Conducting Reproducible Research with Umbrella: Tracking, Creating, and Preserving Execution Environments

Haiyan Meng, Alexander Vyushkov, Matthias Wolf, Anna Woodard and Douglas Thain

University of Notre Dame
Notre Dame, Indiana, USA
October 2016
Observation: it is difficult to reproduce the experiment results published in academic papers!

Alice did the experiments for her paper:

server: lab01.phy.research.org

1) installed software deps (i.e., sim_sort) under /home/alice/software
2) configured environment variables (SIMCOUNT)
3) wrote the analysis script, analysis.py
   /usr/bin/python --> python2.7
4) downloaded the datasets to /home/alice/data

Experiment results -> Figures
Submitted the paper, and it got accepted.
Several months later, Bob read the paper and emailed Alice to ask for help to reproduce the experiment.

Alice searched for analysis.py and sent it to Bob.

Problems Bob encountered: 😮 😞

- analysis.py depends on the setting of the environment variable SIMCOUNT
- analysis.py expects an input file located at `/home/alice/data/file1`
- analysis.py attempts to utilize an executable named sim_sort
- the output of analysis.py overflows Bob's memory and disk
- `/usr/bin/python` on Bob's machine is Python 3.0, which is not backwards compatible with Python 2.7.
• Alice forgot to preserve the SIMCOUNT setting.
• Alice deleted the directory /home/alice/data by accident.
• sim_sort is under version control via Git and can be found, however, Alice forgot the commit id used.
• As for the memory and disk overflow, Alice realized she should have told Bob the experiment requires 6GB memory and 20GB disk space.

Sysadmins update kernel, OS, system software periodically
Hardware upgrade every several years
Network resources from third-party websites

.... Experiment results can NOT be reproduced by others or even the original author!
Lessons

• Publishing scientific results without the detailed execution environments describing how the results were collected makes it difficult or even impossible for the reader to reproduce the work.

• The configurations of the execution environments are too complex to be described easily by authors.

   hardware, kernel, OS, software, data, environ vars
A Framework for Conducting Reproducible Research

- **Tracking execution environments** allows the user to specify all the necessary details about a comprehensive execution environment.

- **Creating execution environments** sandbox techniques like VMs, Linux Containers (i.e., Docker) and user-space tracers (i.e., Parrot).

- **Preserving execution environments** archives data and software deps in the first place into persistent storage services (i.e., Amazon S3).
Tracking Execution
Environments: Umbrella
Specification

Sections:
- hardware
- kernel
- os
- software
- data
- environ
- cmd
- output
- description

os/software/data sections:
- source
- checksum
- size
- format
- mountpoint

```
{  
  "description": "A ray-tracing application which creates video frames.",  
  "hardware": {  
    "arch": "x86_64",  
    "cores": "1",  
    "memory": "1GB",  
    "disk": "3GB"  
  },  
  "kernel": {  
    "name": "linux",  
    "version": ">=2.6.18"  
  },  
  "os": {  
    "name": "redhat",  
    "version": "6.5",  
    "mountpoint": "/",  
    "source": [  
      "http://cct.cse.nd.edu/.../redhat-6.5-x86_64.tar.gz"  
    ],  
    "format": "tgz",  
    "action": "unpack",  
    "checksum": "669ab5ef94af84d273f8f92a86b7907a",  
    "size": "633848940",  
    "uncompressed_size": "1743656960",  
    "ec2": {  
      "ami": "ami-2cf8901c",  
      "region": "us-west-2",  
      "user": "ec2-user"  
    }  
  },  
  "software": {  
    "povray-3.6.1-redhat6-x86_64": {  
      "mountpoint": "/software/povray-3.6.1-redhat6-x86_64",  
      "source": [  
        "http://ccl.cse.nd.edu/.../povray-3.6.1-redhat6-x86_64.tar.gz"  
      ],  
      "format": "tgz",  
      "action": "unpack",  
      "checksum": "b02ba86dd3081a703b4b01dc463e0499",  
      "size": "1471452",  
      "uncompressed_size": "3010560"  
    }  
  },  
  "data": {  
    "4_cubes.pov": {  
      "mountpoint": "/tmp/4_cubes.pov",  
      "source": [  
        "http://ccl.cse.nd.edu/.../4_cubes.pov"  
      ],  
      "format": "plain",  
      "action": "none",  
      "checksum": "c65266cd2b672854b821ed93028a877a",  
      "size": "1757"  
    }  
  },  
  "environ": {  
    "PWD": "/tmp"  
  },  
  "cmd": "povray +I/tmp/4_cubes.pov +O/tmp/frame000.png +K.0 -H50 -W50",  
  "output": {  
    "files": [  
      "/tmp/frame000.png"  
    ],  
    "dirs": [  
      "/tmp/output"  
    ]  
  }  
}
```
## Resource URLs Supported by Umbrella

<table>
<thead>
<tr>
<th>Resource</th>
<th>Example URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Filesystem</td>
<td>/home/hmeng/data/input</td>
</tr>
<tr>
<td>HTTP</td>
<td><a href="http://www.data.com/data/file1">http://www.data.com/data/file1</a></td>
</tr>
<tr>
<td>HTTPS</td>
<td><a href="https://lab01.nd.edu/data/hep/file2">https://lab01.nd.edu/data/hep/file2</a></td>
</tr>
<tr>
<td>Amazon S3</td>
<td>s3+<a href="https://s3.aws.com/.../cubes.pov">https://s3.aws.com/.../cubes.pov</a></td>
</tr>
<tr>
<td>Open Science Framework (OSF)</td>
<td>osf+<a href="https://files.osf.io/v1/.../7559c3a">https://files.osf.io/v1/.../7559c3a</a></td>
</tr>
<tr>
<td>Git Repository</td>
<td>git+<a href="https://github.com/.../cctools.git">https://github.com/.../cctools.git</a></td>
</tr>
<tr>
<td>CernVM File System</td>
<td>cvmfs://cvmfs/cms.cern.ch</td>
</tr>
</tbody>
</table>
Creating Execution Environment: Umbrella Execution Engine

Matching degree between
-- the execution node
-- the specified execution environment

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Kernel</th>
<th>OS</th>
<th>Sandbox Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Utilize the current OS directly</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>OS-level Virtualization Docker, Parrot</td>
</tr>
<tr>
<td>Yes/No</td>
<td>No</td>
<td>No</td>
<td>Hardware Virtualization Local: VirtualBox, VMWare Remote: Amazon EC2</td>
</tr>
</tbody>
</table>
Umbrella Execution Engine - Local

1. User starts Umbrella:  
   $ umbrella run povray.umbrella

2. Umbrella reads and parses  
povray.umbrella

3. Umbrella downloads all the missing  
dependencies

4. Umbrella creates a sandbox to run  
the experiment

5. Umbrella returns the experiment  
results

6. User checks and analyzes the  
experiment results

---

User’s Duty

Umbrella’s Duty
Umbrella Local Cache

• OS-level virtualization
Preserving Execution Environment: Umbrella Archiver

• Uploads the deps into persistent storage services
  – Amazon S3
  – OSF storage service

• Allows the user to mark unreliable deps
  Local dependencies
  Some third-party network dependencies

• Allows the user to set the access permission of uploaded resources
How Our Framework can Help Alice and Bob?

1. Alice composes and tests `povray_local.umbrella` using local dependencies

2. Alice uses Umbrella to archive local deps to OSF

3. Alice publishes the OSF URL of `povray_osf.umbrella`

4. Bob downloads `povray_osf.umbrella` to his machine, and reproduces Alice’s work.

- `povray_local.umbrella`
  - os: `/home/alice/redhat_6.5`
  - software: `/home/alice/povray_3.6.1`
  - data: `/data/4_cubes.pov`

- `OSF Umbrella_Povray_Project`
  - `redhat_6.5.tar.gz`
  - `povray_3.6.1.tar.gz`
  - `4_cubes.pov`
  - `povray_osf.umbrella`

- `OSF URL of povray_osf.umbrella`: https://files.osf.io/.../povray_osf.umbrella

- `povray_osf.umbrella`
  - os: `osf+https://files.osf.io/.../redhat_6.5.tar.gz`
  - software: `osf+https://files.osf.io/.../povray_3.6.1.tar.gz`
  - data: `osf+https://files.osf.io/.../4_cubes.pov`
Evaluation

Umbrella – Python 2.6
Execution mode: Parrot, Docker, EC2

We evaluate our framework via three scientific applications:

- Epidemiology - OpenMalaria
- Scene Rendering - Povray
- High Energy Physics - CMS
### Umbrella Specification File Sizes:

<table>
<thead>
<tr>
<th>Application</th>
<th>OpenMalaria</th>
<th>Povray</th>
<th>CMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Umbrella Spec Size</td>
<td><strong>3.3KB</strong></td>
<td><strong>2.4KB</strong></td>
<td><strong>1.9KB</strong></td>
</tr>
</tbody>
</table>

### Sizes of os/software/data Dependencies of the Evaluated Applications:

<table>
<thead>
<tr>
<th>Application</th>
<th>OS Dems</th>
<th>Software Dems</th>
<th>Data Dems</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMalaria</td>
<td>CentOS 6.6 (69MB/218MB)</td>
<td>openMalaria (2.9MB/13MB) rpm packages (209MB) epel.repo (&lt;1KB)</td>
<td>.xml (28KB) .csv (&lt;1KB) .xsd (196KB)</td>
</tr>
<tr>
<td>Povray</td>
<td>RedHat 6.5 (605MB/1.8GB)</td>
<td>povray (1.5MB/2.9MB)</td>
<td>.pov (1.8KB) .inc (28KB)</td>
</tr>
<tr>
<td>CMS</td>
<td>RedHat 6.5 (605MB/1.8GB)</td>
<td>cmssw (1.3GB) Parrot (23MB/71MB)</td>
<td>.sh (&lt;1KB)</td>
</tr>
</tbody>
</table>
Sizes of os/software/data Dependencies of the Evaluated Applications:

<table>
<thead>
<tr>
<th>Application</th>
<th>OS Deps</th>
<th>Software Deps</th>
<th>Data Deps</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMalaria</td>
<td>CentOS 6.6 (69MB/218MB)</td>
<td>openMalaria (2.9MB/13MB)</td>
<td>.xml (28KB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rpm packages (209MB)</td>
<td>.csv (&lt;1KB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>epel.repo (&lt;1KB)</td>
<td>.xsd (196KB)</td>
</tr>
<tr>
<td>Povray</td>
<td>RedHat 6.5 (605MB/1.8GB)</td>
<td>povray (1.5MB/2.9MB)</td>
<td>.pov (1.8KB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.inc (28KB)</td>
</tr>
<tr>
<td>CMS</td>
<td>RedHat 6.5 (605MB/1.8GB)</td>
<td>cmssw (1.3GB)</td>
<td>.sh (&lt;1KB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parrot (23MB/71MB)</td>
<td></td>
</tr>
</tbody>
</table>

Overheads of Creating Execution Environments:

<table>
<thead>
<tr>
<th>Application</th>
<th>OpenMalaria</th>
<th>Povray</th>
<th>CMS</th>
<th>Permission / Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parrot</td>
<td>N/A</td>
<td>65min (2.40GB)</td>
<td>79min (2.39GB)</td>
<td>non-root/local</td>
</tr>
<tr>
<td>Docker</td>
<td>57min (1.53GB)</td>
<td>68min (4.11GB)</td>
<td>82min (4.19GB)</td>
<td>root/local</td>
</tr>
<tr>
<td>EC2 – m3.medium</td>
<td>113min (225MB)</td>
<td>130min (4.4MB)</td>
<td>211min (94MB)</td>
<td>non-root/remote</td>
</tr>
<tr>
<td>EC2 – m3.large</td>
<td>58min (255MB)</td>
<td>65min (4.4MB)</td>
<td>108min (94MB)</td>
<td>non-root/remote</td>
</tr>
</tbody>
</table>

The parrot and docker sandbox modes are tested on the same machine:

hardware: x86 64    kernel: Linux 2.6.32    OS: RedHat 6.7
<table>
<thead>
<tr>
<th>Application</th>
<th>OS Deps</th>
<th>Software Deps</th>
<th>Data Deps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Povray</td>
<td>RedHat 6.5 (605MB/1.8GB)</td>
<td>povray (1.5MB/2.9MB)</td>
<td>.pov (1.8KB) .inc (28KB)</td>
</tr>
<tr>
<td>CMS</td>
<td>RedHat 6.5 (605MB/1.8GB)</td>
<td>cmssw (1.3GB) Parrot (23MB/71MB)</td>
<td>.sh (&lt;1KB)</td>
</tr>
</tbody>
</table>

### Effectiveness of Umbrella Local Cache:

<table>
<thead>
<tr>
<th>Application (Deps Size)</th>
<th>Cache Size</th>
<th>Delta (Newly Added Deps)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS (2.39GB)</td>
<td>2.39GB</td>
<td>2.39GB (all deps)</td>
<td>79min</td>
</tr>
<tr>
<td>CMS - rerun</td>
<td>2.39GB</td>
<td>0</td>
<td>78min</td>
</tr>
<tr>
<td>Povray (2.40GB)</td>
<td>2.40GB</td>
<td>4.4MB (software and data deps)</td>
<td>64min</td>
</tr>
<tr>
<td>Povray - rerun</td>
<td>2.40GB</td>
<td>0</td>
<td>64min</td>
</tr>
<tr>
<td>Povray – new software deps</td>
<td>2.40GB</td>
<td>4.4MB (software deps)</td>
<td>64min</td>
</tr>
<tr>
<td>Povray – new data deps</td>
<td>2.40GB</td>
<td>28KB (data deps)</td>
<td>64min</td>
</tr>
</tbody>
</table>

The initial size of the Umbrella local cache is 0.
All the tests here were done with the parrot sandbox mode on the same machine:
  - hardware: x86 64
  - kernel: Linux 2.6.32
  - OS: RedHat 6.7
Last Step to Enhance Reproducibility - DOI

<table>
<thead>
<tr>
<th>Application</th>
<th>DOI URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMalaria</td>
<td><a href="http://dx.doi.org/doi:10.7274/R03F4MH3">http://dx.doi.org/doi:10.7274/R03F4MH3</a></td>
</tr>
<tr>
<td>Povray</td>
<td><a href="http://dx.doi.org/doi:10.7274/R0BZ63ZT">http://dx.doi.org/doi:10.7274/R0BZ63ZT</a></td>
</tr>
<tr>
<td>CMS</td>
<td><a href="http://dx.doi.org/doi:10.7274/R0765C7T">http://dx.doi.org/doi:10.7274/R0765C7T</a></td>
</tr>
</tbody>
</table>

Information on this webpage:
- DOI info
- Link to the Umbrella specification file
- Links to the OS deps
- Links to the software deps
- Links to the data deps
- Links to the Umbrella installation docs
- Link to the Umbrella user manual
- Link to the experiment result
Summary

A Framework for Conducting Reproducible Research:

• **Tracking execution environments** (Umbrella Specification)
  Lightweight, persistent and deployable execution environment specs
  Easily shared, expanded, and repurposed

• **Creating execution environments** (Umbrella Execution Engine)
  (re)create execution environments using sandbox techniques like VM, Docker and Parrot.

• **Preserving execution environments** (Umbrella Archiver)
  persistent storage services like Amazon S3 and OSF

  tracking the execution environments as the research process goes
Umbrella: [http://ccl.cse.nd.edu/software/umbrella/](http://ccl.cse.nd.edu/software/umbrella/)

Name: Haiyan Meng  
Email: hmeng@nd.edu

Questions?
Umbrella Execution Engine – EC2

1. User starts Umbrella: $ umbrella run povray.umbrella

2. Umbrella reads and parses povray.umbrella

3. Umbrella starts an Amazon EC2 instance

4. Umbrella sends the Umbrella job to the EC2 instance via scp

5. Umbrella starts the Umbrella job on the EC2 instance via ssh

6. The EC2 instance runs the Umbrella job locally

7. Umbrella fetches the results from the EC2 instance via scp

8. Umbrella returns the experiment results

9. User checks and analyzes the results

**User’s Duty**

**Umbrella’s Duty - Local**

**Umbrella’s Duty - EC2**
How Our Framework can Help Alice and Bob?

S3 link of povray_ec2_s3.umbrella: https://s3.amazonaws.com/povray/povray_ec2_s3.umbrella

povray_ec2_s3.umbrella

- os: ami-2cf8901c (redhat 6.5)
- software: s3+https://s3.amazonaws.com/povray/povray_3.6.1.tar.gz
- data: s3+https://s3.amazonaws.com/povray/4_cubes.pov

Sandbox - Povray
EC2 Instance of ami-2cf8901c

Amazon S3
Umbrella_Povray_Bucket

- povray_ec2_s3.umbrella
- povray_3.6.1.tar.gz
- 4_cubes.pov

<EC2 instance root dir>

- software
- povray 3.6.1
- data
- 4_cubes.pov

Download