PROCEEDINGS OF SCIENCE

# Batch-oriented software appliances

**Riccardo Murri**
*GC3: Grid Computing Competence Center*
*University of Zurich*
*E-mail:* riccardo.murri@gmail.com

**Sergio MAFFIOLETTI***
*GC3: Grid Computing Competence Center*
*University of Zurich*
*E-mail:* sergio.maffioletti@gc3.uzh.ch

This paper presents AppPot, a system for creating Linux software appliances. AppPot can be run as a regular batch or grid job and executed in user space, and requires no special virtualization support in the infrastructure. The main design goal of AppPot is to bring the benefits of a virtualization-based IaaS cloud to existing batch-oriented computing infrastructures. In particular, AppPot addresses the application deployment and configuration on large heterogeneous computing infrastructures: users are enabled to prepare their own customized virtual appliance for providing a safe execution environment for their applications. These appliances can then be executed on virtually any computing infrastructure being in a private or public cloud as well as any batch-controlled computing clusters the user may have access to. We give an overview of AppPot and its features, the technology that makes it possible, and report on experiences running it in production use within the Swiss National Grid infrastructure SMSCG.

*EGI Community Forum 2012 / EMI Second Technical Conference*
*26-30 March, 2012*
*Munich, Germany*

*Speaker.

 

## 1. Introduction

Application deployment and configuration on large heterogeneous systems is a complex infrastructure issue that requires coordination among various system administrators, end users as well as operation teams. This is further complicated when it comes to scientific applications that are, most of the time, not supported on many Linux distributions.

Virtualized infrastructures and software appliances provide an effective solution to these problems but do require a specific infrastructure and a usage model that is markedly different form the batch-oriented processing that is still the backbone of scientific computing.

This paper presents a system (nicknamed "AppPot") to bring the benefits of virtualization-based IaaS clouds to existing batch-oriented computing infrastructures.

AppPot comes in the form of a set of POSIX shell scripts that are installed into a GNU/Linux system image, and modify its start-up sequence to allow controlling task execution via the kernel boot command-line. Another utility is provided to invoke an AppPot system image from the shell command line, and request it to execute a given command.

## 2. Goals and use cases

The following scenarios are meant as an illustration of AppPot's intended use cases. Throughout the paper, we shall describe how the requirements from these use cases translate into design decisions for AppPot, and how well the goals have been met.

### 2.1 Deployment of complex applications

Some software packages (notably, many scientific codes) require complex installation procedures.

1. The application depends on software that is not readily available on the host operating system.

2. The application has a complex or non-standard compilation procedure, and the documentation is scarce.

In a grid infrastructure, this poses an additional problem of scale: all systems administrators must be conversant with the installation procedures of every software piece, and every application must be compatible with all the computing systems available in the infrastructure.

### 2.2 Running self-developed code

A large fraction of research groups are developing their own software applications; oftentimes for computational experiments that are ephemeral, or limited in scope to a local group or niche community.

Leveraging the User-Mode Linux [2] virtualization system, AppPot appliances can run on grid and local clusters as regular batch system jobs, without the need for sysadmin support or root access. This solves both the aforementioned problems:

- AppPot software appliances are a way to implement generic application deployment on a computational grid, and especially to enable users to provide their own software to the computing cluster: a complete AppPot appliance consists of three files, that can be copied to the execution cluster with any available mechanism, including the "stage in" facilities provided by most grid and cluster batch systems.
- Users can use an AppPot Virtual Machine (VM) on their computer for coding, and then run the same VM as a Grid jobs or in a Cloud Infrastructure as a Service (IaaS) infrastructure for larger tests and production runs.

## 3. Architecture and usage

An AppPot appliance appears to a user as consisting of a few files:

1. an *AppPot disk image*, which is a complete GNU/Linux system installed in a partition image file (in "raw" format);
2. an UML Linux kernel;
3. a shell script `apppot-start` used to run a command-line program within the AppPot appliance;
4. a few auxiliary programs that enable optional features of AppPot (networking, I/O streams redirection).

All these files can, all or in part, be installed system-wide so that many users can benefit from a shared installation.

### 3.1 Using AppPot

Users receive an AppPot system image, containing a working installation of a GNU/Linux distribution. Users have full access rights to the AppPot system image thus they can modify it by installing new software, libraries, reference data e.g., their own version of a computational code or a reference dataset that will be used during the computational analysis.

There are three main usage modes of AppPot, detailed below.

**Interactive local execution** In this case AppPot is started on a local machine; the disk image file as well as input data are directly available.

**Batch job on a cluster resource** In a typical cluster setup, appliance image file, User-Mode Linux (UML) kernel and input data are made available to the batch cluster execution node; the `apppot-start` command is invoked by the batch job to run a command non-interactively within the AppPot appliance.

**Grid job** AppPot can also be executed as a grid job on a distributed infrastructure. In this case, the disk image file, execution script and reference data need to be transferred to the destination node before the execution. This is normally achieved by specifying those input files as part of the grid job description file.

## 4. Real-world usage

Let's see now how the issues illustrated in Section 2 can be addressed using AppPot.

### 4.1 Deployment of complex applications

AppPot allows the execution of the appliance through UML; in this way, the appliance contains the complex application as well as its full dependencies and, at the same time, it spares the local site administrators to certify and monitor the application deployment. The execution within UML guarantees that the user cannot gain no more privileges than the executing user already can.

### 4.2 Running development code

In case application source code needs a frequent update cycle, re-deploy the software appliance all the time is not a scalable deployment model. AppPot provides a snapshot/changes mechanism for this: users can create a "changes" file that encodes the differences of the locally-modified appliance with a "base" one. During AppPot boot, the `apppot-init` script will re-create the modified appliance from the base one, by merging in the changes.

### 4.3 Dynamical expansion of clusters

A local cluster execution node can be prepared as an AppPot image that could executed on an accessible external computing infrastructure (the EGI infrastructure or any flavor of public clouds). The specific appliance is submitted each time a site seeks to expand its resources; the AppPot instance could be customized by the site admin to be a replica of the standard compute node, that connects back to the home site using a VPN. This is similar to the "glide-in" mechanism implemented in the Condor batch execution system [1].

## 5. Conclusions

AppPot is currently used in production within the Swiss National Grid Infrastructure SMSCG, supporting several use cases like those presented in this paper. We are collecting feedback on the effectiveness of AppPot in large-scale grid computations; we would like to stress that such effectiveness is not just a function of system performance, but should also include consideration of how it makes large-scale computing more accessible (on the users' side) and manageable (on the systems administrators side).

## A. List of acronyms

| | |
|---|---|
| **IaaS** | Infrastructure as a Service |
| **POSIX** | Portable Operating System Interface |
| **SMSCG** | Swiss Multi-Science Computational Grid |
| **UML** | User-Mode Linux |
| **VM** | Virtual Machine |
| **VPN** | Virtual Private Network |

## References

[1] Thain, D. and Tannenbaum, T. and Livny, M., Distributed computing in practice: The Condor experience, Concurrency and Computation: Practice and Experience, volume 17 2-4, pp 323–356, 2005.

[2] Dike J., User Mode Linux, Prentice Hall, 2006, isbn 978-0131865051.