
PyCon
2016
CHINA

Debug Python

The magic way...

Why

- Inspect/debug any python process anywhere anytime
- Performance profiling for gevent

dtrace

Dynamic tracing: Troubleshooting kernel and application problems, originally written for Solaris by Sun

- **production systems**
- **real time**
- Probes planted in c code
- low performance hit
- Has Linux support, but not included in most distros(even Archlinux)

systemtap

- Probe dtrace markers
- Better scripting support
- Comes with centos distro

Probes

- syscalls/network/io...
- debuginfo
- user defined probes... Python!

Downside

- Markers not in official release
- Need to plant markers and recompile Python...

However!

Redhat has done this for us.

CentOS

Fedora

...

Markers:

python.function.entry filename:string funcname:string lineno:long

python.function.return filename:string funcname:string lineno:long

What to do with Python?

- Execution inspection
- Performance profiling
- Memory profiling
- Which function does the most IO operations.
- ...

Example

Profile gevent patched programs

Problems

- cProfile profile current threads
- All greenlets run in one thread
- -> Results mingled together...

Single Thread cProfiling

cProfiling under gevent

Define probe points

gevent.stp

```
global pystack

probe ves.function.entry = process.library(@1).mark("function__entry")
{
  filename = user_string($arg1);
  funcname = user_string($arg2);
  lineno = $arg3;
}

probe ves.function.return =
process.library(@1).mark("function__return")
{
  filename = user_string($arg1);
  funcname = user_string($arg2);
  lineno = $arg3;
}
```

```
probe greenlet = \
        process.library(@2).\
        function(“g_switch@greenlet.c")
{
  pystack = $target;
}
```

Use Probes

```
probe ves.function.entry
{
  printf("%d => %s => %s in %s:%d\n", pystack,
gevent_indent(1), funcname, filename, lineno);
}

probe ves.function.return
{
  printf("%d => %s <= %s in %s:%d\n", pystack,
gevent_indent(-1), funcname, filename, lineno);
}

probe greenlet {
}
```

```
$ sudo stap -x 10083 gevent.stp \
    "/lib64/libpython2.7.so.1.0" \
    "/home/vagrant/.virtualenvs/zeus_core/lib64/
  python2.7/site-packages/greenlet.so" \
 | tee dump
```

120133424 => 1470406350825677 gunicorn: worke(1804): => __getitem__ in /srv/ves/zeus.ers/.venv/lib/python2.7/site-packages/huskar_sdk/utils/cached_dict.py:108

120133424 => 1470406350825683 gunicorn: worke(1804): <= __getitem__ in /srv/ves/zeus.ers/.venv/lib/python2.7/site-packages/huskar_sdk/utils/cached_dict.py:109

120133424 => 1470406350825692 gunicorn: worke(1804): <= get in /srv/ves/zeus.ers/.venv/lib64/python2.7/_abcoll.py:365

120133424 => 1470406350825698 gunicorn: worke(1804): => get in /srv/ves/zeus.ers/.venv/lib64/python2.7/_abcoll.py:360

120133424 => 1470406350825702 gunicorn: worke(1804): => __getitem__ in /srv/ves/zeus.ers/.venv/lib/python2.7/site-packages/huskar_sdk/utils/cached_dict.py:108

120133424 => 1470406350825707 gunicorn: worke(1804): <= __getitem__ in /srv/ves/zeus.ers/.venv/lib/python2.7/site-packages/huskar_sdk/utils/cached_dict.py:109

120133424 => 1470406350825714 gunicorn: worke(1804): <= get in /srv/ves/zeus.ers/.venv/lib64/python2.7/_abcoll.py:365

120133424 => 1470406350825719 gunicorn: worke(1804): <= is_switched_on in /srv/ves/zeus.ers/.venv/lib/python2.7/site-packages/huskar_sdk/components/switch.py:141

120133424 => 1470406350825723 gunicorn: worke(1804): <= wrapper in /srv/ves/zeus.ers/.venv/lib/python2.7/site-packages/huskar_sdk/utils/__init__.py:39

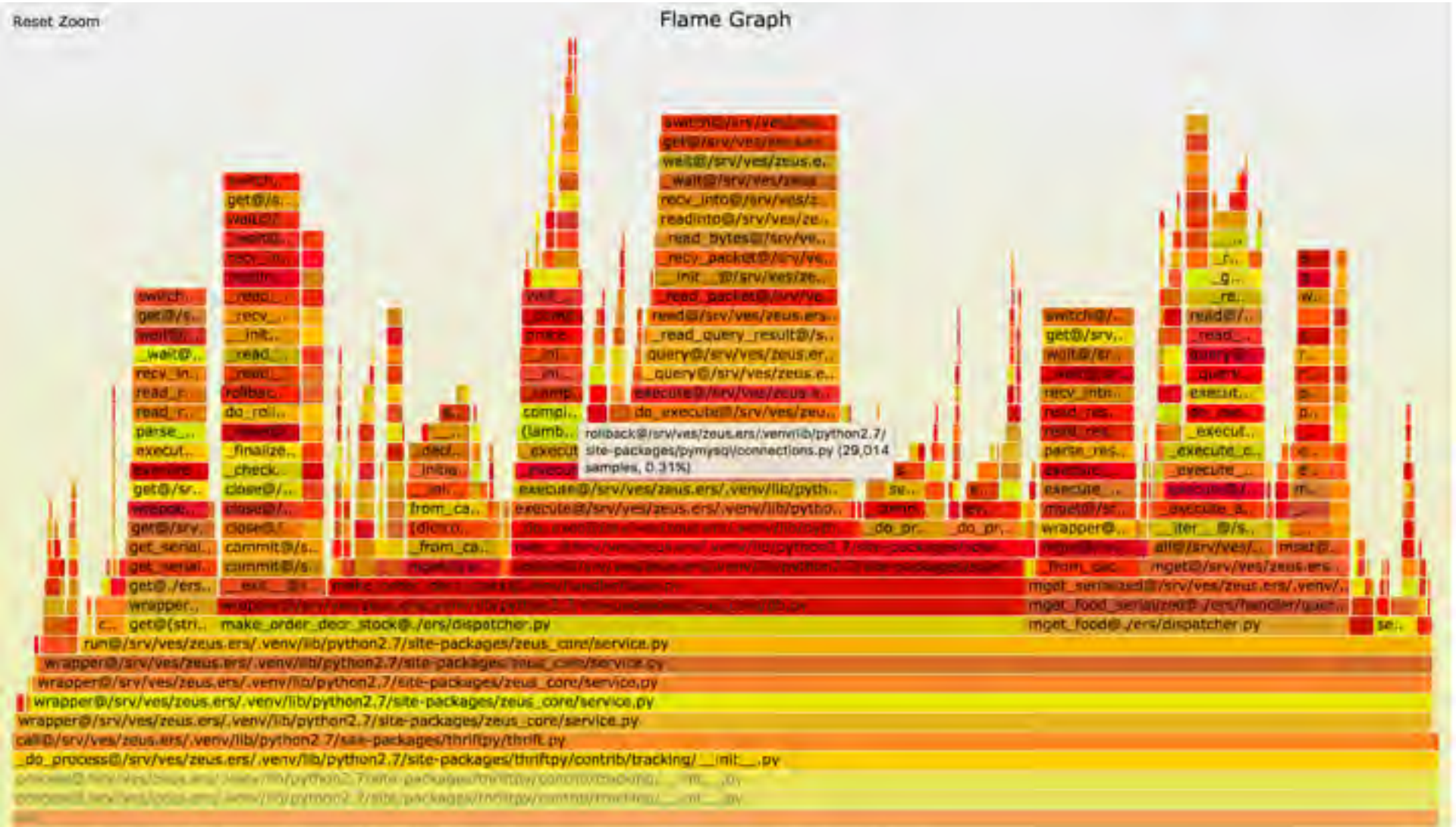
120133424 => 1470406350825727 gunicorn: worke(1804): <= is_enabled in /srv/ves/zeus.ers/.venv/lib/python2.7/site-packages/zeus_core/etrace_wrapper.py:137

Details…

120133424 => 1470406350825692 gunicorn: worke(1804): <= get in /srv/ves/zeus.ers/.venv/lib64/python2.7/_abcoll.py:365

120133424 => 1470406350825698 gunicorn: worke(1804): => get in /srv/ves/zeus.ers/.venv/lib64/python2.7/_abcoll.py:360

Performance profiling…



Other probes

```
probe pyobject_malloc = \
      process.library(@1).\
      function(“PyObject_Malloc")
{
  allocated = $nbytes;
}
```

Implant probes into Python

github.com/wooparadog/python-systemtap

```
from pystap import dtrace_deco

@dtrace_deco
def its_a_test(a_para, that_para=None):
    print a_para, that_para
```

Links:

- https://sourceware.org/systemtap/
- https://github.com/emfree/systemtap-python-tools
- https://github.com/wooparadog/python-systemtap

Take aways

- Python is a c program!
- dtrace + systemtap: debug/profile/inspect anything anywhere anytime(almost)