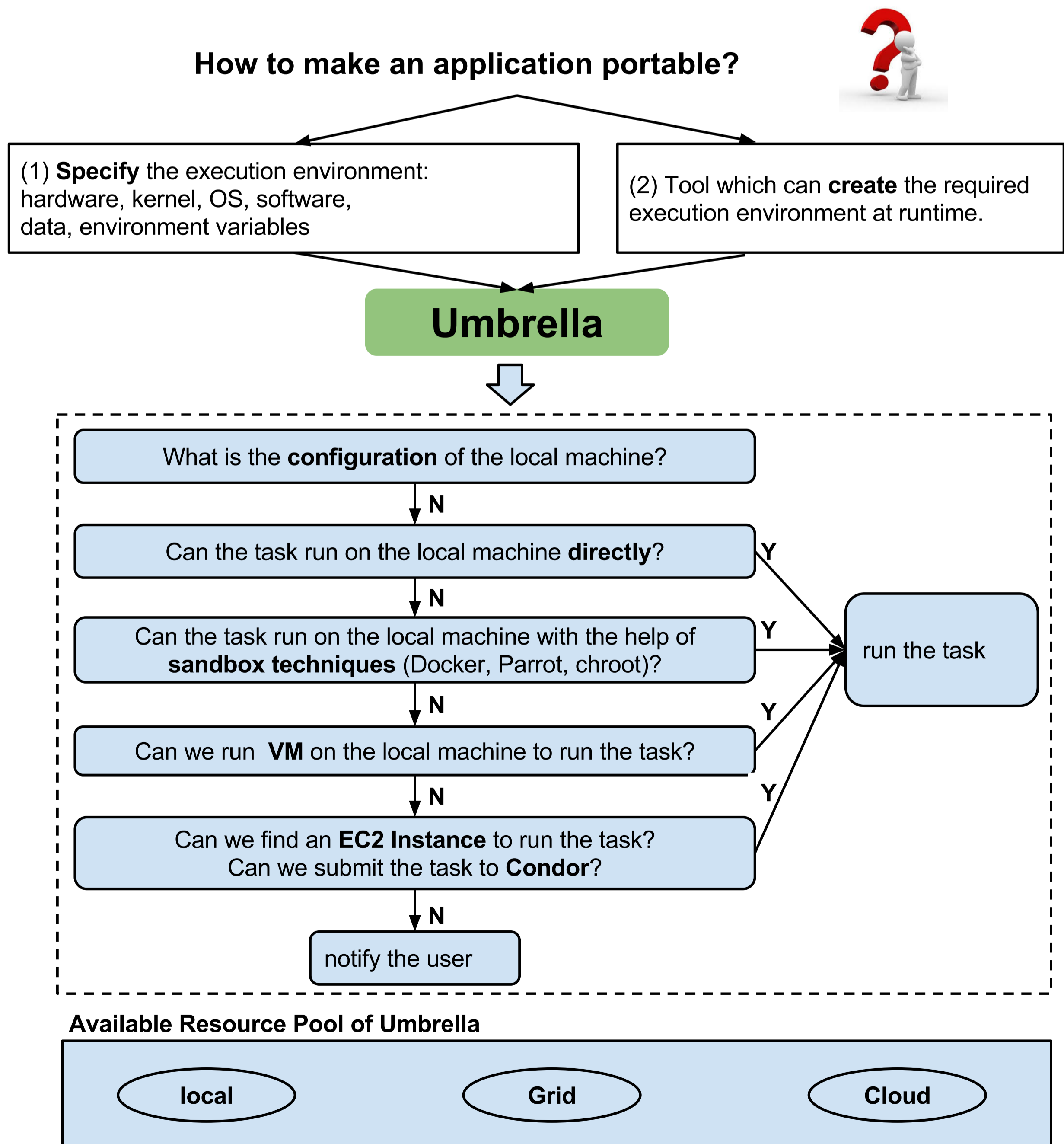
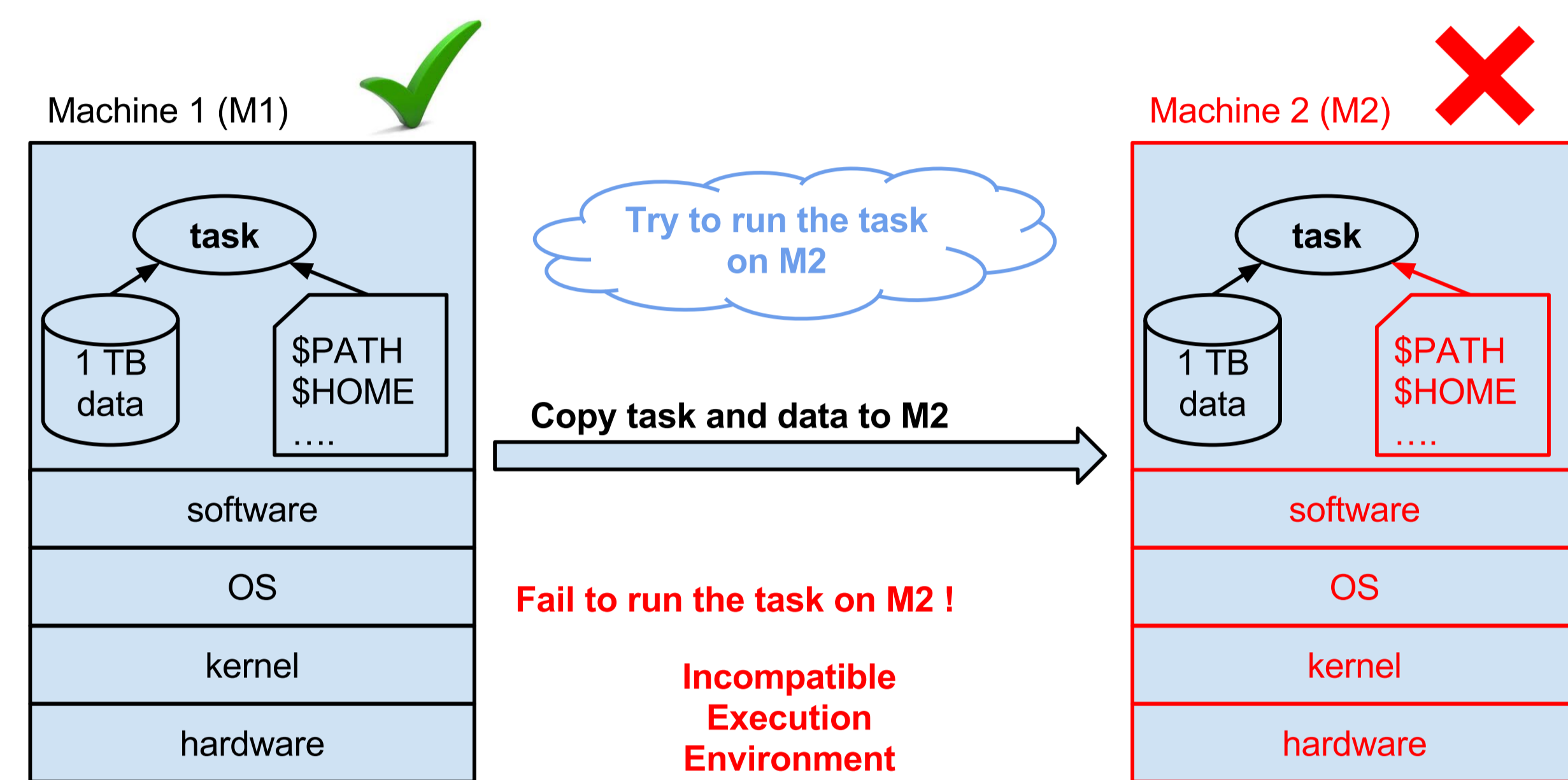


1 Abstract

Environment configuration is a significant challenge in large scale computing. An application that runs correctly on one carefully-prepared machine may fail completely on another machine, creating wasted effort and serious concerns about long-term reproducibility. Virtual machines and system containers provide a partial solution to this problem, in that they allow for the accurate reconstruction of an entire computing environment. However, when used directly, they have the dual problems of significant overhead and a lack of portability. To avoid this problem, we present Umbrella, 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 determines the minimum mechanism necessary to run the task - direct execution, a system container, a local virtual machine, or submission to a cloud or grid environment. We present the overall design of Umbrella and demonstrate its use to precisely execute a high energy physics application across many platforms using combinations of chroot, Docker, Parrot, Condor, and Amazon EC2.

2 Motivation



3 Architecture of Umbrella

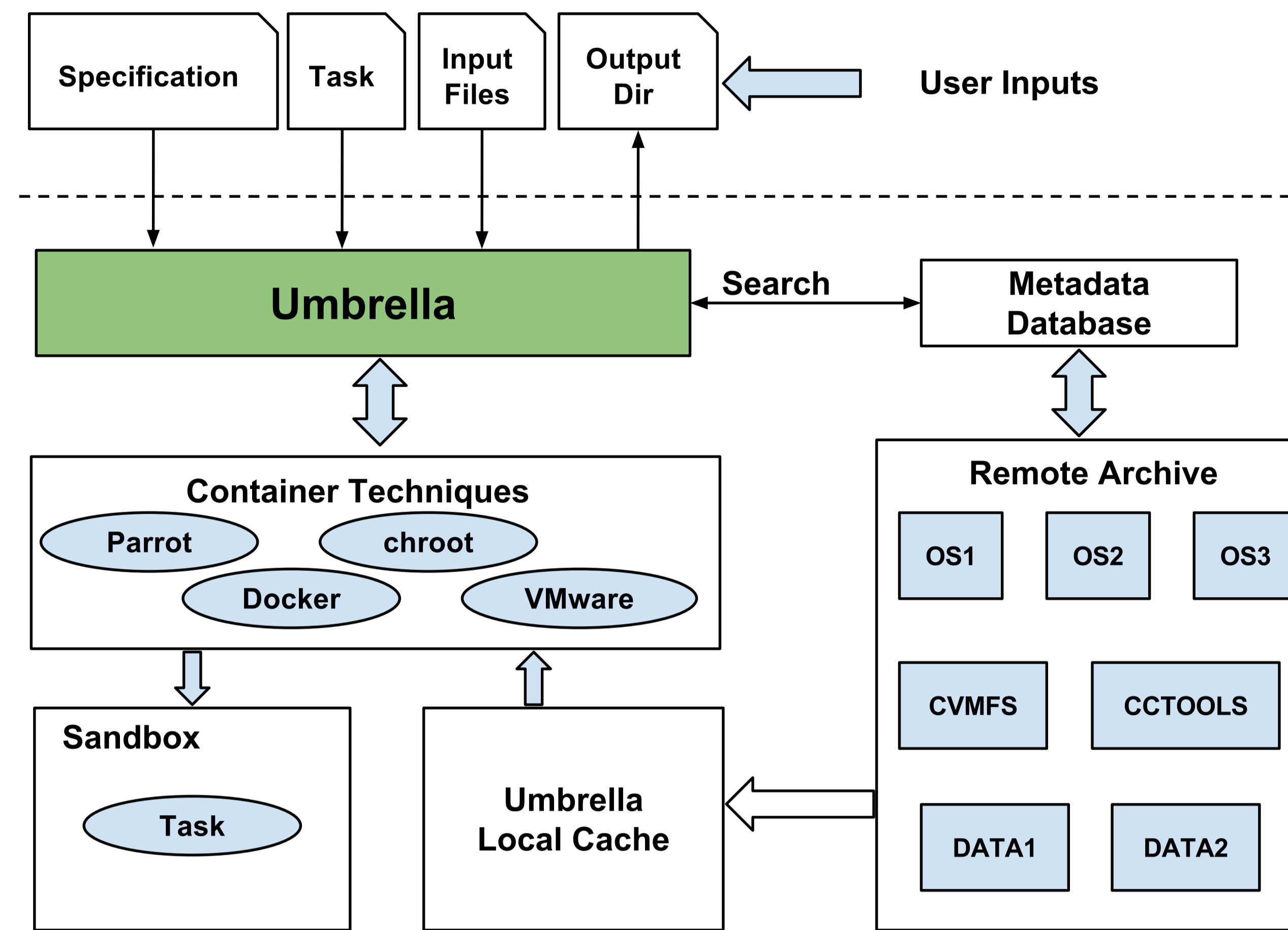


Figure 1: Architecture of Umbrella - Local Execution Engine

4 Specification

Figure 2 shows the specification for a CMS physics application. Umbrella allows a user to specify a dependency in two ways: unique identifier (one referent) and attribute description (a class of referents). The only exception is the `environ` section, which has a fixed syntax: `<env_name>:<env_value>`.

```
{
  "hardware": {
    "platform": "x86_64",
    "cpu cores": "1",
    "memory": "1 GB",
    "disk": "4 GB"
  },
  "kernel": {
    "type": "linux",
    "release": ">=2.6.32"
  },
  "os": {
    "name": "RedHat",
    "version": "6.5",
    "id": "669ab5ef9....6b7907a"
  },
  "software": {
    "cmssw-5.2.5-slc5-amd64": {
      "id": "f6e17cc80....a2f4e70",
      "mountpoint": "/cvmfs/cms.cern.ch"
    }
  },
  "data": {
    "LHE_v12-ttH_samples-2381": {
      "id": "cb9878132....14be8e2",
      "mountpoint": "/tmp/2381.lhe"
    }
  },
  "environ": {
    "CMS_VERSION": "CMSSW_5_2_5"
  }
}
```

Figure 2: Specification Example - CMS Data Analysis

5 Evaluation of Matching Degree

Umbrella deploys the minimum virtualization technology necessary to achieve the desired environment. (S1) If the host machine is fully compatible, the task is run directly. (S2) If the OS is compatible but some additional software or data are needed, Parrot is used to deliver the files. (S3) If only the kernel is compatible, Docker is used to deliver the operating system. (S4) If the kernel is not compatible, a virtual machine is created.

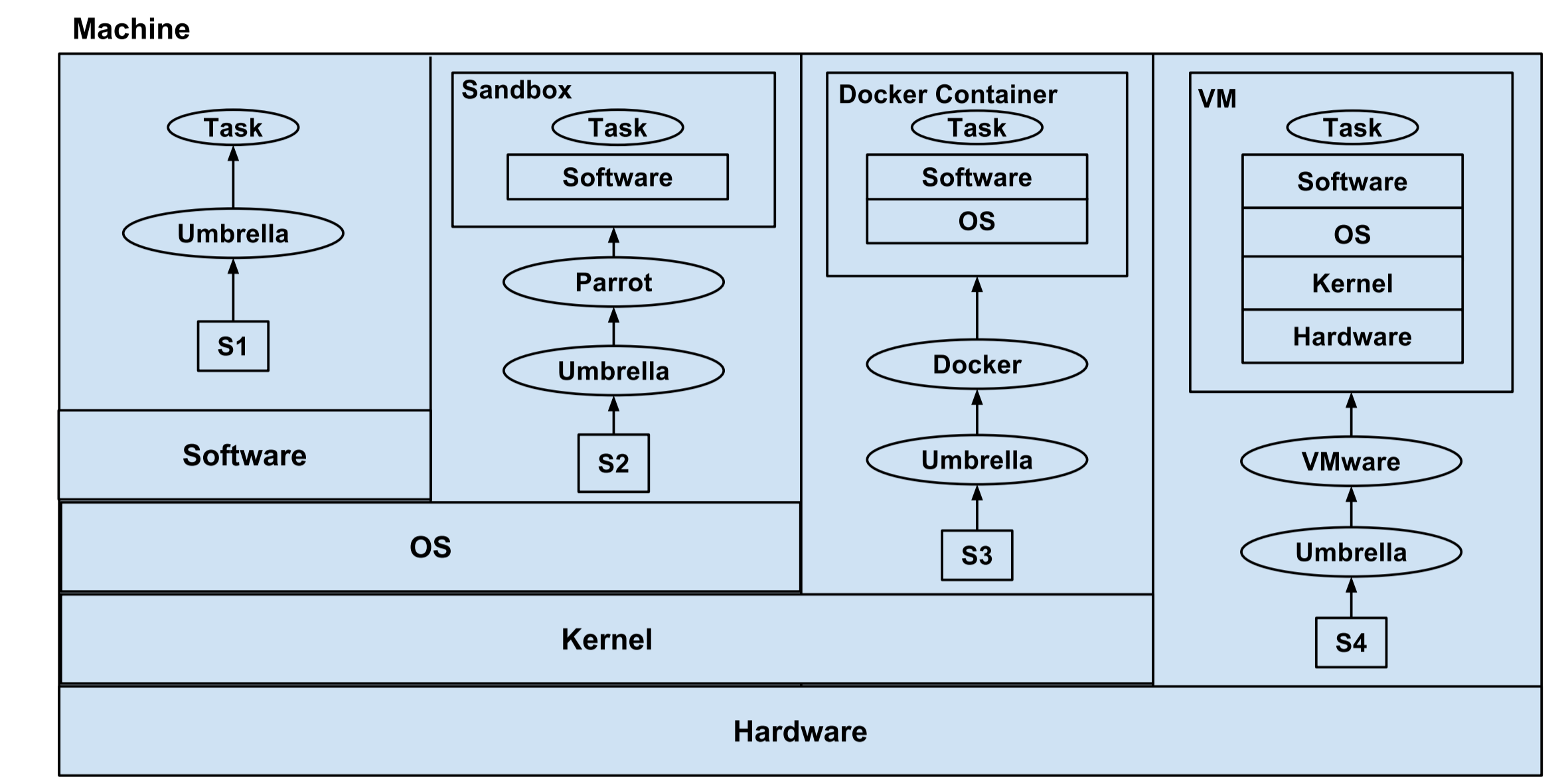


Figure 3: Umbrella Uses Varying Degrees of Virtualization

6 Local Cache

To minimize the execution environment construction time, each software dependency should be pre-built and configured. To improve the portability of archived software, software dependencies should conform to common-used internal organizations.

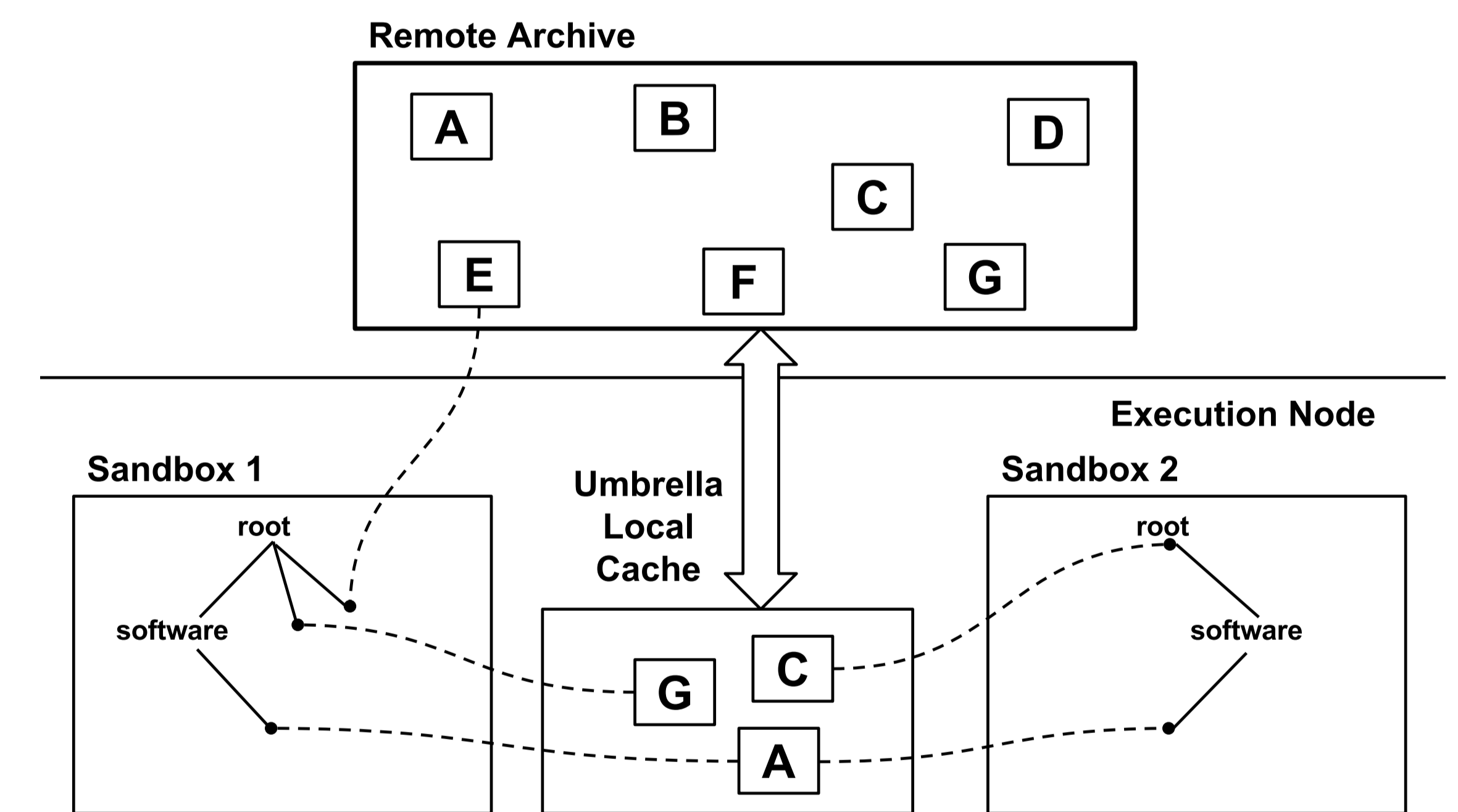


Figure 4: Mounting Mechanism

7 Evaluation

Sandbox Technique	Matching Evaluation	Software Preparation	Sandbox Construction	Application Execution	Post Processing	Total Time	Access Authority
Parrot	<1s	2m 11s	<1s	5m 34s	<1s	7m 45s	any user
chroot	<1s	2m 11s	<1s	4m 33s	<1s	6m 44s	only root
Docker	<1s	2m 11s	1m 24s	4m 35s	3s	8m 13s	docker group users

Figure 5: Time Overhead of Three Sandbox Techniques - Local Execution Engine

Subtask	Time
Start an EC2 Instance	6s
Send Task to VM	2s
Remote Execution	6m 40s
Post Processing	4s

Figure 6: Time Overhead (EC2)

Subtask	Time
Construct Submit File	<1s
Submit Condor Job	<1s
Remote Execution	6m 20s
Post Processing	<1s

Figure 7: Time Overhead (Condor)