Using Parrot in Scientific Workflows

Tim Shaffer
University of Notre Dame

tshaffee1@nd.edu
Misbehaving Tasks

**Problem**: a large number of temp files are accumulating on workers. Some tasks don't clean up properly before exiting.

Enter **Parrot**:

Set up each task with a private `/tmp`, now it’s easy to identify/clean up what a task left behind.
```
sh-4.1$ ls /tmp
concept-generic.png  mount.list  py1.png  sandbox2.png
concept-generic.svg  orbit-gdm   py2.png  sandbox3.png
foo                   orbit-tshaffe1 py3.png  seahorse-4V1b32
foo2                  parrot.213185  py4.png  seahorse-JYSDPc
hsperfdata_condor    parrot-poster.gif py5.png  seahorse-uIUK1d
kde-tshaffe1          private.01    py6.png  tmp.png
keyring-bC1B23        pulse-j10IeCGIGEdv py7.png  virtual-tshaffe1.lUjwS5
keyring-P19Vjf         pulse-mLj04nvS9dzI root
krb5cc_213185_0lGP5o  pulse-zQR5WAFKE1yG sandbox1.png
sh-4.1$  
```
sh-4.1$ ls /tmp
concept-generic.png  mount.list  py1.png  sandbox2.png
concept-generic.svg  orbit-gdm  py2.png  sandbox3.png
foo  orbit-tshaffe1  py3.png  seahorse-4VIb32
foo2  parrot.213185  py4.png  seahorse-JYSDPc
hspferdata_condor  parrot-poster.gif  py5.png  seahorse-uIUk1d
kde-tshaffe1  private.01  py6.png  tmp.png
keyring-bC1B23  pulse-j10IeCGIGEdv  py7.png  virtual-tshaffe1.uJwS5
keyring-P19Vjf  pulse-mLj04nvS9dzI  root
krb5cc_213185_0lGP5o  pulse-zQR5WAFKE1yG  sandbox1.png
sh-4.1$ parrot_run -m
sh-4.1$
Concept-generic.png  mount.list  py1.png  sandbox2.png
concept-generic.svg  orbit-gdm  py2.png  sandbox3.png
foo  orbit-tshaffel  py3.png  seahorse-4VIb32
foo2  parrot.213185  py4.png  seahorse-JYSDPc
hsperfdata_condor  parrot-poster.gif  py5.png  seahorse-uUK1d
kde-tshaffel  private.01  py6.png  tmp.png
keyring-bC1B23  pulse-j10IeCGIGEdv  py7.png  virtual-tshaffel.lUjwS5
keyring-P19Vjf  pulse-mLj04nvS9dzI  root
krb5cc_213185_0lGP5o  pulse-zQR5WAFKE1yG  sandbox1.png
sh-4.1$ ls /tmp
sh-4.1$ ls /tmp
messy_script.sh
sh-4.1$ ./messy_script.sh
writing setup files
making a mess of /tmp
doing other stuff
done!
exiting without cleaning up
sh-4.1$
sh-4.1$ parrot_run -m mount.list sh -l
sh-4.1$ ls /tmp
messy_script.sh
sh-4.1$ ./messy_script.sh
writing setup files
making a mess of /tmp
doing other stuff
done!
exiting without cleaning up
sh-4.1$ ls /tmp
file1  file20  file32  file44  file56  file68  file8  file91
file10  file21  file33  file45  file57  file69  file80  file92
file100  file22  file34  file46  file58  file7  file81  file93
file11  file23  file35  file47  file59  file70  file82  file94
file12  file24  file36  file48  file6  file71  file83  file95
file13  file25  file37  file49  file60  file72  file84  file96
file14  file26  file38  file5  file73  file85  file97
file15  file27  file39  file50  file62  file74  file86  file98
file16  file28  file4  file51  file63  file75  file87  file99
file17  file29  file40  file52  file64  file76  file88  messy_script.sh
file18  file3  file41  file53  file65  file77  file89  more_things
file19  file30  file42  file54  file66  file78  file9  options
file2  file31  file43  file55  file67  file79  file90  setup
sh-4.1$
messy_script.sh

sh-4.1$ ./messy_script.sh
writing setup files
making a mess of /tmp
doing other stuff
done!
exiting without cleaning up

sh-4.1$ ls /tmp
file1 file20 file32 file44 file56 file68 file8 file91
file10 file21 file33 file45 file57 file69 file80 file92
file100 file22 file34 file46 file58 file7 file81 file93
file11 file23 file35 file47 file59 file70 file82 file94
file12 file24 file36 file48 file6 file71 file83 file95
file13 file25 file37 file49 file60 file72 file84 file96
file14 file26 file38 file5 file61 file73 file85 file97
file15 file27 file39 file50 file62 file74 file86 file98
file16 file28 file4 file51 file63 file75 file87 file99
file17 file29 file40 file52 file64 file76 file88 messy_script.sh
file18 file3 file41 file53 file65 file77 file89 more_things
file19 file30 file42 file54 file66 file78 file9 options
file2 file31 file43 file55 file67 file79 file90 setup

sh-4.1$ exit
logout
sh-4.1$
Bonus: keep tasks from snooping around

They probably don't need access to

- /home
- /dev
- /sys
- /proc, maybe others

Alternatively, use a more fine-grained approach, e.g. "only allow a Makeflow job to write to the outputs it specified".
Portable Applications

It’s hard to know what will be available at the execution site.

- missing libraries
- different filesystem layout (e.g. /bin vs. /usr/bin, or packages installed under /opt)
- libraries compiled with features missing
- bad ld.so (*really!*
Portable Applications

Bundle all dependencies, and use Parrot to set up the filesystem.

The app sees a consistent, known-good system configuration.

Parrot can automatically detect dependencies and make a package.
Example: Portable Python

Copying the Python binary to another computer won’t work: we need libraries and dependencies

- bzip2
- db
- expat
- filesystem
- gdbm

- glibc
- iana-etc
- libffi
- linux-api-headers
- openssl
- perl
- python
- tzdata
- zlib
sh-4.1$ parrot_run
> --name-list python.namelist
>
sh-4.1$ parrot_run \
> --name-list python.namelist \
> --env-list python.envlist \
>
sh-4.1$ parrot_run \
> --name-list python.namelist \
> --env-list python.envlist \
> python
Python 2.6.6 (r266:84292, May 22 2015, 08:34:51)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-15)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
sh-4.1$ parrot_package_create --name-list python.namelist --env-list python.envlist --package-path python.pkg
The packaging process has began ...
The start time is: Tue Oct 18 14:21:11 2016
Package Path: python.pkg
Package Size: 102M   python.pkg
The packaging process has finished.
The end time is: Tue Oct 18 14:21:11 2016
sh-4.1$
sh-4.1$ parrot_package_run --package-path python.pkg/ python
sh-4.1$ parrot_package_run --package-path python.pkg/ python
Python 2.6.6 (r266:84292, May 22 2015, 08:34:51)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-15)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
Remote Dependencies

Parrot can make remote resources available through the normal filesystem interface.

Rather than bundling all dependencies (which could be far more than needed on large projects), let Parrot fetch them on demand.

Programs see extra latency on initial access, but only retrieve the parts they actually use.
CVMFS

CernVM Filesystem (CVMFS) takes this approach to distribute experiment software.

Large, frequently updated codebase accessed daily from grid sites all over the world.

No need to explicitly install packages; just start running things, and dependencies are loaded as needed.
CVMFS on HPC

High performance computing (HPC) resources might not have an open internet connection and FUSE.

For the former, we can run an HTTP proxy on the login node.

Since Parrot supports CVMFS, just send a Parrot executable, no FUSE or setuid programs required.
CVMFS on HPC

Experiments are highly dependent on CVMFS to deliver software.

Long-running, compute-bound tasks don't suffer much performance penalty under Parrot.

With Parrot, take advantage of any worker with a working kernel, no need for cluster admins to install extra software.
Questions?

tshaffe1@nd.edu

http://ccl.cse.nd.edu/software/parrot/