Building and Testing a Production Quality Grid Software Distribution for Open Science Grid

A Roy on behalf of the OSG consortium

University of Wisconsin-Madison, Department of Computer Sciences, Madison, WI, USA

Email: roy@cs.wisc.edu

Abstract. We describe how we integrate, build, and test the Open Science Grid (OSG) software stack, which is used to provide a production quality infrastructure for grid sites and users across OSG to run their grid jobs. The software stack is sufficiently complex that building and testing the software stack is non-trivial and requires a variety of types of testing including internal integration testing as well as end-user testing.

1. Introduction

The Open Science Grid (OSG) [1] is a national, distributed computing grid for data-intensive research. It brings together resources from campuses and research communities around the nation and the world into a common, shared grid infrastructure via a common set of software. OSG is a production facility that provides millions of CPU hours per week to scientific researchers.

In order to provide this production quality facility, the OSG provides a reference software distribution to all sites and users. (A *site* accepts computational jobs and/or data to be stored, while a *user* accesses various sites.) While sites and users can choose to provide their own software, the OSG provides testing and support for the reference distribution, and most people in OSG choose to use it.

The OSG software stack is divided into two pieces. The first is the Virtual Data Toolkit (VDT). [2] This contains nearly the entire OSG software stack. (The name "Virtual Data" is historical and derives from the goals of a precursor project, iVDGL.) An important goal of the VDT is to be a grid software distribution that is grid-agnostic. That is, it does not contain specific configuration required to be part of a specific grid. In fact, portions of the VDT are used by a variety of other grids, such as LCG [3], TeraGrid [4], and campus/regional grids such as NYSGrid [5].

The second portion of the OSG software stack contains configuration information specific to OSG. For example, there is a configuration program that knows how to adjust the settings for various software components to point at central OSG services, such as the GRATIA accounting service.

The effort to create the OSG software stack is primarily an integration effort: we write almost none of the software that is in the software stack, but we instead build, test, and package a set of software that we know works together reliably.

This paper describes how we build, test, and distribute the OSG software stack.

1.1. Goals of the OSG software stack

There are three major goals for the OSG software stack:

- The software should include all of the grid software needed by the OSG stakeholders. However, we do not distribute the science applications themselves—just the grid software.
- It should be easy to install and configure the software.
- It should support as many operating system/CPU architecture combinations (a.k.a *platforms*) as possible. We do not mandate a single platform because sites in OSG are autonomous: they do not receive funding to join OSG, and they make their own choices about the kinds of hardware and software they deploy. Also, it is our goal for users to be able to access as many resources as possible, which requires us to support a variety of platforms. In VDT 2.0.0 (the latest release as of this writing), the VDT supports 22 different platforms [6].

1.2. Design choices in the OSG software stack

We have made several design decisions about the software stack.

- *Binary distributions*: Similar to most OS distributions, but different from some such as FreeBSD Ports, we distribute binaries, not source. In our experience, building all of the different software is both complex and slow, so we prefer to do this on behalf of users. For example, software components require many other pieces of software in order to build (such as GNU make, autoconf, or Ant), and often have very specific requirements on the versions of this underlying software. While all of this can be accommodated on a user's computer, it is complex and error-prone.
- *Configuration*: We provide tools for configuring the software to work together in specific scenarios. For example, we provide configuration to install specific web applications into Tomcat, to connect Tomcat with Apache, and to configure specific security settings across Tomcat and Apache.
- *Pacman*: The majority of our software distribution is done with Pacman. [7] Pacman is not a widely used tool, but it has several significant advantages. In particular, users can easily install as any user on the system, not just the root user (like the usual usage of RPM), and this enables easy client tool installations. Also, users can easily install the software simultaneously in multiple locations, which can be very useful for using and testing multiple versions of the software. We do distribute some software as native packages (RPMs and debs) and will increase support in the future: see Section 5 for more details.
- *Focus on integration*: We focus on integrating software that we know works together. In some cases, we will ship software that users might expect to get from their OS (such as MySQL) because we require specific versions that are known to work together reliably.

1.3. An example software integration

The OSG software stack includes a wide variety of grid software such as Condor [8], Globus [9], dCache [10], VOMS [11] the OSG Matchmaker [12] and much more. It also includes non-grid support software such as Apache [13], MySQL [14] and more. As an example of how the OSG software stack integrates these different software components, consider VOMS.

VOMS is software that maintains membership lists for Virtual Organizations (VOs), and issues X509 attribute certificates that can be used as to authorize users. VOMS includes a back-end daemon and a front-end web interface, which itself relies on Tomcat, and relies on a specific version of MySQL. There is also a plug-in for Globus (which accepts job submissions) that compiles against the VOMS libraries and uses the VOMS attribute certificate for authorization. All of these components need to be built, packaged and tested against each other, and configuration scripts were created to link them together easily for users.

2. Related work

Linux distributions face a similar problem to the OSG software stack, though they focus on a single operating system. Also they include software that is lower-level (such as the kernel or basic Linux commands) than are included in the OSG software stack.

Other grid efforts also build and test software stacks. A notable example is the gLite software distribution [15] which is part of the European EGEE project. There are many interesting comparisons and contrasts to draw which we are unable to include due to space constraints.

3. Building the OSG software stack

A principle of our software distribution is that we must be able to build all components that we ship from their source code. While there are a few exceptions to this rule (which we are working to correct), the VDT currently includes more than forty software builds. Building the software can be complicated because we build each one of the software components on many platforms. For example, in VDT 2.0.0 (the latest release at the time of writing), we built most software on 14 to 16 platforms, where "platform" is defined as the combination of a specific OS version (such as RedHat Enterprise Linux 5.x) and CPU architecture (such as x86). In some cases, we do not support a software component on all platforms, or we re-use software on multiple platforms.

Building forty software components on sixteen platforms requires careful management. We rely on the NMI Build and Test Facility [16] at the University of Wisconsin-Madison, which uses the Metronome software [17] to manage builds. We have built a thin layer on top of Metronome that allows us to extract build information from our source code repository, submit a build to Metronome, and stage the binaries from the build facility into the VDT software distribution. While builds often have to be carefully crafted in order to run successfully on multiple platforms, once they have been written we can easily manage the builds and use the resulting software.

To support our goal of smoothly integrating software for users, we sometimes patch the software to provide bug fixes or features needed by OSG stakeholders.

4. Testing the OSG software stack

When supporting the wide variety of platforms we support in the complex software environment we have in OSG, testing is absolutely critical to our success. We have developed a software testing process that works quite well and helps us iron out many of our bugs. An important aspect of this process is that we not only test the software integration, but we also test that the software installation process works correctly.

Our software testing process has three steps: internal daily automated testing by the VDT team, validation on a small testing grid in OSG (the validation testbed), and end-user testing on a larger testing grid in OSG. Each one of these steps is crucial to our testing process.

4.1. VDT internal testing

During development of a new VDT release, we perform daily, automated, integration tests of most of the software in the VDT. By *integration tests*, we try to differentiate these tests from the unit and functional tests that software developers often think about, though there are similarities. We expect that developers will do unit and functional testing and we only perform those tests lightly, or to ensure that old bugs are not re-introduced. Instead our integration tests verify that we have built the software correctly and it works together as expected. Recall our example of how we distribute VOMS in Section 1.3. In our testing, we will probe the entire system: can we create a VOMS attribute certificate and use it to authorize a job submission? This will test the breadth of our installation.

We do not label a platform as "supported" unless we run our tests on that platform. In some cases, we support platforms that are very similar to each other; for example we support CentOS 5 which is a freely-available rebuild of Red Hat Enterprise Linux 5. Because one of our major OSG stakeholders uses CentOS 5, we test on it in addition to Red Hat, to be absolutely certain that the software works on both platforms.

Our testing framework installs the software just as a user would install it, so we are not only testing the software, but the installation process. This is particularly important because we are testing an integrated entire software distribution.

We do not stop testing software once it is released. Because we are testing the installation process and this involves downloading the software from our web site, we want to ensure that nothing has been accidentally changed that would break a user's installation.

We normally run the tests continuously, but we can also run the tests on demand for immediate developer feedback.

Results of tests are examined in two ways. First, there is a daily email at 8:00am, so the VDT developers can look over the tests from the previous night for any problems that need to be addressed. Second, the results of the daily builds are stored in a database so that we can look at past results to compare them.

Finally, we leave the software installed for a day or two, so test failures can be easily examined. Recall that the VDT software can be installed multiple times on a single computer using Pacman. We take advantage of this and keep many recent installations in place. The reports for our test results include the location on disk of the installation, and a developer can often quickly debug a failed test because the installation is still there. Installations are automatically rotated to ensure that we leave sufficient disk space for new tests.

4.2. Validation Testbed

When the VDT team produces a new VDT release or pre-release, it is given to the OSG Validation Testbed (VTB). [18] The VTB team creates the OSG software stack; recall that this is a small addition to the VDT to provide OSG-specific configuration. Once it is created, three to five grid sites install the software. The testers ensure that the installation process goes smoothly with no errors. They then run a small set of "smoke tests", such as a running a simple grid job or transferring a file. While the tests are not extensive, they provide two important kinds of feedback.

First, they are external testers not connected with the VDT development, so they can provide feedback on all aspects of the software installation and usage that is not colored by too intimate a knowledge of the VDT.

Second, their environments are not identical to the ones tested by the VDT team. There is a wide variety of software that they have pre-installed: different Linux kernels, different revisions of the OS, different batch systems, different file systems, and more. Although it cannot reproduce the full range of conditions that will be experienced by end-users, it does provide an important variety that helps eliminate bugs.

The VTB provides feedback and bug reports and these are then are addressed by the VDT team and result in a new version of the VDT and OSG software stacks.

The VTB testing is usually relatively quick: 2–4 weeks. Because it is a small group of testers, interaction with the VDT team tends to be intimate and quick, and ideally there is a quick turnaround on problems.

4.3. Integration Testbed

No matter how good a testing process is, it is important to include end-user testing in scenarios that are as close to production as possible. The OSG Integration Testbed (ITB) [19] is a grid of 10-20 sites. Each site donates a small number of computers and storage to provide a production-like environment to run grid jobs. Unlike the VTB, the testing is not done by the sites, but is done by the end-users in OSG. They are members of the various *Virtual Organizations* (VOs) that have scientific applications that need to be run. They dedicate some time to the ITB testing process and verify that their scientific workflows operate correctly with the new version of the OSG software stack. With a combination of the scientists verifying their workflows and the site administrators watching their ITB sites, we have a good assurance that a new software release works as advertised.

Usually the ITB process does discover a number of problems, which are resolved, provided as software updates and re-tested.

OSG has about 15 active VOs at any given time, but not all of them are able to dedicate time to ITB testing. Therefore, our testing is unable to be 100% comprehensive. However, we find that this

process helps fix problems before the software is released to production, gives confidence to our users that the software really works, and assures VOs and production grid sites that it is safe to upgrade to the new version. This involvement from the community in testing the software has been invaluable.

4.4. Production

After the software has been tested, it is released as a new production release. In our experience, it takes a long time (many months) for grid sites to upgrade to new versions of the software, even if it is generally agreed to work and be a useful upgrade. Site administrators are busy with a wide variety of projects and need to carefully schedule downtime at their sites. Therefore OSG always supports both the current release and the previous release in tandem. As much as possible, we maintain backwards compatibility between the current release and previous release.

4.5. Security fixes or minor updates

The process described above is used for major upgrades that require significant testing. However, we often have minor bug fixes, and these do not require the heavyweight process described above. Similarly, security fixes must sometimes be delivered much more quickly than this process allows.

In these cases, we rely on either just the VDT testing, or perhaps also some VTB testing to validate that the software update will not change behavior. While there is a risk of introducing new bugs or incompatibilities, the reduced time to ship important updates is usually worth the risk.

5. Future plans and conclusions

As any software developer knows, testing is never thorough enough or complete enough. There is always more testing that can be accomplished. In the future, we plan to expand the thoroughness of the VDT internal testing, and add some automated test suites that VTB and ITB grid sites can use to verify the basic functionality of the software.

As successful as Pacman has been in meeting many of our packaging needs, it is often unsatisfying to system administrators who prefer packages that are installed just like the software they receive from their OS vendor. Therefore we have begun producing native packages (RPMs and Debian packages) to better meet the needs of system administrators, and we expect to improve our support for native packages in the coming months.

Even with more work to be done, we have found this three-step process (internal testing, external validation, end-user validation) to be enormously useful in providing a stable, production-ready grid software distribution.

Acknowledgements

Although the author list for this paper is short, many people have contributed to the work described here. While it is too difficult to list everyone, a few groups and people must be acknowledged. The current VDT team includes Tim Cartwright, Alan De Smet, Scot Kronenfeld, Tanya Levshina, and Neha Sharma. Earlier members of the VDT Team Kirill Kireyev and Nate Mueller were instrumental in the initial development of the VDT testing framework. The OSG Integration Testbed effort led by Rob Gardner and the Validation Testbed effort led by Suchandra Thapa have provided not only tireless testing, but also many refinements on how we should test. The OSG Operations Group, currently led by Rob Quick, has helped us move from testing to production many times. The OSG management, particularly Miron Livny and Ruth Pordes, have provided excellent technical guidance and have helped us keep our sights on the big picture. And many more people have provided daily input, feedback, and guidance: a great thanks to you all.

References

[1] The Open Science Grid Web Site, http://www.opensciencegrid.org
Avery, Paul 2007 Open Science Grid: Building and sustaining general cyberinfrastructure using a collaborative approach *First Monday* **12**

[2] The Virtual Data Toolkit web site http://vdt.cs.wisc.edu
[3] Worldwide LHC Computing Grid web site, http://lcg.web.cern.ch/LCG/
[4] TeraGrid web site, http://www.teragrid.org/
[5] NYSGrid web site, http://www.nysgrid.org
[6] VDT 2.0.0 supported platforms, http://vdt.cs.wisc.edu/releases/2.0.0/requirements.html
[7] Pacman web site, http://atlas.bu.edu/~youssef/pacman/
[8] Litzkow M, Livny M, and Mutka M 1988 Condor - A Hunter of Idle Workstations, *Proceedings of the 8th International Conference of Distributed Computing Systems*, 104-11
Thain D, Tannenbaum T, and Livny M, Distributed Computing in Practice: The Condor Experience *Concurrency and Computation: Practice and Experience*, **17** 323-356
[9] Foster I, Kesselman C 1997 Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputing Applications* **11** 115-28
Foster I 2006 Globus Toolkit Version 4: Software for Service-Oriented Systems, *IFIP International Conference on Network and Parallel Computing* 2-13
[10] Ernst M, Fuhrmann P, Gasthuber M, Mkrtchyan T, Waldman C, 2001 dCache: a distributed storage data caching system, *Computing in High-Energy Physics*
[11] Alfieri R, Cecchini R, Ciaschini V, dell'Agnello L, Frohner Á, Gianoli A, Lörenty K, and Spataro F 2003 VOMS, an authorization system for virtual organizations *European Across Grids Conference* 33-40
[12] OSG Matchmaker web site, http://osgmm.sourceforge.net/
[13] Apache httpd web site, http://httpd.apache.org/
[14] MySQL web site, http://www.mysql.com/
[15] gLite web site, http://glite.web.cern.ch/glite/
[16] NMI build and test web site, http://nmi.cs.wisc.edu/
[17] Pavlo A, Couvares P, Gietzel R, Karp A, Alderman I, Livny M, and Bacon C. 2006 The NMI Build and Test Laboratory: Continuous Integration Framework for Distributed Computing Software, *Proceedings of LISA '06: Twentieth Systems Administration Conference* 263-73
[18] OSG validation testbed web site, https://twiki.grid.iu.edu/bin/view/Integration/ValidationTestbed
[19] OSG integration testbed web site, https://twiki.grid.iu.edu/bin/view/Integration/WebHome