davincirunsdk

发行版本 *v0.1.5*

Wh1isper

2022 年 10 月 28 日

参考 API 文档

为类 Jupyter 交互式环境提供 Notebook 友好的 Ascend 分布式训练 SDK

- `davincirun` 命令，支持 Modelarts Ascend 训练作业，不再需要打包 davinci 文件夹
- `init_rank_table` 支持转换 v0.1 hccl json -> v1.0 hccl json
- `start_distributed_train` , `wait_distributed_train` 根据 v1.0 hccl json 启动并等待分布式训练完成
- notebook 友好，`output_notebook=True` 支持在 notebook 中输出分布式训练日志

CHAPTER 1

Installation

```
pip install davincirunsdk
```

CHAPTER 2

仓库地址

- Github Repo
- OpenI

CHAPTER 3

Reference

3.1 公共 API

davincirunsdk.**init_rank_table**() → Dict

SDK，训练作业中用户应使用此函数转换 hccl v0.1 -> v1.0

返回

设置后的环境变量，False 则为未找到 rank_table，跳过设置

返回类型

Dict or False

davincirunsdk.**set_random_ms_cache_dir**()

用于暂时设置 MindSpore compiler 缓存文件夹，用完自动销毁；

这个方法允许你在启动分布式训练后做一些额外的工作，如果不需要，可以使用 start_and_wait_distributed_train

示例

```
>>> with set_random_ms_cache_dir():
>>>     manager = start_distributed_train(train_command)
>>>     ... # do some extra work
>>>     wait_distributed_train(manager)
```

Returns:

davincirunsdk.**start_and_wait_distributed_train**(*command*, *work_dir='./'*, *log_dir='./log'*, ***, *output_notebook=False*, *random_cache_dir=True*, *destroy_when_finished=True*, *raise_exception=True*)

启动并等待分布式训练完成

参数

- **command** (*List*) -- command list，用于启动训练脚本
- **work_dir** (*Path-like string*) -- 工作目录，如果 command 存在相对路径，需要确保从工作目录访问相对路径正确
- **log_dir** (*Path-like string*) -- 日志输出目录
- **output_notebook** -- 默认为 False，当为 True 时，将自动输出日志到 notebook 中；如果在非 notebook 环境中打开，不应当有任何作用
- **random_cache_dir** -- 默认为 True，是否使用随机缓存目录，避免在工作目录下生成大量算子缓存
- **destroy_when_finished** -- 默认为 True，是否在结束时销毁所有子进程；通常及时销毁可以帮助释放 NPU 资源，除非你想深入进程细节
- **raise_exception** -- 默认为 True，是否在子进程失败时 raise exception，以确保外部得到 exception 提示，这在流水线中判断执行结果很有用

示例

```
>>> start_and_wait_distributed_train(train_command)
```

返回

状态码，0 为正常结束，1 为异常

抛出

DistributedRuntimeError -- 分布式训练失败，`raise_exception=True` 可抛出.

davincirunsdk.**start_distributed_train**(*command*, *work_dir='./'*, *log_dir='./log'*, ***, *output_notebook=False*)

启动分布式训练任务

参数

- **command** (*List*) -- command list，用于启动训练脚本
- **work_dir** -- 工作目录，如果 command 存在相对路径，需要确保从工作目录访问相对路径正确
- **log_dir** -- 日志输出目录
- **output_notebook** -- 默认为 False，当为 True 时，将自动输出日志到 notebook 中；如果在非 notebook 环境中打开，不应当有任何作用

示例

```
>>> with set_random_ms_cache_dir():
>>>     manager = start_distributed_train(train_command)
>>>     ... # do some extra work
>>>     wait_distributed_train(manager)
```

返回

FMKManager

davincirunsdk.**wait_distributed_train**(*fmk_manager*, *destroy_when_finished=True*, *raise_exception=True*)

等待分布式训练完成

参数

- **fmk_manager** -- FMKManager, 通常是使用 start_distributed_train 的返回
- **destroy_when_finished** -- 默认为 True，是否在结束时销毁所有子进程；通常及时销毁可以帮助释放 NPU 资源，除非你想深入进程细节
- **raise_exception** -- 默认为 True，是否在子进程失败时 raise exception，以确保外部得到 exception 提示，这在流水线中判断执行结果很有用

返回

状态码，0 为正常结束，1 为异常

抛出

DistributedRuntimeError -- 分布式训练失败，`raise_exception=True` 可抛出.

3.2 notebook

3.2.1 notebook.sdk

davincirunsdk.notebook.sdk.**_set_extra_env**(*rank_table*)

训练任务转换 hccl V0.1 -> v1.0 时，额外适配的环境变量，`Example` 展示了当前的配置

参数

rank_table -- RankTable，可以是 V0 或者 V1

示例

```
>>> os.environ['RANK_START'] = str(rank_start)
>>> os.environ['RANK_SIZE'] = str(rank_table.get_device_num())
```

Returns:

davincirunsdk.notebook.sdk.**generate_rank_table**()

训练作业时用于 hccl v0.1 -> v1.0 转换

如果当前已经有了 v1.0 的 hccl 文件，直接使用 get_rank_table

返回

RankTable, 可能是 RankTableV0 或 RankTableV1

davincirunsdk.notebook.sdk.**get_rank_table**()

读入环境变量中的 RANK_TABLE

返回

RankTableV1

davincirunsdk.notebook.sdk.**init_rank_table**() → Dict

SDK，训练作业中用户应使用此函数转换 hccl v0.1 -> v1.0

返回

设置后的环境变量，False 则为未找到 rank_table，跳过设置

返回类型
Dict or False

davincirunsdk.notebook.sdk.**set_random_ms_cache_dir**()

用于暂时设置 MindSpore compiler 缓存文件夹，用完自动销毁；

这个方法允许你在启动分布式训练后做一些额外的工作，如果不需要，可以使用 start_and_wait_distributed_train

示例

```
>>> with set_random_ms_cache_dir():
>>>     manager = start_distributed_train(train_command)
>>>     ... # do some extra work
>>>     wait_distributed_train(manager)
```

Returns:

davincirunsdk.notebook.sdk.**set_rank_env**(*rank_table*)

这里重新设置了 hccl 文件的地址，主要是针对 V0.1 转换为 V1.0 转换的场景

参数
rank_table -- RankTable，可以是 V0 或者 V1

Returns:

davincirunsdk.notebook.sdk.**start_and_wait_distributed_train**(*command*, *work_dir='./'*, *log_dir='./log'*, ***, *output_notebook=False*, *random_cache_dir=True*, *destroy_when_finished=True*, *raise_exception=True*)

启动并等待分布式训练完成

参数

- **command** (*List*) -- command list，用于启动训练脚本
- **work_dir** (*Path-like string*) -- 工作目录，如果 command 存在相对路径，需要确保从工作目录访问相对路径正确
- **log_dir** (*Path-like string*) -- 日志输出目录
- **output_notebook** -- 默认为 False，当为 True 时，将自动输出日志到 notebook 中；如果在非 notebook 环境中打开，不应当有任何作用
- **random_cache_dir** -- 默认为 True，是否使用随机缓存目录，避免在工作目录下生成大量算子缓存
- **destroy_when_finished** -- 默认为 True，是否在结束时销毁所有子进程；通常及时销毁可以帮助释放 NPU 资源，除非你想深入进程细节
- **raise_exception** -- 默认为 True，是否在子进程失败时 raise exception，以确保外部得到 exception 提示，这在流水线中判断执行结果很有用

示例

```
>>> start_and_wait_distributed_train(train_command)
```

返回
状态码，0 为正常结束，1 为异常

抛出
DistributedRuntimeError -- 分布式训练失败，`raise_exception=True` 可抛出.

davincirunsdk.notebook.sdk.**start_distributed_train**(*command*, *work_dir='./'*, *log_dir='./log'*, ***, *output_notebook=False*)

启动分布式训练任务

参数

- **command** (*List*) -- command list，用于启动训练脚本
- **work_dir** -- 工作目录，如果 command 存在相对路径，需要确保从工作目录访问相对路径正确
- **log_dir** -- 日志输出目录
- **output_notebook** -- 默认为 False，当为 True 时，将自动输出日志到 notebook 中；如果在非 notebook 环境中打开，不应当有任何作用

示例

```
>>> with set_random_ms_cache_dir():
>>>     manager = start_distributed_train(train_command)
>>>     ... # do some extra work
>>>     wait_distributed_train(manager)
```

返回
FMKManager

davincirunsdk.notebook.sdk.**wait_distributed_train**(*fmk_manager*, *destroy_when_finished=True*, *raise_exception=True*)

等待分布式训练完成

参数

- **fmk_manager** -- FMKManager, 通常是使用 start_distributed_train 的返回
- **destroy_when_finished** -- 默认为 True，是否在结束时销毁所有子进程；通常及时销毁可以帮助释放 NPU 资源，除非你想深入进程细节
- **raise_exception** -- 默认为 True，是否在子进程失败时 raise exception，以确保外部得到 exception 提示，这在流水线中判断执行结果很有用

返回
状态码，0 为正常结束，1 为异常

抛出
DistributedRuntimeError -- 分布式训练失败，`raise_exception=True` 可抛出.

3.2.2 notebook.exception

exception davincirunsdk.notebook.exception.**DistributedRuntimeError**
分布式训练失败错误

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python 模块索引

d

索引

符号

D

G

I

S

W

