Talking to PyFoam

And sometimes it talks back

Bernhard F.W. Gschaider

HFD Research GesmbH

Graz
6. July 2017

Outline I

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

What it's about

- This presentation has
  - No pictures
  - No results
- It is about some hardly known features of `PyFoam`
  - Tries to change this
- The features are
  - Talking to OpenFOAM-runs (that are controlled by PyFoam)
  - Finding these runs
  - The runs let you know that they're finished

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

Bernhard Gschaider

- Working with OpenFOAM™ since it was released
  - Still have to look up things in `Doxygen`
- I am not a core developer
  - But I don't consider myself to be an *Enthusiast*
- My involvement in the OpenFOAM™-community
  - Janitor of the `openfoamwiki.net`
  - Author of two additions for OpenFOAM™

swak4foam	Toolbox to avoid the need for C++-programming
PyFoam	Python-library to manipulate OpenFOAM™ cases and assist in executing them

  - In the admin-team of `foam-extend`
  - Organizing committee for the OpenFOAM™ *Workshop*
- The community-activies are not my main work but *collateral damage* from my real work at . . .

Heinemann Fluid Dynamics Research GmbH

The company

- Subsidary company of *Heinemann Oil*
  - Reservoir Engineering
  - Reservoir management

Description

- Located in Leoben, Austria
- Works on
  - Fluid simulations
    - OpenFOAM™ and Closed Source
  - Software development for CFD
    - mainly OpenFOAM™
- Industries we worked for
  - Automotive
  - Processing
  - ...

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

What is it

- PyFoam is a library for
  - Manipulating OpenFOAM-cases
  - Controlling OpenFOAM-runs
- It is written in Python
- Based upon that library there is a number of utilities
  - For case manipulation
  - Running simulations
  - Looking at the results
- All utilities start with `pyFoam` (so TAB-completion gives you an overview)
  - Each utility has an online help that is shown when using the `--help`-option
  - Additional information can be found
    - on `http://openfoamwiki.net`

The pyFoamRunner.py-command

- The most-used program in pyFoam is

```
pyFoamRunner.py --clear --progress --auto auto
```

- This runs an OpenFOAM-solver

auto	is a short-cut ■ checks the solver-entry in the controlDict ■ of course you can specify the solver directly
–auto	checks if this is a parallel case and adds mpirun=/-parallel=
–clear	removes old simulation results
–progress	only writes the current time to the console

- In addition the program
  - Captures the solver output and writes it to a logfile
  - Analyzes the output and writes it to disk
- The pyFoamPlotRunner.py also plots the analyzed data

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

pyFoamRunner.py is three programs

- People who read the `--help` of the utility may have noticed
  - There is a `--no-server-process-option`?
- Why?
- Because `pyFoamRunner.py` is actually three threads:
  1. the actual Python-script that controls everything
  2. the OpenFOAM-program that does the actual calculation
     - its output is read by the controller
  3. a network server that is eager to talk to the actual work
     - With so-called *Remote Procedure Calls*

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

Why a server?

- No need to log into the machine that runs the simulation
- What the server allows us to do
  - Get information about the simulation run
  - Control it (stop, force write etc)
  - Get information about the run (timestep, how long has it been running . . .)
  - Download additional stuff
    - Analyzed data
- To connect to the server one needs
  - The hostname
  - A portnumber
    - Starts with 18000 (or 18100 for secure connections)
    - If more than one server exists on the machine subsequent ports are assigned

Connecting to the server

- There is a program that opens an interactive shell

Write some data now

```
> pyFoamNetShell.py localhost 18100
Connected to server localhost on port 18100
62 available methods found
PFNET> time()
2.77554e-06
PFNET> writtenTimesteps()
['0']
PFNET> write()
True
PFNET> writtenTimesteps()
['0', '9.32435e-06', '9.45121e-06', '9.57934e-06', '9.70876e-06']
PFNET> help
For help on a method type 'help <method>'
Available methods are:
         actualCommandLine
         argv
         commandLine
         configuration
         controlDictUnmodified
         cpuSystemTime
```

Other uses of the net-shell

- To just send one command to the server use the --comman-option

```
pyFoamNetShell.py localhost 18100 --command="stopAtNextWrite()"
```

- The tells the server to stop the simulation the next time data is written
  - Of course you know how to do this with hand-editing `system/controlDict`
    - But this is more secure: a mistake while editing will crash the simulation . . . without writing

Another utility

- pyFoamRunner analyzes the output and collects the data
  - Writes it to disk
  - But doesn't plot it
- pyFoamPlotWatcher.py can process data from the log-files
  - but that is slow
  - and you need access to the machine
- pyFoamRedoPlot reads that data and generates plots

```
pyFoamRedoPlot.py --pickle-file PyFoamRunner.sonicFoam.analyzed/
```

- But for a running process we can ask the server for it

```
pyFoamRedoPlot.py --server localhost 18100
```

- This fetches the data and plots it locally

But how do we find the server ?

- There is a command for this
  - Which can be asked to report other stuff as well

Only two simulations in the network?

```
> pyFoamNetList.py --time --proc --resources
Searching    . + . + .   .   .  Done

Hostname     |  Port  | User        | Command Line
-------------------------------------------------
bgschaid-pc |  18101 |   bgschaid | simpleFoam
  PID:  14766   Working dir: /tmp/pitzDaily
  Time: 750 Timerange: [ 0 , 2000 ]  Mesh created: 0 -> Progress: 37.50% (Total: 37.50%)
  Started: 2017-Jul-03 21:04   Walltime:  112.077s  Estimated End: 2017-Jul-03 21:09
   Max memory: 203.531250 MB  Load 1m: 14.8 - 5m: 13.5 - 15m: 12.6
-------------------------------------------------
bgschaid-pc |  18100 |   bgschaid | sonicFoam
  CPUs:     4   Working dir: /tmp/OAnacaAirfoilState
  Time: 9.71274e-07 Timerange: [ 0 , 0.01 ]  Mesh created: 0 -> Progress: 0.01% (Total: <brk>
       <cont>0.01%)
  Started: 2017-Jul-03 21:05   Walltime:  36.4831s  Estimated End: 2017-Jul-08 05:25
   Max memory: 300.823242 MB  Load 1m: 14.8 - 5m: 13.5 - 15m: 12.6
-------------------------------------------------
```

Time estimates based on calculation time so far

But how did the utility find the servers?

- Through the technical wonder *ZeroConf*
  - Aka: `mDNS`, `Bonjour` (the Apple implementation), `avahi` (Linux)
  - It is a protocol that tries to automatically announce what is available on a network
- `pyFoam`-servers announce themselves on the network
  - can be easily found
    - check with `avahi-browse -a`
- Before that `PyFoam` needed a special server to collect the information
  - The so-called *Meta-Server*
    - Was hard to set up and therefor seldom used
  - Severs and utilities still fall back to this solution if `ZeroConf` doesn't work
- Disadvantages of `ZeroConf`
  - Doesn't cross subnets
    - ask your network-admin for help
  - Might be blocked by the personal firewall on your machine
    - ask your sys-admin

Not totally *zero configuration*

- To make communication secure the server adds two things
  - SSL-communication (to prevent wire-tapping)
  - A public key-authentication
    - to prevent Joe the Intern from killing your runs
- These things have to be set up
  - SSL needs a "server certificate"
    - If there is none `PyFoamRunner` prints the commands to generate one under Linux
  - Public and private key are usually set up automatically
    - If your are working on multiple machines that don't share `$HOME` you've got to distribute one set of `$HOME/.pyFoam/auth/privateKey` / `publicKey` to the others
    - If you want to allow other users to modify your jobs you've got to add a line `<username> <publicKey>` to = $HOME/.pyFoam/auth/myAuthenticatedKeys=
- You need a python library `zeroconf`
  - Check with `pyFoamVersion.py` if it is already there
  - Otherwise install it

```
pip install zeroconf
```

What is my public key

Information about the keys is reported by `pyFoamVersion.py`

`pyFoamVersion.py` knows a lot more than the version

```
> pyFoamVersion.py
...

User information
Username:               bgschaid
Temporary directory:    /tmp/PyFoam_bgschaid
Public key:             6a7856b80827e7d40da8ea4a28033c85:868bbfa95be75f0fd48407237f168e93

Authenticated keys
                test  : 6a7856b80827e7d40da8ea4a28033c85:868bbfa95be75f0fd48407237f168e93
```

- Private key not printed (that would be stupid)
- List of the public keys other people gave you

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

The configuration system

- For things that may differ on systems PyFoam allows to configure them
    - For instance "how to properly call `mpirun`-program"
    - Configurations are organized is sections (for instance `[MPI]`)
        - There can be version specific sections (special treatment for `mpirun` in OpenFOAM 7.8 could be found in `[MPI-7.8]`)
    - In the sections there are keys (for instance `options_openmpi_pre` for additional parameters for `mpirun`)
    - Values for the options can be numbers, strings or Python lists or dictionaries (depends)
- Locations where configurations are found are (also listed by `pyFoamVersion.py`)
    1. Hardcoded in the PyFoam-sources
    2. System-wide in `/etc/pyFoam/`
    3. User-specific in `$HOME/.pyFoam`
    4. Per-case in a file `LocalConfigPyFoam` in the case directory
- Highest number wins

Listing the configuration

What are the currently used Settings

```
> pyFoamDumpConfiguration.py
...
[Network]
allowselfsignedssl: True
nrserverports: 100
personalsslcertificate: /home/bgschaid/.pyFoam/foamServerCertificate.cert
portwait: 1.
privatesslkey: /home/bgschaid/.pyFoam/foamServerCertificate.key
socketretries: 10
sockettimeout: 1.
sslserverdefault: True
startserverport: 18000
startserverportssl: 18100
startserverthread: True
zeroconftimeout: 5.
....
```

These are some settings for the Servers in the last section

Where could we place configuration files?

There is a part in `pyFoamVersion.py` that describes this

Locations of settings

```
> pyFoamVersion.py
....
Path where PyFoam was found (PyFoam.__path__) is ['/home/bgschaid/PyFoam/PyFoam']

Configuration search path: [('file', '/etc/pyFoam/pyfoamrc'), ('directory', '/etc/pyFoam/<brk>
      <cont>pyfoamrc.d'), ('file', '/home/bgschaid/PyFoam/exampleSite/etc/pyfoamrc'), ('<brk>
      <cont>directory', '/home/bgschaid/PyFoam/exampleSite/etc/pyfoamrc.d'), ('file', '/home<brk>
      <cont>/bgschaid/.pyFoam/pyfoamrc'), ('directory', '/home/bgschaid/.pyFoam/pyfoamrc.d')<brk>
      <cont>]
Configuration files (used): ['/home/bgschaid/PyFoam/exampleSite/etc/pyfoamrc.d/helloHook.<brk>
      <cont>cfg']

Installed libraries:
cython                    :  No     Not used. Maybe will by used later to spped up <brk>
      <cont>parts of PyFoam
....
```

Site specific PyFoam-stuff

- Every organization has its own needs
  - Some need their own PyFoam-utilities and modules
  - Want one installation location for all of them
    - without spoiling the original PyFoam-installation
- Setting the environment variable `PYFOAM_SITE_DIR` points to this location

bin	additional scripts (add to `PATH`)
etc	settings
lib	this is special: can be used with `import PyFoam.Site` in scripts

This is also reported

```
> pyFoamVersion.py
...
Checking for PYFOAM_SITE_DIR : Location of non-PyFoam-disctributions script. Set and used <brk>
      <cont>by some Foam-distributions
PYFOAM_SITE_DIR set to /home/bgschaid/PyFoam/exampleSite
MISCONFIGURATION: no directory /home/bgschaid/PyFoam/exampleSite/bin for site-specific <brk>
      <cont>scripts
Site-specific configurations can be added to /home/bgschaid/PyFoam/exampleSite/etc
Site-specific library files can be added to /home/bgschaid/PyFoam/exampleSite/lib Do NOT <brk>
      <cont>add to PYTHONPATH but import as PyFoam.Site
```

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

Tell me when you're finished

- Sometimes it would be nice to get a notification when your run has finished
- For this `pyFoamRunner.py` executes *Hooks*
  - Little python-programs
    - Anything is possible
- It is easy to write your own hooks
- There are already predefined hooks that come with PyFoam
  - Sending mail
  - Contacting a webservice
    - With that you can leverage notification apps like *PushOver* or *PushBullet* to get notifications to your phone
  - Adding entries to a `SQLite` database
- To enable hooks you've just got to modify your configuration

How to specify a hook

- PyFoam has hardcoded example configuration for its hooks
- Adapt and put into a `cfg`-file in your configuration directory
- The section name identifies the hook
  - Start with `postRunHook_` for hooks to run when the simulation ended
    - `preRunHook_` when it starts
  - After that use a unique name
- There are some required entries

enabled	if this is not `True` the hook won't run
module	which type of hook is this
minRuntime	minimum time (in seconds) that the simulation should last before this hook is used (optional. Default: `0`)
stopOnError	should execution stop if there is an error (optional. Default: `False`. If there is a problem with the hook the run is not affected)

- Other parameters depend on the `module`

Example for a notification

Content of ~/.pyFoam/pyfoamrc.d/pushover.cfg

```
[postRunHook_SendToPushover]
enabled: True
header_content-type: application/x-www-form-urlencoded
host: api.pushover.net:443
method: POST
minruntime: 600
module: SendToWebservice
param_message: Case |-casefullname-| ended after |-wallTime-|s
Last timestep: t=|-time-|
Machine: |-hostname-|
Full command: |-commandLine-|
param_title: <!--(if OK)-->Finished<!--(else)-->Failed<!--(end)-->: |-casename-| (|-solver<brk>
      <cont>-|)
param_token: thisIsSecret
param_user: thisAsWell
templates: title message
url: /1/messages
usessl: True
```

This sends me a message every time a simulation that ran longer than 5 minutes ends

Writing your own hooks

- Hooks are Python-programs
  - Anything is possible
    - Play a sound
    - Add an entry to a database
    - ...
- Install into `lib` in `PYFOAM_SITE_DIR`
- For instance: `module` should later be `MyHook`:
  - A file `MyHook.py`
  - In it a class `MyHook`
    - Inherits from `RunHook`
- The class needs two methods

__init__	for initializing the hook
__call__	will be called when the hook is executed

- The class has a method `self.conf()` to access additional configuration data
- The variable `self.runner` gives us access to the script that ran the simulation
  - And all the data it gathered
- Everything else is up to you

An example hook

This hook prints that data PyFoam has gathered and prints a customized message

Content of $(PYFOAM_SITE_DIR)/lib/EchoHook.py

```
from __future__ import print_function

from pprint import pformat

from PyFoam.Infrastructure.RunHook import RunHook
from PyFoam.Basics.TemplateFile import TemplateFile
from PyFoam.ThirdParty.pyratemp import TemplateRenderError

class EchoHook(RunHook):
    def __init__(self,runner,name):
        RunHook.__init__(self,runner,name)
        print("Created",runner,name)
        self.message=self.conf().get("message")
    def __call__(self):
        print("Data:",pformat(self.runner.getData()))
        template=TemplateFile(content=self.message,
                              expressionDelimiter="|-",
                              encoding="ascii")
        print(template.getString(self.runner.getData()))
```

And using it

Content of $(PYFOAM_SITE_DIR)/etc/pyfoamrc.d/helloHook.cfg

```
[preRunHook_hello]
enabled: True
module: EchoHook
message: Starting up
stopOnError: True

[postRunHook_hello]
enabled: True
module: EchoHook
message: Did |-stepNr-| steps in |-casefullname-|
minRunTime: 10
```

Data from the runner is injected into the template string

Outline

1 Introduction
- This presentation
- Who is this?
- PyFoam

2 The server
- A server?
- A server!

3 The configuration
- Basics

4 The hooks
- PyFoam talks

5 The End

How to get this

- PyFoam can be easily installed
  - For everybody

```
sudo pip install PyFoam
```

- Just for you

```
pip install --user PyFoam
```

- Some features require additional libraries
  - Check with `pyFoamVersion.py`
  - Install with pip

```
sudo pip install zeroconf
```

- The version that does `ZeroConf` is not yet released
  - But will be before the weekend

Workshop in Exeter

And at last:

- 12th OpenFOAM-workshop in Exeter
  - 24.-26. July
  - It is in Europe. Sort of
  - Booking possible until the 9th of July
  - Program at `http://openfoamworkshop.org/index.php/at-a-glance/`
  - More than 100 presentations
  - More than 20 modules on Training day
  - Totally new: *3 minute Splash presentations*

Are you registered ?

License of this presentation