Approaches to Build and Run MindSpore on Windows

2023-01-18
OpenMindSpore Project
Silicon Valley System Software Lab

Contents

- Goals
- Solutions
    1. Using WSL 2
    2. Using MSVC
    3. Using DirectML

Goals

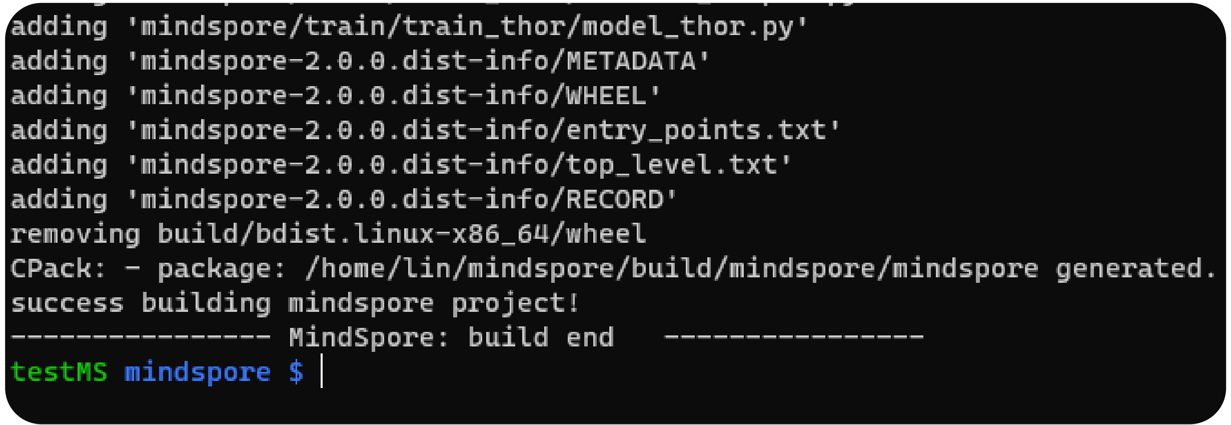
- Most ML platforms, including TF, and PyTorch etc, support Windows environment.
- It is useful to support MindSpore on Windows, including build, training and serving.
- It is the needs of application but the necessary to expand the ecosystem of MindSpore.

Solutions

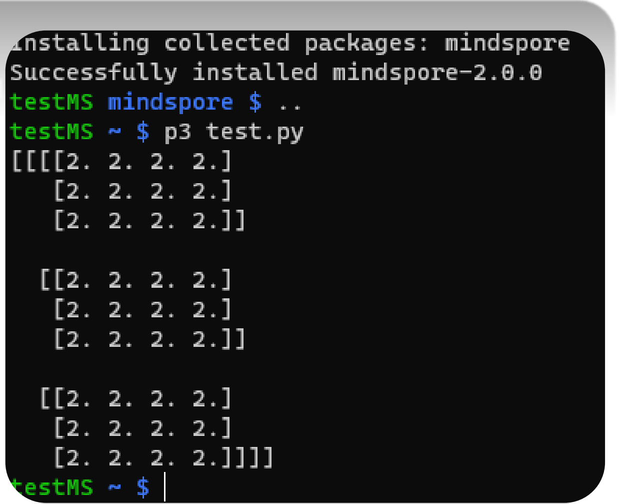
1. The most nature way in Windows, is to build and run MS in WSL2 which has been integrated into latest Windows, e.g. 10/11 for home/business and server 2022 for the infra and cloud. (*Evaluated in W11+WSL2*)
2. To build MS with CMake and MSVC, by including MS libraries (*refer to example of running PyTorch examples*)
3. To leverages DirectML to provide cross-vendor hardware acceleration on Windows and its WSL. Need build a package, mindspore_directml, with MS C++ API.

Solutions – WSL2

- Utilize WSL2 as the bridge to build and run MindSpore on WSL2
- Through tweaks, we can successfully build and run MindSpore-gpu.



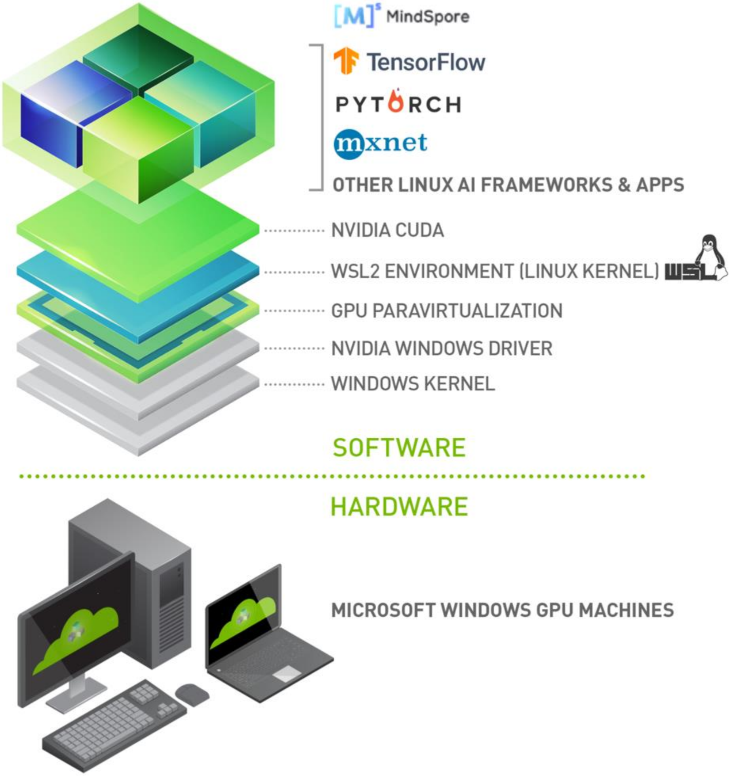
```
adding 'mindspore/train/train_thor/model_thor.py'
adding 'mindspore-2.0.0.dist-info/METADATA'
adding 'mindspore-2.0.0.dist-info/WHEEL'
adding 'mindspore-2.0.0.dist-info/entry_points.txt'
adding 'mindspore-2.0.0.dist-info/top_level.txt'
adding 'mindspore-2.0.0.dist-info/RECORD'
removing build/bdist.linux-x86_64/wheel
CPack: - package: /home/lin/mindspore/build/mindspore/mindspore generated.
success building mindspore project!
---------------- MindSpore: build end   ----------------
testMS mindspore $ 
```



```
Installing collected packages: mindspore
Successfully installed mindspore-2.0.0
testMS mindspore $ ..
testMS ~ $ p3 test.py
[[[[2. 2. 2. 2.]
   [2. 2. 2. 2.]
   [2. 2. 2. 2.]]

  [[2. 2. 2. 2.]
   [2. 2. 2. 2.]
   [2. 2. 2. 2.]]

  [[2. 2. 2. 2.]
   [2. 2. 2. 2.]
   [2. 2. 2. 2.]]]]
testMS ~ $ 
```



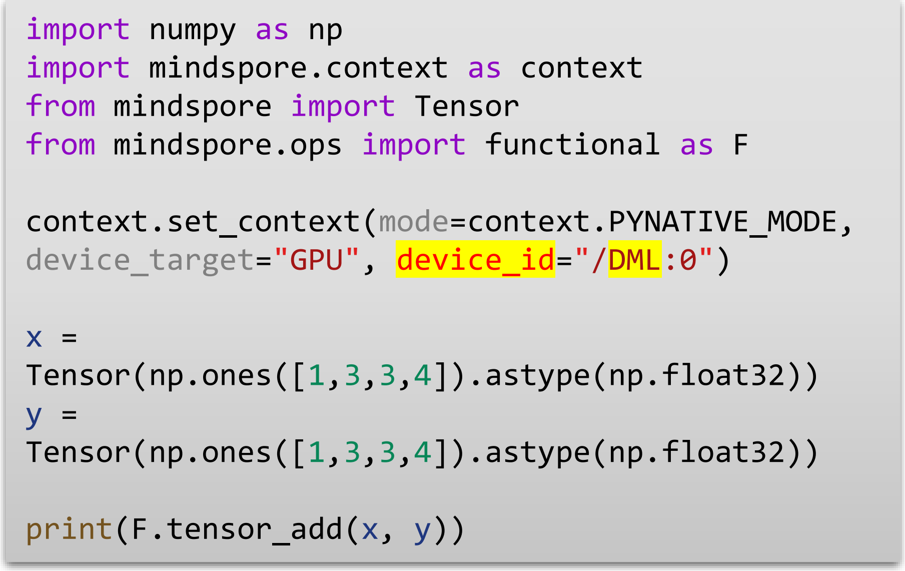
Solutions – CMAKE+MSVC

```
if (MSVC)
  file(GLOB TORCH_DLLS "${MS_INSTALL_PREFIX}/lib/*.dll")
  add_custom_command(TARGET example-app
                     POST_BUILD
                     COMMAND ${CMAKE_COMMAND} -E
copy_if_different
                     ${MS_DLLS}
                     $<TARGET_FILE_DIR:example-app>)
endif (MSVC)
```

Solutions – DirectML

	Windows ML	ONNX Runtime with DirectML	TensorFlow with DirectML	DirectML
Use Case	The best developer experience for ONNX model inferencing on Windows.	Cross platform C API for ONNX model inferencing.	Hardware accelerated model training on any DirectX 12 GPU.	Provides flexibility with direct access to DirectX 12 resources for high-performance frameworks and applications.
Documentation	MS Docs	GitHub	GitHub and MS Docs	GitHub and MS Docs
Distribution	Windows SDK or NuGet: Microsoft.AI.MachineLearning	NuGet: Microsoft.ML.OnnxRuntime.DirectML	PyPI Package: tensorflow-directml	Windows SDK or NuGet: Microsoft.AI.DirectML
DirectML Support	Inference	Inference	Inference and Training	Inference and Training

- MindSpore need develop mindspore-directml package to support DirectML
- It supports multi-GPU. Use DML_VISIBLE_DEVICES to control which GPU(s) get used by DirectML.
- Need support Ops under DirectML.
- There is quite much work to be done for this solution.
- A proposed example as the right.



```
import numpy as np
import mindspore.context as context
from mindspore import Tensor
from mindspore.ops import functional as F

context.set_context(mode=context.PYNATIVE_MODE,
device_target="GPU", device_id="/DML:0")

x =
Tensor(np.ones([1,3,3,4]).astype(np.float32))
y =
Tensor(np.ones([1,3,3,4]).astype(np.float32))

print(F.tensor_add(x, y))
```

Thank You

FUTUREWEI
Technologies