Jupyter notebooks: How do kernels work internally?

Florian Thöle

PyData Zurich

September 29, 2017

About me

- PhD student in Materials Theory at ETH Zürich
- Python/Jupyter notebooks are my workhorse for everything around large-scale quantum-mechanical simulation programs
- Likes to teach – SWC workshops, workshops in my research area, …

@florian_thl florian.thoele@gmail.com

Kernel Restarting

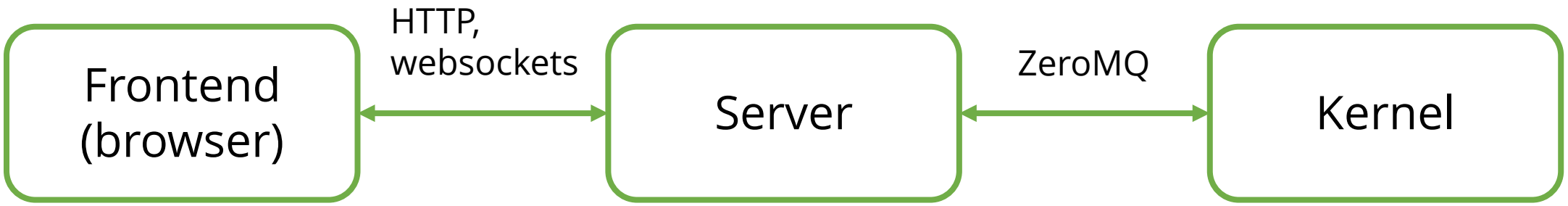
The kernel appears to have died. It will restart automatically.

OK

IPython vs. Jupyter

- Since version 4.0: Split between ipython and jupyter
- IPython: Provide a “more usable” python environment for interactive work
  - REPL and kernel mode
  - Environment on top of a python interpreter, such as CPython, PyPy, ..
- Jupyter: Client/server infrastructure, can use ipython as one possible kernel

Architecture of Jupyter



How does the server know about available kernels?

- KernelManager reads JSON files with kernelspec:

```
$ jupyter kernelspec list
Available kernels:
python3         /Users/thoelef/anaconda3/lib/python3.6/site-packages/ipykernel/resources
ir              /Users/thoelef/Library/Jupyter/kernels/ir
julia-0.4       /Users/thoelef/Library/Jupyter/kernels/julia-0.4
python2         /Users/thoelef/Library/Jupyter/kernels/python2
scala211        /Users/thoelef/Library/Jupyter/kernels/scala211
```

- nb_conda_kernels replaces the standard KernelManager with a custom one that “emulates” kernel confs for all conda environments

Example of KernelSpec file

- Usually, executes python program, but can be any other program that implements the messaging protocol

```
{
    "display_name": "Julia 0.4.5",
    "argv": [
        "/Applications/Julia-0.4.5.app/Contents/Resources/julia/bin/julia",
        "-i",
        "-F",
        "/Users/thoelef/.julia/v0.4/IJulia/src/kernel.jl",
        "{connection_file}"
    ],
    "language": "julia”
}
```

The connection file contains the sockets used for communication


```
{
 "shell_port": 63454,
 "iopub_port": 63455,
 "stdin_port": 63456,
 "control_port": 63457,
 "hb_port": 63458,
 "ip": "127.0.0.1",
 "key": "c78c3588-0ab3-408c-8bc3-55b9cd9ee291",
 "transport": "tcp",
 "signature_scheme": "hmac-sha256",
 "kernel_name": ""
 }
```

Jupyter message protocol

Messages are sent on five channels:

```
{
 'header' : dict,
 'msg_id' : str,
 'msg_type' : str,
 'parent_header' : dict,
 'content' : dict,
 'metadata' : dict,
}
```

- Shell
- Control
- IOPub
- stdin
- heartbeat

http://jupyter-client.readthedocs.io/en/latest/messaging.html

What does a kernel have to do?

- Jupyter Message Protocol
  - 5 different sockets
  - React to different message types from the server:
    - ```
      msg_types = [
          'execute_request', 'complete_request',
          'inspect_request', 'history_request',
          'comm_info_request', 'kernel_info_request',
          'connect_request', 'shutdown_request',
          'is_complete_request',
          # deprecated:
          'apply_request',
      ]
      ```

- Side note: there are also projects that do the communication via websockets or REST API, so that the kernel can run on a remote server (cloud), while the jupyter server is running locally
- Are there any good illustrations for the message protocol?
      ```

How to implement a custom kernel

- Write from scratch, have to start at socket level to send/receive message
  - Can be done in any language

- Use ipython module that takes care of communication
  - Good if python as base layer works
  - Can be used to control REPLs

Live demo!

Live demo script

- write kernelspec
- install with jupyter kernelspec install
- minimal setup with kernelapp start
- introduce do_execute status thingy
- do the send message to iopub line, echo code
- show bash_kernel github page

Examples of languages for which kernels are written

- IJulia, R, PySpark, JavaScript, Matlab, …
- C, Fortran, C#, Scala, …

https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

Summary

- Versatile architecture of Jupyter allows custom kernels for a variety of languages
- Effort to support a new language low when it can be controlled from python (pexpect), higher when implemented from scratch in native language