Pro Python

Ian Chen @ 資策會

2017-04-05

Agenda

- Production Python
- Concurrency and Python
- Python and Big Data Tools

Environments

- Get the slides/code
- Install Vagrant @ https://www.vagrantup.com
- Sample Data @ goo.gl/BxgeJs

- Open your favorite shell emulator(i.e. Cygwin)
- Change to the directory of the downloaded slides/code

```
vagrant plugin install vagrant-hostmanager
```

```
vagrant up
```

```
vagrant ssh
```

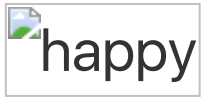
What is included

- Ubuntu 14.04 64bit
- CDH(Hadoop, Spark)

See all details in `bootstrap.sh`

Production Python

Learned how to write Python with Jupyter Notebook



In a production environment,

- Linux
- No GUI
- Version Controlled
- Multiple versions of Python
- Multiple versions of Python libraries

Installing Python in production

```
apt-get install python
```

```
yum install python
```

No No No

What's wrong with installing Python with OS package managers?

- Python versions shipped with OS are old
- Need administrator privilege
- Hard to manage multiple versions of Python

pyenv

https://github.com/pyenv/pyenv-installer

Make sure you have dependencies for building Python from source

```
# List available Python versions to install
pyenv install -l

# Install Python
pyenv install 3.6.1

# Set python version for system default
pyenv global 3.6.1

# Set python version for current directory
pyenv local 3.6.1
```

Advantages of using pyenv

- Manage and install multiple versions of Python
- No administrator privilege needed
- Manage python versions at different contexts

Let's talk about packages

```
pip install requests
```

- Installs packages from Pypi, git repository,etc..

Where are the packages installed?

What is the problem?

- What if we need different package versions for different projects?
- pip installs packages in global context by default

Virtual Environment

```
virtualenv
```

- Creates an isolated directory for python packages
- Was a addon package, become part of standard python packages in version 2.x

```
python -m venv venv
source venv/bin/activate
```

```
pip install requests
pip freeze
```

Manage project packages

```
pip freeze > requirements.txt
pip install -r requirements.txt
```

Logging

Concurrency and Python

- Modern CPUs advances towards more cores rather than higher clock speed
- In order to make our program run faster, we need to make our program utilize resources more *efficiently*
- Make our program concurrent

Concurrency vs Parallelism

Concurrency: Do multiple things in the same context

Parallelism: Separate one thing into multiple pieces and do them at the same time

There are two kinds of programs

- CPU bound
  - Matrix calculation, html parsing
- IO bound
  - Disk IO, network IO, database IO

CPU bound

- Use more cores
- Multi threading
- Multi processing

IO bound

- Use core(s) more efficiently
- Cooperative multi-tasking
- If you are not using the CPU, let someone else use it

Python is single threaded

- Global Interpreter Lock (GIL)

Multi Threading in Python

- Due to the GIL, we can't have real multi threading.
- Running multi threaded programs on multi-core machines actually may make your program slower

Beat the GIL

- multi processing
- forks the python interpreter into separate process
- memory is pickled to different processes

Cooperative multi-tasking

- Greenlets
- AsyncIO

Quick and Dirty cooperative multi-tasking

```
pip install gevent
```

https://github.com/gevent/gevent

Gevent monkey patch

```
from gevent import monkey
monkey.patch_all()
```

implicitly patches all sockets to async sockets

Now we run the program in the event loop

```
jobs = [gevent.spawn(get_page, _url) for _url in urls]

gevent.wait(jobs)
```

Scaling beyond a single machine

welcome to the land of distributed systems

Have a dedicated queue server to manage our tasks

- Redis(via Pub/Sub)
- RabbitMQ
- Kafka

```
pip install rq
```

http://python-rq.org

- Tasks are stored in Redis Queue
- Workers can be started on different machines, as long as they can talk to the Redis Server

Celery is another great library

http://www.celeryproject.org

Python and Big Data Tools

- MRJob
- PySpark
- Dask
- Blaze