Klicken Sie sich durch die Verarbeitung der Tarifprognosen und das Modelltraining

NetApp Solutions

NetApp
April 25, 2024

Inhalt

Klicken Sie sich durch die Verarbeitung der Tarifprognosen und das Modelltraining

Bibliotheken für die Datenverarbeitung und das Modelltraining

In der folgenden Tabelle sind die Bibliotheken und Frameworks aufgeführt, die zum Erstellen dieser Aufgabe verwendet wurden. Alle diese Komponenten wurden vollständig in die rollenbasierte Zugriffs- und Sicherheitskontrolle von Azure integriert.

Bibliotheken/Framework	Beschreibung
Fragen Sie die CuML	Für ML zur Arbeit an GPU, das "CuML-Bibliothek" Bietet Zugriff auf DAS RAPIDS CuML-Paket mit DASK. RAPIDS CuML implementiert gängige ML-Algorithmen wie Clustering, Dimensionierungsreduzierung und Regression. Hochperformante GPU-basierte Implementierungen ermöglichen eine bis zu 100-fache Geschwindigkeit gegenüber CPU-basierten Ansätzen.
Fragen Sie nach cuDF	CuDF umfasst verschiedene andere Funktionen, die GPU-beschleunigte Extraktion, Transformation, Last (ETL) unterstützen, wie z. B. Untersetzung von Daten, Transformationen, One-Hot-Codierung und mehr. Das Team VON RAPIDS unterhält eine "Dask-cudf Bibliothek" Dazu gehören Hilfsmethoden für die Verwendung von Fragen und CuDF.
Ischikit Lernen	Scikit-Learn bietet Dutzende von integrierten Algorithmen und Modellen für maschinelles Lernen an, die als Schätzer bezeichnet werden. Beide "kostenplaner" Kann mit dem an einige Daten angepasst werden "Fit" Methode.

Zum Vergleich haben wir zwei Notebooks für DIE ERSTELLUNG der ML-Pipelines eingesetzt, eines ist der konventionelle Pandas-Scikit-Learn-Ansatz, und das andere ist Distributed Training mit RAPIDS und Dask. Jedes Notebook kann einzeln getestet werden, um die Leistung in Bezug auf Zeit und Umfang zu sehen. Wir decken jedes Notebook einzeln ab, um die Vorteile des Distributed Trainings mit RAPIDS und Dask zu demonstrieren.

Criteo Click Logs Tag 15 in Pandas laden und ein scikit-Learn Zufallswaldmodell trainieren

In diesem Abschnitt wird beschrieben, wie wir Pandas und DataFrames zum Laden von Click Logs-Daten aus dem Criteo Terabyte-Datensatz verwendet haben. Der Anwendungsfall ist in der digitalen Werbung relevant, damit Anzeigenaustausch Nutzer-Profile bauen kann, indem er vorhersagt, ob Werbeanzeigen angeklickt werden oder wenn der Austausch kein genaues Modell in einer automatisierten Pipeline verwendet.

Wir haben Tag 15 Daten aus dem Click Logs Datensatz geladen, insgesamt 45GB. Ausführen der folgenden Zelle im Jupyter-Notebook `CTR-PandasRF-collated.ipynb` Erstellt einen Pandas DataFrame, der die ersten 50 Millionen Zeilen enthält und ein scikit-Learn Zufallswaldmodell erzeugt.

```
%%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)
```

Führen Sie den folgenden Abschnitt in diesem Notizbuch aus, um eine Vorhersage mit einem trainierten zufälligen Waldmodell durchzuführen. Wir haben die letzten 1 Million Zeilen ab Tag 15 als Testsatz genommen, um mögliche Duplikate zu vermeiden. Die Zelle berechnet auch die Genauigkeit der Vorhersage, definiert als Prozentsatz der Vorkommen das Modell sagt genau aus, ob ein Benutzer auf eine Anzeige klickt oder nicht. Informationen zu unbekannten Komponenten in diesem Notebook finden Sie im "Offizielle Scikit-Learn-Dokumentation".

```
# 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))
```

Laden Sie Tag 15 in Damast und trainieren Sie ein Damast CuML zufälligen Wald Modell

Ähnlich wie im vorherigen Abschnitt, laden Criteo Click Logs Tag 15 in Pandas und trainieren ein scikit-Learn Zufallswaldmodell. In diesem Beispiel haben wir DataFrame-Beladung mit Dask cuDF durchgeführt und ein zufälliges Waldmodell in Dask cuML trainiert. Im Abschnitt wurden die Unterschiede in Trainingszeit und Umfang verglichen "„Zeitvergleich Training“."

criteo_dask_RF.ipynb

Dieses Notizbuch importiert `numpy`, `cuml`, Und das nötige `dask` Bibliotheken, wie im folgenden Beispiel gezeigt:

```
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
```

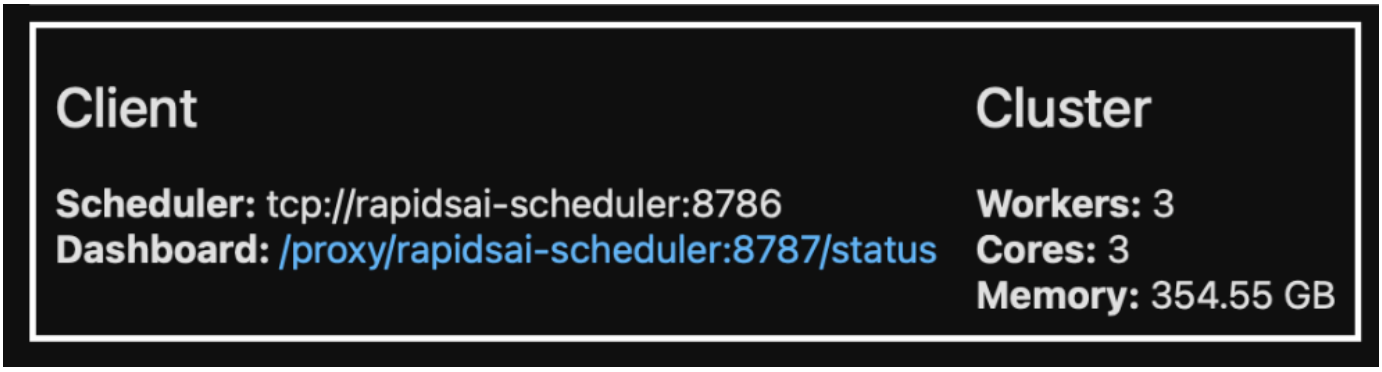
Dask Client() Initiieren.

```
client = Client()
```

Wenn das Cluster ordnungsgemäß konfiguriert ist, können Sie den Status von „Worker“-Nodes anzeigen.

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

In unserem AKS-Cluster wird folgender Status angezeigt:



Beachten Sie, dass Dask das faule Ausführungsmodell verwendet: Anstatt den Verarbeitungscode sofort auszuführen, erstellt Dask stattdessen ein gesteuertes Acyclic Graph (DAG) der Ausführung. DAG umfasst eine Reihe von Aufgaben und ihre Interaktionen, die jeder Mitarbeiter ausführen muss. Dieses Layout bedeutet, dass die Tasks nicht ausgeführt werden, bis der Benutzer DASK anweist, sie auf die eine oder andere Weise auszuführen. Mit DASK haben Sie drei Hauptoptionen:

- **Aufruf Compute() auf einem DataFrame.** dieser Aufruf verarbeitet alle Partitionen und gibt die Ergebnisse dann in den Scheduler für die abschließende Aggregation und Konvertierung in cuDF DataFrame zurück. Diese Option sollte sparsam und nur bei stark reduzierten Ergebnissen verwendet werden, es sei denn, Ihr Scheduler-Node verfügt über keinen Arbeitsspeicher.
- **Call persist() auf einem DataFrame.** dieser Aufruf führt die Grafik aus, aber anstatt die Ergebnisse an den Scheduler-Knoten zurückzugeben, wird sie über den Cluster im Speicher verwaltet, sodass der Benutzer diese Zwischenergebnisse in der Pipeline wiederverwenden kann, ohne dass eine erneute Verarbeitung erforderlich ist.
- **Call Head() auf einem DataFrame.** wie bei cuDF gibt dieser Aufruf 10 Datensätze zurück zum Scheduler-Knoten. Mit dieser Option können Sie schnell überprüfen, ob Ihr DataFrame das gewünschte Ausgabeformat enthält oder ob die Datensätze selbst in Abhängigkeit von Ihrer Verarbeitung und Berechnung sinnvoll sind.

Wenn der Benutzer eine dieser Aktionen nicht anruft, warten die Arbeiter daher nicht, bis der Planer die Verarbeitung initiiert hat. Dieses faul Ausführungsparadigma ist häufig in modernen Parallel- und Distributed Computing Frameworks wie Apache Spark zu finden.

Im folgenden Abschnitt wird ein Zufallswaldmodell mit Hilfe von Dask cuML für verteiltes GPU-beschleunigtes Computing trainiert und die Genauigkeit der Modellvorhersage berechnet.

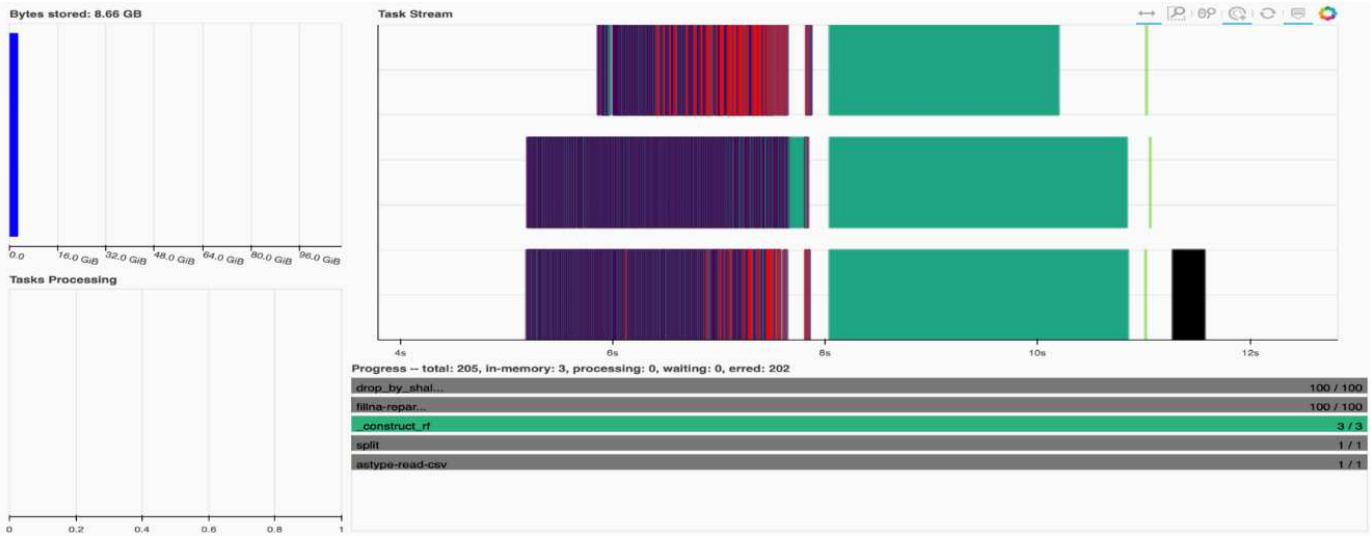
```
Adsf
# Random Forest building parameters
n_streams = 8 # optimization
max_depth = 10
n_bins = 16
n_trees = 10
cuml_model = cumlDaskRF(max_depth=max_depth, n_estimators=n_trees,
n_bins=n_bins, n_streams=n_streams, verbose=True, client=client)
cuml_model.fit(gdf_sliced_small, Y)
# Model prediction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

Überwachung des DASK mit dem systemeigenen Task-Streams-Dashboard

Der "Distributed Scheduler abfragen" Live-Feedback in zwei Formen:

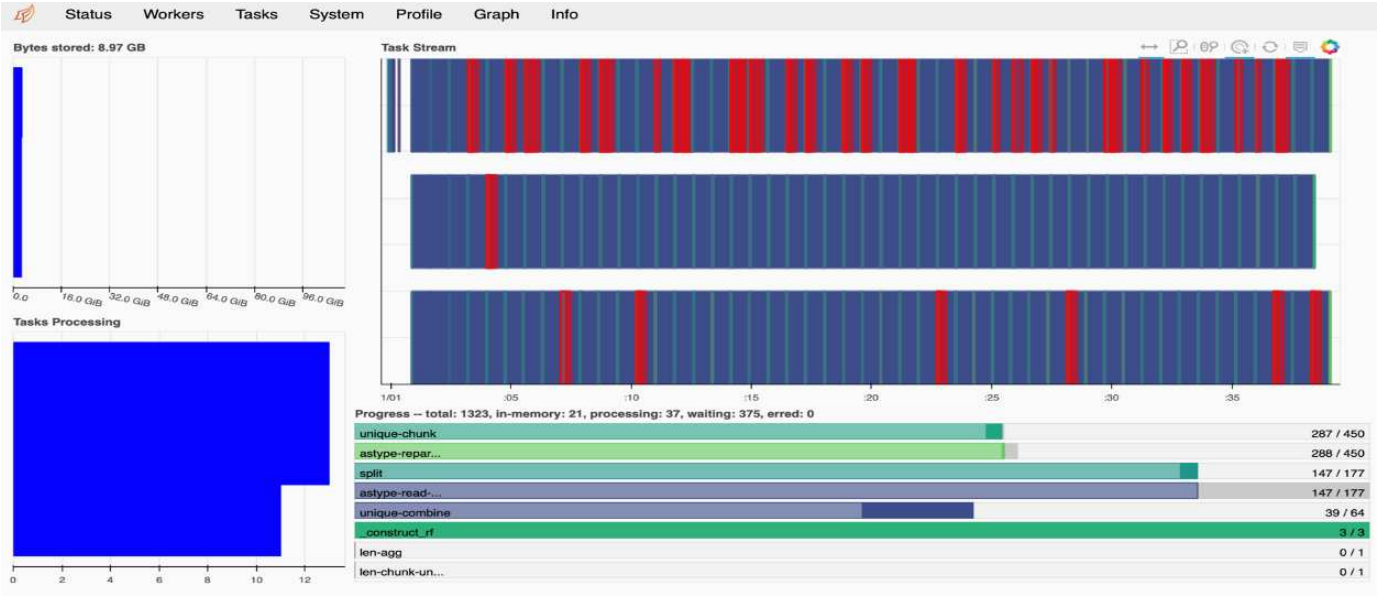
- Ein interaktives Dashboard mit vielen Darstellungen und Tabellen mit Live-Informationen
- Eine Fortschrittsleiste, die für den interaktiven Einsatz in Konsolen oder Notebooks geeignet ist

In unserem Fall zeigt die folgende Abbildung, wie Sie den Fortschritt der Aufgabe überwachen können, einschließlich gespeicherter Bytes, den Task-Stream mit einer detaillierten Aufschlüsselung der Anzahl von Streams und den Fortschritt nach Aufgabennamen mit den ausgeführten Funktionen. In unserem Fall, weil wir drei Arbeiter-Knoten haben, gibt es drei Hauptblöcke des Streams und die Farbcodes bezeichnen verschiedene Aufgaben innerhalb jedes Streams.



Sie haben die Möglichkeit, einzelne Aufgaben zu analysieren und die Ausführungszeit in Millisekunden zu untersuchen oder Hindernisse oder Hindernisse zu identifizieren. Die folgende Abbildung zeigt beispielsweise die Aufgabenströme für die Zufallsphase des Waldmodells. Es werden wesentlich mehr Funktionen ausgeführt, darunter ein einzigartiger Chunk für die DataFrame-Verarbeitung, _construct_rf für die Anpassung der

zufälligen Gesamtstruktur usw. Die meiste Zeit wurde auf DataFrame-Operationen aufgrund der großen Größe (45 GB) von einem Tag Daten aus den Criteo Click Logs verwendet.



Vergleich der Schulungszeit

Dieser Abschnitt vergleicht die Modelltrainings-Zeit mit konventionellem Pandas im Vergleich zu Dask. Bei Pandas haben wir eine kleinere Menge an Daten geladen, weil die Verarbeitungszeit langsamer ist, um Speicherüberlauf zu vermeiden. Daher haben wir die Ergebnisse interpoliert, um einen fairen Vergleich zu bieten.

Die folgende Tabelle zeigt den Vergleich der Rohtrainings-Zeiten, wenn für das Modell der zufälligen Wälder von Pandas deutlich weniger Daten benötigt werden (50 Millionen Zeilen von 20 Milliarden pro Tag 15 des Datensatzes). Diese Stichprobe benötigt nur weniger als 0.25 % aller verfügbaren Daten. Während wir für Dask-cuML das Zufallswaldmodell auf allen 20 Milliarden verfügbaren Reihen trainiert haben. Die beiden Ansätze ergaben eine vergleichbare Trainingszeit.

Vorgehensweise	Schulungszeit
Scikit-Learn: Verwendung von nur 50M Reihen im Tag15 als Trainingsdaten	47 Minuten und 21 Sekunden
RAPIDS-DASK: Alle 20B-Reihen täglich an15 als Trainingsdaten nutzen	1 Stunde, 12 Minuten und 11 Sekunden

Wenn wir die Ergebnisse der Trainingszeit linear interpolieren, wie in der folgenden Tabelle dargestellt, bietet sich ein bedeutender Vorteil für die Verwendung verteilter Schulungen mit Dask. Es würde 13 Tage dauern, bis der konventionelle Pandas scikit-Lernansatz 45GB Daten für einen einzigen Tag mit Klick-Protokollen verarbeitet und trainiert, während der RAPIDS-DASK-Ansatz die gleiche Datenmenge 262.39-mal schneller verarbeitet.

Vorgehensweise	Schulungszeit
Scikit-Learn: Alle 20B-Reihen im Tag15 als Trainingsdaten verwenden	13 Tage, 3 Stunden, 40 Minuten und 11 Sekunden

Vorgehensweise	Schulungszeit
RAPIDS-DASK: Alle 20B-Reihen täglich an15 als Trainingsdaten nutzen	1 Stunde, 12 Minuten und 11 Sekunden

In der vorherigen Tabelle ist zu sehen, dass durch die Verwendung VON RAPIDS mit DASK die Datenverarbeitung und das Modelltraining über mehrere GPU-Instanzen hinweg deutlich kürzer ist als bei der konventionellen Pandas DataFrame-Verarbeitung mit scikit-Learn Modelltrainings. Dieses Framework ermöglicht sowohl vertikale als auch horizontale Skalierung in der Cloud als auch On-Premises in einem Multi-Node-Cluster mit mehreren GPUs.

Überwachen Sie „Dask and RAPIDS“ mit Prometheus und Grafana

Nutzen Sie nach der Implementierung alles Rückschlüsse auf neue Daten. Die Modelle prognostizieren, ob ein Benutzer basierend auf Browsing-Aktivitäten auf eine Anzeige klickt. Die Ergebnisse der Vorhersage werden in einem Damask cuDF gespeichert. Sie können die Ergebnisse mit Prometheus überwachen und in Grafana Dashboards visualisieren.

Weitere Informationen finden Sie unter "RAPIDS AI Medium Post".

Datensatz- und Modellversionierung mit NetApp DataOps Toolkit

Das NetApp DataOps Toolkit für Kubernetes abstrahiert Storage-Ressourcen und Kubernetes-Workloads bis auf die Ebene des Data Science Workspace. Diese Funktionen sind in einer einfachen, benutzerfreundlichen Oberfläche zusammengestellt, die für Data Scientists und Data Engineers konzipiert wurde. Mit der bekannten Form eines Python-Programms ermöglicht das Toolkit Data Scientists und Ingenieuren die Bereitstellung und Zerstörung von JupyterLab-Arbeitsbereichen in nur wenigen Sekunden. Diese Workspaces können Storage-Kapazität im Terabyte- oder sogar Petabyte-Bereich enthalten. Data Scientists können all ihre Trainingsdatensätze direkt in ihren Projektarbeitsbereichen speichern. Die Tage für das separate Management von Workspaces und Daten-Volumes sind vorbei.

Weitere Informationen finden Sie im Toolkit "GitHub Repository".

Jupyter-Notebooks als Referenz

Diesem technischen Bericht sind zwei Jupyter Notebooks zugeordnet:

- **"CTR-PandasRF-collated.ipynb."** Dieses Notebook lädt Tag 15 aus dem Criteo Terabyte Click Protokolldatensatz, verarbeitet und formatiert Daten in einen Pandas DataFrame, trainiert ein Scikit-Learn Zufallswaldmodell, führt Vorhersage aus und berechnet Genauigkeit.
- **"criteo_dask_RF.ipynb."** Dieses Notebook lädt Tag 15 aus dem Criteo Terabyte Click Log Datensatz, verarbeitet und formatiert Daten in einen Damast CuDF, trainiert ein Dusk CuML Zufallswaldmodell, führt

Vorhersage aus und berechnet Genauigkeit. Durch die Nutzung von mehreren Worker-Nodes mit GPUs ist dieser verteilte Daten- und Modellverarbeitungs- und Trainingsansatz äußerst effizient. Je mehr Daten Sie verarbeiten, desto größer ist die Zeitersparnis im Vergleich zu einem herkömmlichen ML-Ansatz. Dieses Notebook lässt sich in der Cloud, vor Ort oder in einer hybriden Umgebung bereitstellen, in der Ihr Kubernetes-Cluster Computing und Storage an verschiedenen Standorten enthält, sofern Ihr Netzwerk-Setup Daten und die Modellverteilung frei ermöglicht.