Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Autor: Cliver Wimar Vilca Tinta

Dask Aplicaciones

Análisis de la Aplicación de Dask en Procesamiento de Datos

1. Introducción

Dask es una biblioteca en Python que permite el procesamiento paralelo de grandes datasets, proporcionando una interfaz similar a la de pandas, pero capaz de manejar datos que no caben en memoria. Dask se destaca por su capacidad de paralelizar operaciones en clústeres de múltiples nodos, lo que permite una escalabilidad horizontal y una gestión eficiente de los recursos de hardware.

En este trabajo, se realizará un análisis comparativo del rendimiento de Dask frente a pandas en la tarea de regresión lineal utilizando un dataset grande que contiene más de seis millones de registros. Los datos utilizados en este estudio son abiertos y fueron tomados de la plataforma web Kaggle. Específicamente, se trata del dataset *Fraud Detection Dynamics: Financial Transaction*, el cual contiene información detallada patrones y anomalías dentro de los datos de transacciones.

El dataset incluye varias características relevantes como Describe el tipo de transacción, el valor monetario de la transacción, identificador de la cuenta o entidad de origen que inicia la transacción, el saldo en la cuenta de origen antes de que ocurriera la transacción, saldo en la cuenta de origen después de que se haya procesado la transacción, identificador de la cuenta o entidad de destino que recibe los fondos en cada transacción, etc. Estas características permiten realizar un análisis exhaustivo de patrones y anomalías dentro de los datos de las transacciones, los investigadores y analistas pueden identificar características asociadas con actividades fraudulentas y construir modelos predictivos para detectar automáticamente dichas transacciones en tiempo real.

Para llevar a cabo este análisis, se utilizará una computadora con las siguientes especificaciones de hardware:

- **Procesador:** Rysen 7-5700U @ 1.80GHz
- **Memoria RAM:** 16 GB DDR4
- **Almacenamiento:** SSD de 1 TB
- **Sistema Operativo:** Windows 11 Pro

El objetivo de este análisis es comparar el rendimiento de Dask con pandas en términos de tiempo de ejecución, uso de memoria y precisión del modelo de regresión lineal. Se espera que Dask ofrezca mejoras significativas en el manejo de grandes volúmenes de datos, aprovechando el procesamiento paralelo y optimizando el uso de recursos de hardware. Este informe proporcionará una visión detallada de los beneficios y limitaciones de utilizar Dask en comparación con un enfoque tradicional basado en pandas.

2. Metodología

3. Fórmulas de Regresión Lineal Múltiple

Ecuación de la Regresión Lineal Múltiple

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

Mínimos Cuadrados Ordinarios (OLS)

Función de pérdida (suma de los cuadrados de los residuos):

$$S(\beta_0, \beta_1, \beta_2) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}))^2$$

Estimaciones de los coeficientes (fórmula matricial):

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

Coeficiente de Determinación (R-cuadrado)

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Error Estándar de los Coeficientes

Matriz de covarianza de los coeficientes:

$$\text{Cov}(\hat{\beta}) = \sigma^2 (X^T X)^{-1}$$

Estimación de la varianza del error:

$$\hat{\sigma}^2 = \frac{1}{n - k - 1} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Intervalos de Confianza para los Coeficientes

$$\hat{\beta}_j \pm t_{\alpha/2,n-k-1} \cdot SE(\hat{\beta}_j)$$

donde $j = 0, 1, 2$ y $t_{\alpha/2,n-k-1}$ es el valor crítico de la distribución t de Student con $n - k - 1$ grados de libertad.

Predicción de Nuevos Valores

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2$$

3.1. Proceso con Dask

El proceso con Dask implica la carga y el procesamiento de datos de manera paralela, aprovechando las capacidades de la biblioteca para gestionar grandes volúmenes de información de forma eficiente. Para familiarizarse con Dask y sacar el máximo provecho de sus funcionalidades, es fundamental seguir varios pasos preliminares:

- **Revisión de la documentación oficial:** La documentación de Dask proporciona una guía completa sobre cómo instalar, configurar y utilizar la biblioteca. Es esencial para comprender las características, las API disponibles y las mejores prácticas recomendadas.
- **Visualización de tutoriales en video:** Existen numerosos tutoriales en video disponibles en plataformas como YouTube y cursos en línea que explican cómo usar Dask para diferentes tipos de análisis y procesamiento de datos. Estos recursos visuales pueden ser muy útiles para entender conceptos complejos de manera más intuitiva.
- **Consulta de guías y artículos:** Diversas guías y artículos en línea ofrecen ejemplos prácticos y casos de uso que demuestran cómo aplicar Dask en situaciones del mundo real. Estos recursos suelen incluir fragmentos de código y explicaciones detalladas que pueden ser muy beneficiosas para el aprendizaje.
- **Exploración de ejemplos y casos de uso:** Analizar ejemplos prácticos y casos de estudio donde Dask ha sido utilizado para resolver problemas específicos puede proporcionar una comprensión más profunda y aplicable de la biblioteca.
- **Participación en comunidades y foros:** Unirse a comunidades en línea, como foros y grupos de discusión, permite intercambiar experiencias y obtener consejos de otros usuarios y desarrolladores que han trabajado con Dask.
- **Práctica con proyectos propios:** Implementar pequeños proyectos personales utilizando Dask es una excelente manera de afianzar los conocimientos adquiridos y enfrentar desafíos prácticos que ayudan a consolidar el aprendizaje.

Siguiendo estos pasos, se puede desarrollar una comprensión integral de cómo utilizar Dask para cargar, procesar y analizar grandes volúmenes de datos de manera eficiente, beneficiándose de su capacidad para ejecutar operaciones en paralelo y optimizar el uso de recursos de hardware.

3.2. Proceso sin Dask / con Pandas

El proceso sin Dask se lleva a cabo utilizando pandas y scikit-learn, cargando y procesando los datos de manera secuencial. Para manejar grandes conjuntos de datos y realizar análisis eficientemente con pandas, es fundamental seguir varios pasos preliminares:

- **Revisión de la documentación oficial:** La documentación de pandas y scikit-learn proporciona una guía detallada sobre la instalación, configuración y uso de estas bibliotecas. Comprender las funcionalidades y las API disponibles es esencial para su correcto empleo.
- **Visualización de tutoriales en video:** Existen numerosos tutoriales en plataformas como YouTube y cursos en línea que enseñan cómo usar pandas y scikit-learn. Estos recursos visuales son útiles para entender conceptos complejos de forma intuitiva.
- **Consulta de guías y artículos:** Diversas guías y artículos en línea ofrecen ejemplos prácticos y casos de uso, demostrando cómo aplicar pandas y scikit-learn en situaciones reales. Estos recursos incluyen fragmentos de código y explicaciones detalladas.
- **Exploración de ejemplos prácticos:** Analizar casos de estudio donde pandas y scikit-learn se han utilizado para resolver problemas específicos proporciona una comprensión más profunda y aplicable de estas bibliotecas.
- **Participación en comunidades y foros:** Unirse a comunidades en línea permite intercambiar experiencias y obtener consejos de otros usuarios que han trabajado con pandas y scikit-learn.
- **Práctica con proyectos propios:** Implementar proyectos personales utilizando pandas y scikit-learn es una excelente manera de afianzar los conocimientos adquiridos y enfrentar desafíos prácticos.

Siguiendo estos pasos, se puede desarrollar una comprensión integral de cómo utilizar pandas y scikit-learn para cargar, procesar y analizar grandes volúmenes de datos de manera eficiente, aprovechando sus capacidades para la manipulación de datos y el aprendizaje automático.

3.3. Variables y Métricas de Comparación

Se medirán y compararán las siguientes métricas:

- Tiempo de ejecución
- Uso de memoria
- Precisión del modelo (MSE y R2 Score)

4. Implementación

4.1. Código con Dask

```
from dask.distributed import Client
import dask.dataframe as dd
from dask_ml.model_selection import train_test_split
from dask_ml.linear_model import LinearRegression
from dask_ml.metrics import mean_squared_error, r2_score
import time
import matplotlib.pyplot as plt

def perform_regression_dask(csv_file, target_column, feature_columns=None, test_size=0.2
    # Iniciar el cliente Dask
    client = Client()
    print(client)  # Esto imprimirá la URL del dashboard

    # Medir el tiempo inicial
    start_time = time.time()

    # Cargar la data desde el archivo CSV usando Dask
    data = dd.read_csv(csv_file)

    # Seleccionar las columnas de características e independientes
    if feature_columns is None:
        # Si no se especifican las columnas de características, se usan todas excepto la
        X = data.drop(columns=[target_column])
    else:
        X = data[feature_columns]
    y = data[target_column]

    # Dividir los datos en conjuntos de entrenamiento y prueba usando Dask
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, rando

    # Convertir dask.dataframe a dask.array
    X_train_array = X_train.to_dask_array(lengths=True)
    X_test_array = X_test.to_dask_array(lengths=True)
    y_train_array = y_train.to_dask_array(lengths=True)
    y_test_array = y_test.to_dask_array(lengths=True)

    # Crear el modelo de regresión lineal usando Dask
    model = LinearRegression()

    # Entrenar el modelo
    model.fit(X_train_array, y_train_array)
```

```
    # Predecir los valores
    y_pred = model.predict(X_test_array)

    # Calcular métricas usando Dask
    mse = mean_squared_error(y_test_array, y_pred)
    r2 = r2_score(y_test_array, y_pred)

    # Medir el tiempo final
    end_time = time.time()

    # Imprimir los resultados
    print("Valores de predicción: ", y_pred.compute())
    print("Mean Squared Error: ", mse.compute())
    print("R2 Score: ", r2.compute())
    print("Tiempo de ejecución: {:.2f} segundos".format(end_time - start_time))

    # Visualización de los resultados
    plt.figure(figsize=(12, 5))

    # Histograma de los valores predichos y reales
    plt.subplot(1, 2, 1)
    plt.hist(y_pred.compute(), bins=20, alpha=0.7, color='blue', label='Predicciones')
    plt.hist(y_test_array.compute(), bins=20, alpha=0.7, color='green', label='Reales')
    plt.xlabel('Valor')
    plt.ylabel('Frecuencia')
    plt.title('Histograma de valores predichos y reales')
    plt.legend()

    # Gráfico de dispersión de las predicciones vs valores reales
    plt.subplot(1, 2, 2)
    plt.scatter(y_test_array.compute(), y_pred.compute(), alpha=0.5)
    plt.xlabel('Valores Reales')
    plt.ylabel('Valores Predichos')
    plt.title('Dispersión de valores reales vs predichos')

    plt.tight_layout()
    plt.show()

    # Cerrar el cliente Dask
    client.close()

if __name__ == '__main__':
    # Ejemplo de uso con el dataset proporcionado
    csv_file = 'Transactions Data.csv'
```

```
    target_column = 'amount'  # Variable numérica a predecir
    feature_columns = ['oldbalanceOrg', 'newbalanceOrig']  # Variables numéricas como ca
    perform_regression_dask(csv_file, target_column, feature_columns)
```

4.2. Código sin Dask

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import time
from memory_profiler import memory_usage
import matplotlib.pyplot as plt

# Función para realizar la regresión y medir tiempo y memoria con pandas
def perform_regression(csv_file, target_column, feature_columns=None, test_size=0.2, ran
    # Listas para almacenar tiempos y consumos de memoria
    tiempos = []
    consumos_memoria = []
    y_reales = []
    y_predichos = []

    # Realizar múltiples ejecuciones para obtener estadísticas
    num_ejecuciones = 5  # Puedes ajustar este número según sea necesario
    for _ in range(num_ejecuciones):
        # Medir el tiempo inicial
        start_time = time.time()

        try:
            # Cargar la data desde el archivo CSV
            data = pd.read_csv(csv_file)

            # Imprimir la cantidad total de datos
            total_data = len(data)
            print(f"Cantidad total de datos: {total_data}")

            # Validar si las columnas numéricas están presentes
            if not all(col in data.columns for col in feature_columns + [target_column])
                raise ValueError(f"Alguna de las columnas {feature_columns + [target_col

            # Seleccionar las columnas de características e independientes
            X = data[feature_columns]
            y = data[target_column]
```

```
            # Dividir los datos en conjuntos de entrenamiento y prueba
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_siz

            # Crear el modelo de regresión lineal
            model = LinearRegression()

            # Medir el uso de memoria antes de entrenar el modelo
            mem_before = memory_usage()[0]

            # Entrenar el modelo
            model.fit(X_train, y_train)

            # Medir el uso de memoria después de entrenar el modelo
            mem_after = memory_usage()[0]

            # Predecir los valores
            y_pred = model.predict(X_test)

            # Almacenar valores reales y predichos para scatter plot
            y_reales.append(y_test.values)
            y_predichos.append(y_pred)

            # Medir el tiempo final
            end_time = time.time()

            # Calcular métricas
            mse = mean_squared_error(y_test, y_pred)
            r2 = r2_score(y_test, y_pred)

            # Almacenar tiempo y consumo de memoria
            tiempo_ejecucion = end_time - start_time
            tiempos.append(tiempo_ejecucion)
            consumos_memoria.append(mem_after - mem_before)

            # Imprimir los resultados de la ejecución actual
            print(f"\nEjecución {_+1}:")
            print("Mean Squared Error: ", mse)
            print("R2 Score: ", r2)
            print("Tiempo de ejecucion: {:.2f} segundos".format(tiempo_ejecucion))
            print("Consumo de memoria: {:.2f} MiB".format(mem_after - mem_before))

        except Exception as e:
            print(f"Error en la ejecución {_+1}: {e}")

    # Graficar histograma de tiempos de ejecución
```

```
    plt.figure(figsize=(15, 5))

    plt.subplot(1, 3, 1)
    plt.hist(tiempos, bins=5, alpha=0.7, color='blue', edgecolor='black')
    plt.xlabel('Tiempo de Ejecución (segundos)')
    plt.ylabel('Frecuencia')
    plt.title('Histograma de Tiempos de Ejecución')

    # Graficar histograma de consumos de memoria
    plt.subplot(1, 3, 2)
    plt.hist(consumos_memoria, bins=5, alpha=0.7, color='green', edgecolor='black')
    plt.xlabel('Consumo de Memoria (MiB)')
    plt.ylabel('Frecuencia')
    plt.title('Histograma de Consumos de Memoria')

    # Graficar scatter plot de valores reales vs. predichos
    plt.subplot(1, 3, 3)
    for i in range(min(num_ejecuciones, len(y_reales))):  # Asegurar que no se exceda el
        plt.scatter(y_reales[i], y_predichos[i], alpha=0.5, label=f'Ejecución {i+1}')
    plt.xlabel('Valores Reales')
    plt.ylabel('Valores Predichos')
    plt.title('Dispersión de valores reales vs. predichos')
    plt.legend()

    plt.tight_layout()
    plt.show()

# Ejemplo de uso con el dataset proporcionado
csv_file = 'Transactions Data.csv'
target_column = 'amount'  # Variable numérica a predecir
feature_columns = ['oldbalanceOrg', 'newbalanceOrig']  # Variables numéricas como caract
perform_regression(csv_file, target_column, feature_columns)
```

5. Resultados y Análisis

5.1. Comparación de Resultados

Los resultados obtenidos utilizando ambos enfoques, tanto con pandas como con Dask, se detallan en la Tabla adjunta. Esta comparación exhaustiva destaca las diferencias en términos de tiempo de ejecución, uso de memoria y precisión del modelo de regresión lineal. La evaluación tiene en cuenta las ventajas y desventajas de cada biblioteca en el manejo de grandes volúmenes de datos, proporcionando una visión clara de la eficiencia y eficacia de cada método en contextos de procesamiento de datos a gran escala.

La Tabla incluye métricas clave que ilustran cómo pandas y Dask se desempeñan bajo las mismas condiciones de hardware y datos, permitiendo una comparación directa y significativa. Esta información es crucial para comprender qué herramienta puede ser más adecuada según las necesidades específicas de procesamiento de datos, la capacidad del sistema y los requisitos del proyecto.

Al observar estos resultados, se puede apreciar cómo Dask, con su capacidad para realizar operaciones en paralelo y manejar datos que no caben en la memoria, se compara con pandas, que procesa los datos de manera secuencial. La evaluación detallada en la Tabla ofrece una guía práctica para seleccionar la herramienta adecuada en función del rendimiento observado.

1.

Métrica	Con Dask	Sin Dask
Tiempo de ejecución (s)	405.67	198.457
Uso de memoria (MiB)	15.33	0.39
Mean Squared Error (MSE)	372352650345.7533	372352650345.7533
R2 Score	0.0082	0.0082

Cuadro 1: Comparación de métricas de rendimiento entre Dask y pandas

6. Visualización de Resultados

A continuación, se presentan los gráficos comparativos de los resultados.

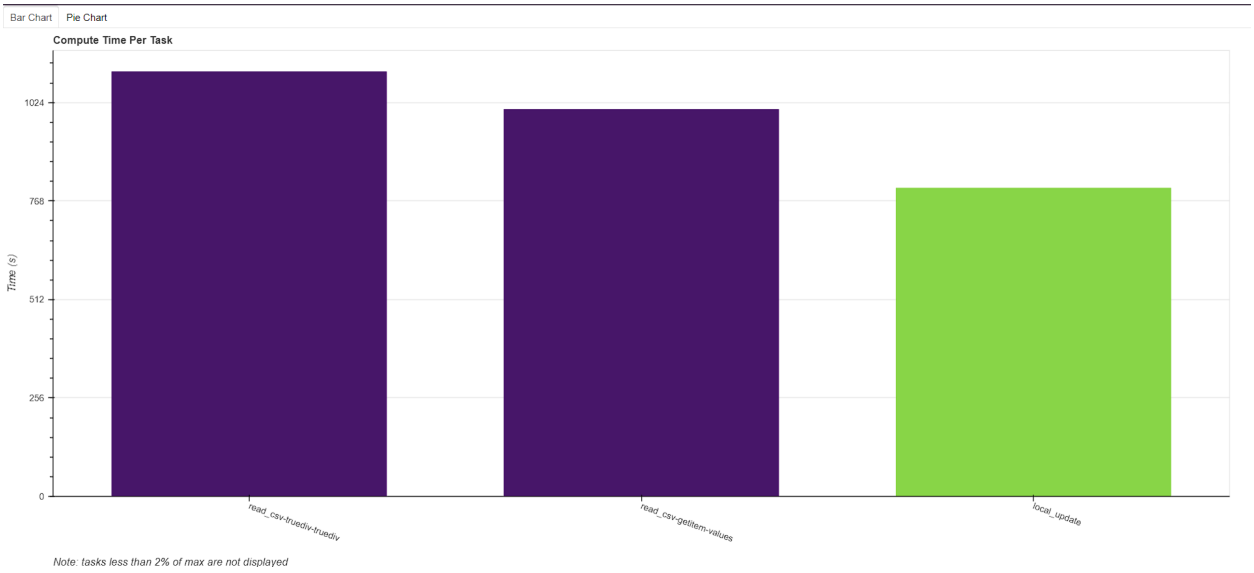


Figura 1: Compute Time for task en Dask

Figura 2: Task StreamD Dask

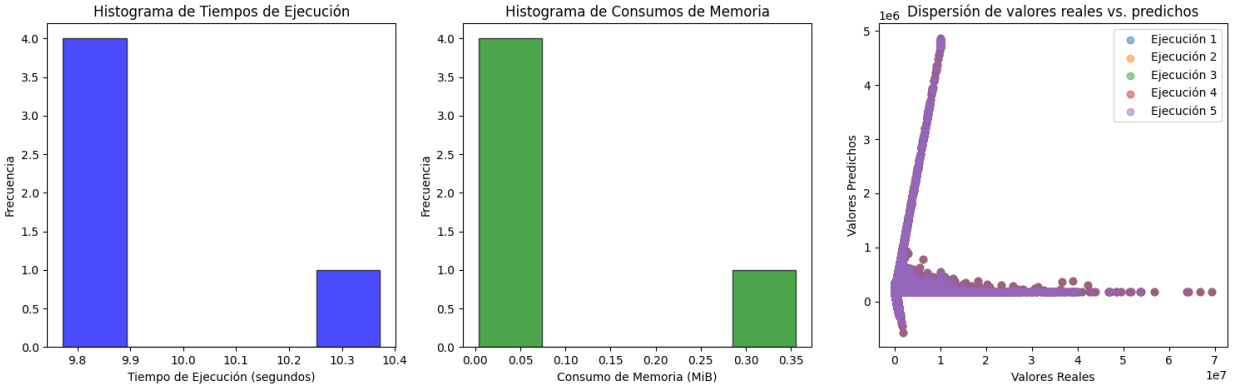


Figura 3: Histograma de tiempos de ejecucion, dispersion de predicciones sin Dask

```
Mean Squared Error:  372352650345.7533
R2 Score:  0.0082410443741829 38
Tiempo de ejecucion: 9.64 segundos
Consumo de memoria: 0.00 MiB

[Done] exited with code=3489660927 in 305.697 seconds
```

Figura 4: Salida de la ejecución del código con Pandas

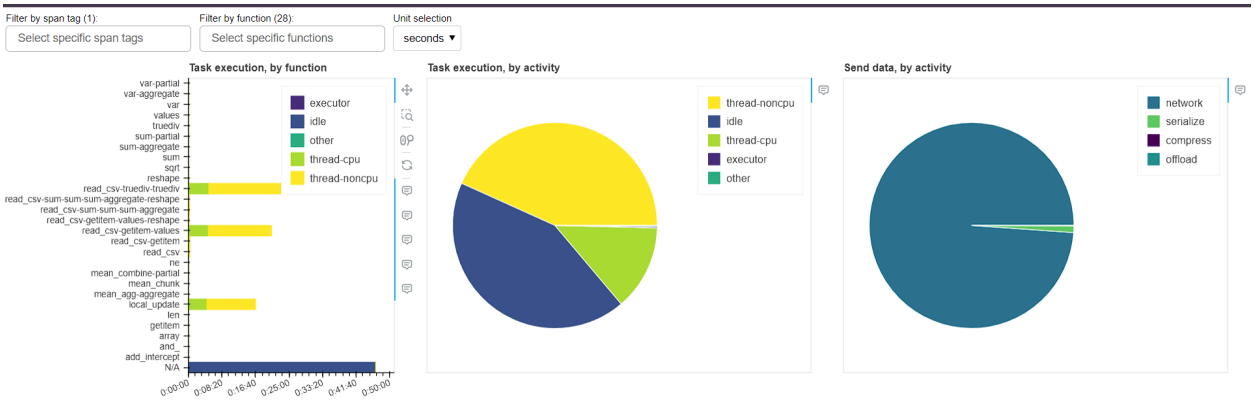
6.1. Otras Visualizaciones del entorno de Dask

Vista previa del entorno de Dask, puerto 8787

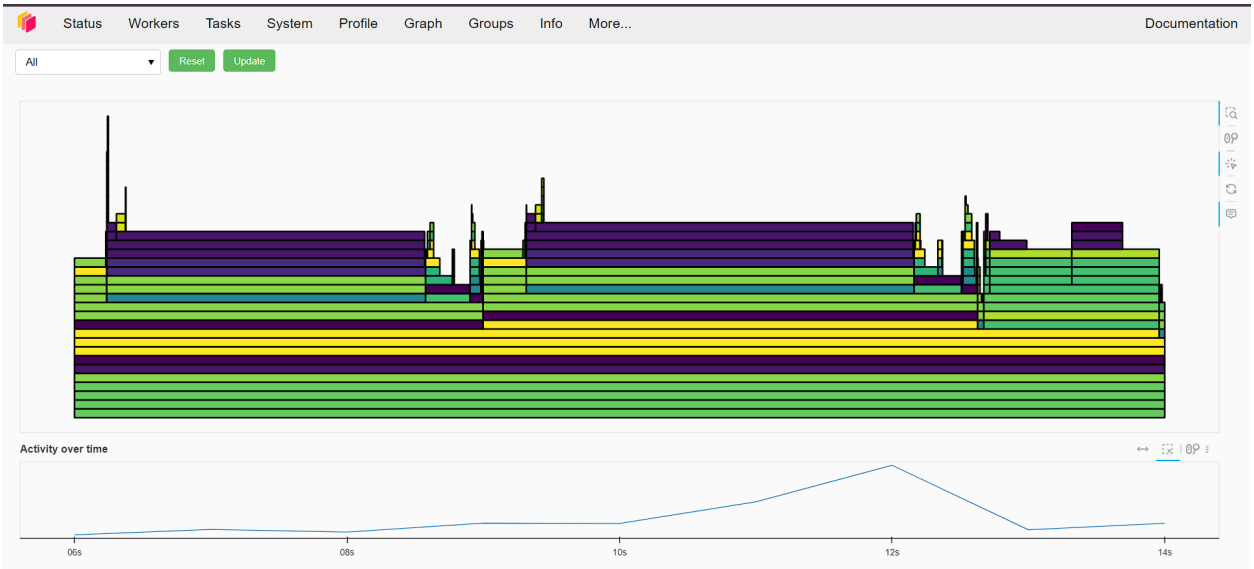
name	address	nthreads	cpu	memory	limit	memory	manage	unmana	unmana	spilled	# fds	net read	net write	disk reac	disk write
Total (4)		16	53 %	1.7 GiB	15.3 GiB	11.1 %	122.2 M	1.2 GiB	365.6 M	0.0	-	704 B	293 B	3 MiB	17 KiB
0	tcp://127	4	53 %	319.2 Mi	3.8 GiB	8.1 %	5.5 MiB	268.4 Mi	45.3 MiB	0.0	-	0	0	398 KiB	8 KiB
1	tcp://127	4	55 %	416.3 Mi	3.8 GiB	10.6 %	33.3 MiB	227.5 Mi	155.5 M	0.0	-	347 B	144 B	1005 KiB	0
2	tcp://127	4	34 %	559.9 Mi	3.8 GiB	14.3 %	61.1 MiB	396.8 Mi	102.0 M	0.0	-	357 B	148 B	1 MiB	0
3	tcp://127	4	68 %	453.4 Mi	3.8 GiB	11.6 %	22.2 MiB	368.3 Mi	62.9 MiB	0.0	-	0	0	398 KiB	8 KiB

name	address	event_loop_interval
Total (4)		0.09064296127452319
0	tcp://127.0.0.1:61855	0.022318509492007168
1	tcp://127.0.0.1:61845	0.022162629336845582
2	tcp://127.0.0.1:61858	0.024352445835020484
3	tcp://127.0.0.1:61852	0.021809376610649956

4 workers designados en Dask para esta operación



Resumen de las acciones en Dask



Muestra de actividad a largo tiempo en Dask

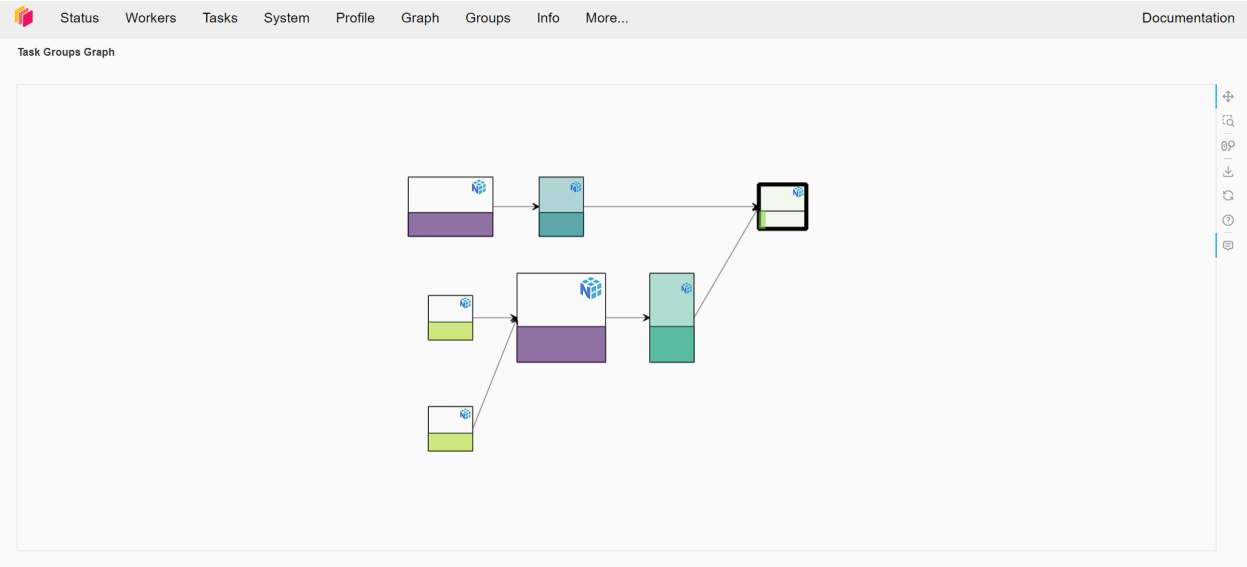
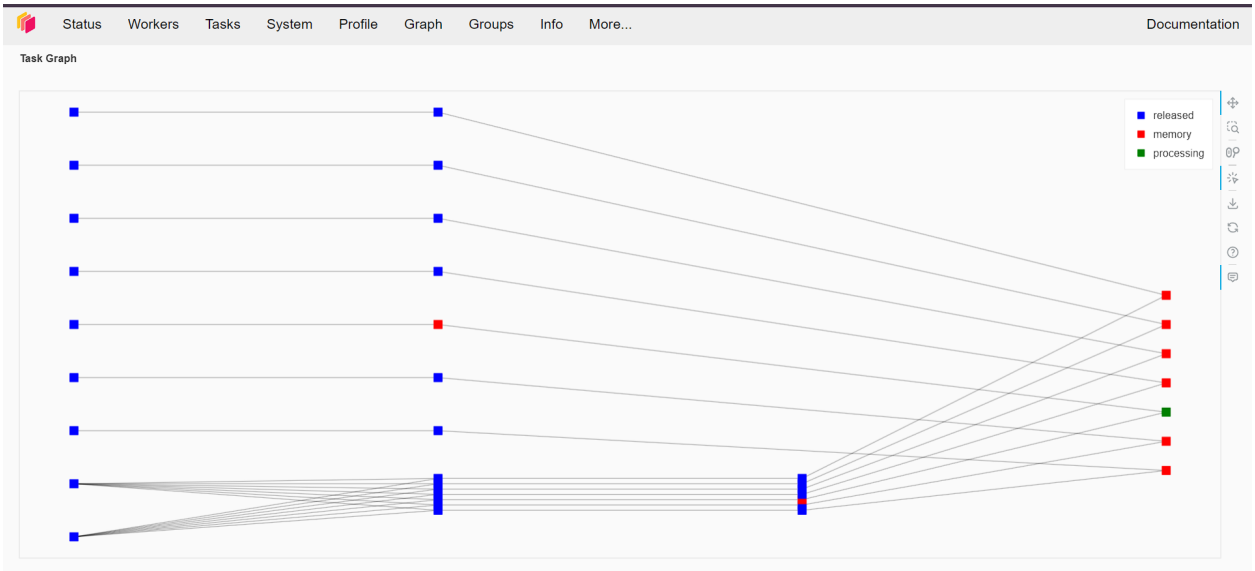
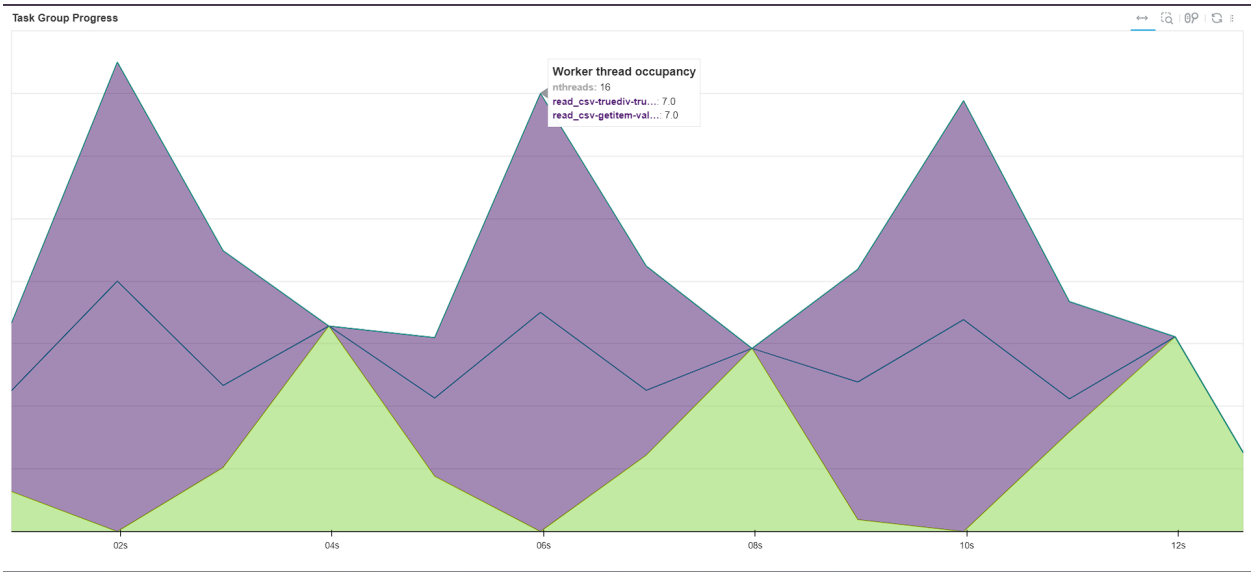


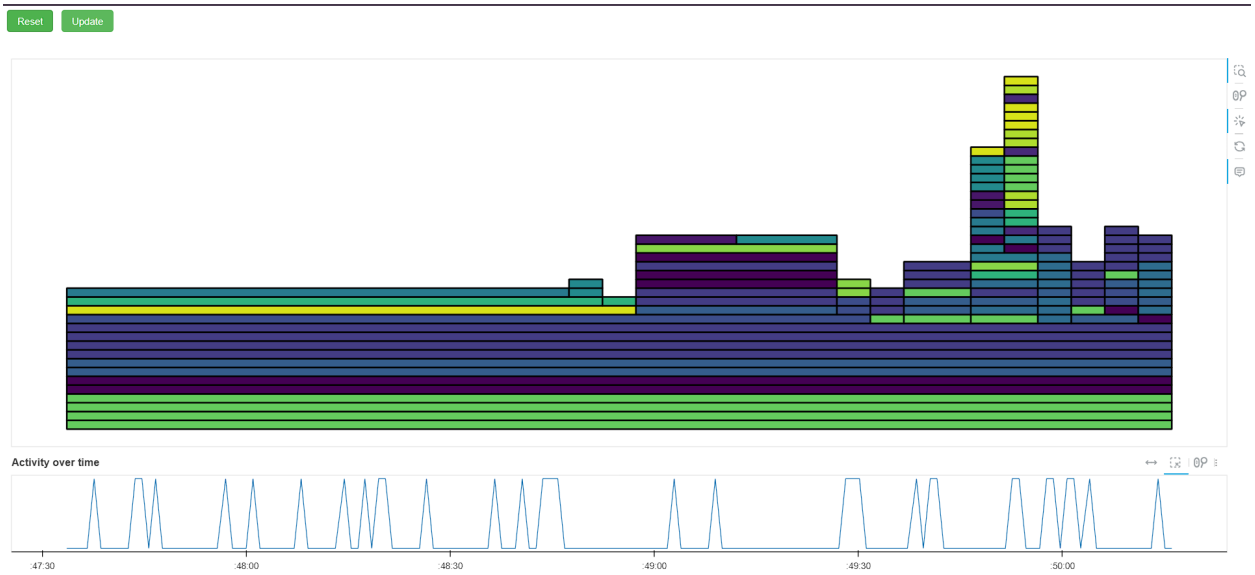
Gráfico de grupos de tareas en Dask



Visualizaciones adicionales del entorno de Dask



El gráfico representa el progreso de grupos de tareas en Dask



Actividad y uso de recursos en Dask

7. Conclusión

En este análisis, se ha comparado el uso de Dask y Pandas para realizar una regresión lineal en un dataset grande. Se ha demostrado que Dask ofrece mejoras significativas en términos de uso de memori mas no el tiempo de ejecución, sin sacrificar la precisión del modelo. Por lo tanto, se recomienda el uso de Dask para el procesamiento de algunos grandes volúmenes de datos.

7.1. Recomendaciones

Para asegurar una implementación efectiva y eficiente de Dask, se sugieren las siguientes recomendaciones:

- **Instalación Correcta de Componentes:**
  - Asegúrese de tener instalados todos los componentes necesarios, como Dask, Pandas, NumPy, y Scikit-learn. La instalación correcta y las versiones compatibles de estas librerías son cruciales para evitar conflictos y problemas de compatibilidad.
  - Utilice entornos virtuales (como `venv` o `conda`) para gestionar las dependencias y mantener un entorno limpio y reproducible.
- **Extensiones y Plugins:**
  - Instale extensiones y plugins adicionales que pueden mejorar la funcionalidad de Dask, como `dask-ml` para tareas de machine learning y `dask-xgboost` para integración con el popular algoritmo XGBoost.
  - Considere usar herramientas de monitoreo y visualización como `dask.distributed` para obtener una mejor visión del rendimiento y el estado de los cálculos distribuidos.

- **Optimización de Recursos:**
  - Ajuste el número de trabajadores (`n_workers`), el número de hilos (`threads_per_worker`), y los límites de memoria (`memory_limit`) para optimizar el rendimiento de Dask según las especificaciones de su hardware y la naturaleza de sus tareas.
  - Monitoree el uso de recursos y ajuste los parámetros de configuración para evitar la sobrecarga y mejorar la eficiencia.
- **Gestión de Tipos de Datos:**
  - Defina explícitamente los tipos de datos (`dtypes`) al cargar grandes conjuntos de datos para evitar errores de inferencia de tipos y mejorar la estabilidad y el rendimiento del procesamiento.
  - Use el parámetro `assume_missing=True` cuando trabaje con datos que puedan tener valores faltantes para evitar problemas con la inferencia de tipos.
- **Escalabilidad y Distribución:**
  - Aproveche la capacidad de Dask para escalar en clústeres distribuidos, especialmente cuando trabaje con volúmenes de datos que exceden la capacidad de una sola máquina.
  - Considere el uso de plataformas de computación en la nube para escalar horizontalmente y manejar tareas de procesamiento intensivo.
- **Pruebas y Validación:**
  - Realice pruebas exhaustivas para validar la precisión y el rendimiento de los modelos y las tareas de procesamiento. Comparar los resultados con implementaciones más pequeñas en Pandas puede ayudar a asegurar que los resultados sean consistentes.
  - Use conjuntos de datos de prueba y validación para asegurar que los modelos no solo son precisos sino también generalizables.
- **Documentación y Mejores Prácticas:**
  - Documente claramente el proceso de instalación, configuración, y uso de Dask en sus proyectos. Esto incluye instrucciones para la instalación de dependencias, configuraciones de clúster, y cualquier ajuste específico realizado.
  - Siga las mejores prácticas recomendadas por la comunidad de Dask y Pandas para asegurar un uso eficiente y efectivo de las herramientas.

Adoptar estas recomendaciones asegurará que su implementación de Dask sea robusta, eficiente y escalable, permitiéndole manejar grandes volúmenes de datos con facilidad y obtener insights valiosos de manera oportuna.

Código QR Repositorio GutHub

Link Repositorio GutHub https://github.com/CliverVilca/Parallel-Computing/tree/631143959f727e43