**Required support libraries**[**ℑ**](https://docs.open-mpi.org/en/main/installing-open-mpi/required-support-libraries.html#required-support-libraries)

Open MPI requires the following support libraries with the minimum listed versions:

| **Library** | **Minimum version** | **Notes** |
| --- | --- | --- |
| [Hardware Locality](https://www.open-mpi.org/projects/hwloc/) | 1.11.0 | This library is required; Open MPI will not build without it. apt-get -y install hwloc |
| [Libevent](https://libevent.org/) | 2.0.21 | This library is required; Open MPI will not build without it. apt-get install libevent-dev |
| [PMIx](https://pmix.org/) | 4.1.2 | This library is required; Open MPI will not build without it. |
| [PRRTE](https://github.com/openpmix/prrte) | 2.0.2 | This library is optional in some environments. PRRTE provides Open MPI’s full-featured mpirun / mpiexec MPI application launchers (the two are identical; they are symbolic links to the same executable). Apt install **libpmi-pmix-dev**  * If your environment uses another MPI application launcher (e.g., Slurm users can use the srun launcher to “direct launch” Open MPI applications), then the use of PRRTE is optional. * If your environment has no other MPI application launcher, then you need to install PRRTE and build Open MPI with PRRTE support. * Open MPI can use the copy of PRRTE embedded in its source code tree, or compile/link against an external PRRTE installation. [See this section for details about how to specify each method](https://docs.open-mpi.org/en/main/installing-open-mpi/configure-cli-options/required-support-libraries.html#label-building-ompi-cli-options-required-support-libraries). |

1. wget <https://download.open-mpi.org/release/open-mpi/v4.1/openmpi-4.1.5.tar.gz>
2. tar xvf openmpi-X.Y.Z.tar.gz
3. cd openmpi-X.Y.Z
4. ./autogen.pl

###You will need very recent versions of GNU Autoconf, Automake, and Libtool. If autogen.pl fails, read the HACKING file. If anything else fails, read the HACKING file. Finally, we suggest reading the HACKING file.

1. mkdir build
2. cd build
3. yum install gcc-c++ autoconfig libtool

./configure --prefix=<path> [...options...] 2>&1 | tee config.out

make [-j N] all 2>&1 | tee make.out

make install 2>&1 | tee install.out

apt install openjdk-17-jre-headless /// for JAVA installation

apt install default-jre

update-alternatives --config java

There are 2 choices for the alternative java (providing /usr/bin/java).

Selection Path Priority Status

------------------------------------------------------------

\* 0 /usr/lib/jvm/java-17-openjdk-amd64/bin/java 1711 auto mode

1 /usr/lib/jvm/java-11-openjdk-amd64/bin/java 1111 manual mode

There are 2 choices for the alternative java (providing /usr/bin/java).manual mode

Selection Path Priority Status

------------------------------------------------------------

\* 0 /usr/lib/jvm/java-17-openjdk-amd64/bin/java 1711 auto mode

1 /usr/lib/jvm/java-11-openjdk-amd64/bin/java 1111 manual mode

2 /usr/lib/jvm/java-17-openjdk-amd64/bin/java 1711 manual mode

Press <enter> to keep the current choice[\*], or type selection number: 1

update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/java to provide /usr/bin/java (java) in manual mode

1. ../configure --prefix=/where/to/install ###/app/openmpi-4.1.5
2. lscpu ###check number of core
3. make -j$(nproc)
4. make install
5. mpiexec –version or mpirun –version
6. export PATH=/app/openmpi-4.1.5/bin:$PATH

MPI with CUDA UBUNTU

1. Install stuff
   1. sudo apt-get install make libevent-dev
   2. apt install glibc-source ###ldconfig
   3. apt-get install -y opal-utils gfortran
   4. sudo apt-get install gcc g++
   5. sudo apt-get install python3.8
   6. sudo apt-get install pip
2. Install cuda using the debian installer or runfile installer [installation guide](https://docs.nvidia.com/cuda/cuda-quick-start-guide/index.html#ubuntu-x86_64-deb)
3. Update path
   1. export PATH=/usr/local/cuda-11.4/bin${PATH:+:${PATH}}
   2. export LD\_LIBRARY\_PATH=/usr/local/cuda-11.4/lib64${LD\_LIBRARY\_PATH:+:${LD\_LIBRARY\_PATH}}
   3. export CUDA\_HOME=/usr/local/cuda-11.4
4. Install Open MPI
   1. [openmpi-4.1.1.tar.gz](https://www.open-mpi.org/software/ompi/v4.1/)
   2. tar -xzf openmpi-4.1.1.tar.gz
   3. cd openmpi-4.1.1
   4. apt-file search mpicc

Version: 4.1.5

Build MPI C bindings: yes

Build MPI C++ bindings (deprecated): no --enable-mpi-cxx

Build MPI Fortran bindings: no

MPI Build Java bindings (experimental): no --enable-mpi-java

Build Open SHMEM support: false (no spml)

Debug build: no

Platform file: (none)

Wrapper --disable-wrapper-runpath

* 1. ../configure --enable-mpi-cxx --enable-mpi-fortran --enable-debug

Error :- collect2: error: ld returned 1 exit status

make[2]: \*\*\* [Makefile:1893: opal\_wrapper] Error 1

make[2]: Leaving directory '/opt/openmpi-4.1.5/build/opal/tools/wrappers'

make[1]: \*\*\* [Makefile:2431: install-recursive] Error 1

make[1]: Leaving directory '/opt/openmpi-4.1.5/build/opal'

make: \*\*\* [Makefile:1913: install-recursive] Error 1

../configure --enable-mpi-cxx --enable-mpi-fortran --enable-debug --enable-wrapper-rpath=/opt/openmpi-4.1.5/opal/tools/wrappers/

* 1. ../configure --with-cuda=/usr/local/cuda-11.4
  2. sudo make all install
  3. If there are failures in the process (like missing make) delete the folder unzip again and repeat
  4. At this point you should be able to run mpicc and mpiexec. If not you may need to add it to your path (look up it should show where it installed it to)
     1. You may also need to set export LD\_LIBRARY\_PATH=/usr/local/lib:$LD\_LIBRARY\_PATH where /usr/local/lib is where openmpi installed the libraries to if you get an error such as mpiexec: error while loading shared libraries: libopen-rte.so.40: cannot open shared object file: No such file or directory
  5. ompi\_info --parsable -l 9 --all | grep mpi\_built\_with\_cuda\_support:value should show mca:mpi:base:param:mpi\_built\_with\_cuda\_support:value:true if you're MPI has cuda support

1. Install anaconda
   1. Download from the anaconda website
   2. bash ./Anaconda3-2021.11-Linux-x86\_64.sh
   3. You probably don't want to have anaconda be initialized at startup as this which set aliases for pip and python
2. Install numba
   1. Make sure CUDA\_HOME is to the path that specifies the cuda that you build OpenMPI with [numba cudatoolkit installation reference](https://numba.pydata.org/numba-doc/latest/cuda/overview.html" \l "cudatoolkit-lookup) export CUDA\_HOME= /usr/local/cuda-11.4
   2. conda install numba
   3. conda install cudatoolkit
3. Run with cuda-aware MPI, you can send NDDeviceArrays over MPI! Device memory can be sent via MPI
   1. Note that you'll have to run with conda's version of python3 (ex: ~/anaconda/bin/python3) and install packages with conda's version of pip (ex: ~/anaconda/bin/pip3)

MPI with CUDA Centos

1) tar xf openmpi-X.Y.Z.tar.gz

2) cd openmpi-X.Y.Z

3) ./autogen.plA

4) mkdir build

5) cd build

6) ../configure --prefix=/where/to/install ###/app/openmpi-4.1.5

7) lscpu ###check number of core

8) make -j$(nproc)

9) make install

Error:- /usr/bin/ld: cannot find -lnuma

/usr/bin/ld: cannot fin collect2: error: ld returned 1 exit status

make[2]: \*\*\* [libmca\_common\_ucx.la] Error 1

make[2]: Leaving directory `/anup/openmpi-4.1.5/opal/mca/common/ucx'

make[1]: \*\*\* [all-recursive] Error 1

make[1]: Leaving directory `/anup/openmpi-4.1.5/opal'

make: \*\*\* [all-recursive] Error 1 d –liberty

#ld –lnuma --verbose

#ld –liberty --verbose

# locate liberty

# locate libnuma.so

Output:-- /usr/lib64/libnuma.so.1

/usr/lib64/libnuma.so.1.0.0

# ln -s /usr/lib/libz.so.1.2.8 /usr/lib/libzlib.so

For extra library required

yum install binutils binutils-devel

#yum install numactl-devel

/usr/bin/ld: cannot find -lbfd

/usr/bin/ld: cannot find -liberty

collect2: error: ld returned 1 exit status

# yum install binutils binutils-devel

make[2]: \*\*\* [sshmem\_ucx\_module.lo] Error 1

make[2]: Leaving directory `/openmpi-3.0.0/oshmem/mca/sshmem/ucx'

make[1]: \*\*\* [all-recursive] Error 1

make[1]: Leaving directory `/openmpi-3.0.0/oshmem'

make: \*\*\* [all-recursive] Error 1

% ./autogen.sh

% ./contrib/configure-release --prefix=$PWD/install

% make -j8 install

Our new configure run looks like this:

**## >** ./configure --prefix=/path/to/openmpi-3.0.0

--with-cuda=/path/to/cuda-8.0/

--with-ucx=/path/to/openucx-ucx-1338d7f

--- MCA component pml:ucx (m4 configuration macro)

checking for MCA component pml:ucx compile mode... dso

checking --with-ucx value... not found

configure: WARNING: Expected file

/path/to/openucx-ucx-1338d7f/include/ucp/api/ucp.h not found

configure: error: **Cannot continue**

git clone https://github.com/openucx/ucx.git ucx-git

cd ucx-git

mkdir build

cd build

 ../configure --prefix=/path/to/ucx-bin/

../contrib/configure-release --prefix=/anup/app/ucx-bin/

make

make install

cat /etc/profile.d/mpi-2.0.0.sh

export PATH=/usr/local/mpi-2.0.0/bin:${PATH}

export LD\_LIBRARY\_PATH=/usr/local/mpi-2.0.0/lib:${LD\_LIBRARY\_PATH}

export C\_INCLUDE\_PATH=/usr/local/openmpi-2.0.0/include:${C\_INCLUDE\_PATH}

locale

locale: Cannot set LC\_CTYPE to default locale: No such file or directory

locale: Cannot set LC\_MESSAGES to default locale: No such file or directory

locale: Cannot set LC\_ALL to default locale: No such file or directory

LANG=en\_IN

LC\_CTYPE="en\_US.UTF-8"

LC\_NUMERIC="en\_US.UTF-8"

LC\_TIME="en\_US.UTF-8"

LC\_COLLATE="en\_US.UTF-8"

LC\_MONETARY="en\_US.UTF-8"

LC\_MESSAGES="en\_US.UTF-8"

LC\_PAPER="en\_US.UTF-8"

LC\_NAME="en\_US.UTF-8"

LC\_ADDRESS="en\_US.UTF-8"

LC\_TELEPHONE="en\_US.UTF-8"

LC\_MEASUREMENT="en\_US.UTF-8"

LC\_IDENTIFICATION="en\_US.UTF-8"

LC\_ALL=en\_US.UTF-8

dpkg-reconfigure locales