Evuación del rendimiento de *Dask* frente a *pandas* en el procesamiento de grandes volúmenes de datos científicos utilizando el dataset CORD-19

Justine López Centeno
Lead University
Costa Rica
justin.lopez@ulead.ac.cr

Oscar Umaña Roque
Lead University
Costa Rica
oscar.umana@ulead.ac.cr

Resumen - El presente trabajo tiene como propósito analizar la eficiencia de la biblioteca Dask en comparación con pandas para el procesamiento de grandes volúmenes de datos científicos. En particular, se utiliza el dataset CORD-19, un conjunto de más de un millón de artículos académicos relacionados con la pandemia de COVID-19, con el fin de evaluar el comportamiento de ambas herramientas en términos de tiempo de ejecución y consumo de memoria. La investigación se desarrolla en un entorno de ejecución en la nube mediante Google Colab, utilizando Python como lenguaje de programación. Este análisis permitirá identificar las ventajas y limitaciones del paralelismo en la manipulación de datos con Dask, así como su aplicabilidad en contextos de análisis científico.

Palabras clave — Dask, Python, procesamiento paralelo, análisis de datos, CORD-19, pandas.

I. INTRODUCCION

En la actualidad, el análisis de grandes volúmenes de datos se ha convertido en un desafío común en la investigación científica y tecnológica. Las bibliotecas tradicionales de Python, como pandas, resultan muy eficientes para el manejo de datasets pequeños o medianos, pero presentan limitaciones significativas al procesar archivos de varios gigabytes o millones de registros. En este contexto surge Dask, una biblioteca de código abierto diseñada para extender las capacidades de pandas mediante un sistema de ejecución paralela que distribuye el trabajo entre múltiples núcleos o incluso clústeres. Gracias a ello, Dask permite manejar conjuntos de datos más extensos y optimizar el uso de recursos computacionales.

El dataset CORD-19 (COVID-19 Open Research Dataset) constituye un excelente caso de estudio, pues contiene más de un millón de publicaciones científicas relacionadas con el COVID-19, SARS-CoV-2 y otras enfermedades respiratorias. Su tamaño y complejidad hacen que el análisis con pandas resulte poco eficiente, por lo que Dask representa una alternativa atractiva para mejorar el rendimiento. El objetivo principal de este trabajo es evaluar la eficacia de Dask frente a pandas en la lectura, transformación y agregación de datos a gran escala, considerando métricas de rendimiento como el tiempo de procesamiento y el uso de memoria.

II. TRABAJO RELACIONADO

El procesamiento eficiente de grandes volúmenes de datos ha motivado el desarrollo de bibliotecas escalables en Python que extienden las capacidades de herramientas tradicionales como pandas. Si bien pandas es ampliamente utilizado para análisis en memoria, diversos trabajos destacan sus limitaciones al enfrentar datasets que superan la capacidad de RAM disponible. En respuesta a este problema, han surgido frameworks capaces de distribuir trabajo computacional, entre los cuales Dask se ha posicionado como una de las soluciones más adoptadas en la comunidad científica.

Dask fue presentado formalmente por Rocklin (2015) como un framework flexible para paralelismo dinámico, construido sobre estructuras familiares como arrays y dataframes, pero diseñadas para ejecutarse en múltiples hilos, procesos o clústeres distribuidos. Desde entonces, múltiples estudios han mostrado su utilidad en dominios como análisis biomédico, aprendizaje automático y simulaciones científicas. Por ejemplo, Thompson et al. demostraron que Dask permite acelerar significativamente flujos de trabajo en genómica mediante particionamiento distribuido. De manera similar, F. Nielsen et al. reportaron mejoras sustanciales en tareas de climatología al emplear Dask para procesamiento de grillas multidimensionales que exceden la memoria del sistema.

En el contexto particular del dataset CORD-19, Wang et al. presentaron la versión inicial del repositorio en 2020 como un esfuerzo para consolidar literatura científica sobre COVID-19 y acelerar la investigación global. Desde entonces, diversos estudios han utilizado CORD-19 para minería de texto, análisis bibliométrico y modelado epidemiológico, frecuentemente recurriendo a herramientas distribuidas como Dask para manejar su tamaño creciente. Investigaciones comparativas han mostrado que, mientras pandas mantiene ventaja en operaciones puramente en memoria, Dask ofrece beneficios significativos en lectura, preprocesamiento y consultas sobre archivos grandes o particionados.

En conjunto, la literatura existente respalda la necesidad de frameworks paralelos para el análisis científico moderno y posiciona a Dask como una alternativa eficaz cuando los datos superan los límites de un solo equipo. Este trabajo continúa esa línea de investigación mediante una evaluación empírica del desempeño de Dask frente a pandas, utilizando el dataset

CORD-19 como caso de estudio realista y de alta exigencia computacional.

III. Marco Teórico

El análisis de datos a gran escala requiere herramientas capaces de manejar eficientemente operaciones que exceden la memoria RAM o que demandan paralelismo. En Python, dos enfoques predominan: bibliotecas orientadas a procesamiento en memoria, como pandas, y frameworks paralelos o distribuidos, como Dask, diseñados para escalabilidad horizontal y vertical. Esta sección presenta los fundamentos técnicos necesarios para comprender el comportamiento de ambas herramientas en experimentos de alto volumen de datos.

1) Pandas: Modelo de Ejecución en Memoria:

Pandas es una biblioteca optimizada para el análisis y manipulación de datos estructurados en un solo equipo. Su arquitectura está basada en:

- DataFrames en memoria: todas las operaciones requieren cargar el dataset completo en RAM, lo que limita su uso a archivos cuyo tamaño no exceda la capacidad del hardware.
- Ejecutación secuencial: las funciones se procesan fila por fila o columna por columna sin paralelizar tareas internas.
- Optimización en C y NumPy: muchas operaciones están escritas en C, lo que mejora la velocidad pero no permite escalado automático a múltiples núcleos.

Pandas es altamente eficiente para análisis exploratorios rápidos, pero su desempeño disminuye drásticamente cuando el dataset supera unos pocos gigabytes.

2) Dask: Modelo de Ejecución Paralelo y Distribuido:

Dask es una biblioteca diseñada para extender la sintaxis familiar de pandas hacia un modelo de procesamiento paralelo. Su funcionamiento interno se basa en tres principios fundamentales:

- DataFrames particionados:

  Un Dask DataFrame divide los datos en múltiples particiones (bloques). Cada partición actúa como un DataFrame independiente que puede procesarse simultáneamente. Esto habilita:

  - Equilibrio de carga entre núcleos,
  - procesamiento de archivos que no caben en memoria y
  - ejecución en clústeres distribuidos si se dispone de ellos.

- Evaluación diferida (Lazy Evaluation):

  A diferencia de pandas, Dask no ejecuta las operaciones inmediatamente. En su lugar, construye un grafo de tareas (task graph) que describe qué operaciones se deben ejecutar, en qué orden, y tecas durante la evaluación.

- Relevancia en el contexto del dataset CORD-19:

  El dataset CORD-19 (más de un millón de registros) excede con facilidad la memoria estándar de un equipo convencional. Su volumen, heterogeneidad y presencia de columnas con tipos inconsistentes lo convierten en un caso ideal para comprobar:

  - El límite práctico de pandas,
  - el impacto del particionamiento de Dask y,
  - la eficacia de su ejecución paralela durante tareas como lectura, filtrado, agregación y conteo.

  Estos fundamentos teóricos justifican el diseño experimental utilizado y explican las diferencias de rendimiento observadas entre las dos bibliotecas durante la evaluación.

IV. Metodología

La metodología se desarrolla en un entorno controlado mediante Google Colab, donde se ejecutan las pruebas con ambas bibliotecas bajo condiciones similares. Las herramientas utilizadas son Python como lenguaje principal, Dask y pandas como bibliotecas de análisis, y el dataset CORD-19 como fuente de datos. El procedimiento incluye la carga del archivo metadata.csv, la inspección de las columnas más relevantes, la ejecución de operaciones de agregación, y la medición del tiempo y uso de memoria tanto en pandas como en Dask. Finalmente, se realiza una comparación cuantitativa para identificar diferencias de rendimiento.

El objetivo de esta sección es describir de manera detallada el diseño experimental utilizado para comparar el desempeño de pandas y Dask en el procesamiento del dataset CORD-19. La metodología se estructuró para garantizar reproducibilidad, control sobre el entorno y mediciones consistentes de rendimiento.

1) Entorno de Ejecución:

Todas las pruebas se realizaron en Google Colab, utilizando un entorno estándar provisto por la plataforma:

- CPU: 2 vCPUs
- RAM: ~12 GB
- Python: versión 3.x
- Librerías:
  - pandas
  - Dask (módulo dask.dataframe)
  - psutil (para medir memoria)

La elección de Colab garantiza un entorno homogéneo y accesible, permitiendo replicar los resultados sin necesidad de hardware especializado.

2) Datos Utilizados:

El dataset empleado fue **CORD-19**, un repositorio público con más de un millón de publicaciones científicas relacionadas con COVID-19. Para este experimento se utilizó el archivo:

- metadata.csv
- Tamaño aproximado: 1.65 Gb
- Columnas relevantes: title, abstract, publish_ time, journal, doi, pubmed_id, arxiv_id

Este archivo se caracteriza por su gran volumen y por presentar inconsistencias comunes en datasets científicos reales, como valores faltantes y variación en los tipos de datos.

3) Configuración de la Ruta y Preparación del Archivo:
El archivo se almacenó en la ruta: /content/metadata.csv. O, alternativamente, en Google Drive cuando fue necesario. Se utilizó codificación "latin1" para evitar errores con caracteres extendidos.

4) Diseño Experimental:
El experimento constó de cuatro etapas principales, aplicadas tanto en pandas como en Dask:

- **Lectura del archivo:**
  - pandas: carga completa en memoria.
  - Dask: lectura paralela en particiones con ejecución diferida.
  - Medidas: tiempo total y memoria adicional utilizada.

  Esta prueba evalúa la capacidad de cada biblioteca para manejar archivos grandes.
- **Creación de la columna publish_year:**

  A partir de la columna publish_time, se generó el año de publicación utilizando conversión a formato fecha.
  - En pandas: ejecución inmediata sobre todo el DataFrame.
  - En Dask: definición del cálculo sin ejecutarlo hasta llamar a .compute().

  Se midió el tiempo requerido por cada biblioteca para realizar esta operación.
- **Agrupación por año:**

  Se realizó un conteo de artículos por año de publicación utilizando:

  groupby("publish_year").size()

  La prueba permitió comparar la eficiencia de operaciones agregadas en memoria vs. distribuidas.
- **Cálculo del Top 10 journals:**

  Se evaluó la frecuencia de publicaciones por revista: value_counts().head(10)

  Esta operación es representativa de análisis bibliométrico básico.
- **Medición del Rendimiento:**

  Se utilizaron dos métricas principales:
  - Tiempo de ejecución: medido con: time.time(). Registrando inicio y fin de cada operación.
  - Uso de memoria: medido con psutil mediante la función: psutil.Process().memory_info().rss. Lo que permitió obtener la diferencia de memoria antes y después de cada tarea.
- **Validación de Errores y Preprocesamiento:**

  Durante la lectura del dataset se realizaron ajustes técnicos para evitar fallas:
  - En pandas:
    - engine="python"
    - on_bad_lines="warn"
  - En Dask:
    - assume_missing=True
    - Forzado de tipos para columnas problemáticas (dtype=object)

  Estas correcciones garantizaron la integridad del experimento y evitaron interrupciones.
- **Reproducibilidad:**

  El código completo fue estructurado en un solo script (analisis_pandas_dask.py) que incluye:
  - Instalación de librerías
  - Medición de memoria
  - Pruebas paralelas en pandas y Dask
  - Impresión de métricas y resultados

  Además, se creó un archivo README con instrucciones de ejecución para garantizar que el experimento pueda replicarse sin dificultades.

V. Objetivos

1) Objetivo general:
Evaluar la eficiencia y escalabilidad de la biblioteca Dask en comparación con pandas para el procesamiento de grandes volúmenes de datos científicos, utilizando el dataset CORD-19 como caso de estudio.

2) Objetivos específicos:
- Implementar flujos de análisis de datos con Dask y pandas utilizando el mismo conjunto de datos CORD-19.
- Medir y comparar el tiempo de ejecución y el consumo de memoria de ambas bibliotecas bajo condiciones equivalentes.
- Analizar la capacidad de Dask para aprovechar el paralelismo en la lectura y transformación de datos masivos.
- Identificar las ventajas y limitaciones de Dask frente a pandas en tareas de análisis científico.
- Documentar los resultados, destacando la aplicabilidad de Dask en entornos de investigación basados en Python y computación en la nube.

VI. JUSTIFICACIÓN

El procesamiento eficiente de grandes volúmenes de información es un desafío central en la investigación científica moderna, especialmente ante el crecimiento exponencial de los datos generados por publicaciones, experimentos y sensores. Las herramientas tradicionales como pandas presentan limitaciones de rendimiento y memoria cuando el tamaño de los datos supera la capacidad de un solo equipo, lo que restringe su uso en contextos de Big Data.

En este sentido, la biblioteca Dask surge como una alternativa de código abierto que extiende la funcionalidad de pandas mediante un modelo de ejecución paralela y distribuida, capaz de operar tanto en entornos locales como en la nube. Dask permite procesar datasets de varios gigabytes de forma eficiente al dividir las tareas en bloques manejables que se ejecutan simultáneamente.

La elección del dataset CORD-19, compuesto por más de un millón de artículos científicos sobre COVID-19, ofrece un escenario realista y exigente para probar el desempeño de ambas bibliotecas. Evaluar la eficiencia de Dask frente a pandas permitirá determinar su viabilidad en análisis científicos a gran escala, aportando evidencia sobre su utilidad en entornos donde la optimización de recursos computacionales es esencial.

Además, el uso de Google Colab como entorno de ejecución proporciona un marco controlado y reproducible para las pruebas, garantizando condiciones uniformes y la posibilidad de escalar los experimentos sin requerir infraestructura física propia.

VII. PLAN DE TRABAJO

El desarrollo del proyecto se estructura en una serie de etapas diseñadas para garantizar un análisis sistemático y reproducible del desempeño de las bibliotecas Dask y pandas en el procesamiento de grandes volúmenes de datos. Cada fase contempla actividades específicas orientadas a la preparación del entorno, la implementación de los experimentos, la recolección de métricas de rendimiento y la elaboración del informe técnico final.

El trabajo se llevará a cabo de manera progresiva, permitiendo una transición ordenada desde la comprensión teórica del problema hasta la validación empírica de los resultados. A continuación, se detalla el plan de trabajo propuesto, indicando las tareas principales, los resultados esperados y la duración estimada de cada etapa.

Etapa	Actividad	Duración
Definición del problema y revisión teórica	Investigación preliminar sobre Dask y pandas, revisión de documentación oficial y artículos académicos.	1 semana
Preparación del entorno y selección del dataset	Configuración de entorno en Google Colab, instalación de dependencias (dask, pandas, psutil) y descarga del dataset CORD-19.	1 semana
Implementación experimental	Desarrollo de notebooks con operaciones comparables (lectura, filtrado, agregación) en pandas y Dask.	2 semanas
Evaluación de rendimiento	Ejecución de pruebas controladas y registro de métricas (tiempo de ejecución y consumo de memoria).	1 semana
Análisis y documentación	Interpretación de resultados, redacción del informe técnico en formato IEEE y publicación en GitLab.	1 semana

VIII. ANÁLISIS DEL DATASET CORD-19

Esta sección presenta el análisis del dataset CORD-19, con el propósito de evaluar el desempeño de pandas y Dask en las primeras etapas del procesamiento de datos: lectura, validación y transformaciones básicas. Dado que el repositorio contiene información masiva y heterogénea, constituye un escenario adecuado para examinar cómo se comportan ambas bibliotecas frente a operaciones fundamentales dentro de un flujo de análisis científico.

1) Descripción del Dataset:

El dataset CORD-19 (COVID-19 Open Research Dataset) es uno de los repositorios más extensos utilizados para investigación sobre COVID-19, SARS-CoV-2 y enfermedades respiratorias relacionadas. Está compuesto por:

a) Más de 1,000,000 de artículos académicos.

b) Más de 400,000 documentos con texto completo.

c) Diversas fuentes como PubMed, WHO, bioRxiv, medRxiv y revistas indexadas.

d) Metadatos clave: title, abstract, publish_time, authors, journal, doi, pubmed_id,arxiv_id, etc.

La magnitud y variedad de los datos justifican el empleo de herramientas capaces de manejar cargas de trabajo superiores a la memoria RAM disponible, lo que convierte a CORD-19 en un caso de estudio ideal para comparar el comportamiento de pandas y Dask.

2) Validación Inicial del Dataset:

Durante la carga del archivo *metadata.csv***,** se identificaron distintos problemas estructurales ampliamente comunes en repositorios de gran tamaño:

a) Validación con pandas

- Se detectaron líneas corruptas y registros incompletos, generando errores del tipo “unexpected end of data”.
- El inconveniente se manejó mediante el parámetro: on_bad_lines="warn". Esto permitió continuar el análisis sin interrumpir la ejecución, aunque conservando la advertencia sobre la integridad variable del archivo.

b) Validación con Dask

- Dask mostró advertencias sobre inconsistencias en los tipos de datos, especialmente en columnas como pubmed_id y arxiv_id.
- Dado que estas columnas pueden contener números, cadenas vacías o valores nulos, se forzó un tipo uniforme mediante:

  dtype={"pubmed_id": "object", "arxiv_id": "object"}
- Esta corrección fue fundamental para evitar fallos posteriores durante las operaciones de particionamiento y computación diferida.

En conjunto, esta etapa evidenció que ambos entornos requieren ajustes específicos para manejar correctamente datasets masivos y heterogéneos.

3) Descripción del Código Implementado:

El script desarrollado en Google Colab ejecuta las siguientes etapas tanto con pandas como con Dask:

a) Lectura del archivo metadata.csv
   - pandas: carga completa en memoria
   - Dask: carga en bloques con ejecución diferida

b) Medición del tiempo y uso de memoria
   - Se utilizaron time.time() y herramientas nativas de Colab para evaluar recursos.

c) Creación de la columna publish_year
   - Conversión de la fecha a un valor numérico basado en el año.

d) Agrupación por año de publicación
   - Conteo de artículos por año para observar tendencias temporales.

e) Cálculo del Top 10 journals
   - Identificación de las revistas con mayor número de publicaciones en el dataset.

f) Comparación de rendimiento
   - Se midieron los tiempos de ejecución en ambas bibliotecas bajo condiciones equivalentes.

Este flujo permitió obtener una evaluación preliminar clara sobre las fortalezas y debilidades de cada herramienta implementada.

4) Resultados:

Los experimentos realizados permitieron comparar el desempeño de pandas y Dask en diferentes etapas del procesamiento del dataset CORD-19. Se midieron tiempos de ejecución y uso de memoria durante la lectura del archivo, la creación de variables derivadas, y la ejecución de operaciones agregadas.

Los resultados obtenidos se presentan a continuación.

a) Lectura del Archivo:

La lectura del archivo metadata.csv mostró diferencias significativas entre ambos enfoques. Pandas requirió cargar todo el dataset en memoria, mientras que Dask solo construyó las particiones y el grafo de tareas de forma diferida.

Operación	Pandas (s)	Pandas (MB)	Dask (s)	Dask (MB)
Lectura del archivo	59.38	618.78	0.083 s	0 (lectura diferida)

Dask fue notablemente más rápido en la carga inicial gracias a su modelo de lectura en bloques y ejecución diferida. Pandas, en contraste, presentó un uso elevado de memoria debido a su arquitectura basada en un DataFrame único cargado por completo.

b) Creación de la columna publish_year:

Ambas bibliotecas permitieron derivar correctamente el año a partir de publish_time. Los tiempos obtenidos fueron:

Operación	Pandas (s)	Dask (s)
Creación de publish_year	1.10	0.013

La ventaja de Dask se explica por su evaluación diferida, la cual registra la operación sin ejecutarla inmediatamente, reduciendo el tiempo medido.

c) Agrupación por Año:

La agrupación por año evidenció comportamientos opuestos a los de la lectura del archivo. Debido al overhead del scheduler y a la necesidad de combinar resultados entre múltiples particiones, Dask requirió más tiempo de ejecución que pandas.

Operación	Pandas (s)	Dask (s)
Agrupación por año	0.022	54.13

En este caso, pandas resultó significativamente más eficiente, dado que las operaciones se ejecutan de manera monolítica en memoria, sin coordinación entre particiones.

d) Cálculo del Top 10 de Revistas:

Se evaluó la frecuencia de publicaciones por revista (journal). Los resultados mostraron un comportamiento similar al de la agrupación por año:

Operación	Pandas (s)	Dask (s)
Top 10 journals	0.248	63.73

Las operaciones de tipo value_counts() son sensibles al particionamiento, y en Dask requieren consolidar conteos parciales antes de producir el resultado final, lo que explica tiempos significativamente mayores.

Estos resultados reflejan el comportamiento natural de ambas bibliotecas según su diseño interno.

IX. DISCUSIÓN DE RESULTADOS

Los resultados obtenidos en la comparación entre pandas y Dask permiten analizar de manera detallada el comportamiento de cada biblioteca bajo condiciones equivalentes de procesamiento. Esta discusión profundiza en las razones técnicas que explican las diferencias observadas en rendimiento, así como en las implicaciones para el análisis de datos científicos a gran escala.

1) Diferencias estructurales entre pandas y Dask:

Los resultados reflejan las arquitecturas internas de ambas herramientas. Pandas opera sobre un único DataFrame residente en memoria, lo que favorece velocidades elevadas en operaciones que no requieren distribución de tareas. No obstante, esto impone un límite estricto: el dataset debe caber completamente en RAM. En contraste, Dask divide el dataset en particiones y ejecuta operaciones mediante un scheduler que coordina tareas paralelas. Este diseño permite el manejo de datos mayores a la capacidad del sistema, pero introduce un costo estructural: el overhead del scheduler y la necesidad de combinar resultados distribuidos.

2) Lectura de datos: ventajas claras de Dask:

La lectura del archivo metadata.csv evidenció que Dask supera ampliamente a pandas en esta etapa. Su modelo de evaluación diferida permite registrar las particiones y construir el grafo de tareas en tiempos mínimos, sin cargar el dataset completo en memoria. Esta diferencia es consistente con estudios previos que destacan la eficiencia de Dask en el preprocesamiento inicial de grandes volúmenes de datos.

En pandas, la lectura es inmediata y monolítica, lo que genera un uso de RAM significativamente mayor. Este comportamiento limita su aplicabilidad en entornos con memoria reducida o con datasets de varios gigabytes.

3) Operaciones agregadas: la fortaleza de pandas:

Los experimentos de agregación (groupby) y conteo (value_counts) mostraron que pandas mantiene una ventaja considerable cuando los datos ya están en memoria. En estas operaciones, pandas logra tiempos de ejecución de milisegundos, mientras que Dask requiere varios segundos o minutos debido a:

- La coordinación de tareas entre particiones.
- La necesidad de combinar resultados parciales.
- El costo adicional del scheduler distribuido.

Este comportamiento es característico de las operaciones CPU-bound en datasets medianos, donde el paralelismo no compensa la sobrecarga administrativa.

4) Equilibrio entre escalabilidad y velocidad:

Los resultados muestran un contraste importante:

- Dask es insuperable en lectura y manejo de datasets masivos, incluso cuando superan la capacidad de memoria.
- Pandas es más rápido en operaciones internas, siempre que el dataset pueda cargarse en RAM.

En otras palabras, el beneficio de Dask emerge principalmente en cargas de trabajo cuyo tamaño o complejidad no pueden ser procesadas por pandas sin fallos de memoria. En análisis medianos, la sobrecarga del paralelismo puede revertir la ventaja de Dask.

5) Implicaciones para el análisis científico

Estos hallazgos tienen implicaciones directas para proyectos de investigación:

- Para análisis exploratorios o datasets moderados, pandas sigue siendo la herramienta preferida debido a su rapidez y simplicidad.
- Para pipelines complejos, datos ruidosos o archivos muy grandes, Dask ofrece un enfoque escalable capaz de distribuir la carga entre múltiples núcleos o equipos.
- En entornos de computación en la nube, Dask facilita extender el procesamiento a clusters completos sin modificar significativamente el código base.

El dataset CORD-19 confirma este comportamiento, ya que su tamaño y heterogeneidad requieren herramientas capaces de manejar eficientemente estructuras distribuidas.

6) Limitaciones del experimento

Algunas limitaciones del presente estudio incluyen:

- El entorno de Colab no permite evaluar completamente el rendimiento de Dask en clusters reales.
- No se midieron operaciones más complejas como joins, merges o cálculos iterativos, donde Dask podría tener ventajas adicionales.
- El tamaño real del archivo procesado puede haber influido en la magnitud de la diferencia observada entre pandas y Dask.

Aun así, los resultados son representativos de escenarios reales de análisis científico.

X. CONCLUSIONES

El presente estudio evaluó el desempeño de las bibliotecas pandas y Dask en el procesamiento del dataset CORD-19, un repositorio de gran magnitud utilizado ampliamente en investigación científica. Los resultados obtenidos permiten extraer conclusiones claras sobre las capacidades y limitaciones de ambas herramientas bajo condiciones de trabajo equivalentes.

En primer lugar, los experimentos demostraron que **Dask ofrece ventajas sustanciales en la lectura y manejo inicial de archivos grandes, gracias a su arquitectura basada en particiones y evaluación diferida**. La lectura del archivo metadata.csv fue especialmente eficiente, utilizando tiempo y memoria significativamente menores que pandas. Esto posiciona a Dask como una alternativa adecuada para entornos donde los datos superan la capacidad de la memoria disponible o cuando se requiere escalabilidad hacia múltiples núcleos o máquinas.

En contraste, **pandas mostró un desempeño superior en operaciones internas de análisis, tales como agrupaciones, conteos y cálculos sobre columnas derivadas**. Estas tareas, al ejecutarse completamente en memoria y sin coordinación entre particiones, se benefician de la simplicidad y optimización del modelo monolítico de pandas. En datasets que caben en RAM, pandas continúa siendo la herramienta más rápida y directa para el análisis exploratorio.

Los resultados confirman que no existe una herramienta universalmente superior, sino que la elección entre pandas y Dask depende del volumen de los datos, de la naturaleza de las operaciones y de las restricciones del entorno computacional. Para cargas ligeras y medianas, pandas sigue siendo la opción más eficiente. Para cargas pesadas, flujos distribuidos o tareas de preprocesamiento a gran escala, Dask se presenta como una solución más robusta y escalable.

Finalmente, el estudio resalta la importancia de seleccionar herramientas alineadas con las demandas reales del análisis científico contemporáneo. El dataset CORD-19, por su tamaño y complejidad, constituye un caso representativo en el que las bibliotecas paralelas y distribuidas adquieren relevancia crítica. Futuras investigaciones podrían extender este análisis hacia operaciones más complejas, entornos de clúster y evaluaciones comparativas con otras herramientas como Apache Spark o Ray.

XI. Instrucciones para Ejecución del Código

El código se ejecutó en Google Colab siguiendo los pasos estándar para garantizar reproducibilidad. Primero, se instalaron las librerías necesarias (dask[dataframe] y psutil). Luego, se cargó el archivo metadata.csv desde la ruta local o desde Google Drive según correspondiera. Una vez configurada la ruta, se ejecutó el script completo sin modificaciones adicionales.

El código realiza automáticamente la lectura del dataset con pandas y Dask, la medición del tiempo y uso de memoria, la creación de la columna publish_year, la agregación por año y el cálculo del Top 10 journals. Todos los resultados se imprimen en consola para permitir la comparación directa del desempeño entre ambas herramientas.

Este procedimiento asegura que cualquier usuario pueda reproducir los experimentos en un entorno controlado y con recursos equivalentes.

Referencias

[1] Wes McKinney. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference.

[2] McKinney, W. (2017). Python for Data Analysis: Data Wrangling with pandas, NumPy, and IPython. O'Reilly Media. Disponible en: https://wesmckinney.com/pages/book.html

[3] Rocklin, M. (2015). Dask: Parallel Computation with Blocked Algorithms and Task Scheduling. Proceedings of the 14th Python in Science Conference. Disponible en: https://doi.org/10.25080/Majora-7b98e3ed-013

[4] Dask Documentation. (2025). Parallel Computing with Dask. Disponible en: https://docs.dask.org/

[5] Kaggle. (2020). COVID-19 Open Research Dataset (CORD-19). Disponible en: https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

[6] Wang, L., Lo, K., Chandrasekhar, Y., et al. (2020). CORD-19: The COVID-19 Open Research Dataset. arXiv. Disponible en: https://arxiv.org/abs/2004.10706

[7] Thompson, N., Pedersen, J., & Lee, S. K. (2020). Scalable Genomic Processing Using Dask. Bioinformatics. Disponible en: https://academic.oup.com/bioinformatics

[8] Nielsen, F., Gonçalves, L., & Paul, K. (2020). Accelerating Climate Data Processing with Dask. Journal of Open Source Software. Disponible en: https://joss.theoj.org/papers/10.21105/joss.02501

[9] Python Software Foundation. (2024). Python 3 Documentation. Disponible en: https://docs.python.org/3/

[10] Zaitsev, A., & Koskinen, P. (2021). Performance Comparison of Distributed Data Analysis Tools: Dask, Spark, and Ray. IEEE Big Data Conference. Disponible en: https://ieeexplore.ieee.org/document/9671501

[11] Mehta, J., & Kumar, R. (2022). Evaluating Python Libraries for Large-Scale Scientific Data Processing. Concurrency and Computation: Practice and Experience. Disponible en: https://onlinelibrary.wiley.com/journal/15320634