



Load Day 15 in Dask and train a Dask cuML random forest model

NetApp Solutions

NetApp
August 18, 2021

This PDF was generated from https://docs.netapp.com/us-en/netapp-solutions/ai/aks-anf_load_day_15_in_dask_and_train_a_dask_cuml_random_forest_model.html on August 18, 2021. Always check docs.netapp.com for the latest.

Table of Contents

Load Day 15 in Dask and train a Dask cuML random forest model	1
criteo_dask_RF.ipynb	1

Load Day 15 in Dask and train a Dask cuML random forest model

[Previous: Load Criteo Click Logs day 15 in Pandas and train a scikit-learn random forest model.](#)

In a manner similar to the previous section, load Criteo Click Logs day 15 in Pandas and train a scikit-learn random forest model. In this example, we performed DataFrame loading with Dask cuDF and trained a random forest model in Dask cuML. We compared the differences in training time and scale in the section [“Training time comparison.”](#)

criteo_dask_RF.ipynb

This notebook imports `numpy`, `cuml`, and the necessary `dask` libraries, as shown in the following example:

```
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
```

Initiate Dask Client().

```
client = Client()
```

If your cluster is configured correctly, you can see the status of worker nodes.

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

In our AKS cluster, the following status is displayed:

Client

Scheduler: `tcp://rapidsai-scheduler:8786`
Dashboard: [/proxy/rapidsai-scheduler:8787/status](#)

Cluster

Workers: 3
Cores: 3
Memory: 354.55 GB

Note that Dask employs the lazy execution paradigm: rather than executing the processing code instantly, Dask builds a Directed Acyclic Graph (DAG) of execution instead. DAG contains a set of tasks and their interactions that each worker needs to run. This layout means the tasks do not run until the user tells Dask to execute them in one way or another. With Dask you have three main options:

- **Call `compute()` on a `DataFrame`.** This call processes all the partitions and then returns results to the scheduler for final aggregation and conversion to cuDF `DataFrame`. This option should be used sparingly and only on heavily reduced results unless your scheduler node runs out of memory.
- **Call `persist()` on a `DataFrame`.** This call executes the graph, but, instead of returning the results to the scheduler node, it maintains them across the cluster in memory so the user can reuse these intermediate results down the pipeline without the need for rerunning the same processing.
- **Call `head()` on a `DataFrame`.** Just like with cuDF, this call returns 10 records back to the scheduler node. This option can be used to quickly check if your `DataFrame` contains the desired output format, or if the records themselves make sense, depending on your processing and calculation.

Therefore, unless the user calls either of these actions, the workers sit idle waiting for the scheduler to initiate the processing. This lazy execution paradigm is common in modern parallel and distributed computing frameworks such as Apache Spark.

The following paragraph trains a random forest model by using Dask cuML for distributed GPU-accelerated computing and calculates model prediction accuracy.

```
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 )
```

[Next: Monitor Dask using native Task Streams dashboard.](#)

Copyright Information

Copyright © 2021 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means-graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system-without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.