CDE: A tool for automatically creating reproducible experimental software packages

Philip Guo
6th-year Ph.D. student
Department of Computer Science
Stanford University
pg@cs.stanford.edu
http://www.stanford.edu/~pgbovine/cde.html
Barriers to reproducible research

Cultural, Political, Behavioral, Institutional, Sociopsychological

Technical
Barriers to reproducible research

It’s really hard to take research code that runs on your machine and get it to run on someone else’s machine, even one with the same OS as yours.
Barriers to reproducible research

Download

graph-tool is released under the GPL. For installation instructions, see the included INSTALL file.

Requirements

You'll need the following in order to build and use graph-tool:

- GCC, version 4.2 or above (version 4.5 is recommended).
- The boost libraries, version 1.42 or above.
- Python, version 2.5 or above (version 3 is not yet supported).
- The expat library.
- The SciPy python module.
- The Numpy python module.
- The CGAL C++ geometry library, version 5 or above.
- Graphviz for graph drawing, with the python bindings enabled (optional).

graph-tool was tested only on GNU/Linux and MacOS X systems, but should also be usable on other systems where the above requirements are available.
Current ways to distribute research code

1. Zip up everything, throw over the fence

Amount of pain your users suffer

Amount of pain you (the author) must suffer
Current ways to distribute research code

<table>
<thead>
<tr>
<th>Amount of pain your users suffer</th>
<th>Amount of pain you (the author) must suffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

2. Source code + Makefiles + configure scripts + documentation
Current ways to distribute research code

- Amount of pain your users suffer
- Amount of pain you (the author) must suffer

3. Integrate with package management system
Current ways to distribute research code

Amount of pain your users suffer

Amount of pain you (the author) must suffer

4. Re-create entire environment within a virtual machine
Current ways to distribute research code

<table>
<thead>
<tr>
<th>Amount of pain your users suffer</th>
<th>Amount of pain you (the author) must suffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

5. Create a robust one-click installer
Current ways to distribute research code

Amount of pain your users suffer

Amount of pain you (the author) must suffer
CDE: Automatic packaging of Code, Data, and Environment

1. Create package on your Linux computer
   Prepend any set of commands with ‘cde’, and CDE runs them and automatically packages up their dependencies

2. Transfer package
   A package is simply a directory of files (~10MB – 500MB)

3. Execute software from within package on any modern Linux computer
   Prepend those same commands with ‘cde-exec’, and CDE runs them natively without any installation
Creating a package with cde

```
cde
  ptrace
  open("/lib/libc.so.6")
  chdir("foo/")
  open("/lib/libc.so.6")
```

Timeline

```
program
  open()

kernel
  open file

  copy file into package
```
Executing a package with cde-exec

Timeline

**cde-exec**

ptrace

**monitored process**

```plaintext
  ptrace
  open("/lib/libc.so.6")
  chdir("foo/")
  open("/lib/libc.so.6")
```

**.kernel**

**program**

- `open()`

**kernel**

- `open file from package`

**cde-exec**

- `rewrite open() argument`
Creating a package with cde

cd /home/pg/expt/
cde python predict_weather.py
Creating a package with cde

```
cd /home/pg/expt/
cde python predict_weather.py
```
Transfer package to target machine

cde-package/

- `cde-root/usr/bin/python`
- `cde-root/usr/lib/libpython2.6.so`
- `cde-root/home/pg/expt/predict_weather.py`
- `cde-root/usr/bin/R`
- `cde-root/usr/local/R/stdlib.R`
- `cde-root/usr/local/R/weatherMod.so`
- `cde-root/home/pg/expt/weather_models.R`
- `cde-root/usr/bin/WeatherSim`
Executing a package with cde-exec

```bash
cd cde-package/cde-root/home/pg/expt/
cde-exec python predict_weather.py
```
“Live” demo

[ To watch the demo video, visit: http://vimeo.com/20256490 ]
Benefits

1. Creating a CDE package is as easy as running your original experiment
2. Works with existing languages and tools
3. Executing a CDE package requires no installation, setup, or root permissions
Limitations

1. Packages might be incomplete
2. Execution is slower (2% - 30%)
3. Cannot emulate custom hardware
4. Only x86 \(\rightarrow\) x86, Linux 2.6 \(\rightarrow\) Linux 2.6
Integrating with other tools

**CDE + VM**: Greater portability than CDE alone, enables longer-term archiving of experiments

**CDE + EC2**: Instant cloud deployment, enables reviewers to ssh/VNC into a public URL and re-run your experiments

**CDE + Git**: Easily collaborate with executable experiment repositories

**CDE + <your tool>**: Let’s seriously talk!
Barriers to reproducible research

It’s really hard to take research code that runs on your machine and get it to run on someone else’s machine, even one with the same OS as yours.
Summary / sales pitch for CDE

**Simple promise:** If you can run a set of commands on your Linux machine, then CDE allows anyone to easily re-run those same commands on their Linux machine.

**Legacy-friendly:** Scientists can work in their favorite programming languages or GUI tools.

**Battle-hardened:** Thousands of downloads, hundreds of subtle bug fixes enable it to work “out-of-the-box”.

**Integration-ready:** Can serve as a layer below other more sophisticated tools. Let’s discuss integration!