regina Documentation

Release 1

Fujiao Liu

December 03, 2016

Contents

rezina is a scalable, distributed and easy to use system for executing python code in parallel across multiple processors or many machines.

rezina provides a simple way to make parallel programming in python more easier, flexible and scalable and shipped with features like periodically schedule, load balance, fail tolerate and dynamically add tasks .

more docs coming soon.

Contents:

CHAPTER 1

Installation

You can install reizna either via the Python Package Index (PyPI) or from source.

To install using pip:

```
pip install rezina
```

1.1 Downloading and installing from source

Before install rezina, building-and-installation pyzmq first.

After pyzmq installed, download the latest version of rezina from PyPI:

http://pypi.python.org/pypi/rezina

and install it by doing the following:

```
pip install /path/to/rezina-0.x.y.tar.gz
```

or

```
tar xvfz rezina-0.x.y.tar.gz
cd rezina-0.x.y
python setup.py install
```

CHAPTER 2

periodically schedule

2.1 Interval

when use `interval` in `tb.restart` or `tb.start` like this `tb.restart(interval=10)`

it will run tasks periodically with given interval, the unit of interval is `seconds` .

if it is omitted, the typology will run only once no matter what `start_time` is.

2.2 Start time

start_time option controls when to start typology for the first time.

start_time is a time string with format '%Y-%m-%d %H:%M:%S' (2016-12-03 23:18:19)

when start_time used in `tb.restart` or `tb.start` like this `tb.restart(start_time="2016-12-03 20:18:03")`

it means the tpyololgy will start to run at "2016-12-03 20:18:03".

2.3 Default value of start_time

actually, every typoloy has a start_time, if start_time is omitted, the default value is used.

condition one

if interval is given and start_time is ommited, the default value is `math.ceil(time.time() / interval) * interval`,

for example:

presume the time we start typology is `2016-12-03 20:18:03`.

if interval is 10, the start_time would be `2016-12-03 20:18:10`

if interval is 5, the start_time would be `2016-12-03 20:18:05`

if interval is 60, the start_time would be `2016-12-03 20:19:00`

condition two

if start_time and interval both are omitted,

the start_tiem will be `now` and run only once

conditon three

if interval is given and start_time is less than `math.ceil(time.time() / interval) - 1) * interval`, it will be this value, this prevent re-run old task when typology restart.

for example:

presume the time we start typology is `2016-12-03 20:18:03`.

if start_time is '2016-11:11 12:30:21' and interval is 10,

the start_time would be `2016-12-03 20:18:00`, this will run immediately and the second run will be at `2016-12-03 20:18:10`

if interval is not given, start_time will be its value and just run only once.

CHAPTER 3

Startup setttings

3.1 Master

`rezina-cli runmaster`

options:

`-H or --host`, the ip of machine running rezina master, the default value is IPv4 address of fully qualified domain name, If name is omitted or empty, it is `127.0.0.1`.

`-P or --port`, master_port, **12345** by default.

`-D`, run master as a daemon process, example: `rezina-cli runmaster -D`

`-L or --log_dir`, rezina log directory, you should use **absolutely path** for this option. every typology has its own log file which name is the same with typology name, you could find the error which tells why typology does not run correctly. it is `~/rezina/log` by default.

`-W or --worksapce`, workerspace directory. you should use **absulotely path** for this option. `~/rezina/workerspace` by default. when we run a python function in a module with rezina, it is actually running on rezina workers, therefore workers must have the module and then import the function and run it.

To do this, rezina will send all files under workspace directory to workers, so you should put python files into workspace directory. but this does not mean we need put all depandencies into workspace, if you imported some third-part python libs in your module, there is no need to put them into workspace too, just make sure all workers also installed these libs and can be imported by python. when typology run, workers will import those libs as python dose and run your function.

`-HP or --http_port`, the port for access web console, **31218** by default, after master started, you could open broswer and go to master_ip:31218 to see the web console.

`-R` refresh (or recreate) DB file, it is `False` by defaut, this is a **error prone** option, rezina master is a sevice set, and every service need a tcp address for communcating with workers, after the first time rezina master started, actually all tcp_address of services include master_ip and master_port are stored in db, if rezina master stoped (killed by accident or poweroff), when we re-run master without -R option, it will use those saved tcp_addresses and then master can still talk to workers.

If your really want change master_ip and master_port, **stop all workers first and restart master with -R option**.

this option is only effect tcp_address of service, the other options(except master_ip and master_port) take effective every time re-run master

3.2 Worker

`rezina-cli runworker`

options:

`-H or --host`, master_ip

`-P or --port`, master_port. (default 12345)

`-WIP or --worker_ip`, worker_ip, this is the ip of machine used to connect master.

`-D` run worker as a daemon process, it is False by default

3.3 Console

`rezina-cli runconsole`

options:

`-H or --host`, master_ip

`-P or --port`, master_port (default 12345)

you could use console to see settings

start console with `rezina runconsole`

run `list settings` in console

3.4 examples

single machine

```
rezina-cli runmaster -D
rezian-cli runworker -D
rezina-cli runconsole
```

multi-workers

```
rezina-cli runmaster -H 192.168.1.100 -P 11111 -D -L /path/to/log -W
/my/exist/dir/contain/python
rezina-cli runworker -H 192.168.1.100 -P 11111 -D -WIP 192.168.1.101
rezina-cli runworker -H 192.168.1.100 -P 11111 -D -WIP 192.168.1.102
rezina-cli runworker -H 192.168.1.100 -P 11111 -D -WIP 192.168.1.103
rezina-cli runconsole -H 192.168.1.100 -P 11111
```

CHAPTER 4

run python code with rezina

4.1 Start master and workers

4.1.1 In single machine

start master with

```
rezina-cli runmaster -D
```

start worker with

```
rezina-cli runworker -D
```

4.1.2 Multi-machines

start master with

```
rezina-cli runmaster -H master_ip -D
```

start worker with

```
rezina-cli runworker -H master_ip -WIP worker1_ip -D
rezina-cli runworker -H master_ip -WIP worker2_ip -D
```

4.2 Write python code and put it in workspace

```
cd ~/rezina/workspace
touch cityweather.py
```

sourc code:

```
#!/usr/bin/evn python


import urllib2
import urllib
import json


def get_cities():
```

```
    cities = ['Beijing', 'Berlin', 'New York', 'London', 'Tokyo', 'Paris',
              'Chicago', 'washington', 'Venice', 'Houston']
    return cities


# get city weather data from yahoo weather api
def get_city_weather(city):
    baseurl = "https://query.yahooapis.com/v1/public/yql?"
    yql_query = "select item.condition.text from weather.forecast \
                 where woeid in (select woeid from geo.places(1) \
                 where text='%s')" % (city)
    yql_url = baseurl + urllib.urlencode({'q': yql_query}) + "&format=json"
    result = urllib2.urlopen(yql_url).read()
    data = json.loads(result)
    # because resule from yahoo api does not include the city name, we add it.
    data['city'] = city
    return data


# process diffrent output and convert data to a simple format
def one_word_conditions_for_city(city_weather_result):
    simple_format_data = {}
    simple_format_data['city'] = city_weather_result['city']
    if city_weather_result['query']['results'] is not None:
        weather = city_weather_result['query']['results']['channel']['item']['condition']['text']
    else:
        weather = "Unkonw"  # simplely set unkonw when result is not avaliable
    simple_format_data['weather'] = weather
    return simple_format_data

if __name__ == "__main__":
    for city in get_cities():
        print one_word_conditions_for_city(get_city_weather(city))
```

4.3 Write typology to run this code in parallel

```
cd ~/rezina/workspace
```

```
touch weathertypo.py
```

source code

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from rezina.utils.network import get_ip
from rezina import TypologyBuilder
from rezina.backends import Stdout

from cityweather import get_cities, get_city_weather, one_word_conditions_for_city

ip = get_ip() # change to your master_ip
tb = TypologyBuilder(ip, 12345, 'weather_typo')
tb.add_hydrant(get_cities).add_notch(get_city_weather, 1, 10)
tb.add_notch(one_word_conditions_for_city, 1, 1)
tb.add_bocca(Stdout, persistent_mode='stream')
```

```
if __name__ == "__main__":
    tb.restart(interval=10)
```

note: replace `ip = get_ip()` to `ip = your_master_ip` if master_ip is given when start master.

4.4 Run code with rezina

run code with

```
python weathertypo.py
```

note: it will run the code but not immediately, it will run like this, presume the time your run the script is `2016-12-03 20:18:03`, the first time run is at `2016-12-03 20:18:10` and the second run is at `2016-12-03 20:18:20` and next.

if you want run it immediately, use start_time like this:

```
tb.restart(start_time="2016-12-03 20:18:03", interval=10")
```

the first run will be at `2016-12-03 20:18:03` and the second is at `2016-12-03 20:18:13`

see *periodically schedule*

Press `ctrl-c` to stop.

4.5 manage typology

you could use console or web console to manage typolog, include start, stop restart remove, launch more process for one task.

```
rezina-cli runconsole -H master_ip
```

access http://master_ip:31218 in broswer.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search