Pulley: Secure Administration of Virtual and Remote Computing Systems

Douglas C. Turner
Coastal Carolina University
P.O. Box 261954
Conway, SC 29528 USA
dcturne1@coastal.edu

Michael Fenn
426 Bedford Rd Fl 2
Pleasantville, NY 10570 USA
mfenn@mfenn.com

Michael A. Murphy
Coastal Carolina University
PO Box 261954
Conway, SC 29528 USA
mmurphy2@coastal.edu

ABSTRACT
Pulley is an agent-based, asynchronous system management tool designed for computer systems to which the system administrator has no direct physical or network access. Pulley is being implemented using the Bash scripting language, with a flexible yet simple architecture designed for ease of application and adaptation to a variety of management situations. Initial test results of an early prototype of the system have been encouraging, as demonstrated through the efficient installation of library and application software on nodes of a cluster computing system.

Categories and Subject Descriptors
K.6.2 [**Management of Computing and Information Systems**]: Installation Management – *Computing equipment management*.

K.6.3 [**Management of Computing and Information Systems**]: Software Management – *Software maintenance, Software selection*.

K.6.4 [**Management of Computing and Information Systems**]: System Management – *Centralization/decentralization*.

General Terms
Management, Reliability, Security.

Keywords
System administration, distributed management, contextualization

1. INTRODUCTION
Computing environments are deployed in a wide variety of installations. Historically, most installations have been in the form of individual systems to which the system administrator has direct access, such as a workstation with an attached keyboard and monitor. However, as these systems are deployed in increasingly diverse environments, maintenance challenges arise. Systems might be deployed in remote locations without easy physical access and with only intermittent network access. In other cases, such as cloud computing environments, systems can be deployed to run on hardware with different owners, creating security restrictions for access. Like any other system, however, computing environments deployed in challenging situations still require ongoing management to update software, resolve issues, and patch security vulnerabilities.

A distinction may be made between *synchronous* and *asynchronous* management techniques. In this paper, synchronous techniques require direct action by the system administrator, who issues one or more commands to implement a configuration change. Such commands are issued via a shell or other terminal interface directly connected to the system, and major configuration changes are typically reserved for scheduled maintenance windows. In contrast, asynchronous techniques are implemented using an agent that runs on each system and carries out configuration changes periodically, without user intervention. Asynchronous systems typically scale better than synchronous systems, into the many thousands of nodes [7]. Synchronous management is the typical mode of operation for a single-user system such as a laptop or workstation, where the owner or administrator interfaces directly with the system. Such management techniques are also employed on clusters of computing systems with parallel Secure Shell (SSH) operations, possibly orchestrated by a tool such as Stoker [9], c3tools, or Tentakel.

Pulley is an asynchronous system administration tool under development for use in four unrelated situations: cluster computing systems, servers, laboratory workstations, and virtual machine appliances deployed on student laptops. In the case of cluster systems, Pulley is intended to integrate with the cluster job scheduler, so that software updates and system administration tasks can be carried out in idle periods between scientific jobs, eliminating the need to schedule a system-wide maintenance window for this purpose. In server environments, Pulley should be invoked by the cron system, so that security updates and configuration changes can be implemented quickly. For workstation systems, Pulley is best invoked during periods of time in which the system is idle or under low user loads, so that updates and configuration changes do not produce a noticeable performance impact on the system. In the final case, in which Pulley is deployed to manage a virtual computing environment used by a student on his or her personal laptop, Pulley is invoked upon demand by the user. In this situation, the user also has administrative access to the machine and can make configuration changes synchronously if desired.

To support the diversity of use cases in which Pulley could be deployed, the system is being designed to be flexible, powerful, and relatively simple. Instead of creating an entire management system with completely customized components, Pulley is designed to integrate into the installed computing environment and automate the tools already present.

The remainder of this paper is organized as follows. First, a brief survey of related work is presented in section 2. Then, the architecture of Pulley is described in section 3. A representative cluster managed by Pulley is presented in section 4, with installation results discussed in section 5. Finally, conclusions and future work are presented in section 6.

2. RELATED WORK

A number of system administration tools have been developed for managing compute clusters, server farms, individual hosts, and virtual machines [3], including Cfengine [2], Bcfg2 [4], Puppet [6], and Chef [12].

In addition to these flexible configuration tools, several special-purpose tools have been developed specifically for cluster computing systems. NPACI Rocks [10] is a well-known and widely used cluster system distribution based on open-source Enterprise Linux compilations. Software sets are grouped into rolls [1], which specify sets of software to be installed rapidly at cluster construction time. Centralized tools for synchronous management of the resulting systems are included [11]. A similar install-time configuration system is OSCAR [8], which uses metapackages to specify initial software sets, with the installed systems managed using synchronous tools.

Although any of these software management frameworks could be used to manage physical and virtual machines on which the user has administrative access, the objective of Pulley is to provide a flexible mechanism that can be deployed in a wide variety of situations involving federated ownership of the systems. Pulley is novel in that a single configuration specification can be deployed using nothing more than a simple HTTP server, and then disparate client systems may update their configurations and carry out administrative actions either in a completely automatic configuration or upon user request. In either case, users can retain administrative control over their own systems, using Pulley to manage only a subset of the overall system configuration. Furthermore, the adaptation of Pulley to new management scenarios does not require any administrator to learn a completely new declarative language to effect configuration changes. Instead, the administrator can leverage commodity Bash commands, which are typically familiar by virtue of Bash being the default shell on most Linux systems. Since Pulley is implemented via standard executable scripts, it can be easily integrated with meta-management layers such as Poncho [5].

3. IMPLEMENTATION ARCHITECTURE

Pulley is being implemented in the Bourne-Again Shell (Bash) scripting language, using an architecture that is conceptually simple and yet simultaneously powerful. This implementation follows the traditional Unix philosophy of linking together preexisting commands on the system to create a more powerful application. For example, Pulley does not implement its own network stack, relying instead on existing, commodity network transfer applications, such as curl and wget.

As illustrated in Figure 1, Pulley obtains its configuration from a networked server. In the general case, this network server is a standard HTTP web server, onto which two files are uploaded: a payload file, containing the configuration details; and a control file, which contains a cryptographic signature of the payload file. Whenever the system administrator updates the configuration, revised versions of both the payload and control files are uploaded to the server.

ACM Southeast'14, March 28-29, 2014, Kennesaw, GA, USA.

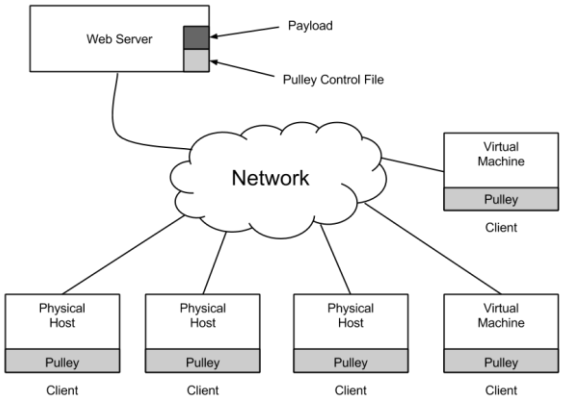

Figure 1. Pulley is designed for use with a simple network architecture, requiring only a commodity web server to host the payload and control files.

When a Pulley client is invoked, it first collects information about the state of the system (Figure 2). Once this collection procedure is complete, the client downloads the control file from the server and compares it to the saved version of the control file it previously acquired. If these two files differ, the Pulley client downloads the payload and validates its contents using the cryptographic signature contained in the control file. If validation succeeds, the new payload file is accepted as the working client configuration. Once this configuration update procedure is complete, the Pulley client invokes configuration agents that utilize system administration tools to effect configuration changes.

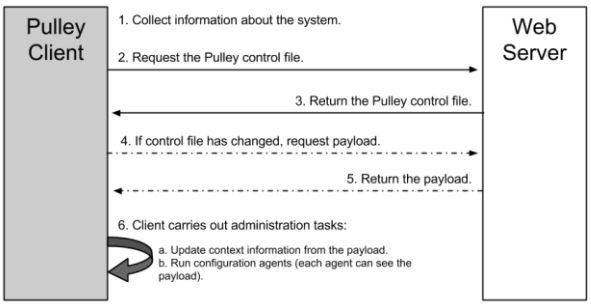

Figure 2. Interaction diagram illustrating Pulley client operation.

Pulley payload files are Bash scripts that are imported (sourced) into the main Pulley application after a download and verification process. These scripts contain directives that enable agents, configure system probes, and deliver configuration information to Pulley. Each directive is given in an imperative format, in the same manner as executing a regular command on the system. Internally, these directives function declaratively by specifying the initial state of Pulley data structures in preparation to run the configuration agents. Figure 3 illustrates a simple Pulley payload

that enables collection of network data and ensures that the “a2ps” package is installed on the system.

```
#!/bin/bash

URL="localserver/slackware64-14.1/slackware64/"
KEY="localserver/GPG-KEY"

enable_probe network
enable_agent package

if agent_enabled package; then
   add_repository --name main --url "${URL}" \
      --pubkey "${KEY}"

   add_package --from main ap/a2ps-4.14-x86_64-5
fi
```

Figure 3. Pulley payload format

The configuration agents are simply Bash scripts that run each time the Pulley client is invoked. These scripts are written to be idempotent, comparing the current state of the system to the desired state, and executing changes when needed to correct the system configuration. Configuration agents are enabled explicitly in the Pulley payload, so that only those agents actually required by the system administrator are executed.

Since the payload and configuration files can be hosted on commodity HTTP servers, the potential exists for a server compromise or man-in-the-middle attack, which would cause the clients to receive a corrupt payload. In order to mitigate this potential vulnerability, a cryptographic signature is applied by the system administrator to the Pulley payload file and stored in the control file, prior to uploading both files to the server. By default, the GPG security application is used to generate these signatures; however, the implementation permits other cryptographic applications to be substituted as desired by the system administrator.

4. CLUSTER CONFIGURATION

In order to perform an initial test of the Pulley implementation, a parallel installation procedure was performed on a set of ten cluster compute nodes, named fiji-00 through fiji-09. These nodes were linked to a local software mirror via Gigabit Ethernet, as illustrated in Figure 4. Each compute node was also connected to the cluster head node via a Double Data Rate (DDR) Infiniband connection. This Infiniband connection was used to mount Network File System (NFS) shares from the cluster head node, containing shared installations of precompiled scientific application software. In contrast, all system-level software was installed via the installation process, using Pulley to effect the installation of all except the bare minimal components of the operating system itself. The Pulley control and payload files were hosted using the Apache web server on the local software mirror.

At installation time, the compute nodes were configured to boot from the network, registering a timestamp in a special-purpose Boot Notification Server at the beginning of the installation process. The initial process of preparing the installation, formatting the system partitions, and installing a minimal subset of software was completed using Red Hat Kickstart. Pulley was then invoked in a post-installation script executed by the Kickstart process.

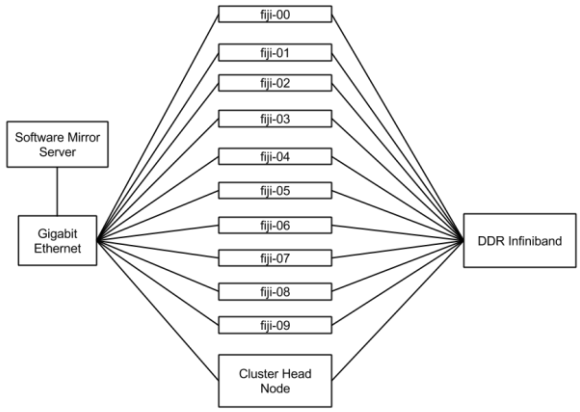

Figure 4. Cluster partition with 10 compute nodes used for testing

After invocation, the Pulley client on each compute node contacted the software mirror server to obtain a system configuration payload. This payload included directives for installing a selection of libraries and application software as well as post-install operations required to integrate the compute nodes into the Kerberos authentication system and connect them to the Slurm resource manager.

Once the configuration process was complete, the Pulley client finished its execution, and the post-install script notified the Boot Notification Server of the completed installation. The Boot Notification Server calculated the difference in timestamps from the beginning of the installation process to the end of the configuration process, producing a recorded installation time for the system. Upon completion of the configuration process, each node rebooted automatically onto the newly installed platform.

5. RESULTS

As presented in Table 1, the installation process over the 10 compute nodes required a mean time of 740.0 seconds, with a standard deviation of 18.40 seconds. The median installation time of 735.1 seconds indicated that the distribution was slightly skewed toward a few systems installing more slowly than the majority. However, since the range of the installation times was 711.2 seconds to 764.0 seconds, this skew was insignificant in the broader context of a system install, with less than one minute of install time variation from the slowest system to the fastest system.

Figure 5 reveals that systems tended to finish in groups, with fiji-02, fiji-04, fiji-06, and fiji-07 finishing at close intervals in the largest group. This behavior was consistent with a parallel installation off a single shared software mirror and likely was caused by the mirror server buffering the installation packages in memory. Once one or more systems became out of sync with the large group, those systems were able to complete the process more quickly or less quickly, depending upon resource contention at the mirror server.

Table 1. Descriptive statistics for the installation process

Metric	Value
Minimum	711.2
Median	735.1
Maximum	764.0
Mean	740.0
Standard Deviation	18.40

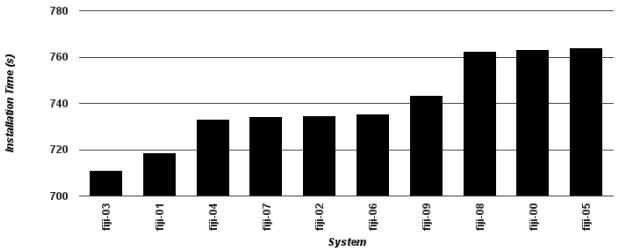

Figure 5. Cluster node installation times

It should be noted that the installation times presented in Table 1 and Figure 5 did not account for the actual boot times of the cluster nodes, since the Boot Notification Server received timestamp notifications only upon starting and finishing the actual installation process. Manual measurements indicated a nominal 100 second boot process for the type of hardware used in the cluster nodes, including the BIOS, disk controller, and network boot initialization processes.

6. CONCLUSIONS AND FUTURE WORK

Pulley is an asynchronous remote system administration tool that is being implemented using a relatively simple architecture in the Bash scripting language. As illustrated by the cluster installation results, Pulley is capable of orchestrating system management processes efficiently. Since the architecture of Pulley enables the payload and control files to be located on a remote HTTP server, the system can be used to manage machines that are not directly reachable by the system administrator.

As development on Pulley progresses, the system is being refined to support multiple Unix-like platforms. Also, modifications are planned to enable the Pulley client to use alternate transports and more flexible mechanisms for delivering the payload and control files. These modifications will enable the system administrator to customize Pulley to match the requirements of specific use cases as well as to integrate Pulley with other system administration and management applications.

7. REFERENCES

[1] G. Bruno, M. J. Katz, F. D Sacerdoti, and P. M. Papadopoulos. Rolls: Modifying a standard system installer to support user-customizable cluster frontend appliances. In *IEEE International Conference on Cluster Computing*, September 2004.

[2] M. Burgess. A tiny overview of Cfengine: Convergent management agent. In *1st International Workshop on Multi-Agent and Robtoic Systems*, 2005.

[3] T. Delaet, W. Joosen, and B. Vanbrabant. A survey of system configuration tools. In *24th Large Installation System Administration Conference*, San Jose, CA, November 2010.

[4] N. Desai, R. Bradshaw, J. Hagedorn, and C. Lueninghoener. Directing change using Bcfg2. In *Twentieth Large Install System Administration Conference (LISA XX)*, Washington, DC, December 2006.

[5] S. Devoid, N. Desai, L. Hochstein. Poncho: Enabling Smart Administration of Full Private Clouds. In *27th Large Installation System Administration Conference*, November 2013.

[6] V. Hendrix, D. Benjamin, and Y. Yao. Scientific cluster deployment and recovery – using puppet to simplify cluster management. *Journal of Physics: Conference Series*, 396:042027, 2012.

[7] M. Merlin. Live Upgrading Thousands of Servers from an Ancient Red Hat Distribution to 10 Year Newer Debian Based One. In *27th Large Installation System Administratioin Conference*, November 2013.

[8] J. Mugler, T. Naughton, and S. L. Scott. OSCAR meta-packages system. In *19th International Symposium on High Performance Computing Systems and Applications*, May 2005.

[9] M. A. Murphy, M. Fenn, L. Abraham, J. A. Canter, B. T. Sterrett, and S. Goasguen. Distributed management of virtual cluster infrastructure. In *Fifth International Workshop on System Management Techniques, Processes, and Services (SMTPS '09)*, Rome, Italy, May 2009.

[10] P. M. Papadopoulos, C. A. Papadopoulos, M. J. Katz, W. J. Link, and G. Bruno. Configuring large high-performance clusters at lightspeed: A case study. In *Clusters and Computational Grids for Scientific Computing 2002*, December 2002.

[11] F. D. Sacerdoti, S. Chandra, and K. Bhatia. Grid systems deployment and management using ROCKS. In *IEEE International Conference on Cluster Computing*, September 2004.

[12] F. Önnberg. Software configuration management: A comparison of Chef, CFEngine and Puppet. Master's thesis, University of Skövde, 2012.