# Building CUDA-aware Open MPI

## 1. How do I build Open MPI with CUDA-aware support?

CUDA-aware support means that the MPI library can send and receive GPU buffers directly. This feature exists in the Open MPI v1.7 series and later. The support is being continuously updated so different levels of support exist in different versions. We recommend you use the latest version for best support.

Configure and build Open MPI >= 2.0.0 with UCX

We recommend [UCX1.4](https://github.com/openucx/ucx/releases/tag/v1.4.0) built with [gdrcopy](https://github.com/NVIDIA/gdrcopy) for the most updated set of MPI features and for better performance.

1. Configure and build UCX to provide CUDA support

shell$ ./configure --prefix=/path/to/ucx-cuda-install --with-cuda=/usr/local/cuda --with-gdrcopy=/usr

shell$ make -j8 install

2. Configure and build Open MPI to leverage UCX CUDA supprt

shell$ ./configure --with-cuda=/usr/local/cuda --with-ucx=/path/to/ucx-cuda-install

shell$ make -j8 install

## Configuring the Open MPI v1.8 series and Open MPI v1.7.3, v1.7.4, and v1.7.5

With Open MPI v1.7.3 and later, the libcuda.so library is loaded dynamically so there is no need to specify a path to it at configure time. Therefore, all you need is the path to the cuda.h header file.

Examples:

1. Search in default location(s). Looks for cuda.h in /usr/local/cuda/include or your default prefix.

shell$ ./configure --with-cuda

2. Search in a specified location. Looks for cuda.h in /usr/local/cuda-v6.0/cuda/include.

shell$ ./configure --with-cuda=/usr/local/cuda-v6.0/cuda

Note that you cannot configure with **--disable-dlopen**, as that will break the ability of the Open MPI library to dynamically load libcuda.so.

Configuring Open MPI v1.7, MPI v1.7.1, and v1.7.2

--with-cuda(=DIR)       Build cuda support, optionally adding DIR/include,

                          DIR/lib, and DIR/lib64

  --with-cuda-libdir=DIR  Search for cuda libraries in DIR

Here are some examples of configure commands that enable CUDA support:

1. Search in default locations. Looks for cuda.h in /usr/local/cuda/include and libcuda.so in /usr/lib64.

shell$ ./configure --with-cuda

2. Search for cuda.h in /usr/local/cuda-v4.0/cuda/include and libcuda.so in default location of /usr/lib64.

shell$ ./configure --with-cuda=/usr/local/cuda-v4.0/cuda

3. Search for cuda.h in /usr/local/cuda-v4.0/cuda/include and libcuda.so in /usr/lib64 (same as previous one).

shell$ ./configure --with-cuda=/usr/local/cuda-v4.0/cuda --with-cuda-libdir=/usr/lib64

If the cuda.h or libcuda.so files cannot be found, then the configure will abort.

**Note:** There is a bug in Open MPI v1.7.2 such that you will get an error if you configure the library with **--enable-static**. To get around this error, add the following to your configure line and reconfigure. This disables the build of the PML BFO which is largely unused anyways. This bug is fixed in Open MPI v1.7.3.

shell$ ./configure --enable-mca-no-build=pml-bfo

See [this FAQ entry](https://www.open-mpi.org/faq/?category=runcuda) for detals on how to use the CUDA support.

## 2. How do I build Open MPI with CUDA-aware support using PGI?

With CUDA 6.5, you can build all versions of CUDA-aware Open MPI without doing anything special. However, with CUDA 7.0 and CUDA 7.5, you need to pass in some specific compiler flags for things to work correctly. Add the following to your configure line.

CFLAGS=-D\_\_LP64\_\_ --with-wrapper-cflags="-D\_\_LP64\_\_ -ta:tesla"

**Update:** With PGI 15.9 and later compilers, the CFLAGS=-D\_\_LP64\_\_ is no longer needed.