ModelArts

Troubleshooting

Issue	01
Date	2024-04-30

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 General Issues

1.1 Incorrect OBS Path on ModelArts

Symptom

- When an OBS bucket path is used in ModelArts, a message indicating that the created OBS bucket cannot be found or message "ModelArts.2791: Invalid OBS path" is reported.
- "Error: stat:403" is reported when you perform operations on an OBS bucket.
- "Permission denied" is reported when a file is downloaded from OBS to Notebook.

Possible Causes

- The OBS bucket and ModelArts are in different regions.
- You do not have access to OBS buckets of other users.
- Access authorization has not been configured on ModelArts.
- Encrypted files are to upload to OBS. ModelArts does not support encrypted OBS files.
- The permissions and access control lists (ACLs) of the OBS bucket are incorrectly configured.
- When a training job is created, the code directory and boot file are configured incorrectly.

Solution

Check whether the OBS bucket and ModelArts are in the same region.

1. Check the region where the created OBS bucket is located.
   a. Log in to **OBS management console**.
   b. On the **Buckets** page, enter the name of created OBS bucket in the search box or locate the bucket in the **Bucket Name** column.
      In the **Region** column, view the region where the created OBS bucket is located.

2. Check the region where ModelArts is deployed.

   Log in to the ModelArts management console and view the region where ModelArts is located in the upper left corner.

3. Check whether the region of the created OBS bucket is the same as that of ModelArts. Ensure that they are the same.

Check whether you have the permission to access the OBS bucket.

Check whether you have the permission to access OBS buckets of other users from a notebook instance.

Check delegation authorization.

Go to the **Global Configuration** page and check whether you have the OBS access authorization. If you do not, see **Configuring Access Authorization (Global Configuration)**.

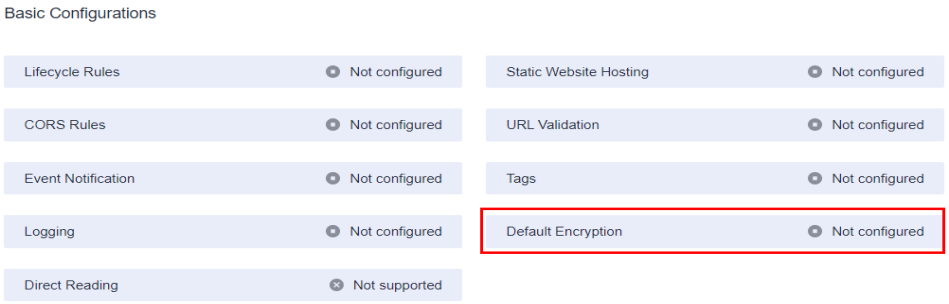
Check whether the OBS bucket is encrypted.

1. Log in to the OBS management console and click the bucket name to go to the **Overview** page.
2. Ensure that default encryption is disabled for the OBS bucket. If the OBS bucket is encrypted, click **Default Encryption** and disable it.

   NOTE

   When you create an OBS bucket, do not select **Archive** or **Deep Archive**. Otherwise, training models will fail.

Figure 1-1 Bucket encryption status



Check whether the OBS file is encrypted.

1. Log in to the OBS management console and click the bucket name to go to the **Overview** page.
2. In the navigation pane on the left, choose **Objects**. The object list is displayed. Click the name of the object that stores files and find the target file. In the **Encrypted** column of the file list, check whether the file is encrypted. File encryption cannot be canceled. In this case, cancel bucket encryption and upload images or files again.

Check the ACLs of the OBS bucket.

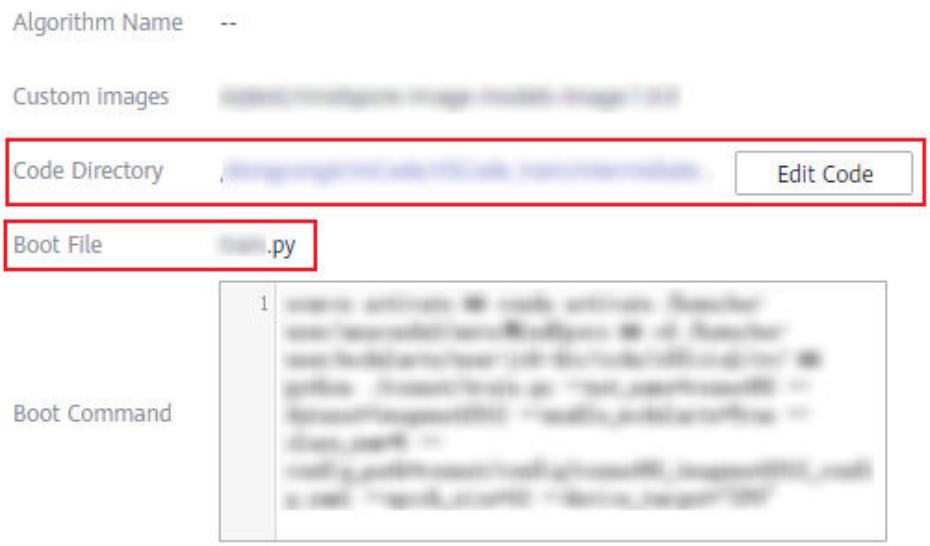
1. Log in to the OBS management console and click the bucket name to go to the **Overview** page.

2. In the navigation pane, choose **Permissions** and click **Bucket ACLs**. Then, check whether the current account has the read and write permissions. If it does not, contact the bucket owner to obtain the permissions.
3. In the navigation pane on the left, choose **Permissions** > **Bucket Policy**, and check whether the current OBS bucket can be accessed by IAM users.

Check the code directory and boot file of a training job.

1. Log in to the ModelArts management console, choose **Training Management** > **Training Jobs**, locate the failed training job, and click its name or ID to go to the job details page.
2. In the pane on the left, check whether the code directory and startup file are correct, and ensure that the OBS file name does not contain spaces.
   - Select an OBS directory for code directory. If a file is selected, the system will display a message indicating an invalid OBS path.
   - The boot file must be in the .py format. Otherwise, the system will display a message indicating an invalid OBS path.

Figure 1-2 Code Directory and Boot File of a training job



If the fault persists, see **Why Can't I Access OBS (403 AccessDenied) After Being Granted with the OBS Access Permission?** for further troubleshooting.

2 ExeML

2.1 Preparing Data

2.1.1 Failed to Publish a Dataset Version

If this fault occurs, the data does not meet the requirements of the data management module. As a result, the dataset fails to be published and the following operations cannot be performed.

Check your data, exclude the data that does not meet the following requirements, and restart the ExeML training task.

ModelArts.4710 OBS Permission Issues

This fault is caused by OBS permissions when ModelArts interacts with OBS. If the message "OBS service Error Message" is displayed, the fault is caused by OBS permissions. Perform the following steps to rectify the fault. If this information is not contained in the error message, the fault is caused by backend services. **Contact Huawei Cloud technical support**.

1. Check whether the current account has OBS permissions.

   Perform this step if you log in to ModelArts as an IAM user.

   Grant the current IAM user with the **Tenant Administrator** permission on global services so that the user has all OBS operation permissions. For details, see **OBS Permissions Management**.

   To restrict the IAM user account' permissions, configure the minimum OBS operation permissions for it. For details, see **Creating a Custom Policy**.

2. Check whether the user has OBS bucket permissions.

   NOTE

   The OBS bucket described in the following steps is specified when you create an ExeML project or the one where the dataset selected during project creation is stored.

   - Check whether the current account has been granted with the read and write permissions on the OBS bucket (specified in bucket ACLs).

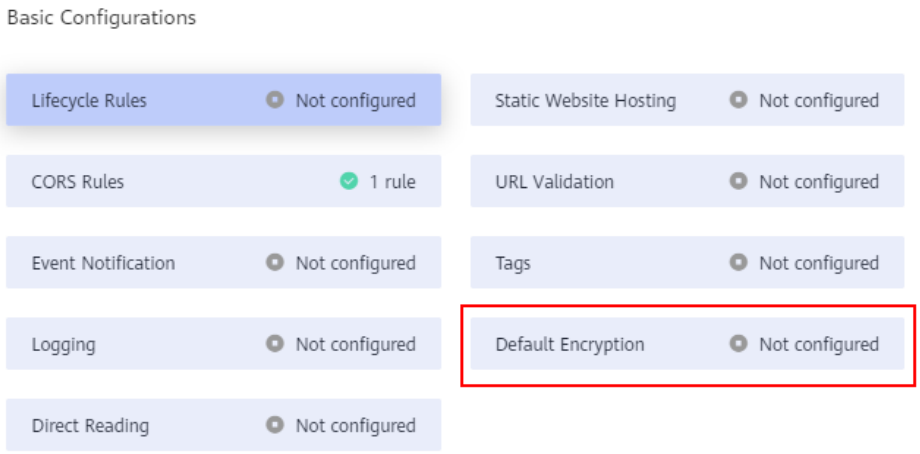
- Go to the OBS management console, select the OBS bucket used by the ExeML project, and click the bucket name to go to the **Overview** page.
- In the navigation pane, choose **Permissions** > **Bucket ACLs**. On the **Bucket ACLs** page that is displayed, check whether the current account has the read and write permissions. If it does not, contact the bucket owner to grant the permissions.

– Check whether the OBS bucket is unencrypted.

i. Go to the OBS management console, select the OBS bucket used by the ExeML project, and click the bucket name to go to the **Overview** page.

ii. Ensure that the default encryption function is disabled for the OBS bucket. If the OBS bucket is encrypted, click **Default Encryption** and change its encryption status.

Figure 2-1 Checking whether the default encryption function is enabled for the OBS bucket

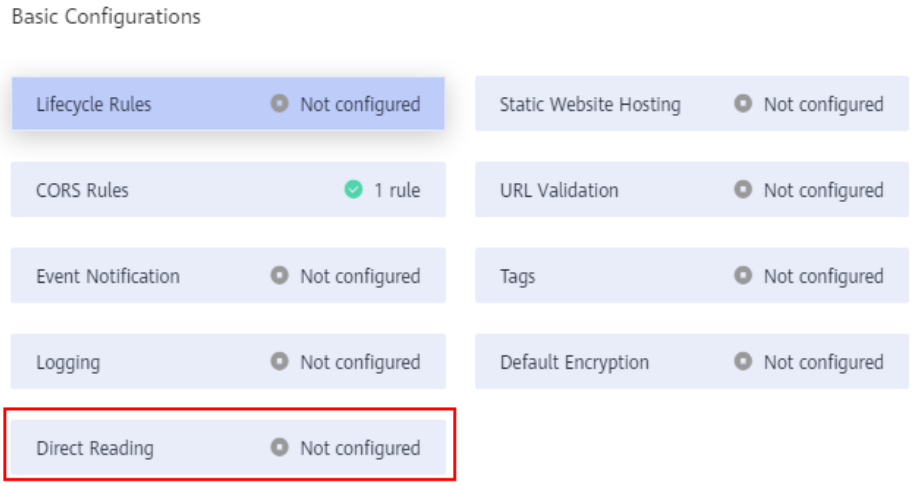


– Check whether the direct reading function of archived data is disabled.

i. Go to the OBS management console, select the OBS bucket used by the ExeML project, and click the bucket name to go to the **Overview** page.

ii. Ensure that the direct reading function is disabled for the archived data in the OBS bucket. If this function is enabled, click **Direct Reading** and disable it.

Figure 2-2 Disabling the direct reading function



ModelArts.4711 Number of Labeled Samples in the Dataset Does Not Meet Algorithm Requirements

Each labeling type must contain at least five images.

ModelArts.4342 Labeling Information Does Not Meet Splitting Conditions

If this fault occurs, modify the labeling data based on the following suggestions and try again.

- At least two multi-label samples (that is, an image contains multiple labels) are required. If you enable dataset splitting when starting training and the number of images with multiple labels is less than 2, the dataset splitting fails. Check your labeling information and ensure that more than two images with multiple labels are labeled.
- After the dataset is split, the label classes contained in the training set and validation set are different. Cause: In the multi-label scenario, after random data segmentation, samples containing a certain type of labels are classified into the training set. As a result, the verification set does not contain the label samples. This issue rarely occurs. You can try to release a new version to handle the issue.

ModelArts.4371 Dataset Version Already Exists

If this error code is displayed, the dataset version already exists. In this case, republish the dataset version.

ModelArts.4712 Datasets Are Being Imported or Synchronized

If the dataset used in ExeML is being imported or synchronized, this error occurs during training. In this case, start the ExeML training task after other tasks are complete.

2.1.2 Invalid Dataset Version

If this issue occurs, the dataset version is successfully released but does not meet the requirements of the ExeML training jobs. As a result, an error message is displayed, indicating that the dataset version does not meet the requirements.

Labeling Information Does Not Meet the Trainning Requirements

For different types of ExeML projects, training jobs have the following requirements on datasets:

- Image classification: There are at least two classes (that is, at least two labels) for the images to be trained, and the number of images in each class cannot be less than 5.
- Object detection: There is at least one class (that is, at least one label) for the images to be trained, and the number of images for each class cannot be less than 5.
- Predictive analytics: The dataset of the predictive analytics task is not managed in a unified manner. Even if the data does not meet the requirements, no fault information is displayed in this issue.
- Sound classification: There are at least two classes (that is, at least two labels) for the audio files to be trained, and the number of audio files in each class cannot be less than 5.
- Text classification: There are at least two classes (that is, at least two labels) for the text files to be trained, and the number of text files in each class cannot be less than 20.

2.2 Training a Model

2.2.1 Failed to Create an ExeML-powered Training Job

This fault is typically caused by a backend service failure. Recreate the training job later. If the fault persists after three retries, contact **HUAWEI CLOUD technical support**.

2.2.2 ExeML-powered Training Job Failed

A training job that is successfully created fails to be executed due to some faults.

To rectify this fault, check whether your account is in arrears first. If your account is normal, rectify the fault based on the job type.

- For details about how to rectify the job training faults related to **Image Classification**, **Sound Classification**, and **Text Classification**, see **Checking Whether Data Exists in OBS**, **Checking the OBS Access Permission**, and **Checking Whether the Images Meet the Requirements**.
- For details about how to rectify the job training faults related to **Object Detection**, see **Checking Whether Data Exists in OBS**, **Checking the OBS Access Permission**, **Checking Whether the Images Meet the Requirements**, and **Checking Whether the Marking Boxes Meet the Object Detection Requirements**.

- For details about how to rectify the job training faults related to **Predictive Analytics**, see **Checking Whether Data Exists in OBS**, **Checking the OBS Access Permission**, and **Troubleshooting of a Predictive Analytics Job Failure**.

Checking Whether Data Exists in OBS

If the images or data stored in OBS is deleted and not synchronized to ModelArts ExeML or datasets, the task will fail.

Check whether data exists in OBS. For Image Classification, Sound Classification, Text Classification, and Object Detection, you can click **Synchronize Data Source** on the **Data Labeling** page of ExeML to synchronize data from OBS to ModelArts.

Checking the OBS Access Permission

If the access permission of the OBS bucket cannot meet the training requirements, the training fails. Do the following to check the OBS permissions:

- Check whether the current account has been granted with the read and write permissions on the OBS bucket (specified in bucket ACLs).
  a. Go to the OBS management console, select the OBS bucket used by the ExeML project, and click the bucket name to go to the **Overview** page.
  b. In the navigation pane, choose **Permissions** and click **Bucket ACLs**. Then, check whether the current account has the read and write permissions. If it does not, contact the bucket owner to obtain the permissions.
- Check whether the OBS bucket is unencrypted.
  a. Go to the OBS management console, select the OBS bucket used by the ExeML project, and click the bucket name to go to the **Overview** page.
  b. Ensure that the default encryption function is disabled for the OBS bucket. If the OBS bucket is encrypted, click **Default Encryption** and change its encryption status.

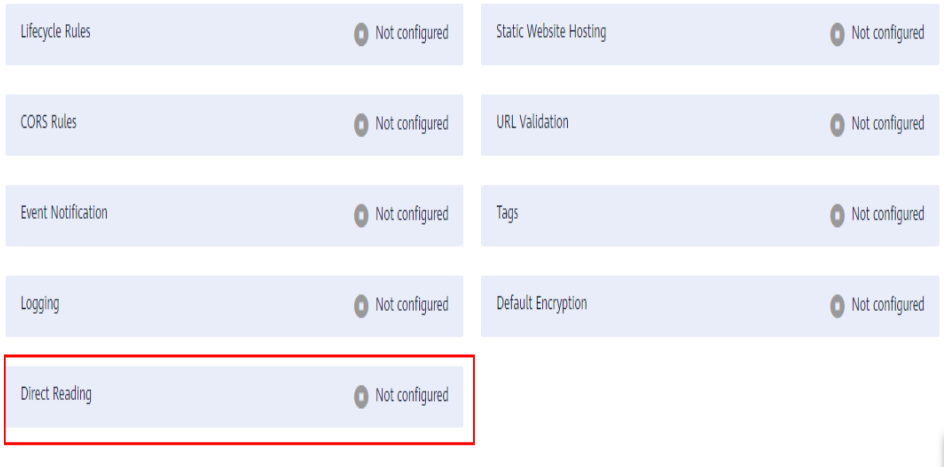
  Figure 2-3 Default encryption status

- Check whether the direct reading function of archived data is disabled.

a. Go to the OBS management console, select the OBS bucket used by the ExeML project, and click the bucket name to go to the **Overview** page.

b. Ensure that the direct reading function is disabled for the archived data in the OBS bucket. If this function is enabled, click **Direct Reading** and disable it.

Figure 2-4 Disabled direct reading



- Ensure that files in OBS are not encrypted.

  Do not select KMS encryption when uploading images or files. Otherwise, the dataset fails to read data. File encryption cannot be canceled. In this case, cancel bucket encryption and upload images or files again.

Figure 2-5 File encryption status

Checking Whether the Images Meet the Requirements

Currently, ExeML does not support four-channel images. Check your data and exclude or delete this format of images.

Checking Whether the Marking Boxes Meet the Object Detection Requirements

Currently, object detection supports only rectangular labeling boxes. Ensure that the labeling boxes of all images are rectangular ones.

If a non-rectangle labeling box is used, the following error message may be displayed:

```
Error bandbox.
```

For other types of projects (such as image classification and sound classification), skip this checking item.

Troubleshooting of a Predictive Analytics Job Failure

1. Check whether the data used for predictive analytics meets the following requirements.

   The predictive analytics task releases datasets without using the data management function. If the data does not meet the requirements of the training job, the job will fail to run.

   Check whether the data used for training meets the requirements of the predictive analytics job. The following lists the requirements. If the requirements are met, go to the next step. If the requirements are not met, adjust the data based on the requirements and then perform the training again.

   - The name of files in a dataset consists of letters, digits, hyphens (-), and underscores (_), and the file name suffix is **.csv**. The files cannot be stored in the root directory of an OBS bucket, but in a folder in the OBS bucket, for example, **/obs-xxx/data/input.csv**.
   - The files are saved in **CSV** format. Use newline characters (**\n** or LF) to separate lines and commas (,) to separate columns of the file content. The file content cannot contain Chinese characters. The column content cannot contain special characters such as commas (,) and newline characters. The quotation marks are not supported. It is recommended that the column content consist of letters and digits.
   - The number of training columns is the same. There are at least 100 different data records (a feature with different values is considered as different data) in total. The training columns cannot contain data of the timestamp format (such as *yy-mm-dd* or *yyyy-mm-dd*). Ensure that there are at least two values in the specified label column and no data is missing. In addition to the label column, the dataset must contain at least two valid feature columns. Ensure that there are at least two values in each feature column and that the percentage of missing data must be lower than 10%. The training data CSV file cannot contain the table header. Otherwise, the training fails. Due to the limitation of the feature filtering algorithm, place the label column in the last column of the dataset. Otherwise, the training may fail.

2. ModelArts automatically filters data and then starts the training job. If the preprocessed data does not meet the training requirements, the training job fails to be executed.

   Filter policies for columns in a dataset:

   - If the vacancy rate of a column is greater than the threshold (0.9) set by the system, the data in this column will be deleted during training.
   - If a column has only one value (that is, the data in each row is the same), the data in this column will be deleted during training.
   - For a non-numeric column, if the number of values in this column is equal to the number of rows (that is, the values in each row are different), the data in this column will be deleted during training.

   After the preceding filtering, if the data in the dataset does not meet the training requirements in Item 1, the training fails or cannot be executed. Complete the data before starting the training.

3. Restrictions for a dataset file:

a. If you use the 2U8G flavor (2 vCPUs and 8 GB of memory), it is recommended that the size of the dataset file be less than 10 MB. If the file size meets the requirements but the data volume (product of the number of rows and the number of columns) is extremely large, the training may still fail. It is recommended that the product be less than 10,000.

   If you use the 8U32G flavor (8 vCPUs and 32 GB of memory), it is recommended that the size of the dataset file be less than 100 MB. If the file size meets the requirements but the data volume (product of the number of rows and the number of columns) is extremely large, the training may still fail. It is recommended that the product be less than 1,000,000.

4. If the fault persists, contact **HUAWEI CLOUD technical support**.

2.2.3 Failed to Train a Model and Error KMS.0314 Occurred

Symptom

When a model is trained in an ExeML project, a message is displayed, indicating that the training failed.

NOTE

This issue applies only to users not of the Huawei Cloud Chinese Mainland website.

Possible Causes

This issue is caused by real-name authentication. When users not of the Huawei Cloud Chinese Mainland website attempt to purchase or use services of the Huawei Cloud Chinese Mainland website, they must be real-name authenticated. If they are not real-name authenticated, this issue occurs.

Solution

To ensure that your ExeML-based AI development project can be properly carried out, **complete real-name authentication** before performing other operations such as model training.

2.3 Deploying a Model

2.3.1 Failed to Submit the Real-time Service Deployment Task

This fault is typically caused by the limited quota of the account.

In an ExeML project, after the deployment is started, the model is automatically deployed as a real-time service. If the number of real-time services exceeds the quota limit, the model cannot be deployed as a service. In this case, an error message is displayed in the ExeML project, indicating that the real-time service deployment task fails to be submitted.

Troubleshooting

- Method 1: Choose **Service Deployment** > **Real-time Services**. On the displayed page, delete services that are no longer used to release resources.
- Method 2: If the deployed real-time service still needs to be used, you are advised to apply for a higher quota.

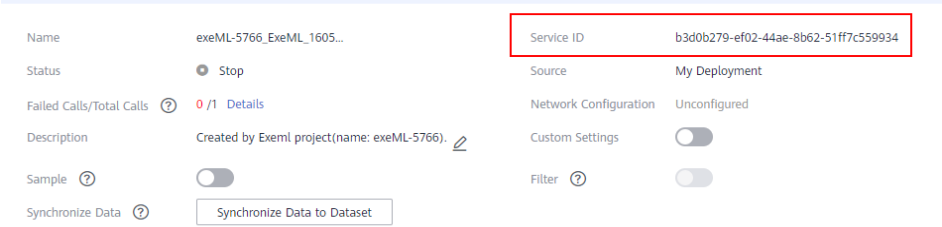
2.3.2 Failed to Deploy a Real-time Service

This fault is typically caused by a backend service failure. You are advised to redeploy the real-time service later. If the fault persists after three retries, obtain the following information and contact **HUAWEI CLOUD technical support**.

- Obtain a service ID.

  Go to the **Service Deployment > Real-Time Services** page. In the service list, find the real-time service deployed in the ExeML task. All the services of ExeML start with **exeML-** Click the service name to go to the service details page. In the basic information area, obtain **Service ID**.

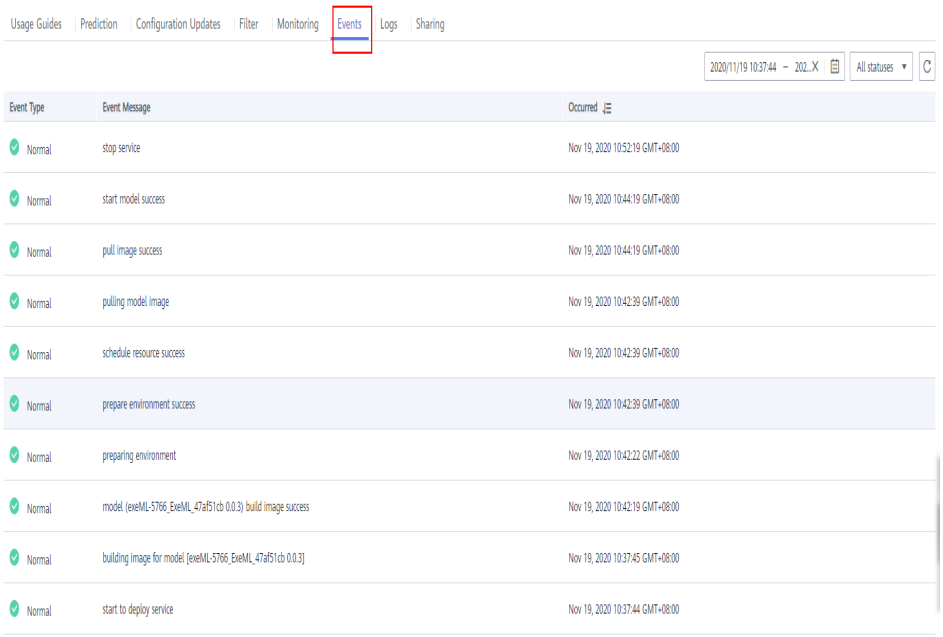
  Figure 2-6 Obtaining a service ID

- Obtain events about the real-time service.

  On the service details page, click the **Events** tab. Take a screenshot of the event information table, and send the screenshot to technical support personnel.

Figure 2-7 Obtaining events



2.4 Publishing a Model

2.4.1 Failed to Submit the Model Publishing Task

This fault is typically caused by a backend service failure. You are advised to recreate the training job later. If the fault persists after three retries, contact **HUAWEI CLOUD technical support**.

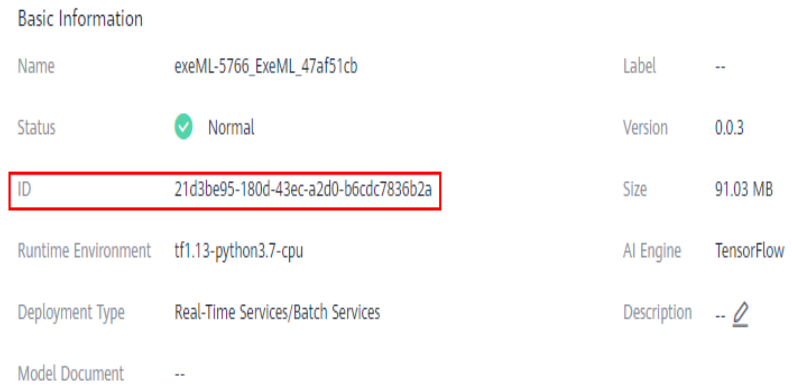
2.4.2 Failed to Publish a Model

This fault is typically caused by a backend service failure. You are advised to recreate the training job later. If the fault persists after three retries, obtain the following information and contact **HUAWEI CLOUD technical support**.

- Obtain a model ID.

  Choose **AI Application Management** > **AI Applications**. In the AI application list, find the applications automatically created in the ExeML task. All the AI applications generated by ExeML start with **exeML-**. Click the model name to go to the model details page. In the **Basic Information** area, obtain the value of **ID**.

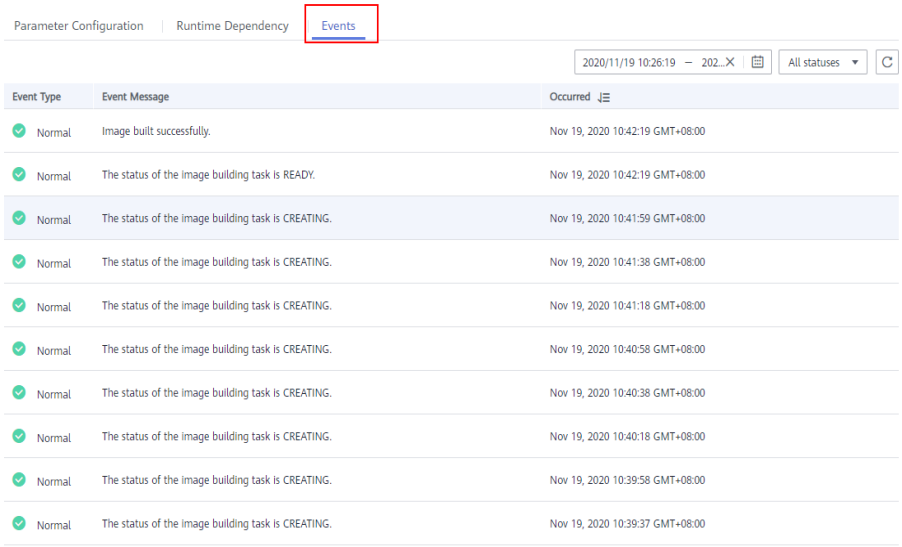
Figure 2-8 Obtaining a model ID



- Obtain model events.

  On the model details page, click the **Events** tab. Take a screenshot of the event information table, and send the screenshot to technical support personnel.

Figure 2-9 Obtaining events



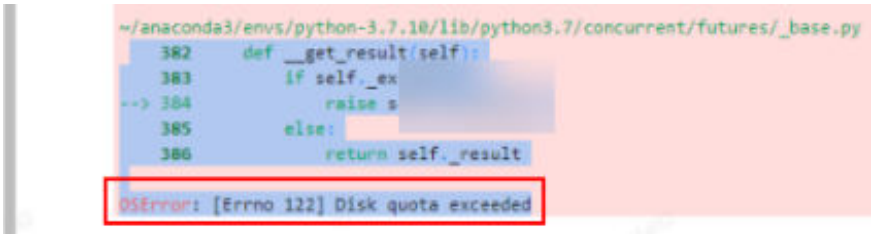
3 DevEnviron

3.1 Environment Configuration Faults

3.1.1 Disk Space Used Up

Symptom

- Error message "No Space left on Device" is displayed when a notebook instance is used.
- Error message "Disk quota exceeded" is displayed when code is executed in a notebook instance.



Possible Causes

- After a file is deleted from the navigation pane on the left of JupyterLab, the file is moved to the recycle bin by default. This occupies memory, leading to insufficient disk space.
- The disk quota is insufficient.

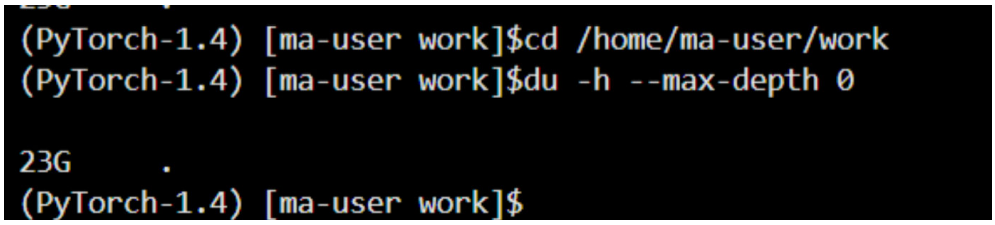
Solution

Check the storage space used by the VM, check the memory used by files in the recycle bin, and delete unnecessary large files from the recycle bin.

1. On the notebook instance details page, view the storage capacity of the instance.

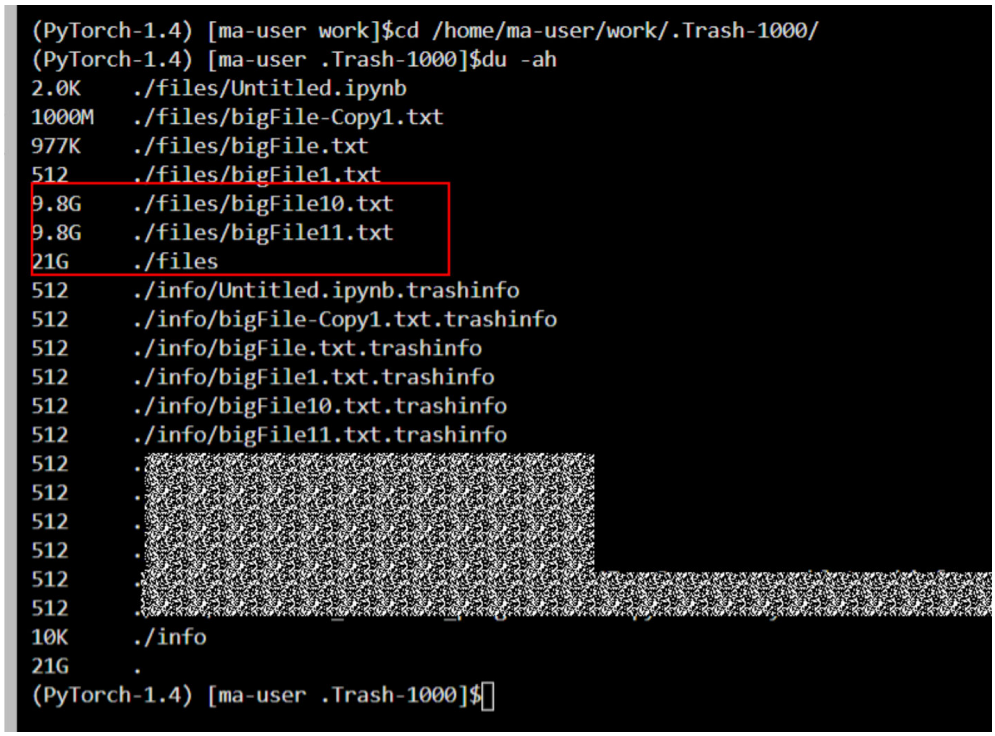
2. Check the storage space used by the VM. The storage space is typically close to the storage capacity.

```
cd /home/ma-user/work
du -h --max-depth 0
```



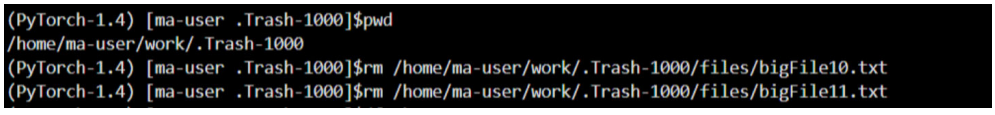
3. Run the following commands to check the memory used by the recycle bin (recycle bin files are stored in **/home/ma-user/work/.Trash-1000/files** by default):

```
cd /home/ma-user/work/.Trash-1000/
du -ah
```



4. Delete unnecessary large files from the recycle bin. Deleted files cannot be restored.

```
rm {File path}
```



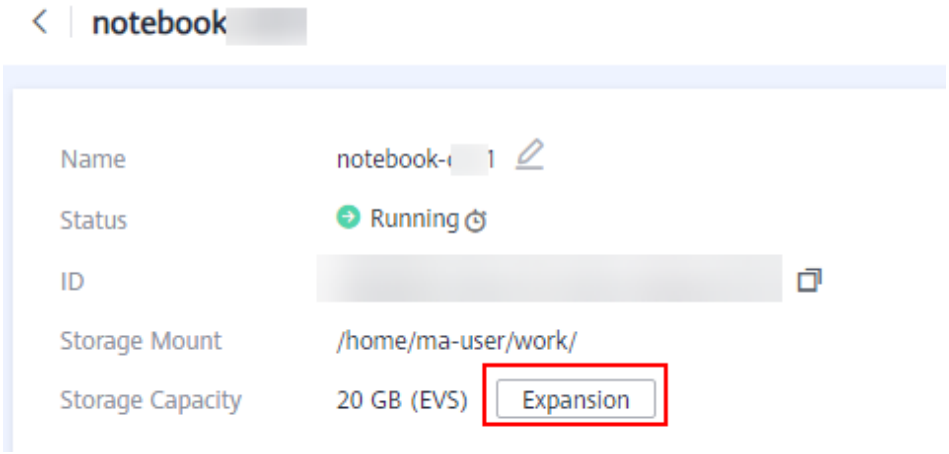
NOTE

If the name of the folder or file you want to delete contains spaces, add single quotation marks to the name.

5. Run the following commands to check the storage space used by the VM again:

```
cd /home/ma-user/work
du -h --max-depth 0
```

6. If the notebook instance uses an EVS disk for storage, expand the storage capacity on the notebook instance details page.



Summary and Suggestions

It is a good practice to delete unnecessary files when using a notebook instance to prevent a training failure caused by insufficient disk space.

3.1.2 An Error Is Reported When Conda Is Used to Install Keras 2.3.1 in Notebook

Symptom

An error is reported when Conda is used to install Keras 2.3.1.


```
conda install keras=2.3.1

/home/ma-user/anaconda3/lib/python3.7/site-packages/requests/__init__.py:91: RequestsDependencyWarning: urllib3 (1.26.12)
  RequestsDependencyWarning)
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: -
Found conflicts! Looking for incompatible packages.
This can take several minutes.  Press CTRL-C to abort.
                                                                                failed

# >>>>>>>>>>>>>>>>>>>>>> ERROR REPORT <<<<<<<<<<<<<<<<<<<<<<

    Traceback (most recent call last):
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/cli/install.py", line 265, in install
        should_retry_solve=(_should_retry_unfrozen or repodata_fn != repodata_fns[-1]),
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/core/solve.py", line 117, in solve_for_transaction
        should_retry_solve)
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/core/solve.py", line 158, in solve_for_diff
        force_remove, should_retry_solve)
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/core/solve.py", line 275, in solve_final_state
        ssc = self._add_specs(ssc)
      File "/home/ma-user/anaconda3/lib/python3.7/site-packages/conda/core/solve.py", line 696, in _add_specs
        raise UnsatisfiableError({})
    conda.exceptions.UnsatisfiableError:
    Did not find conflicting dependencies. If you would like to know which
    packages conflict ensure that you have enabled unsatisfiable hints.

    conda config --set unsatisfiable_hints True
```

Possible Cause

There are network issues with Conda. Run the **pip install** command to install Keras 2.3.1.

Solution

Run the **!pip install keras==2.3.1** command to install Keras.

```
!pip install keras==2.3.1

  Looking in indexes: http://repo.myhuaweicloud.com/repository/pypi/simple
  Collecting keras==2.3.1
    Using cached http://repo.myhuaweicloud.com/repository/pypi/packages/ad/fd/6bfe87920d7f4fd475acd28500a42482b6b84479832bdc0fe9e589a60ceb/Keras-2.3.1-py2.py3-none-any.whl (377 kB
  Requirement already satisfied: keras-applications>=1.0.6 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.0.8)
  Requirement already satisfied: keras-preprocessing>=1.0.5 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.1.2)
  Requirement already satisfied: numpy>=1.9.1 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.21.5)
  Requirement already satisfied: pyyaml in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (5.1)
  Requirement already satisfied: scipy>=0.14 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.7.3)
  Requirement already satisfied: h5py in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (2.8.0)
  Requirement already satisfied: six>=1.9.0 in /home/ma-user/anaconda3/envs/TensorFlow-1.13-gpu/lib/python3.7/site-packages (from keras==2.3.1) (1.16.0)
  Installing collected packages: keras
    Attempting uninstall: keras
      Found existing installation: Keras 2.2.4
      Uninstalling Keras-2.2.4:
        Successfully uninstalled Keras-2.2.4
  Successfully installed keras-2.3.1
```

3.1.3 Error "HTTP error 404 while getting xxx" Is Reported During Dependency Installation in a Notebook

Symptom

An error is reported during dependency installation in a notebook instance. The following shows the error.

```
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /home/ma-user/anaconda3/envs/llama2/lib/python3.10/site-packages (from aiohttp->datasets) (3
.1.0)
Collecting multidict<7.0,>=4.5 (from aiohttp->datasets)
  Using cached http://repo.myhuaweicloud.com/repository/pypi/packages/df/93/34efbfa7aa778b04b365960f52f7071d7942ce386572aac8940ae032dd48/multidict-6.0.2-cp
310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (114 kB)
Collecting async-timeout<5.0,>=4.0.0a3 (from aiohttp->datasets)
  ERROR: HTTP error 404 while getting http://repo.myhuaweicloud.com/repository/pypi/packages/a7/fa/e01228c2938de91d47b307831c62ab9e4001e747789d0b05baf779a6
488c/async_timeout-4.0.3-py3-none-any.whl.metadata
ERROR: 404 Client Error: Not Found for url: http://repo.myhuaweicloud.com/repository/pypi/packages/a7/fa/e01228c2938de91d47b307831c62ab9e4001e747789d0b05ba
f779a6488c/async_timeout-4.0.3-py3-none-any.whl.metadata
(llama2) [ma-user work]$pip install datasets
```

Possible Causes

The dependency is not in the PyPI source or the source is unavailable.

Solution

Run the following command to download the dependency from another source:

pip install -i *Source address Dependency name*

3.1.4 The numba Library Has Been Installed in a Notebook Instance and Error "import numba ModuleNotFoundError: No module named 'numba'" Is Reported

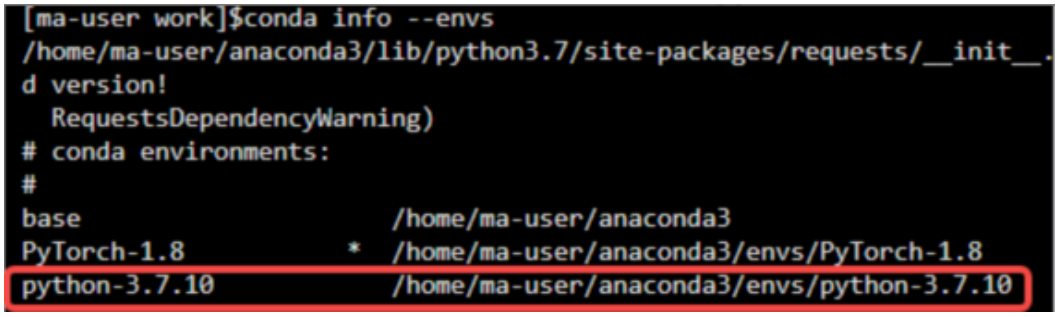
Symptom

After you install the **numba** library in a notebook instance by running the **!pip install numba** command, the library is running properly and is saved as a custom image. However, an error is reported indicating that the library does not exist when you run the script in DataArts Studio.

Possible Causes

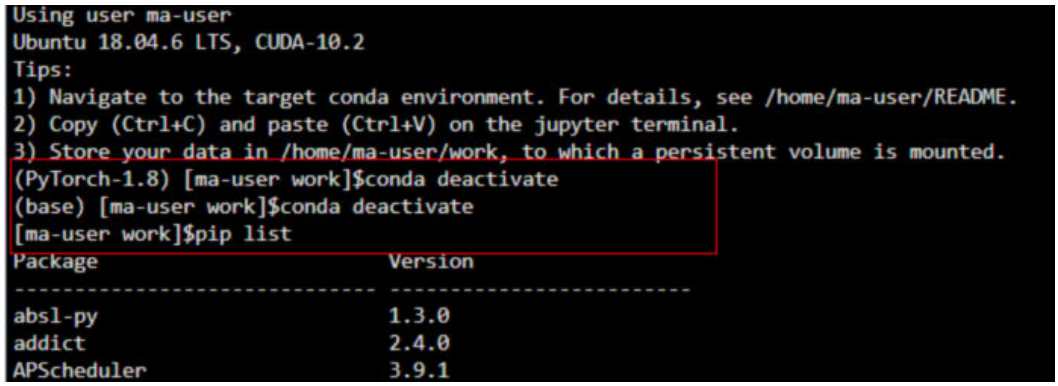
Multiple virtual environments are created and the **numba** library is installed in python-3.7.10, as shown in **Figure 3-1**.

Figure 3-1 Querying virtual environments



Solution

Run the **conda deactivate** command in Termina to exit the current virtual environment and enter the default base environment. Run the **pip list** command to query the installed packages. Install and save the required dependencies, switch to the specified virtual environment, and run the script.



3.2 Instance Faults

3.2.1 Failed to Create a Notebook Instance and JupyterProcessKilled Is Displayed in Events

Symptom

A user failed to create a notebook instance, and **JupyterProcessKilled** was displayed in **Events**.

Possible Causes

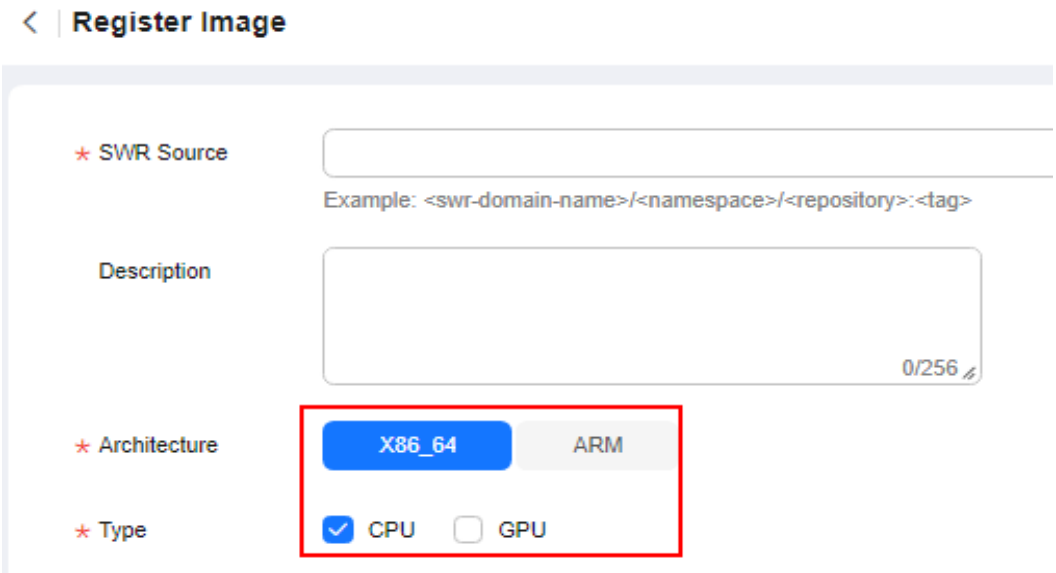
This fault occurs because the Jupyter process is killed. Generally, the notebook instance automatically restarts. If it does not restart, its creation fails. Check whether the failure is caused by the custom image issue.

Solution

Check whether the custom image is correct.

When registering a custom image on the ModelArts console after it is created, ensure that its architecture and type are the same as those of the source image.

Figure 3-2 Registering an image



3.2.2 What Do I Do If I Cannot Access My Notebook Instance?

Troubleshoot the issue based on error code.

A Black Screen Is Displayed When a Notebook Instance Is Opened

A black screen is displayed after a notebook instance is opened, which is caused by a proxy issue. Change the proxy to rectify the fault.

A Blank Page Is Displayed When a Notebook Instance Is Opened

- If a blank page is displayed after a notebook instance is opened, clear the browser cache and open the notebook instance again.
- Check whether the ad filtering component is installed for the browser. If yes, disable the component.

Error 404

If this error is reported when an IAM user creates an instance, the IAM user does not have the permissions to access the corresponding storage location (OBS bucket).

Solution

1. Log in to the OBS console using the primary account and grant access permissions for the OBS bucket to the IAM user. For details about the operation, see **Granting an IAM User the Specified Permissions for a Bucket**.
2. After the IAM user obtains the permissions, log in to the ModelArts console, delete the instance, and use the OBS path to create a notebook instance.

Error 503

If this error is reported, it is possible that the instance is consuming too many resources. If this is the case, stop the instance and restart it.

Error 504

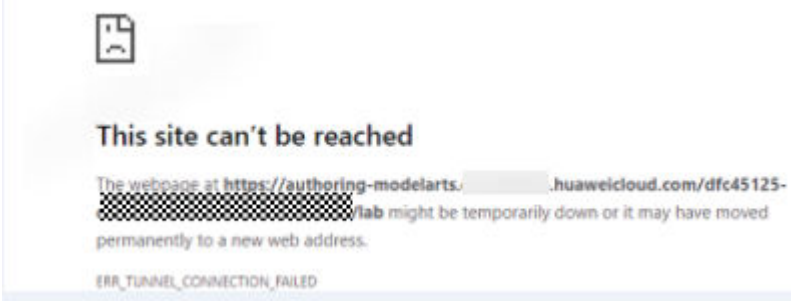
If this error is reported, **submit a service ticket** or contact customer service.

Error 500

Notebook JupyterLab cannot be opened, and error 500 is reported. The possible cause is that the disk space in the **work** directory is used up. In this case, identify the fault cause and clear the disk by referring to **Disk Space Used Up**.

Error "This site can't be reached"

After a notebook instance is created, click **Open** in the **Operation** column. The error message shown in the following figure is displayed.



To solve the problem, copy the domain name of the page, add it to the **Do not use proxy server for addresses beginning with** text box, and save the settings.

3.2.3 What Should I Do When the System Displays an Error Message Indicating that No Space Left After I Run the pip install Command?

Symptom

In the notebook instance, error message "No Space left..." is displayed after the **pip install** command is run.

Solution

You are advised to run the **pip install --no-cache **** command instead of the **pip install **** command. Adding the **--no-cache** parameter can solve such problem.

3.2.4 What Do I Do If the Code Can Be Run But Cannot Be Saved, and the Error Message "save error" Is Displayed?

If the notebook instance can run the code but cannot save it, the error message "save error" is displayed when you save the file. In most cases, this error is caused by a security policy of Web Application Firewall (WAF).

On the current page, some characters in your input or output of the code are intercepted because they are considered to be a security risk. Submit a service ticket and contact customer service to check and handle the problem.

3.2.5 Why Is a Request Timeout Error Reported When I Click the Open Button of a Notebook Instance?

When a notebook container breaks down due to memory overflow or other reasons, if you click the **Open** button of the notebook instance, a request timeout error is displayed.

In this case, wait for about half a minute or so until the container is restored, and then click **Open** again.

3.2.6 ModelArts.6333 Error Occurs

Symptom

When you use a notebook instance, the ModelArts.6333 error is displayed.

Possible Cause

The fault may be caused by instance overload. The notebook instance automatically restores. Refresh the page and wait for several minutes. The common cause is that the memory is used up.

Solution

When this error occurs, the notebook instance automatically restores. You can refresh the page and wait for several minutes.

The common cause is that the memory is used up. You can use the following methods to rectify the fault.

- Method 1: Replace the notebook instance with a resource with higher specifications.
- Method 2: Adjust the parameters in the code to reduce memory occupation. If the memory is still insufficient after the code is modified, use method 1.
  a. Call the sklearn method **silhouette_score(addr_1,siteskmeans.labels)** and specify the **sample_size** parameter to reduce memory occupation.
  b. When calling the **train** method, you can try to decrease the value of **batch_size**.

3.2.7 What Can I Do If a Message Is Displayed Indicating that the Token Does Not Exist or Is Lost When I Open a Notebook Instance?

Symptom

You shared your notebook URL with others, but they receive an error message "... lost token or incorrect token...." when attempting to access the URL.

Possible Cause

They do not have the token of the account.

Solution

Add the token of the notebook owner to the end of the URL.

3.3 Code Running Failures

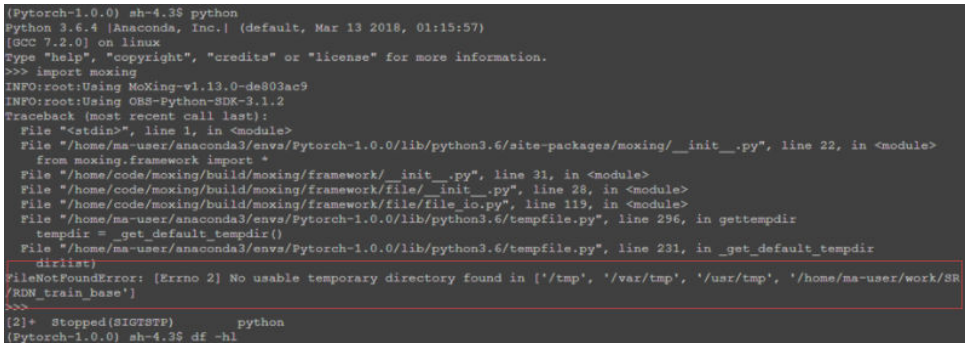
3.3.1 Error Occurs When Using a Notebook Instance to Run Code, Indicating That No File Is Found in /tmp

Symptom

When the a notebook instance is used to run code, the following error occurs:

```
FileNotFoundError: [Error 2] No usable temporary directory found in ['/tmp', '/var/tmp', '/usr/tmp', 'home/ma-user/work/SR/RDN_train_base']
```


Figure 3-3 Code running error



Possible Cause

Check whether a large amount of data is saved in **/tmp**.

Solution

1. Go to the **Terminal** page. In the **/tmp** directory, run the **du -sh *** command to check the space usage of the directory.

   ```
   sh-4.3$cd /tmp
   sh-4.3$du -sh *
   4.0K    core-js-banners
   0       npm-19-41ed4c62
   6.7M    v8-compile-cache-1000
   ```

2. Delete unnecessary large files.

   a. Delete the sample file **test.txt**: **rm -f /home/ma-user/work/data/test.txt**

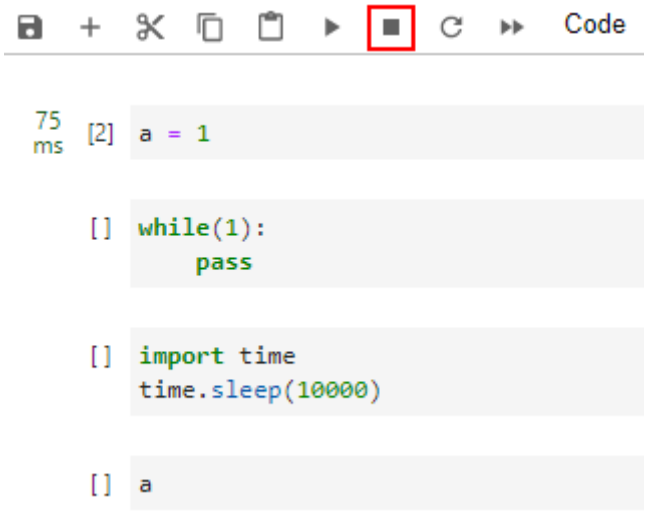
   b. Delete the sample folder **data**: **rm -rf /home/ma-user/work/data/**

3.3.2 What Do I Do If a Notebook Instance Won't Run My Code?

If a notebook instance fails to execute code, you can locate and rectify the fault as follows:

1. If the execution of a cell is suspended or lasts for a long time (for example, the execution of the second and third cells in **Figure 3-4** is suspended or lasts for a long time, causing execution failure of the fourth cell) but the notebook page still responds and other cells can be selected, click **interrupt the kernel** highlighted in a red box in the following figure to stop the execution of all cells. The notebook instance retains all variable spaces.

Figure 3-4 Stopping all cells



2. If the notebook page does not respond, close the notebook page and the ModelArts console. Then, open the ModelArts console and access the notebook instance again. The notebook instance retains all the variable spaces that exist when the notebook instance is unavailable.
3. If the notebook instance still cannot be used, access the **Notebook** page on the ModelArts console and stop the notebook instance. After the notebook instance is stopped, click **Start** to restart the notebook instance and open it. The instance will have preserved all the spaces for the variables that were unable to run.

3.3.3 Why Does the Instance Break Down When dead kernel Is Displayed During Training Code Running?

The notebook instance breaks down during training code running due to insufficient memory caused by large data volume or excessive training layers.

After this error occurs, the system automatically restarts the notebook instance to fix the instance breakdown. In this case, only the breakdown is fixed. If you run the training code again, the failure will still occur. To solve the problem of insufficient memory, you are advised to create a new notebook instance and use a resource pool of higher specifications, such as a dedicated resource pool, to run the training code. An existing notebook instance that has been successfully created cannot be scaled up using resources with higher specifications.

3.3.4 What Do I Do If cudaCheckError Occurs During Training?

Symptom

The following error occurs when the training code is executed in a notebook:

```
cudaCheckError() failed : no kernel image is available for execution on the device
```

Possible Cause

Parameters **arch** and **code** in **setup.py** have not been set to match the GPU compute power.

Solution

For GP Vnt1 GPUs, the GPU computing power is **-gencode arch=compute_70,code=[sm_70,compute_70]**. Set the compilation parameters in **setup.py** accordingly.

3.3.5 What Do I Do If Insufficient Space Is Displayed in DevEnviron?

If space is insufficient, use notebook instances with EVS disks.

Upload the code and data of the affected notebook instance to an OBS bucket. Then, create a notebook instance with EVS disks, and download the data from OBS to the new notebook instance. For details, see **How Do I Upload a File from a Notebook Instance to OBS or Download a File from OBS to a Notebook Instance?**

3.3.6 Why Does the Notebook Instance Break Down When opencv.imshow Is Used?

Symptom

When opencv.imshow is used in a notebook instance, the notebook instance breaks down.

Possible Causes

The cv2.imshow function in OpenCV malfunctions in a client/server environment such as Jupyter. However, Matplotlib does not have this problem.

Solution

Display images by referring to the following example. Note that OpenCV displays BGR images while Matplotlib displays RGB images.

Python:

```
from matplotlib import pyplot as plt
import cv2
img = cv2.imread('Image path')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('my picture')
plt.show()
```

3.3.7 Why Cannot the Path of a Text File Generated in Windows OS Be Found In a Notebook Instance?

Symptom

When a text file generated in Windows is used in a notebook instance, the text content cannot be read and an error message may be displayed indicating that the path cannot be found.

Possible Causes

The notebook instance runs Linux and its line feed format (CRLF) differs from that (LF) in Windows.

Solution

Convert the file format to Linux in your notebook instance.

Shell:

```
dos2unix File name
```

3.3.8 What Do I Do If No Kernel Is Displayed After a Notebook File Is Created?

Symptom

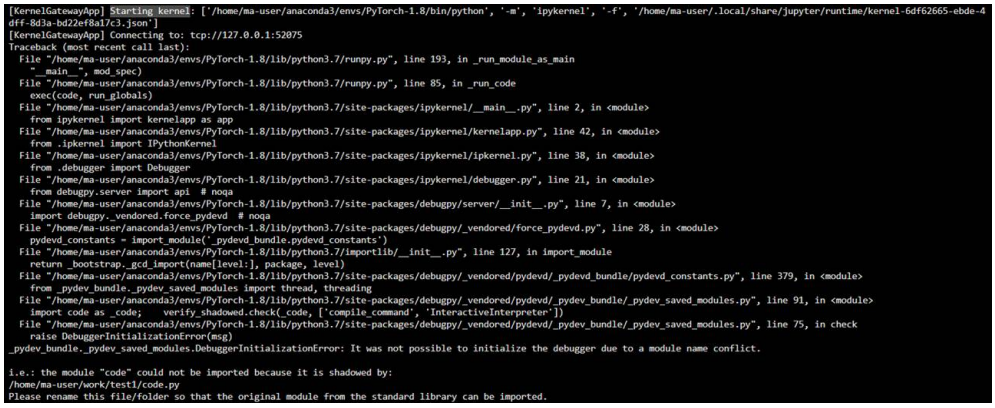
After a notebook file is created, "No Kernel" is displayed in the upper right corner of the page.

Possible Causes

The **code.py** file in the work directory conflicts with the name of the import code file on which the kernel depends.

Solution

1. View the latest log file starting with **kernelgateway** in **/home/ma-user/log/** and search for the logs near **Starting kernel**. If the stack similar to the following is displayed, the possible cause is that the name of the **code.py** file in the work directory conflicts with the name of the import code file on which the kernel depends.



2. To resolve this issue, rename the **code.py** file in the work directory.

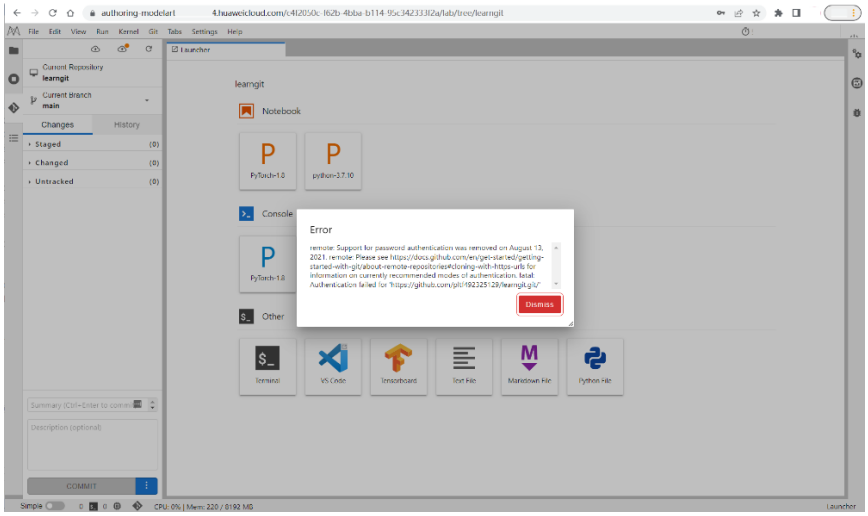
   code.py and **select.py** are typically prone to conflict.

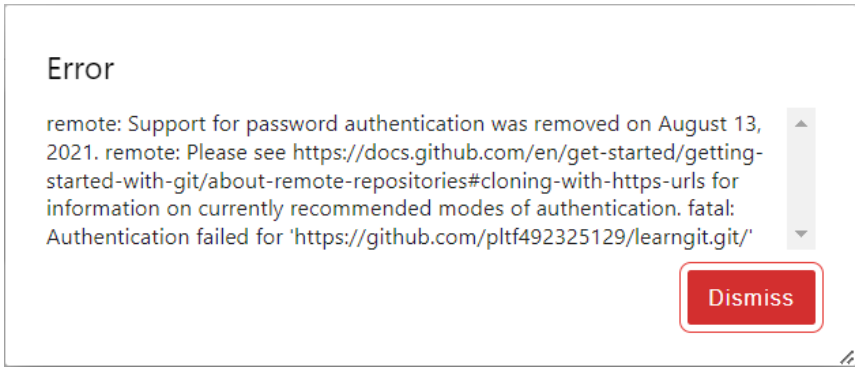
3.4 JupyterLab Plug-in Faults

3.4.1 What Do I Do If the Git Plug-in Password Is Invalid?

Symptom

If the Git plug-in is used in JupyterLab, when a private repository is cloned or a file is pushed, an error occurs.



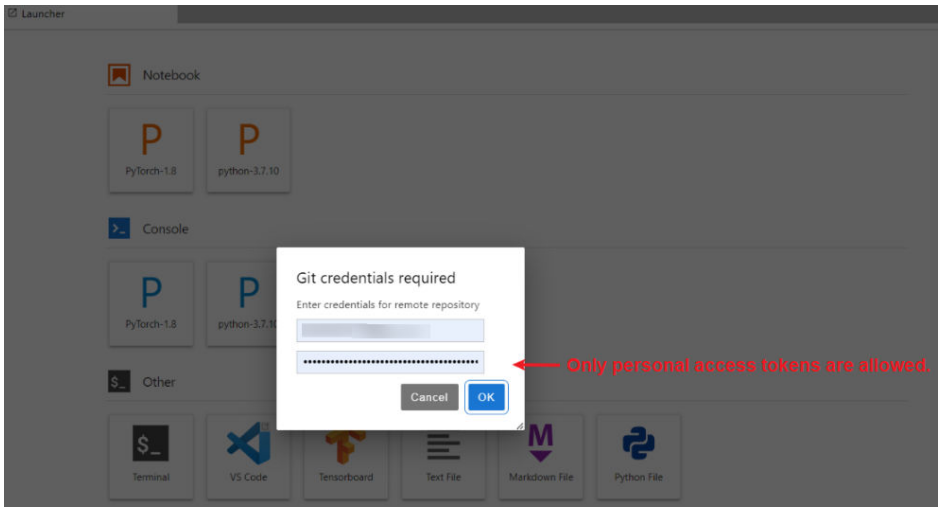


Possible Causes

The authorization using a password has been canceled in GitHub. When cloning a private repository or pushing a file, you are required to enter a token in the authorization text box.

Solution

Use a token for authorization. When cloning a private repository or pushing a file, enter the token in the authorization text box. For details about how to obtain a token, see **Using the Git Plug-in**.



3.5 Save an Image Failures

3.5.1 What If the Error Message "there are processes in 'D' status, please check process status using'ps -aux' and kill all the 'D' status processes" or "Buildimge,False,Error response from daemon,Cannot pause container xxx" Is Displayed When I Save an Image?

Symptom

- When an image is saved in a notebook instance, error "there are processes in 'D' status, please check process status using 'ps -aux' and kill all the 'D' status processes" is displayed.
- When an image is saved in a notebook instance, error "Buildimge,False,Error response from daemon: Cannot pause container xxx" is displayed.

Possible Causes

If there is a process in the **D** state in the notebook instance, saving an image will fail.

Solution

1. Run the **ps -aux** on the terminal to check the process.


```
(PyTorch-1.8) [ma-user work]$ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ma-user        1  0.0  0.0   4532   392 ?        Ss   10:47   0:00 /modelarts/authoring/scrip
ma-user        8  0.0  0.0  22028  2196 ?        S    10:47   0:00 /bin/bash /modelarts/autho
ma-user      103  0.0  0.2 137000 76276 ?        SN   10:47   0:02 /modelarts/authoring/noteb
ma-user      115  0.0  0.0  13444   808 ?        S    10:47   0:00 /bin/bash /modelarts/autho
ma-user      116  0.0  0.0   7940   660 ?        S    10:47   0:00 tee /home/ma-user/log/note
ma-user      119  1.5  0.3 3800480 130936 ?      Sl   10:47   0:47 /modelarts/authoring/noteb
ma-user     3134  0.0  0.0  38536 18876 pts/0    SNs  10:58   0:00 /bin/bash -l
ma-user    11045  0.0  0.0   4388   392 pts/0    DN+  11:37   0:00 ./d_process
ma-user    11046  0.0  0.0   4388   392 pts/0    SN+  11:37   0:00 ./d_process
ma-user    11069  4.2  0.0  22148  2408 pts/1    SNs  11:37   0:00 /bin/bash -l
ma-user    11128  0.0  0.0   7936   656 ?        S    11:37   0:00 sleep 3
ma-user    11131  0.0  0.0  37796  1616 pts/1    RN+  11:37   0:00 ps -aux
(PyTorch-1.8) [ma-user work]$
```

2. Run the **kill -9 <pid>** command to stop the process. Then, save the image again.

3.5.2 What Do I Do If Error "container size %dG is greater than threshold %dG" Is Displayed When I Save an Image?

Symptom

When an image is saved in a notebook instance, error "container size %dG is greater than threshold %dG" is displayed.

Possible Causes

The size of the notebook container exceeded the threshold.

Solution

Reduce the container size. The size of a notebook container consists of the image size and the size of the files newly installed in the container. To resolve this issue, use either of the following methods:

- Reduce the size of the files newly installed in the container.
  a. Delete the files newly installed in a notebook instance. For example, if a large number of files have been downloaded to the notebook instance, delete them. This method applies only to directories other than the **/home/ma-user/work** and **/cache** directories. The persistent storage data in **home/ma-user/work** will not be stored in the created container image, and the temporary files stored in **/cache** do not consume the container storage space.
  b. If no file can be deleted or it is unknown which files can be deleted, use the same image to create a notebook instance. When using the new notebook instance, minimize software package installations or file downloads to reduce the container size.
- Reduce the size of the image file.

  If you are not sure which packages or files do not need to be installed, use a small image to create a notebook instance and install the required software or files in it. Among all the public images, **mindspore1.7.0-py3.7-ubuntu18.04** takes the minimum size.

3.5.3 What Do I Do If Error "too many layers in your image" Is Displayed When I Save an Image?

Symptom

When an image is saved, error "too many layers in your image" is displayed.

Possible Causes

The image selected for creating the target notebook instance is a bring-your-own image or a custom image that has been saved for multiple times. No image can be saved for the notebook instance that is created using such an image.

Solution

Use a public image or another custom image to create a notebook instance and save the image.

3.5.4 What Do I Do If Error "The container size (xG) is greater than the threshold (25G)" Is Reported When I Save an Image?

Symptom

The error **The container size (30G) is greater than the threshold (25G)** is reported when an image is saved, and the image fails to be created.

Possible Causes

To save an image, you need to run the **docker commit** command on the agent of a resource cluster node. Administrative data will be uploaded and updated automatically. Each time you run the command, the image becomes larger. After the image is saved for multiple times, its actual size is larger than it shows. If the image is too large, various problems may occur. You can rebuild the original image environment and save the image to solve the problem.

Solution

Rebuild the original image environment. You can use a base image with minimized installation and run the dependencies. Clear the installation cache and save the image.

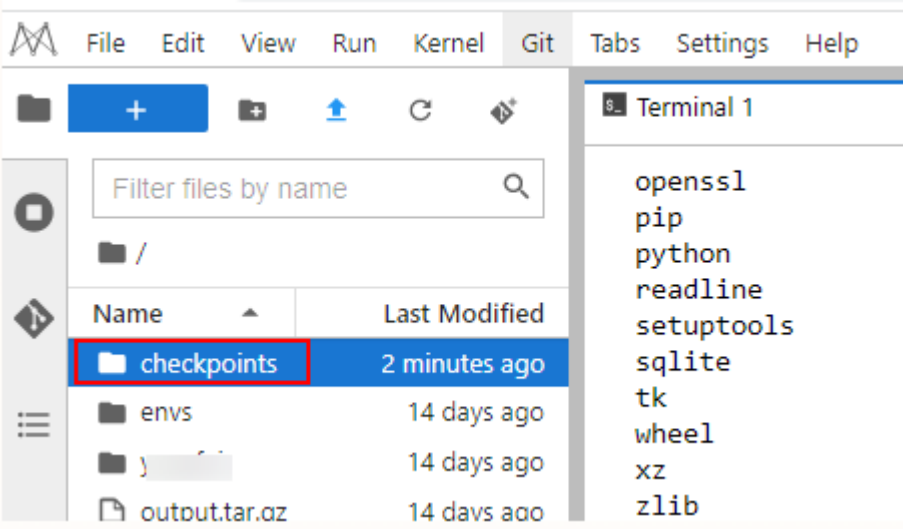
3.6 Other Faults

3.6.1 Failed to Open the checkpoints Folder in Notebook

checkpoints is a keyword in notebook. If a created folder is named **checkpoints**, the folder will not be opened, renamed, or deleted on JupyterLab. To access **checkpoints**, you have two options: either execute the command line in the

terminal to load the checkpoint files, or create a folder and transfer the checkpoint data to that folder.

Figure 3-5 Unavailable checkpoints in the JupyterLab navigation pane



Procedure

Open the terminal and perform operations using the CLI.

Method 1: Run the **cd checkpoints** command to open the **checkpoints** folder.

Method 2: Create a folder and move the data in the **checkpoints** folder to that folder.

1. Run the **mkdir** *xxx* command to create a folder, in which *xxx* is the folder name. Do not use **checkpoints** to name the folder.
2. Move the data in the **checkpoints** folder to the new folder and delete the **checkpoints** folder in the root directory.

```
mv checkpoints/* xxx
rm -r checkpoints
```

3.6.2 Failed to Use a Purchased Dedicated Resource Pool to Create New-Version Notebook Instances

Symptom

A dedicated resource pool that has been purchased cannot be selected for creating a notebook instance, resulting in the creation failure.

A message is displayed, indicating that the development environment has not been initialized in the dedicated resource pool.

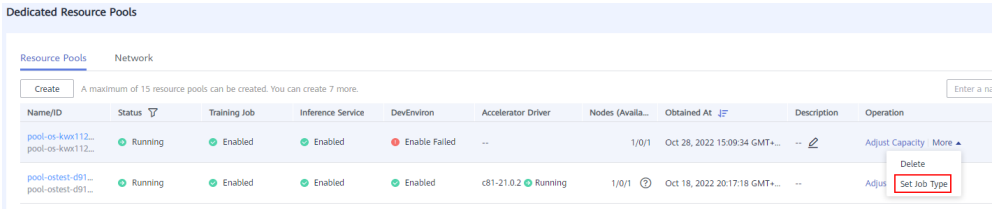
Possible Causes

A newly purchased dedicated resource pool can be used to create notebook instances only after its development environment is initialized.

Solution

Initialize the development environment on the dedicated resource pool page.

Step 1 Go to the **Dedicated Resource Pools** page and choose **More** > **Set Job Type** in the **Operation** column.



Step 2 In the **Set Job Type** dialog box, select **DevEnviron** and click **OK**. Then, the development environment is being initialized. After its status changes to **Running**, the newly purchased dedicated resource pool can be used to create notebook instances.

Figure 3-6 Setting job type to DevEnviron

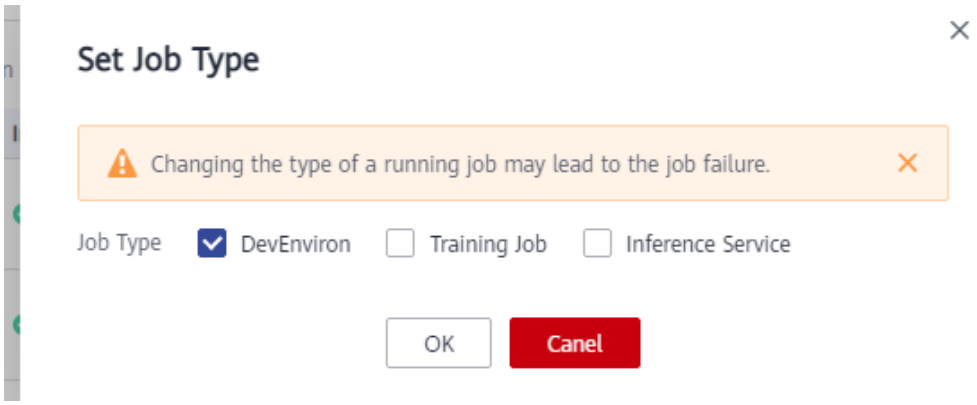
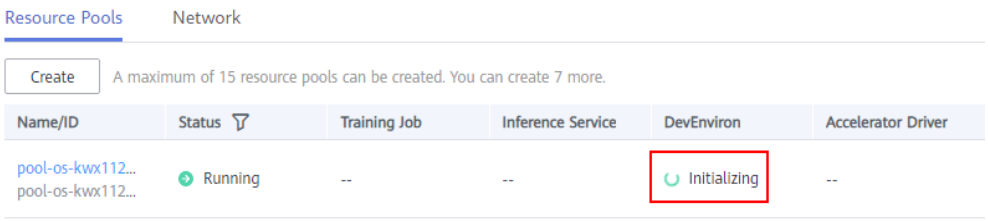


Figure 3-7 Initializing the development environment

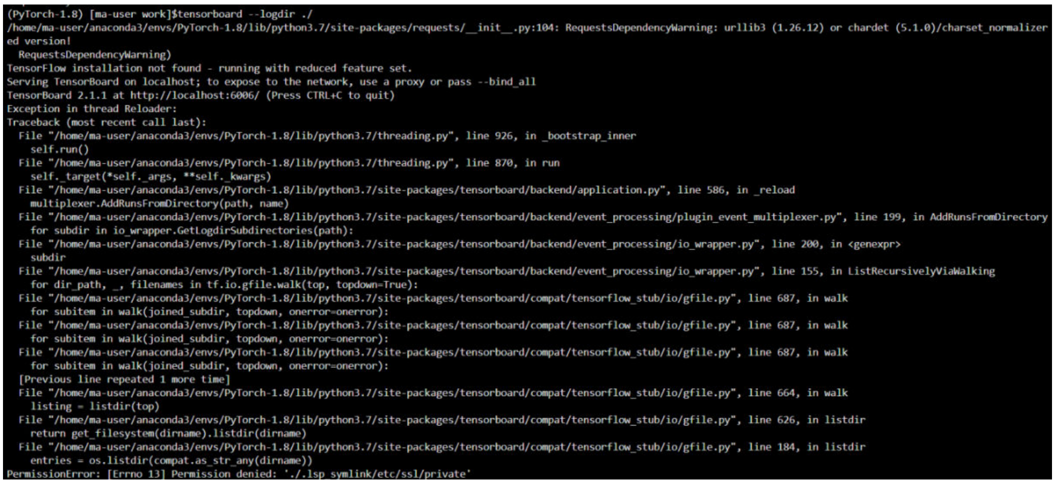


----End

3.6.3 Error Message "Permission denied" Is Displayed When the tensorboard Command Is Used to Open a Log File in a Notebook Instance

Symptom

When the **tensorboard --logdir ./** command is executed on the terminal of a notebook instance, the error message "[Errno 13] Permission denied..." is displayed.



Possible Causes

The current directory contains files on which you do not have permission.

Solution

Create a folder (for example, **tb_logs**), place the TensorBoard log file (for example, **tb.events**) in this folder, and run the tensorboard command. The following is an example command:

```
mkdir -p ./tb_logs
mv tb.events ./tb_logs
tensorboard --logdir ./tb_logs
```

4 Training Jobs

4.1 OBS Operation Issues

4.1.1 Error in File Reading

Symptom

- How to read the **json** and **npy** files when creating a training job.
- How the training job uses the cv2 library to read files.
- How to use the torch package in the MXNet environment.
- The following error occurs when the training job reads the file:
  ```
  NotFoundError (see above for traceback): Unsucessful TensorSliceReader constructor: Failed to find any matching files for xxx://xxx
  ```

Possible Cause

In ModelArts, user's data is stored in OBS buckets, but training jobs are running in containers. Therefore, users cannot access files in OBS buckets by accessing local paths.

Solution

If an error occurs when you read a file, you can use MoXing to copy data to a container and then access the data in the container. For details, see **1**.

You can also read files based on the file type. For details, see **Reading .json files**, **Reading .npy files**, and **Using the cv2 library to read files**, and **Using the torch package in the MXNet environment**.

1. If an error occurs when you read a file, you can use MoXing to copy data to a container and then access the data in the container as follows:
   ```
   import moxing as mox
   mox.file.make_dirs('/cache/data_url')
   mox.file.copy_parallel('obs://bucket-name/data_url', '/cache/data_url')
   ```
2. To **read .json files**, run the following code:
   ```
   json.loads(mox.file.read(json_path, binary=True))
   ```


3. To **use numpy.load to read .npy files**, run the following code:
   - Using the MoXing API to read files from OBS
     ```
     np.load(mox.file.read(_SAMPLE_PATHS['rgb'], binary=True))
     ```
   - Using the file module of MoXing to read and write OBS files
     ```
     with mox.file.File(_SAMPLE_PATHS['rgb'], 'rb') as f:
     np.load(f)
     ```
4. To **use the cv2 library to read files**, run the following code:
   ```
   cv2.imdecode(np.fromstring(mox.file.read(img_path), np.uint8), 1)
   ```
5. To **use the torch package in the MXNet environment**, run the following code:
   ```
   import os
   os.sysytem('pip install torch')
   import torch
   ```

4.1.2 Error Message Is Displayed Repeatedly When a TensorFlow-1.8 Job Is Connected to OBS

Symptom

After a training job is started based on TensorFlow-1.8 and the **tf.gfile** module is used to connect to OBS in code, the following log information is frequently printed:

```
Connection has been released. Continuing.
Found secret key
```

Possible Cause

This problem occurs in TensorFlow-1.8. This log is of the INFO level and is not error information. You can set an environment variable to shield logs of the INFO level. The environment variable must be set before the **import tensorflow** or **import moxing** command is executed.

Solution

Set the environment variable **TF_CPP_MIN_LOG_LEVEL** in code to shield logs of the INFO level. Detailed operations are as follows:

```
import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import tensorflow as tf
import moxing.tensorflow as mox
```

The mapping between **TF_CPP_MIN_LOG_LEVEL** and log levels is as follows:

```
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"]='1'    # Default level of logs to be displayed. All information is displayed.
os.environ["TF_CPP_MIN_LOG_LEVEL"]='2'    # Only warning and error information is displayed.
os.environ["TF_CPP_MIN_LOG_LEVEL"]='3'    # Only error information is displayed.
```


4.1.3 TensorFlow Stops Writing TensorBoard to OBS When the Size of Written Data Reaches 5 GB

Symptom

The following error message is displayed for a ModelArts training job:

```
Encountered Unknown Error EntityTooLarge
Your proposed upload exceeds the maximum allowed object size.:
If the signature check failed. This could be because of a time skew. Attempting to adjust the signer
```

Possible Cause

The size of files to be uploaded at a time is limited to 5 GB in OBS. TensorFlow may save the summary file in local cache. Therefore, when flush is triggered each time, the summary file overwrites the original file on OBS. If the size of the file exceeds 5 GB, the file stops being written.

Solution

If this problem occurs during the running of a training job, use the following method for troubleshooting.

1. You are advised to use the following local cache method:
   ```
   import moxing.tensorflow as mox
   mox.cache()
   ```

4.1.4 Error "Unable to connect to endpoint" Error Occurs When a Model Is Saved

Symptom

An error occurs in the log when a model is saved in a training job. The error details are as follows:

InternalError (see above for traceback): : Unable to connect to endpoint

Possible Cause

When OBS connections are unstable, the following error may occur: **Unable to connect to endpoint**

Solution

Add code to solve the problem of unstable OBS connections. You can add the following code at the beginning of the existing code so that TensorFlow can read and write ckpt and summary information in local cache mode:

```
import moxing.tensorflow as mox

mox.cache()
```


4.1.5 Error Message "BrokenPipeError: Broken pipe" Displayed When OBS Data Is Copied

Symptom

The error message is displayed when MoXing is used to copy data for a training job.

Figure 4-1 Error log

```
  readable=readable)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 358, in _make_put_request
  chunkedMode, methodName=methodName, readable=readable)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 390, in _make_request_with_retry
  raise e
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 369, in _make_request_with_retry
  _redirectLocation, skipAuthentication=skipAuthentication)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 436, in _make_request_internal
  conn = self._send_request(connect_server, method, path, header_config, entity, port, scheme, redirect, chunkedMode)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/client.py", line 586, in _send_request
  entity(util.conn_delegate(conn))
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/util.py", line 250, in entity
  conn.send(chunk)
File "/home/work/anaconda/lib/python3.6/site-packages/moxing/framework/file/src/obs/util.py", line 154, in send
  self.conn.send(data)
File "/home/work/anaconda/lib/python3.6/http/client.py", line 986, in send
  self.sock.sendall(data)
File "/home/work/anaconda/lib/python3.6/ssl.py", line 972, in sendall
  v = self.send(byte_view[count:])
File "/home/work/anaconda/lib/python3.6/ssl.py", line 941, in send
  return self._sslobj.write(data)
File "/home/work/anaconda/lib/python3.6/ssl.py", line 642, in write
  return self._sslobj.write(data)
BrokenPipeError: [Errno 32] Broken pipe
```

Possible Causes

The possible causes are as follows:

- In a large-scale distributed job, multiple nodes are concurrently copying files in the same bucket, leading to traffic control in the OBS bucket.
- There is a large number of OBS client connections. During the polling between processes or threads, an OBS client connection timed out if the server does not respond to it within 30 seconds. As a result, the server released the connection.

Solution

1. If the issue is caused by traffic control, the error code shown in the following figure is displayed. In this case, submit a service ticket. For details about OBS error codes, see **OBS Server-Side Error Codes**.

   Figure 4-2 Error log

   ```
   [ModelArts Service Log]2021-01-21 11:35:42,178 - file_io.py[line:652] - ERROR: Fail
               func=<bound method ObsClient.getObjectMetadata of <moxing.fram
               args=('bucket-816', 'AIRAW_AJ/c00454567/TeleQtj/23_zyl_J_quad_TeleN
               kwargs={}
   [ModelArts Service Log]2021-01-21 11:35:42,178 - file_io.py[line:658] - ERROR:
               stat:503
               errorCode:None
               errorMessage:None
               reason:Service Unavailable
               request-id:000001772302B34C9019B2408F9FF1B2
               retry:0
   ```


2. If the issue is caused by the large number of client connections, especially for files larger than 5 GB, OBS APIs cannot be directly called. In this case, use multiple threads to copy data. The timeout duration set on the OBS server is 30s. Run the following commands to reduce the number of processes:

```
# Configure the number of processes.
os.environ['MOX_FILE_LARGE_FILE_TASK_NUM']=1
import moxing as mox

# Copy files.
mox.file.copy_parallel(src_url=your_src_dir, dst_url=your_target_dir, threads=0, is_processing=False)
```

NOTE

When creating a training job, you can use the environment variable **_PARTIAL_MAXIMUM_SIZE** to configure the threshold (in bytes) for downloading large files in multiple parts. If the size of a file exceeds the threshold, the file will be downloaded in multiple parts concurrently.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.1.6 Error Message "ValueError: Invalid endpoint: obs.xxxx.com" Displayed in Logs

Symptom

When TensorBoard is used to directly write data in an OBS path for a training job, an error is displayed.

Figure 4-3 Error log

```
Traceback (most recent call last):
  File "/home/work/anaconda/lib/python3.6/threading.py", line 916, in _bootstrap_inner
    self.run()
  File "/home/work/anaconda/lib/python3.6/site-packages/tensorboardX/event_file_writer.py", line 219, in run
    self._record_writer.flush()
  File "/home/work/anaconda/lib/python3.6/site-packages/tensorboardX/event_file_writer.py", line 69, in flush
    self._py_recordio_writer.flush()
  File "/home/work/anaconda/lib/python3.6/site-packages/tensorboardX/record_writer.py", line 187, in flush
    self._writer.flush()
  File "/home/work/anaconda/lib/python3.6/site-packages/tensorboardX/record_writer.py", line 89, in flush
    s3 = boto3.client('s3', endpoint_url=os.environ.get('S3_ENDPOINT'))
  File "/home/work/anaconda/lib/python3.6/site-packages/boto3/__init__.py", line 91, in client
    return _get_default_session().client(*args, **kwargs)
  File "/home/work/anaconda/lib/python3.6/site-packages/boto3/session.py", line 263, in client
    aws_session_token=aws_session_token, config=config)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/session.py", line 835, in create_client
    client_config=config, api_version=api_version)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/client.py", line 85, in create_client
    verify, credentials, scoped_config, client_config, endpoint_bridge)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/client.py", line 287, in _get_client_args
    verify, credentials, scoped_config, client_config, endpoint_bridge)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/args.py", line 107, in get_client_args
    client_cert=new_config.client_cert)
  File "/home/work/anaconda/lib/python3.6/site-packages/botocore/endpoint.py", line 261, in create_endpoint
    raise ValueError("Invalid endpoint: %s" % endpoint_url)
ValueError: Invalid endpoint: obs.myhuaweicloud.com
```


Possible Causes

It is unstable to use TensorBoard to directly write data in OBS.

Solution

Locally write data and then copy it back to OBS.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.1.7 Error Message "errorMessage:The specified key does not exist" Displayed in Logs

Symptom

When MoXing is used to access an OBS path, the following error is displayed:

```
ERROR:root:
stat:404
errorCode:NoSuchKey
errorMessage:The specified key does not exist.
```

Possible Causes

The possible causes are as follows:

The object is unavailable in the bucket. For details about OBS error codes, see **OBS Server-Side Error Codes**.

Solution

1. Check whether the OBS path and object are in correct format.
2. Use the local PyCharm to remotely access notebook for debugging.

Summary and Suggestions

Before creating a training job, use a ModelArts development environment to debug training code. This maximally eliminates errors in code migration.

- Use in-cloud notebook for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.2 In-Cloud Migration Adaptation Issues

4.2.1 Failed to Import a Module

Symptom

The following error occurs in the log when a module is imported to a ModelArts training job:

```
Traceback (most recent call last):File "project_dir/main.py", line 1, in <module>from module_dir import module_file
ImportError: No module named module_dir
ImportError: No module named xxx
```

Possible Cause

- When a training job is imported to the module, the previous two error messages are displayed in the log. The possible causes are as follows:

  Before running code locally, you need to add **project_dir** to **PYTHONPATH** or install **project_dir** in **site-package**. However, on ModelArts, you can add **project_dir** to **sys.path** to solve this problem.

  Use **from module_dir import module_file** to import a package. The code structure is as follows:

  ```
  project_dir
  |- main.py
  |- module_dir
  |  |- __init__.py
  |  |- module_file.py
  ```

- When a training job is imported to the module, the error message **"ImportError: No module named xxx"** is displayed in the log. It can be determined that the environment does not contain the Python package on which the user depends.

Solution

- When a training job is imported to the module, the previous two error messages are displayed in the log. The solution is as follows:

  a. Ensure that the imported module contains **__init__.py** used for creating **module_dir**. **Possible Cause** provides the code structure.

  b. Because the location of **project_dir** in the container is unknown, use an absolute path by adding **project_dir** to **sys.path** in file **main.py**, and import the following information:

  ```
  import os
  import sys
  # __file__ is the absolute path of the main.py script.
  # os.path.dirname(__file__) is the parent directory of main.py, that is, the absolute path of project_dir.
  current_path = os.path.dirname(__file__)
  sys.path.append(current_path)
  # Import other modules after sys.path.append is executed.
  from module_dir import module_file
  ```

- When a training job is imported to the module, the error message **"ImportError: No module named xxx"** is displayed in the log. Add the following code to install the dependency package:

  ```
  import os
  os.system('pip install xxx')
  ```


4.2.2 Error Message "No module named .*" Displayed in Training Job Logs

Perform the following operations to locate the fault:

1. **Checking Whether the Dependency Package Is Available**
2. **Checking Whether the Dependency Package Path Can Be Detected**
3. **Checking Whether the Selected Resource Flavor Is Correct**
4. **Summary and Suggestions**

Checking Whether the Dependency Package Is Available

If the dependency package is unavailable, use either of the following methods to install it:

- Method 1 (recommended): **When you create an algorithm**, place the required file or installation package in the code directory.

  The required file varies depending on the dependency package type.

  - **If the dependency package is an open-source installation package**

    Create a file named **pip-requirements.txt** in the code directory, and specify the dependency package name and version in the format of *Package name*==*Version* in the file.

    For example, the OBS path specified by **Code Directory** contains model files and the **pip-requirements.txt** file. The code directory structure is as follows:

    ```
    |---OBS path to the model boot file
        |---model.py              # Model boot file
        |---pip-requirements.txt   # Customized configuration file, which specifies the name and version of the dependency package
    ```

    The following shows the content of the **pip-requirements.txt** file:

    ```
    alembic==0.8.6
    bleach==1.4.3
    click==6.6
    ```

  - **If the dependency package is a WHL package**

    If the training backend does not support the download of open-source installation packages or the use of custom WHL packages, the system cannot automatically download and install the package. In this case, place the WHL package in the code directory, create a file named **pip-requirements.txt**, and specify the name of the WHL package in the file. The dependency package must be in WHL format.

    For example, the OBS path specified by **Code Directory** contains model files, the WHL file, and the **pip-requirements.txt** file. The code directory structure is as follows:

    ```
    |---OBS path to the model boot file
        |---model.py              # Model boot file
        |---XXX.whl               # Dependency package. If multiple dependencies are required, place all of them here.
        |---pip-requirements.txt   # Customized configuration file, which specifies the name of the dependency package
    ```

    The following shows the content of the **pip-requirements.txt** file:

    ```
    numpy-1.15.4-cp36-cp36m-manylinux1_x86_64.whl
    tensorflow-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
    ```


- Method 2: Add the following code to the boot file to install the dependency package:
```
import os
os.system('pip install xxx')
```

In method 1, the dependency package can be downloaded and installed before the training job is started. In method 2, the dependency package is downloaded and installed during the running of the boot file.

Checking Whether the Dependency Package Path Can Be Detected

Before executing code locally, add **project_dir** to **PYTHONPATH** or install **project_dir** in **site-package**. ModelArts enables you to add **project_dir** to **sys.path** to resolve this issue.

Run **from module_dir import module_file** to import a package. The code structure is as follows:

```
project_dir
|- main.py
|- module_dir
|  |- __init__.py
|  |- module_file.py
```

Checking Whether the Selected Resource Flavor Is Correct

Error message "No module named npu_bridge.npu_init" is displayed for a training job.

```
from npu_bridge.npu_init import *
ImportError: No module named npu_bridge.npu_init
```

Check whether the flavor used by the training job supports NPUs. The possible cause is that the job selected a non-NPU flavor, for example, a GPU flavor. As a result, an error occurs when NPUs are used.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the in-cloud notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.2.3 Failed to Install a Third-Party Package

Symptom

- **How to install custom library functions** for ModelArts, for example, **apex**.
- The following error occurs when a third-party package is installed in the ModelArts training environment:
```
xxx.whl is not a supported wheel on this platform
```


Possible Cause

Error **xxx.whl is not a supported wheel on this platform** occurs, because the format of the name of the installed file is not supported. For details about the solution, see **2**.

Solution

1. **Installing the third-party package**

   a. For an existing package in **pip**, run the following code to install it:

   ```
   import os
   os.system('pip install xxx')
   ```

   b. For a package that do not exist in **pip**, for example, **apex**, use the following method to upload the installation package to an OBS bucket. In this example, the installation package has been uploaded to **obs://cnnorth4-test/codes/mox_benchmarks/apex-master/**. Add the following code to the boot file to install the package:

   ```
   try:
       import apex
   except Exception:
       import os
       import moxing as mox
       mox.file.copy_parallel('obs://cnnorth4-test/codes/mox_benchmarks/apex-master/', '/cache/apex-master')
       os.system('pip --default-timeout=100 install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--cuda_ext" /cache/apex-master')
   ```

2. **Installation error**

   If the **xxx.whl** file fails to be installed, perform the following steps to solve the problem:

   a. If the **xxx.whl** file fails to be installed, add the following code to the boot file to check the file name and version supported by the **pip** command.

   ```
   import pip
   print(pip.pep425tags.get_supported())
   ```

   The supported file names and versions are as follows:

   ```
   [('cp36', 'cp36m', 'manylinux1_x86_64'), ('cp36', 'cp36m', 'linux_x86_64'), ('cp36', 'abi3', 'manylinux1_x86_64'), ('cp36', 'abi3', 'linux_x86_64'), ('cp36', 'none', 'manylinux1_x86_64'), ('cp36', 'none', 'linux_x86_64'), ('cp35', 'abi3', 'manylinux1_x86_64'), ('cp35', 'abi3', 'linux_x86_64'), ('cp34', 'abi3', 'manylinux1_x86_64'), ('cp34', 'abi3', 'linux_x86_64'), ('cp33', 'abi3', 'manylinux1_x86_64'), ('cp33', 'abi3', 'linux_x86_64'), ('cp32', 'abi3', 'manylinux1_x86_64'), ('cp32', 'abi3', 'linux_x86_64'), ('py3', 'none', 'manylinux1_x86_64'), ('py3', 'none', 'linux_x86_64'), ('cp36', 'none', 'any'), ('cp3', 'none', 'any'), ('py36', 'none', 'any'), ('py3', 'none', 'any'), ('py35', 'none', 'any'), ('py34', 'none', 'any'), ('py33', 'none', 'any'), ('py32', 'none', 'any'), ('py31', 'none', 'any'), ('py30', 'none', 'any')]
   ```

   b. Change **faiss_gpu-1.5.3-cp36-cp36m-manylinux2010_x86_64.whl** to **faiss_gpu-1.5.3-cp36-cp36m-manylinux1_x86_64.whl**, and run the following code to install the package:

   ```
   import moxing as mox
   import os

   mox.file.copy('obs://wolfros-net/zp/AI/code/faiss_gpu-1.5.3-cp36-cp36m-manylinux2010_x86_64.whl','/cache/faiss_gpu-1.5.3-cp36-cp36m-manylinux1_x86_64.whl')
   os.system('pip install /cache/faiss_gpu-1.5.3-cp36-cp36m-manylinux1_x86_64.whl')
   ```

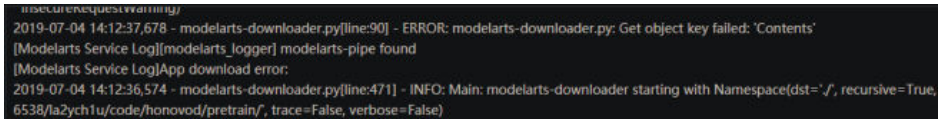

4.2.4 Failed to Download the Code Directory

Symptom

The code directory fails to be downloaded during training job running, and the following error message is displayed. See **Figure 4-4**.

```
ERROR: modelarts-downloader.py: Get object key failed: 'Contents'
```

Figure 4-4 Failure of getting content



Possible Cause

The code directory specified during training job creation does not exist. As a result, the training fails.

Solution

Check whether the code directory specified during training job creation, that is, the OBS bucket path, is correct based on the error cause. There are two methods to check whether it exists.

- Log in to the OBS console using the current account, and search for the OBS buckets, folders, and files in the path to check whether the code directory exists.
- Using APIs to check whether the directory exists: Run the following command in code to check whether the directory exists:

  ```
  import moxing as mox
  mox.file.exists('obs://obs-test/ModelArts/examples/')
  ```

4.2.5 Error Message "No such file or directory" Displayed in Training Job Logs

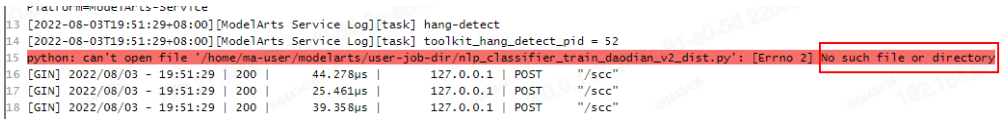
Symptom

If a training job failed, error message "No such file or directory" is displayed in logs.

If a training input path is unreachable, error message "No such file or directory" is displayed.

If a training boot file is unavailable, error message "No such file or directory" is displayed.

Figure 4-5 Example log for an unavailable training boot file



Possible Causes

- If the training input path is unreachable, the path is incorrect. Perform the following operations to locate the fault:
  a. **Checking Whether the Affected Path Is an OBS Path**
  b. **Checking Whether the Affected Path Is Available**
- If the training boot file is unavailable, the path to the training job boot command is incorrect. Rectify the fault by referring to **Checking the File Boot Path of a Training Job Created Using a Custom Image**.
- Multiple processes or workers read and write the same file. If SFS is used, check whether multiple nodes concurrently write the same file. Analyze the code and check whether multiple processes write the same file. It is a good practice to prevent multiple processes or nodes from concurrently reading and writing the same file.

Checking Whether the Affected Path Is an OBS Path

When using ModelArts, store data in an OBS bucket. However, the OBS path cannot be used to read data during the execution of the training code.

The reason is as follows:

After a training job is created, the training performance is poor if the running container is directly connected to OBS. To prevent this issue, the system automatically downloads the training data to the local path of the running container. Therefore, an error occurs if an OBS path is used in training code. For example, if the OBS path to the training code is **obs://bucket-A/training/**, the training code will be automatically downloaded to **${MA_JOB_DIR}/training/**.

For example, the OBS path to the training code is **obs://bucket-A/XXX/{training-project}/**, where **{training-project}** is the name of the folder where the training code is stored. During training, the system will automatically download the data from OBS **{training-project}** to the local path of the training container (**$MA_JOB_DIR/{training-project}/**).

If the affected path is to the training data, perform the following operations to resolve this issue (see **Parsing Input and Output Paths** for details):

1. When creating an algorithm, **set the code path parameter**, which defaults to **data_url**, in the input path mapping configuration.
2. Add a hyperparameter, which defaults to **data_url**, to the training code. Use **data_url** as the local path for inputting the training data.

Checking Whether the Affected Path Is Available

The code developed locally needs to be uploaded to the ModelArts backend. It is likely to incorrectly set the path to a dependency file in training code.

You are suggested to use the following general solution to obtain the absolute path to a dependency file through the OS API.

Example:

```
|---project_root              # Root directory for code
  |---BootfileDirectory       # Directory where the boot file is located
```



```
  |---bootfile.py          # Boot file
|---otherfileDirectory     # Directory where other dependency files are located
  |---otherfile.py         # Other dependency files
```

Do as follows to obtain the path to a dependency file, **otherfile_path** in this example, in the boot file:

```
import os
current_path = os.path.dirname(os.path.realpath(__file__)) # Directory where the boot file is located
project_root = os.path.dirname(current_path) # Root directory of the project, which is the code directory set on the ModelArts training console
otherfile_path = os.path.join(project_root, "otherfileDirectory", "otherfile.py")
```

Checking the File Boot Path of a Training Job Created Using a Custom Image

Take OBS path **obs://obs-bucket/training-test/demo-code** as an example. The training code in this path will be automatically downloaded to **${MA_JOB_DIR}/demo-code** in the training container, where **demo-code** is the last-level directory of the OBS path and can be customized.

If you use a custom image to create a training job, the system will automatically run the image boot command after the code directory is downloaded. The boot command must comply with the following rules:

- If the training startup script is a .py file, **train.py** for example, the boot command can be **python ${MA_JOB_DIR}/demo-code/train.py**.
- If the training startup script is an .sh file, **main.sh** for example, the boot command can be **bash ${MA_JOB_DIR}/demo-code/main.sh**,

where **demo-code** is the last-level directory of the OBS path and can be customized.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use in-cloud notebook for debugging. For details, see **JupyterLab Overview and Common Operations**.
- Use a local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Operation Process in a Local IDE**.

4.2.6 Failed to Find the .so File During Training

Symptom

During the execution of a ModelArts training job, the following error message is displayed in the log and the training failed:

```
libcudart.so.9.0 cannot open shared object file no such file or directory
```

Possible Cause

The CUDA version of the .so file generated during compilation is different from that of the training job.

Solution

If the CUDA version in the compilation environment is different from that in the training environment, an error will occur when a training job runs. For example, this error occurs if the .so file generated in the TensorFlow 1.13 development environment of CUDA version 10 is used in the TensorFlow 1.12 training environment of CUDA version 9.0.

To resolve this issue, perform the following operations:

1. Add the following command before executing a training job to check whether the .so file is available. If the .so file is available, go to **2**. Otherwise, go to **3**.
   ```
   import os;
   os.system(find /usr -name *libcudart.so*);
   ```
2. Configure the environment variable **LD_LIBRARY_PATH** and issue the training job again.

   For example, if the path for storing the .so file is **/use/local/cuda/lib64**, configure **LD_LIBRARY_PATH** as follows:
   ```
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
   ```
3. Run the following command to check whether the CUDA version of the training environment supports the .so file:
   ```
   os.system("cat /usr/local/cuda/version.txt")
   ```
   a. If so, import an external .so file (download it from the browser) and configure **LD_LIBRARY_PATH** in **2**.

   b. If not, replace the engine and issue the training job again. Alternatively, use a custom image to create a job. For details, see **Using a Custom Image to Train Models**.

4.2.7 ModelArts Training Job Failed to Parse Parameters and an Error Is Displayed in the Log

Symptom

The ModelArts training job failed to parse parameters, and the following error occurs:

```
error: unrecognized arguments: --data_url=xxx://xxx/xxx
error: unrecognized arguments: --init_method=tcp://job
absl.flags._exceptions.UnrecognizedFlagError:Unknown command line flag 'task_index'
```

Possible Cause

- The parameters are not defined.
- In the training environment, the system may input parameters that are not defined in the Python script. As a result, the parameters cannot be parsed, and an error is displayed in the log.

Solution

1. Define the parameters. The following is a code sample for reference:
   ```
   parser.add_argument('--init_method', default='tcp://xxx',help="init-method")
   ```
2. Replace **args = parser.parse_args()** with **args, unparsed = parser.parse_known_args()**. The following is a code sample:


```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument('--data_url', type=str, default=None, help='obs path of dataset')
args, unparsed = parser.parse_known_args()
```

4.2.8 Training Output Path Is Used by Another Job

Symptom

The following error message is displayed when a training job is created: Operation failed. Other running job contain train_url: /bucket-20181114/code_hxm/

Possible Cause

According to the error information, the same training output path is used by another job when a training job is created.

Solution

A training output path can be used by only one job in the running, queuing, or initializing state.

If this error occurs, check and re-set the training output path of the training job to avoid the job creation failure.

4.2.9 Error Message "RuntimeError: std::exception" Displayed for a PyTorch 1.0 Engine

Symptom

When a PyTorch 1.0 image is used, the following error message is displayed:

```
"RuntimeError: std::exception"
```

Possible Causes

The soft link of libmkldnn in the PyTorch 1.0 image conflicts with that of the native Torch. For details, see **conv1d fails in PyTorch 1.0**.

Solution

1. This issue is caused by library conflict in the environment. To resolve this issue, add the following code at the very beginning of the boot script:

   ```
   import os
   os.system("rm /home/work/anaconda3/lib/libmkldnn.so")
   os.system("rm /home/work/anaconda3/lib/libmkldnn.so.0")
   ```

2. Use the local PyCharm to remotely access notebook for debugging.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.

- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.2.10 Error Message "retCode=0x91, [the model stream execute failed]" Displayed in MindSpore Logs

Symptom

When MindSpore is used for training, the following error message is displayed:

```
[ERROR] RUNTIME(3002)model execute error, retCode=0x91, [the model stream execute failed]
```

Possible Causes

The speed of reading data cannot keep up with the model iteration speed.

Solution

1. Reduce shuffle operations during preprocessing.
   ```
   dataset = dataset.shuffle(buffer_size=x)
   ```
2. Disable data preprocessing, which may affect system performance.
   ```
   NPURunConfig(enable_data_pre_proc=Flase)
   ```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.2.11 Error Occurred When Pandas Reads Data from an OBS File If MoXing Is Used to Adapt to an OBS Path

Symptom

If MoXing is used to adapt to an OBS path, an error occurs when pandas of a later version reads data from an OBS file.

```
1. 'can't decode byte xxx in position xxx'
2. 'OSError:File isn't open for writing'
```

Possible Causes

MoXing does not support Pandas of a later version.

Solution

1. After the OBS path is adapted, change the file access mode from **r** to **rb** and change the **_write_check_passed** value in **mox.file.File** to **True**, as shown in the following is sample code:
   ```
   import pandas as pd
   import moxing as mox
   ```



```
mox.file.shift('os', 'mox')  # Replace the open operation of the operating system with the operation for adapting the mox.file.File to the OBS path.

param = {'encoding': 'utf-8'}
path = 'xxx.csv'
with open(path, 'rb') as f:
    f._wirte_check_passed = True
    df = pd.read_csv(ff, **param)
```

2. Use the local PyCharm to remotely access notebook for debugging.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.2.12 Error Message "Please upgrade numpy to >= xxx to use this pandas version" Displayed in Logs

Symptom

Dependency conflicts occur when other packages are installed. There are special requirements on the NumPy library. However, NumPy cannot be uninstalled. The error message similar to the following is displayed:

```
your numpy version is 1.14.5.Please upgrade numpy to >= 1.15.4 to use this pandas version
```

Possible Causes

Both Conda and pip packages are installed. Some packages cannot be uninstalled.

Solution

Perform the following operations to resolve this issue:

1. Uninstall the components that can be uninstalled in NumPy.
2. Delete the NumPy folder in the **site-packages** directory.
3. Install the required version again.

```
import os
os.system("pip uninstall -y numpy")
os.system('rm -rf /home/work/anaconda/lib/python3.6/site-packages/numpy/')
os.system("pip install numpy==1.15.4")
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.2.13 Reinstalled CUDA Version Does Not Match the One in the Target Image

Symptom

An error occurs after the engine version is reinstalled or a new CUDA package is compiled based on the existing image.

```
1. "RuntimeError: cuda runtime error (11) : invalid argument at /pytorch/aten/src/THC/
THCCachingHostAllocator.cpp:278"
2. "libcudart.so.9.0 cannot open shared object file no such file or directory"
3. "Make sure the device specification refers to a valid device. The requested device appears to be a
GPU,but CUDA is not enabled"
```

Possible Causes

The possible cause is as follows:

The CUDA version of the newly installed package does not match the CUDA version in the image.

Solution

Use the local PyCharm to remotely access notebook for debugging and installation.

1. Remotely log in to the selected image and run **nvcc -V** to obtain the CUDA version of the image.
2. Reinstall Torch. Ensure that the version matches the one obtained in the previous step.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **JupyterLab Overview and Common Operations**.
- Use a local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Operation Process in a Local IDE**.

4.2.14 Error ModelArts.2763 Occurred During Training Job Creation

Symptom

When a training job is created, error code ModelArts.2763 is displayed, indicating that the selected instance is invalid.

Possible Causes

The selected training flavor does not match the algorithm.

For example, the algorithm supports GPUs, but Ascend flavor is selected for creating the training job.

Solution

1. Check the training resource flavor configured in the algorithm code.
2. Check whether the resource flavor selected during training job creation is correct. If not, create a training job with the correct resource flavor.

4.2.15 Error Message "AttributeError: module '***' has no attribute '***'" Displayed Training Job Logs

Symptom

Error message "AttributeError: module '***' has no attribute '***'" is displayed in the logs of a training job, for example, "AttributeError: module 'torch' has no attribute 'concat'".

Possible Causes

The possible causes are as follows:

- The Python package is incorrectly used. There is no required variable or method in the Python package.
- The Python package version in the third-party pip source has been updated. As a result, the version of the Python package installed in the training job may also change. If a training job ran properly originally, but this issue occurs in the training job later, consider this cause.

Solution

- Use notebook for debugging.
- Specify a version for installation, for example, **pip install xxx==***1.x.x*.
- The third-party pip source may be updated at any time. To prevent this issue from occurring, create a custom image. For details, see **Using a Custom Image to Train Models (Model Training)**.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.2.16 System Container Exits Unexpectedly

Symptom

After a training job is created, the system container exits unexpectedly.

Figure 4-6 Error logs

```
[ModelArts Service Log]2022-10-11 19:17:35,178 - file_io.py[line:728] - WARNING: Retry=4, Wait=3.2, Timestamp=1665487055.178172, Function=getObject, args=
('modelarts-cn-north-4-test', 'modelarts/code-test'), kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:17:38,405 - file_io.py[line:728] - WARNING: Retry=3, Wait=6.4, Timestamp=1665487058.4054542, Function=getObject, args=
('modelarts-cn-north-4-test', 'modelarts/code-test'), kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:17:44,832 - file_io.py[line:728] - WARNING: Retry=2, Wait=12.8, Timestamp=1665487064.8322, Function=getObject, args=
('modelarts-cn-north-4-test', 'modelarts/code-test'), kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:17:57,663 - file_io.py[line:728] - WARNING: Retry=1, Wait=25.6, Timestamp=1665487077.6639552, Function=getObject,
args=('modelarts-cn-north-4-test', 'modelarts/code-test'), kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:18:23,266 - file_io.py[line:741] - ERROR: Failed to call:
    func=<bound method ObsClient.getObject of <moxing.framework.file.src.obs.client.ObsClient object at 0x7fb332a2a910>>
    args=('modelarts-cn-north-4-test', 'modelarts/code-test')
    kwargs={loadStreamInMemory:False, cache:False, }
[ModelArts Service Log]2022-10-11 19:18:23,267 - file_io.py[line:748] - ERROR:
    stat:404
    errorCode:NoSuchKey
    errorMessage:The specified key does not exist.
    reason:Not Found
    request-id:00000183C6C4010C66D399E000C0E366
    retry:0
[ModelArts Service Log]2022-10-11 19:18:23,267 - modelarts-downloader.py[line:90] - ERROR: modelarts-downloader.py: Download directory failed: [Errno
{'status': 404, 'reason': 'Not Found', 'errorCode': 'NoSuchKey', 'errorMessage': 'The specified key does not exist.', 'body': None, 'requestId':
'00000183C6C4010C66D399E000C0E366', 'hostId': 'tMe7zkB1evwTBxJLL7XbKfb68lgFdhz/M+xo5tgabLcrgw90DRbnKiHVYoJmYHPT', 'header': [('x-reserved', 'amazon, aws
and amazon web services are trademarks or registered trademarks of Amazon Technologies, Inc'), ('request-id', '00000183C6C4010C66D399E000C0E366'), ('id-2',
'32AAAQAAEAABAAAQAAEAABAAAQAAEAABCSdfbsDDEx4QVqrcVsoqSC/Io8A16gNg'), ('content-type', 'application/xml'), ('date', 'Tue, 11 Oct 2022 11:17:57 GMT'),
('content-length', '310')]]] file or directory or bucket not found.
```

Possible Causes

The possible causes are as follows:

1. An error occurred in OBS.
   a. Unavailable file: The specified key does not exist.
   b. Insufficient OBS permissions
   c. OBS traffic limiting
   d. Others
2. The disk space is insufficient.

Solution

1. For an OBS error:
   a. Unavailable file: The specified key does not exist.

      For details, see **Error Message "errorMessage:The specified key does not exist" Displayed in Logs**.
   b. Insufficient OBS permissions

      For details, see **What Should I Do If Error "stat:403 reason:Forbidden" Is Displayed in Logs When a Training Job Accesses OBS**.
   c. OBS traffic limiting

      For details, see **Error Message "BrokenPipeError: Broken pipe" Displayed When OBS Data Is Copied**.
   d. Others

      For details, see **OBS Server-Side Error Codes**. Alternatively, obtain the request ID and contact OBS customer service.
2. For insufficient disk space:

   For details, see **Common Issues Related to Insufficient Disk Space and Solutions**.

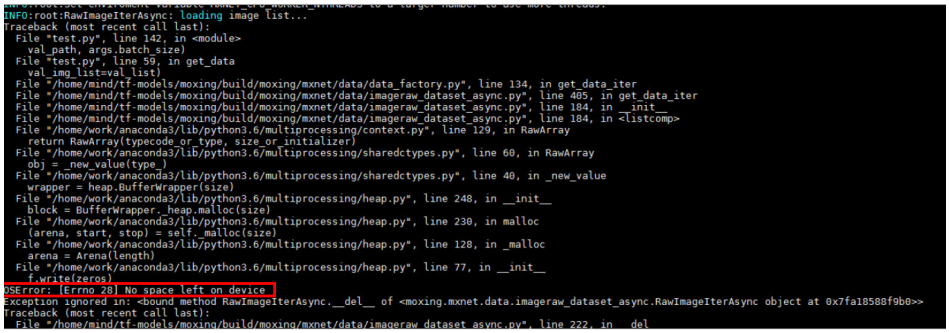
4.3 Hard Faults Due to Space Limit

4.3.1 Downloading Files Timed Out or No Space Left for Reading Data

Symptom

When data, code, or model is copied during training, the error message "No space left on device" is displayed.

Figure 4-7 Error log



Possible Causes

The possible causes are as follows:

- The disk space is insufficient.
- When a distributed job is executed, the **docker base size** configuration does not take effect on certain nodes. As a result, the storage space of the **/** root directory in the container is only the default value of 10 GB, which should be 50 GB, leading to the job training failure.
- The storage space is sufficient, but the error message "No Space left on device" is still displayed.

  If there are a large number of files in the same directory, the kernel creates an index table to accelerate file retrieval. If a large number of files are created in a short period of time, the number of indexes reaches the upper limit, and an error occurs.

  NOTE

  The issue occurs depending on the following factors:

  - A longer file name leads to a smaller upper limit for the number of files.
  - A smaller block size leads to a smaller upper limit for the number of files. (There are three block sizes, 1024 bytes, 2048 bytes, and 4096 bytes. The default size is 4096 bytes.)
  - The issue is more likely to occur if files are created in a shorter period of time. The reason is as follows: There is a cache, the size of which is determined based on the preceding two factors. When the number of files in the directory is large, the cache is enabled. The resources are released if they are not used.

Solution

1. Rectify the fault by following the operations described in **Error Message "write line error" Displayed in Logs**.

2. If the issue occurs only on certain nodes used by the distributed job, submit a service ticket to isolate the faulty nodes.
3. If the issue is caused by EulerOS restrictions, take the following measures:
   - Reduce the number of files in a single directory.
   - Slow down the file creation speed.
   - Disable the **dir_index** attribute of the Ext4 file system, which may affect the file retrieval performance. For details, see https://access.redhat.com/solutions/29894.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.3.2 Insufficient Container Space for Copying Data

Symptom

When a ModelArts training job was running, the error below was printed in the log. As a result, data failed to be copied to the container.

```
OSError:[Errno 28] No space left on device
```

Possible Causes

The container space is insufficient for downloading data.

Solution

1. Check if data is downloaded to the **/cache** directory. Each GPU node has a **/cache** directory with 4 TB of storage. Check if the directory is experiencing an excessive creation of files simultaneously, which will run out of inodes, leading to a shortage of space.
2. Check whether GPU resources are used. If CPU resources are used, **/cache** and the code directory share 10 GB of memory. As a result, the memory is insufficient. In this case, use GPU resources instead.
3. Add the following environment variable to the code:
   ```
   import os
   os.system('export TMPDIR=/cache')
   ```

4.3.3 Error Message "No space left" Displayed When a TensorFlow Multi-node Job Downloads Data to /cache

Symptom

During training job creation, error message "No space left" is displayed when a TensorFlow multi-node job downloads data to **/cache**.

Possible Cause

In a TensorFlow multi-node job, the **parameter server** (ps) and **worker** roles are started. The **ps** and **worker** roles are scheduled to the same machine. Training data is useless for **ps**. Therefore, the ps-related logic in code does not need to download the training data. If **ps** also downloads data to **/cache**, the actually downloaded data will be doubled. For example, if 2.5 TB data has been downloaded, the program displays a message indicating that space is insufficient because **/cache** has only 4 TB available space.

Solution

When a TensorFlow multi-node job is used to download data, the correct download logic is as follows:

```
import argparse
parser = argparse.ArgumentParser()
parser.add_argument("--job_name", type=str, default="")
args = parser.parse_known_args()

if args[0].job_name != "ps":
    copy.....................................
```

4.3.4 Size of the Log File Has Reached the Limit

Symptom

An error occurs during the running of a ModelArts training job, indicating that the size of the log file has reached the limit.

```
modelarts-pope: log length overflow(max:1073741824; already: 107341771; new:90), process will continue running silently
```

Possible Cause

Error information indicates that the size of the log file has reached the limit. After this error occurs, the volume of logs does not increase and the background continues to run.

Solution

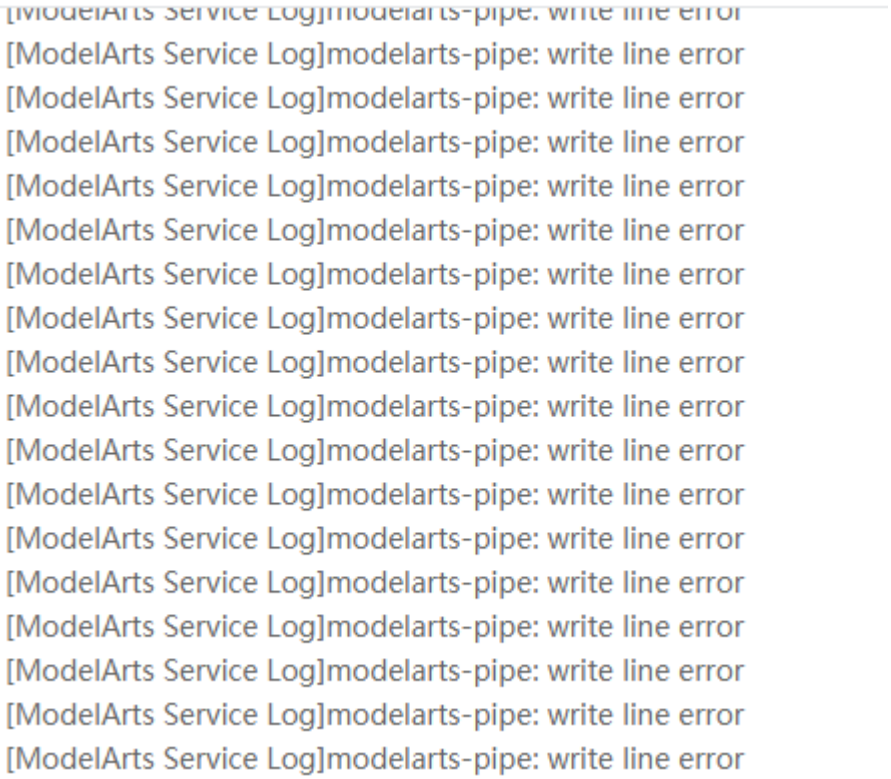
Reduce unnecessary log output from the boot file.

4.3.5 Error Message "write line error" Displayed in Logs

Symptom

During program running, a large number of error messages "write line error" are generated. This issue recurs each time the program runs at a specific progress.

Figure 4-8 Error log



Possible Causes

The possible causes are as follows:

- Core files are generated during the program running and exhaust the storage space in the **/** root directory.
- The 3.5 TB of storage space in the **/cache** directory is used up by the local data and files stored in it.

NOTE

The disk space for in-cloud training consists of the space from the following directories:

1. The **/** root directory, which is specified by **base size** in Docker. The default value is 10 GB. On the cloud, the value has been changed to 50 GB.
2. The **/cache** directory, which is 3.5 TB typically.

Solution

1. If core files are generated in the training job's work directory, add the code below at the beginning of the boot script to disable the generation of the core files.

```
import os
os.system("ulimit -c 0")
```

2. Check whether the dataset and checkpoint file have used up the storage space of the **/cache** directory.
3. Use the local PyCharm to remotely access notebook for debugging.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **JupyterLab Overview and Common Operations**.
- Use a local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Operation Process in a Local IDE**.

4.3.6 Error Message "No space left on device" Displayed in Logs

Symptom

When data, code, or model is copied during training, the error message "No space left on device" is displayed.

Figure 4-9 Error log

```
INFO:root:RawImageIterAsync: loading image list...
Traceback (most recent call last):
  File "test.py", line 142, in <module>
    val_path, args.batch_size)
  File "test.py", line 59, in get_data
    val_img_list=val_list)
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/data_factory.py", line 134, in get_data_iter
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/imageraw_dataset_async.py", line 405, in get_data_iter
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/imageraw_dataset_async.py", line 184, in __init__
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/imageraw_dataset_async.py", line 184, in <listcomp>
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/context.py", line 129, in RawArray
    return RawArray(typecode_or_type, size_or_initializer)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/sharedctypes.py", line 60, in RawArray
    obj = _new_value(type_)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/sharedctypes.py", line 40, in _new_value
    wrapper = heap.BufferWrapper(size)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/heap.py", line 248, in __init__
    block = BufferWrapper._heap.malloc(size)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/heap.py", line 230, in malloc
    (arena, start, stop) = self._malloc(size)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/heap.py", line 128, in _malloc
    arena = Arena(length)
  File "/home/work/anaconda3/lib/python3.6/multiprocessing/heap.py", line 77, in __init__
    f.write(zero)
OSError: [Errno 28] No space left on device
Exception ignored in: <bound method RawImageIterAsync.__del__ of <moxing.mxnet.data.imageraw_dataset_async.RawImageIterAsync object at 0x7fa18588f9b0>>
Traceback (most recent call last):
  File "/home/mind/tf-models/moxing/build/moxing/mxnet/data/imageraw_dataset_async.py", line 222, in __del
```

Possible Causes

The possible causes are as follows:

- The disk space is insufficient.
- When a distributed job is executed, the **docker base size** configuration does not take effect on certain nodes. As a result, the storage space of the **/** root directory in the container is only the default value of 10 GB, which should be 50 GB, leading to the job training failure.
- The storage space is sufficient, but the error message "No Space left on device" is still displayed.

  If there are a large number of files in the same directory, the kernel creates an index table to accelerate file retrieval. If a large number of files are created in a short period of time, the number of indexes reaches the upper limit, and an error occurs.

NOTE

The issue occurs depending on the following factors:

- A longer file name leads to a smaller upper limit for the number of files.
- A smaller block size leads to a smaller upper limit for the number of files. (There are three block sizes, 1024 bytes, 2048 bytes, and 4096 bytes. The default size is 4096 bytes.)
- This issue is more likely to occur if files are created in a shorter period of time.

Solution

1. Rectify the fault by following the operations described in **Error Message "write line error" Displayed in Logs**.
2. If the issue occurs only on certain nodes used by the distributed job, submit a service ticket to isolate the faulty nodes.
3. If the issue is caused by EulerOS restrictions, take the following measures:
   - Reduce the number of files in a single directory.
   - Slow down the file creation speed.
   - Disable the **dir_index** attribute of the Ext4 file system, which may affect the file retrieval performance. For details, see https://access.redhat.com/solutions/29894.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.3.7 Training Job Failed Due to OOM

Symptom

If a training job failed due to out of memory (OOM), possible symptoms as as follows:

1. Error code 137 is returned.
2. The log file contains error information with keyword **killed**.

Figure 4-10 Error log

```
Traceback (most recent call last):
  File "/home/ma-user/modelarts/user-job-dir/addernet-firstlast/main-imgnet.py", line 261, in <module>
    main()
  File "/home/ma-user/modelarts/user-job-dir/addernet-firstlast/main-imgnet.py", line 251, in main
    loss,acc = train_and_test(e, opt.alpha_start)
  File "/home/ma-user/modelarts/user-job-dir/addernet-firstlast/main-imgnet.py", line 243, in train_and_test
    acc = test(epoch, alpha_start, False)
  File "/home/ma-user/modelarts/user-job-dir/addernet-firstlast/main-imgnet.py", line 222, in test
    output = net(images, epoch, alpha_start)
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/nn/modules/module.py", line 541, in __call__
    result = self.forward(*input, **kwargs)
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/nn/parallel/data_parallel.py", line 152, in forward
    outputs = self.parallel_apply(replicas, inputs, kwargs)
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/nn/parallel/data_parallel.py", line 162, in parallel_apply
    return parallel_apply(replicas, inputs, kwargs, self.device_ids[:len(replicas)])
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/nn/parallel/parallel_apply.py", line 75, in parallel_apply
    thread.start()
  File "/home/ma-user/anaconda/lib/python3.6/threading.py", line 851, in start
    self._started.wait()
  File "/home/ma-user/anaconda/lib/python3.6/threading.py", line 551, in wait
    signaled = self._cond.wait(timeout)
  File "/home/ma-user/anaconda/lib/python3.6/threading.py", line 295, in wait
    waiter.acquire()
  File "/home/ma-user/anaconda/lib/python3.6/site-packages/torch/utils/data/_utils/signal_handling.py", line 66, in handler
    _error_if_any_worker_fails()
RuntimeError: DataLoader worker (pid 38077) is killed by signal: Killed.
```

3. Error message "RuntimeError: CUDA out of memory." is displayed in logs.

Figure 4-11 Error log

```
Traceback (most recent call last):
  File "memory_test.py", line 47, in <module>
    tmp_tensor = torch.empty(int(total_memory * 0.45), dtype=torch.int8, device='cuda')
RuntimeError: CUDA out of memory. Tried to allocate 14.29 GiB (GPU 0; 14.29 GiB total capacity; 0 bytes
already allocated; 14.29 GiB free; 0 bytes reserved in total by PyTorch)
```

4. Error message "Dst tensor is not initialized" is displayed in TensorFlow logs.

Possible Causes

The possible causes are as follows:

- GPU memory is insufficient.
- OOM occurred on certain nodes. This issue is typically caused by the node fault.

Solution

1. Modify hyperparameter settings to release unnecessary tensors.
   a. Modify network parameters, such as **batch_size**, **hide_layer**, and **cell_nums**.
   b. Release unnecessary tensors.
      ```
      del tmp_tensor
      torch.cuda.empty_cache()
      ```
2. Use the local PyCharm to remotely access notebook for debugging.
3. If the fault persists, submit a service ticket to locate the fault or even isolate the affected node.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

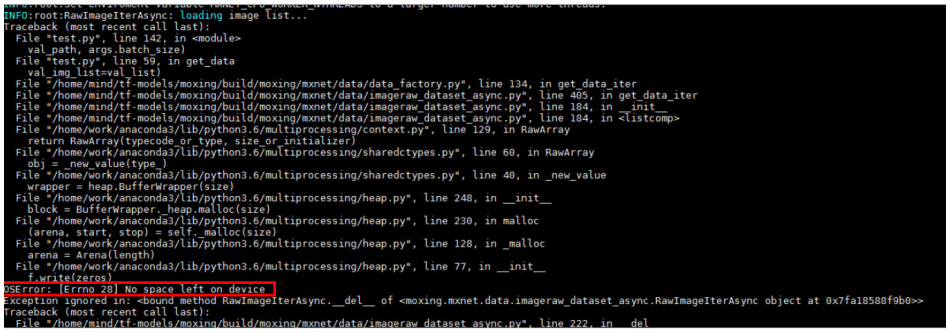
4.3.8 Common Issues Related to Insufficient Disk Space and Solutions

This section centrally describes common issues related to insufficient disk space and solutions to these issues.

Symptom

When data, code, or model is copied during training, error message "No space left on device" is displayed.

Figure 4-12 Error log



Possible Causes

The possible causes are as follows:

- The storage space in the **/cache** directory is used up by the local data and files stored in it.
- Data is decompressed when being processed. As a result, the data volume increases, and finally the storage space in the **/cache** directory is used up.
- Data is not saved in **/cache** or **/home/ma-user/** (**/cache** will be softly connected to **/home/ma-user/**). As a result, the system directory is fully occupied. The system directory supports only basic running of system functions. It cannot be used to store large volumes of data.
- During the training of certain jobs, checkpoint files will be generated and updated. If historical checkpoint files are not deleted after an update, the **/cache** directory will be exhausted.
- The storage space is sufficient, but the error message "No Space left on device" is still displayed. This may be triggered by insufficient inodes or an error in the file index cache of the operating system. As a result, no file can be created in the system disk, and finally data disks are used up.

NOTE

The conditions for triggering an error in the file index cache are as follows:

- A longer file name leads to a smaller upper limit for the number of files.
- A smaller block size leads to a smaller upper limit for the number of files. (There are three block sizes, 1024 bytes, 2048 bytes, and 4096 bytes. The default size is 4096 bytes.)
- This issue is more likely to occur if files are created in a shorter period of time. The reason is as follows: There is a cache, the size of which is determined based on the preceding two factors. When the number of files in the directory is large, the cache will be enabled and released with the files.

- Core files are generated during the program running and exhaust the storage space in the / root directory.

Solution

1. Obtain the sizes of the dataset, decompressed dataset, and checkpoint file and check whether they have exhausted the disk space.
2. If the volume of data exceeds the **/cache** size, use SFS to attach more data disks for expanding the storage size.
3. Save the data and checkpoint in **/cache** or **/home/ma-user/**.
4. Check the checkpoint logic and ensure that historical checkpoints are deleted so that they will not use up the storage space in **/cache**.
5. If the file size is smaller than the **/cache** size, and the number of files exceeds 500,000, the issue may be caused by insufficient inodes or an error in the file index cache of the operating system. In this case, do as follows to resolve this issue:
   - Reduce the number of files in a single directory.
   - Slow down the file creation speed. For example, during data decompression, add a sleep period of 5s before decompressing the next piece of data.
6. If core files are generated in the training job's work directory, add the code below at the beginning of the boot script to disable the generation of the core files. (debug code in a development environment before adding the code):

```
import os
os.system("ulimit -c 0")
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.4 Internet Access Issues

4.4.1 Error Message "Network is unreachable" Displayed in Logs

Symptom

When PyTorch is used, the following error message will be displayed in logs after **pretrained** in **torchvision.models** is set to **True**:

```
'OSError: [Errno 101] Network is unreachable'
```

Possible Causes

For security purposes, ModelArts internal training nodes are not allowed to access the Internet.

Solution

1. Change the **pretrained** value to **False**, download the pre-trained model, and load the path to this model.

```
import torch
import torchvision.models as models

model1 = models.resnet34(pretrained=False, progress=True)
checkpoint = '/xxx/resnet34-333f7ec4.pth'
state_dict = torch.load(checkpoint)
model1.load_state_dict(state_dict)
```

2. Use the local PyCharm to remotely access notebook for debugging.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.4.2 URL Connection Timed Out in a Running Training Job

Symptom

In a running training job, a URL connection timeout error occurs.

```
urllib.error.URLERROR:<urlopen error [Errno 110] Connection timed out>
```

Possible Causes

For security purposes, ModelArts is not allowed to access the Internet to download data.

Solution

Download the required data to a local directory and upload it to OBS. Then, access the OBS path from ModelArts to obtain the data.

4.5 Permission Issues

4.5.1 What Should I Do If Error "stat:403 reason:Forbidden" Is Displayed in Logs When a Training Job Accesses OBS

Symptom

When a training job accesses OBS, an error occurs.

Figure 4-13 Error log

```
ERROR:root:Failed to call:
    func=<bound method ObsClient.getObjectMetadata of <moxing.framework.file.src.obs.client.ObsClient object at 0x7fddb4ad06d0>>
    args=('bucket-cv-competition-bj4', 'fangjiemin/output/')
    kwargs={}
ERROR:root:
    stat:403
    errorCode:None
    errorMessage:None
    reason:Forbidden
    request-id:00000179D5ACCAC445CAA1A71019C9D0
    retry:0
```

Possible Causes

The possible causes are as follows:

- The OBS permission is incorrect. As a result, data cannot be read.

Solution

Verify that OBS permissions are correctly assigned. If the problem persists, troubleshoot by following the instructions provided in **Why Can't I Access OBS (403 AccessDenied) After Being Granted with the OBS Access Permission?**.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.
- If an error occurred in OBS, identify the cause based on the error information, including the error code and message. For details about OBS error codes, see **OBS Server-Side Error Codes**.

4.5.2 Error Message "Permission denied" Displayed in Logs

Symptom

When a training job accesses the attached EFS disks or executes the .sh boot script, an error occurs.

- [Errno 13]Permission denied: '/xxx/xxxx'

  Figure 4-14 Error log

  ```
  Traceback (most recent call last):
    File "codes/prepare_listdir.py", line 11, in <module>
      rec_file_list = os.listdir(recurrent_path)
  OSError: [Errno 13] Permission denied: '/data/recurrent'
  ```

- bash: /bin/ln: Permission denied
- bash:/home/ma-user/.pip/pip.conf: Permission Denied (in a custom image)
- tee: /xxx/xxxx: Permission denied cp: cannot stat '' No such file or directory (in a custom image)

Possible Causes

The possible causes are as follows:

- [Errno 13]Permission denied: '/xxx/xxxx'
  - When data is uploaded, the ownership and permissions to the file are not changed. As a result, the work user group does not have the permission to access the training job.
  - After the .sh file in the code directory is copied to the container, the execution permission is not granted for the file.
- bash: /bin/ln: Permission denied

  For security purposes, the ln command is not supported.
- bash:/home/ma-user/.pip/pip.conf: Permission Denied

  After the version of training jobs is switched from V1 to V2, the UID of the **ma-user** user is still **1102**.
- tee: /xxx/xxxx: Permission denied cp: cannot stat '': No such file or directory

  The used startup script is **run_train.sh** of an earlier version. Some environment variables in the script are unavailable in the training jobs of the new version.
- The APIs using the Python file concurrently read and write the same file.

Solution

1. Add permissions to access the attached EFS disks so that the permissions are the same as those of user group (1000) used in the training container. For example, if the **/nas** disk is attached, run the following command:

   ```
   chown -R 1000: 1000 /nas
   Or
   chmod 777 -R /nas
   ```

2. If the execution permission has not been granted for the .sh file used by the custom image, run **chmod +x xxx.sh** to grant the permission before starting the script.
3. On the ModelArts console, if a training job is created using a custom image, a V2 container image is started using UID 1000 by default. In this case, change the UID of the **ma-user** user from 1102 to 1000. To obtain the sudo permission, comment out the sudoers line.


```
FROM {your-v1-custom-docker-image or other docker-image}

USER root

# prepare moxing_framework and seccomponent package
# and chmod/chown moxing_framework package to the right permission or owner (ma-user)

RUN groupadd ma-group -g 1000 && \
    useradd -d /home/ma-user -m -u 1000 -g 1000 -s /bin/bash ma-user && \
    chmod 770 /home/ma-user && \
    # usermod -a -G work ma-user && \
    # alien -i seccomponent-1.0.2-2.0.release.x86_64.rpm && \
    chmod 770 /root && \
    # or silver bullet of files permission
    # chmod -R 777 /root && \
    usermod -a -G root ma-user

# ENV LD_LIBRARY_PATH=/usr/local/seccomponent/lib:$LD_LIBRARY_PATH

# RUN echo "ma-user  ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers

# RUN pip install moxing_framework-2.0.0.rc2.4b57a67b-py2.py3-none-any.whl

USER ma-user
WORKDIR /home/ma-user
```

4. Migrate environment variables from V1 training jobs to V2 training jobs.
   - Use V2 **MA_NUM_HOSTS** (the number of selected training nodes) to replace V1 **DLS_TASK_NUMBER**.
   - Use V2 **VC_TASK_INDEX** (or **MA_TASK_INDEX** that will be available later) to replace V1 **DLS_TASK_INDEX**. Obtain the environment variable using the method provided in the demo script for compatibility.
   - Use V2 **${MA_VJ_NAME}-${MA_TASK_NAME}-0.${MA_VJ_NAME}:6666** to replace V1 **BATCH_CUSTOM0_HOSTS**.
   - Use V2 **${MA_VJ_NAME}-${MA_TASK_NAME}-{N}.${MA_VJ_NAME}:6666** to replace V1 **BATCH_CUSTOM{N}_HOSTS** generally.
5. Check whether there are settings that allow concurrent reading and writing of the same file in the code. If so, modify the settings to forbid this operation.

   If a job uses multiple cards, the same code for reading and writing data may be available on each card. In this case, do as follows to modify the code:

```
import moxing as mox
from mindspore.communication import init, get_rank, get_group_size
init()
rank_id = get_rank()
# Enable only card 0 to download data.
if rank_id % 8 == 0:
    mox.file.copy_parallel('obs://bucket-name/dir1/dir2/', '/cache')
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use in-cloud notebook for debugging. For details, see **JupyterLab Overview and Common Operations**.

- Use a local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Operation Process in a Local IDE**.

4.6 GPU Issues

4.6.1 Error Message "No CUDA-capable device is detected" Displayed in Logs

Symptom

An error similar to the following occurs during the running of the program:

```
1. 'failed call to cuInit: CUDA_ERROR_NO_DEVICE:  no CUDA-capable device is detected'
2. 'No CUDA-capable device is detected although requirements are installed'
```

Possible Causes

The possible causes are as follows:

- **CUDA_VISIBLE_DEVICES** has been incorrectly set.
- CUDA operations are performed on GPUs with IDs that are not specified by **CUDA_VISIBLE_DEVICES**.

Solution

1. Do not change the **CUDA_VISIBLE_DEVICES** value in the code. Use its default value.
2. Ensure that the specified GPU IDs are within the available GPU IDs.
3. If the error persists, print the **CUDA_VISIBLE_DEVICES** value and debug it in the notebook, or run the following commands to check whether the returned result is **True**:

```
import torch
torch.cuda.is_available()
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.6.2 Error Message "RuntimeError: connect() timed out" Displayed in Logs

Symptom

When PyTorch is used for distributed training, the following error occurs.

Figure 4-15 Error log

```
INFO - 03/23/21 17:20:50 - 0:00:04 - Building data done with 1331166 images loaded.
Traceback (most recent call last):
  File "swav-master/main_swav.py", line 500, in <module>
    main()
  File "swav-master/main_swav.py", line 191, in main
    mp.spawn(main_worker, nprocs=args.ngpu, args=())
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/multiprocessing/spawn.py", line 171, in spawn
    while not spawn_context.join():
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/multiprocessing/spawn.py", line 118, in join
    raise Exception(msg)
Exception:

-- Process 2 terminated with the following error:
Traceback (most recent call last):
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/multiprocessing/spawn.py", line 19, in _wrap
    fn(i, *args)
  File "/cache/user-job-dir/swav-master/main_swav.py", line 231, in main_worker
    rank=args.rank)
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/distributed/distributed_c10d.py", line 397, in init_process_group
    store, rank, world_size = next(rendezvous_iterator)
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/distributed/rendezvous.py", line 168, in _env_rendezvous_handler
    store = TCPStore(master_addr, master_port, world_size, start_daemon)
RuntimeError: connect() timed out.
```

Possible Causes

If data is copied before this issue occurs, data copy on all nodes is not complete at the same time. If you perform **torch.distributed.init_process_group()** when data copy is still in progress on certain nodes, the connection timed out.

Solution

If the issue is caused by asynchronous data copy between nodes and no barrier occurs, perform **torch.distributed.init_process_group()** before copying data, copy data based on **local_rank()==0**, call **torch.distributed.barrier()**, and wait until data copy is complete on all nodes. For details, see the following code:

```
import moxing as mox
import torch

torch.distributed.init_process_group()
if local_rank == 0:
    mox.file.copy_parallel(src,dst)

torch.distributed.barrier()
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.6.3 Error Message "cuda runtime error (10) : invalid device ordinal at xxx" Displayed in Logs

Symptom

A training job failed, and the following error is displayed in logs.

Figure 4-16 Error log

```
  main()
File "train.py", line 278, in main
  torch.cuda.set_device(args.local_rank)
File "/home/work/anaconda/lib/python3.6/site-packages/torch/cuda/__init__.py", line 300, in set_device
  torch._C._cuda_setDevice(device)
RuntimeError: cuda runtime error (10) : invalid device ordinal at /pytorch/torch/csrc/cuda/Module.cpp:37
```

Possible Causes

The possible causes are as follows:

- The **CUDA_VISIBLE_DEVICES** setting does not comply with job specifications. For example, you select a job with four GPUs, and the IDs of available GPUs are 0, 1, 2, and 3. However, when you perform CUDA operations, for example **tensor.to(device="cuda:7")**, tensors are specified to run on GPU 7, which is beyond the available GPU IDs.
- GPUs are damaged on resource nodes if CUDA operations are performed on a GPU with a specified ID. As a result, the number of GPUs that can be detected is less than the selected specifications.

Solution

1. Perform CUDA operations on the GPUs with IDs specified by **CUDA_VISIBLE_DEVICES**.
2. If a GPU on a resource node is damaged, contact technical support.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **JupyterLab Overview and Common Operations**.
- Use a local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Operation Process in a Local IDE**.

4.6.4 Error Message "RuntimeError: Cannot re-initialize CUDA in forked subprocess" Displayed in Logs

Symptom

When PyTorch is used to start multiple processes, the following error message is displayed:

```
RuntimeError: Cannot re-initialize CUDA in forked subprocess
```

Possible Causes

The multi-processing startup mode is incorrect.

Solution

For details, see **Writing Distributed Applications with PyTorch**.


```
"""run.py:"""
#!/usr/bin/env python
import os
import torch
import torch.distributed as dist
import torch.multiprocessing as mp

def run(rank, size):
    """ Distributed function to be implemented later. """
    pass

def init_process(rank, size, fn, backend='gloo'):
    """ Initialize the distributed environment. """
    os.environ['MASTER_ADDR'] = '127.0.0.1'
    os.environ['MASTER_PORT'] = '29500'
    dist.init_process_group(backend, rank=rank, world_size=size)
    fn(rank, size)


if __name__ == "__main__":
    size = 2
    processes = []
    mp.set_start_method("spawn")
    for rank in range(size):
        p = mp.Process(target=init_process, args=(rank, size, run))
        p.start()
        processes.append(p)

    for p in processes:
        p.join()
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.6.5 No GPU Is Found for a Training Job

Symptom

The following error message is displayed during the running of a ModelArts training job:

```
failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
```

Possible Cause

According to error information, the error cause is that the training job running program cannot read the GPU.

Solution

Check whether the following configuration information is added to code and set the GPU visible to the program based on the error message:

```
os.environ['CUDA_VISIBLE_DEVICES'] = '0,1,2,3,4,5,6,7'
```


In the preceding information, **0** is a GPU ID of the server. The GPU ID can be 0, 1, 2, 3, or the like, indicating a GPU ID visible to the program. If the configuration information is not added, the GPU corresponding to the ID is unavailable.

4.7 Service Code Issues

4.7.1 Error Message "pandas.errors.ParserError: Error tokenizing data. C error: Expected .* fields" Displayed in Logs

Symptom

When pandas is used to read CSV data, the following error is displayed in logs, and the training job failed:

```
pandas.errors.ParserError: Error tokenizing data. C error: Expected 4 field
```

Possible Causes

The number of columns in each row of the CSV file is different.

Solution

Use either of the following methods to resolve this issue:

- Check the CSV file and delete the lines with extra columns.
- Run the following commands to ignore the lines with extra columns:

```
import pandas as pd
pd.read_csv(filePath,error_bad_lines=False)
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.7.2 Error Message "max_pool2d_with_indices_out_cuda_frame failed with error code 0" Displayed in Logs

Symptom

After PyTorch 1.3 is upgraded to 1.4, the following error message is displayed:

```
"RuntimeError:max_pool2d_with_indices_out_cuda_frame failed with error code 0"
```

Possible Causes

The PyTorch 1.4 engine is incompatible with that of PyTorch 1.3.

Solution

1. Run the following commands to add contiguous data:
```
images = images.cuda()
pred = model(images.permute(0, 3, 1, 2).contigous())
```
2. Roll back to PyTorch 1.3.
3. Use the local PyCharm to remotely access notebook for debugging.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.7.3 Training Job Failed with Error Code 139

Symptom

The training job failed, and error code 139 is returned.

Possible Causes

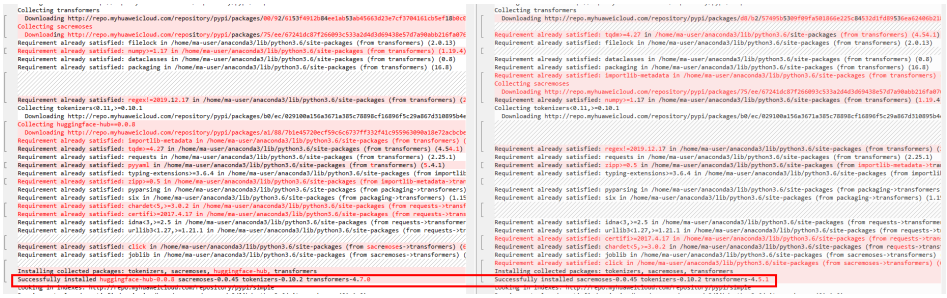
The possible causes are as follows:

- Certain pip packages in the pip source have been updated, leading to data incompatibility. For example, an error occurs when the transformers package is imported after the package update.
- The user code has a bug, leading to memory overwriting or unauthorized memory access.
- An unknown system error occurs. In this case, create the training job again. If the fault persists, submit a service ticket.

Solution

1. If the training job succeeded before and no modification has been made, compare the logs in the two cases and check whether any dependency package has been updated in the pip source.

Figure 4-17 Log comparison



2. Use the local PyCharm to remotely access notebook for debugging.
3. If the fault persists, contact technical support engineers.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **JupyterLab Overview and Common Operations**.
- Use a local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Operation Process in a Local IDE**.

4.7.4 Debugging Training Code in the Cloud Environment If a Training Job Failed

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.7.5 Error Message "'(slice(0, 13184, None), slice(None, None, None))' is an invalid key" Displayed in Logs

Symptom

The following error message is displayed during training:

```
TypeError: '(slice(0, 13184, None), slice(None, None, None))' is an invalid key
```

Possible Causes

The data selected for segmentation is incorrect.

Solution

Run the following command to resolve the issue:

```
X = dataset.iloc[:,:-1].values
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.7.6 Error Message "DataFrame.dtypes for data must be int, float or bool" Displayed in Logs

Symptom

The following error message is displayed during training:

```
DataFrame.dtypes for data must be int, float or bool
```

Possible Causes

The possible cause is as follows:

The training data is not of the int, float, or bool type.

Solution

Run the following commands to convert the error column:

```
from sklearn import preprocessing
lbl = preprocessing.LabelEncoder()
train_x['acc_id1'] = lbl.fit_transform(train_x['acc_id1'].astype(str))
```

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **JupyterLab Overview and Common Operations**.
- Use a local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Operation Process in a Local IDE**.

4.7.7 Error Message "CUDNN_STATUS_NOT_SUPPORTED" Displayed in Logs

Symptom

The following error message is displayed during PyTorch training:

```
RuntimeError: cuDNN error: CUDNN_STATUS_NOT_SUPPORTED. This error may appear if you passed in a non-contiguous input.
```

Possible Causes

The input data is not of contiguous type, which is not supported by cuDNN.

Solution

1. Disable cuDNN before training.

   ```
   torch.backends.cudnn.enabled = False
   ```

2. Convert the input data into contiguous data.

   ```
   images = images.cuda()
   images = images.permute(0, 3, 1, 2).contigous()
   ```


Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.7.8 Error Message "Out of bounds nanosecond timestamp" Displayed in Logs

Symptom

When pandas.to_datetime is used to convert time, the following error message is displayed:

```
pandas._libs.tslibs.np_datetime.OutOfBoundsDatetime: Out of bounds nanosecond timestamp: 1-01-02 13:20:00
```

Possible Causes

The time is out of the permitted range. For details, see the **official document**.

Solution

Check the time. Timestamps in pandas are in the unit of nanosecond. Ensure that the time is within the following permitted range:

- Earliest time: 1677-09-22 00:12:43.145225
- Latest time: 2262-04-11 23:47:16.854775807

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.7.9 Error Message "Unexpected keyword argument passed to optimizer" Displayed in Logs

Symptom

After Keras is upgraded to 2.3.0 or later, the following error message is displayed:

```
TypeError: Unexpected keyword argument passed to optimizer: learning_rate
```

Possible Causes

Certain parameters have been renamed in Keras. For details, see **Keras 2.3.0**.

Figure 4-18 API changes

- Rename `lr` to `learning_rate` for all optimizers.

Solution

Rename **learning_rate lr**.

Summary and Suggestions

Before creating a training job, use the ModelArts development environment to debug the training code to maximally eliminate errors in code migration.

- Use the online notebook environment for debugging. For details, see **Using JupyterLab to Develop a Model**.
- Use the local IDE (PyCharm or VS Code) to access the cloud environment for debugging. For details, see **Using the Local IDE to Develop a Model**.

4.7.10 Error Message "no socket interface found" Displayed in Logs

Symptom

An NCCL debug log level is set in a distributed job executed using a PyTorch image.

```
import os
os.environ["NCCL_DEBUG"] = "INFO"
```

The following error message is displayed.

Figure 4-19 Error log

```
job0879f61e-job-base-pda-2-0:712:712 [0] bootstrap.cc:37 NCCL WARN Bootstrap : no socket interface found
job0879f61e-job-base-pda-2-0:712:712 [0] NCCL INFO init.cc:128 -> 3
job0879f61e-job-base-pda-2-0:712:712 [0] NCCL INFO bootstrap.cc:76 -> 3
job0879f61e-job-base-pda-2-0:712:712 [0] NCCL INFO bootstrap.cc:245 -> 3
job0879f61e-job-base-pda-2-0:712:712 [0] NCCL INFO bootstrap.cc:266 -> 3
Traceback (most recent call last):
  File "train_net.py", line 1923, in <module>
    main_worker(args)
  File "train_net.py", line 355, in main_worker
    network = torch.nn.parallel.DistributedDataParallel(network, device_ids=device_ids, find_unused_parameters=True)
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/nn/parallel/distributed.py", line 298, in __init__
    self.broadcast_bucket_size)
  File "/home/work/anaconda/lib/python3.6/site-packages/torch/nn/parallel/distributed.py", line 480, in _distributed_broadcast_coalesced
    dist._broadcast_coalesced(self.process_group, tensors, buffer_size)
RuntimeError: NCCL error in: /pytorch/torch/lib/c10d/ProcessGroupNCCL.cpp:374, internal error
Traceback (most recent call last):
```

Possible Causes

The environment variables **NCCL_IB_TC**, **NCCL_IB_GID_INDEX**, and **NCCL_IB_TIMEOUT** are not configured. As a result, the communication is slow and unstable, and the IB communication is interrupted.

Solution

Add environment variables to the code.

```
import os
os.environ["NCCL_IB_TC"] = "128"
```



```
os.environ["NCCL_IB_GID_INDEX"] = "3"
os.environ["NCCL_IB_TIMEOUT"] = "22"
```

4.7.11 Error Message "Runtimeerror: Dataloader worker (pid 46212) is killed by signal: Killed BP" Displayed in Logs

Symptom

During the running of a training job, error message "Runtimeerror: Dataloader worker (pid 46212) is killed by signal: Killed BP" is displayed in logs.

Possible Causes

The Dataloader process exits because the batch size is too large.

Solution

Decrease the batch size.

4.7.12 Error Message "AttributeError: 'NoneType' object has no attribute 'dtype'" Displayed in Logs

Symptom

Code can run properly in the notebook Keras image. When tensorflow.keras is used for training, error message "AttributeError: 'NoneType' object has no attribute 'dtype'" is displayed.

Possible Causes

The NumPy version of the training image is different from that in the notebook instance.

Solution

Print the NumPy version in the code and check whether the version is 1.18.5. If the version is not 1.18.5, run the following command at the beginning of the code:

```
import os
os.system('pip install numpy==1.18.5')
```

If the error persists, modify the preceding code as follows:

```
import os
os.system('pip install numpy==1.18.5')
os.system('pip install keras==2.6.0')
os.system('pip install tensorflow==2.6.0')
```

4.7.13 Error Message "No module name 'unidecode'" Displayed in Logs

Symptom

After the configuration file of the Tacotron 2 model downloaded from the master branch of MindSpore open-source Gitee is modified and then uploaded to

ModelArts for training, error message "No module name 'unidecode'" is displayed in logs.

Possible Causes

The Unidecode name of the **requirements.txt** file is incorrect, where **U** should be lowercase. As a result, the Unidecode module is not installed in the training job environment.

Solution

Change **Unidecode** in **requirements.txt** to **unidecode**.

Summary and Suggestions

Add the following line to the training code:

```
os.system('pip list')
```

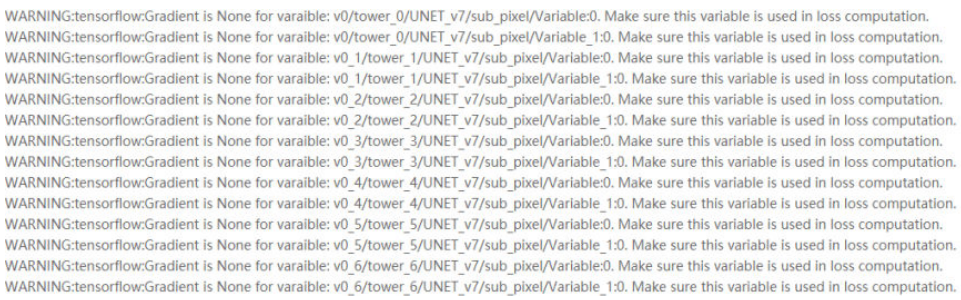
Run the training job and check whether the required module is available in logs.

4.7.14 Distributed Tensorflow Cannot Use tf.variable

Symptom

The following error occurs when **tf.variable** is used across multiple machines and multiple GPUs: **WARNING:tensorflow:Gradient is None for variable:v0/tower_0/UNET_v7/sub_pixel/Variable:0.Make sure this variable is used in loss computation**

Figure 4-20 Distributed Tensorflow unavailable



Possible Cause

Distributed TensorFlow needs to use **tf.get_variable** instead of **tf.variable**.

Solution

Replace **tf.variable** in the boot file with **tf.get_variable**.

4.7.15 When MXNet Creates kvstore, the Program Is Blocked and No Error Is Reported

Symptom

When **kv_store = mxnet.kv.create('dist_async')** is used to create **kvstore**, the program is blocked. For example, run the following code. If **end** is not displayed, the program is blocked.

```
print('start')
kv_store = mxnet.kv.create('dist_async')
print('end')
```

Possible Cause

The possible cause of a worker block is that the server cannot be connected.

Solution

Place the following code before **import mxnet** in **Boot File** to check the communication status between nodes. In addition, ps can be resent.

```
import os
os.environ['PS_VERBOSE'] = '2'
os.environ['PS_RESEND'] = '1'
```

In the preceding code, **os.environ['PS_VERBOSE'] = '2'** indicates that all communication information is printed. **os.environ['PS_RESEND'] = '1'** indicates that the Van instance resends the message if it does not receive the ACK message within the milliseconds set by **PS_RESEND_TIMEOUT**.

4.7.16 ECC Error Occurs in the Log, Causing Training Job Failure

Symptom

The following error occurs during the running of the training job log:
RuntimeError: CUDA error: uncorrectable ECC error encountered

Possible Cause

ECC errors

Solution

If there are more than 64 ECC errors, the system automatically isolates the faulty nodes. After the isolation, restart the training job to check whether the fault is rectified. If the training job fails again or is suspended due to an unisolated node, contact technical support.

4.7.17 Training Job Failed Because the Maximum Recursion Depth Is Exceeded

Symptom

An error occurs for a ModelArts training job.

```
RuntimeError: maximum recursion depth exceeded in __instancecheck__
```

Possible Causes

The training failed because the recursion depth exceeded the default recursion depth of Python.

Solution

If the maximum recursion depth is exceeded, increase the recursion depth in the boot file as follows:

```
import sys
sys.setrecursionlimit(1000000)
```

4.7.18 Training Using a Built-in Algorithm Failed Due to a bndbox Error

Symptom

When a training job is created using a built-in algorithm, the training failed with the following error message in the log:

```
KeyError: 'bndbox'
```

Possible Causes

Non-rectangles are used for labeling training sets. However, the built-in algorithm does not support datasets labeled by a non-rectangle.

Solution

This issue can be resolved in either of the following ways:

- Method 1: Use a common framework to develop a model that supports polygon-labeled datasets.
- Method 2: Use rectangles to label the datasets. Then, start the training job again.

4.7.19 Training Job Status Is Reviewing Job Initialization

Symptom

When **Algorithm Source** is set to **Custom** during training job creation, the training job status is **Reviewing Job Initialization**.

Possible Cause

When a custom image is running for the first time, the image needs to be reviewed first. After the image is reviewed, you can create a job. That is, the current status is **Reviewing Job Initialization**.

4.7.20 Training Job Process Exits Unexpectedly

Symptom

Running a training job failed, and error information similar to the following is displayed in logs:

```
[Modelarts Service Log]Training end with return code: 137
```

Possible Causes

According to the log, the exit code of the training job is 137. The training process starts after the user code is executed. Therefore, the exit code mentioned in this section is generated after the code for training job is executed. Common error codes include codes 247 and 139.

- Exit code: 137 or 247

  The possible cause is that the memory overflows. To resolve this issue, you can reduce the data volume, decrease the **batch_size** value, optimize the code, or aggregate and replicate the data.

  NOTE

  The size of data files is not equal to the memory usage. Therefore, evaluate the memory usage.

- Exit code: 139

  Check the version of the installation package. There may be a package conflict.

Troubleshooting

According to the error information, the error is caused by the user code.

You can use either of the following methods to locate the fault:

- Debug the code online (only available for the non-distributed code).
  a. Apply for a development environment instance with the same specifications in the development environment (notebook).
  b. Debug the user code in the notebook and find the improper code snippet.
  c. Find a solution by searching the key code snippet and exit code in a search engine.
- Locate the fault based on the training logs.
  a. Identify the improper code snippet based on the logs.
  b. Print the improper code snippet to obtain more detailed log information.
  c. Run the training job again to locate the improper code snippet.

4.7.21 Stopped Training Job Process

Symptom

The training job process is stopped and the logs are interrupted.

Possible Causes

- CPU soft lock

  The decompression of a large number of files may cause CPU soft lock and node restart. You can suspend the decompression for the specified amount of time by invoking sleep method when decompressing a large number of files. For example, every time 10,000 files are decompressed, the decompression stops for 1 second.
- Storage limitation

  Use data disks based on specifications. For details about a data disk size, see **What Are Sizes of the /cache Directories for Different Resource Specifications in the Training Environment?**
- CPU overload

  Reduce the number of threads.

Troubleshooting

According to the error information, the error is caused by the user code.

You can use either of the following methods to locate the fault:

- Debug the code online (only available for the non-distributed code).
  a. Apply for a development environment instance with the same specifications in the development environment (notebook).
  b. Debug the user code in the notebook and find the improper code snippet.
  c. Find a solution by searching the key code snippet and exit code in a search engine.
- Locate the fault based on the training logs.
  a. Identify the improper code snippet based on the logs.
  b. Print the improper code snippet to obtain more detailed log information.
  c. Run the training job again to locate the improper code snippet.

4.8 Training Job Suspended

4.8.1 Locating Training Job Suspension

Overview

A training job may be suspended due to unknown reasons. If the suspension cannot be detected promptly, resources cannot be released, leading to a waste. To minimize resource cost and improve user experience, ModelArts provides

suspension detection for training jobs. With this function, suspension can be automatically detected and displayed on the log details page. You can also enable notification so that you can be promptly notified of job suspension.

Detection Rules

Determine whether a job is suspended based on the monitored job process status and resource usage. A process is started to periodically monitor the changes of the two metrics.

- Job process status: If the process I/O of a training job changes, the next detection period starts. If the process I/O of the job remains unchanged in multiple detection periods, the resource usage detection starts.
- Resource usage: If the process I/O remains unchanged, the system collects the GPU usage within a certain period of time and determines whether the resource usage changes based on the variance and median of the GPU usage within the period. If the GPU usage is not changed, the job is suspended.

CAUTION

- Due to the limitation of detection rules, there is a certain error probability in suspension detection. If the suspension is caused by the logic of job code (for example, long-time sleep), ignore it.

Constraints

Suspension can be detected only for training jobs that run on GPUs.

Procedure

Suspension detection is automatically performed during job running. No additional configuration is required. After detecting that a job is suspended, the system displays a message on the training job details page, indicating that the job may be suspended. If you want to be notified of suspension (by SMS or email), enable event notification on the job creation page.

Cases

Common cases and solutions to training job suspension are as follows:

Data Replication Suspension

Suspension Before Training

Suspension During Training

Suspension in the Last Training Epoch

4.8.2 Data Replication Suspension

Symptom

The system stops responding when **mox.file.copy_parallel** is called to copy data.

Solution

- Run the following commands to copy files or folders:
```
import moxing as mox
mox.file.set_auth(is_secure=False)
```
- Run the following command to copy a single file that is greater than 5 GB:
```
from moxing.framework.file import file_io
```
Run **file_io._LARGE_FILE_METHOD** to check the version of the MoXing API. Output value **1** indicates V1 and **2** indicates V2.

Run **file_io._NUMBER_OF_PROCESSES=1** to resolve the issue for the V1 API.

To resolve the issue for the V2 API, run **file_io._LARGE_FILE_METHOD = 1** to switch to V1 and perform operations required in V1. Alternatively, run **file_io._LARGE_FILE_TASK_NUM=1** to resolve this issue.
- Run the following command to copy a folder:
```
mox.file.copy_parallel(threads=0,is_processing=False)
```

4.8.3 Suspension Before Training

If a job is trained on multiple nodes and suspension occurs before the job starts, add **os.environ["NCCL_DEBUG"] = "INFO"** to the code to view the NCCL debugging information.

Symptom 1

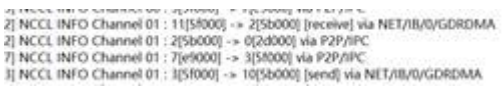
The job is suspended before the NCCL debugging information is displayed in logs.

Solution 1

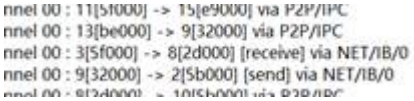
Check the code for parameters such as **master_ip** and **rank**. Ensure that these parameters are specified.

Symptom 2

The GDR information is displayed only on certain nodes of a multi-node training job.



The possible cause of the suspension is GDR.



Solution 2

Set **os.environ["NCCL_NET_GDR_LEVEL"] = '0'** at the beginning of the program or ask the O&M personnel to add the GDR information to the affected nodes.

Symptom 3

Communication information such as "Got completion with error 12, opcode 1, len 32478, vendor err 129" is displayed. The current network is unstable.

Solution 3

Add the following environment variables:

- **NCCL_IB_GID_INDEX=3**: enables RoCEv2. RoCEv1 is enabled by default. However, RoCEv1 does not support congestion control on switches, which may lead to packet loss. In addition, later-version switches do not support RoCEv1, leading to a RoCEv1 failure.
- **NCCL_IB_TC=128**: enables data packets to be transmitted through the queue 4 of switches, which is RoCE-compliant.
- **NCCL_IB_TIMEOUT=22**: enables a longer timeout interval. Generally, there is a network interruption lasting about 5s if the network is unstable and then the timeout message is returned. Change the timeout interval to 22s, indicating that the timeout message will be returned in about 20s (4.096 μs x 2 ^ timeout).

4.8.4 Suspension During Training

Symptom 1

According to the logs of the nodes on which a training job runs, an error occurred on a node but the job did not exit, leading to the job suspension.

Solution 1

Check the error cause and rectify the fault.

Symptom 2

The job is stuck in sync-batch-norm or the training speed is lowered down. If sync-batch-norm is enabled for PyTorch, the training speed is lowered down because all node data must be synchronized on each batch normalization layer in every iteration, which leads to heavy communication traffic.

```
from sync_batchnorm import SynchronizedBatchNorm1d, DataParallelWithCallback

sync_bn = SynchronizedBatchNorm1d(10, eps=1e-5, affine=False)
sync_bn = DataParallelWithCallback(sync_bn, device_ids=[0, 1])
```

Solution 2

Disable sync-batch-norm, or upgrade the PyTorch version to 1.10.

Symptom 3

The job is stuck in TensorBoard.

```
writer = SummaryWriter('./path/to/log')
```

Solution 3

Set a local path for storage, for example, **cache/tensorboard**. Do not store data in OBS.

Symptom 4

When PyTorch dataloader is used to read data, the job is stuck in data reading, and logs stop to update.

```
[05/16 12:01:54][INFO] logging.py: 95: json_stats: {"cur_iter": "161", "eta": "8:05:50", "split": "test_iter", "time_diff": 38.25511}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "161", "eta": "8:05:50", "split": "test_iter", "time_diff": 38.25510}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "161", "eta": "8:05:50", "split": "test_iter", "time_diff": 38.25495}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "161", "eta": "8:05:49", "split": "test_iter", "time_diff": 38.25407}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "161", "eta": "8:05:50", "split": "test_iter", "time_diff": 38.25510}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "161", "eta": "8:05:50", "split": "test_iter", "time_diff": 38.25511}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "161", "eta": "8:05:50", "split": "test_iter", "time_diff": 38.25519}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "162", "eta": "0:06:47", "split": "test_iter", "time_diff": 0.53575}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "162", "eta": "0:06:47", "split": "test_iter", "time_diff": 0.53553}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "162", "eta": "0:06:47", "split": "test_iter", "time_diff": 0.53554}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "162", "eta": "0:06:47", "split": "test_iter", "time_diff": 0.53556}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "162", "eta": "0:06:47", "split": "test_iter", "time_diff": 0.53553}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "162", "eta": "0:06:47", "split": "test_iter", "time_diff": 0.53555}
[05/16 12:01:54][INFO] logging.py: 95: json_stats: {"cur_iter": "162", "eta": "0:06:47", "split": "test_iter", "time_diff": 0.53562}
INFO:timesformer.utils.logging:json_stats: {"cur_iter": "162", "eta": "0:06:47", "split": "test_iter", "time_diff": 0.53548}
```

Solution 4

When using dataloader to read data, set **num_work** to a small value.

```
from torch.utils.data import DataLoader

train_loader = DataLoader(dataset=train_data, batch_size=train_bs, shuffle=True, num_worker=4)

valid_loader = DataLoader(dataset=valid_data, batch_size=valid_bs, num_worker=4)
```

4.8.5 Suspension in the Last Training Epoch

Symptom

Logs showed that an error occurred in split data. As a result, processes are in different epochs, and uncompleted processes are suspended because they do not receive response from other processes. As shown in the following figure, some processes are in epoch 48 while others are in epoch 49 at the same time.

```
loss exit lane:0.12314446270465851
step loss is 0.29470521211624146
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:2 Epoch:[48][20384/all] Data Time 0.000(0.000) Net Time 0.705(0.890) Loss 0.3403(0.3792)LR 0.00021887
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:1 Epoch:[48][20384/all] Data Time 0.000(0.000) Net Time 0.705(0.891) Loss 0.3028(0.3466) LR 0.00021887
[2022-04-26 13:57:20,757][INFO][train_epoch]:Rank:4 Epoch:[49][20384/all] Data Time 0.000(0.147) Net Time 0.705(0.709) Loss 0.3364(0.3414)LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:3 Epoch:[49][20384/all] Data Time 0.000 (0.115) Net Time 0.706(0.814) Loss 0.3345(0.3418) LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:0 Epoch:[49][20384/all] Data Time 0.000(0.006) Net Time 0.704(0.885) Loss 0.2947(0.3566) LR 0.00021887
[2022-04-26 13:57:20,758][INFO][train_epoch]:Rank:7 Epoch:[49][20384/all] Data Time 0.001 (0.000) Net Time 0.706 (0.891) Loss 0.3782(0.3614) LR 0.00021887
```



```
[2022-04-26 13:57:20,759][INFO][train_epoch]:Rank:5 Epoch:[48][20384/all] Data Time 0.000(0.000) Net Time 0.706(0.891) Loss 0.5471(0.3642) LR 0.00021887
[2022-04-26 13:57:20,763][INFO][train_epoch]:Rank:6 Epoch:[49][20384/all] Data Time 0.000(0.000) Net Time 0.704(0.891) Loss 0.2643(0.3390)LR 0.00021887
stage 1 loss 0.4600560665130615 mul_cls_loss loss:0.01245919056236744 mul_offset_loss 0.44759687781333923 origin stage2_loss 0.048592399805784225
stage 1 loss:0.4600560665130615 stage 2 loss:0.048592399805784225 loss exit lane:0.10233864188194275
```

Solution

Split tensors to align data.

4.9 Running a Training Job Failed

4.9.1 Troubleshooting a Training Job Failure

Symptom

A training job is in **Failed** state.

Cause Analysis and Solution

- The error "MoxFileNotExistsException(resp, 'file or directory or bucket not found.')" is displayed in the training logs.
  - Cause: The **train_data_obs** directory is not found when MoXing copies files.
  - Solution: Correct the address of the **train_data_obs** directory and restart the training job.

  NOTICE

  Do not delete any objects from the OBS directory while MoXing is downloading them. This will cause the download to fail.

- The error **CUDA capability sm_80 is not compatible with the current PyTorch installation.The current PyTorch install supports CUDA capabilities sm_37 sm_50 sm_60 sm_70'** is displayed in the training logs.
  - Cause: The CUDA version of the image used by the training job supports only the sm_37, sm_50, sm_60, and sm_70 accelerator cards. The sm_80 accelerator card is not supported.
  - Solution: Use a custom image to create a training job and install the target CUDA and PyTorch versions.
- The error "ERROR:root:label_map.pbtxt cannot be found. It will take a long time to open every annotation files to generate a tmp label_map.pbtxt." is displayed in the training logs.
  - If you use an algorithm that you subscribed to from AI Gallery, make sure the data label is accurate.
  - If you use an object detection algorithm, make sure the label box of the data is non-rectangular.

NOTE

Object detection algorithms support only rectangular label boxes.

- The error "RuntimeError: The server socket has failed to listen on any local network address. The server socket has failed to bind to [::]:29500 (errno: 98 - Address already in use). The server socket has failed to bind to 0.0.0.0:29500 (errno: 98 - Address already in use)." is displayed in the training logs.
  - Cause: The port number of the training job is not unique.
  - Solution: Change the port number in the code and restart the training job.
- The error "WARNING: root: Retry=7, Wait=0.4, Times tamp=1697620658.6282516" is displayed in the training logs.
  - Cause: The MoXing version is too old.
  - Solution: Contact technical support engineers to upgrade MoXing to 2.1.6 or later.

4.9.2 An NCCL Error Occurs When a Training Job Fails to Be Executed

Symptom

The training job fails to be executed. The training job logs contain NCCL-related errors, such as "NCCL timeout", "RuntimeError: NCCL communicator was aborted on rank 7", "NCCL WARN Bootstrap: no socket interface found", and "NCCL INFO Call to connect returned Connection refused, retrying".

Possible Causes

NCCL is a library that provides primitives for communication between GPUs. It implements collective communication and point-to-point send/receive primitives. If a training job reports an NCCL error, you can adjust the NCCL environment variables to solve the problem.

Solution

1. Go to the details page of the training job, click the **Logs** tab, and view the NCCL error.
   - If the error message **NCCL timeout** or **RuntimeError: NCCL communicator was aborted on rank 7** is displayed, InfiniBand Verbs times out. Click **Rebuild** in the upper right corner to create a training job again. Set the environment variable **NCCL_IB_TIMEOUT** to **22**. Submit the training job and wait until the job is completed.
   - If the error message **NCCL WARN Bootstrap : no socket interface found** or **NCCL INFO Call to connect returned Connection refused, retrying** is displayed, NCCL cannot find the communication network adapter or access the IP address. Check whether the **NCCL_SOCKET_IFNAME** environment variable is set in the training code. This environment variable is automatically injected by the system and does not need to be set in the training code. After the **NCCL_SOCKET_IFNAME** environment variable is removed from the training code, click **Rebuild** in the upper

right corner to create a training job again. After the training job is submitted, wait until the job is completed.

2. Wait and check whether the status of the training job changes to **Completed**.
   - If yes, no further action is required.
   - If no, contact technical support to check the node status.

Summary and Suggestions

- The **NCCL_SOCKET_IFNAME** environment variable is used to specify the name of the network adapter for communication. **NCCL_SOCKET_IFNAME=eth0** means that only the eth0 network adapter is used for communication. This environment variable is automatically injected by the system. Because the name of the communication network adapter is not fixed, this environment variable should not be set by default in the training code.
- The **NCCL_IB_TIMEOUT** environment variable is used to control InfiniBand Verbs timeout. The default value used by NCCL is **18**. The value ranges from 1 to 22.

4.9.3 Troubleshooting Process

Symptom

A training job using a custom image failed.

Locating Method

1. Determine the image source.
   - Check whether the base image of the custom image is from ModelArts. Use a base image provided by ModelArts to create a custom image. For details, see **Using a Base Image to Create a Training Image**.
   - If the image is from a third party, check with the creator of the custom image for how to use this image.
2. Determine the size of the custom image.

   Do not use a custom image larger than 15 GB. The size should not exceed half of the container engine space of the resource pool. Otherwise, the start time of the training job is affected.

   The container engine space of ModelArts public resource pool is 50 GB. By default, the container engine space of the dedicated resource pool is also 50 GB. You can customize the container engine space when creating a dedicated resource pool.
3. Determine the error type.
   - If an error message is displayed indicating that a file could not be found, see **Error Message "No such file or directory" Displayed in Training Job Logs**.
   - If an error message is displayed indicating that a package could not be found, see **Error Message "No module named .*" Displayed in Training Job Logs**.
   - An error occurred in the Ascend startup script or initialization script.

Check whether the script is obtained from the official website and whether the script is used strictly following the instructions provided in official documents. For example, check whether the script name and path are correct.

- The driver version is incompatible with the underlying driver.

  Before upgrading the driver of a custom image, check whether the upgraded version is supported by the underlying driver. **Obtain the supported driver versions**.

- You are not allowed to access a file.

  The possible cause is that the user of the custom image is different from that of the job container. In this case, modify the Dockerfile.

  ```
  RUN if id -u ma-user > /dev/null 2>&1 ; \
  then echo 'The ModelArts user already exists.' ; \
  else echo 'The ModelArts user does not exist.' && \
  groupadd ma-group -g 1000 && \
  useradd -d /home/ma-user -m -u 1000 -g 1000 -s /bin/bash ma-user ; fi && \
  chmod 770 /home/ma-user && \
  chmod 770 /root && \
  usermod -a -G root ma-user
  ```

- For other issues, search for solutions in **training failure cases**.

Summary and Suggestions

Before using a custom image for training jobs, create the image by following the **custom image specifications**. which also provides end-to-end examples for your reference.

4.9.4 A Training Job Created Using a Custom Image Is Always in the Running State

Symptom

A training job created using a custom image is always in the running state.

Cause Analysis and Solution

The log message below indicates that the CPU architecture of the custom image does not match that of the resource pool node.

```
standard_init_linux.go:215: exec user process caused "exec format error"
libcontainer: container start initialization failed: standard_init_linux.go:215: exec user process caused "exec format error"
```

This usually happens when the resource type and specifications are incorrectly set during job creation. For example, a custom image that uses the Arm CPU architecture should have NPU specifications, but x86 CPU or x86 GPU specifications are chosen instead.

4.9.5 Failed to Find the Boot File When a Training Job Is Created Using a Custom Image

Symptom

When a custom image is used to create a training job, error message "no such file or directory" is displayed.

Possible Causes

The directory of the boot file for running the command is incorrect.

Solution

Perform the following operations to check whether the boot file directory is correct:

When using a custom image to create a training job on ModelArts, set **Algorithm Type** to **Custom algorithm** and **Boot Mode** to **Custom image**.

If the OBS path to the boot script is **obs://bucket-name/app/code/train.py**, set the code directory to **/bucket-name/app/code/** when creating a job. After the code directory is set, run the following command so that the selected **code** folder can be downloaded to the **/home/ma-user/modelarts/user-job-dir** directory of the training container:

```
bash /home/ma-user/modelarts/user-job-dir/run_train.sh  # Training command (using custom images)
```

Run the following command:

```
bash /home/ma-user/modelarts/user-job-dir/run_train.sh python /home/ma-user/modelarts/user-job-dir/
code/train.py {python_file_parameter}  # Training command (using custom images)
```

4.9.6 Running a Job Failed Due to Persistently Rising Memory Usage

Symptom

A training job is in the **Failed** state.

Possible Causes

The memory usage continues to rise, leading to the training job failure.

Solution

1. View the logs and monitoring data of the training job to check whether there are any OOM errors.
   - If yes, go to **2**.
   - If there are no OOM errors but the monitoring metrics show anomalies, go to **3**.
2. Check whether there is any code in the training script that keeps using resources and prevents them from being allocated efficiently.

- If yes, optimize the code and wait until the job runs properly.
- If no, either upgrade the resource specifications allocated to the training job or contact technical support.

3. Restart the training job. Use CloudShell to log in to the training container to check the memory metrics and see if the memory usage spikes.
   - If yes, check the training job logs generated when the memory usage spikes and improve the relevant code logic to lower the memory consumption.
   - If no, either upgrade the resource specifications allocated to the training job or contact technical support.

4.10 Training Jobs Created in a Dedicated Resource Pool

4.10.1 No Cloud Storage Name or Mount Path Displayed on the Page for Creating a Training Job

Symptom

On the page for creating a training job, there is no option for the cloud storage and mount path.

Possible Causes

The network of the target dedicated resource pool is not connected, or no SFS has been created.

Solution

In the dedicated resource pool list, click the ID or name of the target resource pool to go to its details page. Click **Configure NAS VPC** in the upper right corner to check whether NAS VPC has been enabled. If the NAS VPC name and NAS subnet ID on the details page are left blank, NAS VPC is not enabled. In this case, enable NAS VPC.

If an error message is displayed after you attempt to enable it, the possible cause is that a VPC peering connection has been created for the VPC. In this case, delete the VPC peering connection and try again.

4.10.2 Storage Volume Failed to Be Mounted to the Pod During Training Job Creation

Symptom

The training job remains in the **Creating** state. When you check the events of the training job, error message "Unable to mount volumes for pod xxx ... list of unmounted volumes=[nfs-x]" is displayed.

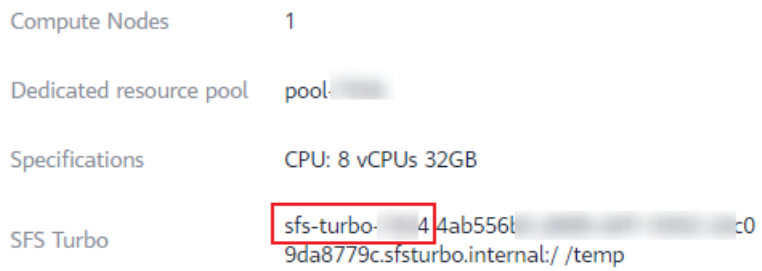
Possible Cause

For your SFS Turbo file system to function correctly, it must reside within a VPC network that is interconnected with the network of the dedicated resource pool. This connection is essential to ensure that the SFS can be successfully mounted to any training job executed within the dedicated resource pool. Disconnected network may lead to mounting failure.

Procedure

1. Go to the training job details page and obtain the SFS Turbo name.

   Figure 4-21 Obtaining SFS Turbo name

2. Log in to the SFS console, locate the SFS Turbo mounted to the training job, and click it to go to the details page. Obtain the VPC, security group, and endpoint information.
   - VPC: value of **VPC**
   - Security group: value of **Security Group**
   - Endpoint: value of **Shared Path** excludes ":/", for example, the shared path is **4ab556b5-d689-44f1-9302-24c09daxxxxc.sfsturbo.internal:/**, then the SFS Turbo endpoint is 4ab556b5-d689-44f1-9302-24c09daxxxxc.sfsturbo.internal.
3. Check whether the VPC CIDR block meets the following requirements:

   Requirement 1: To prevent CIDR block conflicts with the dedicated resource pool, the SFS Turbo CIDR block cannot overlap with 192.168.20.0/24 (default CIDR block of the dedicated resource pool). Go to the resource pool details page and check **Network** to obtain the actual CIDR block of the dedicated resource pool.

   Requirement 2: To prevent network conflicts with the container, the SFS Turbo CIDR block cannot overlap with 172 CIDR block (used by the container network).
   - If the requirements are not met, modify the VPC CIDR block of SFS Turbo. The recommended value is 10.*X.X.X*. For details, see **Modifying the CIDR Block of a VPC**.
   - If the requirements are met, go to the next step.
4. Check whether the VPC CIDR block of SFS Turbo is limited by a security group rule.

   Create a training job in the selected dedicated resource pool without mounting SFS Turbo. Once the job is in the **Running** state, access the **worker-0** instance via Cloud Shell. Execute the command **curl {sfs-turbo-**

endpoint}:{port} to verify if the ports are open. The ports that SFS Turbo requires for inbound traffic are 111, 445, 2049, 2051, 2052, and 20048. For details, see **Security Group** in **Create a File System**. For details about how to use Cloud Shell, see **Logging In to a Training Container Using Cloud Shell**.

- If yes, modify the security group configurations. For details, see **Modifying a Security Group Rule**.
- If there is no such a security group rule, perform the following steps.

5. Check whether SFS Turbo is normal.

   Create an ECS that uses the same CIDR block as SFS Turbo and mount the SFS Turbo to the ECS. If mounting failed, SFS Turbo is abnormal.

   a. If SFS Turbo is abnormal, contact SFS technical support.
   b. If SFS Turbo is normal, contact ModelArts technical support.

4.11 Training Performance Issues

4.11.1 Training Performance Deteriorated

Symptom

When a ModelArts algorithm is used for training, it will take more time than expected for training.

Possible Causes

The possible causes are as follows:

1. The job code or training parameters have been modified.
2. The GPU hardware for training malfunctions.

Solution

1. Check whether the training code and parameters have been modified.
2. Check whether the allocation of the CPU, memory, GPU, snt9, or Infiniband resources complies with the expectation.
3. Use CloudShell to log in to the Linux and check the GPU working status.
   - Run the **nvidia-smi** command to check whether the GPU is working properly.
   - Run the **nvidia-smi -q -d TEMPERATURE** command to check the temperature. If the temperature is too high, the training performance deteriorates.

5 Inference Deployment

5.1 AI Application Management

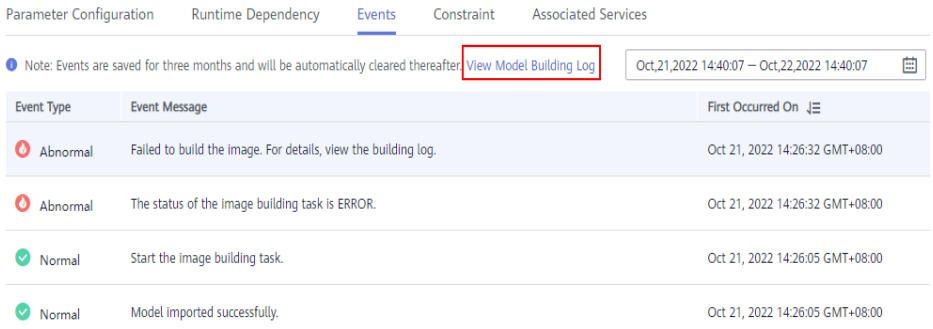
5.1.1 Creating an AI Application Failed

Fault Locating and Troubleshooting

There are two cases of an AI application creation failure: An error occurred during the AI application creation or API calling; the command for creating an AI application was successfully issued, but the creation failed.

1. For case 1, the issue is generally caused by invalid input parameters. In this case, rectify the fault as prompted.
2. For case 2, do as follows to rectify the fault:
   - On the AI application details page, view the events on the **Events** tab page. Analyze the failure cause based on the events and rectify the fault.
   - If the AI application is in the state of a building failure, click **View Model Building Log** on the **Events** tab page on the AI application details page. The building log provides details about the failure. Rectify the fault based on the cause.

Figure 5-1 View Model Building Log



Common Issues

1. Dockerfiles are not allowed in a model file directory.

   According to model building logs, "Not only a Dockerfile in your OBS path, please make sure, The dockerfile list" is displayed, indicating that the file directory is incorrect and that the file should be removed from the directory.

   Figure 5-2 Error message for an incorrect Dockerfile directory

2. The pip software package version is different from the version recorded in logs.

Figure 5-3 Incorrect pip software package version

3. Error message "exec /usr/bin/sh: exec format error" is displayed in model building logs.

   This issue is generally due to the inconsistency between the used system engine and the system engine for creating the image. For example, an x86 image is used but it is displayed as Arm.

   View the configured system engine on the AI application details page.

5.1.2 Suspended Account or Insufficient Permission to Import AI Applications

Symptom

When an AI application is imported, the system displays a message, indicating that the account has been suspended.

Possible Causes

Possible causes are as follows:

1. The account is frozen due to arrears.
2. The account does not have the permission to access the target workspace.
3. The operation is performed by an IAM user, who has not been granted with AI application permissions from the tenant account.

NOTICE

For details, see **Permissions Policies and Supported Actions**.

Solution

1. If the account is frozen due to arrears, top up the account and wait until the account is unfrozen.
2. If the issue is due to insufficient permission, grant the permission of importing AI applications to the account. For details, see **Creating a Custom Policy**.

5.1.3 Failed to Build an Image or Import a File When an IAM user Creates an AI Application

Symptom

- When an IAM user creates an AI application, creating an image failed. The failure log indicates that downloading the OBS file failed.

```
#============================== download_obs_file ==============================
Successfully to download file cnnorth7-infer-model1/2d3d3591-1aaa-49e8-af22-483d0a6b31bc/Dockerfile from OBS
Successfully to download file cnnorth7-infer-model1/2d3d3591-1aaa-49e8-af22-483d0a6b31bc/model/config.json from
OBS
Get object size from OBS failed! errorCode: None, errorMsg: None
Download directory from OBS failed! Exception: (None, None)
Download file from OBS failed! Exception: ('Get object size from OBS failed! ', OBSException(None, None))
Download directory from OBS failed! Exception: ('Download file from OBS failed! ', Exception('Get object size
from OBS failed! ', OBSException(None, None)))
Retry for the 1 th times
```

- When an IAM user creates an AI application, either of the following prompts are displayed: **Failed to copy model file due to obs exception. Please Check your obs access right.** and **User %s does not have obs:object:PutObjectAcl permission**. The AI application fails to be created due to OBS import exceptions or permission issues.

Possible Causes

Using ModelArts requires OBS authorization. ModelArts users require OBS system permissions. The IAM permissions of an IAM user are configured by their tenants. If a tenant does not grant the OBS **putObjectAcl** permission to their IAM users, this issue occurs.

Solution

NOTE

For details about how to create a custom policy for OBS permissions on which ModelArts depends, see **Example Custom Policies of OBS**.

Assign custom policy permissions to the target user on the IAM console. For details, see **Creating a Custom Policy**.

1. Log in to the IAM console, choose **Permissions** > **Policies/Roles**, and click **Create Custom Policy** in the upper right corner to create a custom policy.

Figure 5-4 Adding permissions on IAM

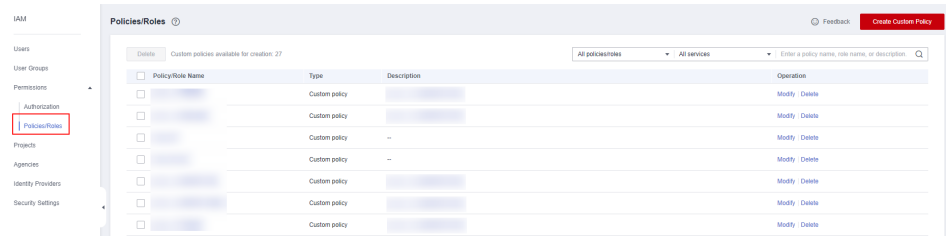
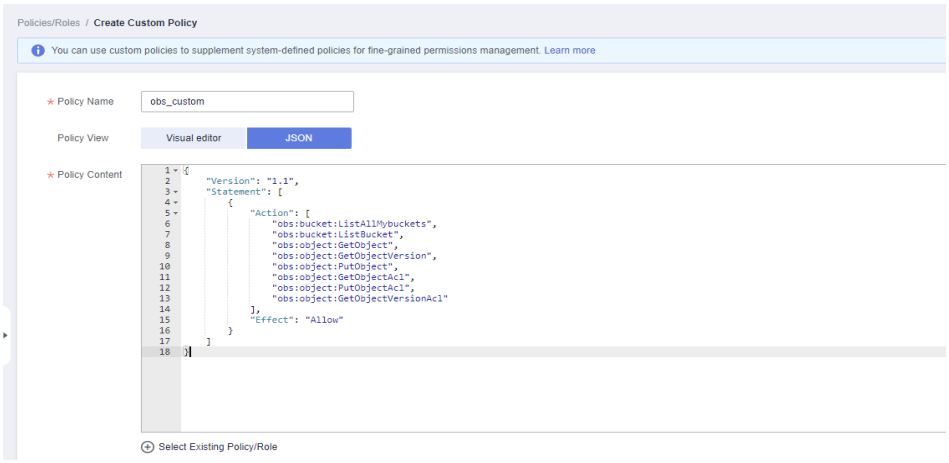


Figure 5-5 Creating a custom policy

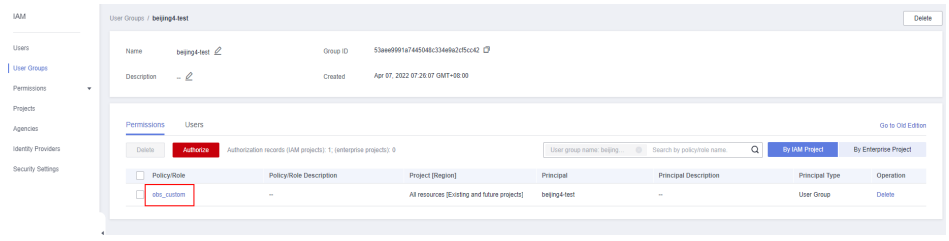


An example custom policy is as follows:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "obs:bucket:ListAllMybuckets",
                "obs:bucket:ListBucket",
                "obs:object:GetObject",
                "obs:object:GetObjectVersion",
                "obs:object:PutObject",
                "obs:object:GetObjectAcl",
                "obs:object:PutObjectAcl",
                "obs:object:GetObjectVersionAcl"
            ],
            "Effect": "Allow"
        }
    ]
}
```

2. Assign custom policy permissions to the user group to which the IAM user belongs.

Figure 5-6 Assigning permissions to an IAM user



5.1.4 Obtaining the Directory Structure in the Target Image When Importing an AI Application Through OBS

Symptom

When I create an AI application, customized files and folders are stored in the OBS directory specified by a meta model source, and these files and folders will be copied to the target image. What is the path to the copied files and folders?

Possible Causes

When an AI application is imported through OBS, ModelArts copies all files and folders in the specified OBS directory to a path specified in the image. You can obtain the path in the image by using **self.model_path**.

Solution

For details about how to obtain the path in an image, see **Specifications for Model Inference Coding**.

5.1.5 Failed to Obtain Certain Logs on the ModelArts Log Query Page

Symptom

I used a base image to import AI applications through OBS and wrote some inference code for implementing the inference logic. After an error occurred, I attempted to use the fault logs to locate the fault. However, certain logs were not displayed on the log query page in ModelArts.

Possible Causes

To display the logs of an inference service, print the logs on the console through coding. Python logging used by inference base images allows the display of only warning logs. To display INFO logs, set the log level to INFO in the code.

Solution

In the PY file for the inference code, set the default level of logs output to the console to **INFO**. The example code is as follows:

```
import logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
```

5.1.6 Failed to Download a pip Package When an AI Application Is Created Using OBS

Symptom

Creating an AI application using OBS failed. Logs showed that downloading the pip package failed, for example, downloading the NumPy 1.16 package failed.

Possible Causes

Possible causes are as follows:

1. The package is not available in the pip source. The default pip source is pypi.org. Check whether the package of the target version is available in pypi.org and check the package installation restrictions.
2. The downloaded package does not match the architecture in the base image. For example, an x86 package is downloaded for Arm, or a Python 3 package

is downloaded for Python 2. For details about the runtime environment of a base image, see **Available Inference Base Images**.

3. The sequence of configuring package dependencies is incorrect.

Solution

1. Log in to pypi.org and check whether the required installation package is available. If the package is unavailable, use the WHL package and place it into the OBS directory where the model is stored.
2. Check whether the installation restrictions and dependencies of the package are met.
3. If there are package dependencies, configure the dependencies in a correct sequence. For details, see **How Do I Edit the Installation Package Dependency Parameters in a Model Configuration File When Importing a Model?**

5.1.7 Failed to Use a Custom Image to Create an AI application

Symptom

When I used a custom image to create an AI application, the creation failed.

Possible Causes

Possible causes are as follows:

- The URL of the image used for importing the AI application is invalid or the image is unavailable.
- SWR operation permissions are not included in the agency authorization configured on ModelArts.
- The IAM user does not obtain SWR operation permissions from the tenant.
- The image used is from another account.
- The image used is a public image.

Solution

1. Go to the SWR console and check whether the target image is available and whether the URL of the image is the same as the actual one, including the spelling and letter cases in the URL.
2. Check whether SWR operation permissions are included in the agency authorization configured on ModelArts. To do so, go to the **Global Configuration** page on ModelArts and view the authorization details. If no SWR operation permissions are configured, go to the IAM console and grant the permissions to the target agency.

Figure 5-7 Global Configuration

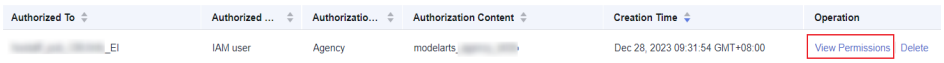


Figure 5-8 Entrance to permissions modification in IAM

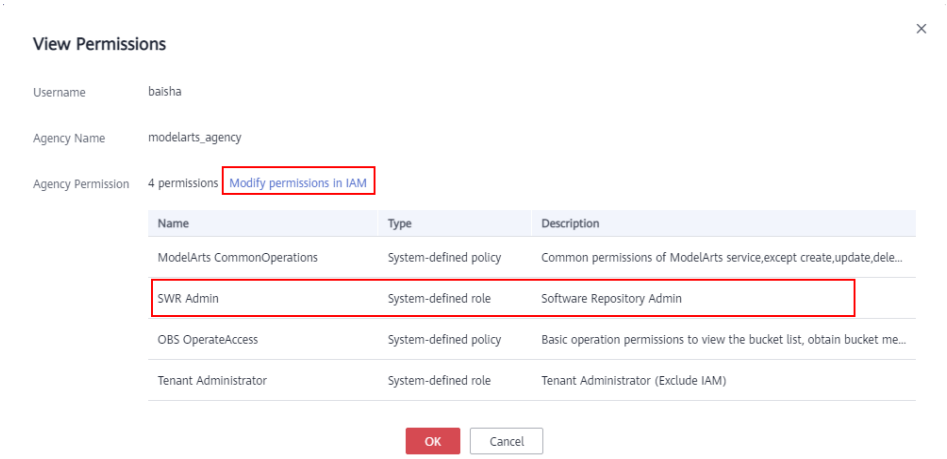
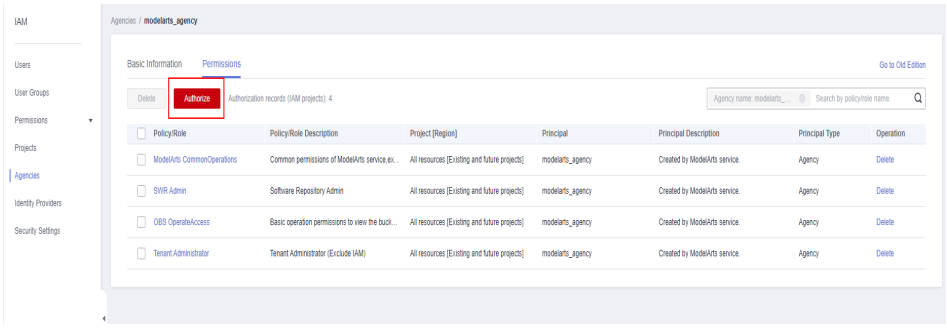


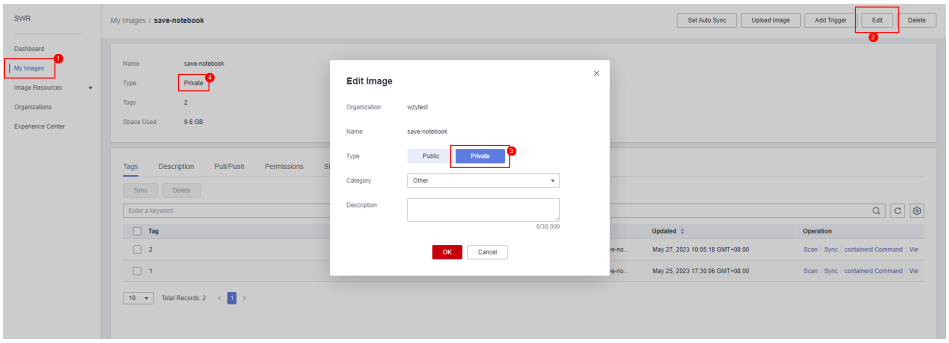
Figure 5-9 Authorizing an agency



3. Set a private image

   Log in to SWR, choose **My Images** in the navigation pane on the left to view image details. Click **Edit** in the upper right corner and set **Type** to **Private**.

Figure 5-10 Changing the image type to private



5.1.8 Insufficient Disk Space Is Displayed When a Service Is Deployed After an AI Application Is Imported

Symptom

After an AI application is imported, message "No space left on device" is displayed during service deployment.

Possible Causes

ModelArts uses containers to deploy services. There are size limitations for containers to run. If the size of your model file, custom file, or system file exceeds the Docker size, a message will be displayed, indicating that the image space is insufficient.

Solution

The maximum Docker size for a container in a public resource pool is 10 GB, and that for a container in a dedicated resource pool is 30 GB.

If the AI application is imported from OBS or a training job, the total size of the base image, model files, code, data files, and software packages cannot exceed the limit.

If the AI application is imported from a custom image, the total size of the decompressed image and image dependencies cannot exceed the limit.

5.1.9 Error Occurred When a Created AI Application Is Deployed as a Service

Symptom

After an AI application is created, an error occurred when it is deployed as a service.

Possible Causes

When an AI application is imported using a custom or base image, many service logics are customized. Any error in the logics will result in a service deployment or prediction failure.

Solution

1. After deploying a service failed, go to the service details page and view deployment logs to identify the failure cause. (Ensure that standard input and output functions are used for code output. Otherwise, the output will not be displayed on the ModelArts console.) Find the code based on the error in the logs to locate the fault.

5.1.10 Invalid Runtime Dependency Configured in an Imported Custom Image

Symptom

When a custom image is imported through an API to create an AI application, the runtime dependency is configured, but the pip dependency package is not properly installed.

Possible Causes

An imported custom image does not support the runtime dependency. The system does not automatically install the required pip dependency package.

Solution

Create a custom image again.

Install the pip dependency package (for example, the Flask dependency package) in the Dockerfile file that is used to create the image.

```
# Configure the Huawei Cloud source and install Python, Python3-PIP, and Flask.
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak && \
sed -i "s@http://.*security.ubuntu.com@http://repo.huaweicloud.comxxx@g" /etc/apt/sources.list && \
sed -i "s@http://.*archive.ubuntu.com@http://repo.huaweicloud.comxxx@g" /etc/apt/sources.list && \
apt-get update && \
apt-get install -y python3 python3-pip && \
pip3 install  --trusted-host https://repo.huaweicloud.comxxx -i https://repo.huaweicloud.comxxx/repository/pypi/simple  Flask
```

5.1.11 Garbled Characters Displayed in an AI Application Name Returned When AI Application Details Are Obtained Through an API

Symptom

When details about an AI application are obtained through an API, garbled characters are displayed in a returned AI application name (**model_name**). For example, the AI application name (**model_name**) is **query_vec_recall_model**, but the name returned from the API is **query_vec_recall_model_b**.

Figure 5-11 Garbled characters in an AI application name

Possible Causes

If an AI application name contains underscores (_), these characters must be escaped.

Solution

Add the **exact_match** parameter to the request and set the parameter value to **true** to ensure that the returned value of **model_name** is correct.

5.1.12 The Model or Image Exceeded the Size Limit for AI Application Import

Symptom

When an AI application is imported, a prompt says that the model or image exceeds the limit.

Possible Causes

If the AI application is imported using OBS or training, the total size of the basic image, model files, code, data files, and downloaded software packages exceeds the limit.

If the AI application is imported using a custom image, the total size of the decompressed image and image dependencies exceeds the limit.

Solution

Downsize the model or image and import the AI application again.

5.1.13 A Single Model File Exceeded the Size Limit (5 GB) for AI Application Import

Symptom

When an AI application is imported, a prompt says that a single model file exceeded the size limit (5 GB).

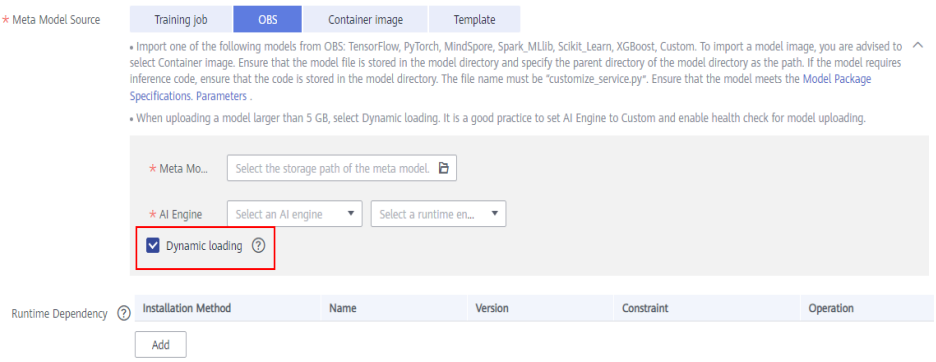
Possible Causes

If dynamic loading is not used, a single model file cannot exceed 5 GB. Otherwise, the AI application fails to be imported.

Solution

- Downsize the model file and import the AI application again.
- Use the dynamic loading function to import the AI application.

Figure 5-12 Using dynamic loading

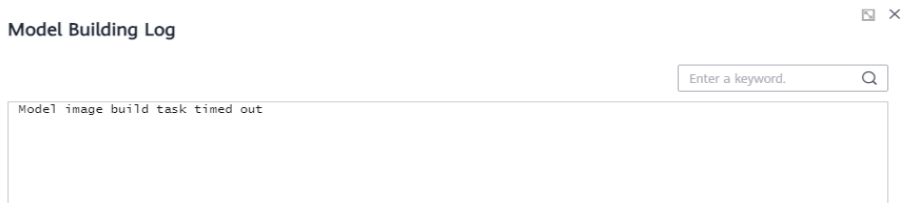


5.1.14 Creating an AI Application Failed Due to Image Building Timeout

Symptom

The AI application fails to be created. A message is displayed showing "Model image build task timed out" , and no detailed build log is generated.

Figure 5-13 Building the model image timed out



Possible Cause

ImagePacker has a timeout limit when building images. The default value is 30 minutes (which may vary in different regions). If building a model image times out, the building task will fail. In this case, the message "Model image build task timed out" is displayed, and no detailed build log is generated.

Solution

- Prepare the dependency packages to be downloaded and built beforehand to save time. You can install the running environment dependency using an offline wheel package. When installing the offline wheel package, ensure that the wheel package and model file are stored in the same directory.
- Optimize the model code to improve the efficiency of building model images.

5.2 Service Deployment

5.2.1 Error Occurred When a Custom Image Model Is Deployed as a Real-Time Service

Symptom

A model fails to be deployed as a real-time service. On the real-time service details page, the message "failed to pull image, retry later" is displayed on the **Events** tab page while no information is displayed on the **Logs** tab page.

Solution

This fault is typically caused by the excessive size of the model you have deployed. Do the following:

- Simplify the model, re-import it, and deploy it as a real-time service.

- Purchase a dedicated resource pool and use it to deploy the model as a real-time service.

5.2.2 Alarm Status of a Deployed Real-Time Service

Symptom

A deployed real-time service is in the **Alarm** state.

Solution

The prediction using a real-time service that is in the **Alarm** state may fail. Perform the following operations to locate the fault and deploy the service again:

1. Check whether there are too many prediction requests on the backend.

   If you call APIs for prediction, check whether there are too many prediction requests. A large number of prediction requests lead to the alarm state of the real-time service.
2. Check whether the service memory is functional.

   Check whether memory overflow or leakage occurs in the inference code.
3. Check whether the model is running properly.

   If the model fails, for example, the associated resources are faulty, check inference logs.
4. Check whether there is an abnormal amount of instance pods.

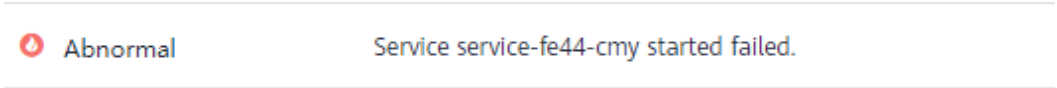
   If O&M engineers have deleted abnormal instance pods, the alarm "Service error. There are *XXX* abnormal instances." may occur in the event. Once the alarm is displayed, the service automatically starts a new normal instance to restore to the normal state. The process may take a while.

5.2.3 Failed to Start a Service

Symptom

After a service is started, the system displays a message, indicating a container startup failure.

Figure 5-14 Service startup failure



Possible Causes

Possible causes are as follows:

- **The AI application is faulty and cannot be started.**
- **The port configured in the image is incorrect.**
- **The health check is incorrectly configured.**
- **The model inference code customize_service.py is incorrectly edited.**

- **The image fails to be pulled.**
- **Scheduling failed due to insufficient resources.**

Faulty AI Application

If the image used for creating an AI application is faulty, recreate the image by following the instructions provided in **Creating a Custom Image and Using It to Create an AI Application**. Ensure the image can be started properly and the expected data can be returned through curl on the local host.

Incorrect Port in the Image

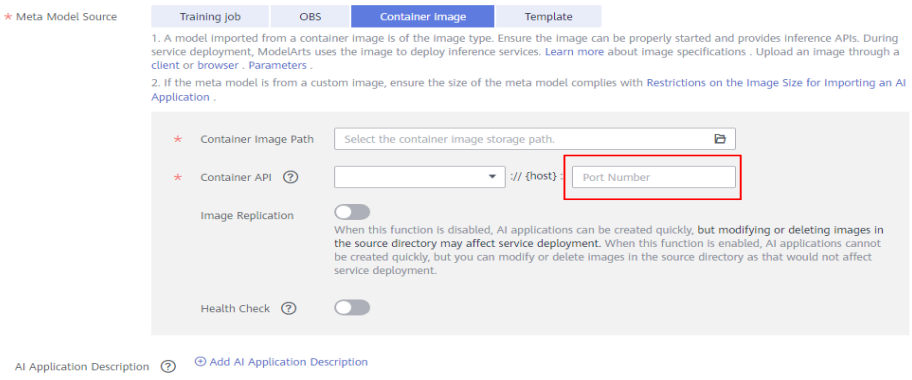
The port enabled in the image is not 8080, or the port enabled in the image is different from the port configured during AI application creation. As a result, the register-agent cannot communicate with the AI application during service deployment. After a certain period of time (20 minutes at most), it is considered that starting the AI application failed.

If this fault occurs, check the port enabled in the custom image code and the port configured during AI application creation. Ensure that the two ports are the same. If you do not specify a port during AI application creation, ModelArts will listen to port 8080 by default. In this case, the port enabled in the custom image code must be 8080.

Figure 5-15 Port enabled in the custom image code

```
# host must be "0.0.0.0", port must be 8080
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=8080)
```

Figure 5-16 Port configured during AI application creation



Incorrect Health Check Configuration

If health check is enabled in the image, perform the following operations to locate the fault:

- Check whether the health check port runs properly.

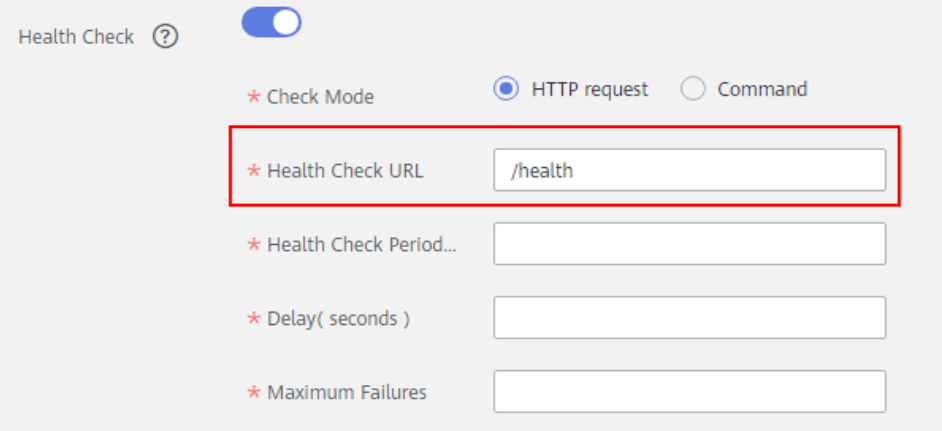
  If health check is enabled in a custom image, check whether the health check API is functional during image test. For details about how to test an image

locally, see **Building a Custom Image and Using It to Create an AI Application**.

- Check whether the health check address configured during AI application creation is the same as the actual one.

  If the AI application is created using a base image provided by ModelArts, the health check URL must be **/health** by default.

  Figure 5-17 Configuring the health check URL



Incorrect customize_service.py

Check service runtime logs to locate the fault and rectify it.

Pulling an Image Failed

If the service fails to be started and a message is displayed indicating that the image fails to be pulled, see **What Do I Do If an Image Fails to Be Pulled When a Service Is Deployed, Started, Upgraded, or Modified?**

Scheduling Failed Due To Insufficient Resources

The service fails to be started, and a message is displayed indicating that resources are insufficient and service scheduling fails. For details, see **What Do I Do If Resources Are Insufficient When a Service Is Deployed, Started, Upgraded, or Modified?**.

Insufficient Memory

The service fails to be started, and a message is displayed indicating that the memory is insufficient. For details, see **What Can I Do if the Memory Is Insufficient?**.

5.2.4 What Do I Do If an Image Fails to Be Pulled When a Service Is Deployed, Started, Upgraded, or Modified?

Possible Causes

The available disk space of the node is smaller than the image size.

Solution

1. Reduce the image size.
2. If the problem persists after the image size is reduced, contact the system administrator.

5.2.5 What Do I Do If an Image Restarts Repeatedly When a Service Is Deployed, Started, Upgraded, or Modified?

Possible Causes

There is a bug in the container image code.

Solution

Debug the container image code based on container logs, create the AI application again, and deploy the application as a real-time service.

5.2.6 What Do I Do If a Container Health Check Fails When a Service Is Deployed, Started, Upgraded, or Modified?

Possible Causes

Calling the container health check API failed. The possible causes are as follows:

- The health check is incorrectly configured for the image.
- The health check is incorrectly configured for the AI application.

Solution

Check container logs for the cause of the health check failure.

- If the health check is incorrectly configured for the image, debug the code, create an image again and then the AI application, and use the new AI application to deploy the service. For details about how to configure the image health API for an image, see parameter **health** in **Specifications for Writing the Model Configuration File**.
- If the health check is incorrectly configured for the AI application, create a new AI application or create a version of the existing AI application, correctly configure the health check, and use the new AI application or version to deploy the service. For details about the AI application health check, see parameter **Health Check** in **Creating and Importing a Model Image**.

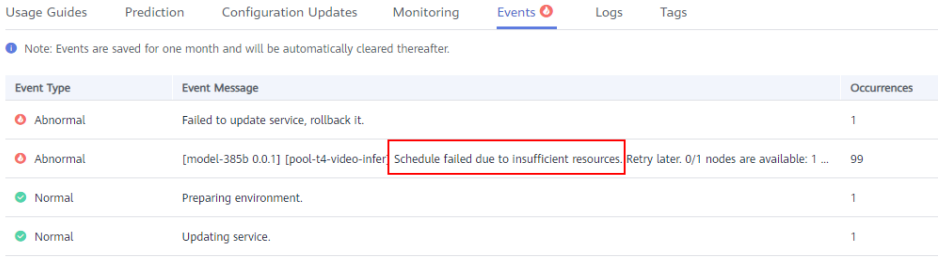
5.2.7 What Do I Do If Resources Are Insufficient When a Service Is Deployed, Started, Upgraded, or Modified?

Symptom

The service fails to be started, and an error message is displayed, indicating that resources are insufficient and service scheduling fails. ("Schedule failed due to

insufficient resources. Retry later." or "ModelArts.3976: No resources are available for the selected specification.")

Figure 5-18 Schedule failed due to insufficient resources



Possible Causes

- The configured instance specifications are beyond the remaining CPU or memory resources. ("insufficient CPU" / "insufficient memory")
- The disk capacity cannot meet the requirements of the model. ("x node(s) had taint {node.kubernetes.io/disk-pressure: }" / "No space")

Solution

When resources are insufficient, ModelArts retries for three times. If resources are released during these retries, the service can be successfully deployed.

If resources are still insufficient after three retries, the service deployment fails. In this case, perform the following operations to resolve this issue:

- If the service is to be deployed in a public resource pool, wait until other users release resources.
- If the service is to be deployed in a dedicated resource pool, select lower container specifications or custom specifications to deploy the service on the premise that the model requirements are met.
- Expand the capacity of the current resource pool before deploying the service. To expand the capacity of the public resource pool, contact the system administrator. To expand the capacity of the dedicated resource pool, refer to **Resizing a Resource Pool**.
- If the disk space is insufficient, try again to schedule the instance to another node. If the disk space of a single instance is still insufficient, contact the system administrator to use proper specifications.

  NOTE

  If an AI application imported though a large model is used to deploy the service, ensure that the disk space of the dedicated resource pool is greater than 1 TB (1000 GB).

5.2.8 Error Occurred When a CV2 Model Package Is Used to Deploy a Real-Time Service

Symptom

An error occurred when a CV2 model package is used to deploy a real-time service.

Possible Causes

When a meta model is imported from OBS, the service base image is used. However, the base image does not provide the SO data on which CV2 depends. Therefore, ModelArts does not support the import of CV2 model packages from OBS.

Solution

Use the CV2 model package to create a custom image, upload the custom image to SWR, import a meta model from the container image, and deploy a real-time service. For details about how to create a custom image, see **Creating a Custom Image and Using It to Create an AI Application**.

5.2.9 Service Is Consistently Being Deployed

Symptom

A service retains in the **Deploying** state. No obvious error is found in AI application logs.

Possible Causes

The AI application port is typically incorrect. Check whether the port for creating the AI application is correct.

Solution

Check the AI application port. If it is not configured, the default port 8080 is used. If you have changed the port number in the configuration file of the custom image, configure the correct port number when deploying the AI application.

For details, see **How Do I Change the Default Port to Create a Real-Time Service Using a Custom Image?**

5.2.10 A Started Service Is Intermittently in the Alarm State

Symptom

The traffic for prediction is not heavy, but the following error frequently occurs:

- Backend service internal error. Backend service read timed out
- Send the request from gateway to the service failed due to connection refused, please confirm your service is connectable

- Send the request from gateway to the service failed due to connection timeout, please confirm your service is able to process the new request

Possible Causes

After a prediction request is sent, the service stops and then starts.

Solution

Check the image used by the service, identify the cause of the service stop, and rectify the fault. Re-create the AI application and use it to deploy a service.

5.2.11 Failed to Deploy a Service and Error "No Module named XXX" Occurred

Symptom

Deploying a service failed. The system displays error message "No Module named XXX".

Possible Causes

"No Module named XXX" indicates that the dependency module is not imported to the model.

Solution

Import the required dependency module to the model through inference code.

For example, when you attempt to deploy a PyTorch AI application as a real-time service, the system displays error message "ModuleNotFoundError: No module named 'model_service.tfserving_model_service'". In this case, configure "from model_service.pytorch_model_service import PTServingBaseService" in **customize_service.py**. Example code:

```
import log
from model_service.pytorch_model_service import PTServingBaseService
```

5.2.12 Insufficient Permission to or Unavailable Input/Output OBS Path of a Batch Service

Symptom

1. An input/output path is unavailable, and the following error message is displayed:
   ```
   "error_code": "ModelArts.3551",
   "error_msg": "OBS path xxxx does not exist."
   ```
2. When the access to an input/output path is denied, the following error message is displayed:
   ```
   "error_code": "ModelArts.3567",
   "error_msg": "OBS error occurs because Access Denied."
   ```


Possible Causes

ModelArts.3551: The OBS path for data input or output does not exist.

ModelArts.3567: The OBS path for data input or output is available, but the current account does not have the permission to access the path.

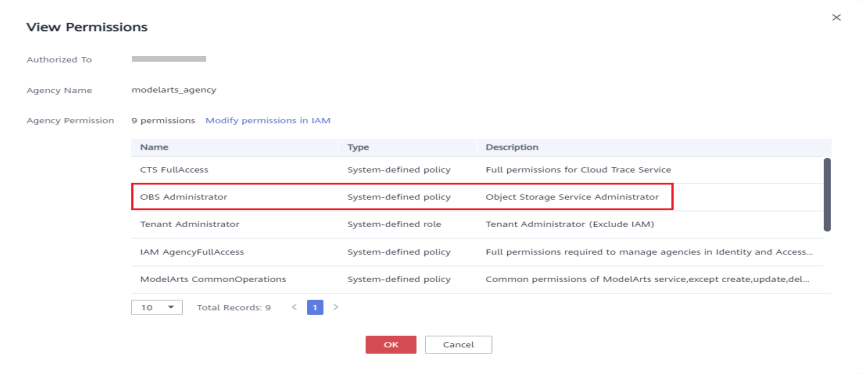
Solution

ModelArts.3551: Check whether the data input path is available in OBS. If not, create an OBS path as required. If the path is available but the error persists, submit a service ticket to apply for technical support.

ModelArts.3567: You can access only the OBS path under your own account. To read the OBS data of other users through ModelArts, configure an agency. Otherwise, the access is denied.

Log in to the ModelArts management console. In the navigation pane, choose **Settings**. Click **View Permissions** to check whether the OBS agency permission is configured.

Figure 5-19 Viewing permissions



If an agency already exists but the error persists, submit a service ticket for technical support.

5.2.13 Error "No CUDA runtime is found" Occurred When a Real-Time Service Is Deployed

Symptom

When a real-time service is deployed, the following error occurred: No CUDA runtime is found, using CUDA_HOME='/usr/local/cuda'.

Possible Causes

According to the error "No CUDA runtime is found" in logs, CUDA runtime was not found.

Solution

Perform the following operations:

1. Check whether a GPU flavor is selected for deploying the real-time service.
2. Add **os.system('nvcc -V)** into **customize_service.py** to view the CUDA version of the image. For details about how to write **customize_service.py**, see **Specifications for Writing Model Inference Code**.
3. Check whether the CUDA version matches the installed MMCV version.

NOTE

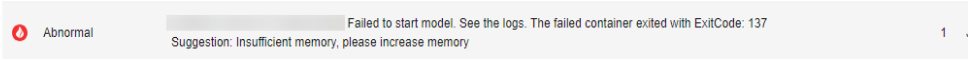
Selecting a GPU flavor if the model and inference script require GPUs.

5.2.14 What Can I Do if the Memory Is Insufficient?

Symptom

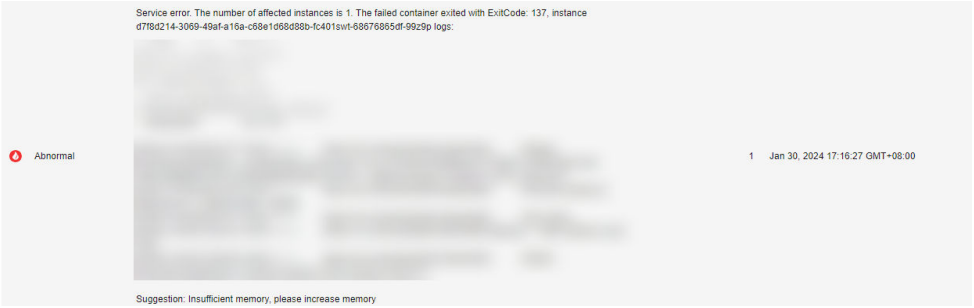
- The deployment or upgrade of a real-time service fails and information similar to the following is displayed in the event.

Figure 5-20 Example 1 of a message indicating insufficient memory



- An alarm is generated for a running service, and the following suggestion is displayed in the event: "Insufficient memory, please increase memory."

Figure 5-21 Example 2 of a message indicating insufficient memory



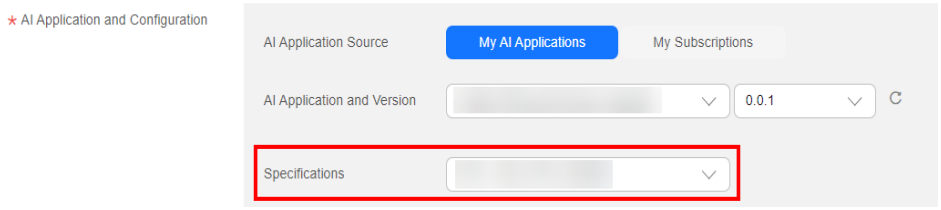
Possible Causes

- If this message is displayed during deployment or upgrade, the memory size of the chosen compute node is insufficient for the application deployment, and you need to increase the memory.
- If an alarm is generated for a running service, memory overflow occurs due to code problems, or the service usage is too large so the memory requirement increases.

Solution

- When deploying or upgrading a real-time service, select a compute node with larger memory.

Figure 5-22 Compute node specifications



- If an alarm is generated for a running service, check whether memory overflow occurs due to code problems, or more memory is required due to heavy service usage. If more memory is required, upgrade the real-time service and select a compute node with larger memory.

5.3 Service Prediction

5.3.1 Service Prediction Failed

Symptom

After a real-time service is deployed and running, an inference request is sent to the service, but the inference failed.

Cause Analysis and Solution

Service prediction involves multiple phases, including the client, Internet, APIG, dispatcher, and model service. A fault in any phase may lead to a prediction failure.

Figure 5-23 Prediction process

1. If an "APIG.XXXX" error occurs, the request is intercepted on API Gateway due to a fault.

   Rectify the fault by referring to **Error "APIG.XXXX" Occurred in a Prediction Failure**.

   The following shows the other cases in which a request is intercepted on API Gateway:

   - **Method Not Allowed**
   - **Request Timed Out**

2. If a "ModelArts.XXXXX" error occurs, the request is intercepted on the dispatcher due to a fault.

   Rectify the fault by referring to the methods provided in the following typical cases:

   - **Error ModelArts.4302 Occurred in Real-Time Service Prediction**
   - **Error ModelArts.4302 Occurred in Real-Time Service Prediction**
   - **Error ModelArts.4503 Occurred in Real-Time Service Prediction**

3. If an inference image is used and an "MR.XXXX" error occurs, the request has been sent to the model service, and the fault is generally due to a bug in model inference code.

   Identify the cause of the prediction failure based on the error information in logs, debug the model inference code, and import the model again for prediction.

   Rectify the fault by referring to **Error MR.0105 Occurred in Real-Time Service Prediction**.
4. In other cases, check whether the client and the Internet are accessible.
5. If the fault persists, contact the system administrator.

5.3.2 Error "APIG.XXXX" Occurred in a Prediction Failure

A request is intercepted on API Gateway due to a fault, and error "APIG.XXXX" occurs.

Rectify the fault by referring to the methods provided in the following typical cases:

- **APIG.0101 Incorrect Prediction URL**
- **APIG.0201 Request Body Oversized**
- **APIG.0301 Authentication Failed**
- **APIG.1009 Unmatched AppKey and AppSecret**

For more details about API Gateway error codes and solutions, see **API Error Codes**.

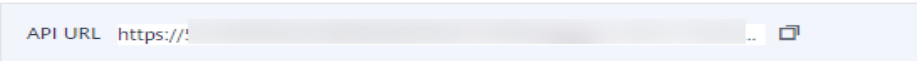
APIG.0101 Incorrect Prediction URL

If the prediction URL is incorrect, API Gateway intercepts the request and reports error message "APIG.0101:The API does not exist or has not been published in the environment". In this case, go to the real-time service details page and obtain the correct API address on the **Usage Guides** tab page.

NOTE

If you have specified a custom path in the configuration file, add this path to the called API path. For example, if you have specified custom path **/predictions/poetry**, the called API path will be *{API address}*/**predictions/poetry**.

Figure 5-24 Obtaining an API address



APIG.0201 Request Body Oversized

If a request body is oversized, API Gateway intercepts the request and reports error message "APIG.0201:Request entity too large". Reduce the prediction request body and try again.

If you perform prediction by calling an API address, the maximum size of the request body is 12 MB. If the size of the request body exceeds 12 MB, the request will be intercepted.

If you perform prediction on the **Prediction** tab of the service details page, the maximum size of the request body is 8 MB. The size limit varies between the two tab pages because they use different network links.

Figure 5-25 Request error APIG.0201

APIG.0301 Authentication Failed

If an API is called for service prediction or a token is used for application authentication, a correct token must be obtained. If the token is invalid, API Gateway intercepts the request and reports error message "APIG.0301:Incorrect IAM authentication information: decrypt token fail". Obtain the correct token and enter it in **X-Auth-Token** for prediction. For details about how to obtain a token, see **Obtaining a User Token Through Password Authentication**.

APIG.1009 Unmatched AppKey and AppSecret

If the AppKey and AppSecret used for service prediction do not match, error message "APIG.1009:AppKey or AppSecret is invalid" is displayed.

Obtain the AppKey and AppSecret and access the real-time service using application authentication. For details, see **Access Authenticated Using an Application**.

5.3.3 Error ModelArts.4206 Occurred in Real-Time Service Prediction

Symptom

After a real-time service is deployed and running, an inference request is sent to the service, but error ModelArts.4206 occurred.

Possible Causes

ModelArts.4206 indicates that the request traffic on an API exceeded the preset threshold. To ensure stable service running, ModelArts limits the inference request traffic on a single API.

Solution

Reduce the inference request traffic on an API. If ultra-high concurrency is required, submit a service ticket.

5.3.4 Error ModelArts.4302 Occurred in Real-Time Service Prediction

Symptom

After a real-time service is deployed and running, an inference request is sent to the service, but error ModelArts.4302 occurred.

Cause Analysis and Solution

Error ModelArts.4302 may occur in multiple scenarios. The following describes two typical scenarios:

1. "error_msg": "Gateway forwarding error. Failed to invoke backend service due to connection refused. "

   This error occurs in either of the following cases:

   - The traffic exceeded the threshold that can be processed by the model. In this case, reduce the traffic or increase the number of model instances.
   - The image is faulty. In this case, separately run the image and check whether it is functional.

2. "error_msg": "Due to self protection, the backend service is disconnected, please wait moment."

   This error occurs due to excessive number of model errors. A large number of model errors trigger dispatcher circuit breaker, leading to a prediction failure. In this case, check the result returned by the model and handle these errors. Adjust request parameters or reduce the request traffic for higher model calling success rate.

5.3.5 Error ModelArts.4503 Occurred in Real-Time Service Prediction

Symptom

After a real-time service is deployed and running, an inference request is sent to the service, but error ModelArts.4503 occurred.

Cause Analysis and Solution

Error ModelArts.4503 may occur in multiple scenarios. The following describes typical scenarios:

1. Communication error

   Request error: {"error_code":"ModelArts.4503","error_msg":"Failed to respond due to backend service not found or failed to respond"}

   To ensure high performance, ModelArts reuses the connections to the same model service. According to the TCP protocol, a disconnection can be initiated either by the client or server of a connection. Disconnecting a connection requires a four-way handshake. If the model service (server) initiates a disconnection, but the connection is being used by ModelArts (client), a communication error occurs and this error code is returned.

If your model is imported from a custom image, set **keep-alive** of the web server used by the custom image to a larger value. This prevents a disconnection request initiated from the server. If you use Gunicorn as the web server, configure the **keep-alive** value by running the **Gunicorn** command. Models imported from other sources have been configured in the service.

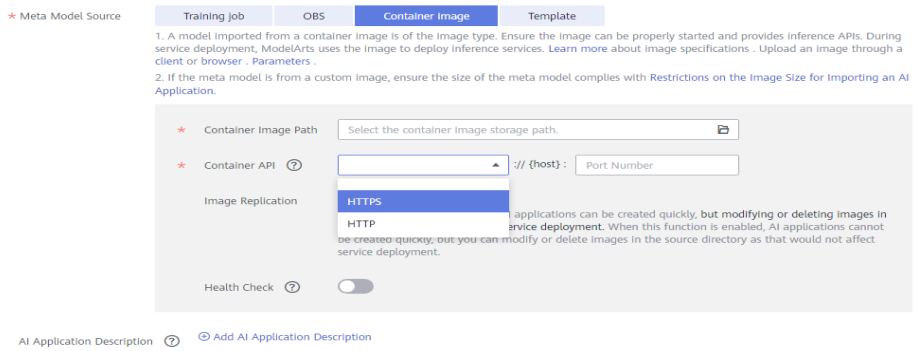
2. Protocol error

   Request error: {"error_code":"ModelArts.4503", "error_msg":"Failed to find backend service because SSL error in the backend service, please check the service is https"}

   If the model used for deploying a real-time service is imported from a container image, this error occurs when the protocol used by the container API is incorrectly configured.

   For security purposes, all ModelArts inference requests are HTTPS-compliant. When you import a model from a container image, ModelArts allows the image to use HTTPS or HTTP. However, you must specify the protocol used by the image in **Container API**.

   Figure 5-26 Container API



   If the **Container API** value is inconsistent with the value provided by your image, for example, **Container API** is set to **HTTPS** but your image actually uses HTTP, the preceding error occurs.

   To resolve this issue, create an AI application version, select the correct protocol (HTTP or HTTPS), and deploy a real-time service again or update the existing real-time service.

3. Long prediction time

   The following error is reported: {"error_code": "ModelArts.4503", "error_msg": "Failed to find backend service because response timed out, please confirm your service is able to process the request without timeout. "}

   Due to the limitation of API Gateway, the prediction duration of each request does not exceed 40 seconds. A prediction is successful if the entire process takes a time not longer than the time limit. The process involves sending data to ModelArts, performing prediction, and sending the prediction result back. If a prediction takes a time longer than the time limit or ModelArts cannot respond to frequent prediction requests, this error occurs.

   Take the following measures to resolve this issue:

- If a prediction request is oversized, the request times out due to slow data processing. In this case, optimize the prediction code to shorten the prediction time.
- A complex model leads to slow inference. Optimize the model to shorten the prediction time.
- Increase the number of instances or select a compute node flavor with better performance. For example, use GPUs instead of CPUs to improve the service processing performance.

4. Service error

   The following error is reported: {"error_code": "ModelArts.4503","error_msg": "Backend service respond timeout, please confirm your service is able to process the request without timeout. "}

   Service logs are as follows:

   ```
   [2022-10-24 11:37:31 +0000] [897] [INFO] Booting worker with pid: 897
   [2022-10-24 11:41:47 +0000] [1997] [INFO] Booting worker with pid: 1997
   [2022-10-24 11:41:22 +0000] [1897] [INFO] Booting worker with pid: 1897
   [2022-10-24 11:37:54 +0000] [997] [INFO] Booting worker with pid: 997
   ```

   The service malfunctions and restarts repeatedly. As a result, prediction requests cannot be sent to the service instance.

   Take the following measures to resolve this issue:

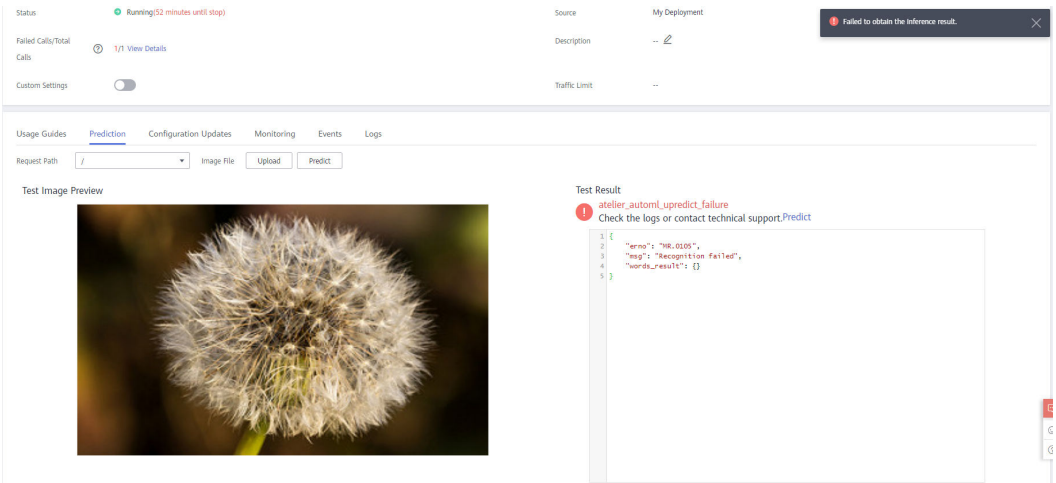
   - Reduce the number of prediction requests and check whether the fault is resolved. If the fault does not recur, the service process exits due to heavy load. In this case, increase the number of instances or improve the instance specifications.
   - The inference code is defective. Debug the code to rectify the fault.

5.3.6 Error MR.0105 Occurred in Real-Time Service Prediction

Symptom

During the prediction in a running real-time service, error { "erno": "MR.0105", "msg": "Recognition failed","words_result": {}} occurred.

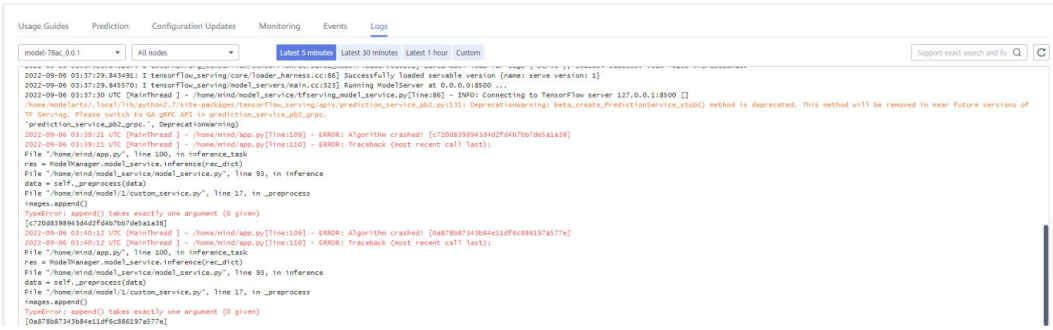
Figure 5-27 Prediction failed



Possible Causes

Locate the fault by analyzing the error log on the **Logs** tab of the real-time service details page.

Figure 5-28 Error log



According to the error log shown in the preceding figure, the prediction failure is caused by the model inference code.

Solution

According to the error log, mandatory parameters are missing in the append() method. To rectify the fault, modify the code in the model inference code file **customize_service.py** to transfer proper parameters to the append() method.

For details about how to edit the model inference code, see **Specifications for Model Inference Coding**.

5.3.7 Method Not Allowed

Symptom

Error message "Method Not Allowed" is displayed during service prediction.

Possible Causes

The APIs registered by default for service prediction must be called using POST. If you use GET, API Gateway will intercept the request.

Solution

Use POST to call the API.

5.3.8 Request Timed Out

Symptom

The prediction request times out, and the error {"error_code": "ModelArts.4205","error_msg":"Connection time out."} is reported.

Possible Causes

If a request times out, there is a high probability that the request is intercepted by API Gateway. Check the API Gateway and model.

Solution

1. Run the **:curl -kv** *{Prediction address}* command on the local host to check whether the API Gateway is reachable. If the request timed out, check the local firewall, proxy, and network configurations.
2. Check whether the model is started or the duration for the model to process a single request. Due to the limitation of API Gateway, the duration of a single prediction cannot exceed 40s. If the duration exceeds 40s, the system will return a timeout error by default.

5.3.9 Error Occurred When an API Is Called for Deploying a Model Created Using a Custom Image

If an error occurs when an API is called for service deployment, check the following items:

1. Check whether POST is used in the configuration file for the model API.
2. Check whether the URL in the configuration file contains a customized path, for example, **/predictions/poetry** (the default path is **/**).
3. Check whether the called path in the body of the API request contains a customized path, for example, **{API address}/predictions/poetry**.

5.3.10 Error "DL.0105" Occurred During Real-Time Inference

Symptom

Error "DL.0105" occurred during real-time inference. Error log: "TypeError: 'float' object is not subscriptable".

Possible Causes

According to the error logs, a float data record is accessed as an object subscript.

Solution

Change **x[0][i]** in the model inference code to **x[i]** and deploy the real-time inference service again.

6 MoXing

6.1 Error Occurs When MoXing Is Used to Copy Data

Symptom

1. When you call **moxing.file.copy_parallel()** to copy a file from the OBS bucket for a development environment to another bucket, the file is not visible in the target bucket.
2. An error occurs when MoXing is used to copy data. Example:
   - The following error occurs when MoXing is used to copy OBS data in the ModelArts development environment: **keyError: 'request-id'**
   - The error **No files to copy** occurs when ModelArts uses MoXing to copy data.
   - socket.gaierror: [Errno -2] Name or service not known
   - ERROR:root:Failed to call:

     func=<bound method ObsClient.getObject of <obs.client.ObsClient object at 0x7fd705939710>>

     args=('bucket', 'data/TFRecord/HY_all_inside/no_adjust_light_3/09_06_6x128x128_0000000212.tfrecord')
3. When MoXing is used to copy data, an error message is displayed, indicating that the operation timed out. Example:
   - TimeoutError: [Errno 110] Connection timed out
   - WARNING:root:Retry=9,Wait=0.1, Timestamp = 1567152567.5327423

Possible Cause

The possible causes are as follows:

- The source file does not exist.
- The target OBS path is incorrect or the two OBS paths are not in the same region.
- Space of the training job is insufficient.

Solution

Check the following items based on the error message:

1. Check whether the first parameter of **moxing.file.copy_parallel()** contains a file. If it contains no file, the error message "No files to copy" is displayed.
   - If the file exists, go to **2**.
   - If the file does not exist, ignore the error and proceed with subsequent operations.
2. Check whether the OBS path where data is copied is in the same region as the development environment or training job.

   Log in to the ModelArts management console, and view the region where ModelArts resides. Log in to OBS Console, and view the region where the OBS bucket resides. Check whether they are in the same region.
   - If they are in the same region, go to step **3**.
   - If they are not in the same region, create a bucket and a folder in OBS that is in the same region as ModelArts, and upload data to the bucket.
3. Check whether the OBS path is **obs://xxx**. You can check whether the OBS path exists as follows:

   mox.file.exists('obs://bucket_name/sub_dir_0/sub_dir_1')
   - If the path exists, go to **4**.
   - If the path does not exist, change it to an available OBS path.
4. Check whether the used resource is a CPU. The **/cache** directory of the CPU and the code directory share 10 GB. The possible cause is insufficient space. You can run the following command in code to check the disk size:

   os.system('df -hT')
   - If disk space is sufficient, go to **5**.
   - If disk space is insufficient, use GPU resources.
5. If data fails to be copied using MoXing in a notebook instance, run the **df -hT** command on the **Terminal** page to check the space size and check whether the failure cause is insufficient space. You can use EVS to attach disks when creating a notebook instance.

If code is correct but the problem persists, submit a service ticket to get professional technical support.

6.2 How Do I Disable the Warmup Function of the Mox?

Symptom

When the TensorFlow version of the training job Mox is running, 50 steps are executed for four times before the job is formally running.

Warmup indicates a process of using a small learning rate to train several epochs first. Network parameters are randomly initialized. If a large learning rate is used at the beginning, the value may be unstable. This is why warmup is used. After the

training process is basically stable, the originally set initial learning rate can be used for training.

Possible Cause

There are multiple execution modes for distributed TensorFlow. Mox executes 50 steps for four times to record the execution time, and selects the model with the minimum execution time.

Solution

When creating a training job, add **variable_update=parameter_server** in **Running Parameter** to disable the warmup function of Mox.

6.3 Pytorch Mox Logs Are Repeatedly Generated

Symptom

The Pytorch engine of a frequently-used framework is used as an algorithm source of a ModelArts training job. During the running of the training job, Mox versions for each epoch will be printed in the Pytorch Mox log. The log details are as follows:

```
INFO:root:Using MoXing-v1.13.0-de803ac9
INFO:root:Using OBS-Python-SDK-3.1.2
INFO:root:Using MoXing-v1.13.0-de803ac9
INFO:root:Using OBS-Python-SDK-3.1.2
```

Possible Cause

Pytorch creates multiple processes in spawn mode. Each process invokes the Mox to download data in multi-process mode. In this case, subprocesses are destroyed and recreated repeatedly, and Mox is imported repeatedly. As a result, a large amount of Mox version information is printed.

Solution

To avoid repeated output of the Pytorch Mox logs of the training job, you need to add the following code to the boot file. When **MOX_SILENT_MODE = "1"**, Mox version information can be blocked in the log.

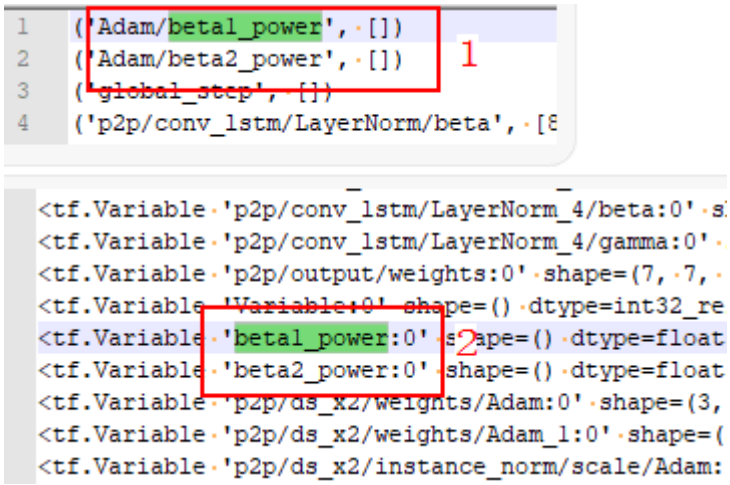
```
import os
os.environ["MOX_SILENT_MODE"] = "1"
```


6.4 Does moxing.tensorflow Contain the Entire TensorFlow? How Do I Perform Local Fine Tune on the Generated Checkpoint?

Symptom

When MoXing is used to train a model, **global_step** is placed in the Adam name range. The non-MoXing code does not contain the Adam name range. See **Figure 6-1**. In the figure, **1** indicates MoXing code, and **2** indicates non-MoXing code.

Figure 6-1 Sample code



Solution

Fine tune is a process of using a model that is trained by others and your own data to train a new model. It is equivalent to using the several top layers of a model trained by others to extract shallow features, and then making the features fall into our own classification.

Generally, the accuracy of a newly trained model increases gradually from a very low value. However, fine tune allows you to obtain a better effect after a relatively small number of iterations. The advantage of fine tune is that it prevents you from training a model from scratch and improves training efficiency. Fine tune is a good choice when the data volume is not large.

All APIs contained in **moxing.tensorflow** have been optimized for TensorFlow. The actual APIs inside are the native APIs of TensorFlow.

If non-MoXing code does not contain the Adam name range, add the following content to non-MoXing code:

```
with tf.variable_scope("Adam"):
```

When adding code, you are advised to use **tf.train.get_or_create_global_step()** instead of **global_step**.

6.5 Copying Data Using MoXing Is Slow and the Log Is Repeatedly Printed in a Training Job

Symptom

- Copying data using MoXing is slow in a ModelArts training job.
- The log **INFO:root:Listing OBS** is repeatedly printed.

Figure 6-2 Repeated log printing

```
INFO:root:Listing OBS: 77000
INFO:root:Listing OBS: 78000
INFO:root:Listing OBS: 79000
INFO:root:Listing OBS: 80000
INFO:root:Listing OBS: 81000
INFO:root:Listing OBS: 82000
INFO:root:Listing OBS: 83000
INFO:root:Listing OBS: 84000
INFO:root:Listing OBS: 85000
INFO:root:Listing OBS: 86000
INFO:root:Listing OBS: 87000
INFO:root:Listing OBS: 88000
INFO:root:Listing OBS: 89000
```

Possible Cause

1. The possible causes for slow data copying are as follows:
   - Reading data from OBS will make data reading become a training bottleneck, resulting in slow iteration.
   - Data fails to be read from OBS due to environment or network issues. As a result, the job fails.
2. The log is printed repeatedly. The log indicates that the file is being read from the remote end. After the file list is read, data starts to be downloaded. If there are many files, this process takes a long time.

Solution

When creating a training job, you can save data to OBS. You are advised not to use the OBS APIs of TensorFlow, MXNet, and PyTorch to directly read data from OBS.

- If the file is small, you can save data on OBS as a **.tar** package. When starting the training, download the package from OBS to the **/cache** directory and decompress the package.
- If the file is large, save data as multiple **.tar** packages and invoke multiple processes in the entry script to decompress data in parallel. You are advised not to save discrete files to OBS. Otherwise, data download will be slow.
- In a training job, use the following code to decompress the **.tar** package:

```
import moxing as mox
import os
```



```
mox.file.copy_parallel("obs://donotdel-modelarts-test/AI/data/PyTorch-1.0.1/tiny-imagenet-200.tar", '/cache/tiny-imagenet-200.tar')
os.system('cd /cache; tar -xvf tiny-imagenet-200.tar > /dev/null 2>&1')
```

6.6 Failed to Access a Folder Using MoXing and Read the Folder Size Using get_size

Symptom

- The folder cannot be accessed using MoXing.
- The folder size read by using **get_size** of MoXing is 0.

Possible Cause

To use MoXing to access a folder, you need to add the **recursive=True** parameter. The default value is **False**.

Solution

Obtain the size of an OBS folder.

```
mox.file.get_size('obs://bucket_name/sub_dir_0/sub_dir_1', recursive=True)
```

Obtain the size of an OBS file.

```
mox.file.get_size('obs://bucket_name/obs_file.txt')
```


7 APIs or SDKs

7.1 "ERROR: Could not install packages due to an OSError" Occurred During ModelArts SDK Installation

Symptom

When ModelArts SDKs are installed, the following error message is displayed: "ERROR: Could not install packages due to an OSError: [WinError 2] The system cannot find the file specified: 'c:\python39\Scripts\ephemeral-port-reserve.exe' -> 'c:\python39\Scripts\ephemeral-port-reserve.exe.deleteme".

Possible Causes

The role of the login user is incorrect.

Solution

Log in to the system as the administrator, press **Windows+R**, enter **cmd**, and run the following command:

```
python -m pip install --upgrade pip
```

7.2 Error Occurred During Service Deployment After the Target Path to a File Downloaded Through a ModelArts SDK Is Set to a File Name

Symptom

A ModelArts SDK was used to download a file from OBS, and the target path was set to the file name. No error was reported in the local IDE, but an error occurred when the target AI application was deployed as a real-time service.

Sample code:


```
session.obs.download_file (obs_path, local_path)
```

The error message is as follows:

```
2022-07-06 16:22:36 CST [ThreadPoolEx] - /home/work/predict/model/customize_service.py[line:184] -
WARNING: 4 try: IsADirectoryError(21, 'Is a directory'). update products failed!
```

Possible Causes

The target path (**local_path**) was incorrectly set in code.

Solution

Set **local_path** to a folder and ensure the folder name extension ends with a slash (/).

7.3 A Training Job Created Using an API Is Abnormal

Symptom

When you call an API to create a training job (CPU specifications for the dedicated resource pool), the training job status changes from **Creating** to **Abnormal**, and specifications information on the training job details page is **--**.

Possible Causes

A parameter that is not supported by dedicated resource pools of CPU specifications is used in the API call.

Solution

Make sure that the API request body does not contain **flavor_id** because this parameter is not supported by dedicated resource pools of CPU specifications

8 Change History

Released On	Description
2024-01-18	Added: • **A Training Job Created Using a Custom Image Is Always in the Running State** • **Troubleshooting a Training Job Failure** • **A Training Job Created Using an API Is Abnormal** • **Running a Job Failed Due to Persistently Rising Memory Usage** Added **An NCCL Error Occurs When a Training Job Fails to Be Executed**.
2023-11-23	Added **An NCCL Error Occurs When a Training Job Fails to Be Executed**.
2023-11-08	Added: • **Failed to Create a Notebook Instance and JupyterProcessKilled Is Displayed in Events** • **Storage Volume Failed to Be Mounted to the Pod During Training Job Creation**

Released On	Description
2023-09-07	Added **The Model or Image Exceeded the Size Limit for AI Application Import**. Added **A Single Model File Exceeded the Size Limit (5 GB) for AI Application Import**. Added **What Do I Do If an Image Fails to Be Pulled When a Service Is Deployed, Started, Upgraded, or Modified?**. Added **What Do I Do If an Image Restarts Repeatedly When a Service Is Deployed, Started, Upgraded, or Modified?**. Added **What Do I Do If a Container Health Check Fails When a Service Is Deployed, Started, Upgraded, or Modified?**. Added **What Do I Do If Resources Are Insufficient When a Service Is Deployed, Started, Upgraded, or Modified?**.
2023-08-31	Deleted "DevEnviron (Notebook of Old Version)".
2023-08-30	Deleted "OBS Operation Issues" in *DevEnviron (New Notebook)* and "Why Error: 403 Forbidden Is Displayed When I Perform Operations on OBS?" in *General Issues*. Moved OBS documentation into **General Issues > Incorrect OBS Path on ModelArts**.
2022-11-01	Modified the document structure. Added cases related to AI application management. Added **service prediction failure** cases.
2022-08-31	Added case **Error MR.0105 Occurred in Real-Time Service Prediction**.
2022-08-26	Added a general OBS case: **Incorrect OBS Path on ModelArts**
2022-08-15	Added cases related to **training job suspension**.
2022-01-04	Added OBS download permission cases.
2021-12-15	Added case **Error ModelArts.2763 Occurred During Training Job Creation**.
2021-09-15	Added training job troubleshooting cases.
2021-07-16	Revised the contents of training job. Deleted an outdated item of troubleshooting from training jobs. Added content of troubleshooting to training jobs. **Training Job Process Exits Unexpectedly** **Stopped Training Job Process**

Released On	Description
2020-12-10	Added the troubleshooting guide for ExeML. **Failed to Publish a Dataset Version** **Invalid Dataset Version** **Failed to Create an ExeML-powered Training Job** **ExeML-powered Training Job Failed** **Failed to Submit the Model Publishing Task** **Failed to Publish a Model** **Failed to Submit the Real-time Service Deployment Task** **Failed to Deploy a Real-time Service**
2019-11-25	This is the first official release.

