New PPM installation guide

This is a very brief guide on compiling the latest PPM development sources on different systems:

Ubuntu/MAC OSX

pre-requisites

You need to have ruby +1.9.2 installed (ruby 2.0.0 will not work!). Use RVM (http://rvm.io/), since this will give you a recent local ruby install and all the tools to keep your ruby gem packages up-to-date. Rvm can easily be installed by (you will need curl installed):

```
curl -L https://get.rvm.io | bash -s stable --ruby
source ~/.rvm/scripts/rvm
```

If you encounter problems with a missing libyaml (this seems to be often the problem on clusters since they usually don't have packages installed that need libyaml) then you might have to install it too using rvm:

```
rvm pkg install libyaml
rvm reinstall 1.9.3 --with-libyaml-dir=/home/<username>/.rvm/usr
rvm use 1.9.3
```

On MacOS X 10.7 and newer, Apple made clang the default C compiler, but Ruby needs GCC to compile. You hence have to first install GCC. The easiest is using homebrew. Homebrew installs GCC in /usr/local/bin by default. We find that GCC 4.5 works to compile Ruby 1.9.3; other version may not work and give you compiler errors. If you setup differs, please adjust the path below. Then, install Ruby 1.9.3 using:

```
rvm pkg install libyaml
rvm reinstall 1.9.3 --with-gcc=/usr/local/bin/gcc-4.5 --with-libyaml-dir=/Users/<username>/.rvm/usr
rvm use 1.9.3
```

After following the RVM installation instructions and successfully installing the latest ruby, please install following packages

```
gem install cucumber rspec rake thor configatron antlr3
```

For PPM you need metis4 and the blas library. You should get a patched version of metis either from the ppm-library.org website or your colleague.

Finally, to compile PPM you will need make, sed and a **recent** Fortran compiler (gfortran 4.6.2, or ifort 12.1.2)

If you are installing on a recent Debian/Ubuntu? you can get your build system ready by calling:

```
apt-get install build-essential gcc gfortran libblas-dev libatlas-dev libatlas-base-dev\
                libfftw3-dev libyaml-dev
```

On Mac OSX you should consider installing fink, macports or homebrew and installing the dependencies using one of those package managers. As of August 2012 there is no recent version of gfortran. You can find a 4.6.2 prerelease on the gfortran project page. On Mac OSX you do not need to install blas/atlas since it is part of the built-in Accelerate Framework.

Git clone

(Please check also the GitIntro page)

If you haven't obtained the PPM sources yet, clone cg, ppmcore, ppmnumerics and optionally webcg

```
git clone git@ppm.mpi-cbg.de:cg
git clone git@ppm.mpi-cbg.de:ppm ppmcore
git clone git@ppm.mpi-cbg.de:ppmnumerics

#optional - if you are interested in this,
#please first check whether you have the rights to clone this repository

git clone git@ppm.mpi-cbg.de:webcg
```

Anonymous access if you don't have a PPM developer's account:

```
git clone http://ppm.mpi-cbg.de/git/ppmnumerics.git
git clone http://ppm.mpi-cbg.de/git/ppm.git
git clone http://ppm.mpi-cbg.de/git/cg.git
```

After cloning all repos, change to the develop branches.

```
cd cg
git checkout develop
cd ..

cd ppmcore
git checkout develop
cd ..

cd ppmnumerics
git checkout develop
cd ..
```

This might not work with older git versions (<1.6). instead you must call

```
cd cg
git checkout -b develop origin/develop
cd ..

cd ppmcore
git checkout -b develop origin/develop
cd ..

cd ppmnumerics
git checkout -b develop origin/develop
cd ..
```

or update your git (if possible)

setting up the client generator

```
cd cg
rake antlr
bin/ppm --help
```

`rake antlr` regenerates the parser code. This is done using the original Java code and hence requires a Java

Runtime Environment installed on the system. This is never needed when using PPML (cg) from a gem package or as a prepared package, but should be called if you are working with the latest code from git to ensure that your parser is able to handle all Fortran/PPML code. if this fails and you have no Java available, please check with one of the maintainers to get the latest gem.

You should see a list of commands that can be called with the client generator ppm utility. If you get an error message it's time to check your rvm and gems install!

Update PATH variable to include ppm client generator utility

```
export PATH=$PATH:/path/to/cg/bin/
```

compiling ppm

If you are running on a Mac OSX you should first install gnu-sed and a GNU cpp. Both are available via homebrew. run something like this:

```
cd ppmcore
./configure --enable-mpi=openmpi LDFLAGS="-L/Users/<username>/metis/gcc/lib -framework Accelerate"
```

Afterwards, you need to edit Makefile to point sed to the gnu-sed (gsed) you have just installed. You also need to edit Makefile.in and point CPP to the GNU cpp. The standard Apple cpp will not work. Then, hit

```
make
```

On Linux the command will look like this:

```
cd ppmcore
./configure --enable-mpi=openmpi LDFLAGS="-L/home/<username>/metis/gcc/lib -latlas -lblas"
make
```

Once the library finished compiling you can optionally run the unit tests

```
make ftest
```

compiling ppm numerics

ppm numerics can only be compiled after ppmcore is ready. The compilation process is very similar:

If you are running on a Mac OSX you should run something like this:

```
cd ppmnumerics
./configure --enable-mpi=openmpi --with-ppm=/Users/<username>/ppm/ppmcore \
            LDFLAGS="-L/Users/<username>/metis/gcc/lib -framework Accelerate"
make
```

On Linux the command will look like this:

```
cd ppmnumerics
./configure --enable-mpi=openmpi --with-ppm=/home/<username>/ppm/ppmcore \
            LDFLAGS="-L/home/<username>/metis/gcc/lib -latlas -lblas"
make
```

Once the library finished compiling you can again optionally run the unit tests

```
make ftest
```

Fedora/RedHat?

pre-requisites

The following packages are needed in order to install PPM library correctly:

- Ruby 1.9.2
  - cucumber
  - rspec
  - rake
  - thor
  - configatron
  - antlr3
  - sinatra (optional)
  - tilt (optional)
- BLAS
- atlas (?)
- metis4
- fftw

We first install Ruby fftw BLAS and atlas packages with

```
yum install ruby ruby-devel blas-devel atlas-devel fftw-devel
```

Then we install additional ruby components with gem

```
gem install cucumber rspec rake thor configatron antlr3 sinatra tilt
```

To install metis4 you should get a patched version from (http://ppm-library.org/) website or from the official metis web site, but be sure to download the version 4.0.x, because some symbol are changed in newer versions. In order to compile PPM metis, you have to change CC variable inside makefile.in to a valid C++ compiler (gcc for example, or icc if you have an Intel compiler)

```
./MakeMETIS
```

Finally, to compile PPM you will need make, sed and a **recent** Fortran compiler (gfortran 4.6.2)

```
yum install make sed gcc-gfortran
```

Git clone

(Please check also the GitIntro page)

If you haven't obtained the PPM sources yet, clone cg, ppmcore, ppmnumerics and optionally webcg

```
git clone git@ppm.mpi-cbg.de:cg

git clone git@ppm.mpi-cbg.de:ppm ppmcore

git clone git@ppm.mpi-cbg.de:ppmnumerics
```

```
#optional
git clone git@ppm.mpi-cbg.de:webcg
```

After cloning all repos, change to the develop branches.

```
cd cg
git checkout develop
cd ..

cd ppmcore
git checkout develop
cd ..

cd ppmnumerics
git checkout develop
cd ..
```

setting up the client generator

```
cd cg
rake antlr
```

Check the client generator ppm utility

```
bin/ppm --help
```

You should see a list of commands that can be called with the client generator ppm utility. If you get an error message it's time to check your rvm and gems install!

Update PATH variable to include ppm client generator utility, this is required to correctly compile PPM

```
export PATH=$PATH:/path/to/cg/bin/ppm
```

compiling ppm

The command will look like this with MPI enabled

```
cd ppmcore
./configure --enable-mpi=openmpi LDFLAGS="-L/home/<username>/metis/lib -L/usr/lib64/atlas -latlas -lblas"
make
```

Once the library finished compiling you can optionally run the unit tests

```
make ftest
```

compiling ppm numerics

ppm numerics can only be compiled after ppmcore is ready. The compilation process is very similar:

The command will look like this with MPI enabled:

```
cd ppmnumerics
./configure --enable-mpi=openmpi --with-ppm=/home/<username>/ppm/ppmcore \
            LDFLAGS="-L/home/<username>/metis/gcc/lib -L/usr/lib64/atlas -latlas -lblas"
make
```

Once the library finished compiling you can again optionally run the unit tests

```
make ftest
```

Finished!

You can now go back to the cg directory. in it's examples subdirectory you can find some ppm_cg clients that can be now built with `ppm build` (first make sure necessary files and folders are updated).

optional

For webcg you will need the gems sinatra and tilt to be installed. Run the sinatra webcg server from the webcg directory and visit the site on that host. More documentation coming soon!

Try a PPML example

There are several examples in cg/examples. Let's try one of them: a plankton dynamics simulation!

```
cd cg/examples/plankton
```

First, you need to create the directories for the auto-generated Fortran source code and the compiled object files (if they do not already exist):

```
mkdir gen
mkdir gen/obj
mkdir bin
```

Next, you need to edit the file called "settings". You want to change "conf.ppm.base_path" to point to the directory where you installed ppmcore and "conf.ppm.numerics.base_path" to the directory where ppmnumerics resides. If you are on a Mac with a non-GNU sed, you also have to add the line

```
conf.build.sed = "/usr/local/bin/gsed"
```

or wherever you have installed GNU sed when compiling PPMcore. Next, type

```
ppm generate
```

from the "plankton" directory. This compiles the PPML client code "plankton.ppm" to standard Fortran2003 PPM client code. If you want, you can look at the auto-generated Fortran code in the "gen/" directory. Next, type

```
ppm build
```

to compile the Fortran code. You can find the resulting executable in the "bin/" directory. Start the simulation by:

```
cd bin
cp ../Ctrl .
mpirun -np 2 ./plankton
```

Change the number after "-np" to change the number of parallel MPI processes used in the simulation.