How to compile Tensor Flow with SSE and and AVX instructions on Windows?

With the latest version of Tensor Flow now on windows, I am trying to get everything working as efficiently as possible. However, even when compiling from source I still can't seem to figure out how to enable the SSE and AVX instructions.

The default process: https://github.com/tensorflow/tensorflow/tree/r0.12/tensorflow/contrib/cmake has no mention of how to do this.

The only reference I have found has been using Google's Bazel: How to compile Tensorflow with SSE4.2 and AVX instructions?

Does anyone know of an easy way to turn on these advanced instructions using MSBuild? I hear they give at least a 3X speed up.

To help those looking for a similar solution, this is the warning I am currently getting looks like this: https://github.com/tensorflow/tensorflow/tree/r0.12/tensorflow/contrib/cmake

I am using Windows 10 Professional on a 64 bit platform, Visual Studio 2015 Community Edition, Anaconda Python 3.6 with cmake version 3.6.3 (later versions dont' work for Tensor Flow)

Any help appreciated!

c++ windows msbuild tensorflow

edited May 23 '17 at 12:34 Community ♦ 1 1

asked Mar 5 '17 at 1:28 Aerophilic 530 3 15

side note, they give "at most" 3x speed-up. You'll see this speed-up if your computation is mostly huge matrix multiplies – Yaroslav Bulatov Mar 5 '17 at 19:03

3 Answers

Well, I tried to fix that, but I am not sure if it really worked.

In `CMakeLists.txt` you will find the following statements:

```
if (tensorflow_OPTIMIZE_FOR_NATIVE_ARCH)
  include(CheckCXXCompilerFlag)
  CHECK_CXX_COMPILER_FLAG("-march=native" COMPILER_OPT_ARCH_NATIVE_SUPPORTED)
```

On MSVC platform, the test failes because MSVC doesn't support `-march=native` flag. I modified the statements like below:

```
if (tensorflow_OPTIMIZE_FOR_NATIVE_ARCH)
  include(CheckCXXCompilerFlag)
  CHECK_CXX_COMPILER_FLAG("-march=native" COMPILER_OPT_ARCH_NATIVE_SUPPORTED)
  if (COMPILER_OPT_ARCH_NATIVE_SUPPORTED)
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -march=native")
  else()
    CHECK_CXX_COMPILER_FLAG("/arch:AVX" COMPILER_OPT_ARCH_AVX_SUPPORTED)
    if(COMPILER_OPT_ARCH_AVX_SUPPORTED)
      set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} /arch:AVX")
    endif()
  endif()
endif()
```

By doing this, cmake would check if `/arch:AVX` is available and use it. Accordinf to MSDN and MSDN, SSE2 support is enabled by default for x86 compiling but not available for x64 compiling. For x64 compiling you can choose to use AVX or AVX2. I used AVX above because my CPU only supports AVX, youcan try AVX2 if you have a compatible CPU.

By compiling use the above `CMakeLists.txt` , the compiling preocedure was much slower than official release, and warning about 'AVX/AVX2' disappeared, but warning about SSE/SSE2/3/4.1/4.2 still exists. I think these warnings can be ignored because there's no SSE support for x64 MSBuild.