Guideline for Installing ppln c++ code

The code structure of Lu's c++ ppln code is slightly rearranged and some scripts (CMake and shell script) are provided so that this code can be built by CMake automatically.

Setting up MPI library

Message Passing Interface (MPI) is a library for distributed computing and is required by ppln code. Maxwell has already provided MPI environment so that every user can load it on demand without installation.

To load it permanently, in terminal, we type:

```
user@maxwell:~$ echo "module load mpi/mpich-3.2-x86_64" >> ~/.bashrc
user@maxwell:~$ source ~/.bashrc
```

Install CMake 3.x (building tool)

Building fftw library (one of the external package ppln code relies on) requires CMake 3.x. However, Maxwell only provides CMake 2.x, we have to install newer version by ourselves.

A shell script "install_camke.sh" is provided to make this process fully automatic.

One can execute this script, for example:

```
user@maxwell:~$ sh install_cmake.sh ~/software
```

This will make a directory "software" in home directory and install camke to "~/software/bin". Also, the path of cmake will be regisried to "~/.bashrc" file so that cmake can be accessed everywhere.

Install ppln code

extract ppln.tar and change directory to ppln

```
user@maxwell:~$ tar xvf ppln.tar
user@maxwell:~$ cd ./ppln
user@maxwell:ppln/$
```

Inside ppln directory, do:

```
user@maxwell:ppln/$ mkdir build
user@maxwell:ppln/$ cd ./build
user@maxwell:ppln/build/$ cmake ..
user@maxwell:ppln/build/$ make -j 10
```

All the external packages (fftw, eigen, boost) and ppln will be downloaded and installed to "ppln/build" automatically.

Compile simulation file

A typical code structure for ppln simulation should be:

```
#include <some third party libraries>
#include "ppln.hpp"

int main()
{
    some code block
}
```

A sample file "sample.cpp" for ppln simulation is already in "ppln" directory.
A script "compile_ppln.sh" is provided so that one can compile the simulation file in this manner:

```
user@maxwell:ppln/$ sh compile_ppln.sh sample.cpp
```

This will generate an executable binary file "sample" in the same directory.

Then, one can try to run this sample simulation by executing the binary file:

```
user@maxwell:ppln/$ ./sample
```