



Trabalho Final

**CONSTRUINDO APLICAÇÕES DE LARGA
ESCALA
COM FRAMEWORKS DE PROGRAMAÇÃO
PARALELA/DISTRIBUÍDA**

**ANTONIO ALDISIO - 20/2028211
FERNANDO MIRANDA - 19/0106566
LORRANY SOUZA - 18/0113992**

AGENDA

01 INTRODUÇÃO

02 OBJETIVO

03 METODOLOGIA

04 REQUISITOS DA
PERFORMANCE

05 REQUISITOS DA
ELASTICIDADE

06 ANÁLISE DOS
RESULTADOS

07 CONCLUSÃO

Jogo da vida **INTRODUÇÃO**

- Criado por John Conway em 1968
- Visa projetar conjunto de regras matemáticas simples que é capaz de formar padrões complexos de vida
- Faz a simulação das gerações sucessivas de uma sociedade de organismos vivos.

Jogo da vida **INTRODUÇÃO**

- Regras:
 - Células vivas com menos de 2 vizinhas vivas morrem por abandono
 - Células vivas com mais de 3 vizinhas vivas morrem de superpopulação
 - Células mortas com exatamente 3 vizinhas vivas tornam-se vivas
 - As demais células mantêm seu estado anterior.

Jogo da vida

INTRODUÇÃO

- Antigamente era desenvolvido normalmente com quadros negros, papel, lápis, menos com o uso do computador.
- Nos anos 70, analistas resolveram utilizar os mainframes da IBM para criar algoritmos aptos a reproduzir o jogo computacionalmente.
- O algoritmo precisava de muita capacidade computacional e demorava até uma noite inteira para rodar.

Trabalho final **OBJETIVO**

Construir uma aplicação para ela se comportar como uma aplicação de larga escala atendendo os requisitos de performance e de escala.

Trabalho final
METODOLOGIA

Objetivos: pesquisa exploratória

Abordagem: quantitativa

Procedimento adotado: bibliográfica

REQUISITOS DE PERFORMANCE

Requisitos de **PERFORMANCE**

O requisito de performance é fundamental no desenvolvimento de sistemas e aplicações, pois define critérios essenciais para o desempenho do software. Ele abrange diversos aspectos, como tempo de resposta, capacidade, escalabilidade e eficiência de recursos.

Requisitos de **PERFORMANCE**

- Apache Spark
- MPI
- OpenMP

Performance **APACHE SPARK**

- O Spark é uma plataforma de processamento de dados de código aberto, projetada para realizar análise e processamento de grandes volumes de dados em escala distribuída.
- Sua arquitetura possui o RDD (Resilient Distributed Dataset), que permite que os dados sejam distribuídos por vários nós de um cluster de computadores, garantindo assim a disponibilidade dos dados mesmo em casos de falhas de hardware.

JOGO DA VIDA COM SPARK

```
if __name__ == "__main__":
    sc = SparkContext(appName="GameOfLife")

    powmin = 2
    powmax = 4

    for pow in range(powmin, powmax + 1):
        tam = 1 << pow

        t0 = wall_time()
        tabulIn, tabulOut = InitTabul(tam)
        t1 = wall_time()

        iterations = 2 * (tam - 3)
        for _ in range(iterations):

            broadcasted_tabulIn = sc.broadcast(tabulIn)

            rdd = sc.parallelize(range(1, tam + 1))

            rdd.foreach(lambda i: UmaVida((broadcasted_tabulIn.value, tabulOut, tam, i)))

            tabulIn, tabulOut = tabulOut, tabulIn

        t2 = wall_time()
```

JOGO DA VIDA COM SPARK

```
is_correct = Correto(tabulIn, tam)
global_is_correct = sc.parallelize([is_correct]).reduce(lambda x, y: x and y)

if sc.getConf().get('spark.driver.host') == 'localhost':
    if global_is_correct:
        print("***Ok, RESULTADO CORRETO**")
    else:
        print("***Nok, RESULTADO ERRADO**")

t3 = wall_time()
print("-----RESULTADO-----")
print("tam=%d; tempos: init=%7.7f, comp=%7.7f, fim=%7.7f, tot=%7.7f" %
      (tam, t1 - t0, t2 - t1, t3 - t2, t3 - t0))
print("-----RESULTADO-----\n\n")
```

```
sc.stop()
```

Performance **OPENMP**

- O OpenMP (Open Multi-Processing) é uma API (Interface de Programação de Aplicativos) de programação paralela que foi projetada para facilitar a criação de aplicações que aproveitam o poder de processadores multicore e multiprocessadores

JOGO DA VIDA COM OPENMP

```
void UmaVida(int* tabulIn, int* tabulOut, int tam) {
    int i, j, vizviv;

    #pragma omp parallel for private(i, j, vizviv) shared(tabulIn, tabulOut)
    for (i=1; i<=tam; i++) {
        for (j=1; j<=tam; j++) {
            vizviv = tabulIn[ind2d(i-1,j-1)] + tabulIn[ind2d(i-1,j  )] +
                    tabulIn[ind2d(i-1,j+1)] + tabulIn[ind2d(i  ,j-1)] +
                    tabulIn[ind2d(i  ,j+1)] + tabulIn[ind2d(i+1,j-1)] +
                    tabulIn[ind2d(i+1,j  )] + tabulIn[ind2d(i+1,j+1)];
            if (tabulIn[ind2d(i,j)] && vizviv < 2)
                tabulOut[ind2d(i,j)] = 0;
            else if (tabulIn[ind2d(i,j)] && vizviv > 3)
                tabulOut[ind2d(i,j)] = 0;
            else if (!tabulIn[ind2d(i,j)] && vizviv == 3)
                tabulOut[ind2d(i,j)] = 1;
            else
                tabulOut[ind2d(i,j)] = tabulIn[ind2d(i,j)];
        } /* fim-for */
    } /* fim-for */
} /* fim-UmaVida */
```

Performance **MPI**

- MPI (Message Passing Interface) é uma especificação de biblioteca de programação amplamente utilizada para o desenvolvimento de aplicações paralelas e distribuídas.

JOGO DA VIDA COM MPI

```
int main(int argc, char** argv) {  
    int pow, rank = 0, size = 0;  
    int i, tam, *tabulIn, *tabulOut;  
    char msg[9];  
    double t0, t1, t2, t3;  
  
    MPI_Init(&argc, &argv);  
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);  
    MPI_Comm_size(MPI_COMM_WORLD, &size);
```

JOGO DA VIDA COM MPI

```
for (pow = POWMIN; pow <= POWMAX; pow++) {  
    tam = 1 << pow;  
  
    t0 = wall_time();  
  
    tabulIn  = (int*) malloc ((tam+2)*(tam+2)*sizeof(int)*tam);  
    tabulOut = (int*) malloc ((tam+2)*(tam+2)*sizeof(int)*tam);  
  
    InitTabul(tabulIn, tabulOut, tam);  
    t1 = wall_time();  
  
    MPI_Barrier(MPI_COMM_WORLD);  
  
    MPI_Bcast(tabulIn, (tam+2)*(tam+2), MPI_INT, 0, MPI_COMM_WORLD);
```

JOGO DA VIDA COM MPI

```
for (i = 0; i < 2*(tam-3); i++) {
    UmaVida(tabulIn, tabulOut, tam);
    UmaVida(tabulOut, tabulIn, tam);
}

t2 = wall_time();

MPI_Gather(rank == 0 ? MPI_IN_PLACE : tabulIn, (tam+2)*(tam+2), MPI_INT,
          tabulIn, (tam+2)*(tam+2), MPI_INT, 0, MPI_COMM_WORLD);

MPI_Barrier(MPI_COMM_WORLD);

if (rank == 0) {
    if (Correto(tabulIn, tam))
        printf("*RESULTADO CORRETO*\n");
    else
        printf("*RESULTADO ERRADO*\n");

    t3 = wall_time();

    printf("tam=%d; tempos: init=%7.7f, comp=%7.7f, fim=%7.7f, tot=%7.7f \n",
          tam, t1 - t0, t2 - t1, t3 - t2, t3 - t0);
}
```

JOGO DA VIDA COM OPENMP + MPI

```
void UmaVida(int* tabulIn, int* tabulOut, int tam) {
    int i, j, vizviv;

    #pragma omp parallel for private(j, vizviv)
    for (i = 1; i <= tam; i++) {
        for (j = 1; j <= tam; j++) {
            vizviv = tabulIn[ind2d(i-1,j-1)] + tabulIn[ind2d(i-1,j)] +
                    tabulIn[ind2d(i-1,j+1)] + tabulIn[ind2d(i,j-1)] +
                    tabulIn[ind2d(i,j+1)] + tabulIn[ind2d(i+1,j-1)] +
                    tabulIn[ind2d(i+1,j)] + tabulIn[ind2d(i+1,j+1)];
            if (tabulIn[ind2d(i,j)] && vizviv < 2)
                tabulOut[ind2d(i,j)] = 0;
            else if (tabulIn[ind2d(i,j)] && vizviv > 3)
                tabulOut[ind2d(i,j)] = 0;
            else if (!tabulIn[ind2d(i,j)] && vizviv == 3)
                tabulOut[ind2d(i,j)] = 1;
            else
                tabulOut[ind2d(i,j)] = tabulIn[ind2d(i,j)];
        }
    }
}
```

REQUISITOS DE ELASTICIDADE

Requisitos de **ELASTICIDADE**

A elasticidade é um requisito crucial para garantir a escalabilidade dinâmica e adaptativa de sistemas e aplicações em ambientes computacionais modernos. Ela consiste na capacidade do software de ajustar-se automaticamente diante das mudanças na demanda e carga de trabalho, permitindo a expansão ou redução flexível de recursos.

Requisitos de **ELASTICIDADE**

- Kubernetes
- Prometheus
- Grafana
- Kafka
- Kibana

Elasticidade **KUBERNETES**

- O Kubernetes é uma plataforma de orquestração de contêineres de código aberto.
- Sua arquitetura modular e suas funcionalidades permitem que os desenvolvedores e administradores de sistemas criem e gerenciem implantações complexas de contêineres de maneira eficiente, garantindo alta disponibilidade, escalabilidade horizontal e uma resiliência impressionante.

Elasticidade **KUBERNETES**

- Criação de um Dockerfile para cada versão do jogo da vida.
- Criação do kubernetes na nuvem com o DigitalOcean.

SCRIPT CRIAÇÃO KUBERNETES

```
#!/bin/bash
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-archive-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" |
sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

sudo rm /etc/containerd/config.toml
sudo systemctl restart containerd

# Master
kubeadm config images pull
sudo kubeadm init
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubectl create -f https://docs.projectcalico.org/manifests/calico.yaml
```

Elasticidade **PROMETHEUS**

- É um sistema de monitoramento e alerta de código aberto, amplamente utilizado na comunidade de infraestrutura de TI e DevOps.

Elasticidade **GRAFANA**

- É uma plataforma de análise e visualização de dados de código aberto, altamente popular entre engenheiros de sistemas e analistas de dados.

Elasticidade **KAFKA**

- Kafka é uma plataforma de streaming de dados distribuída e de código aberto, desenvolvida para atender aos desafios de lidar com grandes volumes de dados em tempo real.

Elasticidade **KIBANA**

- É uma plataforma de visualização e exploração de dados projetada para trabalhar em conjunto com o Elasticsearch, que é um mecanismo de busca e análise de dados em tempo real.

ANÁLISE DOS RESULTADOS

PERFORMANCE

MPI X OPENMP

```
**RESULTADO CORRETO**
tam=8; tempos: init=0.0000072, comp=0.0004809, fim=0.0000432, tot=0.0005312
**RESULTADO CORRETO**
tam=16; tempos: init=0.0000122, comp=0.0005910, fim=0.0000110, tot=0.0006142
**RESULTADO CORRETO**
tam=32; tempos: init=0.0000198, comp=0.0025001, fim=0.0000041, tot=0.0025239
**RESULTADO CORRETO**
tam=64; tempos: init=0.0000148, comp=0.0029690, fim=0.0000091, tot=0.0029929
**RESULTADO CORRETO**
tam=128; tempos: init=0.0000658, comp=0.0221212, fim=0.0000339, tot=0.0222208
**RESULTADO CORRETO**
tam=256; tempos: init=0.0003059, comp=0.1776402, fim=0.0001268, tot=0.1780729
**RESULTADO CORRETO**
tam=512; tempos: init=0.0011899, comp=1.4168899, fim=0.0004990, tot=1.4185789
**RESULTADO CORRETO**
tam=1024; tempos: init=0.0046749, comp=11.5849011, fim=0.0019889, tot=11.5915649
```

```
*RESULTADO CORRETO*
tam=8; tempos: init=0.0000160, comp=0.0001299, fim=0.0000749, tot=0.0002208
*RESULTADO CORRETO*
tam=16; tempos: init=0.0000119, comp=0.0005031, fim=0.0000660, tot=0.0005810
*RESULTADO CORRETO*
tam=32; tempos: init=0.0000391, comp=0.0040669, fim=0.0004270, tot=0.0045331
*RESULTADO CORRETO*
tam=64; tempos: init=0.0000610, comp=0.0191939, fim=0.0001762, tot=0.0194311
*RESULTADO CORRETO*
tam=128; tempos: init=0.0001020, comp=0.0877199, fim=0.0010931, tot=0.0889151
*RESULTADO CORRETO*
tam=256; tempos: init=0.0003040, comp=0.6924989, fim=0.0081141, tot=0.7009170
*RESULTADO CORRETO*
tam=512; tempos: init=0.0011821, comp=5.5893581, fim=0.0485759, tot=5.6391160
*RESULTADO CORRETO*
tam=1024; tempos: init=0.0047510, comp=46.5171821, fim=0.3277030, tot=46.8496361
```

OPENMP

MPI

MPI+OPENMP X SPARK

```
*RESULTADO CORRETO*
tam=8; tempos: init=0.0000050, comp=0.4657459, fim=0.0302219, tot=0.4959729
*RESULTADO CORRETO*
tam=16; tempos: init=0.0000081, comp=0.2762260, fim=0.6744199, tot=0.9506540
*RESULTADO CORRETO*
tam=32; tempos: init=0.0000210, comp=2.5332839, fim=0.0161819, tot=2.5494869
*RESULTADO CORRETO*
tam=64; tempos: init=0.0000300, comp=5.0677681, fim=0.0065670, tot=5.0743651
*RESULTADO CORRETO*
tam=128; tempos: init=0.0000911, comp=10.0013568, fim=0.0120990, tot=10.0135469
*RESULTADO CORRETO*
tam=256; tempos: init=0.0003290, comp=11.1436789, fim=0.0097809, tot=11.1537888
*RESULTADO CORRETO*
tam=512; tempos: init=0.0030000, comp=41.2540061, fim=0.1178930, tot=41.3748991
```

MPI+OPENMP

```
-----RESULTADO-----
tam=2; tempos: init=0.0000069, comp=0.0000012, fim=1.4233835, tot=1.4233916
-----RESULTADO-----
-----RESULTADO-----
tam=4; tempos: init=0.0000050, comp=0.3049867, fim=0.1665599, tot=0.4715517
-----RESULTADO-----
-----RESULTADO-----
tam=8; tempos: init=0.0000072, comp=1.5053077, fim=0.1317985, tot=1.6371133
-----RESULTADO-----
-----RESULTADO-----
tam=16; tempos: init=0.0000057, comp=3.2093337, fim=0.1520126, tot=3.3613520
-----RESULTADO-----
```

SPARK





ELASTICIDADE

KUBERNETES

Nodes


Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	CPU capacity (cores)	Memory requests (bytes)	Memory limits (bytes)	Memory capacity (bytes)	Pods	Create
<div><div></div><div>kube-f9ruq</div></div>	<div>beta.kubernetes.io/arch: amd64</div> <div>beta.kubernetes.io/instance-type: s-2vcpu-4gb</div> <div>beta.kubernetes.io/os: linux</div> <div>Show all</div>	True	1.10 (58.00%)	502.00m (26.42%)	1.90	2.14Gi (70.32%)	1.77Gi (58.43%)	3.04Gi	13 (11.82%)	<div>a day ago</div> <div></div>
<div><div></div><div>kube-f9ruy</div></div>	<div>beta.kubernetes.io/arch: amd64</div> <div>beta.kubernetes.io/instance-type: s-2vcpu-4gb</div> <div>beta.kubernetes.io/os: linux</div> <div>Show all</div>	True	1.10 (58.00%)	1.40 (73.79%)	1.90	1.48Gi (48.65%)	1.77Gi (58.43%)	3.04Gi	19 (17.27%)	<div>a day ago</div> <div></div>
<div><div></div><div>kube-f9ruf</div></div>	<div>beta.kubernetes.io/arch: amd64</div> <div>beta.kubernetes.io/instance-type: s-2vcpu-4gb</div> <div>beta.kubernetes.io/os: linux</div> <div>Show all</div>	True	802.00m (42.21%)	302.00m (15.89%)	1.90	3.00Gi (98.75%)	2.73Gi (89.74%)	3.04Gi	10 (9.09%)	<div>a day ago</div> <div></div>




KUBERNETES

Pods										
Name	Namespace	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑	
 jogodavida-deployment-65dc4468c7-qklzw	kafka	registry.digitalocean.com/pspd/mpi:0.0.22	<div>app: jogodavida</div> <div>pod-template-hash: 65dc4468c7</div>	kube-f9ruy	Running	0	-	-	3 hours ago	⋮
 kafka-broker-5c779b5d99-dp6hf	kafka	wurstmeister/kafka	<div>app: kafka-broker</div> <div>pod-template-hash: 5c779b5d99</div>	kube-f9ruy	Running	0	-	-	9 hours ago	⋮
 zookeeper-7d46888797-whp9t	kafka	wurstmeister/zookeeper	<div>app: zookeeper</div> <div>pod-template-hash: 7d46888797</div>	kube-f9ruy	Running	0	-	-	9 hours ago	⋮
 prometheus-kube-prometheus-stack-prometheus-0	kube-prometheus-stack	<div>quay.io/prometheus/prometheus:v2.42.0</div> <div>quay.io/prometheus-operator/prometheus-config-reloader</div>	<div>app.kubernetes.io/instance: kube-prometheus-stack-prometheus</div> <div>app.kubernetes.io/managed-by: prometheus-operator</div>	kube-f9ruq	Running	0	-	-	a day ago	⋮

TOTAL: 42 PODS


PROMETHEUS

 Prometheus Alerts Graph Status ▾ Help



Targets

All scrape pools ▾ All Unhealthy Collapse All

 Filter by endpoint or labels

☒ Unknown ☒ Unhealthy ☒ Healthy

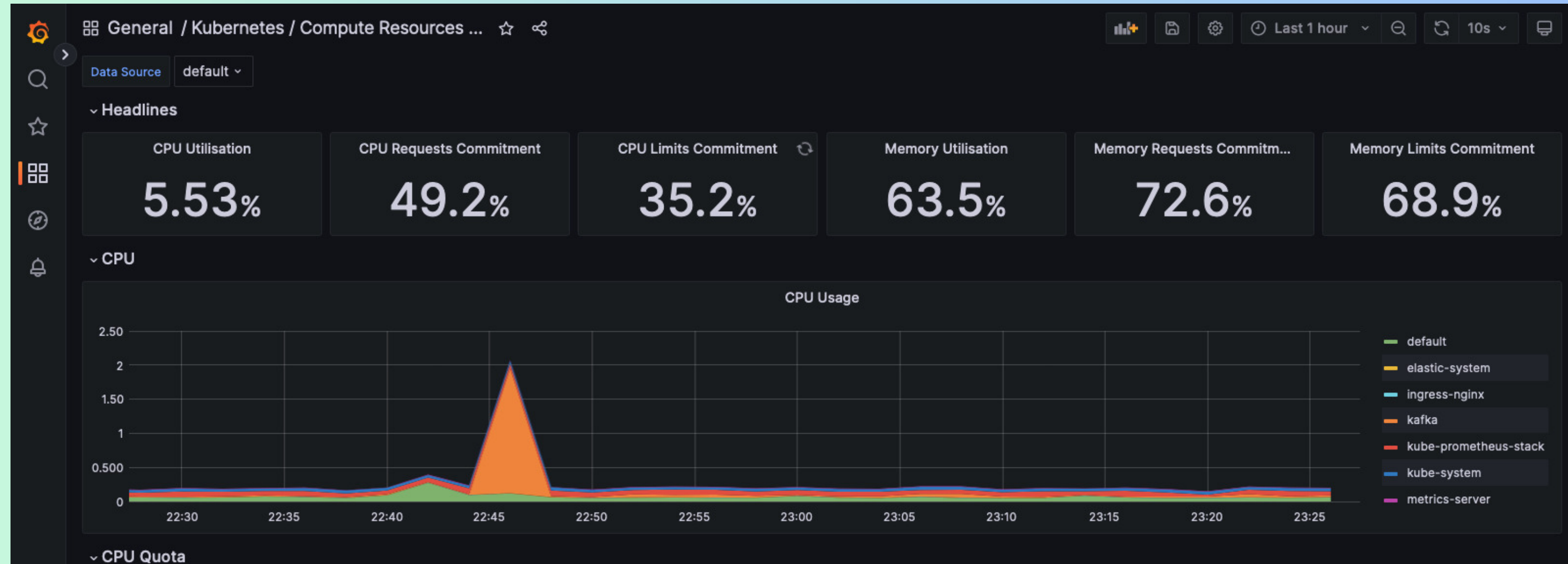
serviceMonitor/kube-prometheus-stack/kube-prometheus-stack-alertmanager/0 (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.244.0.162:9093/metrics	UP	<div>container="alertmanager" endpoint="http-web"</div> <div>instance="10.244.0.162:9093"</div> <div>job="kube-prometheus-stack-alertmanager"</div> <div>namespace="kube-prometheus-stack"</div> <div>pod="alertmanager-kube-prometheus-stack-alertmanager-0"</div> <div>service="kube-prometheus-stack-alertmanager"</div>	25.509s ago	4.706ms	

serviceMonitor/kube-prometheus-stack/kube-prometheus-stack-apiserver/0 (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://100.65.36.105/metrics	UP	<div>endpoint="https" instance="100.65.36.105:443"</div> <div>job="apiserver" namespace="default"</div> <div>service="kubernetes"</div>	12.39s ago	234.241ms	

GRAFANA



KAFKA

```
def call_c_program(mensagem):
    powmin, powmax, codeSelector = mensagem.split()
    powmin = powmin.decode('utf-8')
    powmax = powmax.decode('utf-8')
    codeSelector = codeSelector.decode('utf-8')

    if (codeSelector == 'mpi'):
        os.system(f"OMP_NUM_THREADS=4 mpirun -np 4 ./teste {powmin} {powmax}")
    else:
        os.system(f"python3 jogodavida.py {powmin} {powmax}")
        read_file_and_send_to_es(codeSelector)

def consume_kafka_topic(topic):
    consumer = Consumer({
        'bootstrap.servers': '10.245.179.235:9092',
        'group.id': 'foo',
        'auto.offset.reset': 'earliest',
        'session.timeout.ms': 6000,
    })

    try:
        consumer.subscribe([topic])
        while True:
            msg = consumer.poll(1.0)

            if msg is None:
                continue
```


CONCLUSÃO

OBRIGADO!!