# Implementação de um cluster Kubernetes com a plataforma Dojot para Aplicações de Internet das Coisas

**Jean Moraes[1], Elen Lobato[1], Denis Rosário[1], Ubiratan Bezerra [1], Eduardo Cerqueira[1], Maria Tostes[1], Andréia Antloga[2]**

[1] Universidade Federal do Pará (UFPA) – Belém, PA – Brazil

[2]Norte Energia S.A – Brasília, DF – Brazil

{jean.anjos.moraes, elen.lobato}@itec.ufpa.br, {denis, bira}@ufpa.br

{cerqueira, tostes}@ufpa.br, andreianascimento@norteenergiasa.com.br

***Abstract.*** *The container-based virtualization technique has been widely used through the adoption of technologies such as Docker and Kubernetes that allow the creation and management of containers. Thus, the present article used Prometheus and Grafana to analyze the performance of a Kubernetes cluster running the Dojot platform and receiving data from a simulator. It was possible to observe that the use of CPU and RAM resources are higher in the machines Workers of the cluster, where the more Dojot services the machine performs the greater the consumption of resources. It was concluded that the use of Kubernetes together with its management and monitoring features are recommended for applications that need to be scalable.*

***Resumo.*** *A técnica de virtualização baseada em contêineres vem sendo amplamente utilizada através da adoção de tecnologias como Docker e Kubernetes que permitem a criação e o gerenciamento de contêineres. Assim, o presente artigo utilizou o Prometheus e o Grafana para analisar a performance de um cluster Kubernetes executando a plataforma Dojot e recebendo dados de um simulador. Foi possível observar que o uso dos recursos de CPU e memória RAM são maiores nas máquinas Workers do cluster, onde quanto mais serviços da Dojot a máquina executa maior o consumo dos recursos. Concluiu-se que a utilização do Kubernetes juntamente com os seus recursos de gerenciamento e monitoramento são recomendáveis para aplicações que precisam ser escaláveis.*

## 1. Introdução

A prática de virtualização tem se tornado cada vez mais usual, por grandes e pequenas empresa, como forma de melhorar os serviços por elas oferecido, reduzir a infraestrutura, o consumo de energia e recursos humanos [Dewi et al. 2019]. Consequentemente, essas empresas acabam diminuindo os seus custos e aumentando o seu lucro. A virtualização pode ser realizada da forma mais tradicional utilizando Máquinas Virtuais (VMs), ou de forma mais recente através de contêineres [Abuabdo and Al-Sharif 2019].

A virtualização baseada em contêineres é a que mais tem ganhado força nos últimos anos [Pereira Ferreira and Sinnott 2019]. Conhecida também como virtualização a nível de sistema operacional [Dewi et al. 2019] ou virtualização leve [Abuabdo and Al-Sharif 2019]. A virtualização baseada em contêineres permite dividir um servidor em várias instâncias, chamadas de "Contêineres", compartilhando assim o mesmo kernel do sistema operacional.

Uma das soluções mais utilizadas para criar os contêineres é o Docker, e para gerenciar os contêineres é o Kubernetes. O Kubernetes segue o modelo mestre-escravo, onde cada máquina é chamada de "nó", a máquina mestre é chamada de "nó *Master*"e os

escravos são chamados de "nós *Workers*". Um mestre é usado para gerenciar contêineres Docker em vários nós do Kubernetes. Os nós por sua vez podem ser máquinas físicas ou virtuais, que servem para executar os contêineres. O mestre e seus nós controlados constituem um “cluster Kubernetes” [Foundation 2021].

Assim, é possível implantar uma aplicação nos contêineres Docker através do mestre Kubernetes. Normalmente, uma aplicação é dividida em um ou mais componentes, chamadas de contêineres, que são executadas em um ou mais "pods", onde cada pod é geralmente restrito em relação a uma quantidade máxima de recursos que pode consumir [Foundation 2021]. Assim, uma das maiores vantagens do Kubernetes é o uso do hardware (RAM, disco e CPU) de forma otimizada e inteligente, o que ajuda a economizar recursos [Dewi et al. 2019].

No que tange às aplicações em contêineres que utilizam Docker e Kubernetes em sua infraestrutura, pode-se citar a plataforma Dojot que tem como seu principal desenvolvedor o Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD). A Dojot é uma aplicação *open source* que nasceu com o objetivo de desenvolver e demonstrar tecnologias para as cidades inteligentes, visando facilitar o desenvolvimento de soluções e o ecossistema de Internet das Coisas (IoT) [CPqD 2020]. Especificamente, com a plataforma Dojot é possível criar perfis de usuários, *templates*, dispositivos e fluxos de processamento de dados. Em relação aos dispositivos eles podem representar dispositivos reais ou virtuais e podem receber diversos tipos de dados [Mota et al. 2019]. Os dados recebidos podem ser visualizados em tempo real na interface web da Dojot, como também podem ser consumidos por aplicações [CPqD 2020].

Nesse contexto, foi montado um sistema considerando um cluster Kubernetes com a plataforma Dojot no âmbito do Projeto de Pesquisa e Desenvolvimento (P&D) intitulado de “Sistema Inteligente de Gestão Eficiente de Mobilidade Elétrica Multimodal (SIMA)"que vem sendo desenvolvido na Universidade Federal do Pará (UFPA). O projeto foi aprovado em uma chamada especial de P&D de Mobilidade Elétrica da Agência Nacional de Energia Elétrica (ANEEL) e é financiado pela Norte Energia. O projeto prevê a aquisição de dois ônibus elétricos, a construção de um barco elétrico e contará ainda com uma grande infraestrutura composta por eletropostos, sistema fotovoltaico, armazenamento de energia elétrica em baterias, medidores, uma rede de transmissão de dados, um sistema de gestão, entre outras aplicações.

Todos esses dispositivos enviarão diversos tipos de dados para a plataforma Dojot que será o *middleware* do projeto. Esses dados serão processados para fornecer informações e ajudar nos processos de tomadas de decisões do sistema de gestão proposto e as demais aplicações. Assim, tanto a plataforma Dojot que está presente no cluster Kubernetes do projeto, quanto o sistema de gestão e as demais aplicações que também estarão presentes no cluster, precisam ser escaláveis. Portanto, é preciso implantar e testar o sistema de cluster Kubernetes que contém a plataforma Dojot e todas as demais aplicações. Desse modo, este artigo apresenta e avalia um sistema de um cluster Kubernetes quando a plataforma Dojot recebe uma grande quantidade de dados. Será utilizando o *Prometheus* e *Grafana*, que são recursos que fazem parte do Kubernetes, para analisar o uso da CPU e memória RAM das máquinas do cluster. A principal motivação para o desenvolvimento deste trabalho é verificar se o cluster criado irá suportar os dispositivos e aplicações até agora previstos no projeto supracitado.

O restante do artigo será apresentado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve o sistema criado. A Seção 4 mostra os resultados obtidos. A seção 5 aponta alguns trabalhos futuros. Por fim, a Seção 6 apresenta a conclusão.

## 2. Trabalhos Relacionados

Em [Medel et al. 2016] foi analisado o desempenho de um cluster do Kubernetes composto por duas máquinas através de um modelo de gerenciamento que consi-

dera recursos como quantidade de RAM e número de núcleos disponíveis. Em [Sukhija and Bautista 2019] foi proposta uma arquitetura semelhante à empregada neste artigo utilizando Prometheus, Grafana e outras plataformas preditivas, para monitorar e gerenciar proativamente um data center que utiliza Kubernetes. Ainda mais semelhante com a análise realizada neste artigo foi o trabalho de [Dewi et al. 2019], onde também foi criado um cluster Kubernetes com três nós (sendo uma máquina *master* e duas máquinas *worker*) em um servidor físico e foi avaliada a carga da CPU nós do cluster.

Já em relação à análise de desempenho de um cluster Kubernetes utilizando especificamente a plataforma Dojot, poucos trabalhos são encontrados na literatura. Isso ocorre porque a Dojot é relativamente nova, pois ela foi lançada em 2017. Em [Cesila et al. 2020] foi apresentada uma solução de monitoramento para uma *network slice* em nuvens computacionais para ambientes multidomínio, onde uma das redes monitoradas é de um cluster Kubernetes com a plataforma Dojot, implantado em um servidor. Para o monitoramento também foi utilizado o Grafana e a análise realizada verificou o tempo de criação de três *Datas Centers*.

O que todos os trabalhos têm em comum é a utilização do Kubernetes para gerenciar contêineres que servem para armazenar aplicações escaláveis. Já o que os diferencia é a quantidade de nós presentes no cluster e a forma como os dados que alimentam as aplicações presentes nos clusters são gerados. O diferencial deste artigo em relação aos demais artigos já encontrados na literatura está na análise do uso dos recursos de memória RAM e CPU do cluster Kubernetes sendo usados exclusivamente com a plataforma Dojot recebendo dados de um simulador. Esses dados são semelhantes aos dados reais que serão enviados pelos sistemas e aplicações que estarão presentes no projeto SIMA, ao qual o cluster é destinado. Isso irá possibilitar o monitoramento dos recursos supracitado e a previsão de o quanto a aplicação Dojot ainda pode ser escalável.

## 3. Descrição do Sistema

Esta Seção apresenta a arquitetura do sistema desenvolvido, onde são descritos os *softwares* utilizados e suas respectivas funções. Além disso, são descritas as configurações do cluster Kubernetes implantado e a forma como a plataforma Dojot foi configurada.

### 3.1. Arquitetura do Sistema

O sistema proposto é composto por um cluster Kubernetes, a plataforma Dojot, um simulador de dados Python, e os *softwares* Ansible, Prometheus e Grafana, como pode ser observado na Figura 1. O Kubernetes tem a função de gerenciar o contêiner do sistema, o Virtual Box foi utilizado para criar as máquinas virtuais que compõem o cluster, a Dojot é o *middleware* do sistema é quem recebem os dados. O simulador de dados tem a função de criar *templates*, *devices* e enviar dados para a Dojot.

O Ansible foi utilizado para implantar os componentes descritos anteriormente de forma automatizada através de *Playbooks*. O Prometheus possibilita gravar dados em tempo real, permitindo assim o monitoramento e alertas de eventos. Ele está integrado ao Grafana para possibilitar o monitoramento de todo o cluster. Por fim, o Grafana permite observar o desempenho do cluster Kubernetes onde a Dojot foi implantada. Através dele é possível gerar *dashboards*, gráficos e tabelas, com base nos dados recebidos na Dojot e gravados no Prometheus. O sistema criado foi implantado dentro de um servidor que possui as especificações técnicas descritas na Tabela 1.

### 3.2. Configuração do Cluster

O cluster Kubernetes deste sistema foi criado em VMs com o *software Virtual Box*. Foram criadas três VMs, ou seja, três "nós"dentro do cluster Kubernetes, são eles: *Master*, *Worker 1* e *Worker 2*. A Figura 2 apresenta a arquitetura do cluster. A seguir estão listadas as descrições de cada um dos nós do cluster: $i$) O nó *Master* administra o cluster Kubernetes. Nele estão localizados o Ansible, o simulador e não há nenhum serviço da Dojot

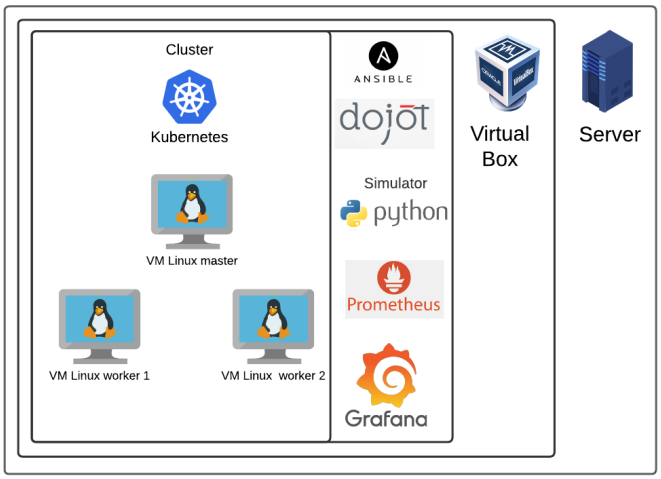



**Figura 1. Arquitetura do Sistema.**

**Tabela 1. Especificações Técnicas do Servidor.**

| Modelo | Power Edge T440 |
|---|---|
| Sistema Operacional | Linux Ubuntu Server 18.04 |
| Processador | Intel Xeon Silver 4210 (2 unidades) |
| Quantidade de Núcleos | 40 |
| Memória | 64GB |
| HD | 5TB |

sendo executado nele. $ii)$ Nos nós *Workers* são executados os serviços da Dojot. Para um ambiente onde há uma carga grande de dispositivos é recomendado a utilização de no mínimo 2 nós *Workers* no cluster Kubernetes. Na Tabela 2 são apresentados os recursos alocados a cada um dos nós.

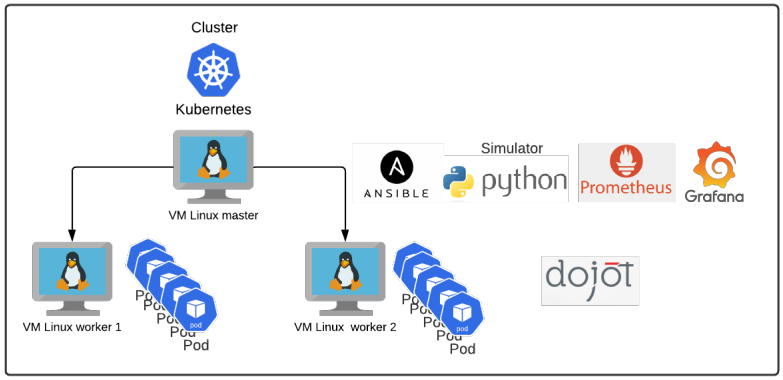



**Figura 2. Arquitetura do Cluster Kubernetes.**

### 3.3. Configuração da Dojot

Após a instalação da Dojot, através de linha de comando, a configuração da Dojot pode ser realizada tanto através de suas Interface de Programação de Aplicações (APIs, sigla em inglês), quanto através da sua Interface Gráfica do Usuário (GUI, singla em inglês). Sendo que, a configuração da Dojot através da GUI é mais fácil para os usuários finais. O ambiente da Dojot pode ser configurado para suportar cenários com muitos dispositivos IoT conectados (até 100 mil dispositivos) ou cenários de poucos dispositivos IoT

**Tabela 2. Versão dos componentes do sistema.**

| Nó | CPU | RAM | HD |
|---|---|---|---|
| Master | 8 Cores | 13.7 GB | 100 GB |
| Worker 1 | 8 Cores | 13.7 GB | 300 GB |
| Worker 2 | 8 Cores | 13.7 GB | 300 GB |

(até 500 dispositivos). Para este artigo, a plataforma foi configurada para o ambiente de poucos dispositivos, que de acordo com documentação em [CPqD 2020], suporta até 500 dispositivos enviando mensagens de 100 bytes a cada 15 segundos.

Utilizando um simulador em linguagem Python baseado nos comandos disponíveis na documentação oficial da Dojot [CPqD 2020], os *templates* e *devices* foram criados de forma automática na Dojot, através de APIs. O simulador também foi utilizado para enviar dados de telemetria para os *devices* criados. Dessa forma, foram criados 18 *templates* e 64 *devices*, conforme mostram, respectivamente, as Figuras 3 e 4. Esse cenário representa o cenário do projeto, onde cada *device* criado representa um dispositivo físico, que pode ser fixo, como por exemplo, os eletropostos, medidores e sistemas elétrico, ou móveis, como os ônibus e barco elétrico. Esses dispositivos estarão enviando dados continuamente para a plataforma Dojot e estes dados posteriormente serão consumidos por outras aplicações. Nas Figuras 3 e 4 é possível visualizar alguns *templates* e *devices* nomeados com a palavra "mensal"que representam dispositivos virtuais que irão receber os dados de cálculo de indicadores e variáveis mensais, como por exemplo, tensão, corrente, potência, entre outras.

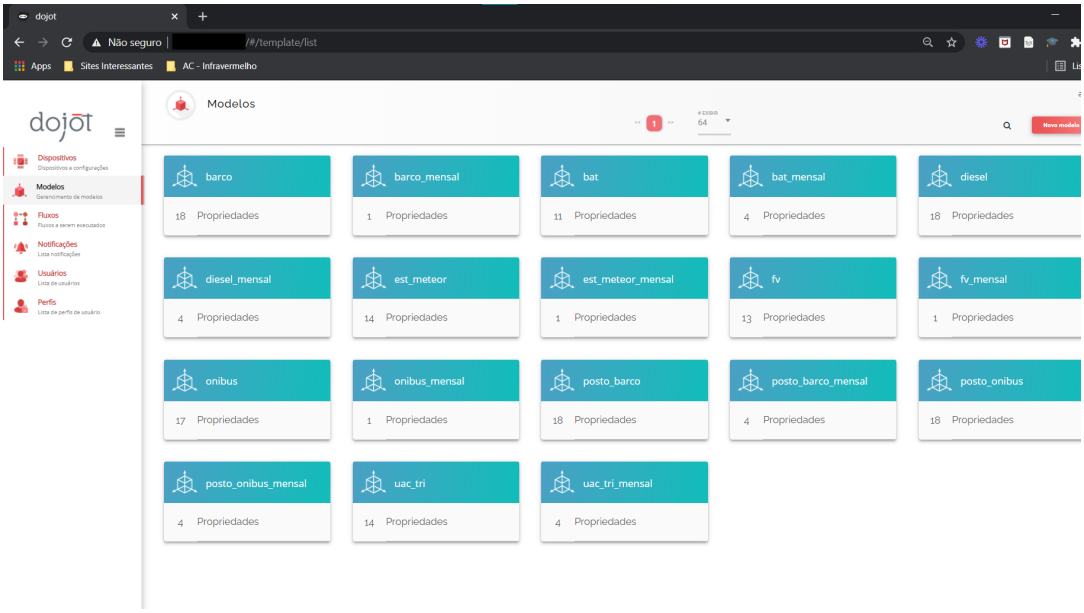



**Figura 3. *Templates* criados na Dojot.**

## 4. Resultados e Análises

Para avaliar o desempenho do *cluster Kubernetes* criado, foi executado o sistema da plataforma Dojot com 64 dispositivos IoT recebendo dados simultaneamente a cada 10 minutos. Foram utilizados 64 dispositivos, porque até agora essa é a quantidade necessária de dispositivos no contexto do projeto de mobilidade. As métricas utilizadas para gerar os resultados são relativas ao uso de recursos disponíveis por cada VM. Sendo a primeira métrica a porcentagem de uso de CPU e a segunda de quantidade de uso de memória

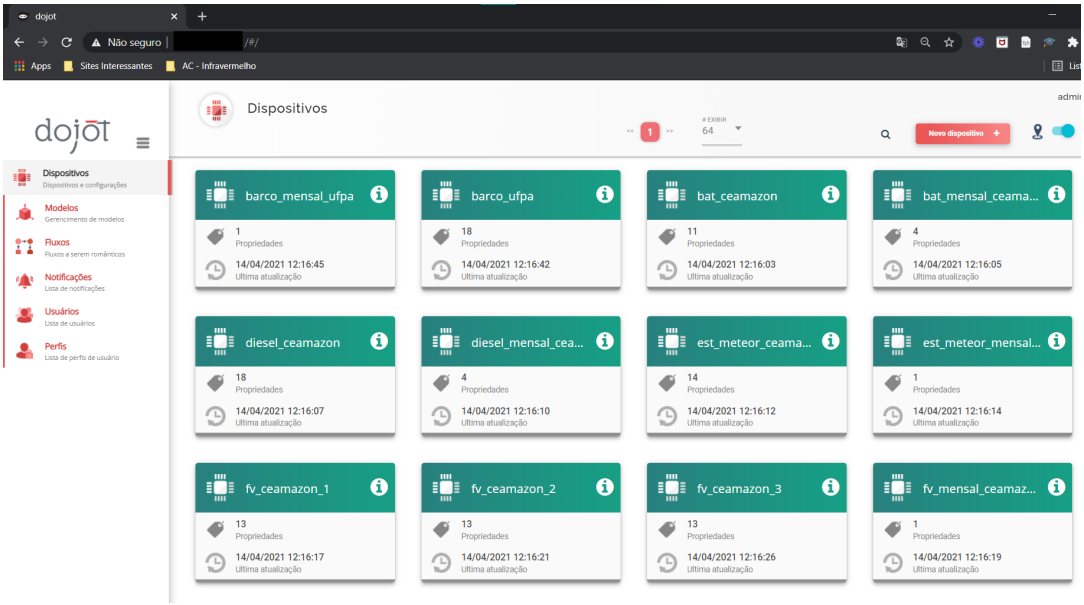
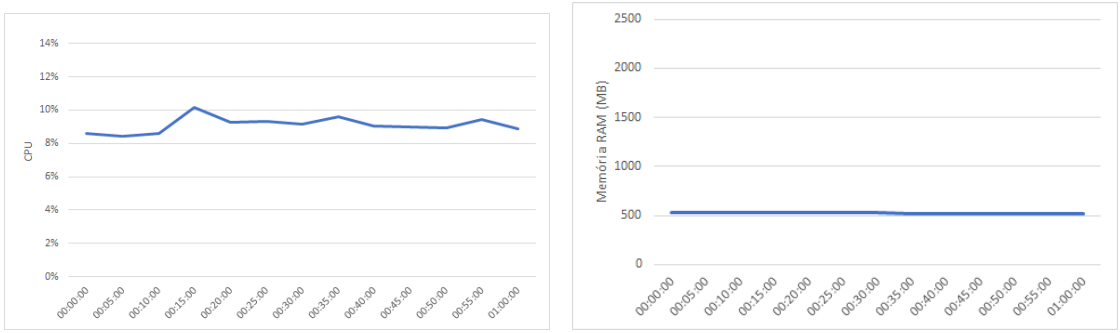



**Figura 4. *Devices* criados na Dojot.**



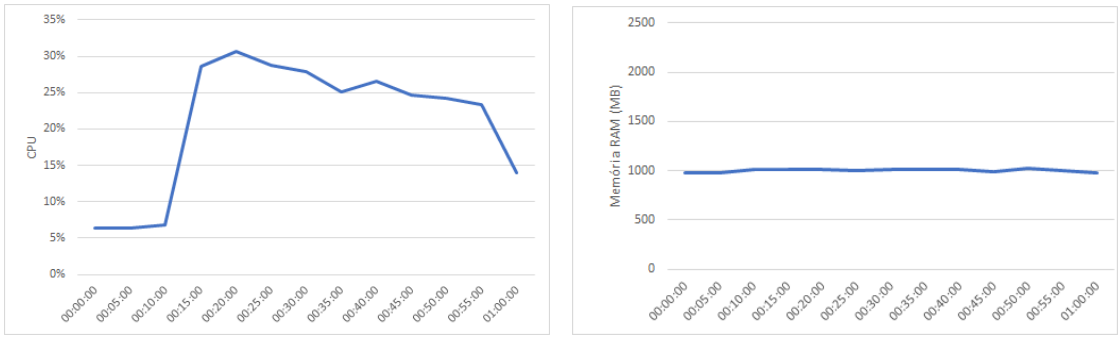

(a) Porcentagem de uso de CPU. (b) Quantidade de uso de memória RAM.

**Figura 5. Resultados para a máquina *Master***

RAM. Ressalta-se que foi utilizado tempo de uma (1) hora para o tempo simulado, que é o mesmo tempo de execução do experimento.

A partir do gráfico da Figura 5(a) pode-se observar que a quantidade de uso de CPU da máquina *Master* manteve um comportamento próximo ao constante, realizando apenas pequenas variações ao longo do experimento, onde o valor médio aproximado é de 9%. Já na Figura 5(b) é possível visualizar uma curva constante de valor médio de 525 MB para o consumo de memória RAM da VM. Nesse sentido, observa-se que apesar de haver uma alta carga de envio de dados por dispositivos ao *cluster*, a máquina *Master* apresenta valores baixos de uso de recursos. O que é esperado, uma vez que a VM *Master* tem responsabilidade apenas de executar serviços de gerenciamento do *cluster* e a carga de execuções da aplicação Dojot são todas nas máquinas *Worker 1* e *Worker2*, seguindo o modelo mestre-escravo do cluster Kubernetes.

Por meio da Figura 6(a), a qual representa o uso de CPU pela máquina *Worker 1*, constata-se um comportamento crescente a partir dos 10 minutos iniciais e um comportamento decrescente a partir dos 7 minutos finais do experimento. Esses períodos representam, respectivamente, o momento em que os dispositivos estabelecem conexão e o momento em que os dispositivos finalizam o recebimento de dados. Assim, essa métrica atingiu o valor máximo de 31,46%, médio de 22% e mínimo de 5,56%. Ademais,




(a) Porcentagem de uso de CPU (b) Quantidade de uso de memória RAM

**Figura 6. Resultados para a máquina Worker1**

na Figura 6(b) é observado um valor constante de 1 GB para o uso de memória RAM da máquina, onde se conclui que apesar de haver uma alta carga de comunicação com dispositivos pela aplicação Dojot, o recurso sumariamente consumido é o de CPU.

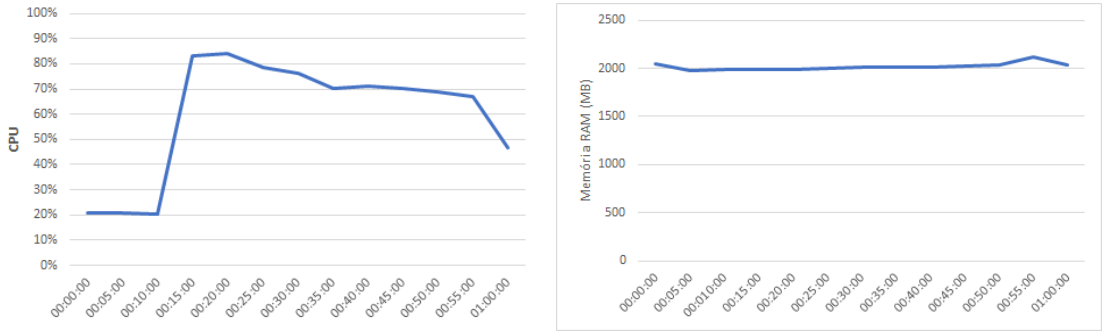
A Figura 7(a) demonstra o uso de CPU pela máquina *Woker 2*. Nesse sentido, constata-se que há um comportamento similar da curva de consumo de recurso da *Worker 1*, onde há um crescente uso de CPU nos minutos iniciais do experimento, porque existe um estabelecimento de conexão pelos dispositivos comunicantes. Por outro lado, também há um rápido decréscimo de consumo durante o período final do experimento, com o término das transmissões de dados. Para a porcentagem de uso de CPU no experimento, a máquina *Worker 2* estabeleceu valor máximo de 85,28%, valor médio de 61,67% e valor mínimo de 20,12%. Em relação ao consumo de recurso de memória RAM, igualmente às outras máquinas, a máquina *Worker 2* representou um comportamento de uso constante de 2,11 GB. Assim, infere-se que quanto maior a carga de envio de dados por dispositivos, maior o uso de CPU. O fato de os valores dos recursos utilizados terem sido maiores na *Worker 2* em relação a *Worker 1* ocorre porque existem mais serviços da Dojot sendo executados na *Worker 2*. Assim, os resultados obtidos tanto para o *Worker 1*, quanto para a *Worker 2*, pois conforme supracitado, a carga de execuções da Dojot é realizada nessas máquinas. Para diminuir o consumo de recurso em ambas as máquinas, o que pode ser feito é a adição de mais recursos nessas máquinas *Workers* já existentes, ou a adição de mais máquinas *Workers* no *cluster*.

## 5. Trabalhos Futuros

Como trabalhos futuros pretendesse realizar um estudo de escalabilidade para se verificar se o *cluster* criado realmente irá suportar a quantidade máxima de 500 dispositivos, conforme descrito na documentação da Dojot [CPqD 2020].

## 6. Conclusão

Com o desenvolvimento de aplicações de computação cada vez mais robustas no mercado, tornou-se cada vez mais proveitoso aplicar tecnologias promissoras de virtualização por contêineres como *Docker* e *Kubernetes*. A plataforma *Kubernetes* dispõe de diversas vantagens de gerenciamento e monitoramento de recursos para uma aplicação de larga escala. Nesse sentido, o presente artigo apresentou uma análise de consumo de recursos pela plataforma Dojot em um *cluster Kubernetes* utilizando um cenário de dezenas de dispositivos transmitindo dados paralelamente. Desse modo, com base nos resultados, pode-se depreender o gerenciamento inteligente de recursos de CPU e memória RAM pelo *Kubernetes* ao longo das máquinas virtuais do *cluster*.




(a) Porcentagem de uso de CPU (b) Quantidade de uso de memória RAM

**Figura 7. Resultados para a máquina Worker2**

## Agradecimentos

Os autores do artigo agradecem o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES); e da Norte Energia S. A. para a realização do presente trabalho, no âmbito do projeto de P&D ANEEL 07427-0319/2019 intitulado "Sistema Inteligente de Gestão Eficiente de Mobilidade Elétrica Multimodal", realizado através da Chamada Estratégica ANEEL nº 22/2018.

## Referências

Abuabdo, A. and Al-Sharif, Z. A. (2019). Virtualization vs. containerization: Towards a multithreaded performance evaluation approach. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–6.

Cesila, C. H. et al. (2020). A multi-domain monitoring architecture for cloud network slicing: Arquitetura de monitoramento para fatiamento de redes em nuvens computacionais para ambientes multidomínio.

CPqD (2020). Dojot soluções para iot - plataforma de desenvolvimento para iot. `http://www.dojot.com.br/`. (Accessed on 03/09/2021).

Dewi, L. P., Noertjahyana, A., Palit, H. N., and Yedutun, K. (2019). Server scalability using kubernetes. In *2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)*, pages 1–4.

Foundation, T. L. (2021). Kubernetes documentation | kubernetes. `https://kubernetes.io/docs/home/`. (Accessed on 04/15/2021).

Medel, V., Rana, O., Bañares, J. , and Arronategui, U. (2016). Modelling performance resource management in kubernetes. In *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, pages 257–262.

Mota, R., Riker, A., and Rosário, D. (2019). Adjusting group communication in dense internet of things networks with heterogeneous energy sources. In *11th Brazilian Symposium on Ubiquitous and Pervasive Computing*, Porto Alegre, RS, Brasil. SBC.

Pereira Ferreira, A. and Sinnott, R. (2019). A performance evaluation of containers running on managed kubernetes services. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 199–208.

Sukhija, N. and Bautista, E. (2019). Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus. In *IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, pages 257–262.