Preserving Scientific Codes with Umbrella

Haiyan Meng
The Cooperative Computing Lab
University of Notre Dame
Motivation

Running an application on a new execution environment may fail:

- Incompatible Hardware
- Mismatched Kernel
- Different Operating System
- Missing Software/Data Dependencies
- Wrong Software Version
- Incorrect Environment Variables, like PATH, HOME

What will happen if you want to run the same application on ~1000 different machines??
### Notre Dame Condor Pool - 2015

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine number</td>
<td>4157</td>
</tr>
<tr>
<td>Hardware Architecture</td>
<td>X86_64, i386, i686</td>
</tr>
<tr>
<td>Kernel version</td>
<td>25 kernel (2.6.18 – 3.10.0)</td>
</tr>
<tr>
<td>OS</td>
<td>Linux, Mac</td>
</tr>
<tr>
<td>Linux Distribution</td>
<td>RHEL, Debian, CentOS</td>
</tr>
<tr>
<td>RHEL Versions</td>
<td>5.5, 5.9, 5.10, 5.11, 6.4, 6.5, 6.6, 7.0</td>
</tr>
<tr>
<td>CPU number</td>
<td>1, 2, 4, 8, 12, 16, 24, 32, 64</td>
</tr>
<tr>
<td>Memory Size</td>
<td>Max: 1TB Min: 984 MB</td>
</tr>
<tr>
<td>Disk Size</td>
<td>Max: 1.7TB Min: 5GB</td>
</tr>
<tr>
<td>Docker support</td>
<td>50 out of 4157</td>
</tr>
<tr>
<td>CVMFS support</td>
<td>2 out of 4157</td>
</tr>
</tbody>
</table>
Problem:

The new execution environment is incompatible.

How to **specify** and **reconstruct** the execution environments for scientific applications?

Portable and Reproducible
Possible Solutions

Possible Solutions:
Virtual Machines
Disk Cloning
Parrot Packaging Tool

Problems:
Overhead (time and space)
- only miss input data
- Only miss environment variables
Difficult to extend/repurpose
Umbrella: an organized way to specify execution environment

User:
provide a lightweight specification which specifies the complete execution environment

hardware, kernel, OS, software, data, environ, cmd

Umbrella:
Parse the specification
Create the execution environment
- VMs, Linux Containers (Docker), User-Space ptrace tool (Parrot)
Umbrella Specification

Sections:
- hardware
- kernel
- os
- software
- data
- environ
- command
- output

description ....

os/software/data Sections:
- source
- checksum
- size

format
- mountpoint
<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.amazonaws.com/.../cubes.pov">https://s3.amazonaws.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>
1. User starts Umbrella:
   $ umbrella run mysim.umbrella

2. Umbrella parses specification

3. Umbrella downloads dependencies

4. Umbrella recreates the mysim app
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>
</tbody>
</table>
| Yes       | Yes    | No  | OS-level Virtualization
            Docker, Parrot                                    |
| Yes/No    | No     | No  | Hardware Virtualization
            Local: VirtualBox, VMWare
            Remote: Amazon EC2                                  |
Umbrella Local Cache

• OS-level virtualization
Umbrella Command Example - Parrot

cd cctools-6.0.7-source/umbrella/example/povray

umbrella \
--spec povray_S.umbrella \n--localdir /tmp/umbrella_test/ \n--output "/tmp/frame000.png=/tmp/umbrella_test/parrot_povray_S/output.png" \n--sandbox_mode parrot \n--log umbrella.log \nrun
Umbrella Command Example - Docker

cd cctools-6.0.7-source/umbrella/example/povray

umbrella \
   --spec povray_S.umbrella \
   --localdir /tmp/umbrella_test/ \
   --output /tmp/umbrella_test/docker_povray_S \
   --output "/tmp/frame000.png=/tmp/umbrella_test/docker_povray_S/output.png" \
   --sandbox_mode docker \
   --log umbrella.log \
run
Technology Preview: Umbrella

Umbrella is a tool for specifying and materializing comprehensive execution environments, from the hardware all the way up to software and data. A user simply invokes Umbrella with the desired task, and Umbrella parses the specification, determines the minimum mechanism necessary to run the task, downloads missing dependencies, and executes the application through the available minimal mechanism, which may be direct execution, a system container (Parrot, Docker, chroot), a local virtual machine (i.e., VirtualBox), or submission to a cloud environment (i.e., Amazon EC2) or grid environment (i.e., HTCondor).

An Umbrella specification includes six sections: hardware, kernel, os, software, data, and environ. By specifying the dependencies of an application clearly and materializing the execution environment during runtime automatically, the application becomes portable and reproducible.

Umbrella involves multiple sandboxing and virtualization techniques, however, the key idea of Umbrella is to construct a sandbox for an application during runtime by mounting all of the os, software, and data dependencies into a virtual root filesystem without copying them. The usage of mounting mechanism allows multiple sandboxes share the same dependencies concurrently.

More Info
- Download Umbrella
- Umbrella User's Manual
- Mailing List

Presentations

Publications
(Having papers with tag umbrella. See all papers instead.)

Haiyan Meng, Douglas Thain, Alexander Vuyulilekho, Matthias Wolf, and Anna Woodard.
Conducting Reproducible Research with Umbrella: Tracking, Creating, and Preserving Execution Environments.

Douglas Thain, Peter Sivo, and Haiyan Meng.
Techniques for Preserving Scientific Software Executions: Preserve the Mess or Encourage Cleanliness?
12th International Conference on Digital Preservation (IPRES), November, 2015. DOI: 10.7274/R0C2353M

Haiyan Meng and Douglas Thain.
Umbrella: A Portable Environment Creator for Reproducible Computing on Clusters, Clouds, and Grids.
Workshop on Virtualization Technologies in Distributed Computing (VTDC) at HPDC, June, 2015. DOI: 10.1145/2755972.2755982