

Using GPUs for Cloudera Machine Learning Projects

Date published: 2020-07-16

Date modified: 2024-07-31



Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Using GPUs for Cloudera Machine Learning projects.....	4
Heterogeneous GPU clusters.....	4
Testing GPU Setup.....	7

Using GPUs for Cloudera Machine Learning projects

A GPU is a specialized processor that can be used to accelerate highly parallelized computationally-intensive workloads. Because of their computational power, GPUs have been found to be particularly well-suited to [deep learning](#) workloads. Ideally, CPUs and GPUs are used in tandem for data engineering and data science workloads. A typical machine learning workflow involves data preparation, model training, model scoring, and model fitting. You can use existing general-purpose CPUs for each stage of the workflow, and optionally accelerate the math-intensive steps with the selective application of special-purpose GPUs. For example, GPUs allow you to accelerate model fitting using frameworks such as [Tensorflow](#), [PyTorch](#), and [Keras](#).



Important: Starting with Cloudera Machine Learning (CML) 1.5.3, GPU nodes are verified to be working on CentOS 7.9 and RHEL 8.8.

By enabling GPU support, data scientists can share GPU resources available on Cloudera Machine Learning Workspaces. Users can request a specific number of GPU instances, up to the total number available, which are then allocated to the running session or job for the duration of the run.

For information on installing your GPUs, see *Cloudera Private Cloud Installation Software Requirements*, below.

Related Information

[CDP Private Cloud Experiences Installation Software Requirements](#)

[Provision an ML Workspace](#)

[Testing GPU Setup](#)

[GPU node setup](#)

Heterogeneous GPU clusters

When using heterogeneous GPU clusters to run sessions and jobs, the available GPU accelerator labels need to be selected during the creation of the workload.

Selecting GPUs for a workload

The workload configuration displays a dropdown menu for the possible GPU accelerator labels, available on the Kubernetes cluster.

Figure 1: Selecting GPUs for workload when starting a new session

Start A New Session

Session Name

Runtime

Editor ⓘ

Kernel ⓘ

Edition ⓘ

Version
2024.02

Configure additional runtime options in [Project Settings](#).

☐ Enable Spark ⓘ

Runtime Image
- docker-private.infra.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.9-standard:2024.02.1-b4

Resource Profile

vCPU / GiB Memory

GPU Type

Tesla-V100-PCIE-32GB (Default)

Tesla-P100-PCIE-12GB

Number of GPUs

Figure 2: Selecting GPUs for workload when starting a new job

doctest / ales-test / Jobs / New Job

Editor ⓘ

Please select one

Kernel ⓘ

Please select one

Edition ⓘ

Please select one

Version

Please select one

Configure additional runtime options in [Project Settings](#).

Enable Spark ⓘ

✓

Spark 2.4.7 - CDP 7.1.7.2035

Runtime Image

Schedule

Manual

Resource Profile

vCPU / GiB Memory

2 vCPU / 4 GiB Memory

GPU Type

Tesla-T4 (Default)

Number of GPUs

0 GPU

Timeout In Minutes (optional)

30

☐ Kill on timeout

Jobs exceeding timeout send warning email if notifications enabled.

Environment Variables

Name	Value	Actions	Hide/Show Value
		<div>Add</div>	<div></div>

Environment variables will override the [project environment](#).

0 GPU

1 GPU

2 GPUs

If you are unsure about which GPU accelerator label to use, you can use the default value set by the Cloudera Machine Learning Workspace administrator.



Note:

The number of GPUs you can provision for a single workload, is limited by what is available on a single node.

APIv2 GPU workloads

APIv2 GPU workloads can also target a specific GPU.

The following code example presents details on identifying the GPU accelerator label:

```
# Instantiate the cml api client
import cmlapi
client = cmlapi.default_client()
# List the available gpu accelerator labels
client.list_all_accelerator_node_labels()

# Use the gpu accelerator label in the request body of the workload
job_body = cmlapi.CreateJobRequest(
    project_id="some_project_id",
    name="job_with_gpu_accelerator",
    script="some_job_script.py",
    kernel="some_runtime_kernel",
    runtime_identifier="some_runtime",
    nvidia_gpu = 1,
    accelerator_label_id = 2
)

apiv2client.create_job(
    project_id="some_project_id", body=job_body)
```

Existing APIv2 calls which do not specify a GPU accelerator ID can be launched using the default GPU accelerator, set by the Cloudera Machine Learning Administrator.

A Cloudera Machine Learning workload is backed by a Kubernetes Pod, which can only reserve as many resources as there are available on a single node.

Related Information

[CML API v2](#)

Testing GPU Setup

Use these code samples to test that your GPU setup works with several common deep learning libraries. The specific versions of libraries depend on the particular GPU used and the GPU driver version. You can use this testing for GPU setup using Legacy Engines.

1. Go to a project that is using the CUDA engine and click Open Workbench.
2. Launch a new session with GPUs.
3. Run the following command in the workbench command prompt to verify that the driver was installed correctly:

```
! /usr/bin/nvidia-smi
```

4. Use any of the following code samples to confirm that the new engine works with common deep learning libraries.

PyTorch

```
!pip3 install torch==1.4.0
from torch import cuda
assert cuda.is_available()
assert cuda.device_count() > 0
print(cuda.get_device_name(cuda.current_device()))
```



Note: The PyTorch installation requires at least 4 GB of memory.

Tensorflow

```
!pip3 install tensorflow-gpu==2.1.0
from tensorflow.python.client import device_lib
assert 'GPU' in str(device_lib.list_local_devices())
device_lib.list_local_devices()
```

Keras

```
!pip3 install keras
from keras import backend
assert len(backend.tensorflow_backend._get_available_gpus()) > 0
print(backend.tensorflow_backend._get_available_gpus())
```