# Pandas:

## Parallel-pandas:

" Parallel-pandas " proporciona una forma sencilla de paralelizar operaciones en Pandas DataFrame utilizando todos los núcleos de la CPU disponibles en tu máquina. Esto puede ser especialmente útil para acelerar el procesamiento de grandes conjuntos de datos y para tareas que implican operaciones intensivas en CPU, como aplicar funciones a columnas, filtrar datos, etc.

La manera en que funciona " Parallel-pandas " es que divide el DataFrame en partes más pequeñas y luego aplica las operaciones en paralelo a cada una de estas partes utilizando múltiples procesos. Esta técnica puede resultar en una mejora significativa del rendimiento en comparación con el procesamiento secuencial, especialmente en máquinas con múltiples núcleos de CPU.

## Dask:

Dask es una librería de Python que se utiliza para realizar computación paralela y distribuida en grandes conjuntos de datos. Algunas de las funcionalidades y usos principales de Dask incluyen:

* **Escalabilidad:** Dask permite manejar conjuntos de datos que no caben en la memoria RAM de una sola máquina, ya que puede escalar a través de múltiples nodos de un clúster.
* **Computación paralela:** Dask permite realizar operaciones de manera paralela, lo que puede mejorar significativamente el rendimiento en comparación con el procesamiento secuencial.
* **Manejo de datos grandes:** Dask proporciona estructuras de datos que son similares a las de NumPy (para arreglos multidimensionales) y Pandas (para dataframes), pero que pueden manejar conjuntos de datos mucho más grandes, distribuyendo la carga de trabajo entre múltiples procesos o nodos.
* **Integración con ecosistemas existentes:** Dask se integra bien con las librerías de análisis de datos populares en Python, como NumPy, Pandas, Scikit-learn y otras, lo que permite a los usuarios aprovechar la escalabilidad y paralelismo de Dask en sus flujos de trabajo existentes.
* **Computación distribuida:** Dask ofrece la posibilidad de ejecutar operaciones en paralelo en clústeres de computadoras, distribuyendo las tareas a través de múltiples nodos para acelerar el procesamiento de grandes volúmenes de datos.

Parece hacer una implementación específica para geopandas “dask-geopandas”.

## Bibliografía:

* <https://pypi.org/project/parallel-pandas/>
* <https://pypi.org/project/dask-geopandas/>
* <https://towardsdatascience.com/easily-parallelize-your-calculations-in-pandas-with-parallel-pandas-dc194b82d82f>
* <https://docs.dask.org/en/latest/>
* <https://github.com/nalepae/pandarallel>

# Numpy (no son específicas de numpy):

## Multiprocessing:

La librería “multiprocessing” en Python se utiliza para la programación concurrente y paralela. Proporciona una interfaz para la creación y administración de procesos, permitiendo que los programas utilicen múltiples procesadores en una máquina.

* **Creación de procesos:** Permite crear nuevos procesos en Python utilizando la clase Process. Estos procesos pueden ejecutar funciones en paralelo con el proceso principal.
* **Comunicación entre procesos:** Proporciona mecanismos como colas (Queue), tuberías (Pipe) y bloqueos (Lock, Semaphore, Event, etc.) para la comunicación y la sincronización entre procesos.
* **Paralelismo:** Permite aprovechar los múltiples núcleos de CPU de una máquina para ejecutar tareas en paralelo, lo que puede mejorar el rendimiento de programas que realizan tareas intensivas en CPU.
* **Multiprocesamiento en sistemas operativos multiplataforma:** multiprocessing es compatible con múltiples sistemas operativos, lo que lo hace útil para desarrollar aplicaciones que deben ejecutarse en diferentes plataformas.
* **Programación concurrente:** Además del paralelismo, multiprocessing también puede usarse para la programación concurrente, permitiendo que varias tareas se ejecuten de manera simultánea y gestionando la concurrencia de forma segura.

## Concurrent.futures:

La librería “concurrent.futures” en Python proporciona una interfaz de alto nivel para la ejecución concurrente de código, tanto de manera asíncrona como paralela.

* **Ejecución asíncrona de tareas:** concurrent.futures proporciona la clase ThreadPoolExecutor y la clase ProcessPoolExecutor, que permiten ejecutar múltiples tareas de manera asíncrona utilizando un grupo de hilos o un grupo de procesos, respectivamente.
* **Interfaz simple y fácil de usar:** La librería concurrent.futures ofrece una interfaz de alto nivel que simplifica la ejecución de tareas concurrentes. Esto permite a los desarrolladores escribir código más legible y mantenible para manejar tareas concurrentes.
* **Programación de subprocesos y procesos**: La librería ofrece tanto un pool de hilos (ThreadPoolExecutor) como un pool de procesos (ProcessPoolExecutor), lo que permite a los desarrolladores elegir entre ejecutar tareas concurrentes utilizando hilos o procesos, dependiendo de los requisitos específicos de la aplicación.
* **Gestión de concurrencia y paralelismo:** concurrent.futures se encarga de la gestión de hilos o procesos, así como de la asignación de tareas a los diferentes hilos o procesos. Esto facilita la escritura de código concurrente y paralelo sin tener que preocuparse por los detalles de implementación.
* **Facilidad de integración:** La librería concurrent.futures se integra bien con otras partes de la biblioteca estándar de Python, como las funciones map y submit, así como con la instrucción with, lo que facilita su uso en diferentes contextos y aplicaciones.

## Bibliografía:

* <https://numpy.org/doc/stable/reference/random/multithreading.html>
* <https://docs.python.org/3/library/concurrent.futures.html>
* <https://docs.python.org/3/library/multiprocessing.html>