

Manuel Muñoz A.

Lab N°5

Actividad practica

Módulo 5

Intrucciones: Durante esta actividad investigaremos las diferencias de performance entre un programa ejecutado en un contenedor docker y uno ejecutado directamente en una máquina virtual. Realice capturas de pantalla de todos sus pasos.

1. Ayudándose con ejemplos en internet, obtenga un programa en Python que cree un archivo con los números del 1 al 100.000.

Se crea el programa main.py para lo solicitado:

```
import time

inicio = time.time()

nombre_archivo = "./numeros_1_a_100000.txt"

# Creando y escribiendo en el archivo
with open(nombre_archivo, 'w') as archivo:
    for numero in range(1, 100001):
        archivo.write(f"{numero}\n")

fin = time.time()
tiempo_ejecucion = fin - inicio
print(f"Tiempo de ejecución: {tiempo_ejecucion} segundos")
```

2. Ejecute el programa en un contenedor docker y mida los tiempos de ejecución. También ejecute el programa en una maquina EC2 equivalente al contenedor y mida los tiempos de ejecución.

Se crean os maquinas equivalentes EC2 con sistema operativo Ubuntu 22_04, con puerto 22 ssh abierto. Y con llave .pem para acceder.

lab5 ✕ Quitar los filtros < 1 > ⚙								
<input type="checkbox"/>	Name ↗	ID de la instancia	Estado de la instancia ▼	Tipo de instancia ▼	Comprobación de estado	Estado de la alarma	Zona de disponib	
<input type="checkbox"/>	lab5-no-docker	i-07a5b37e267ae76d3	En ejecución 🔍 🔍	t2.micro	🟢 2/2 comprobaciones superadas..	View alarms	+	us-east-1a
<input type="checkbox"/>	lab5-docker	i-0c9f52dad13878622	En ejecución 🔍 🔍	t2.micro	🟢 2/2 comprobaciones superadas..	View alarms	+	us-east-1a

					< 1 >		⚙
▼	Monitoreo ▼	Nombre del grupo de seguridad ▼	Nombre de la clave ▼	Hora de lanzamiento ▼	▼	Detalles... ▼	
	disabled	launch-wizard-1	vockey1	2023/11/26 19:34 GMT-3		Linux/UNIX	
	disabled	launch-wizard-1	vockey1	2023/11/26 19:34 GMT-3		Linux/UNIX	

En la instancia **lab5-docker** se instala docker

```
sudo su

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-
release -y

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io -y

sudo usermod -aG docker $USER
```

Comprobar correcta instalación mediante:

docker version

output:

```
Client: Docker Engine - Community
Version:      24.0.7
API version:  1.43
Go version:   go1.20.10
Git commit:   afdd53b
Built:        Thu Oct 26 09:07:41 2023
OS/Arch:      linux/amd64
Context:      default

Server: Docker Engine - Community
Engine:
Version:      24.0.7
API version:  1.43 (minimum version 1.12)
Go version:   go1.20.10
Git commit:   311b9ff
Built:        Thu Oct 26 09:07:41 2023
OS/Arch:      linux/amd64
```

```
Experimental:      false
containerd:
  Version:         1.6.25
  GitCommit:       d8f198a4ed8892c764191ef7b3b06d8a2eeb5c7f
runc:
  Version:         1.1.10
  GitCommit:       v1.1.10-0-g18a0cb0
docker-init:
  Version:         0.19.0
  GitCommit:       de40ad0
```

Luego clonamos el siguiente repositorio:

<https://github.com/mimunoz11/lab-performance-docker/>

que posee el siguiente contenedor:

```
# Usar una imagen base de Python
FROM python:3.11

# Establecer el directorio de trabajo en el contenedor
WORKDIR /usr/src/app

# Copiar el script de Python al contenedor
COPY . .

# Comando para ejecutar el script
CMD ["python", "./main.py"]
```

Junto al programa main.py

Por último, ejecutamos el programa en el contenedor

```
docker build -t lab5 .
docker run --name lab5 lab5
docker run lab5
```

En la instancia **lab5-no-docker** no se instala docker solo se ejecuta main.py

python3 main.py

Tiempos de ejecución

Podemos ver los tiempos de ejecución

```

Last login: Sun Nov 26 22:48:10 2023 from 186.189.97.44
ubuntu@ip-172-31-26-250:~$ sudo su
root@ip-172-31-26-250:/home/ubuntu# docker run lab5
Tiempo de ejecución: 0.028526782989501953 segundos
root@ip-172-31-26-250:/home/ubuntu# docker run lab5
Tiempo de ejecución: 0.02684187889099121 segundos
root@ip-172-31-26-250:/home/ubuntu# docker run lab5
Tiempo de ejecución: 0.02711176872253418 segundos
root@ip-172-31-26-250:/home/ubuntu# docker run lab5
Tiempo de ejecución: 0.02718067169189453 segundos
root@ip-172-31-26-250:/home/ubuntu# docker run lab5
Tiempo de ejecución: 0.028055667877197266 segundos
root@ip-172-31-26-250:/home/ubuntu# docker run lab5
Tiempo de ejecución: 0.02708578109741211 segundos
root@ip-172-31-26-250:/home/ubuntu# docker run lab5
Tiempo de ejecución: 0.02752685546875 segundos
root@ip-172-31-26-250:/home/ubuntu#

File "/home/ubuntu/lab-performance-docker/main.py", line 8, in <module>
  with open(nombre_archivo, 'w') as archivo:
PermissionError: [Errno 13] Permission denied: './numeros_1_a_100000.txt'
ubuntu@ip-172-31-21-16:~/lab-performance-docker$ sudo su
root@ip-172-31-21-16:/home/ubuntu/lab-performance-docker# python3 main.py
Tiempo de ejecución: 0.02818584442138672 segundos
root@ip-172-31-21-16:/home/ubuntu/lab-performance-docker# python3 main.py
Tiempo de ejecución: 0.028662919998168945 segundos
root@ip-172-31-21-16:/home/ubuntu/lab-performance-docker# python3 main.py
Tiempo de ejecución: 0.029660463333129883 segundos
root@ip-172-31-21-16:/home/ubuntu/lab-performance-docker# python3 main.py
Tiempo de ejecución: 0.02882242202758789 segundos
root@ip-172-31-21-16:/home/ubuntu/lab-performance-docker# python3 main.py
Tiempo de ejecución: 0.02903151512145996 segundos
root@ip-172-31-21-16:/home/ubuntu/lab-performance-docker# python3 main.py
Tiempo de ejecución: 0.02947068214416504 segundos
root@ip-172-31-21-16:/home/ubuntu/lab-performance-docker# python3 main.py
Tiempo de ejecución: 0.02836012840270996 segundos
root@ip-172-31-21-16:/home/ubuntu/lab-performance-docker#

```

Se concluye que docker es más rápido.

4. ¿Es posible utilizar algún servicio para medir los tiempos? ¿O alguna otra telemetría del proceso?

Para medir tiempos y recopilar telemetría de procesos en aplicaciones que se ejecutan en contenedores Docker o en instancias EC2, hay varias herramientas y servicios disponibles. Estos pueden proporcionar información detallada sobre el rendimiento, el uso de recursos y otros aspectos importantes. Aquí hay algunas opciones:

Herramientas Integradas en el Host:

Linux Performance Tools: Si estás ejecutando Docker en un sistema Linux, herramientas como top, htop, vmstat, iostat, y perf pueden ser útiles para monitorear el uso de recursos y el rendimiento en tiempo real.

CloudWatch en AWS para EC2: Si estás utilizando instancias EC2, AWS CloudWatch es una herramienta poderosa para monitorear y recopilar métricas de rendimiento. Puedes monitorear métricas como el uso de CPU, la actividad de red, la actividad del disco y más.

Herramientas de Profiling para Python:

cProfile: Es un módulo integrado en Python que puedes utilizar para realizar un perfil detallado del tiempo de ejecución de tu script. Proporciona información sobre la frecuencia y duración de las llamadas a funciones.

timeit: Útil para medir el tiempo de ejecución de pequeños fragmentos de código con alta precisión.

Herramientas de Monitoreo y Telemetría:

Prometheus y Grafana: Prometheus es un sistema de monitoreo y alerta de código abierto que puede recopilar y almacenar métricas en tiempo real. Grafana se puede utilizar para visualizar esos datos. Puedes configurar ambos para recopilar métricas de contenedores Docker y sistemas operativos.

Datadog: Es una plataforma de monitoreo que proporciona capacidades de observabilidad en tiempo real para aplicaciones en la nube, incluidas las que se ejecutan en contenedores y en EC2.

Herramientas Específicas de Docker:

Docker Stats y Docker Monitoring Tools: docker stats proporciona un resumen en vivo de los recursos que están utilizando tus contenedores. Además, hay herramientas de terceros como cAdvisor o Portainer que ofrecen una interfaz más detallada y amigable para el monitoreo de contenedores.

APM (Application Performance Management) Tools:

Herramientas como New Relic, Dynatrace, o AppDynamics pueden integrarse en tu aplicación para proporcionar análisis en profundidad del rendimiento, trazas de transacciones, y mucho más. Para utilizar estas herramientas, generalmente necesitarás realizar una configuración inicial, como instalar agentes de software en tus contenedores o instancias EC2, configurar la recopilación de métricas y, en algunos casos, modificar tu aplicación para emitir métricas específicas. La elección de las herramientas dependerá de tus necesidades específicas de monitoreo y del entorno en el que estés trabajando.