Big Data

Actualmente los procesadores tienen uno o varios cores, donde cada core puede realizar distintas tareas. En general una descripción simple sobre como funcionan las dos opciones son las siguientes :

1. **Procesadores de un solo núcleo:** Estos procesadores tienen un único núcleo de procesamiento. Imagina que es como tener un trabajador en una fábrica que puede hacer una tarea a la vez. Si tienes múltiples tareas para realizar, este trabajador las hará una tras otra en secuencia. Es eficiente para tareas que no requieren un gran paralelismo, como navegar por la web, escribir documentos o reproducir videos.

Ahora, hablemos de los procesadores con varios núcleos:

1. **Procesadores multinúcleo:** Estos tienen más de un núcleo, lo que les permite realizar múltiples tareas simultáneamente. Siguiendo con la analogía de la fábrica, sería como tener varios trabajadores, cada uno con su propia área de trabajo. Pueden colaborar en tareas complejas o manejar varias tareas simples al mismo tiempo. Por ejemplo, podrían renderizar un video, ejecutar un juego y descargar archivos simultáneamente sin afectar significativamente el rendimiento.

Dentro de los procesadores multinúcleo, hay diferentes configuraciones:

- **Simétrico de múltiples núcleos (SMP):** Todos los núcleos tienen la misma jerarquía y capacidad de ejecución. Esto significa que cada núcleo puede realizar cualquier tarea asignada.
- **Asimétrico de múltiples núcleos (AMP):** En esta configuración, los núcleos pueden tener diferentes capacidades y roles. Algunos podrían ser más potentes y estar dedicados a tareas intensivas, mientras que otros podrían ser más eficientes energéticamente y utilizarse para tareas simples.

Los procesadores multinúcleo son muy comunes en la actualidad, ya que permiten un mejor aprovechamiento de los recursos y un rendimiento más eficiente, especialmente en entornos donde se ejecutan múltiples aplicaciones o tareas exigentes al mismo tiempo.

Pandas

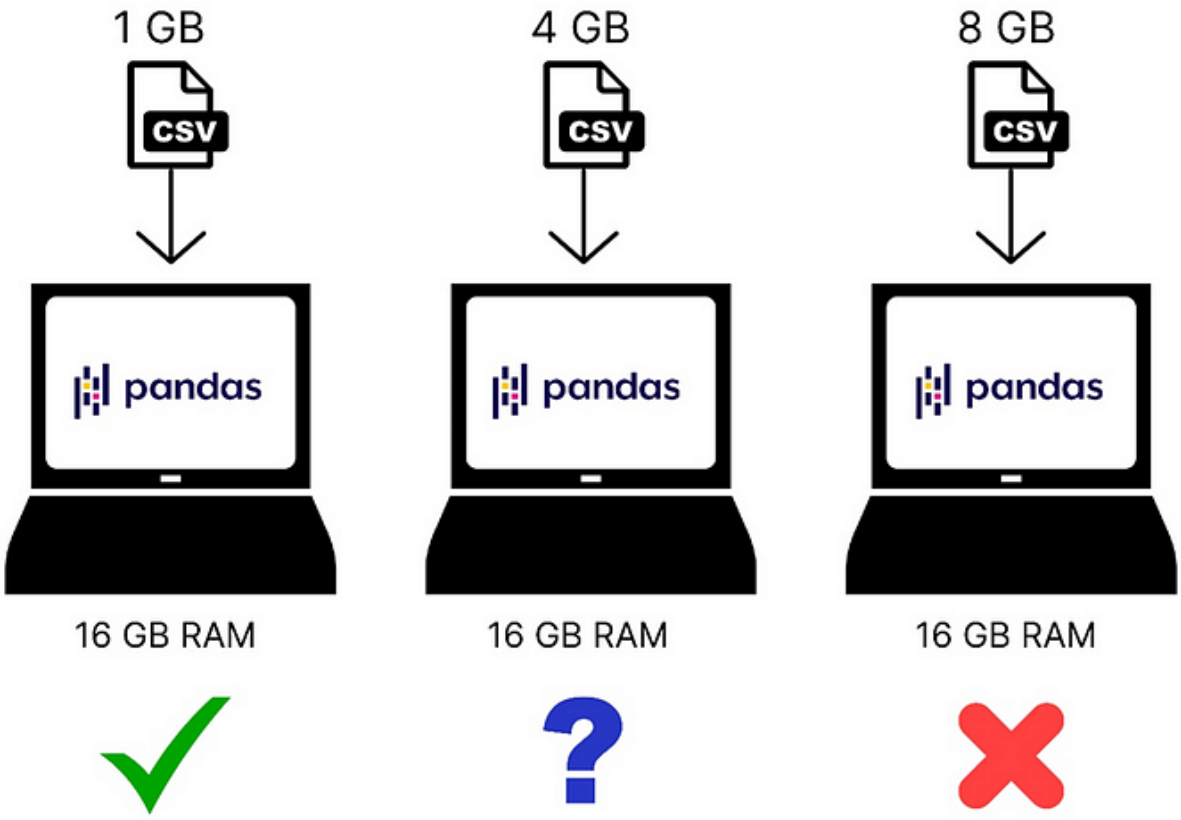
Pandas es una librería de Python diseñada específicamente para el análisis de datos y la manipulación de estructuras de datos tabulares. Funciona como una herramienta poderosa para trabajar con datos estructurados de manera eficiente y flexible. Aquí te explico cómo funciona y qué características principales ofrece:

1. **Estructuras de datos fundamentales:**
    - **DataFrame:** Esta es la estructura de datos principal en pandas. Se trata de una tabla bidimensional con filas y columnas etiquetadas. Cada columna puede contener diferentes tipos de datos (números, texto, fechas, etc.).
    - **Series:** Una estructura de datos unidimensional que se utiliza para almacenar secuencias de datos, similar a una columna en un DataFrame.
2. **Funcionalidades clave:**
    - **Carga y escritura de datos:** Pandas puede cargar datos desde una variedad de fuentes, como archivos CSV, Excel, bases de datos SQL, entre otros. También puede escribir datos en diferentes formatos.
    - **Indexación y selección:** Permite acceder y seleccionar datos utilizando etiquetas de filas y columnas, así como índices numéricos. Esto facilita la extracción de datos específicos de un DataFrame.
    - **Operaciones de limpieza y transformación:** Ofrece funciones para limpiar datos, manejar valores nulos, renombrar columnas, filtrar filas, calcular estadísticas, aplicar funciones a los datos, etc.
    - **Agregación y agrupamiento:** Permite agrupar datos según ciertas columnas y realizar operaciones de agregación, como sumas, promedios, conteos, etc., dentro de cada grupo.
    - **Visualización de datos:** Pandas se integra bien con otras librerías de Python, como Matplotlib y Seaborn, para crear gráficos y visualizaciones a partir de los datos.
3. **Funcionamiento interno:**
    - Pandas está construido sobre las librerías NumPy y matplotlib, lo que le proporciona un rendimiento eficiente y capacidades avanzadas para el cálculo numérico y la visualización de datos.

- Utiliza estructuras de datos optimizadas en términos de memoria y rendimiento, lo que lo hace adecuado para trabajar con grandes conjuntos de datos.

Performance :

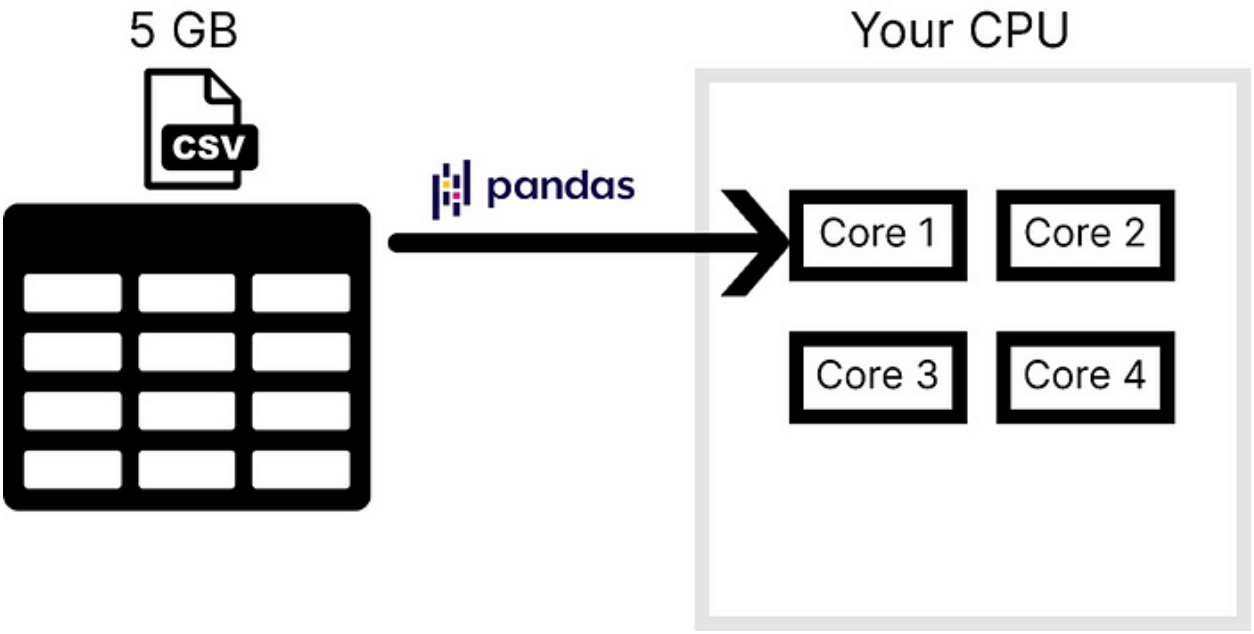
Sabemos que Pandas utiliza la memoria de la computadora (RAM) para cargar conjuntos de datos. Pero si tu computadora tiene una memoria (RAM) de hasta 8 GB, ¿por qué Pandas aún es incapaz o tarda en cargar un conjunto de datos de 2 GB? La razón es que cargar un archivo de 2 GB usando Pandas no solo requiere 2 GB de RAM sino mucho más que eso, porque el requisito total de memoria se basa en el tamaño del conjunto de datos junto con las operaciones que vas a realizar en ese conjunto de datos, y hasta proyectos pequeños contienen muchas operaciones de Pandas para lograr la tarea.



Además, Pandas solo utiliza un núcleo de tu sistema operativo, lo que hace que el proceso sea lento. En otras palabras, podríamos decir que Pandas no admite el paralelismo (dividir un problema en tareas más pequeñas). Incluso si puedes cargar una gran cantidad de datos

usando Pandas, una sola operación puede llevar muchos minutos en mostrar el resultado, debido a que utiliza un solo núcleo a la vez.

Digamos que mi laptop tiene 4 núcleos, así es cuántos núcleos está usando Pandas en mi laptop al cargar un archivo CSV:



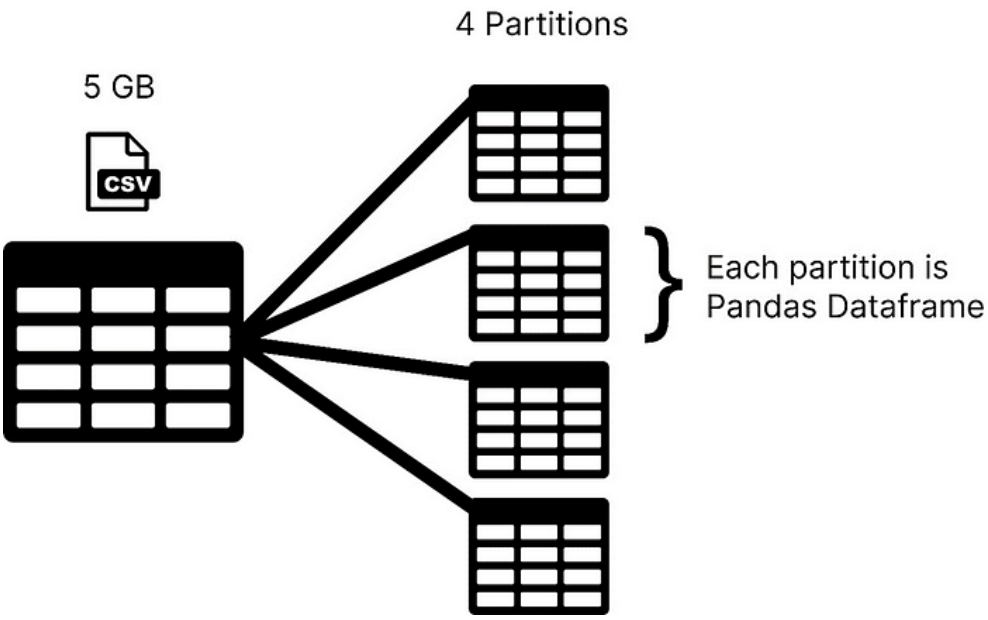
Hemos visto dos razones principales para no utilizar Pandas en conjuntos de datos grandes: uno es el uso de memoria de la computadora y el segundo es la falta de paralelismo. En el caso de NumPy y Scikit-learn, también son incapaces de cargar conjuntos de datos enormes debido a estos mismos problemas.

Para superar estos dos problemas principales, existe una biblioteca de Python llamada Dask, que nos da la capacidad de realizar operaciones de Pandas, NumPy y aprendizaje automático en conjuntos de datos grandes.

DASK

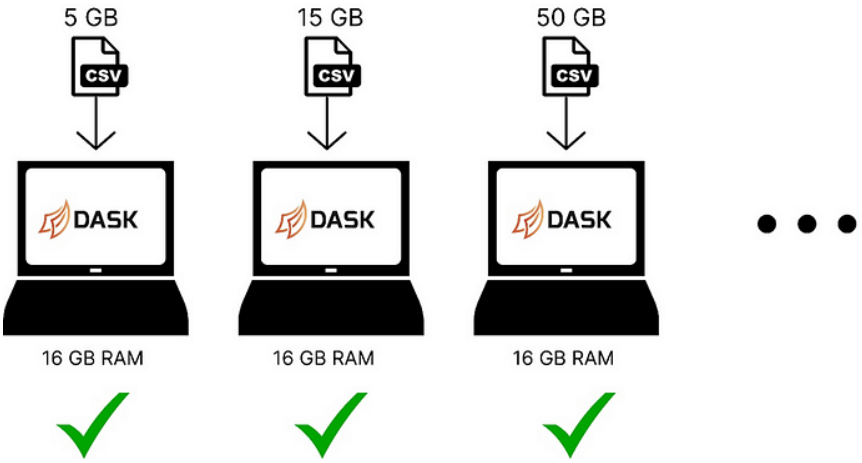
Dask carga tu conjunto de datos en particiones, pero no como un marco de datos combinado, como suele hacer Pandas. En Dask, cada partición de tu conjunto de datos se considera un marco de datos de Pandas.

Así es como se ve:



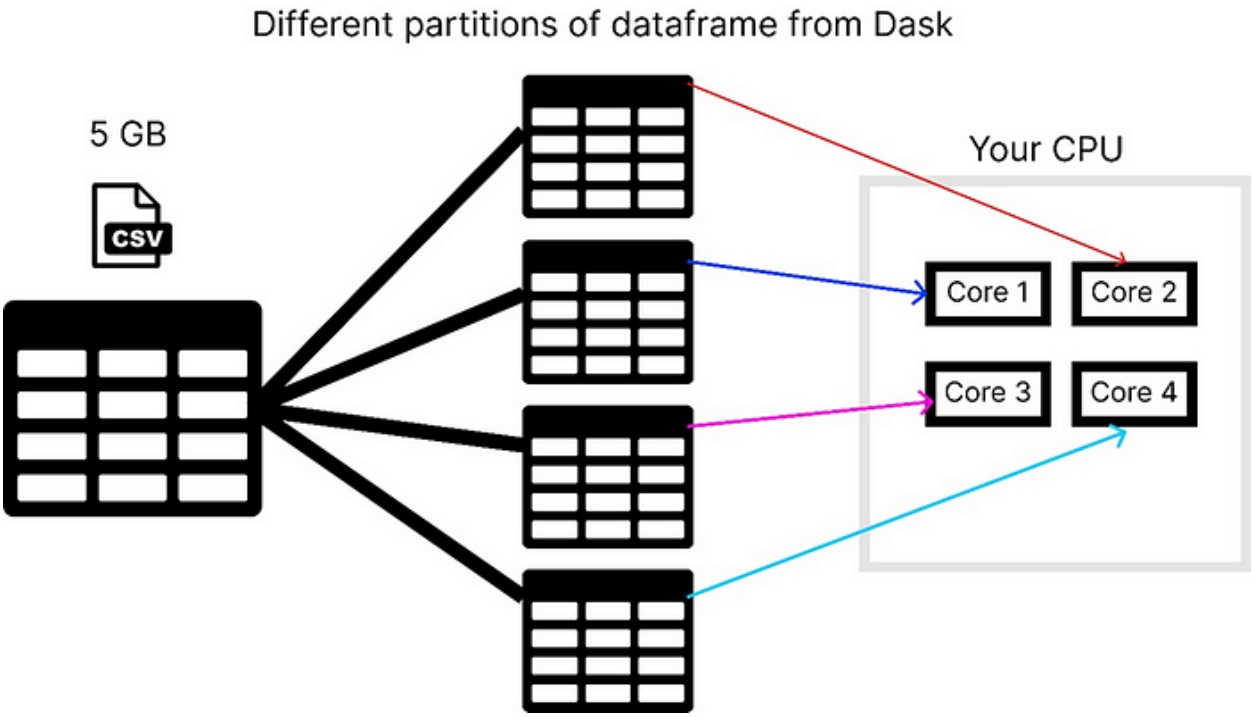
Dask carga las particiones una por una, por lo que no tienes que preocuparte por obtener errores de asignación de memoria.

Aquí tienes una comparación rápida de diferentes tamaños de conjunto de datos cargados en la memoria de la computadora utilizando Dask:



Dask resuelve el problema del paralelismo al dividir los datos en múltiples particiones, cada partición se asigna a un núcleo separado, lo que hace que los cálculos sean más rápidos en tus conjuntos de datos.

Digamos que mi laptop tiene 4 núcleos, así es como Dask los está utilizando al cargar un archivo CSV de 5 GB:



Luego de obtener una visión general de por qué utilizar Dask, ahora nos estamos moviendo hacia el código. Te recomiendo que leas la documentación oficial una vez que termines este blog.