# Appendix
# Hints for Making Your Computer a Parallel Machine

Practical advises for the installation of required supporting software for parallel program execution on different operating systems are given. Note that this guide and Internet links can change in the future, and therefore always look for the up-to-date solution proposed by software tools providers.

## A.1 Linux

***OpenMP***
OpenMP 4.5 has been a part of GNU GCC C/C++, the standard C/C++ compiler on Linux, by default since GCC's version 6, and thus it comes preinstalled on virtually any recent mainstream Linux distribution. You can check the version of your GCC C/C++ compiler by running command

```
$ gcc --version
```

The first line, e.g., something like

```
gcc (Ubuntu 6.3.0-12ubuntu2) 6.3.0 20170406
```

contains the information about the version of GCC C/C++ compiler (6.3.0 in this example).

Utility `time` can be used to measure the execution time of a given program. Furthermore, Gnome's System Monitor or the command-line utilities `top` (with separate-cpu-states displayed—press 1 once `top` starts) and `htop` can be used to monitor the load on individual logical cores.



R. Trobec et al., *Introduction to Parallel Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-319-98833-7

***MPI***

The message passing interface (MPI) standard implementation can be already provided as a part of the operating system, most often as MPICH [1] or Open MPI [2,10]. If it is not, it can usually be installed through the provided package management systems, for example, the apt in Ubuntu:

```
sudo apt install libmpich-dev
```

The MPICH is an open high-performance and widely portable implementation of the MPI, which is well maintained and supports the latest standards of the MPI. MPICH runs on parallel systems of all sizes, from multi-core nodes to computer clusters in large supercomputers. Alternative open-source implementations exist, e.g., Open MPI, with similar performances and user interface. Other implementations are dedicated to a specific hardware, and some of them are commercial; however, the beauty of the MPI remains, your program will possibly execute with all of the MPI implementations, eventually after some initial difficulties. In the following, we will mostly use the acronym MPI, regardless of the actual implementation of the standard, except in cases if such a distinguishing is necessary.

Invoke the MPI execution manager: `>mpiexec` to check for the installed implementation of the MPI library on your computer. Either a note that the program is currently not installed or a help text will be printed. Let us assume that an Open MPI library is installed on your computer. The command: `>mpiexec -h` shows all the available options. Just a few of them will suffice for testing your programs.

We start working with typing the first program. Make your local directory, e.g., with OpenMPI:

```
>mkdir OMPI
```

retype the "Hello World" program from Sect. 4.3 in your editor and save your code in file `OMPIHello.c`. Compile and link the program with a setup for maximal speed:

```
>mpicc -O3 -o OMPIHello OMPIHello.c
```

which, besides compiling, also links appropriate MPI libraries with your program. Note that on some system an additional option `-lm` could be needed for correct inclusion of all required files and libraries.

The compiled executable can be run by:

```
>mpiexec -n 3 OMPIHello
```

The output of the program should be in three lines, each line with a notice from a separate process:

```
Hello world from process 0 of 3
Hello world from process 1 of 3
Hello world from process 2 of 3
```

as the program has run on three processes, because the option `-n 3` was used. Note that the line order is arbitrary, because there is no rule about the MPI process execution order. This issue is addressed in more detail in Chap. 4.

***OpenCL***
First of all you need to download the newest drivers to your graphics card. This is important because OpenCL will not work if you do not have drivers that support OpenCL. To install OpenCL, you need to download an implementation of OpenCL. The major graphic vendors NVIDIA, AMD, and Intel have both released implementations of OpenCL for their GPUs. Besides the drivers, you should get the OpenCL headers and libraries included in the OpenCL SDK from your favorite vendor. The installation steps differ for each SDK and the OS you are running. Follow the installation manual of the SDK carefully. For OpenCL headers and libraries, the main options you can choose from are *NVIDIA CUDA Toolkit*, *AMD APP SDK*, or *Intel SDK for OpenCL*. After the installation of drivers and SDK, you should the OpenCL headers:

```
#include<CL/cl.h>
```

If the OpenCL header and library files are located in their proper folders, the following command will compile an OpenCL program:

```
gcc prog.c -o prog -l OpenCL.
```

## A.2 macOS

***OpenMP***
Unfortunately, the LLVM C/C++ compiler on macOS comes without OpenMP support (and the command `gcc` is simply a link to the LLVM compiler). To check your C/C++ compiler, run

```
$ gcc --version
```

If the output contains the line

```
Apple LLVM version 9.1.0 (clang-902.0.39.1)
```

where some numbers might change from one version to another, the compiler most likely do not support OpenMP. To use OpenMP, you have to install the original GNU

GCC C/C++ compiler (use MacPorts or Homebrew, for instance) which prints out something like

```
gcc-mp-7 (MacPorts gcc7 7.3.0_0) 7.3.0
```

informing that this is indeed the GNU GCC C/C++ compiler (version 7.3.0 in this example).

The running time can be measured in the same way as on Linux (see above). Monitoring the load on individual cores can be performed using macOS's Activity Monitor (open its CPU Usage window) or `htop` (but not with macOS's `top`).

***MPI***

In order to use MPI on macOS systems, XDeveloper and GNU compiler must be installed. Download XCode from the Mac App Store and install it by double-clicking the downloaded .dmg file. Use the command: `>mpiexec` to check for installed implementation of the MPI library on your computer. Either a note that the program is currently not installed or a help text will be printed.

If the latest stable release of Open MPI is not present, download it, for example, from the Open Source High Performance Computing website: https://www.open-mpi.org/. To install Open MPI on your computer, first extract the downloaded archive by typing the following command in your terminal (assuming that the latest stable release in 3.0.1):

```
>tar -zxvf openmpi-3.0.1.tar.gz
```

Then, prepare the `config.log` file needed for the installation. The `config.log` file collects information about your system:

```
>cd openmpi-3.0.1
>./configure --prefix=/usr/local
```

Finally, make the executables for installation and finalize the installation:

```
>make all
>sudo make install
```

After successful installation of Open MPI, we start working by typing our first program. Make your local directory, e.g., with `>mkdir OMPI`.

Copy or retype the "Hello World" program from Sect. 4.3 in your editor and save your code in file `OMPIHello.c`.

Compile and link the program with

```
>mpicc -O3 -o OMPIHello OMPIHello.c
```

Execute your program "Hello World" with

```
>mpiexec  -n 3 OMPIHello.
```

The output of the program should be similar to the output of the "Hello World" MPI program from Appendix A.1.

***OpenCL***

If you are using Apple Mac OS X, the Apple's OpenCL implementation should already be installed on your system. MAC OS X 10.6 and later ships with a native implementation of OpenCL. The implementation consists of the OpenCL application programming interface, the OpenCL runtime engine, and the OpenCL compiler.

OpenCL is fully supported by Xcode. If you use Xcode, all you need to do is to include the OpenCL header file:

```
#include <OpenCL/opencl.h>.
```

## A.3 MS Windows

***OpenMP***

There are several options for using OpenMP on Microsoft Windows. To follow the examples in the book as closely as possible, it is best to use Linux Subsystem for Windows 10. If a Linux distribution brings recent enough version of GNU GCC C/C++ compiler, e.g., Debian, one can compile OpenMP programs with it. Furthermore, one can use commands `time`, `top`, and `htop` to measure and monitor programs.

Another option is of course using Microsoft Visual C++ compiler. OpenMP has been supported by it since 2005. Apart from using it from within Microsoft Visual Studio, one can start `x64` Native Tools Command Prompt for VS 2017 where programs can be compiled and run as follows:

```
> cl /openmp /O2 hello-world.c
> set OMP_NUM_THREADS=8
> hello-world.exe
```

With PowerShell run in `x64` Native Tools Command Prompt for VS 2017, programs can be compiled and run as

```
> powershell
> cl /openmp /O2 fibonacci.c
> $env:OMP_NUM_THREADS=8
> ./fibonacci.exe
```

Within PowerShell, the running time of a program can be measured using the command `Measure-Command` as follows:

```
> Measure-Command {./hello-world.exe}
```

Regardless of the compiler used, the execution of the programs can be monitored using Task Manager (open the CPU tab within the Resource Monitor).

***MPI***

More detailed instructions for installation of necessary software for compiling and running the Microsoft MPI can be found, for example, on https://blogs.technet.microsoft.com/windowshpc/2015/02/02/how-to-compile-and-run-a-simple-ms-mpi-program/. A short summary is listed below:

- Download stand-alone redistributables for Microsoft SDK `msmpisdk.msi` and Microsoft MPI `MSMpiSetup.exe` installers from https://www.microsoft.com/en-us/download/confirmation.aspx?id=55991, which will provide execute utility for MPI programs `mpiexec.exe` and MPI service—process manager `smpd.exe`.
- Set the MS-MPI environment variables in a terminal window by `C:\Windows\System32>set MSMPI`, which should print the following lines, if the installation of SDK and MSMPI has been correctly completed:

```
MSMPI_BIN=C:\Program Files\Microsoft MPI\Bin\
MSMPI_INC=C:\Program Files (x86)\Microsoft SDKs\MPI\Include\
MSMPI_LIB32=C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x86\
MSMPI_LIB64=C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x64\
```

  The command: `>mpiexec` should respond with basic library options.
- Download Visual Studio Community C++ 2017 from https://www.visualstudio.com/vs/visual-studio-express/ and install the compiler on your computer, e.g., by selecting a simple desktop development installation.
- You will be forced to restart your computer. After a restarting, start Visual Studio and create `File/New/Project/Windows Console Application`, named, e.g., MSMPIHello, with default settings except the following:

  1. To include the proper header files, open `Project Property pages` and insert in `C/C++/General` under `Additional Include Directories`:
     `$(MSMPI_INC);$(MSMPI_INC)\x64`
     if 64-bit solution will be built. Use `..\x86` for 32 bits.
  2. To set up the linker library in Project Property pages insert in `Linker/General` under `Additional Library Directories`:
     `$(MSMPI_LIB64)`
     if 64-bit platform will be used. Use `$(MSMPI_LIB32)` for 32 bits.
  3. In `Linker/Input` under `Additional Dependencies` add:
     `msmpi.lib;`

4. Close the Project Property window and check in the main Visual Studio window that `Release` solution configuration is selected and select also a solution platform of your computer, e.g., x64.

- Copy or retype "Hello World" program from Sect. 4.3 and build the project.
- Open a Command prompt window, change directory to the folder where the project was built, e.g., `..\source\repos\MSMPIHello\x64\Debug` and run the program from the command window with execute utility:

```
mpiexec -n 3 MSMPIHello
```

that should result in the same output as in Appendix A.1, with three lines, each with a notice from a separate process.

***OpenCL***

First of all you need to download the newest drivers to your graphics card. This is important because OpenCL will not work if you do not have drivers that support OpenCL. To install OpenCL, you need to download an implementation of OpenCL. The major graphic vendors NVIDIA, AMD, and Intel have both released implementations of OpenCL for their GPUs. Besides the drivers, you should get the OpenCL headers and libraries included in the OpenCL SDK from your favorite vendor. The installation steps differ for each SDK and the OS you are running. Follow the installation manual of the SDK carefully. For OpenCL headers and libraries, the main options you can choose from are *NVIDIA CUDA Toolkit*, *AMD APP SDK*, or *Intel SDK for OpenCL*. After the installation of drivers and SDK, you should the OpenCL headers:

```
#include<CL/cl.h>
```

Suppose you are using Visual Studio 2013, you need to tell the compiler where the OpenCL headers are located and tell the linker where to find the OpenCL .lib files.

# References


1. MPICH: High-performance portable MPI. https://www.mpich.org/. Accessed 28 Dec 2017
2. Open MPI: A high performance message passing library. https://www.open-mpi.org/. Accessed 28 Dec 2017
3. Atallah, M., Blanton, M. (eds.): Algorithms and Theory of Computation Handbook, chap. 25. Chapman and Hall, Boca Raton (2010)
4. Chandra, R., Menon, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J.: Parallel Programming in OpenMP. Morgan Kaufmann, Burlington (2000)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press, Cambridge (2009)
6. Dally, W.J., Towles, B.: Principles and Practices of Interconnection Networks. Morgan Kaufmann, Burlington (2004)
7. Duato, J., Yalamanchili, S., Ni, L.: Interconnection Networks. Morgan Kaufmann, Burlington (2002)
8. Flynn, M.J., Mencer, O., Milutinovic, V., Rakocevic, G., Stenstrom, P., Trobec, R., Valero, M.: Moving from petaflops to petadata. Commun. ACM **56**(5), 39–42 (2013). https://doi.org/10.1145/2447976.2447989
9. Foster, I.: Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley Longman Publishing Co., Inc, Boston (1995)
10. Gabriel, E., Fagg, G.E., Bosilca, G., Angskun, T., Dongarra, J.J., Squyres, J.M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R.H., Daniel, D.J., Graham, R.L., Woodall, T.S.: Open MPI: Goals, concept, and design of a next generation MPI implementation. Proceedings. 11th European PVM/MPI Users' Group Meeting, pp. 97–104. Budapest, Hungary (2004)
11. Gaster, B., Howes, L., Kaeli, D.R., Mistry, P., Schaa, D.: Heterogeneous Computing with OpenCL, 1st edn. Morgan Kaufmann Publishers Inc., San Francisco (2011)
12. Grama, A., Gupta, A., Karypis, V., Kumar, V.: Introduction to Parallel Computing, 2nd edn. Pearson (2003)
13. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (1979)




R. Trobec et al., *Introduction to Parallel Computing*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-319-98833-7

14. Jason Long: Hands On OpenCL: An open source two-day lecture course for teaching and learning OpenCL. https://handsonopencl.github.io/ (2018). Accessed 25 Jun 2018
15. Khronos Group: OpenCL: The open standard for parallel programming of heterogeneous systems. https://www.khronos.org/opencl/f (2000–2018). Accessed 25 Jun 2018
16. MPI Forum: MPI: A message-passing interface standard (Version 3.1). Technical report, Knoxville (2015)
17. Munshi, A., Gaster, B., Mattson, T.G., Fung, J., Ginsburg, D.: OpenCL Programming Guide, 1st edn. Addison-Wesley Professional, Reading (2011)
18. OpenMP architecture review board: OpenMP application programming interface, version 4.5, November 2015. http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf (1997–2015). Accessed 18 Dec 2017
19. OpenMP architecture review board: OpenMP application programming interface examples, version 4.5.0, November 2016. http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf (1997–2016). Accessed 18 Dec 2017
20. OpenMP architecture review board: OpenMP. http://www.openmp.org (2012). Accessed 18 Dec 2017
21. van der Pas, R., Stotzer, E., Terboven, C.: Using OpenMP - The Next Step: Affinity, Accelerators, Tasking, and SIMD (Scientific and Engineering Computation). The MIT Press, Cambridge (2017)
22. Robič, B.: The Foundations of Computability Theory. Springer, Berlin (2015)
23. Scarpino, M.: A gentle introduction to OpenCL. http://www.drdobbs.com/parallel/a-gentle-introduction-to-opencl/231002854. Accessed 10 Apr 2018
24. Scarpino, M.: OpenCL in action: how to accelerate graphics and computations. Manning Publications (2011). http://amazon.com/o/ASIN/1617290173/
25. Trobec, R.: Two-dimensional regular d-meshes. Parallel Comput. **26**(13–14), 1945–1953 (2002)
26. Trobec, R., Vasiljević, R., Tomašević, M., Milutinović, V., Beivide, R., Valero, M.: Interconnection networks in petascale computer systems: a survey. ACM Comput. Surv. **49**(3), 44:1–44:24 (2016)

# Index




R. Trobec et al., *Introduction to Parallel Computing*, Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-319-98833-7