

**INSTITUTO
FEDERAL**
Santa Catarina

Câmpus
São José

Avaliação de desempenho de diferentes tecnologias de contêineres em dispositivos de borda baseados em ARM

Curso: Engenharia de Telecomunicações
Disciplina: ADS29009 – Avaliação de Desempenho de Sistemas
Professor: Eraldo Silveira e Silva

Aluno

João Pedro Menegali Salvan Bitencourt

Cenário

Quantidade crescente de dispositivos IoT

- Servidores recebem uma quantidade cada vez maior de dados;
- Servidores gastam mais recursos de processamento e memória RAM;
- Aumento no custo e complexidade de manutenção.

Uma solução

Dispositivos de borda

- Mais próximos aos dispositivos IoT;
- Possuem quantidades limitadas de memória RAM e capacidade de processamento;
- Contêineres são os candidatos ideais por serem leves e possuírem isolamento.

Contêineres

Características

- Ambiente isolado em um sistema operacional, possível graças à mecanismos de *kernel* como *Cgroup* e *Namespaces*;
- Aplicações não possuem dependências no sistema operacional hospedeiro;
- Possibilidade de executar múltiplas instâncias da mesma aplicação:
 - Ideal para ambiente de teste com diferentes versões;
- Abstrai o processo de instalação de uma dada aplicação;
- Possibilidade de executar aplicações não suportadas em um sistema operacional:
 - Aplicações recentes em sistemas antigos;
 - Aplicações antigas em sistemas modernos.
- Modularização de uma solução de *software*.

Tecnologia de Contêineres

Ferramentas

- Docker (Open Container Initiative):
 - *Runtime containerd*;
 - Modelo cliente-servidor;
 - Executado por padrão com permissão de *root*.
- Podman (Open Container Initiative):
 - *Runtime runc, crun, runv*, entre outros;
 - Modelo independente;
 - Executado por padrão com permissão de usuário desprivilegiado. Pode ser executado por usuário privilegiado. Dessa forma, é mais seguro do que as outras opções.
- Singularity (Open Container Initiative e próprio):
 - Modelo independente;
 - Para compatibilidade com OCI exige a execução com usuário privilegiado.

Estudo

Tecnologia dentro dos contêineres

- Algoritmos utilizados em aplicações de visão computacional:
 - Haar Cascades:
 - Detecção de objetos, muito utilizado no reconhecimento facial.
 - HOG (Histogram of Oriented Gradients):
 - Detecção de objetos, otimizado para componentes humanos.
 - CNN (Convolutional Neural Networks) com YOLO (You Only Look Once);
- Carga de trabalho comuns em ciência de dados;

Parâmetros Fixados

Cenário de testes

- Três tecnologias de contêineres: Docker, Podman e Singularity, em nós de borda com arquitetura de processador ARM;
- O nó recebe imagens e dados de rastreo do serviço Fitbit;
- As imagens e dados de rastreo são analisados e enviados à nuvem para processamento posterior;
- Há interface de rede com e sem fio, e os testes foram executados nas combinações:
 - IoT e nó com fio;
 - IoT e nó sem fio.
- Há dois conjuntos de aplicações:
 - Visão computacional;
 - Ciência de dados.

Cenário de testes - Imagem

- A cada segundo, o contêiner recebe uma imagem ou um trecho de dados dos sensores do dispositivo IoT.
- No primeiro caso, as imagens são enviadas pelos sensores do IoT contendo nome e o tamanho da mesma;
- Ao receber a mesma, o processo de detecção é realizado, o resultado é salvo no contêiner, e os recursos utilizados são salvos em um arquivo;
- As imagens possuem extensão JPG e são salvas com o um nome correspondente ao que foi detectado (rosto, veículo, corpo ou objeto);
- Foi utilizado um conjunto de dados com 1000 imagens de dimensões 1280x720 pixels;
- Dois conjuntos de imagens foram utilizados:
 - Corpo e rosto compartilham o mesmo conjuntos de dados;
 - A detecção de veículos utilizou um conjunto separados de 1000 imagens.

Cenário de testes - Imagem

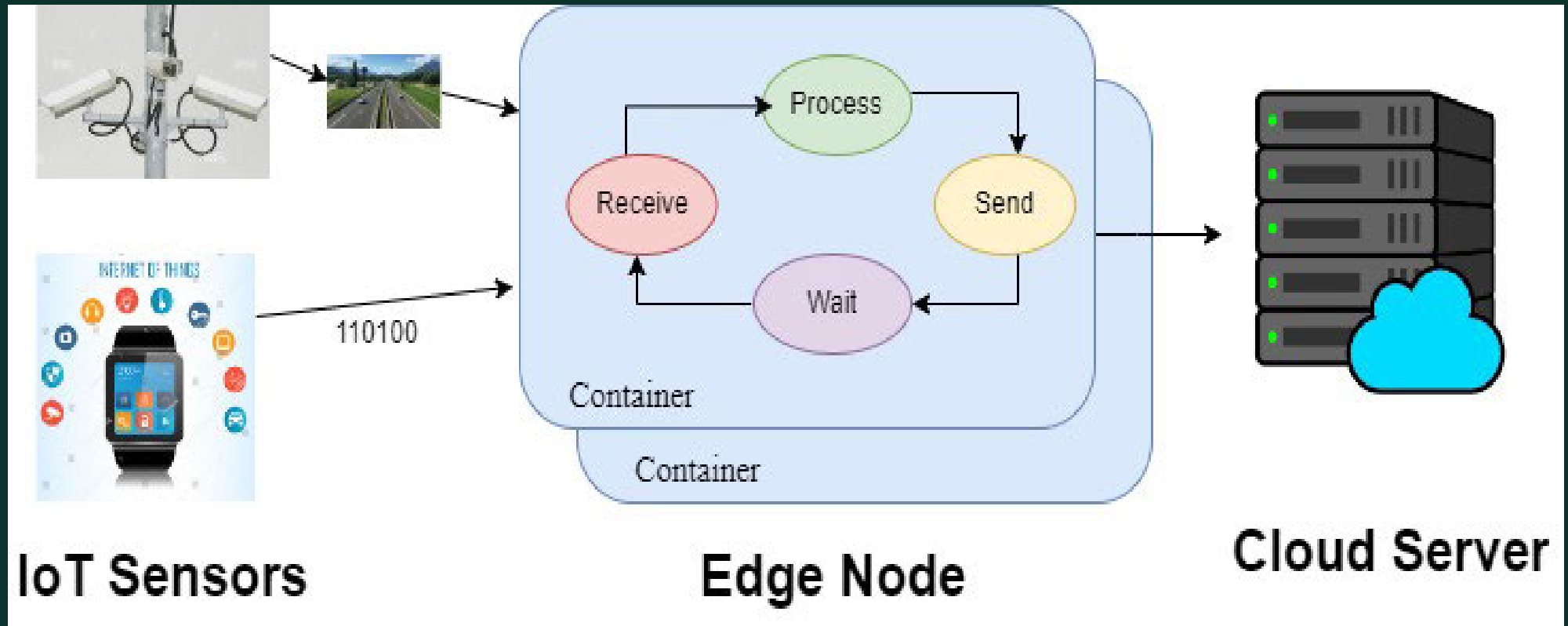
- Foi utilizado um conjunto de dados com 1000 imagens de dimensões 1280x720 pixels;
- Dois conjuntos de imagens foram utilizados:
 - Corpo e rosto compartilham o mesmo conjuntos de dados;
 - A detecção de veículos utilizou um conjunto separados de 1000 imagens;
- O conjunto de dados para a detecção de objetos combinou 500 imagens do primeiro conjunto (corpo e rosto) e 500 imagens do segundo conjunto (veículos).

Cenário de testes – Dados de rastreo

- A cada 5 segundos, o contêiner recebe trechos de dados do Fitbit, salva-os dentro de si, faz a limpeza dos dados, realiza a análise dos dados e transmite o resultado para a nuvem;
- Somente dados de Atividades Diárias foram utilizados para manter a consistência e padronização de parâmetros para cada contêiner;
- Alguns atributos fornecidos pelo conjunto:
 - Data da atividade;
 - Quantidade de passos total;
 - Distância total;
 - Calorias.
- O resultado consiste no cálculo da média de passos para cada usuário e encontrar o usuário com a maior quantidade de passos.

Diagrama do sistema

Diagrama do sistema



Sistema proposto

Métricas

Métricas observadas

- Capacidade computacional:
 - uso de CPU em porcentagem;
 - Uso de memória RAM em MB.
- Desempenho da rede:
 - Quantidade de dados recebidos e transmitidos em NET I/O.
- Armazenamento:
 - Quantidade de bytes escritos e lidos pelo sistema de arquivos do contêiner em BLOCK I/O.
- Realizadas métricas estatísticas dos recursos acima como média, mediana, máximo, mínimo e desvio padrão.

Medições

Itens medidos

- Tempo de espera: mede o período em que uma aplicação fica na fila antes da execução. Permite observar a eficiência no escalonamento e alocação recursos pelo contêiner;
- Tempo de recebimento: mede a eficiência do contêiner quando um fluxo de dados é recebido;
- Tempo de processamento: Indica a eficiência do contêiner em relação a capacidade computacional utilizada pelos algoritmos testados;
- Utilização de recursos: Indica quão eficiente um contêiner é em relação a gerência na alocação de recursos como CPU e RAM;
- Taxa de transferência (*throughput*): Mede a taxa em que um contêiner consegue processar uma certa quantidade de tarefas ou requisições, indicando a capacidade geral de processamento.

Hardware

Hardware – Nó de borda

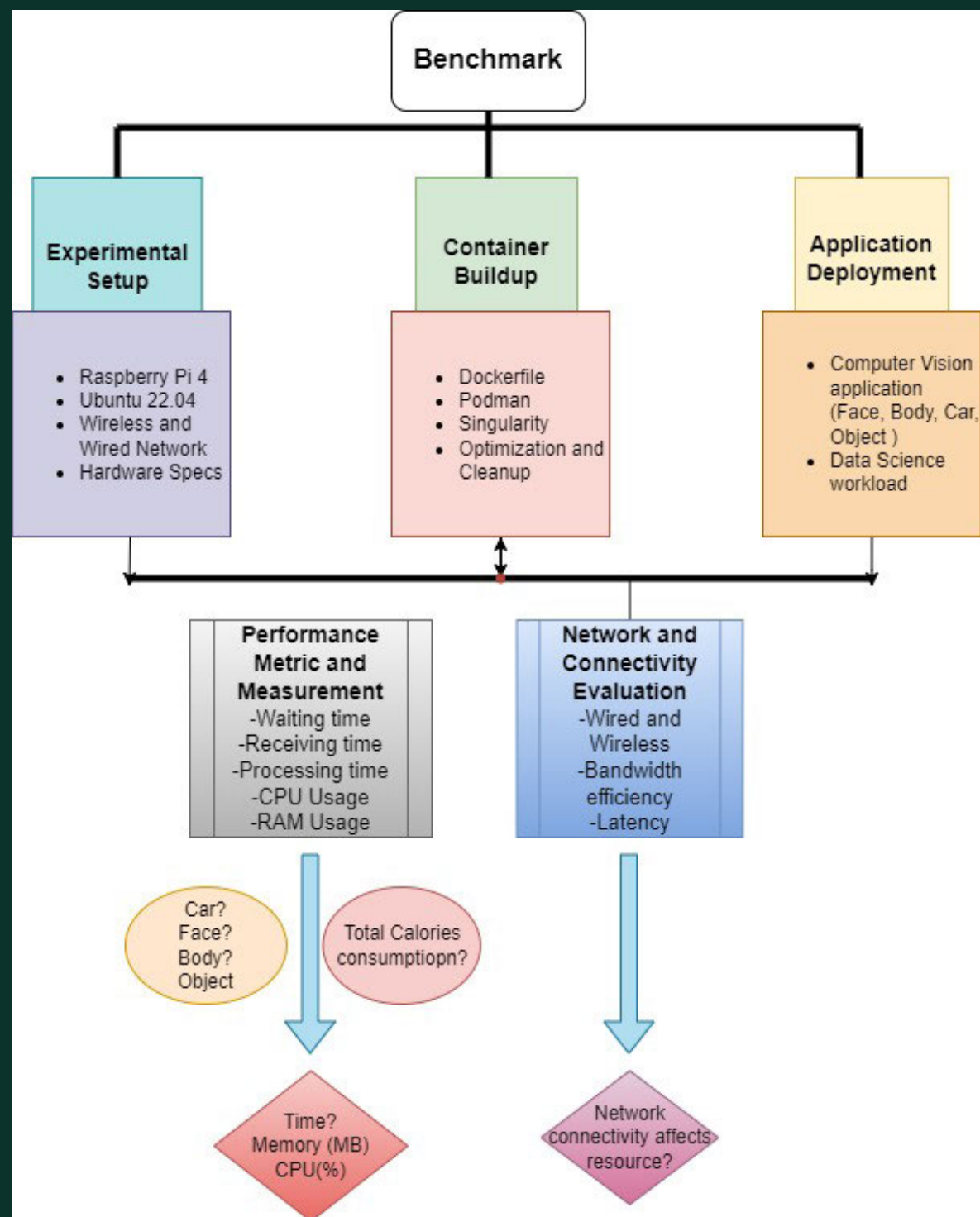
Arquitetura:	ARM 64
SoC:	Broadcom BCM2711
Processador:	ARM 1,5 GHz CPU
Núcleos de CPU:	4 núcleos Cortex A-72
Memória RAM:	4 GB
Sistema Operacional:	Ubuntu 22.04, 64 bits
Consumo de energia:	500 mA
Cartão Micro-SD:	Disponível
Rede sem fio:	2,4 GHz e 5 GHz
Bluetooth:	5.0

- Para a rede, utilizado o roteador ASUS RT-N56U Dual Band Wireless N600 Gigabit.

Criação dos contêineres

Ambiente para testes

- Foi utilizado um *Dockerfile* para construir a imagem utilizada na criação dos contêineres utilizados pelo Docker, Podman e Singularity. Dessa forma, mantém-se a equidade do ambiente;
- O uso de recurso do Docker e Podman são monitorados através dos comandos:
 - `docker stats` → Docker
 - `podman stats` → Podman
- Adicionalmente, foi utilizado o *psutil* do *Python* para também realizar o monitoramento dos uso de recursos;
- Os contêineres criados já estão otimizados, contendo apenas o necessário para executar os testes;
- Para manter a uniformidade nos testes, em todos os contêineres foram utilizadas as mesmas bibliotecas do OpenCV e os mesmos *scripts* de envio e recebimento escritos em Python.



Fluxo dos componentes do teste de desempenho.

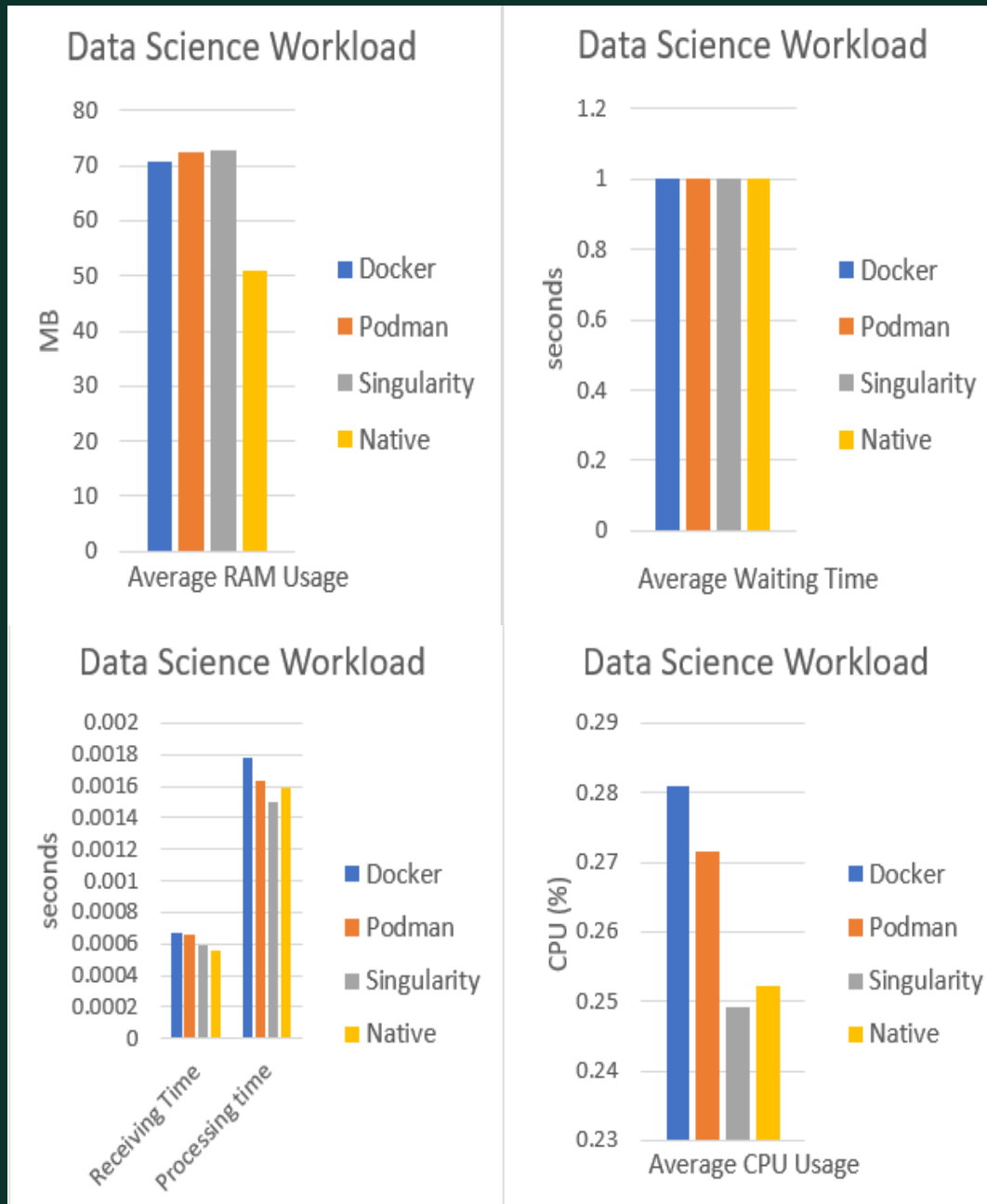
Fatores e níveis

Fatores e níveis

- Cada tecnologia de contêiner representa um fator;
 - Dois fatores:
 - Usar contêiner;
 - Não usar contêiner.
- A carga de trabalho de ciência de dados e os algoritmos de visão computacional em cada tecnologia de contêiner, juntamente com essas mesmas tarefas executadas de forma nativa, compõem dezesseis níveis:
 - 3 tecnologias de contêineres X 4 tipos de tarefas executadas
+
 - As 4 tipos de tarefas executadas sem o uso de contêiner.

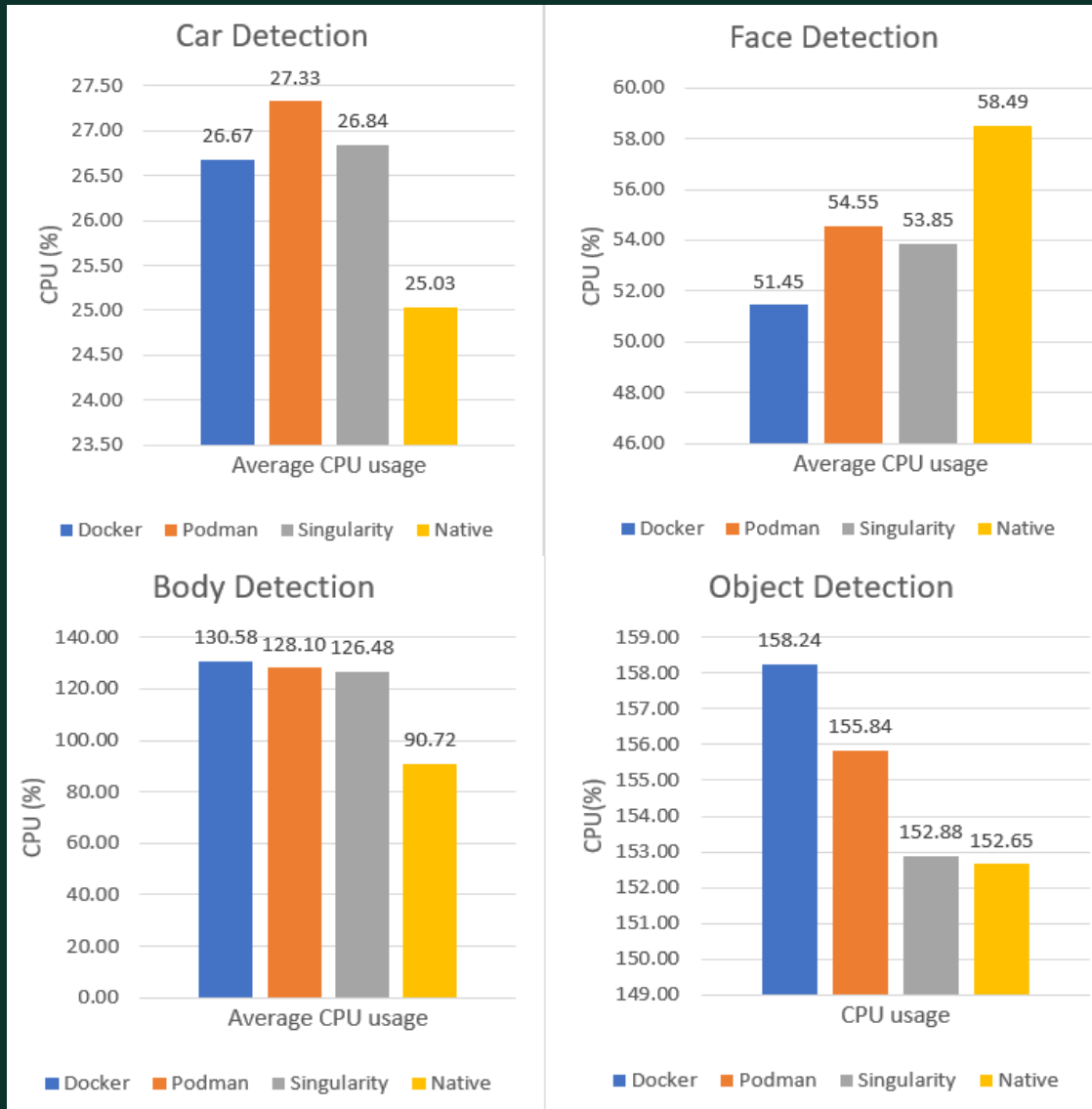
Gráfico das medições

Carga da ciência de dados



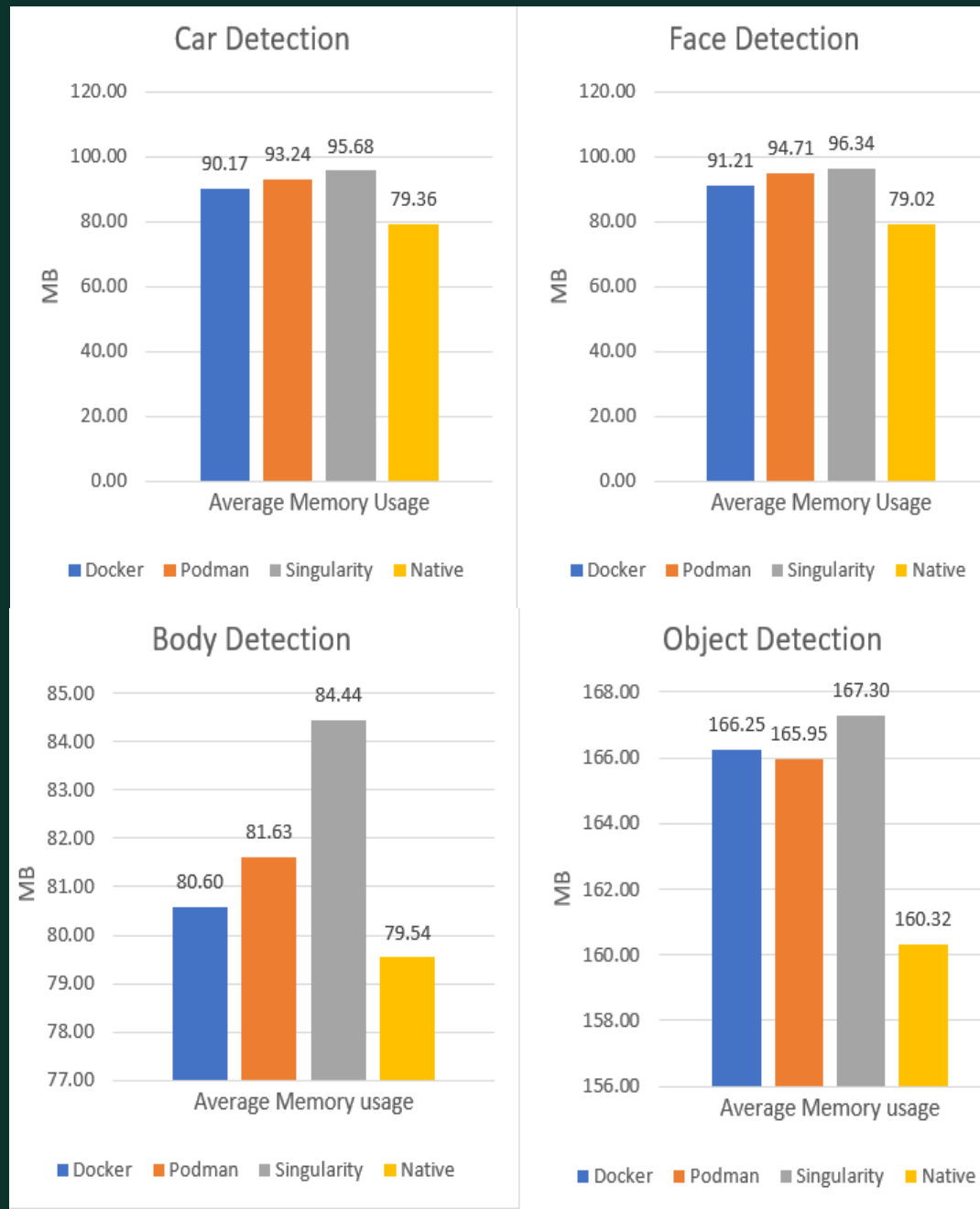
Tempo médio de recebimento, processamento, espera e média de uso de CPU e memória RAM para diferentes contêineres da carga da ciência de dados.

Detecção – Carro, Rosto, Corpo e Objeto



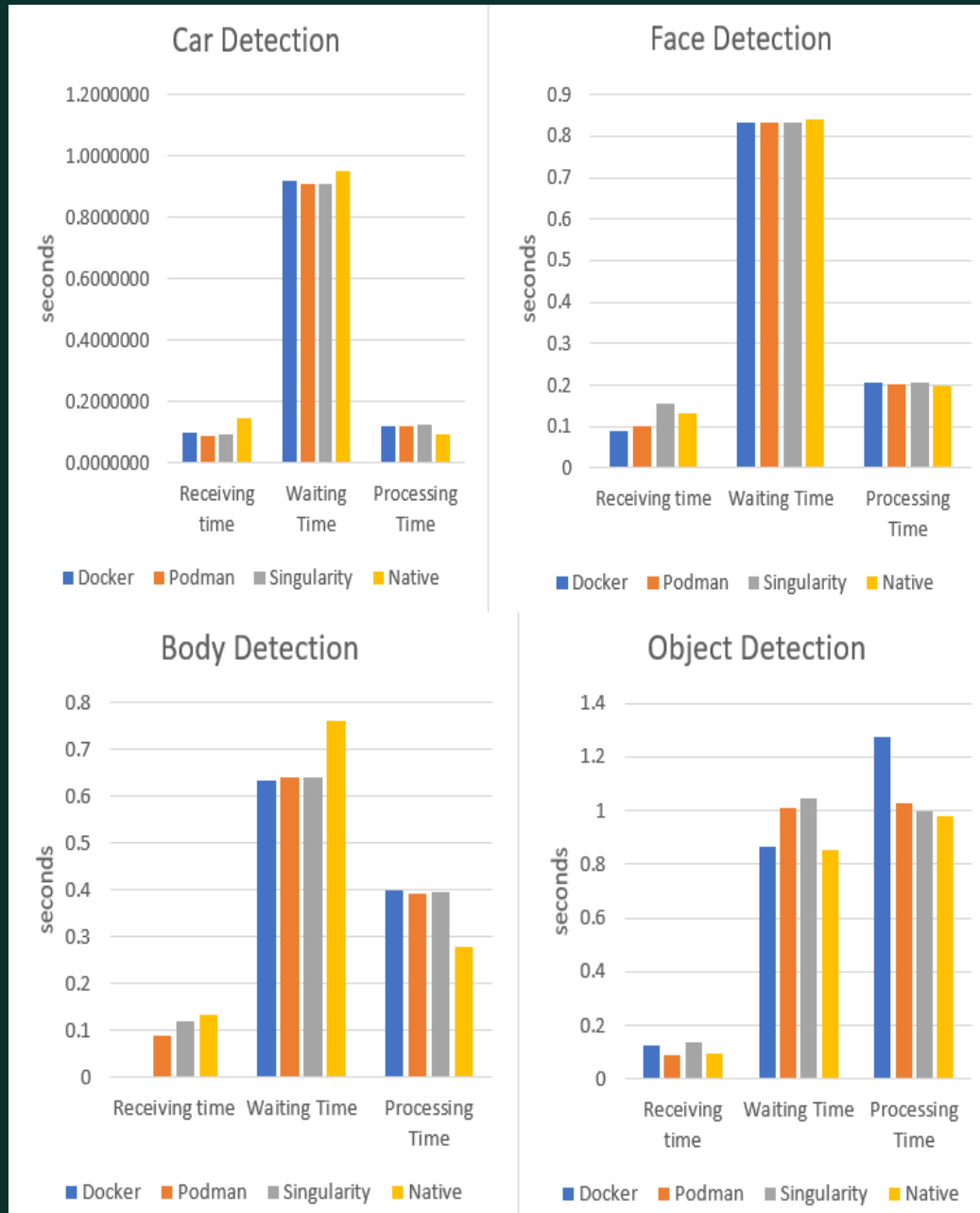
Uso médio de CPU para diferentes contêineres com aplicação de visão computacional em conexão sem fio para sem fio.

Detecção – Carro, Rosto, Corpo e Objeto



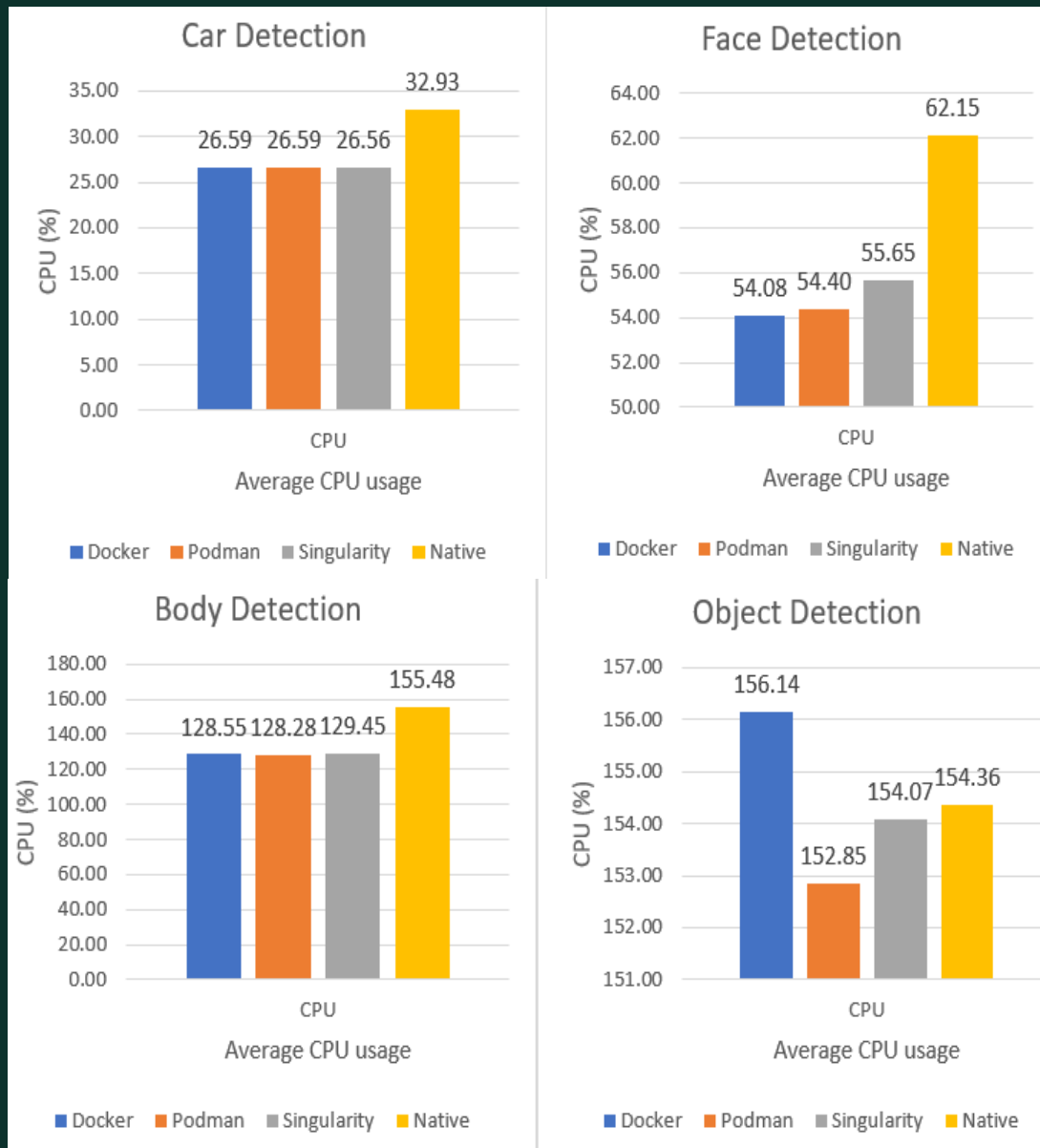
Uso médio de memória RAM para diferentes contêineres com aplicações de visão computacional em conexão sem fio para sem fio.

Detecção – Carro, Rosto, Corpo e Objeto



Tempo médio de recebimento, processamento e espera para diferentes contêineres com aplicação de visão computacional em rede sem fio para sem fio.

Detecção – Carro, Rosto, Corpo e Objeto



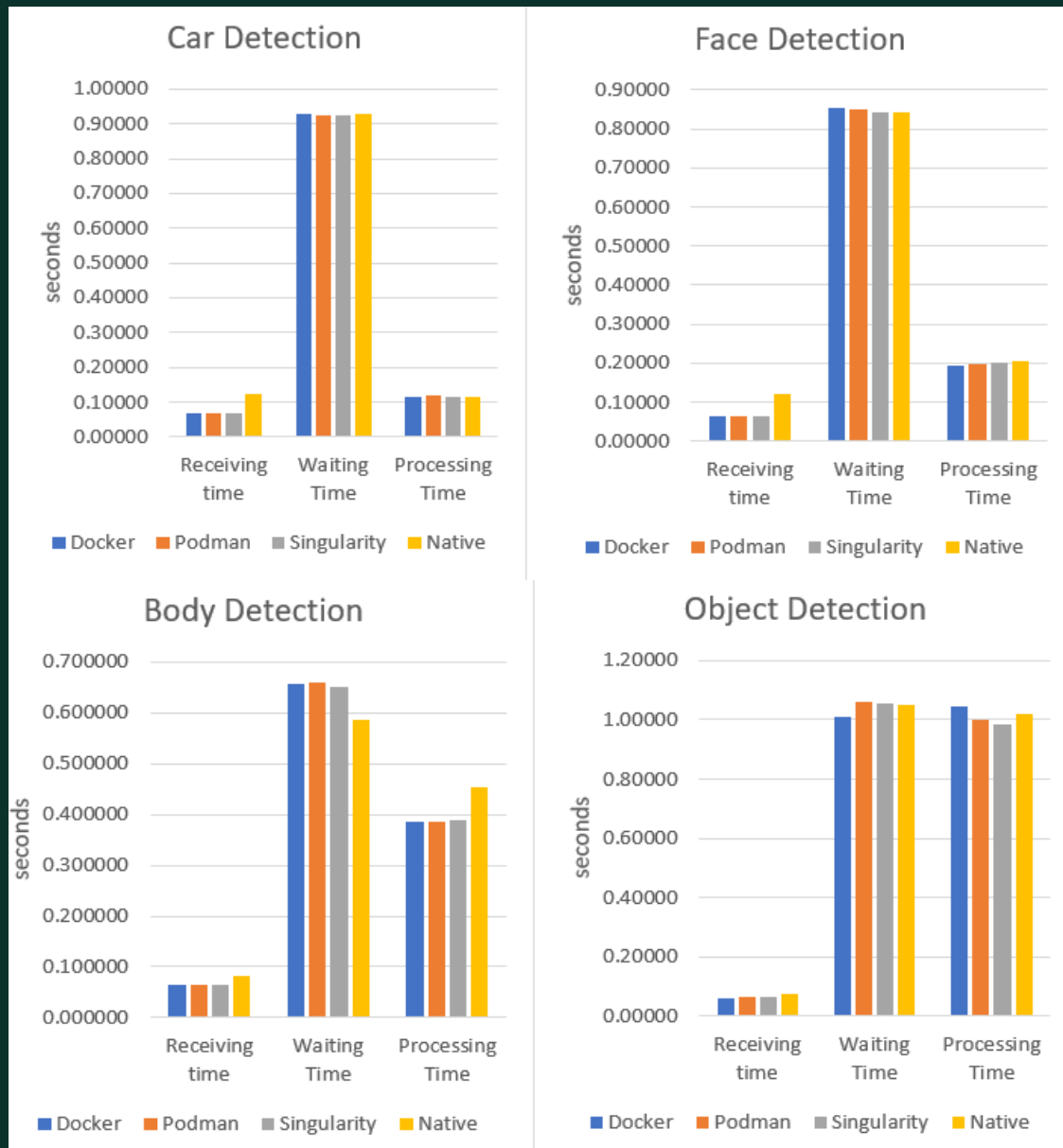
Uso médio de CPU para diferentes contêineres com aplicação de visão computacional em rede totalmente cabeada.

Detecção – Carro, Rosto, Corpo e Objeto



Uso médio de memória RAM para diferentes contêineres com aplicação de visão computacional em rede totalmente cabeada.

Detecção – Carro, Rosto, Corpo e Objeto



Tempo médio de recebimento, processamento e espera para diferentes contêineres com aplicação de visão computacional em rede totalmente cabeada.

Sumarização

Aplicação0							
	Critério	Métricas de desempenho	Docker	Podman	Singularity	Nativo	Melhor
Detecção de veículos	Menor é melhor	Tempo de espera (seg.)	0,92	0,91	0,90	0,95	Singularity
	Menor é melhor	Tempo de recebimento (seg)	0,10	0,09	0,08	0,15	Singularity
	Menor é melhor	Tempo de processamento (seg)	0,12	0,12	0,13	0,09	Docker
	Menor é melhor	Uso de CPU (%)	26,67	27,33	26,84	25,03	Docker
	Menor é melhor	Uso de memória RAM (MB)	90,17	93,24	95,68	79,36	Docker
Detecção de rostos	Menor é melhor	Tempo de espera (seg.)	0,83	0,83	0,83	0,84	Empate
	Menor é melhor	Tempo de recebimento (seg)	0,09	0,10	0,16	0,13	Docker
	Menor é melhor	Tempo de processamento (seg)	0,21	0,2	0,2	0,19	Empate
	Menor é melhor	Uso de CPU (%)	51,45	54,55	53,85	58,49	Docker
	Menor é melhor	Uso de memória RAM (MB)	91,21	94,71	96,34	79,02	Docker
Detecção de corpo	Menor é melhor	Tempo de espera (seg.)	0,63	0,64	0,64	0,76	Docker
	Menor é melhor	Tempo de recebimento (seg)	0,08	0,09	0,12	0,13	Docker
	Menor é melhor	Tempo de processamento (seg)	0,4	0,39	0,40	0,28	Podman
	Menor é melhor	Uso de CPU (%)	130,58	128,1	126,48	90,72	Singularity
	Menor é melhor	Uso de memória RAM (MB)	80,59	81,63	84,44	79,54	Docker
Detecção de objeto	Menor é melhor	Tempo de espera (seg.)	0,86	1,01	1,05	0,85	Docker
	Menor é melhor	Tempo de recebimento (seg)	0,13	0,09	0,14	0,09	Podman
	Menor é melhor	Tempo de processamento (seg)	1,27	1,03	0,99	0,98	Singularity
	Menor é melhor	Uso de CPU (%)	160,24	155,84	148,88	143,65	Singularity
	Menor é melhor	Uso de memória RAM (MB)	166,65	165,95	167,30	160,32	Podman
Carga de ciência de dados	Menor é melhor	Tempo de espera (seg.)	0,99	0,98	0,99	0,99	Empate
	Menor é melhor	Tempo de recebimento (seg)	0,001	0,001	0,001	0,001	Empate
	Menor é melhor	Tempo de processamento (seg)	0,002	0,002	0,001	0,001	Singularity
	Menor é melhor	Uso de CPU (%)	0,28	0,27	0,24	0,25	Singularity
	Menor é melhor	Uso de memória RAM (MB)	70,84	72,51	72,74	50,91	Docker

Conclusão pessoal

Aprendizados

- Apesar dos contêineres utilizarem a mesma imagem base, o diferentes tecnologias causaram diferentes resultados;
- O estabelecimento de métricas claras permite uma melhor análise dos dados;
- A tecnologia de contêiner Singularity, designada para HPC (*High Performance Computing*), teve um dos melhores desempenhos, juntamente com o Docker;
- Dependendo do cenário, diferentes métricas podem ser determinantes para a escolha da tecnologia de contêiner utilizada, ou seja, em ambientes com restrição de memória RAM, opta-se pelos que possuíram o menor consumo. O mesmo vale para o uso de CPU;
- Na avaliação da rede, casos como a detecção de rosto e carro, os tempos foram muito próximos tanto para comunicação sem fio para sem fio, quando em um cenário de rede totalmente cabeada.