CUDA compile problems on Windows, Cmake error: No CUDA toolset found

Ask Question

Asked 4 years, 7 months ago Modified 3 months ago Viewed 33k times

11

so I've been successfully working on my CUDA program on my Linux but I would like to support Windows platform as well. However, I've been struggling with correctly compiling it. I use :

- Windows 10
- Cmake 3.15
- Visual Studio 2017
- CUDA Toolkit 10.1

When using the old **deprecated** Cmake CUDA support of using `find_package(CUDA 10.1 REQUIRED)` it correctly reports the correct path to the **toolkit** when using it. However, it is my understanding that the latest Cmake does not properly support the old method anymore and that `cuda_add_library` etc don't properly link anymore. So I have **reformatted** my 'CMakeLists.txt' file to the following based on this:

```
cmake_minimum_required(VERSION 3.8 FATAL_ERROR)
project(myproject LANGUAGES CXX CUDA)

add_library(mylib SHARED mycudalib.cu)

# My code requires C++ 11 for the CUDA library, not sure which ones of these
# will do the trick correctly. Never got the compiler this far.
target_compile_features(mylib PUBLIC cxx_std_11)
SET(CMAKE_CXX_STANDARD 11)
SET(CMAKE_CUDA_STANDARD 11)


set_target_properties( mylib PROPERTIES CUDA_SEPARABLE_COMPILATION ON)

add_executable(test_mylib test.cpp)

target_link_libraries(test_mylib mylib ${CUDA_CUFFT_LIBRARIES})
```

However, I get the following error from line 2:

```
CMake Error at C:/Program Files/CMake/share/cmake-3.15/Modules/CMakeDetermineCompi
lerId.cmake:345 (message):
  No CUDA toolset found.
Call Stack (most recent call first):
  C:/Program Files/CMake/share/cmake-3.15/Modules/CMakeDetermineCompilerId.cmake:3
2 (CMAKE_DETERMINE_COMPILER_ID_BUILD)
  C:/Program Files/CMake/share/cmake-3.15/Modules/CMakeDetermineCUDACompiler.cmak
e:72 (CMAKE_DETERMINE_COMPILER_ID)
  CMakeLists.txt:2 (project)
```

I've tried a variation of suggestions online such as adding the following to 'CMakeLists.txt':

```
set(CMAKE_CUDA_COMPILER "C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v10.1/
bin/nvcc")
```

or adding the following variable to Cmake:

```
CUDACXX                              C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v10.1
```

This is the 'CMakeLists.txt' file I use on Linux to compile succesfully. The difference is there I use Cmake 3.5 and CUDA **Toolkit** 9.0:

```
cmake_minimum_required(VERSION 3.5)
project( myproject)
find_package(CUDA 9.0 REQUIRED)
if(CUDA_FOUND)
        list(APPEND CUDA_NVCC_FLAGS "-std=c++11")
endif(CUDA_FOUND)

cuda_add_library(mylib SHARED mycudalib.cu)
cuda_add_executable(test_mylib test.cpp)
target_link_libraries(test_mylib mylib ${CUDA_CUFFT_LIBRARIES})
```

Featured on Meta

- Updates to the Acceptable Use Policy (AUP) – January 2024
- What would you like to change about the moderator election process?
- Discussions update: Expansion to all tags in February
- Temporary policy: Generative AI (e.g., ChatGPT) is banned

Related

5	configuring CMake to setup CUDA on Windows
8	CMake does not properly find CUDA libra
1	CMake 3.x + CUDA - compilation busted
8	CUDA with visual studio and cmake
1	cmake finds cuda but fails to find cuda libraries on Windows
5	CMake 3.4.3 Can't find CUDA on windows
0	Cmake can't find the cuda.h file
0	CMake cannot find CUDA: "Could not find cmake module file: CMakeDetermineCUDACompiler.cmake"
0	Compile error CMAKE with CUDA on Visu Studio C++
6	CUDA 12 C++20 support with CMake not working

Hot Network Questions

- Why does my perpetual CMBR rocket ship not work? Why does the CMBR have infinite energy?
- Is Dark Matter really invisible?
- Render white noise texture in Cycles
- Are identical sets distinct objects?
- What is considered good/acceptable grade in Germany?
- Boy who is also a prankster gets body swapped with his school principal
- Do Blu-ray drives have an RID?
- Frequencies are inherently positive physical quantity.... what does it mean?
- A confusion in understanding a solution to a number theory problem
- What is this strange and possibly old white 2x4 brick?
- Can The President counter Mayorkas impeachmen with a pardon?
- Modular Equivalence
- Use the pyrolytic mode in ovens for cooking
- Did Allied bombers return from enemy airspace with undropped bombs?
- Sufficient conditions for refactoring
- How to make searches not appear at the bottom of the window

c++ cmake compiler-errors cuda nvcc

Share Improve this question Follow

asked Jun 17, 2019 at 18:09
Mineral
357 ● 1 ● 2 ● 11

Add a comment

8 Answers

Sorted by: Highest score (default)

18

For Windows 10, VS2019 Community, and CUDA 11.3, the following worked for me:

1. Extract the full installation package with 7-zip or WinZip
2. Copy the four files from this extracted directory `.\visual_studio_integration\CUDAVisualStudioIntegration\extras\visual_studio_integration\MSBuildExtensions` into the MSBuild folder of your VS2019 install `C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\MSBuild\Microsoft\VC\v160\BuildCustomizations`

The four files are:

- CUDA 11.3.props
- CUDA 11.3.targets
- CUDA 11.3.xml
- Nvda.Build.CudaTasks.v11.3.dll

I had tried **installing** (and reinstalling) CUDA with Visual Studio Integration, but CMake wasn't able to find the CUDA installation (even with CUDA_PATH and CMAKE_CUDA_COMPILER defined).

Share Improve this answer Follow

answered Jun 24, 2021 at 18:16
bjacobowski
829 ● 7 ● 6

4 Note that you actually install the CUDA **toolkit** from an **executable** (not extract from 7-zip). Then, in the CUDA subfolder you listed (e.g. `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2\extras\visual_studio_integration\MSBuildExtensions` for CUDA 10.2, you'll find the 4 files you listed. Those you copy to the MS Visual Studio folder you listed. – adam.hendry Aug 18, 2021 at 22:32

Add a comment

12

I have tried it on a different PC now and it works fine. So I had absolutely no idea why it's not working on this one. As CUDA_PATH is correctly setup in my system variables.

Then looking into it further, by **uninstalling** the 'Build Tools' of Visual Studio and only having the Community IDE **installed**, CMake used the IDE instead of the Build Tools and then it started working fine.

Share Improve this answer Follow

edited Jun 19, 2019 at 11:21

answered Jun 19, 2019 at 10:53
Mineral
357 ● 1 ● 2 ● 11

15 I just ran into the same issue with the Build Tools. If you want to keep the Build Tools **installed**, you just need to copy everything from: C:\Program Files\NVIDIA GPU **Computing** Toolkit\CUDA\v11.4\extras\visual_studio_integration\MSBuildExtensions To: C:\Program Files (x86)\Microsoft Visual Studio\2019\BuildTools\MSBuild\Microsoft\VC\v160\BuildCustomizations Change your CUDA and VS versions in those paths as necessary. For some reason the CUDA **toolkit** installer doesn't consider the Build Tools installs when choosing where to add the integrations. – Matthew Dixon Aug 23, 2021 at 22:04

After replicating this copy step, the error **persists**. Has anyone found additional **constraints** or workarounds? – ROS Jan 15, 2023 at 1:46

@MatthewDixon your comment should be made an answer. thanks – Ji_in_coding Jan 1 at 6:08

Add a comment

"Ghost", as in the noun

What is the most significant currently active organization using quadratic voting?

How to find out about my bicycle

Book about siblings in a magical world?

Is this a rule in German language that words that end with "stand" always have "der" artikel?

How to get the numbers from 50 - 100 with the numbers 2, 0, 2, 4

Is axiom of replacement nicely stateable in the language of ETCS?

Relativity, Lorentz Transforms and the time it takes to perceive

Question feed

3

Look at this. It may solve your issues. https://gitlab.kitware.com/cmake/cmake/issues/19029

Seems like Nvidia cuda installer has some issues with **installing** the VS integration with vs 2017. Check if you can find this file in your vs **installing** path.

```
C:/Program Files (x86)/Microsoft Visual
Studio/2017/Professional/Common7/IDE/VC/VCTargets/BuildCustomizations/CUDA
10.1.xml
```

Share Improve this answer Follow

answered Jun 24, 2019 at 11:50
Neng Qian
75 ● 1 ● 8

This file is missing on my system (well, I am using 2019, so it's missing from `C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE\VC\VCTargets`). Is there an installer option which will installs that file? – user2023370 Sep 10, 2019 at 10:29

I have no idea. Finally, I decided to use instead just use VS 2015. And it works quite well so far. – Neng Qian Sep 10, 2019 at 18:55

Add a comment

3

I was trying to build darknet from source and came across this issue.

What resolved it for me was the following:

- making sure no other Visual Studio or Visual Studio Build Tool was installed except for VS2019. (I configured this using the uninstall feature of the ~1 mb vs_community.exe installer program)
- REINSTALLING CUDA 10.1, using the 2.5 gb installer, and in that process, making sure 'VS Integration' is installed (for me... this was a 'reinstall' since I had already installed it, but with a bunch of VS2019,VS2017 + Build Tools all installed at once!!) during the installation.

At that point, my cudnn files were still in the bin/lib/include folder of the 10.1 installation, and I hit "Configure" in CMake again.

Success! No errors. (CMake 3.18, VS2019, CUDA 10.1.243, cudnn 7.6.5)

Share Improve this answer Follow

answered Aug 27, 2020 at 0:02
Ryu S.
1,637 ● 2 ● 23 ● 42

Add a comment

2

I just have the same issue of No CUDA toolset found with different versions, and my system:

-Windows 11 -Cmake 3.20.0 -Visual Studio 2019 -CUDA **Toolkit** 11.6

Some netizens said that it happened if you **installed** Visual Studio before you install CUDA. So, I tried and reinstall CUDA, finally it work now. You also can try it. Good luck.

enter image description here

Share Improve this answer Follow

answered Jan 23, 2022 at 7:24
James
21 ● 1

Add a comment

2

I had a similar problem, and probably @James claim is right, it is just visual studio and cuda integration **mismatch**. I followed @bjacobowski's solution. For any future reference, I integrated CUDA 12.1 and Visual Studio 2022 community edition.

- I copied the four files from `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.1\extras\visual_studio_integration\MSBuildExtensions`
- And pasted into `C:\Program Files\Microsoft Visual Studio\2022\Community\MSBuild\Microsoft\VC\v170\BuildCustomizations`

Share Improve this answer Follow

answered Aug 25, 2023 at 16:03
bim
652 ● 7 ● 18

Yeah, confirmed that copying these four files fixes the problem with VS2020. – Jorge M. Londoño P. Nov 6, 2023 at 16:07

Add a comment

1

For anyone battling CMake and CUDA, my solution to this problem was to add the following to the CMake command:

```
-DCMAKE_GENERATOR_TOOLSET="cuda=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA
\v12.2"
```

As well as copying the 4 VS integration files (from `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v12.2\extras\visual_studio_integration\MSBuildExtensions`) to both:

- `C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools\MSBuild\Microsoft\VC\v170\BuildCustomizations`
- and `C:\Program Files\Microsoft Visual Studio\2022\Community\MSBuild\Microsoft\VC\v170\BuildCustomizations`

After all this CMake was happy with CUDA and started compiling. I didn't reinstall CUDA or anything.

Share Improve this answer Follow

answered Sep 12, 2023 at 2:51
Roy Shilkrot
3,229 ● 31 ● 26

Add a comment

0

For anyone else who has a similar problem, I FINALLY realized my issue! My cuda toolset was x64, but my cmake build script by default was trying to build x86. Passing -A x64 to my cmake command fixed the issue. In my particular case, I was building **llama**.cpp, and my command was:

cmake .. -DLLAMA_CUBLAS=ON -A x64

Share Improve this answer Follow

answered Oct 26, 2023 at 20:06
alvion
2,063 ● 3 ● 16 ● 23

Add a comment