



# 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

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

**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:**

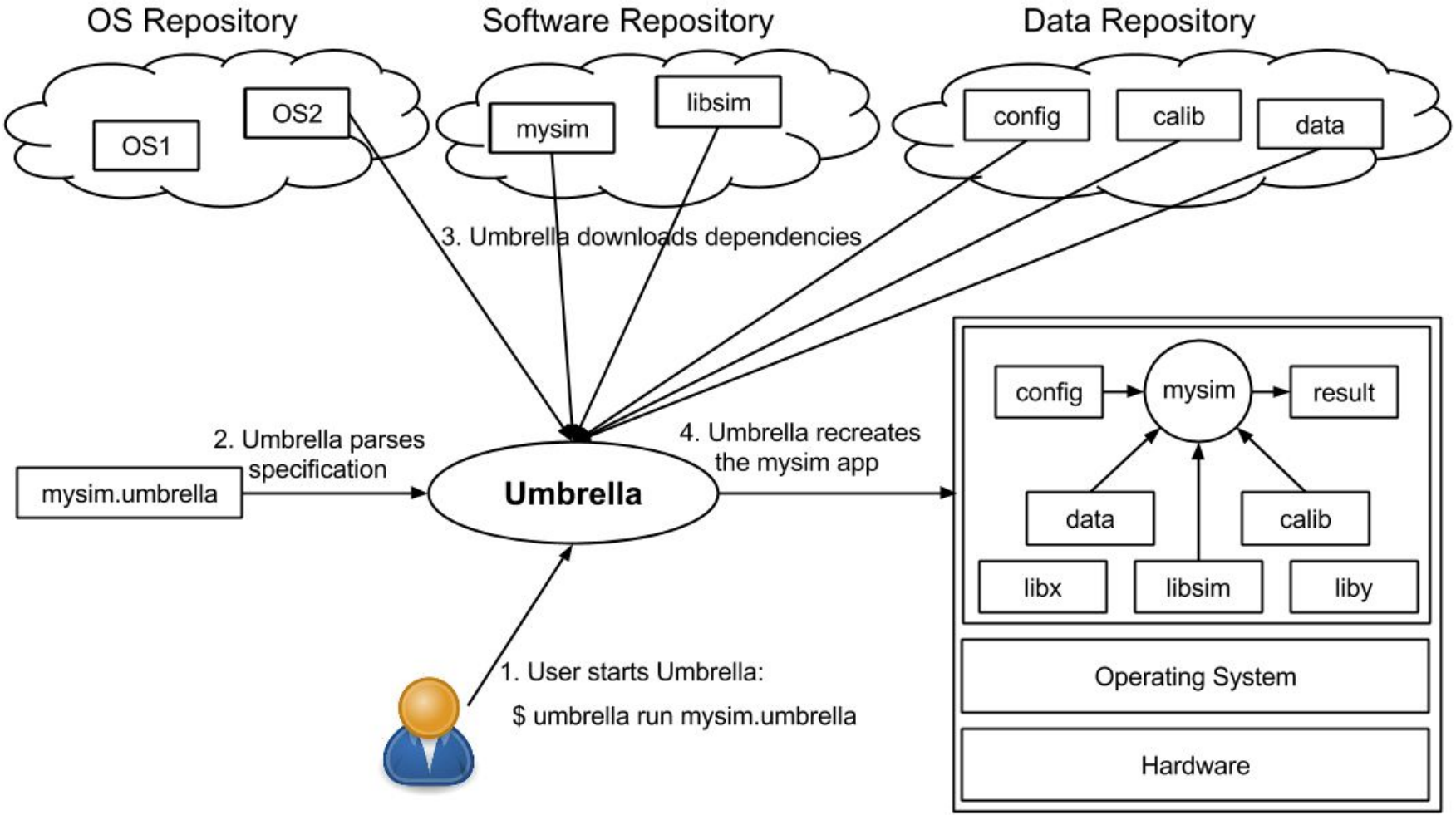
source checksum size

format mountpoint


```
{
  "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://ccl.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

| Resource | Example URL |
| --- | --- |
| Local Filesystem | /home/hmeng/data/input |
| HTTP | http://www.data.com/data/file1 |
| HTTPS | https://lab01.nd.edu/data/hep/file2 |
| Amazon S3 | s3+https://s3.aws.com/…/cubes.pov |
| Open Science Framework (OSF) | osf+https://files.osf.io/v1/…/7559c3a |
| Git Repository | git+https://github.com/…/cctools.git |
| CernVM File System | cvmfs://cvmfs/cms.cern.ch |


OS Repository
Software Repository
Data Repository
OS1
OS2
mysim
libsim
config
calib
data
3. Umbrella downloads dependencies
2. Umbrella parses specification
mysim.umbrella
Umbrella
4. Umbrella recreates the mysim app
config
mysim
result
data
calib
libx
libsim
liby
Operating System
Hardware
1. User starts Umbrella:
$ umbrella run mysim.umbrella

# Creating Execution Environment: Umbrella Execution Engine

**Matching degree** between

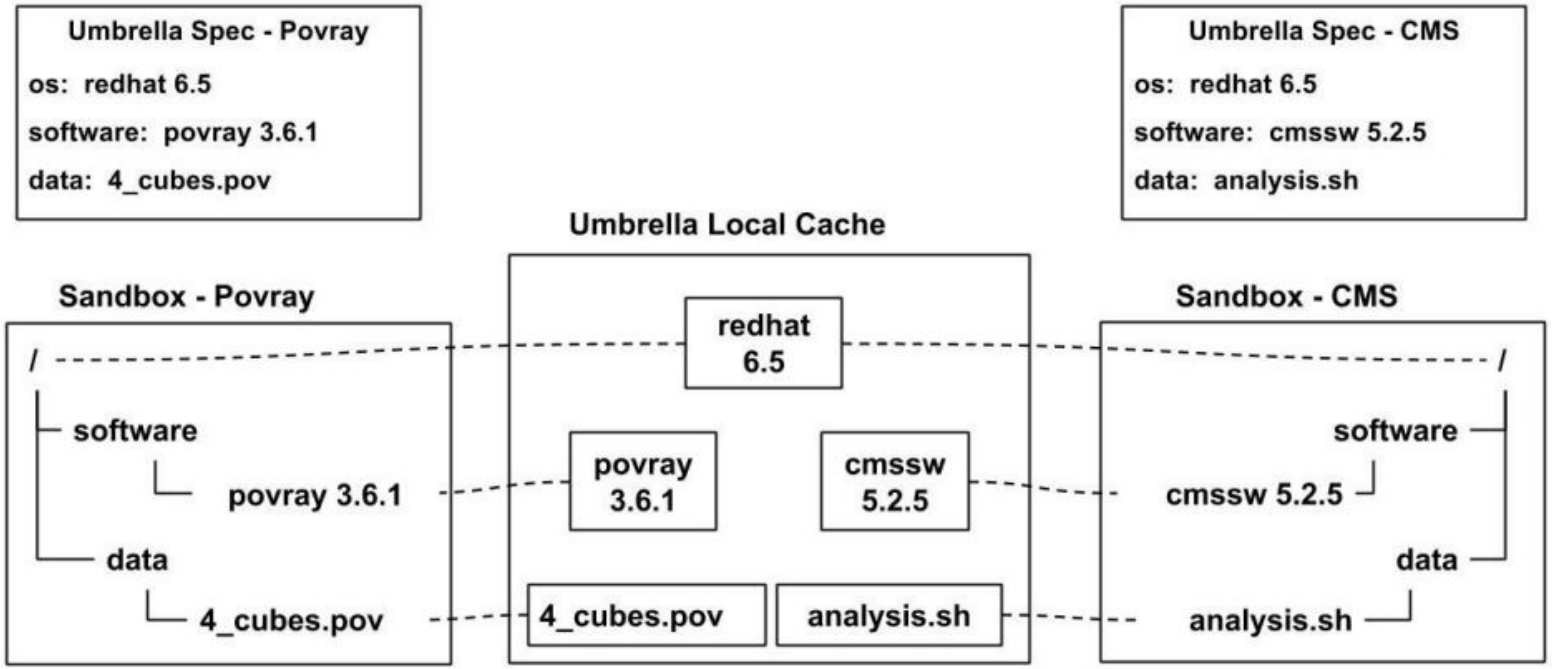
-- the execution node

-- the specified execution environment

| Hardware | Kernel | OS | Sandbox Techniques |
|---|---|---|---|
| Yes | Yes | Yes | Utilize the current OS directly |
| Yes | Yes | No | OS-level Virtualization<br>Docker, Parrot |
| Yes/No | No | No | Hardware Virtualization<br>Local: VirtualBox, VMWare<br>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 \
--localdir /tmp/umbrella_test/ \
--output "/tmp/frame000.png=/tmp/umbrella_test/parrot_povray_S/output.png" \
--sandbox_mode parrot \
--log umbrella.log \
run
```

# 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
```

**CCL Home**

**Research**

- Papers
- Projects
- People
- Jobs
- REU

**Software**

- Download
- Manuals
- Makeflow
- Work Queue
- Parrot
- Chirp
- Confuga
- Umbrella
- SAND
- AWE

**Community**

- Annual Meeting
- Workshops
- Forum
- Getting Help
- Highlights
- For Developers

**Operations**

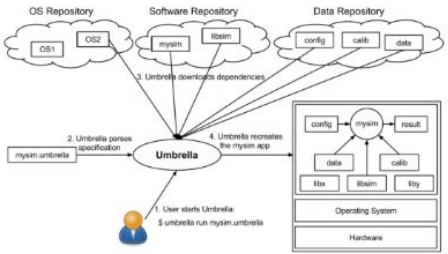
- Work Queue Display
- Condor Display
- Condor Pool
- Condor Log Analyzer
- Hadoop Cluster
- BXGrid
- Internal

## 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., VMware), 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 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

- Techniques for Preserving Scientific Software Executions: Preserve the Mess or Encourage Cleanliness? [Talk]. 12th International Conference on Digital Preservation (iPres 2015), Chapel Hill, North Carolina, November, 2015.
- Umbrella: A Portable Environment Creator for Reproducible Computing on Clusters, Clouds, and Grids [Talk]. 8th International Workshop on Virtualization Technologies in Distributed Computing (VTDC 2015) at HPDC, Portland, Oregon, June, 2015.
- Umbrella: A Portable Environment Creator for Reproducible Computing on Clusters, Clouds, and Grids [Poster]. 4th Greater Chicago Area Systems Research Workshop (GCASR) 2015 , Chicago, Illinois, April, 2015.

### Publications

(Showing papers with tag **umbrella**. See all papers instead.)

Haiyan Meng, Douglas Thain, Alexander Vyushkov, Matthias Wolf, and Anna Woodard,
**Conducting Reproducible Research with Umbrella: Tracking, Creating, and Preserving Execution Environments**,
*IEEE Conference on e-Science*, October, 2016.

Douglas Thain, Peter Ivie, 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/R0CZ353M

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: 10.1145/2755979.2755982