

www.qconferences.com
www.qconbeijing.com

QCon

伦敦 | 北京 | 东京 | 纽约 | 圣保罗 | 上海 | 旧金山

London · Beijing · Tokyo · New York · Sao Paulo · Shanghai · San Francisco

QCon全球软件开发大会

International Software Development Conference

搜狐邮箱的Python经验

彭 一

yipeng@sohu-inc.com

今天的话题

- 优雅地发布Python项目
- 优秀的WEB框架-Tornado
- 加速你的Python代码
- 简化你的C代码

优雅的发布Python项目

抓狂的现象

- 第3方依赖多，下载速度慢
- 内网服务器不能连外网
- 老版本不再提供下载
- GFW



pypiserver

- 是用来搭建本地python package的工具
- 优点

- 轻量简单，易安装

<https://pypi.python.org/packages/source/p/pypiserver/pypiserver-1.1.0.zip>

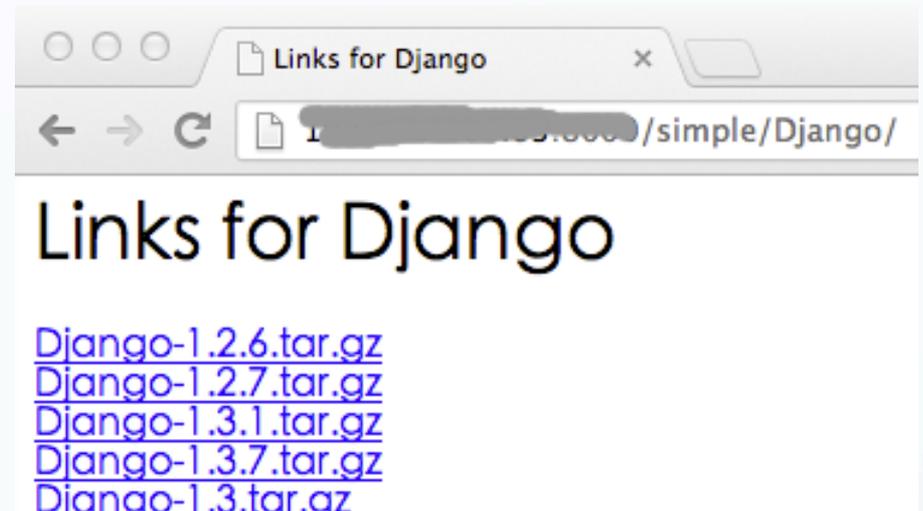
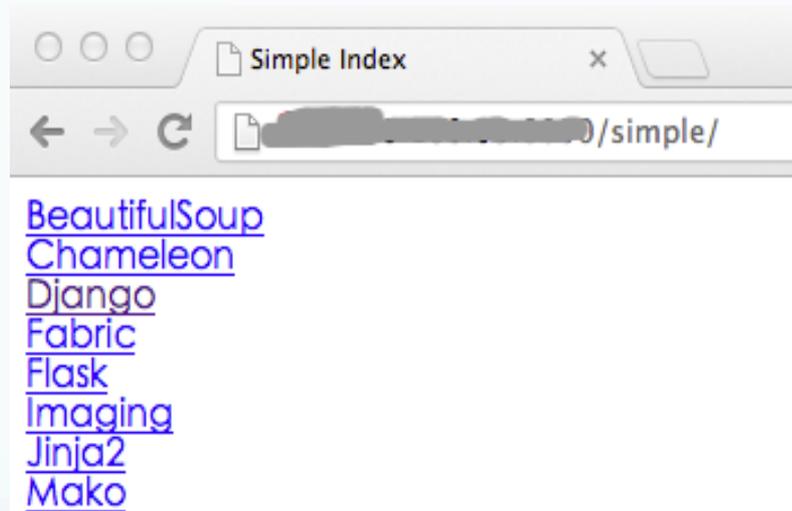
- 方便使用

```
gunicorn -w4 -b0.0.0.0:8080 'pypiserver:app( "/var/www/pypidir" )' -D
```

- package方便管理

```
pip install --no-install Django -d /var/www/pypydir
```

pypiserver



fabric

- 一个用python实现的，用于项目的推送的工具，类似于ruby中的capistrano
- 优点
 - 加强版SSH
 - 支持本地和远程操作
 - 参数灵活
 - 完整的日志输入

fabric-example

- fabfile.py

```
1 #coding: utf8
2 import demo
```

- demo.py

```
1 #coding: utf8
2 from fabric.api import env, local,
3                               task, run
4 env.user = 'yipeng'
5
6 @task
7 def test():
8     env.hosts = ['192.168.160.63']
9
10 @task
11 def ls(keyword):
12     run('ls %s' % keyword)
13
```

fabric-example

- 执行

```
PengYi-MacBook-Pro:~ ryan$ fab demo.test demo.ls:*.zip
[192.168.160.63] Executing task 'demo.ls'
[192.168.160.63] run: ls *.zip
[192.168.160.63] out: avmilter.zip  centerdata2_just_mysql_0.1.zip
[192.168.160.63] out:

Done.
Disconnecting from 192.168.160.63... done.
```

- 更多漂亮的DSL

- fab Project.production git.tar:v1.1 common.deploy
- fab Project.production common.rollback
- fab Project.production common.health:ProjectName
-

优秀的WEB框架-Tornado

代码简单

```
from web import application

class Index:

    def GET(self):
        return 'Hello, GET ! \n'

    def POST(self):
        return 'Hello, POST ! \n'

urls = ( '/', 'Index' )

if __name__ == '__main__' :
    app = application(urls, globals())
    app.run()
```

```
import tornado.ioloop
from tornado.web import RequestHandler
from tornado.web import Application

class Index(RequestHandler):

    def get(self):
        self.write('Hello, GET ! \n' )

    def post(self):
        self.write('Hello, POST! \n' )

urls = ( ('/', Index), )

if __name__ == '__main__' :
    app = Application(urls).listen(8000)

    tornado.ioloop.IOLoop.instance().start()
```

性能优秀

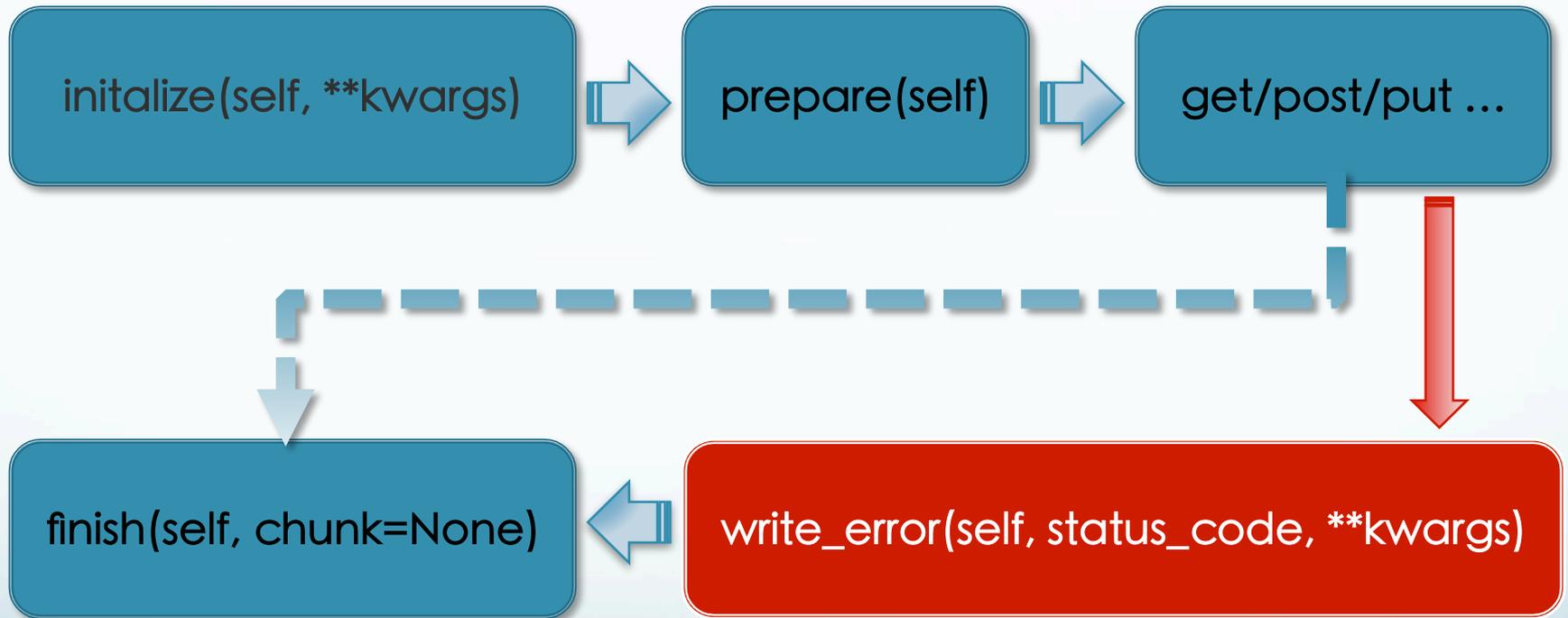
- 硬件配置：2*CPU 8G Mem 100Disk
- 2k并发，5w访问

部署	QPS	Success
2*Nginx + 4*Tornado	4402	100%
2*Nginx + flup + 4*Django	4207	88%
2*Nginx + flup + 4*webpy	无法响应	

成熟

- v3.0
- 案例
 - <http://friendfeed.com>
 - <http://www.zhihu.com>

RequestHanlder生命周期



异步请求

```
import tornado.web as web

class AsyncHandler(RequestHandler):

    @web.asynchronous
    def get(self):

        url = 'http://example.com'

        hc = AsyncHTTPClient()

        hc.fetch(url, callback=self.on_fetch)

    def on_fetch(self, response):

        do_something_with (response)

        self.render('template.html')
```

```
import tornado.web as web
import tornado.gen as gen

class GenAsyncHandler(RequestHandler):

    @web.asynchronous
    @gen.engine
    def get(self):

        url = 'http://example.com'

        hc = AsyncHTTPClient()

        response = yield hc.fetch(url)

        do_something_with (response)

        self.render('template.html')
```

Connection

- 基于MySQLdb封装

- 使用技巧

- 全局

```
db = Connection(host = 'localhost:3306', database =  
'centerdata', user = 'root', password = '')
```

- 防止 mysql server gone away

```
ping_db = lambda: db.query('show variables' )
```

```
tornado.ioloop.PeriodicCallback(ping_db, 600 * 1000).start()
```

Tornado+Django

Tornado的高性能



Django的ORM



Django的Admin

创建项目结构

- `django-admin.py startproject Demo`
- `cd Demo`
- `python manage.py startapp app`
- `touch application.py`
- `mv app/views.py app/controllers.py`

```
Demo
|-- __init__.py
|-- __init__.pyc
|-- app
|   |-- __init__.py
|   |-- controllers.py
|   |-- models.py
|   `-- tests.py
|-- application.py
|-- manage.py
|-- settings.py
|-- settings.pyc
`-- urls.py

1 directory, 11 files
```

修改settings.py

```
ADMIN_MEDIA_PREFIX = '/admin/media/'
```

```
INSTALLED_APPS = (  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.sites',  
    'django.contrib.admin',  
    'app'  
)
```

修改urls.py

```
1 from django.conf.urls.defaults import patterns, include
2 from django.contrib import admin
3 import django
4 import os
5
6 admin.autodiscover()
7 DJANGO_PATH = os.path.dirname(django.__file__)
8 MEDIA_ROOT = os.path.join(DJANGO_PATH, 'contrib', 'admin', 'media')
9 urlpatterns = patterns('',
10     (r'^admin/media/(?P<path>.*)$', 'django.views.static.serve',
11         {'document_root': MEDIA_ROOT, 'show_indexes': False}
12     ),
13     (r'^admin/', include(admin.site.urls)),
14 )
```

修改application.py

```
1  #coding: utf-8
2  import os
3  os.environ['DJANGO_SETTINGS_MODULE'] = 'settings'
4
5  from django.core.handlers.wsgi import WSGIHandler
6  from tornado import options, wsgi
7  import tornado.netutil as netutil
8  import tornado.httpserver
9  import settings
10
11  options.define('app_port', default=9201, type=int, help='Run on the given port')
12  options.define('admin_port', default=9200, type=int, help='Run admin platform on the given port')
13
14  if __name__ == "__main__":
15      options.parse_command_line()
16      address, app_port, admin_port = '0.0.0.0', options.options.app_port, options.options.admin_port
17
18  def app_listen():
19      import app.urls
20      application = tornado.web.Application(app.urls.urls, **settings.tornado_env)
21      server = tornado.httpserver.HTTPServer(application)
22      server.listen(app_port)
23      print 'run application on (%s:%s)' % (address, app_port)
24
25  def admin_listen():
26      wsgi_app = wsgi.WSGIContainer(WSGIHandler())
27      tornado_app = tornado.web.Application([
28          ('/admin/(.*)', tornado.web.FallbackHandler, dict(fallback=wsgi_app)),
29      ])
30      tornado.httpserver.HTTPServer(tornado_app).listen(admin_port)
31      print 'run admin platform on (%s:%s)' % (address, admin_port)
32
33  admin_listen()
34  app_listen()
35
36  tornado.ioloop.IOLoop.instance().start()
```

事务处理

利用django的TransactionMiddleware

```
3 class TransactionMiddleware(object):
4     """
5     Transaction middleware. If this is enabled, each view function
6     will be run with commit_on_response activated – that way a
7     save() doesn't do a direct commit, the commit is done when
8     a successful response is created. If an exception happens,
9     the database is rolled back.
10    """
11    def process_request(self, request):
12        """Enters transaction management"""
13        transaction.enter_transaction_management()
14        transaction.managed(True)
15
16    def process_exception(self, request, exception):
17        """Rolls back the database and leaves transaction management"""
18        if transaction.is_dirty():
19            transaction.rollback()
20        transaction.leave_transaction_management()
21
22    def process_response(self, request, response):
23        """Commits and leaves transaction management."""
24        if transaction.is_managed():
25            if transaction.is_dirty():
26                transaction.commit()
27            transaction.leave_transaction_management()
28        return response
```

自动事务处理

```
class BaseController(object):
```

```
    def prepare(self):
```

```
        transaction.enter_transaction_management()  
        transaction.managed(True)
```

```
    def write_error(self, status_code, **kwargs):
```

```
        if transaction.is_dirty():  
            transaction.rollback()  
        transaction.leave_transaction_management()
```

```
    def finish (self, chunk=None):
```

```
        if transaction.is_managed():  
            if transaction.is_dirty():  
                transaction.commit()  
            transaction.leave_transaction_management()
```

一些思考

- 数据库是否要异步?
- 有耗时长长的同步操作?
- 存在大文件上传?

加速你的Python

例子-Python

```
def my_concat():  
    num = 20000000  
    i = 0  
    s1 = 'abcdefghijklmnopqrstuvwxy'  
    s2 = '0123456789'  
    s3 = ''  
    while i < num:  
        s3 = '%s|%s' % (s1, s2)  
        i += 1
```

```
my_concat()
```

例子-纯C

```
void my_concat() {
    int num = 20000000;
    int i = 0;
    char* s1 = "abcdefghijklmnopqrstuvxyz";
    char* s2 = "0123456789";
    char s3[256];
    while (i < num) {
        sprintf(s3, "%s|%s", s1, s2);
        i += 1;
    }
    printf("concatStrings: %s\n", s3);
}
```

swig

- example.h
- example.c

```
void my_concat();
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "example.h"
```

```
void my_concat() {
    int num = 20000000;
    int i = 0;
    char* s1 = "abcdefghijklmnopqrstuvxyz";
    char* s2 = "0123456789";
    char s3[256];
    while (i++ < num) {
        sprintf(s3, "%s|%s", s1, s2);
    }
    printf("concatStrings: %s\n", s3);
}
```

swig

- example.i

```
1 // python要import的package name
2 %module example
3 ▼ %{
4     // 里面写要用的头文件
5     #include "example.h";
6 ▲ %}
7 // 外面写要给python暴露的方法
8 void my_concat();
```

- 编译

- swig -python example.i # 生成example.py 和 example_wrap.c
- gcc example.c example_wrap.c -fPIC -I/usr/include/python2.7/
-shared -o example.so # 生成so文件, 注意一定要下划线开始

- 执行

```
[root@mx9 py]# time python -c'import example; example.my_concat()'
concatStrings: abcdefghijklmnopqrstuvwxyz|0123456789

real    0m5.863s
user    0m5.860s
sys     0m0.004s
```

ctypes

- pointer.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void add(int* a, int* b, int* r) {
5     *r = *a + *b;
6 }
```

- 编译成so

```
gcc pointer.c -fPIC -shared -o pointer.so
```

- python调so

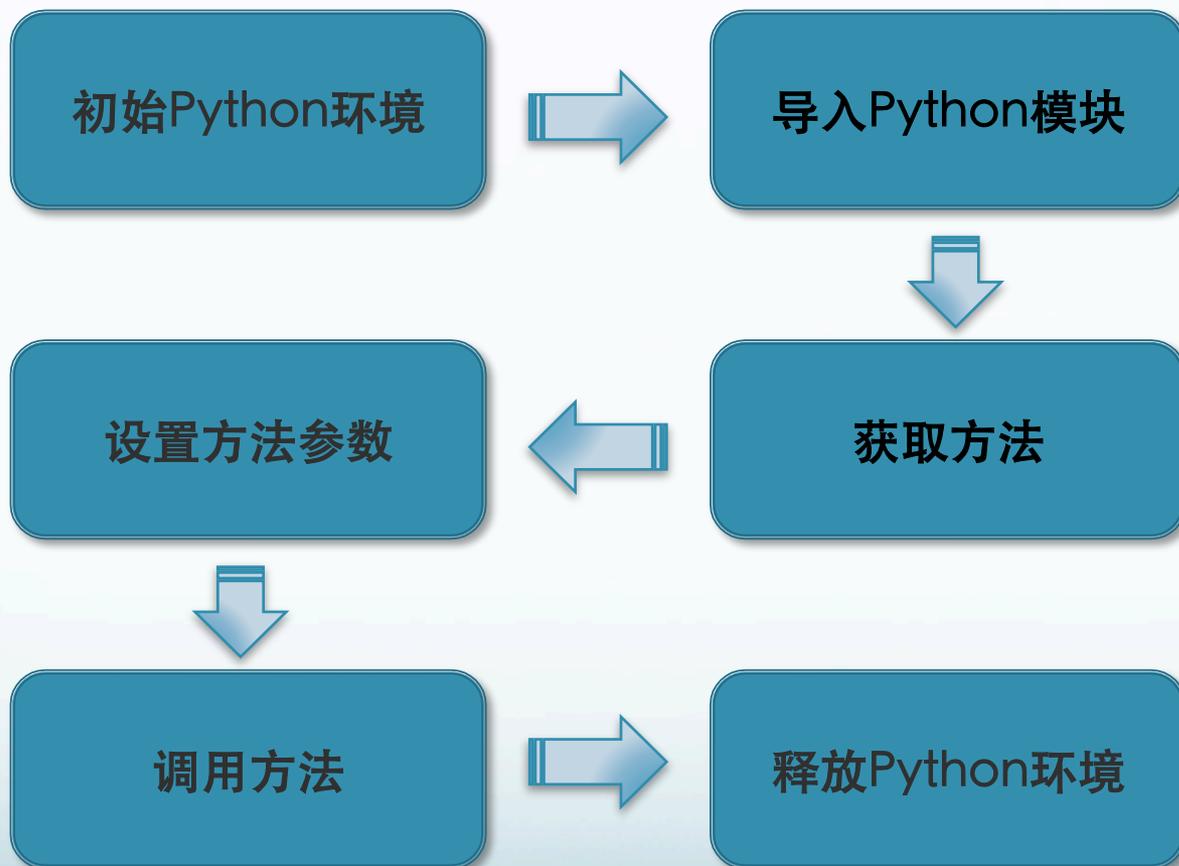
```
1 #coding: utf8
2 import os
3 from ctypes import c_int, cdll, POINTER, byref
4
5 cur_path = os.path.dirname(__file__)
6 so_path = os.path.join(cur_path, 'pointer.so')
7 so_path = os.path.normpath(so_path)
8 pointer = cdll.LoadLibrary(so_path)
9
10 a, b, r = c_int(10), c_int(20), c_int(0)
11 pointer.add(byref(a), byref(b), byref(r))
12 print r.value
```

简化你的C代码

应用场景

- 修改一个第3方的C库
- 业务复杂的C代码
- 业务功能、程序稳定 为主要要求
- 性能可以有所损耗

C调用Python步骤



例子-demo.py

```
def deliver(ip, sender, receiver, data):
```

```
    """ ip, sender 是否有效, 是否在黑名单;
```

```
        recevier 是否有效, 是否设置转发,
```

```
            是否自动回复;
```

```
        data 是否有违禁内容
```

```
        ... 其它复杂的业务逻辑
```

```
    """
```

```
    # 非常复杂的业务逻辑实现代码
```

例子-C代码

```
1 // 导入python的头文件
2 #include <Python.h>
3
4 int deliver(char* ip, char* sender,
5             char* receiver, char* data ) {
6     int has_err = 1;
7     Py_Initialize(); // 初始化python环境
8
9     PyObject* module = NULL, *func = NULL, *args = NULL;
10
11     if (!(module = PyImport_ImportModule("demo")) // 导入demo这个module
12         || !(func = PyObject_GetAttrString(module, "deliver")) // 得到deliver方法
13         || !(args = Py_BuildValue("sss", ip, sender, receiver, data)) // 创建参数
14         ) {
15         goto Err;
16     }
17
18     PyObject_CallObject(func, args) // 调用函数
19     has_err = 0;
20 Err:
21     if (module) Py_DECREF(module); // 引用计数减1
22     if (func) Py_DECREF(func);
23     if (args) Py_DECREF(args);
24     if (has_err) PyErr_Print(); // 输入错误
25
26     Py_Finalize(); // 释放python环境
27     return 0;
28 }
```

其它用法

- NULL写成Py_None

- Py_DECREF 和 Py_XDECREF

PyObject* p = Py_None; if (p) Py_DECREF(p) ⇔ Py_XDECREF(p)

- 设置tuple类型的参数

PyObject* py_tuple = PyTuple_New(size)

PyTuple_SetItem(py_tuple, idx, v)

- 设置dict类型的参数

PyObject* py_dict = PyDict_New()

PyDict_SetItem(py_dict, k, v)

回顾

- 优雅的发布Python项目
 - pypiserver
 - fabric
- 优秀的WEB框架-Tornado
 - 基本用法
 - 与Django相结合
- 加速你的Python代码
 - swig
 - ctypes
- 简化你的C代码

谢谢



@InfoQ



infoqchina

软件
正在改变世界!