My story about installing and using ParFE

Doz. Dr. Dieter H. Pahr, TU-Vienna

12.7.2010

Contents

1 Foreword 2

2 Compile PARMETIS 2

3 Compile TRILINOS 3

4 Compile PARFE 4

5 Pre-Processing with ParFE 6

6 Run ParFE 7

7 Post-Processing with ParFE 8

8 Finial Statement 8

1 Foreword

Before you start do download, install, and use ParFE you should be sure that ParFE is what you are looking for. If you like to solve linear solid mechanics problems with some hundred millions degrees of freedom on a super computer with hundreds or thousands of CPUs you are right here. But if you like to solve standard FEM problems, ParFE might not be the right software.

I am working in the field of biomechanics and there we like to run such big models. So I downloaded/compiled and used ParFE. After a discussion with the current developer - Cyril Flaig - I decided to share my knowledge about ParFE with other new users. Thus, I summarized this "story". I indent to present all needed steps to be able to compile and run Parfe on your own PC or work station for test purposes. To do the same on your HPC is a different story.

Coming to the main topic what you need first is to install/compile some software which is needed by ParFE namely

- Parmetis and
- Trilinos

Usually before I start, I check which version of Parmetis, Trilinos and other software like c++ compilers, mpi versions, hdf libraries etc. is currently supported by the ParFE developers. That helps to avoid a lot of troubles.

2 Compile PARMETIS

After these checks you first need to compile Parmetis. Proceed in the following way:

- download the required version from www
- unzip it in a local directly where you have all rights to read/write files
- go to this local Parmetis directory and say "make" in a shell (sorry there is no GUI you can use, but shells are much nice after you got used to it)

Parmetis should compile rather fast.

3 Compile TRILINOS

As second step we move to Trilinos:

- download the required version from the www
- unzip it (same as before)
- setup a configure file

This is a simple text file. Here is my configure file (for ubuntu/kubuntu intrepid 64 bit):

```
#!/bin/bash
#------------ File myConfigure TRILINOS ---------------------------------------
rm -r LINUX
mkdir LINUX
cd LINUX
$HOME/software/trilinos-8.0.8/configure \
  --prefix=$HOME/software/trilinos-8.0.8/LINUX \
  --with-gnumake \
  --enable-mpi --with-mpi-compilers \
  --with-incdirs="-I/usr/include -I$HOME/software/ParMetis-3.1 -I$HOME/software/ParMetis-3.1/METISLib" \
  --with-ldflags="-L/usr/lib -L$HOME/software/ParMetis-3.1" \
  --with-libs="-lparmetis -lmetis" \
  --with-blas=-lblas \
  --with-lapack=-llapack \
  --with-flibs="-lg2c" \
  --disable-default-packages \
  --disable-tests \
  --disable-examples \
  --enable-aztecoo \
  --enable-epetra \
  --enable-epetraext \
  --enable-ifpack \
  --enable-ml \
  --enable-amesos \
  --enable-teuchos \
  --with-ml_metis \
```

Simply copy this content to a file (e.g myConfigure) and give the file executeable rights (`chmod u+x myConfigure`).

You see I created a directory `LINUX` inside the local trilinos directory, changed into this directory and run the configure (you don't have to make an own directory, but if you work on different machines its an advantage to have the compiled code in different folders). In my case the trilinos source is located in a directory named:

```
$HOME/software/trilinos-8.0.8
```

The `--prefix` says where the compiled trilinos version should be placed during installation. You see that at this point also the link to Parmetis is made (thus we have to compile Parmetis first). Maybe you have to install other libraries (e.g. lapack, blas, openmpi, etc) to be able to run the configure successfully. After you generated the configure file you say:

```
> ./myConfigure
```

in the local trilinos directory (`$HOME/software/trilinos-8.0.8`). The configure may take some time. After a successful configure you should find a directory named `LINUX` in the local trilinos directory. Go to this directory and say

```
> make
```

After quite some time the process should be ended successfully. After that you say in the shell:

```
> make install
```

This command copies all required libs to the directory selected with –prefix. In my case under `$HOME/software/trilinos-8.0.8/LINUX/lib`. There you find static libraries like `libamesos.a, libepetra.a`, ... If you have to recompile Trilinos you should always say:

```
> make clean
```

before you start the make process. If you could compile Trilinos successfully your are nearly done!

4 Compile PARFE

Please install the hdf5 library before you proceed. HDF5 is a special file format which allows you to read and write huge amounts of data in parallel.

Finally we will compile ParFE. The good news are that ParFE used the same configure and compile techniques as Trilinos. Please proceed in the following way:

- download Parfe from sourceforge (use svn to get the latest version, don't use the version in source forge, the svn link is given under source forge)

- you see that there is a trunk and branch (of Cyril) directory of ParFE. You properly need the trunk not the branch version.
- go into the trunk folder and proceed in the same way as with trilinos. Here is my configure file (I am sure you have to adopt it):

```
#!/bin/bash
#------------ File myConfigure for PARFE ------------------------------------
rm -r LINUX
mkdir LINUX
cd LINUX
$HOME/software/parfe/trunk/configure \
  --prefix="$HOME/software/parfe/trunk/LINUX" \
  --with-mpi-compilers \
CC=/usr/bin/mpicc \
CPP=/usr/bin/mpicxx \
CXX=/usr/bin/mpicxx \
FC=/usr/bin/mpif77 \
F77=/usr/bin/mpif77 \
  --with-mpi-incdir="/usr/include/mpi" \
  --with-parmetis="$HOME/software/ParMetis-3.1" \
  --with-incdirs="-I/usr/include -I$HOME/software/ParMetis-3.1  ....
    .... -I$HOME/software/ParMetis-3.1/METISLib -I$HOME/software/trilinos-8.0.8/LINUX/include" \
  --with-ldflags="-L$HOME/software/trilinos-8.0.8/LINUX/lib -L$HOME/software/ParMetis-3.1 -L/usr/lib" \
  --with-hdf5 \
  --with-libs="-lhdf5 -lml -lifpack -laztecoo -lamesos -lepetraext -lepetra -lteuchos"
```

after a successful configure you can say in the LINUX directory:

$>$ `make`

Note that everthing is compiled with the "`mpicxx`" compiler. This allows you to run the produced codes on more than only one CPU. We are nearly done. Go to `LINUX/examples` directory in the local ParFE directory. There you find several executeables:

- new_pfaim.exe
- pfaim.exe
- convert.exe

Usually you need "new_pfaim.exe". This is the new executeable to run ParFE.

5 Pre-Processing with ParFE

If you come from the FEM world you know that basic steps of an analysis are:

- Pre-Processing i.e. define mesh (nodes+elements), boundary conditions (forces, displacement), material parameters, and solution parameters.
- Solution, here done with ParFE, needs an input file and generates an output file
- Post-Processing i.e. look at stresses, strains, displacements, etc.

The input file for ParFE is the *.mesh.h5 file. You can find one in "..parfe/trunk/mesh" which is called "cube_s01_fea.mesh.h5". If you like to open the file with an editor - you will fail because it's a binary file (not an ASCII file). You can used either

- hdfdump (text based)
- hdfview (java GUI from HDF Group)

to open this file. Open this file with hdfview and you see that there are

- Parameters
- Mesh
- Boundary conditions
- Solution (may be only after analysis)

in the file. You can also see the corresponding values. But know you will recognize that you can not export this file format with any software even if you have a hdf5 writer. You have a voxel file (from your CT i.e. *.dicom, *.aim, . . .) and not a hdf5 file with elements, nodes, BCs, etc. in it. To get this file you need a pre-processor. I wrote a simple pre-processor by myself which can convert *.mhd file (VTK/Paraview image files) to *.mesh.h5 files. Please contact us/me if you are interested in the pre-processor.

As a further point you my recognize that there is already a "Solution" in the file. This means in case of ParFE the input file and the output file is the same file. If you generate this file by a self written Pre-Processor you do not need to add the "Solution". It will be added automatically (or updated) by ParFE after the run. This means that the model (mesh, material, BCs) and the solution (stress, strain, displacement, etc) are in the same file. At this time you may think this is stupid but the advantage is that you have to store the mesh only once (not in the input and output file as done by classical FEM solvers) which makes sense if you have input files with 10 GB and more!

6 Run ParFE

Lets start with the most interesting point. If you say

```
> ./new_pfaim.exe --help
```

you see several option how you can run your model. Now you are properly more confused than before. If you don't know what preconditioning means, stop here and read some literature (e.g. the 2008 paper of Peter Arbenz).

We start our first ParFE analysis with:

```
> mpirun -np 2 ../LINUX/example/new_pfaim.exe --precond=matrixfree --filename=cube_s01_fea.mesh.h5
```

On my machine I have open mpi installed and run the job using 2 CPUs in this example. I used the matrixfree preconditioner (because this uses a minimum of memory). Note that the solution in the hdf5 file will be overwritten (or newly generated) after the run. But how many CPU's do I need? The rule is the you should have 1 CPU (or core) per 1 Million DOF in the model. Ususally you need a machine with hundreds of CPUs but only few GB RAM per CPU (e.g. 2 GB).

I tried my first analysis on a classical FEM server with 4 CPUs and 64 GB RAM and got a numerical problem because some arrays within a CPU (node) are limited to 2 GB. So one trick to overcome this was to say

```
> mpirun -np 32 ../LINUX/example/new_pfaim.exe --precond=matrixfree --filename=...
```

This of course is not very efficient but you can utilize the 64 GB of this machine. Thus I could run 70 Mio DOFs on this machine within some hours!

7 Post-Processing with ParFE

Finally you like to look at the results of your analysis. There are two ways:

- look at them visually using Paraview (free visualizer based on VTK)
- reading data from the *.mesh.h5 file.

The graphical visualization with paraview needs the *.mesh.h5 file and an additional *.xdmf file which can be generated by a simple script (provided together with ParFE). If you like to visualize the files you have to set up a proper machine for the visualization with Paraview using mpi.

If you like to read directly values from the *.mesh.h5 file you usually have to write your on code based on hdf5 to parse/read/write data from the ParFE results file. The HDF5 library is well documented and there are a lot of examples in the web. But together with the GUI "hdfview" is a rather simple task.

8 Finial Statement

That's it! I know this description is not comprehensive and maybe not 100% correct. Tell me if you find major bugs/errors (`pahr@ilsb.tuwien.ac.at`).

Have Fun with ParFE - Dieter H. Pahr