Haga clic en el procesamiento de datos de predicción de velocidad y el entrenamiento de modelos

NetApp Solutions

NetApp
April 25, 2024

Tabla de contenidos

Haga clic en el procesamiento de datos de predicción de velocidad y el entrenamiento de modelos

Bibliotecas para el procesamiento de datos y el entrenamiento de modelos

En la tabla siguiente se enumeran las bibliotecas y los marcos que se utilizaron para generar esta tarea. Todos estos componentes se han integrado completamente con los controles de seguridad y acceso basados en roles de Azure.

Bibliotecas/marco de trabajo	Descripción
CuML DASK	Para QUE EL ML funcione en la GPU, el "Biblioteca de cuML" Ofrece acceso al paquete cuML DE RAPIDS con DASK. RAPIDS cuML implementa algoritmos DE ML más conocidos, como los métodos de clustering, reducción de dimensiones y regresión, con implementaciones basadas en GPU de alto rendimiento que ofrecen una velocidad de hasta 100 veces superior a los métodos basados en CPU.
DASK cuDF	CuDF incluye varias otras funciones que admiten la extracción, transformación y carga (ETL) acelerada por GPU, como la subconfiguración de datos, transformaciones, codificación en caliente, etc. El equipo DE RAPIDS mantiene un "biblioteca dask-cudf" Eso incluye métodos auxiliares para usar DASK y cuDF.
Formación en Scikit	Scikit-Learn proporciona docenas de algoritmos y modelos de aprendizaje automático integrados, llamados estimadores. Cada uno "estimator" se puede ajustar a algunos datos mediante su "encajar" método.

Hemos utilizado dos cuadernos para construir los gasoductos ML para su comparación; uno es el método convencional de curscikit-aprender de pandas, y el otro es el entrenamiento distribuido con RAPIDS y Dink. Cada portátil se puede probar individualmente para ver el rendimiento en términos de tiempo y escala. Cubrimos cada bloc de notas individualmente para demostrar las ventajas de la formación distribuida con RAPIDS y Dink.

Cargue Criteo haga clic en el día 15 de los registros en pandas y entrena un cikit-aprende el modelo de bosque aleatorio

En esta sección se describe cómo utilizamos Pandas y DASK DataFrames para cargar datos Click Logs del conjunto de datos Criteo Terabyte. El caso de uso es relevante en la

publicidad digital para intercambios de anuncios para crear perfiles de usuarios al predecir si se hará clic en anuncios o si el intercambio no está utilizando un modelo exacto en una canalización automatizada.

Se cargaron los datos del día 15 desde el conjunto de datos Click Logs, sumando 45 GB. Ejecutar la siguiente celda en el portátil Jupyter `CTR-PandasRF-collated.ipynb` Crea un DataFrame de pandas que contiene los primeros 50 millones de filas y genera un modelo de bosque aleatorio cikit-aprender.

```
%%time
import pandas as pd
import numpy as np
header = ['col'+str(i) for i in range (1,41)] #note that according to
criteo, the first column in the dataset is Click Through (CT). Consist of
40 columns
first_row_taken = 50_000_000 # use this in pd.read_csv() if your compute
resource is limited.
# total number of rows in day15 is 20B
# take 50M rows
"""
Read data & display the following metrics:
1. Total number of rows per day
2. df loading time in the cluster
3. Train a random forest model
"""
df = pd.read_csv(file, nrows=first_row_taken, delimiter='\t',
names=header)
# take numerical columns
df_sliced = df.iloc[:, 0:14]
# split data into training and Y
Y = df_sliced.pop('col1') # first column is binary (click or not)
# change df_sliced data types & fillna
df_sliced = df_sliced.astype(np.float32).fillna(0)
from sklearn.ensemble import RandomForestClassifier
# Random Forest building parameters
# n_streams = 8 # optimization
max_depth = 10
n_bins = 16
n_trees = 10
rf_model = RandomForestClassifier(max_depth=max_depth,
n_estimators=n_trees)
rf_model.fit(df_sliced, Y)
```

Para realizar la predicción utilizando un modelo de bosque aleatorio entrenado, ejecute el siguiente párrafo en este cuaderno. Tomamos las últimas filas de un millón del día 15 como conjunto de pruebas para evitar cualquier duplicación. La celda también calcula la precisión de la predicción, definida como el porcentaje de ocurrencias que el modelo predice con precisión si un usuario hace clic o no en un anuncio. Para revisar

cualquier componente desconocido en este cuaderno, consulte "documentación oficial de scikit-aprender".

```
# testing data, last 1M rows in day15
test_file = '/data/day_15_test'
with open(test_file) as g:
    print(g.readline())

# dataFrame processing for test data
test_df = pd.read_csv(test_file, delimiter='\t', names=header)
test_df_sliced = test_df.iloc[:, 0:14]
test_Y = test_df_sliced.pop('col1')
test_df_sliced = test_df_sliced.astype(np.float32).fillna(0)
# prediction & calculating error
pred_df = rf_model.predict(test_df_sliced)
from sklearn import metrics
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(test_Y, pred_df))
```

Cargar día 15 en DASK y entrenar un modelo de bosque aleatorio DASK cuML

De una manera similar a la sección anterior, cargue Criteo Click Logs Day 15 en Pandas y entrena un cikit-aprende el modelo de bosque aleatorio. En este ejemplo, realizamos la carga de DataFrame con DASK cuDF y entrenamos un modelo de bosque aleatorio en DASK cuML. Hemos comparado las diferencias en el tiempo de formación y el escalado en la sección "“Comparación del tiempo de formación”."

criteo_dask_RF.ipynb

Este portátil importa `numpy`, `cuml`, y lo necesario `dask` bibliotecas, como se muestra en el siguiente ejemplo:

```
import cuml
from dask.distributed import Client, progress, wait
import dask_cudf
import numpy as np
import cudf
from cuml.dask.ensemble import RandomForestClassifier as cumlDaskRF
from cuml.dask.common import utils as dask_utils
```

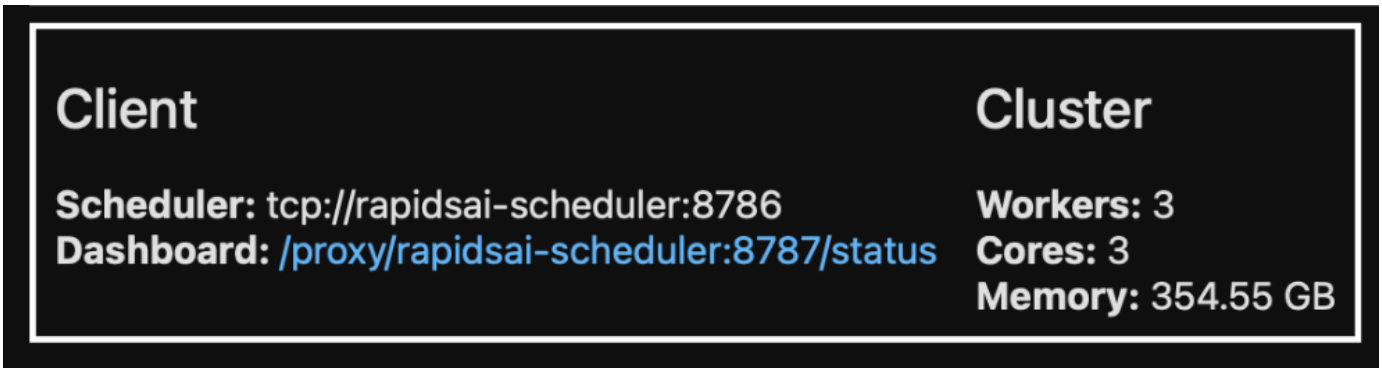
Inicie cliente DASK().

```
client = Client()
```

Si su clúster está configurado correctamente, puede ver el estado de los nodos de trabajo.

```
client
workers = client.has_what().keys()
n_workers = len(workers)
n_streams = 8 # Performance optimization
```

En nuestro clúster AKS, se muestra el siguiente estado:



Tenga en cuenta que DASK emplea el paradigma de ejecución lenta: En lugar de ejecutar el código de procesamiento al instante, DASK crea en su lugar un gráfico cíclico dirigido (DAG) de ejecución. DAG contiene un conjunto de tareas y sus interacciones que cada trabajador necesita ejecutar. Este diseño significa que las tareas no se ejecutan hasta que el usuario le indique a DASK que las ejecute de una forma u otra. Con DASK tiene tres opciones principales:

- **Call comput() en un DataFrame.** esta llamada procesa todas las particiones y, a continuación, devuelve los resultados al planificador para la agregación final y conversión a cuDF DataFrame. Esta opción debe usarse con moderación y sólo en resultados muy reducidos a menos que el nodo del programador se quede sin memoria.
- **Call persistent() en un DataFrame.** esta llamada ejecuta el gráfico, pero, en lugar de devolver los resultados al nodo del planificador, los mantiene en la memoria a través del clúster para que el usuario pueda reutilizar estos resultados intermedios en la canalización sin necesidad de volver a ejecutar el mismo procesamiento.
- **Call head() en un DataFrame.** al igual que con cuDF, esta llamada devuelve 10 registros al nodo del planificador. Esta opción se puede utilizar para comprobar rápidamente si el DataFrame contiene el formato de salida deseado o si los propios registros tienen sentido, en función del procesamiento y cálculo.

Por lo tanto, a menos que el usuario llama a cualquiera de estas acciones, los trabajadores se sientan inactivos esperando que el programador inicie el procesamiento. Este paradigma de ejecución perezosa es común en marcos informáticos modernos en paralelo y distribuidos como Apache Spark.

En el siguiente párrafo se entrena un modelo de bosque aleatorio mediante el uso de DASK cuML para computación acelerada por GPU distribuida y se calcula la precisión de predicción del modelo.

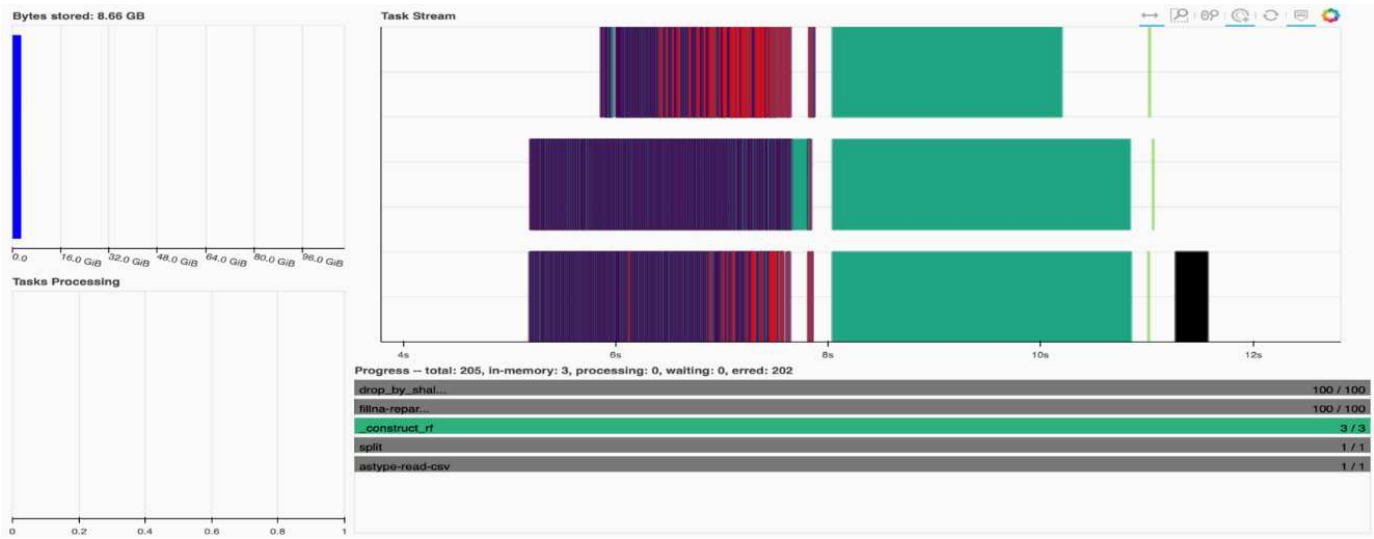
```
Adsf
# Random Forest building parameters
n_streams = 8 # optimization
max_depth = 10
n_bins = 16
n_trees = 10
cuml_model = cumlDaskRF(max_depth=max_depth, n_estimators=n_trees,
n_bins=n_bins, n_streams=n_streams, verbose=True, client=client)
cuml_model.fit(gdf_sliced_small, Y)
# Model prediction
pred_df = cuml_model.predict(gdf_test)
# calculate accuracy
cu_score = cuml.metrics.accuracy_score( test_y, pred_df )
```

Supervisión de tarea mediante el panel de control de flujos de tareas nativo

La "Planificador distribuido DASK" proporciona comentarios en directo de dos formas:

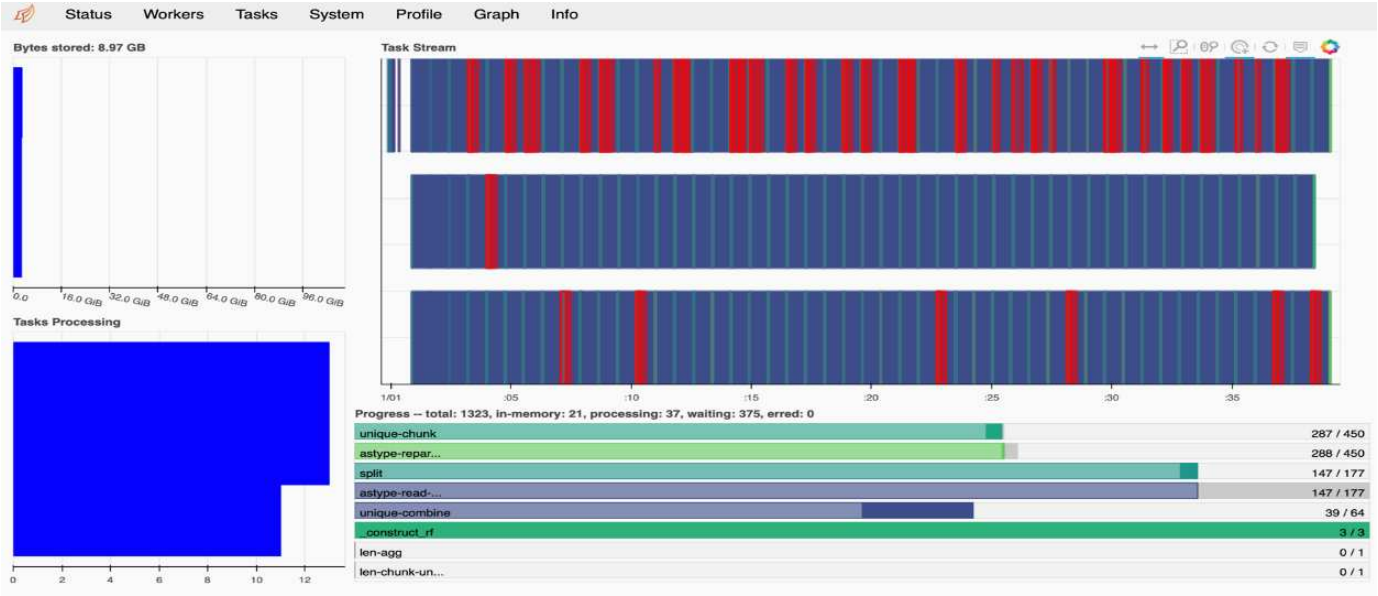
- Un panel interactivo que contiene muchos trazados y tablas con información en directo
- Una barra de progreso adecuada para uso interactivo en consolas o portátiles

En nuestro caso, la siguiente figura muestra cómo puede supervisar el progreso de la tarea, incluidos los bytes almacenados, el flujo de tareas con un desglose detallado del número de flujos y el progreso por los nombres de tareas con las funciones asociadas ejecutadas. En nuestro caso, debido a que tenemos tres nodos de trabajo, hay tres partes principales del flujo y los códigos de color denotan diferentes tareas dentro de cada flujo.



Tiene la opción de analizar tareas individuales y examinar el tiempo de ejecución en milisegundos o identificar cualquier obstáculo o impedimento. Por ejemplo, la siguiente figura muestra los flujos de tareas para la etapa de ajuste del modelo de bosque aleatorio. Se están ejecutando muchas más funciones, incluido el fragmento único para el procesamiento de DataFrame, _construct_rf para ajustar el bosque aleatorio, etc. La mayor parte

del tiempo se ha empleado en operaciones DataFrame debido al gran tamaño (45GB) de un día de datos de los registros de clic de Criteo.



Comparación del tiempo de entrenamiento

Esta sección compara el tiempo de entrenamiento del modelo utilizando pandas convencionales en comparación con el DASK. Para Pandas, cargamos una cantidad menor de datos debido a la naturaleza del tiempo de procesamiento más lento, para evitar que se desbordara la memoria. Por lo tanto, interpolamos los resultados para ofrecer una comparación justa.

La siguiente tabla muestra la comparación del tiempo de entrenamiento bruto cuando hay significativamente menos datos utilizados para el modelo de bosque aleatorio de pandas (50 millones de filas de 20 mil millones por día 15 del conjunto de datos). Esta muestra sólo utiliza menos del 0.25% de todos los datos disponibles. Mientras que para DASK-cuML entrenamos el modelo de bosque aleatorio en las 20 mil millones de filas disponibles. Los dos enfoques dieron lugar a un tiempo de capacitación comparable.

Enfoque	Tiempo de entrenamiento
Scikit-Learn: Usando sólo 50 m de filas en el día 15 como datos de entrenamiento	47 minutos y 21 segundos
RAPIDS-Dask: Utilizando todas las filas 20B del día 15 como datos de entrenamiento	1 hora, 12 minutos y 11 segundos

Si interpolamos los resultados del tiempo de entrenamiento linealmente, como se muestra en la siguiente tabla, hay una ventaja significativa a utilizar el entrenamiento distribuido con DASK. Tomaría el enfoque convencional de Pandas scikit-Learn 13 días para procesar y entrenar 45GB de datos para un solo día de registros tecleo, mientras que EL enfoque RAPIDS-DASk procesa la misma cantidad de datos 262.39 veces más rápido.

Enfoque	Tiempo de entrenamiento
Scikit-Learn: Usando todas las filas 20B en el día15 como datos de entrenamiento	13 días, 3 horas, 40 minutos y 11 segundos

Enfoque	Tiempo de entrenamiento
RAPIDS-Dask: Utilizando todas las filas 20B del día 15 como datos de entrenamiento	1 hora, 12 minutos y 11 segundos

En la tabla anterior, puede ver que usando RAPIDS con Dink para distribuir el procesamiento de datos y el entrenamiento de modelos en varias instancias de GPU, el tiempo de ejecución es significativamente más corto en comparación con el procesamiento convencional de Pandas DataFrame con el entrenamiento de modelos scikit-Learn. Este marco permite un escalado vertical y horizontal en el cloud, así como en las instalaciones, en un clúster multinodo con varias GPU.

Monitorizar Dink y RAPIDS con Prometheus y Grafana

Una vez que todo se pone en marcha, ejecute inferencias sobre nuevos datos. Los modelos predicen si un usuario hace clic en un anuncio basado en actividades de navegación. Los resultados de la predicción se almacenan en un cuDF de DASK. Puede supervisar los resultados con Prometheus y visualizar en paneles Grafana.

Para obtener más información, consulte este tema "RAPIDS AI Media Post".

Creación de versiones de conjuntos de datos y modelos con el kit de herramientas de operaciones de datos de NetApp

El kit de herramientas DataOPS de NetApp para Kubernetes abstrae los recursos de almacenamiento y las cargas de trabajo de Kubernetes hasta el nivel de espacio de trabajo de ciencia de datos. Estas funciones se presentan en una interfaz sencilla y fácil de usar diseñada para científicos e ingenieros de datos. Utilizando la forma familiar de un programa de Python, el kit de herramientas permite a científicos e ingenieros de datos aprovisionar y destruir espacios de trabajo de JuppyterLab en cuestión de segundos. Estas áreas de trabajo pueden contener terabytes o incluso petabytes de capacidad de almacenamiento, lo que permite a los científicos de datos almacenar todos sus conjuntos de datos de entrenamiento directamente en sus espacios de trabajo de proyectos. Han pasado los días de gestionar los espacios de trabajo y los volúmenes de datos por separado.

Para obtener más información, visite el Kit de herramientas "Repositorio de GitHub".

Portátiles Juppyter para referencias

Existen dos cuadernos Juppyter asociados a este informe técnico:

- **"CTR-PandasRF-collated.ipynb."** Este cuaderno carga el día 15 desde el conjunto de datos de registros Criteo Terabyte Click, procesa y formatea datos en un DataFrame de Pandas, entrena un modelo de bosque aleatorio Scikit-Learn, realiza predicción y calcula la precisión.
- **"criteo_dask_RF.ipynb."** Este cuaderno carga el día 15 desde el conjunto de datos de registros Criteo Terabyte Click, procesa y formatea datos en un CuDF DASK, entrena un modelo de bosque aleatorio

DASK cuML, realiza predicción y calcula la precisión. Al aprovechar varios nodos de trabajo con GPU, este método de procesamiento y entrenamiento de datos distribuidos y modelos es altamente eficiente. Cuantos más datos procese, mayor será el ahorro de tiempo que se consigue con el método DE ML convencional. Puede implementar este portátil en el cloud, en las instalaciones o en un entorno híbrido en el que el clúster de Kubernetes contenga recursos informáticos y de almacenamiento en diferentes ubicaciones, siempre y cuando su configuración de red permita el movimiento libre de datos y la distribución de modelos.