A close-up of a logo

Description automatically generated

**Práctica 1 - Implementación Sistema Distribuido Naive**

**Javier Escobar Serrano**

**Grupo A**

**Arquitectura Big Data**

**3º Grado en Ingeniería Matemática e Inteligencia Artificial**

# Índice

[Introducción 3](#_Toc177138896)

[Metodología 4](#_Toc177138897)

[Resultados 6](#_Toc177138898)

[Conclusión 9](#_Toc177138899)

# Introducción

### Contexto.

Esta es la primera práctica de la asignatura de Big Data. Actúa como introducción al problema de un sistema distribuido.

### Problema.

Se trata de resolver el problema de un sistema distribuido de estructura master-slave.

### Objetivos.

La practica consta de un master que tiene un array de 100 números ordenados que quiere sumar.

Para ello hace uso de 2 slaves a los que reparte la tarea de manera equitativa.

Tras mandarle la mitad del array, estos lo reciben y suman.

Finalmente los slaves devuelven el numero y el master lo suma.

El objetivo de la practica es manejar de manera adecuada sockets con varios nodos.

# Metodología

### Entorno de desarrollo.

Para esta practica he utilizado Python 3.12 junto a la librería de socket y select para conectar los nodos.

Para ejecutarlo he utilizado el sistema distribuido del cluster de ICAI con un sistema operativo de Ubuntu.

### Diseño de la solución.

El sistema consta de un master que controla 2 slaves.

A yellow rectangular sign with white text

Description automatically generated

Slave 2

(10.0.1.43)

De esta manera, se separa la tarea en dos partes iguales para utilizar expansión horizontal.

Tras esto, se manda cada mitad del array a cada slave.

A computer screen shot of code

Description automatically generated

Tras esto, cada socket suma los numeros del array y devuelve el resultado.

A screenshot of a computer code

Description automatically generated

Finalmente, el maestro recibe los resultados y los suma, imprimiendo por pantalla el resultado.

Para hacerlo de manera simultanea he hecho uso de select para que se ejecute de manera paralela.

A screenshot of a computer code

Description automatically generated

### Pruebas realizadas.

Para probar el programa en local, se ha intentado usar la ip de loopback: 127.0.0.1 como valor de la host.

Esta IP es una dirección especial que un host utiliza para dirigir el tráfico hacia sí mismo. De este modo, especificando HOST=‘127.0.0.1’ tanto en master.py como en slave.py y ejecutando cada uno en una terminal, podréis simular el entorno del clúster en vuestro ordenador y testear cambios que hagáis en los scripts conforme los estéis desarrollando. De este modo, os ahorráis tener que subir el script por SCP al clúster para testear cada cambio que hagáis.

Sin embargo, para este problema no es posible ya que no se puede usar 2 veces la ip de 127.0.0.1. Por ello se comprueba directamente en el cluster.

# Resultados

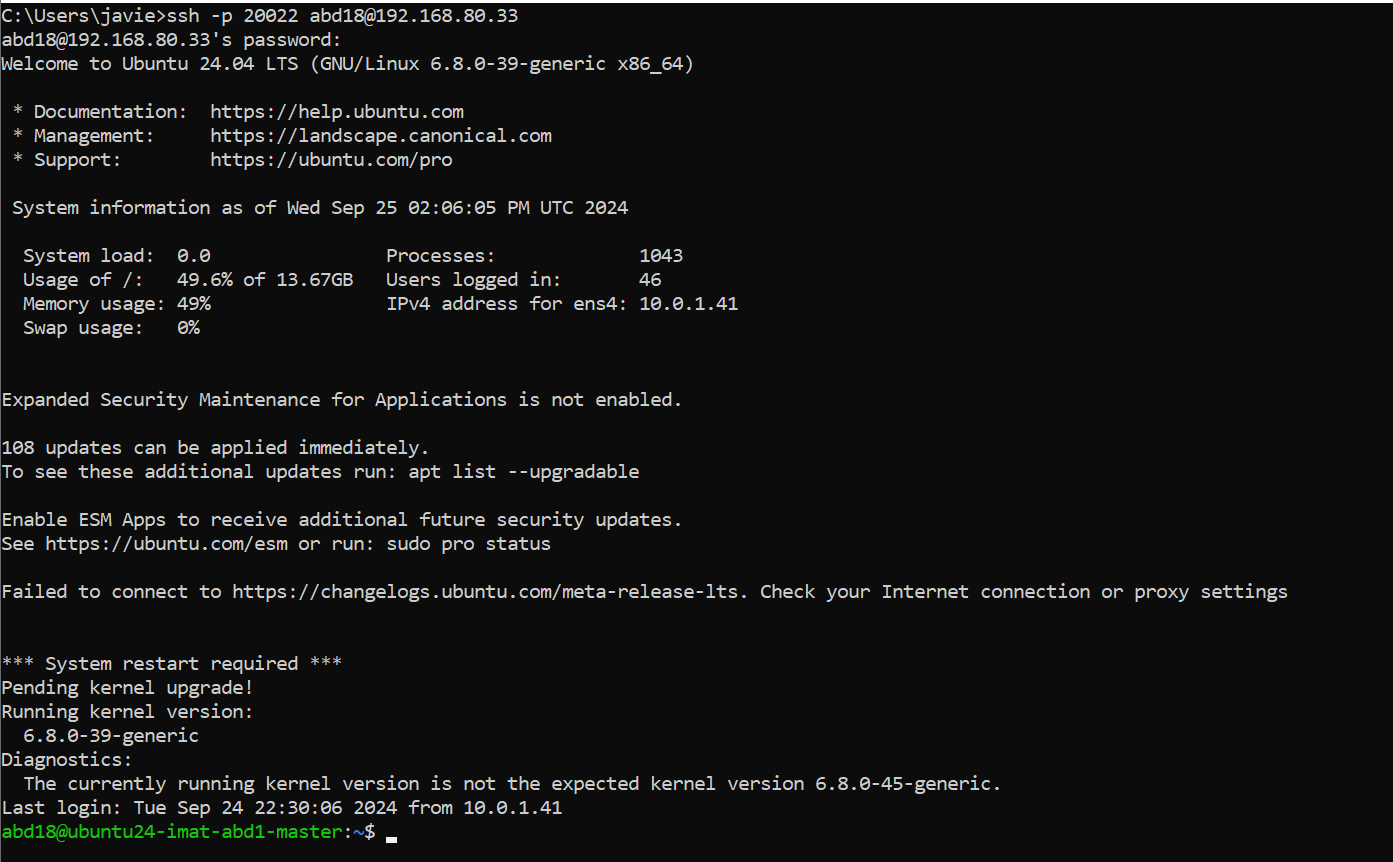
### Descripción de los resultados.

Tras ejecutar en los nodos, tenemos el resultado de 5050. El primero seria 1275 y el segundo 3775.

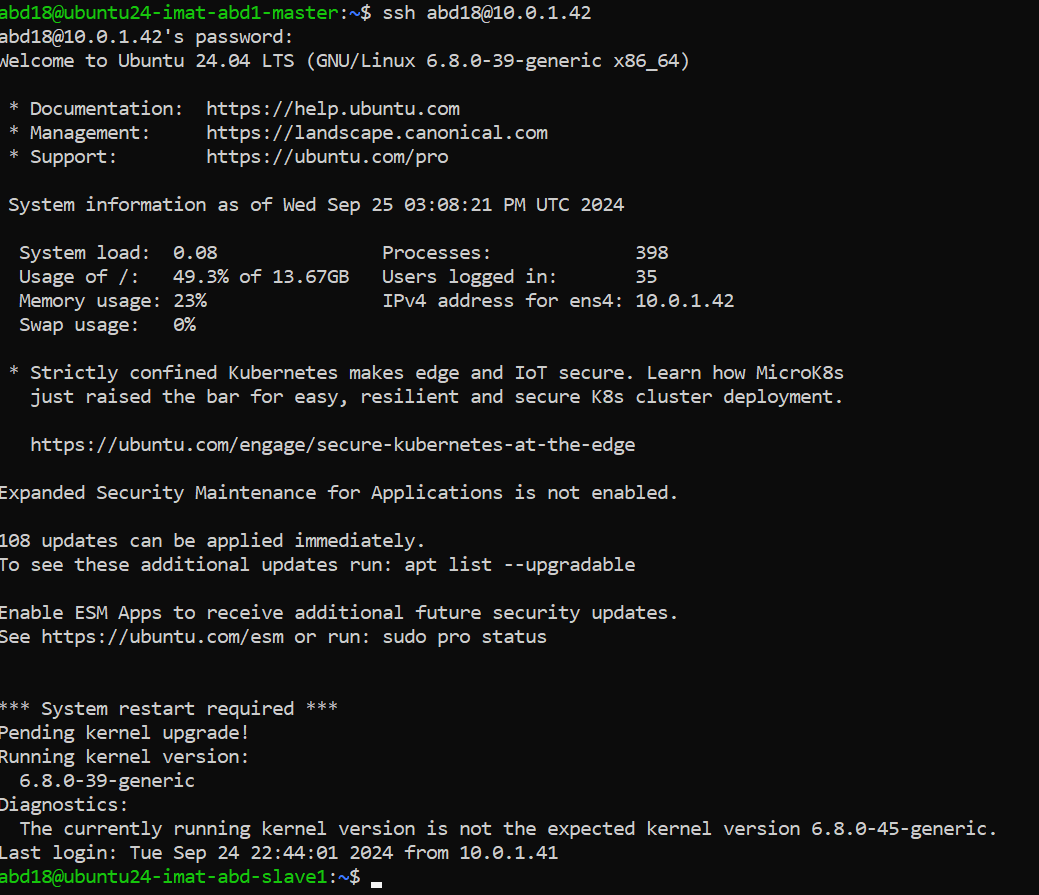
### Pantallazos de la ejecución.

Primero me conecto a cada uno de los nodos.

Master:



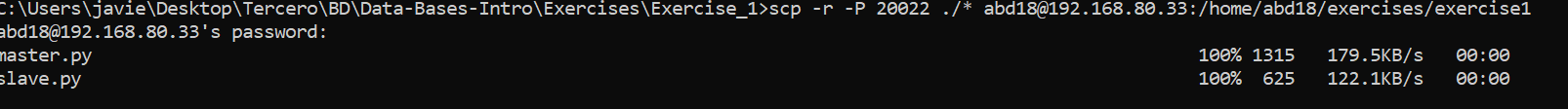
Slaves:



**A screenshot of a computer program

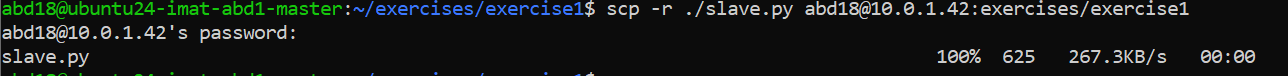
Description automatically generated**

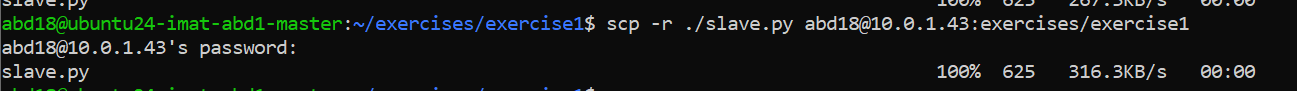
Finalmente le mando los archivos, primero al master y desde el master a cada slave.



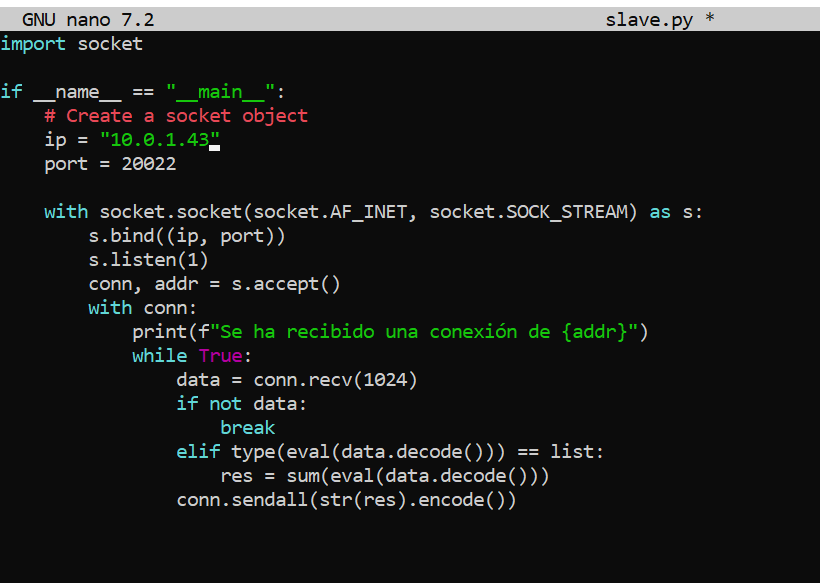


Tras esto, se lo envio desde el master a cada slave.



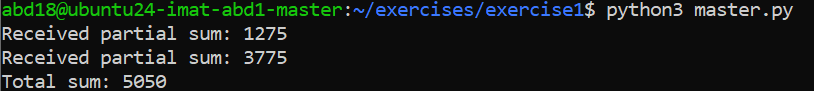


Antes de ejecutar, debemos cambiar la ip usando nano slave.py



Finalmente, se ejecuta:  



# Conclusión

Tras la practica he comprendido la dificultad de utilizar los sistemas distribuidos. El código más básico se convierte en algo complejo y la necesidad de librerías adicionales le añade una capa extra de abstracción.

Sin embargo, la utilidad de usar un sistema como este es bastante visible. Aunque en esta práctica hemos usado un ejemplo bastante sencillo, se puede ver la utilidad en tareas más complejas.

Finalmente, se ha obtenido el resultado deseado.