

DR. GABRIEL LEON PAREDES, PhD

gleon@ups.edu.ec

www.linkedin.com/in/gabrielleonp

Cloud Computing, Smart Cities & High-Performance
Computing

Cuenca, Ecuador

COMPUTO PARALELO



Cloud Computing-Smart Cities-High
Performance Computing



Cloud Computing

Calculo Paralelo - Dr. Gabriel A. León Paredes,
PhD

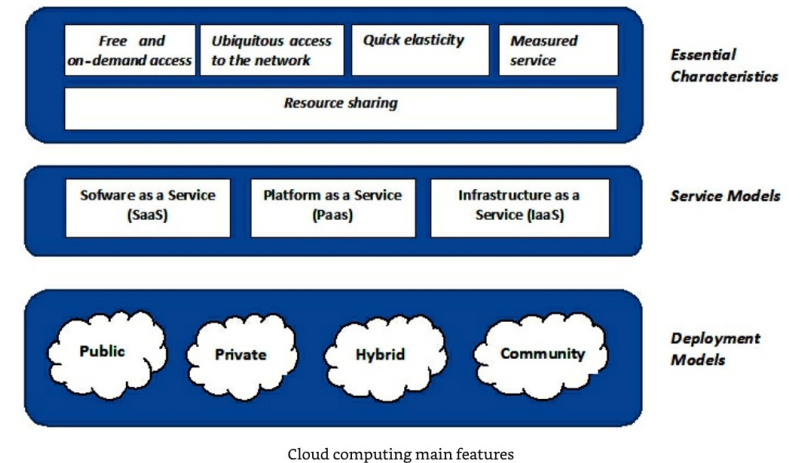
martes, 16 de mayo de 2023

Introducción

- Hay muchas definiciones de computación en la nube, cada una de las cuales tiene diferentes interpretaciones y significados. El Instituto Nacional de Estándares y Tecnología (NIST) ha tratado de proporcionar una explicación detallada y oficial (<https://csrc.nist.gov/publications/detail/sp/800-145/final>):
- “La computación en la nube es un modelo para permitir el acceso de red ubicuo, conveniente y bajo demanda a un grupo compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo de gestión o Interacción del proveedor de servicios. Este modelo de nube se compone de cinco características esenciales, tres modelos de servicio y cuatro modelos de implementación.”
- Características
 - **Acceso gratuito y bajo demanda:** esto permite a los usuarios acceder, a través de interfaces fáciles de usar, a los servicios ofrecidos por el proveedor sin interacción humana.
 - **Acceso ubicuo a la red:** los recursos están disponibles en toda la red y se puede acceder a ellos, a través de dispositivos estándar, como teléfonos inteligentes, tabletas y computadoras personales.
 - **Elasticidad rápida:** es la capacidad de la nube para aumentar o reducir los recursos asignados de manera rápida y automática, como hacer que parezca que son infinitos para el usuario. Esto proporciona una gran escalabilidad al sistema.
 - **Servicio medido:** los sistemas en la nube monitorean constantemente los recursos ofrecidos y los optimizan automáticamente en función del uso estimado. De esta forma, el cliente solo paga por los recursos que realmente se usan en esa sesión en particular.
 - **Uso compartido de recursos:** el proveedor proporciona sus recursos a través de un modelo de múltiples inquilinos para que puedan ser asignados y reasignados dinámicamente, en función de la solicitud del cliente, y utilizados por múltiples consumidores.

Virtualización

- Otra característica (no incluida en la definición de NIST, pero que es la base de la computación en la nube) es el concepto de virtualización. Esta es la posibilidad de ejecutar múltiples sistemas operativos en los mismos recursos físicos, lo que garantiza numerosas ventajas, como la escalabilidad, la reducción de costos y una mayor velocidad para proporcionar nuevos recursos a los clientes.
- Los enfoques más comunes para la virtualización son los siguientes:
 - **Contenedores**
 - **Máquinas virtuales**
- Ambas soluciones tienen casi las mismas ventajas en lo que respecta al aislamiento de las aplicaciones, pero funcionan en diferentes niveles de virtualización porque los contenedores virtualizan el sistema operativo y las máquinas virtuales virtualizan el hardware.
- Esto significa que los contenedores son más portátiles y eficientes. La aplicación más común para virtualizar a través de contenedores es Docker.





Plataformas en la nube

Calculo Paralelo - Dr. Gabriel A. León Paredes,
PhD

martes, 16 de mayo de 2023



Google Cloud Platform

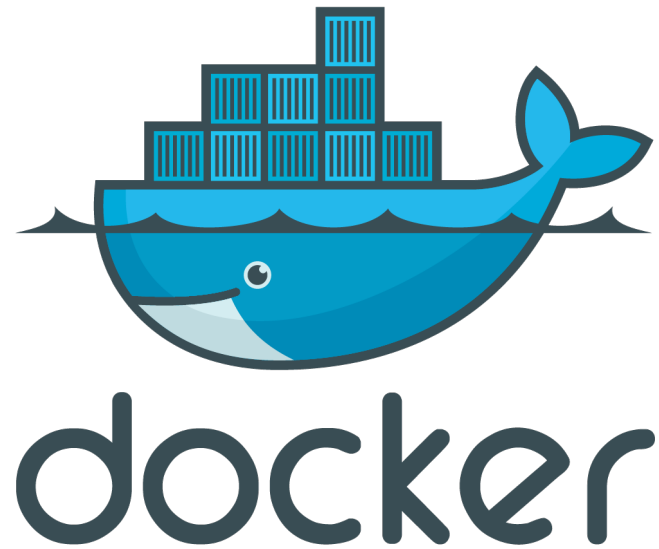
Introducción

- Las plataformas de computación en la nube son conjuntos de software y tecnologías que permiten la entrega de recursos en la nube (recursos a demanda, escalables y virtualizados).
- Entre las plataformas más populares se encuentran las de Google y, por supuesto, el hito de la computación en la nube: Amazon Web Services (AWS).
- Ambos admiten Python como lenguaje de desarrollo.

martes, 16 de mayo de 2023

Calculo Paralelo - Dr. Gabriel A. León Paredes, PhD

Docker

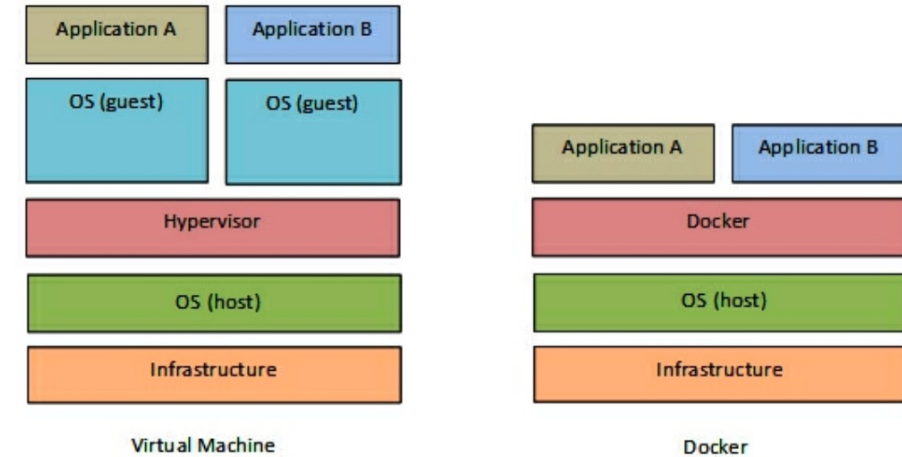


Introducción

- Los contenedores son entornos de virtualización. Incluyen todo lo que necesita el software, a saber: bibliotecas, dependencias, sistemas de archivos e interfaces de red.
- A diferencia de las máquinas virtuales clásicas, todos los elementos mencionados comparten el núcleo con la máquina en la que se están ejecutando. De esta manera, el impacto en el uso de los recursos del nodo host se reduce considerablemente.
- Esto hace que el contenedor sea una tecnología muy atractiva en términos de escalabilidad, rendimiento y aislamiento. Los contenedores no son tecnología joven; tuvieron éxito con el lanzamiento de Docker en 2013. Desde entonces, han revolucionado por completo los estándares utilizados para el desarrollo y la administración de aplicaciones.
- Docker es una plataforma de contenedores basada en la implementación de Linux Containers (LXC), que amplía la funcionalidad de esta tecnología con la capacidad de administrar contenedores como imágenes autocontenidas, y agrega herramientas adicionales para coordinar su ciclo de vida y guardar su estado.
- La idea de la contenedorización es precisamente permitir que una aplicación determinada se ejecute en cualquier tipo de sistema, ya que todas sus dependencias ya están incluidas en el contenedor. De esta manera, la aplicación se vuelve altamente portátil y puede probarse e implementarse fácilmente en cualquier tipo de entorno, tanto local como, sobre todo, en la nube.

Arquitectura

- Pero aunque las máquinas virtuales y contenedores tienen características comunes, son tecnologías profundamente diferentes, de la misma manera, que debemos comenzar a pensar en cómo las arquitecturas de nuestras aplicaciones son diferentes.
 - Una posible arquitectura de software adecuada para una infraestructura de contenedor es la arquitectura clásica de microservicios. La idea es dividir la aplicación en muchos componentes pequeños, cada uno con su propia tarea específica, que puedan intercambiar mensajes y cooperar entre sí. El despliegue de estos componentes tendrá lugar individualmente, en forma de muchos contenedores.
 - En las máquinas virtuales, una herramienta llamada hipervisor se encarga de reservar (estática o dinámicamente) una cierta cantidad de recursos del sistema operativo host para dedicarlos a uno o más sistemas operativos, llamados invitados o hosts. Un sistema operativo invitado estará completamente aislado del sistema operativo host. Este mecanismo es muy costoso en términos de recursos, por lo que la idea de combinar un microservicio con una máquina virtual es completamente imposible.



Microservice architecture in virtual machine and Docker implementation

Instalación

Windows

- La instalación es bastante simple: una vez que haya descargado el instalador, simplemente ejecútelo y listo.
 - <https://docs.docker.com/docker-for-windows/install/>
- El proceso de instalación es generalmente muy lineal. Lo único que necesita atención es la fase final de la instalación, en la que puede ser necesario habilitar las funciones de Hyper-V. Si es así, aceptamos y reiniciamos la máquina.
- Una vez que se reinicia la computadora, el ícono Docker debería aparecer en la bandeja del sistema en la parte inferior derecha de la pantalla.
- Abra el símbolo del sistema o la consola de PowerShell y verifique si todo está bien ejecutando el comando de versión de docker:
 - `docker version`

Ubuntu


- Puede instalar Docker Engine de diferentes maneras, según sus necesidades:
 - <https://docs.docker.com/engine/install/ubuntu/>
- La mayoría de los usuarios configuran los repositorios de Docker e instalan desde ellos, para facilitar la instalación y las tareas de actualización. Este es el enfoque recomendado.
- Algunos usuarios descargan el paquete DEB, lo instalan manualmente y administran las actualizaciones de forma completamente manual. Esto es útil en situaciones como la instalación de Docker en sistemas con espacios sin acceso a Internet.
- En entornos de prueba y desarrollo, algunos usuarios eligen usar scripts de conveniencia automatizados para instalar Docker.

Dockerizando Python

- Imaginemos que queremos implementar una aplicación web simple (usando el módulo Flask) en la dirección localhost, puerto 5000. Crear un espacio de trabajo (path)
 - El primer paso es crear el archivo python, que llamaremos Dockerfile.py, contiene el despliegue de la aplicación Web.
 - El segundo paso es crear un archivo de configuración de Docker.
 - Cambiar la versión a Python:3.5-alpine
 - Cambiar a que instale las dependencias con pip3
 - El tercer paso es crear un archivo de requirements.txt para que se instalen dependencias extras.
 - Agregar flask
- Finalmente, necesitamos crear una imagen de la aplicación dockerize.py:
 - `sudo docker build --tag my-python-app --network host .`
- Esto crea y compila la imagen *my-python-app*.
- Después de construir la imagen my-python-app, puede ejecutarla como un contenedor:
 - `sudo docker run -p 5000:5000 my-python-app`
- Abrir un navegador y solicitar la URL
 - localhost:5000/

```
1  #!/usr/bin/env python3
2
3  from flask import Flask
4  app = Flask(__name__)
5  @app.route("/")
6  def hello():
7      return "Hello World!"
8  if __name__ == "__main__":
9      app.run(host="0.0.0.0", port=int("5000"), debug=True)
```

```
1  FROM python:alpine3.7
2  COPY . /app
3  WORKDIR /app
4  RUN pip install -r requirements.txt
5  EXPOSE 5000
6  CMD python ./dockerize.py
```



Docker + PyCUDA

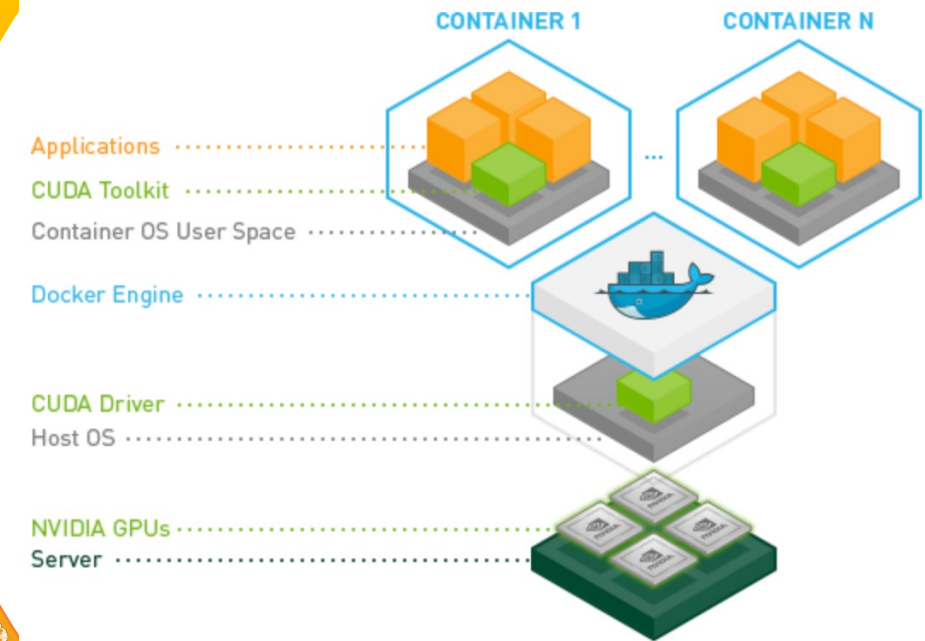
Calculo Paralelo - Dr. Gabriel A. León Paredes,
PhD

martes, 16 de mayo de 2023

Nvidia Container Toolkit

NVIDIA Container Toolkit permite a los usuarios construir y ejecutar contenedores Docker acelerados por GPU. El kit de herramientas incluye una biblioteca de tiempo de ejecución de contenedor y utilidades para configurar automáticamente los contenedores para aprovechar las GPU de NVIDIA. La documentación completa y las preguntas frecuentes están disponibles en el wiki del repositorio.

<https://github.com/NVIDIA/nvidia-docker>



Contenedores Docker para CUDA

Se proporcionan tres tipos de imágenes:

- **base:** incluye el tiempo de ejecución CUDA (cudart)
- **runtime:** se basa en la base e incluye las bibliotecas de matemáticas CUDA y NCCL. Una imagen runtime también incluye cuDNN.
- **devel:** se basa en el tiempo de ejecución e incluye encabezados, herramientas de desarrollo para crear imágenes CUDA. Estas imágenes son particularmente útiles para compilaciones de múltiples etapas.

Los Dockerfiles para las imágenes son de código abierto y tienen licencia bajo BSD de 3 cláusulas.

Para obtener más información, <https://hub.docker.com/r/nvidia/cuda/>

Versiones:

- <https://gitlab.com/nvidia/container-images/cuda/blob/master/doc/supported-tags.md>

Ejemplo (Test nvidia-smi with the 10.0-base CUDA image)

- `docker run --gpus all nvidia/cuda:10.0-base nvidia-smi`

Dockerizando PyCUDA

- Imaginemos que queremos implementar una aplicación web simple (usando el módulo Flask) en la dirección localhost, puerto 5000. Crear un espacio de trabajo (hola_mundo_cuda)
 - El primer paso es crear el archivo Python (Dockerfile.py) dentro del espacio de trabajo que llamaremos
 - El segundo paso es crear un archivo de configuración de Docker.
 - Agregar configuraciones para el uso de CUDA y PyCUDA
- Finalmente, necesitamos crear una imagen de la aplicación dockerize.py:
 - `sudo docker build --tag my-python-cuda-app --network host . hola_mundo_cuda`
- Esto crea y compila la imagen *my-Python-cuda-app*.
- Después de construir la imagen my-python-app, puede ejecutarla como un contenedor:
 - `sudo docker run --gpus all -p 5000:5000 my-Python-cuda-app`
- Abrir un navegador y solicitar la URL
 - localhost:5000/

```
dockerize.py
FROM nvidia/cuda:10.0-cudnn7-devel-ubuntu18.04

RUN apt-get -qq update && \
    apt-get -qq install build-essential && \
    python3-pip && \
    pip3 install pycuda && \
    pip3 install flask

COPY . /app
WORKDIR /app
EXPOSE 5000

CMD python3 ./dockerize.py
```

```
#!/usr/bin/env python3
```

```
from flask import Flask
import pycuda.driver as drv
drv.init()
```

```
app = Flask(__name__)
@app.route("/")
def hello():
    res = "%d device(s) found." % (drv.Device.count())
    for ordinal in range(drv.Device.count()):
        dev = drv.Device(ordinal)
        res += ", Device #%d: %s" % (ordinal, dev.name())
        res += ", Compute Capability: %d.%d" % (dev.compute_capability())
        res += ", Total Memory: %s KB" % (dev.total_memory()//(1024))
    return res
```

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=int("5000"), debug=True)
```

Referencias

- Zaccone, Giancarlo. Python Parallel Programming Cookbook: Over 70 recipes to solve challenges in multithreading and distributed system with Python 3, 2nd Edition . Packt Publishing.