PyCon China 2016

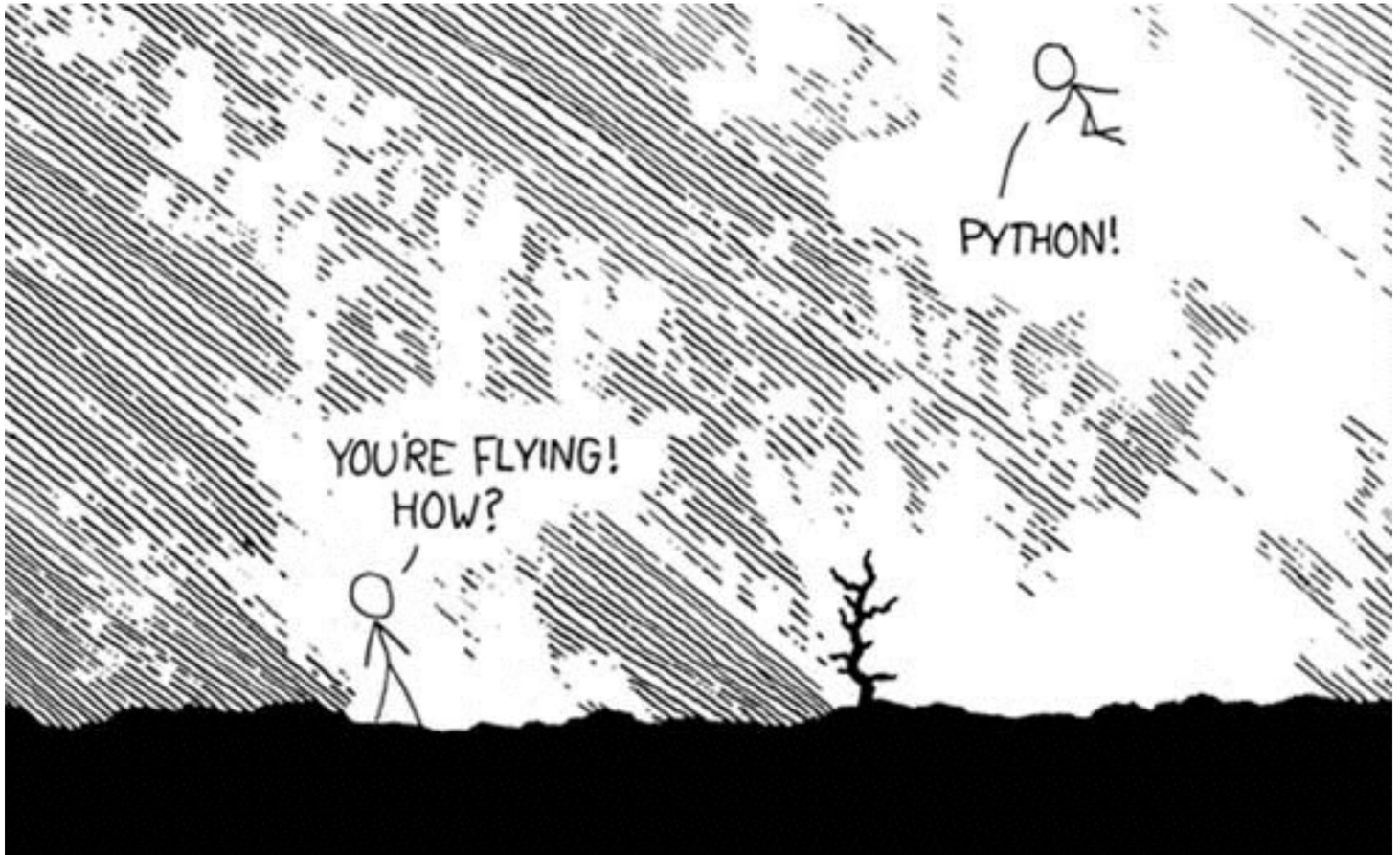
腾讯云

腾讯云的Python实践

腾讯云布道师/CVM技术负责人 李力

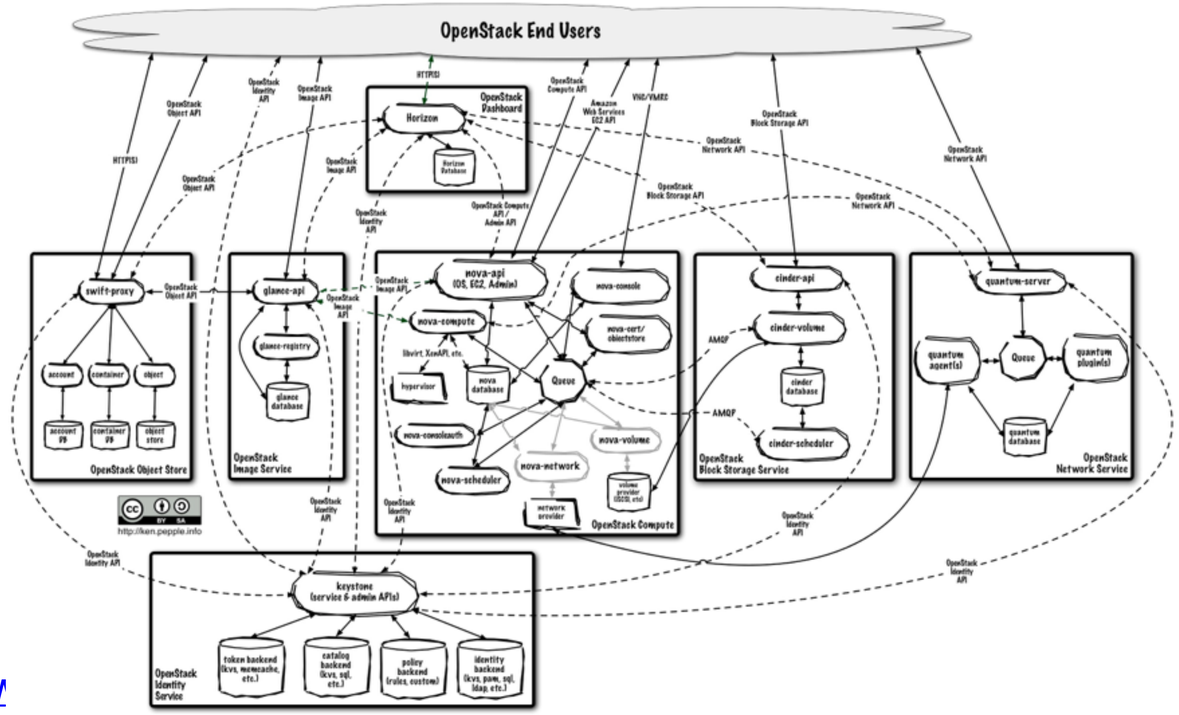
我和Python

- 2007年：开始接触Python
- 2012年：在腾讯云推广Python



公有云和Python

- 公有云
  - 堆叠引入复杂性
  - 快速迭代与稳定性的矛盾
  - 大量IO密集型的服务
  - Linux系统编程和网络编程
- Python
  - 快
- 引用
  - https://www.zhihu.com/question/40635350/answ
  - https://www.zhihu.com/question/34511860/answer/104123438



从C++到Python：语言特性

- Python和C++都与Linux天然贴合
  - 系统编程API与glibc保持一致
  - strace/gdb/lsof工具箱仍然适用
- Python是能够运行的伪代码
  - 开发效率 vs 运行效率
- Python容易与C/C++交互
  - Python.h/ctypes
- Python更容易保持项目整洁
- ……
- 可以早点下班

从C++到Python：数据结构

Python类型	类似的C++类型
int	long
long	(大整数，C++0x N1744提案，未通过)
bool	bool
float	double
complex	Boost::Math
str	std::string
list	std::vector<PyObject*>
tuple	std::tuple<PyObject*…> (C++11)
dict	__gnu_cxx::hash_map<PyObject*, PyObject*>
set	__gnu_cxx::hash_set<PyObject* >
type	-
None	NULL
file	FILE*

从C++到Python：代码转换

```
CSomeType items = getitems(args);
CSomeType results;
for (CSomeType::iterator it=items.begin(); it != items.end(); ++it)
{
    if (condition(*it))
    {
        results.push(process(*it));
    }
}
```

```
results = [process(item) for item in getitems(args) if condition(item)]
```

```
CSomeType items = getitems(args);
CSomeType result;
std::remove_copy_if(items.begin(), items.end(),
        back_inserter(result), not_condition);
std::for_each(result.begin(), result.end(), process);
```

从C++到Python：调试

```
[root@kvm_ ~]# ps -aux | grep moa
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
root     22764  0.0  0.0  97892  7792 ?        S    Oct28   0:31 m      t          -d
root     22765  0.0  0.0 171764  7896 ?        Sl   Oct28   3:16 m      t          -d
root     22766  0.2  0.0 326516 10644 ?        Sl   Oct28  23:55 m      t          -d
root     22767  2.8  2.3 2014648 1522896 ?     Sl   Oct28 328:35 m      t          -d
root     37617  0.0  0.0   6428   592 pts/10   S+   17:29   0:00 grep moa
[root@kvm_ ~]#
```

```
# cat /proc/22767/smaps
# gdb -p 22767
> x/1000c 0x7fc4e95001f0
```

```
0x7fc4e95001f0:  "<domain type='kvm'>\n  <name>bc507d48-58be-4a67-8641-df75cb53221e</nam
rrentMemory unit='KiB'>1048576</c"...
0x7fc4e95002b8:  "urrentMemory>\n  <vcpu placement='static' cpuset='6-11,18-23'>1</vcpu>
9325364d73b</entry>\n        <entr"...
0x7fc4e9500380:  "y name='uuid'>bc507d48-58be-4a67-8641-df75cb53221e</entry>\n      </sy
dev='hd'/>\n    <smbios mode='sysin"...
0x7fc4e9500448:  "fo'/>\n  </os>\n  <features>\n    <acpi/>\n    <apic/>\n    <pae/>\n
ck offset='utc'>\n    <timer name='pit' t"...
0x7fc4e9500510:  "ickpolicy='delay'/>\n    <timer name='rtc' track='guest'/>\n    <timer
on_reboot>\n  <on_crash>restart</on"...
0x7fc4e95005d8:  "_crash>\n  <devices>\n    <emulator>/usr/local/bin/qemu-system-x86_64<
/>\n      <source file='/instancei"...
```

```
68          try:
69              self.realXmlConf = ET.fromstring(self.domain.XMLDesc(2))
70          except Exception as e:
71              errmSg = "get real xml conf failed, %s, %s" % (self.uuid, e)
72              raise Exception(errMsg)
```

```
# export LD_PRELOAD=/root/patched-libvirt/libvirtmod.so ./start.py
```

引用：https://bugzilla.redhat.com/show_bug.cgi?id=1140998

牛刀小试：接入层改造

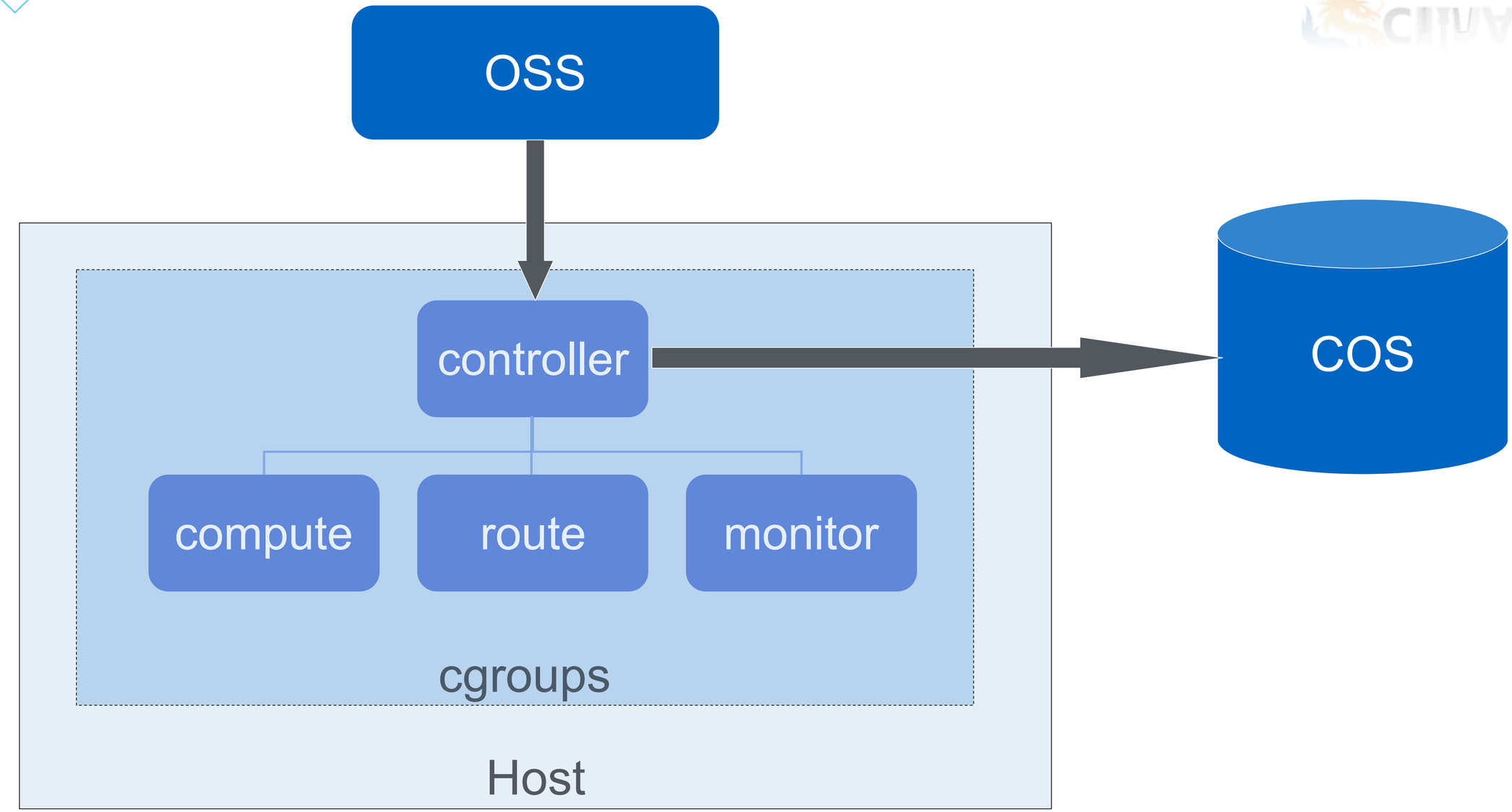
- 简化CGI开发工作
- 使接口更加现代化

Python用于系统编程

- 标准库与posix规范天然贴合
- 从sh/awk/sed/perl到Python
- 优秀的文本处理和分析能力
- 完备的网络功能

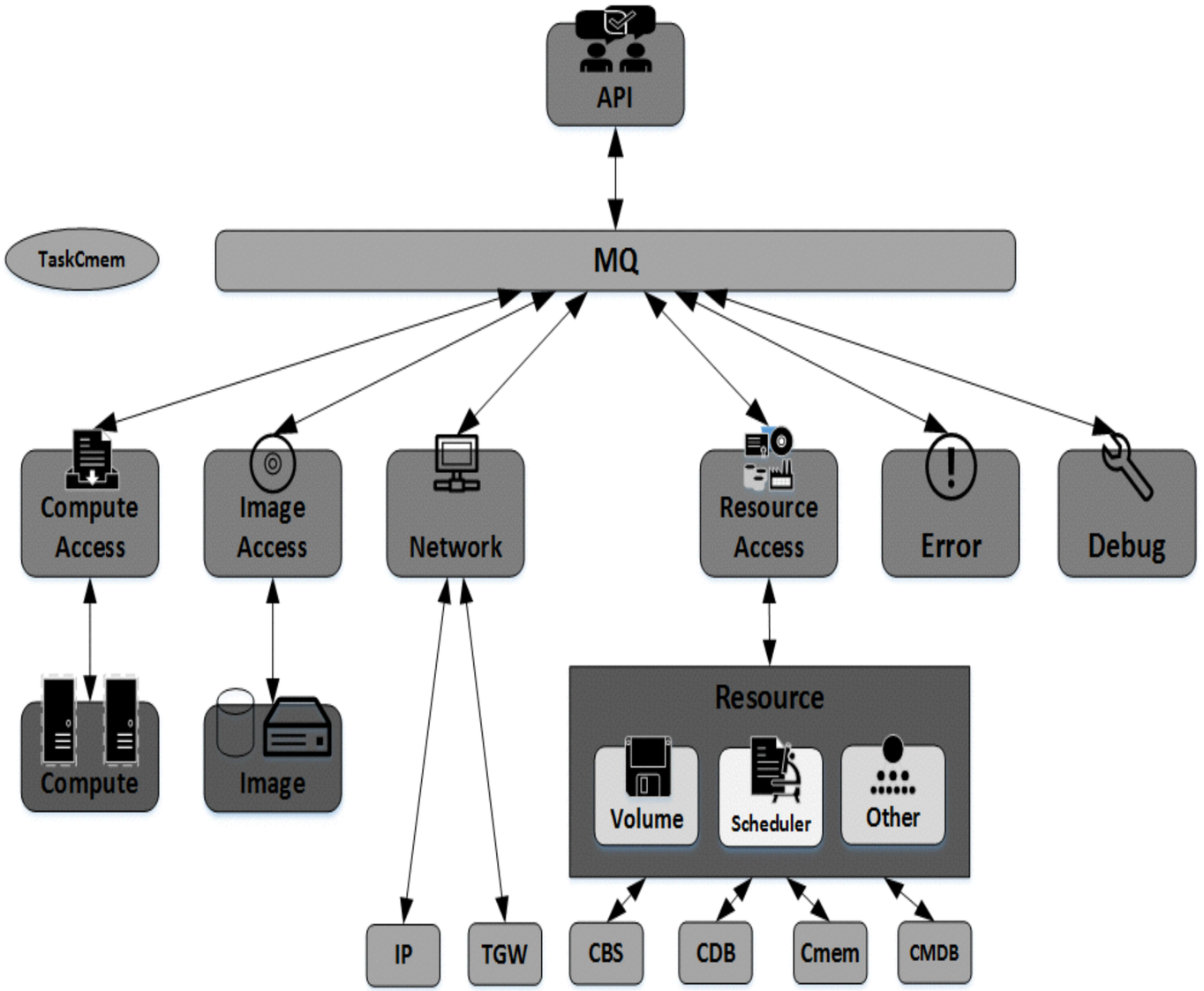
```
# 举例：Python实现daemonize
def daemonize():
    pid = os.fork()
    if pid > 0:
        sys.exit(0)
    os.chdir('/')
    os.setsid()
    os.umask(0)
    pid = os.fork()
    if pid > 0:
        sys.exit(0)
```

宿主机包管理工具



- 引用：https://www.qcloud.com/product/cos.html

云调度系统：框架



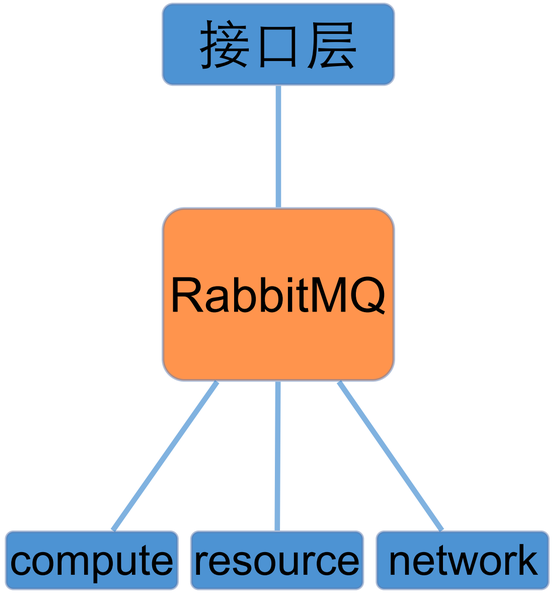
- 平行扩展、简洁高效、异步、无状态、**高可用**
- 如以太网的**共享信道**
- 如CGI的**逻辑抽离**
- 如git的**一切可追溯**
- 如sql的**事务回滚**

- 引用：https://www.zhihu.com/question/34511860/answer/104123438

云调度系统：taskflow

msgid	141592
mode	normal
cursor	0
parameters	cpu=2 mem=4096
steps	0: resource select_host 1: network alloc_ip 2: compute create_vm 3: resource update_info

消息格式　　创建虚拟机的消息



类似于OpenStack taskflow: https://wiki.openstack.org/wiki/TaskFlow

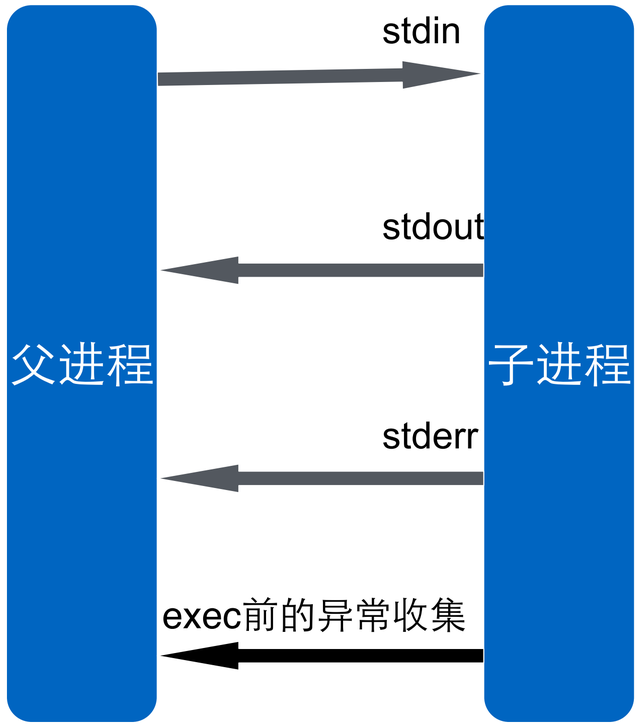
subprocess

- 小心PATH环境变量的不同
- 避免使用不必要的外部程序
  - 字符串处理相关的程序
  - 计算相关的程序
  - 网络连接相关的程序

```
In [1]: from subprocess import Popen, PIPE

In [2]: %timeit int(Popen('ps aux | wc -l', stdout=PIPE, shell=True).stdout.read().strip()) - 3
100 loops, best of 3: 10.2 ms per loop

In [3]: from os import listdir

In [4]: %timeit len(filter(str.isdigit, listdir('/proc')))
10000 loops, best of 3: 75.7 µs per loop
```

subprocess

- 避免shell=True
- 注意管道的buffer size

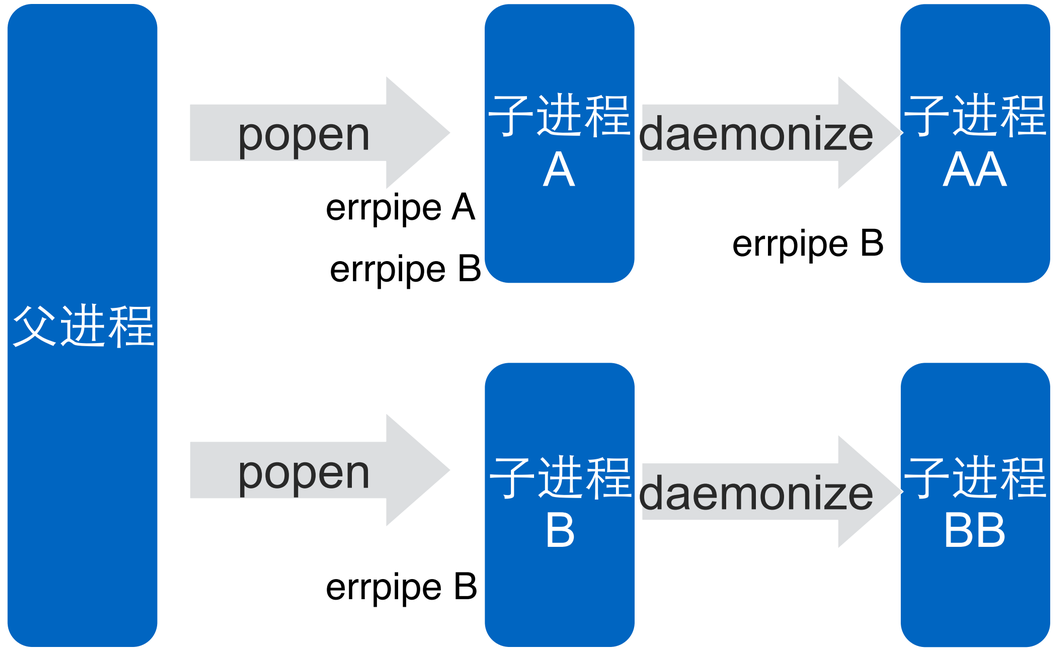



```
In [1]: from subprocess import Popen, PIPE

In [2]: def get_out(size):
            child = Popen(['python', '-c', 'print "a" * %d' % size], stdout=PIPE)
            print child.pid
            child.wait()
            return len(child.stdout.read())

In [3]: get_out(65535)
        17065
Out[3]: 65536

In [*]: get_out(65536)
        17066
```

```
In [*]: !strace -p 17066
        Process 17066 attached
        write(1, "\n", 1
```

subprocess

- 通过设置close_fds=True防止资源泄漏



引用: http://zhihu.com/question/38202274/answer/121776786

ctypes用于系统编程

- ipython + ctypes: 交互式的Linux API REPL
- 遗留so或计算密集so融入到Python工程
- Python用户态程序与内核模块通信

交互式Linux API REPL

腾讯云

准备工作

In [2]:
```
file_path = '/tmp/ctypes-test-stdio'
content = 'hello world!'
```

In [3]:
```
import ctypes
glibc = ctypes.CDLL(None)
```

In [4]:
```
# 导入函数
for fun in 'fopen fclose fflush fprintf setvbuf'.split():
    locals()[fun] = getattr(glibc, fun)
```

In [5]:
```
def write_content(stream, count):
    for i in range(count):
        fprintf(stream, content)
```

全缓冲的标准IO

In [6]:
```
stream = fopen(file_path, 'w')
```

In [7]:
```
write_content(stream, 1)
```

In [8]:
```
!cat $file_path
```

In [9]:
```
fflush(stream)
```

Out[9]: 0

In [10]:
```
!cat $file_path
```

```
hello world!
```

In [11]:
```
%time write_content(stream, 10000)
```

```
CPU times: user 1.66 ms, sys: 1.65 ms, total: 3.31 ms
Wall time: 2.94 ms
```

In [12]:
```
fclose(stream)
```

Out[12]: 0

无缓冲的标准IO

In [13]:
```
stream = fopen(file_path, 'w')
```

In [14]:
```
setvbuf(stream, 0, 2, 0)  # _IONBF = 2
```

Out[14]: 0

In [15]:
```
write_content(stream, 1)
```

In [16]:
```
!cat $file_path
```

```
hello world!
```

In [17]:
```
%time write_content(stream, 10000)
```

```
CPU times: user 4.28 ms, sys: 4.16 ms, total: 8.44 ms
Wall time: 8.11 ms
```

In [18]:
```
fclose(stream)
```

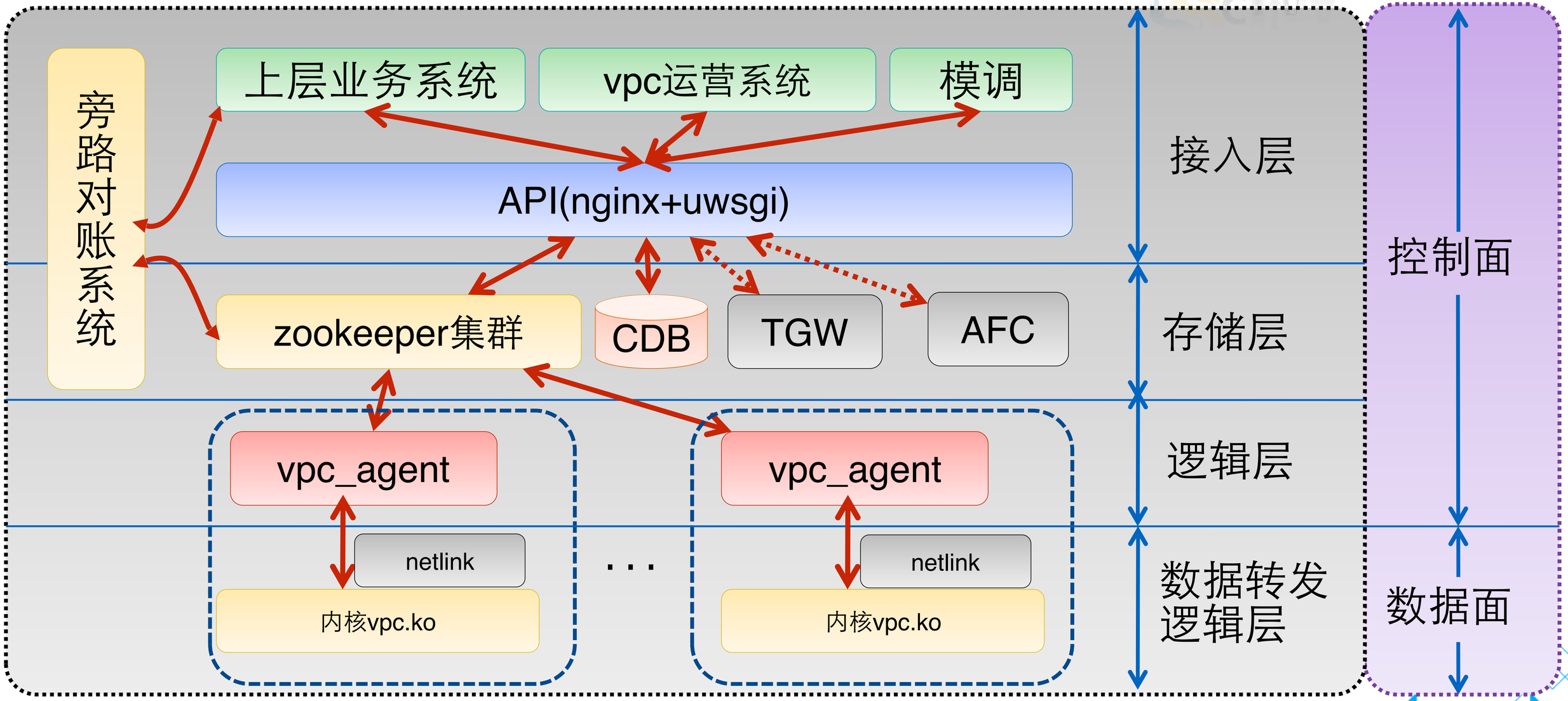
Out[18]: 0

与内核模块通信

- ctypes调用ioctl
- ctypes/struct实现netlink
- 引用：git://repo.or.cz/iotop.git

```
import ctypes

libc = ctypes.CDLL(None)

class SOCKADDR_NL(ctypes.Structure):
    _fields_ = [("nl_family", ctypes.c_ushort),
                ("nl_pad",    ctypes.c_ushort),
                ("nl_pid",    ctypes.c_int),
                ("nl_groups", ctypes.c_int)]

def _nl_bind(descriptor, addr):
    addr = SOCKADDR_NL(socket.AF_NETLINK, 0, os.getpid(), 0)
    return libc.bind(descriptor.fileno(),
                     ctypes.pointer(addr),
                     ctypes.sizeof(addr))

def _nl_getsockname(descriptor):
    addr = SOCKADDR_NL(0, 0, 0, 0)
    len = ctypes.c_int(ctypes.sizeof(addr));
    libc.getsockname(descriptor.fileno(),
                     ctypes.pointer(addr),
                     ctypes.pointer(len))
    return addr.nl_pid, addr.nl_groups;

def _nl_send(descriptor, msg):
    return libc.send(descriptor.fileno(), msg, len(msg), 0);

def _nl_recv(descriptor, bufs=16384):
    addr = SOCKADDR_NL(0, 0, 0, 0)
    len = ctypes.c_int(ctypes.sizeof(addr))
    buf = ctypes.create_string_buffer(bufs)

    r = libc.recvfrom(descriptor.fileno(),
                      buf, bufs, 0,
                      ctypes.pointer(addr), ctypes.pointer(len))

    ret = ctypes.string_at(ctypes.pointer(buf), r)
    return ret, (addr.nl_pid, addr.nl_groups)
```

腾讯云VPC路由管理

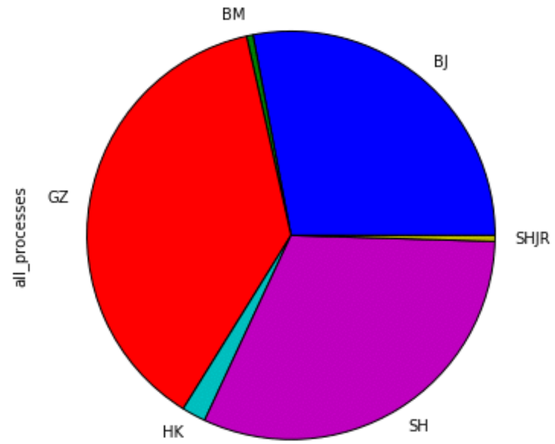

旁路对账系统
上层业务系统
vpc运营系统
模调
API(nginx+uwsgi)
zookeeper集群
CDB
TGW
AFC
vpc_agent
netlink
内核vpc.ko
vpc_agent
netlink
内核vpc.ko
接入层
存储层
逻辑层
数据转发逻辑层
控制面
数据面

Python数据分析

- API调用分析：ipython notebook + pandas
- 日志分析：ipython notebook + pyspark

结论十九：地域调用总数饼状图

```
In [150]: df_iaas_group_region['all_processes'].plot(kind='pie', figsize=(6, 6))
Out[150]: <matplotlib.axes._subplots.AxesSubplot at 0x571b1d0>
```



北京、上海、广州是最活跃的地域，活跃程度大致相当

我和Python(二)

- 简历分析系统
- 火车票、医院挂号、购物、买月饼
- 微信公众号
- ……

PyCon 2016 CHINA

腾讯云

Thank You！