivos baseada no Apache Kafka. A comunicação entre os dispositivos e a plataforma é intermediada e padronizada por uma camada de apresentação genérica baseada em *webservice* REST, a qual abstrai os detalhes a respeito da plataforma dos dispositivos. Tal interface permite a integração de variados tipos e cargas de dados entre diferentes dispositivos de maneira confiável na plataforma, a qual é responsável pelo armazenamento das mensagens transmitidas e sua disseminação às aplicações consumidoras.

A Figura 1 apresenta a visão geral da plataforma proposta, na qual os dispositivos são produtores de informações que geram dados de eventos e os enviam para o *webservice* via HTTP. A facilidade para gerar requisições HTTP em diversos dispositivos reduz o impacto negativo da heterogeneidade dos sistemas IoT para o envio de informações para as aplicações. A principal função do *webservice* é enviar as cargas de dados dos dispositivos

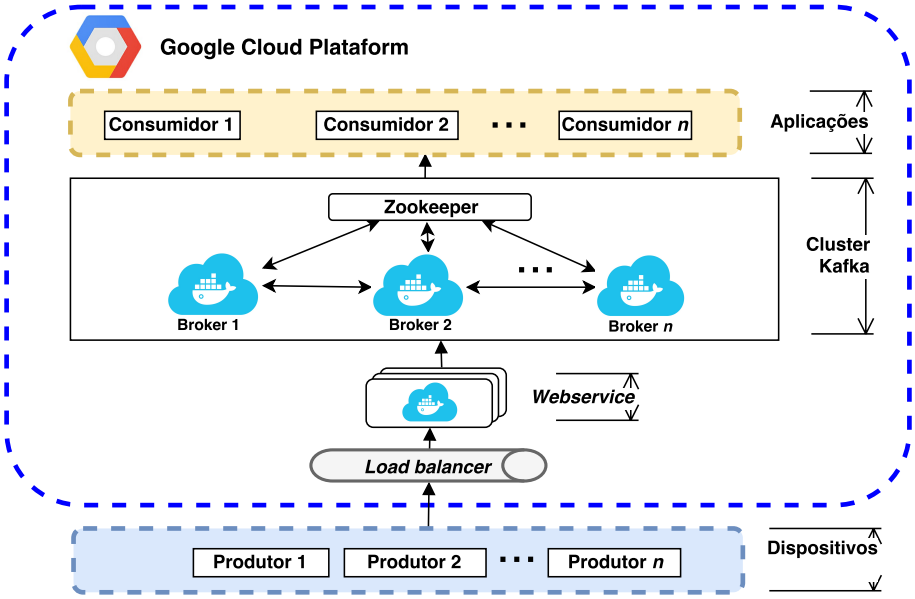

Figura 1. Visão geral da plataforma proposta

aos clusteres do Kafka por meio de quatro componentes, como ilustrado no diagrama da Figura 3: API, serviços, configuração e entidades.

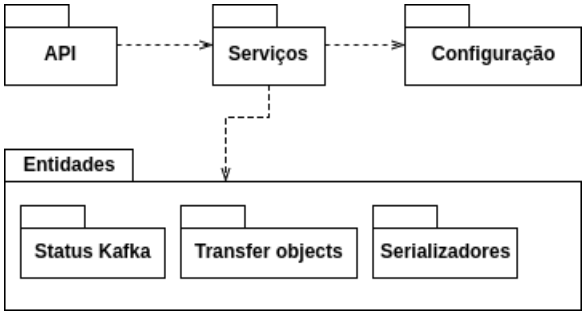

Figura 2. Diagrama dos componentes do *webservice*

O *webservice* está sempre disponível para o fluxo das informações, sendo assim faz-se necessário o design da arquitetura de maneira distribuída com múltiplas instâncias do *webservice*. Com isso, foi possível estabelecer um intermediário, localizado na fronteira entre a Internet e a rede interna da *Google Cloud Platform*(GCP) [1], responsável por encaminhar as requisições dos produtores às várias instâncias do *webservice* balanceando a carga de trabalho. O uso do *load balancer* permite o aumento da escalabilidade do *webservice* possibilitando a qualquer momento a adição e/ou remoção de novas e/ou subtilizadas instâncias.

Após o balanceamento da carga de trabalho pelo *load balancer* as mensagens trafegam pela rede interna, denominada conjunto de servidores. Os conjuntos de servidores recebem as mensagens no padrão exigido pelo Kafka, uma vez que são interpretadas e convertidas pela instância do *webservice*. Neste cenário, as instâncias do *webservice* atuam como publicadores (*publishers*) de conteúdo diretamente aos *brokers* do Kafka.

[1] https://cloud.google.com/

Ainda, o *Zookeeper* foi utilizado para o gerenciamento dos tópicos do Kafka e o monitoramento do estado dos conjuntos de servidores. Os elementos da plataforma são executados em contêineres Docker, que possui seu ciclo de vida gerenciado pelo Kubernetes, o qual reduz o tempo de configuração para projetos de aplicações conteinerizadas.

Apesar do grande poder do Apache Kafka, seu uso direto por dispositivos IoT é dificultado por dois fatores: (i) a linguagem utilizada que implementa a sua API padrão; e (ii) o conhecimento de informações específicas (parâmetros) para se comunicarem diretamente com ele, tais como o tópico, chave da mensagem, informação de serialização e confirmação de recebimento. Com esses problemas em mente, foi necessário a implementação de uma camada de apresentação que abstraiu tais complexidades, gerando uma interface padronizada e simples para que dispositivos heterogêneos consigam enviar mensagens ao Kafka de maneira rápida e eficiente. A validação de desempenho da plataforma proposta é apresentada a seguir.

A especificação da interface desenvolvida em relação a três aspectos: (i) Administração de tópicos; (ii) Envio de mensagens e; (iii) Prontidão e saúde do *webservice*. Os parâmetros utilizados na comunicação entre os dispositivos e a plataforma estão descritos na Tabela 1.

Tabela 1. Parâmetros de Comunicação

Parâmetros	**Descrição**
Método HTTP	**GET**, **PUT**, **POST** ou **DELETE**
URI	URL para envio da requisição, e.g. /api/v1/<nome do projeto>/admin/topics
Content-Type da requisição	Tipo do conteúdo da requisição e.g. application
Content-Type da resposta	Tipo do conteúdo da resposta e.g. application/json
Estrutura da carga	arquivo JSON ou binário
Resposta	Arquivo no formato JSON

A Administração de tópicos é dividida em Criação de tópicos, Listagem de tópicos e Remoção de Tópicos. Abaixo segue a estrutura do código para a criação e listagem de tópicos (Algoritmos 5 e 4). Para a remoção é necessário apenas enviar o método DELETE com os parâmetros referente ao tópico que deseja-se remover.

No Envio de mensagens é possível realizar tanto o Envio de mensagens JSON quanto arquivos. Para o envio de arquivos é necessário adicionar o código binário relacionado ao arquivo desejado. O Algoritmo 3 apresenta a estrutura do código do envio de mensagens utilizando JSON.

Algoritmo 1: CRIAÇÃO DE TÓPICOS

```
1 {
2 “topic”: <nome do tópico dentro do projeto>
3 “partitions”: <número de partições a serem criadas no Kafka>
4 “replicationFactor”: <fator de replicação do tópico>
5 }
```

A Prontidão e saúde do *webservice* pode ser feita em dois passos. O primeiro realiza-se um Ping para verificar se o serviço está aceitando requisições. O ping é o *endpoint* básico para determinar a prontidão do serviço. Enquanto que segundo passo verifica se a aplicação está saudável com base na verificação da saúde do produtor de

Algoritmo 2: LISTAGEM DE TÓPICOS

```
1 {
2 “status”: <código de resposta>
3 “topics”: [<lista de tópicos do projeto>]
4 }
```

Algoritmo 3: ENVIO DE MENSAGENS JSON

```
1 {
2 “message”: <corpo da mensagem>
3 }
```

mensagens e do cliente administrativo do Kafka. Ambos os passos são feitos pelo método **GET**. O Algoritmo 4 mostra apresenta a estrutura da resposta para tais passos.

Algoritmo 4: RESPOSTA PARA REQUISIÇÕES

```
1 {
2 “status": <código de resposta>
3 “message": <detalhamento do ocorrido>
4 }
```

4. Avaliação de desempenho

Esta seção apresenta a metodologia utilizada para gerar os resultados por meio de uma avaliação de desempenho. A plataforma proposta foi implantada e avaliada no GCP. A plataforma foi avaliada variando a quantidade de dispositivos (10, 20, 30 e 40) em função do tamanho do arquivo (1KB, 200KB, 400KB, 800KB, 1600KB e 3200KB). Durante os experimentos, foram instanciados 3 três brokers com um tópico para ser publicado. Como o objetivo dos experimentos é avaliar o desempenho da plataforma, foram utilizadas as seguintes métricas: (i) tempo de resposta para enviar uma requisição; (ii) taxa de transferência de dados entre os dispositivos e a plataforma; e (iii) taxa de mensagens por segundo. O conjunto de experimentos foram executados 33 vezes com 95% de confiança de acordo com a distribuição t-student, como apresentado a seguir.

4.1. Impacto dos resultados obtidos

Inicialmente, foi avaliado o tempo de resposta para cada grupo de dispositivos em função da variação do tamanho das mensagens requisitadas, como apresentado na Figura 3. Nota-se que há uma tendência do aumento do tempo de resposta à medida que a carga de dados enviadas pelos dispositivos aumentam. Isto faz sentido, pois a banda de rede estava sendo compartilhada entre todos os dispositivos. Portanto, quanto mais dispositivos, mais tempo será necessário para enviar os dados para a plataforma, aumentando assim o tempo de resposta.

Em seguida, foi avaliado o tráfego de dados da plataforma alterando o tamanho das mensagens requisitadas, como apresentado na Figura 4. Observa-se que apesar de haver um aumento no tráfego de dados, a plataforma possui uma estabilidade para payloads a partir de 800KB. Isso demonstra que quanto mais dispositivos, melhor é o desempenho

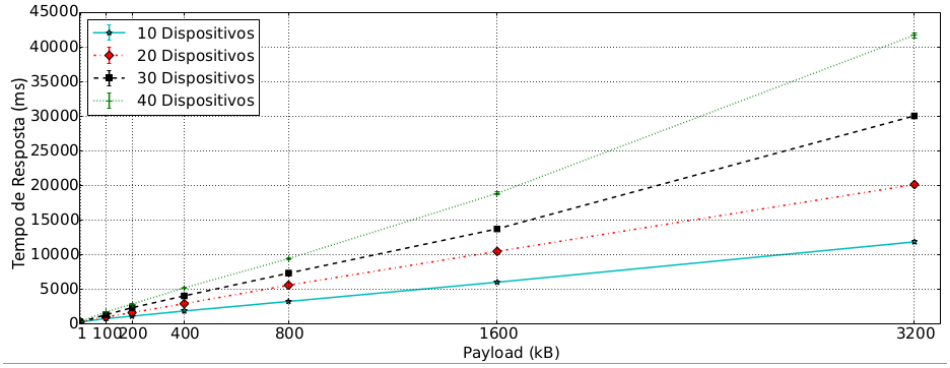

Figura 3. Tempo de resposta em função da carga enviada.

na transmissão. Isso ocorre pois as mensagens não são armazenadas na memória, diferentemente de outras plataformas que apresentam uma degradação de desempenho com o aumento de dispositivos conectados e enviando dados em paralelo. Portanto, os resultados mostram que a plataforma proposta é satisfatória e permite inclusive o uso em aplicações de tempo real.

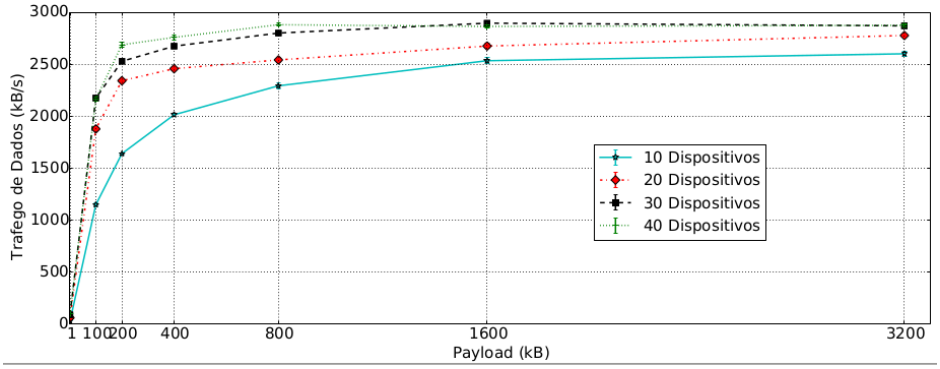

Figura 4. Tráfego de entrada em função da carga enviada.

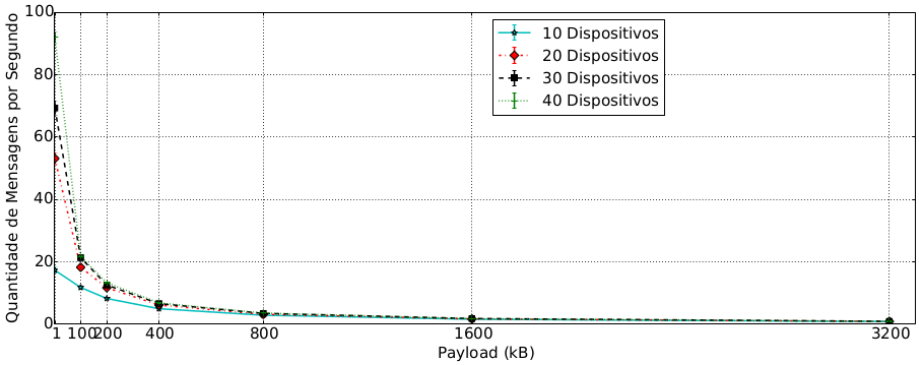

Figura 5. Taxa de entrada de mensagens em função da carga enviada.

Por fim, foi avaliada a quantidade de mensagens entregue por segundo em função da variação da carga enviada, como apresentado na Figura 5. Como pode ser observado, *payloads* menores apresentam uma taxa de entrega de mensagens maior quando comparado com o aumento do *payloads*. Para 30 dispositivos enviando uma mensagem de 1kB

cada, a taxa de mensagens por segundo foi de aproximadamente 70 mensagens/segundos. Isto é, menos de um segundo para a recepção e disponibilidade para processamento. No contexto IoT, os eventos tendem a ter cargas muito baixas, geralmente da ordem de centenas de bytes, considerando que uma das limitações é o uso da rede e a energia gasta na transmissão. Portanto, os dados do gráfico sugerem que a plataforma pode ser útil para aplicações IoT de tempo real ou quase real, pois conseguirá ingerir um volume alto de mensagens, disponibilizando-as com baixa latência para processamento rápido pelas aplicações.

5. Considerações finais

Este artigo apresentou uma nova plataforma de IoT em nuvem com base no Apache Kafka. A plataforma foi capaz de lidar com a integração dos dados e heterogeneidade dos dispositivos. Para isso, implementou-se uma interface de comunicação com o *web-service* REST que abstrai os detalhes a respeito da plataforma com os dispositivos. Com isso, a plataforma armazena as mensagens dos dispositivos, enviando-as para as aplicações consumidoras. Os resultados reais mostraram que a integração do *webservice* com o Kafka é satisfatória, oferecendo taxas de transferências rápidas, da ordem de megabytes por segundo. A ingestão e processamento em tempo real é, portanto, possível com essa plataforma.

Agradecimentos

José R. Torres Neto agradece o apoio financeiro concedido pela FAPESP (processos, 2016/25865-2 e 2017/23655-3) por financiar parte desta pesquisa que é derivada do projeto de Doutorado.

Referências

Botta, A., De Donato, W., Persico, V., and Pescapé, A. (2014). On the integration of cloud computing and internet of things. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 23–30. IEEE.

Gartner, I. (2017). Gartner says 8.4 billion connected "things"will be in use in 2017, up 31 percent from 2016.

Kreps, J., Narkhede, N., Rao, J., et al. (2011a). Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, pages 1–7.

Kreps, J., Narkhede, N., Rao, J., et al. (2011b). Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, pages 1–7.

Triawan, M. A., Hindersah, H., Yolanda, D., and Hadiatna, F. (2016). Internet of things using publish and subscribe method cloud-based application to nft-based hydroponic system. In *System Engineering and Technology (ICSET), 2016 6th International Conference on*, pages 98–104. IEEE.

Wiska, R., Habibie, N., Wibisono, A., Nugroho, W. S., and Mursanto, P. (2016). Big sensor-generated data streaming using kafka and impala for data storage in wireless sensor network for co 2 monitoring. In *Big Data and Information Security (IWBIS), International Workshop on*, pages 97–102. IEEE.

Zhou, B. and Buyya, R. (2018). Augmentation techniques for mobile cloud computing: A taxonomy, survey, and future directions. *ACM Computing Surveys (CSUR)*, 51(1):13.