Asterclient Documentation

Release 0

Martin Ortbauer

Sep 27, 2017

Contents

Asterclient is in it's usage very similar to asrun program. For a normal run you will need:

- a *Profile File*.yml file which holds the basic data about the calculations (it is not needed for a very simple run where specify all relevant data on the commandfile, this isn't supported up to now)
- several code aster *Commandfile*'s
- maybe some *Distributionfile* if you want to run a parametric study

CHAPTER 1

Motivation

The motivation for writing asterclient came out of some frustration since the asrun is documented quiet poorly specially on the topic of parametric studies. I thought it should be much easier and straight forward to run a paramtric study and in general a simple calculation.

CHAPTER 2

Usage

After succesful *Installation* you need at least the following to run a calculation[1]:

- profile.yml
- commandfile

For more details read the documentation of the *Profile File* file and and the *Distributionfile* file.

Asterclient currently has two commands available, `info` and `run`, where info can give you some information on your profile and run does the actual work, for more info just type `asterclient info -h` or `asterclient run -h`. The run command needs at least a profile specified, assumed you navigate into the examples `examples/basic/` directory you just need to type:

```
asterclient run -p profile.yml
```

Basic Example

For a full basic example see the examples directory example

[1] it is further assumed that you have a working code aster installation on your box

CHAPTER 3

Detailed Documentation

Profile File

The profile file is comparable to code aster export file used by asrun and astk. The difference to that format is basically it's format which has to be valid yaml, and is hopefully easier in it's data structure. The available keys are described below in detail. A full example profile file can be downloaded `here`.

`project`
: This specifies the projectname for the calculations and has only informative character and is optional.

`srcdir`
: Here you specify the source directory for the calculation, if you omit it it will point to the directory currently run the client and is therfore optional.

`outdir`
: Here you specify win which directory the calculation results should be saved to. Be careful, asterclient will overwrite files or directories if the have have same name as some results. If you specify a relative path it will be considered relative to the directory you run asterclient from.

`meshfile`
: The meshfile key specifies the path to the meshfile for the calculations, if relative to the *`srcdir`* or absolute.

`calculations`
: This is a list of all known calculations, for example some stress calculation or some fatigue calculation of the same project and the same mesh. Every calclation needs a name and a commandfile, fr example:

```
- name: "stress"
  commandfile: "stress.comm"
- name: "fatigue"
  commandfile: "fatigue.comm"
```

: This will provide to calculations named stress and fatigue with the associated commandfiles. If you want to run some calculation which needs some results of some other calculation as it's input you need to specify the poursuite key, for example:

```
- name: "post"
  commandfile: "post.comm"
  poursuite: "stress"
```

This would tell asterclient that the calculation `post` needs the results of the calculation `stress` as it's input, of course therefore you need first to calculate `stress` before you can calculate `post`.

name
The name of the calculation.

commandfile
The commandfile associated with the specified calculation.

resultfiles
A list of additional (in addition to the standard protocol output and glob.1 and pick.1) result files. You specify a file with a name and Logical Unit Number LU (see codeasterglossary under UNITE), for example:

```
- example.med: 80
- buckling.med: 81
```

Which would specify two files one with the name example.med and a LU numbe rof 80 and one with the name buckling.med and a LU number of 81. They could can now be refered to in the commandfile of the calculation for example like:

```
IMPR_RESU(FORMAT='MED',
        UNITE=81,
        RESU=....
        )
```

If you want to write some result files through python then you also need to add these files here other wise they won't get copied from the work directory to the result directory, you can also use globbing here. For example:

```
- protocol: ".rst"
```

would result into copying of the file `protocol.rst` from the working directory to the result directory.:

```
- protocol: "*.rst"
```

would result in copying all files starting with `protocol` and with the extension `rst` to the result directory, if there isn't any file found or if the file is empty you get a warning.

distributionfile
If you want to run a parametric study, which means that you have calculations which need basically the same commandfile but with different values, the you just specify a distributionfile with tis confval. The explanation on how the distributionfile needs to look like see *Distributionfile*. For information on how to use the specified parameters in the commandfile see *Commandfile*.

Distributionfile

The distributionfile is a simple python file which can contain any valid python code, but needs at least to provide a variable called `parameters` which is a list holding the various parameters for the parametric studies. The list must contain tuples with two entries, where the first entry is a string containing the name of that study and as second entry a python dict containing all parameters. It could for example look like:

```
#coding=utf-8

parameters = [
    ('study_A',{'a':3,'b':2}),
    ('study_B',{'a':1,'b':19})
    ]
```

In your commandfile you could acces these variables now through `params['a']` or `params['b']` respectively, assuming that you have specified the distributionfile correctly in the *Profile File*.

Commandfile

This is a completely standard code aster commandfile, nothing special, except that you can acces parameters of a parametric study like:

```
params['parameter']
```

See *Distributionfile* for more detailed information on how to specify the parameters and *Profile File* for information on how to specify a distributionfile.

An example of an advanced commandfile can be downloaded `here`. For much better documented examples see the caelinux wiki

CHAPTER 4

Installation

The installation is very easy, just download the *Source* and type:

```
python setup.py install
```

CHAPTER 5

Source

The sourcecode lives at github, feel free to fork mee as much as you like, feedback is appreciated.

Indices and tables

- genindex
- search

Index