Using Globus With FreeBSD

Brooks Davis, Craig Lee
The Aerospace Corporation
El Segundo, CA
{brooks,lee}@aero.org

Abstract

After years of development by the high performance computing (HPC) community, grid computing has hit the mainstream as one of the hottest buzzwords in computing technology today. This paper examines the issues involved in integrating FreeBSD with the Globus Toolkit, the de facto standard for grid computing. Particular attention is paid to interactions between Globus and FreeBSD's package management system. Ways in which FreeBSD could be enhanced to make it a more attractive platform for Globus are also discussed.

1 Introduction

Grid computing has been a hot topic in the high performance computing (HPC) community for many years now. From the earliest beginnings with the I-WAY [DeFanti] at SuperComputing 1995 to today's commercial grid products such as Oracle 10g, grid computing has entered the mainstream. Most major OS vendors including Sun, Microsoft, and IBM have grid products.

By now, most people in the computing field have heard of grid computing, but many may still ask, what is a grid? Grid computing seems to mean different things to different people. Some people associate grid computing with a specific software package such as the Globus Toolkit [Foster1] or Sun Grid Engine [SGE]. Others think of peer-to-peer systems as their prototypical grid. One early definition is from The Globus Alliance's leaders, Ian Foster and Carl Kesselman: "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [Foster3]. A more recent definition from Wolfgang Gentzsch is "a Grid is a hardware and software infrastructure that provides dependable, consistent, and pervasive access to resources to enable sharing of computational resources, utility computing, autonomic computing, collaboration among virtual organizations, and distributed data processing, among others" [Gentzsch].

First published in the Proceedings of the FREENIX Track: 2004 USENIX Annual Technical Conference (FREENIX '04)

The original concept of computational grids was an analogy to the electrical power grid. As the power grid provides transparent access to power from various generators run by many different operators, a computational grid provides access to distributed computer resources controlled by multiple organizations. These organizations use grid technologies to form virtual organizations which share resources to solve problems. Examples of this sharing might include computing resources such as clusters or supercomputers, sensors, instruments, or data collections.

While grid computing may have historically started in the high performance computing community, addressing the fundamental issues of *scalable deployment* for distributed information discovery, resource management, workflow management, security, fault tolerance, etc., means that grid technology will actually be applicable in many areas of the computing ecosystem. A recent compilation on grid computing [Berman] provides more information on its origins, current state, applications, and future directions.

The groundswell of interest in grid computing from both academia and industry motivated the creation of the Global Grid Forum (GGF) [GGF], a standards body modeled after the IETF. In February, 2002, work on the Open Grid Services Architecture (OGSA) and the Open Grid Services Infrastructure (OGSI) was announced at GGF. OGSI is a standard based on web services, specifically "a Grid service is a Web service that conforms to a set of conventions (interfaces and behaviors) that define how a client interacts with a Grid service" [Tuecke]. In January,

2004, work on the Web Service Resource Framework (WSRF) was announced at GlobusWorld and subsequently at GGF in March, 2004 [WSRF]. WSRF is intended to complete the convergence of grid and web services started by OGSA, whereby a service client can explicitly specify the state (resource) used by a service on any particular invocation. This essentially allows grid/web services to be considered *stateless*.

In the remainder of this paper we talk about the Globus Toolkit and its status under FreeBSD. First, we give an overview of the Toolkit. We follow this with a discussion of the state of the Toolkit under FreeBSD with detailed coverage of the "impedance mismatches" between the Toolkit and the FreeBSD Ports Collection. Next comes a discussion of ways to integrate FreeBSD and Globus more effectively followed by concluding remarks.

2 The Globus Toolkit

The Globus Toolkit is developed by the Globus Alliance (formerly the Globus Project) and is the de facto standard for grid computing infrastructure in high performance computing. Two versions of the Globus Toolkit are currently available with a third version on the way. The Globus Toolkit 2 (GT2) is an extended version of the original services. It is implemented in C and uses a combination of standard protocols such as FTP and LDAP in conjunction with proprietary protocols. The Globus Toolkit 3 (GT3) is a new implementation of grid services based on OGSI standards. GT3 services are implemented in a combination of C and Java. All GT2 services are included in GT3 as well as new, OGSI-compliant implementations of some services also provided by GT2. The Globus Toolkit 4 (GT4) is to be WSRF-based with a release planned for later this year.

The Globus Toolkit provides some central infrastructure and a set of orthogonal services. The most visible component of the central infrastructure is the Grid Security Infrastructure (GSI), an X.509 certificate based authentication mechanism [Foster2]. GSI provides single sign-on access to resources in multiple independent administrative domains and provides delegation of credentials via proxy certificates. On each host (and potentially on a per-service basis) a mapping is established between distinguished names and local user accounts. GSI is identical in the latest releases of GT2 and GT3 and will presumably be so in GT4.

The Globus Toolkit provides job management services via the Globus Resource Allocation Manager (GRAM) service [GRAM]. GRAM provides an interface between users or meta-schedulers and local resource managers such as Sun Grid Engine (SGE), the Portable Batch System (PBS), or a trivial fork manager. The GT2 and GT3 implementations differ in that GT3 adds a Java-based OGSI-compliant implementation. The GT4 implementations will be WSRF-compliant.

File transfer is provided by the GridFTP service. GridFTP is a set of extensions to the FTP protocol [Allcock]. These extensions include, GSI security on control and data channels, parallel transfers, partial file transfers, third-party transfers, and authenticated data channels. Both versions of the Toolkit use the C implementation.

Resource discovery is handled by the Monitoring and Discovery Service (MDS) [MDS]. The MDS is composed of two parts: (1) the Grid Resource Information Service (GRIS) which provides information about a resource, and (2) the Grid Index Information Service (GIIS) which aggregates data from GRISs to provide support for searching. In GT3, separate GRIS implementations have been removed because all OGSI services can function as their own GRIS. This should be true of the WSRF services as well.

Other services provided by the Toolkit include a Replicate Location Service and, in GT3, OGSI-compliant database services.

GT2 supports most of the major HPC platforms including AIX, HP-UX, Irix, Linux, and Solaris. It is also expected to work on most basically POSIX-like OSes. It consists of over 61 packages, some of which are open source software such as OpenSSL and OpenLDAP and some of which are products of the Globus Alliance and their contributers. The parts of GT3 that are not GT2 components are primarily Java-based Web Services.

To address the complexities of building these multiple packages on different platforms, the Grid Packaging Tools (GPT) were developed [Bletzinger]. Like the FreeBSD ports collection and the Debian package tools, GPT supports patching and building applications from source and creating binary packages for installation on multiple machines.

One unique feature of GPT is the concept of flavors. Flavors encapsulate several coarse-grained compilations options. An example of a flavor is `gcc32dbgpthr` which represents code compiled with GCC for a 32-bit architecture with debugging enabled and POSIX threads used. This is necessary in HPC environments because users often want to pick and choose among these options in order to eke every last bit of performance from their application. As a result, site administrators may find it necessary to install many different variations of parallel libraries such as those included with the Globus Toolkit. GPT facilitates this by naming all programs and libraries according to their flavor. This allows each package to be installed multiple times.

3 The FreeBSD Ports Collection

To understand many of the issues with integrating FreeBSD and Globus, it is necessary to understand the FreeBSD ports collection [FreeBSD1, FreeBSD2]. The ports collection is a collection of the patches and build procedures needed to download, build, and install applications. A dependency system ensures that any necessary dependences are installed before building or installing a given software package. When a port is installed, the list of files associated with it is recorded in a packing list. This list is used to remove the package or to bundle it up into a binary package. Not all ports can be packaged for legal or technical reasons, but most of the more than 10,000 ports have this capability.

Ports can hide much of the complexity involved in installation a large piece of software like the Globus Toolkit. This can significantly simplify the process of installing the software. However, porting large complex systems like the Globus Toolkit can pose significant challenges. Many of these are caused by "impedance mismatches" where the ports collection and the software being ported have different views of the way things should work.

One ports related feature that affects the Globus Toolkit is `BSDPAN`. `BSDPAN` causes standard Perl modules to be integrated with the FreeBSD packaging system. Any Perl module which uses `MakeMaker` is registered as a package. This results in the ability to upgrade modules which are available as ports via the usual methods of upgrading ports. It also allows easy removal of unneeded Perl modules via the package tools.

4 FreeBSD and Globus

FreeBSD is not currently being actively supported by the Globus Alliance, but GT2 builds on FreeBSD 5.x (except on amd64.) The core client software works in limited test. The fork job manager and the GridFTP daemon have been verified to work. The GRIS is at least partially functional. The current lack of information provider scripts for FreeBSD make the output of the GRIS uninformative at this point. This problem should be easily remedied by adding appropriate scripts. A longer lasting solution to this problem might be for the Toolkit to depend on a standard, external data collector such as Ganglia for some tasks. That would avoid the need to write collectors for each OS by pushing the problem off to someone else. We have not yet tested more complex uses of Globus such as SGE transfer queues over Globus [Seed].

We have created a basic port of GT2 and a supporting port of GPT. In creating these ports, we found a number of conflicts with the ports collection. Some of these are fundamental philosophical differences while others are more straightforward conflicts between Globus Toolkit defaults and the expectations of the ports collection. Many of the conflicts are actually with GPT and its assumptions rather then with the Toolkit itself.

The first few issues with GPT have to do with its use of Perl modules. In an attempt to reduce the number of dependencies required to install GPT and to keep a known baseline of module versions, the GPT installation tarball includes all the Perl modules required for operation. From the perspective of keeping the installation instructions short, this works well. Unfortunately, it comes into direct conflict with the FreeBSD package system.

One of the principles of the ports collection is that packages should not install a piece of software that is also available as a separate port. This ensures that maintenance of each piece is centralized and that OS specific patches can all be applied in one place. It also avoids multiple installations of identical files. To mitigate this, the `BSDPAN` module makes any Perl module that is installed the usual way into a package. This is nice for Perl modules that get randomly installed by hand because many of them are already ports so they can be easily upgraded by just updating the ports collection. Unfortunately, this leads to weird effects with GPT. GPT installs

standard modules in a non-standard location. On the next port upgrade cycle where there is a new version of the modules, they will be removed from there and installed in the standard location thus defeating GPT's goal of keeping the versions consistent. This probably does not matter much, but it can cause some confusion with the FreeBSD packaging tools and may result in strange double installations of some modules.

In the FreeBSD port, we have removed the code to install these standard modules and added dependencies on ports for those modules. The current solution involves patching the install scripts. Ideally, GPT should gain the ability to not install modules that already exist either based on a test for their existence or on a command line switch. In addition to installing standard modules, GPT uses a slightly modified version of the `Archive::Tar` Perl module to allow it to produce better error messages in the face of corrupt archives. This causes a number of minor problems. The problems in the previous paragraph apply, but are made worse by the fact that the next upgrade will replace this version of `Archive::Tar` with a standard one losing the modifications.

Assuming the modifications to `Archive::Tar` are necessary for GPT there are three possible solutions to this problem. One is to get the changes merged into the official release so it can be used instead of the GPT version. Given the nature of the changes, this seems unlikely. A second solution is to install the module under another name such as under `Grid::GPT::Tar` where its nonstandard behavior will remain intact and will be harmless. A third solution would be to rewrite GPT to not require these changes. The third solution would be best since that would allow the use of a system version of `Archive::Tar`.

In the port of GPT we simply use the system version of `Archive::Tar` which seems to work fine. In the context of ports, corrupt archives are not a significant issue since both the size and MD5 checksum of the archives are verified before anything is done. When built this way, GPT appears to function correctly.

One final minor issue with GPT and Perl modules is that GPT installs all Perl modules including its internal ones in a non-standard location, overriding any Perl defaults. Specifically, modules are always installed under `$GPT_LOCATION/lib/perl/`. This is fine if `$GPT_LOCATION` isn't the same prefix as the Perl install, but when it is, this is less than ideal as the system can end up with extra parallel Perl module trees. It isn't a serious problem, but it would be nice if GPT could be told to let `MakeMaker`install the modules where it wants to instead of under a hardwired location. We currently ignore this issue in the port.

Another, less fundamental, GPT issue is caused by the use of bundles. Bundles are a set of packages that generally have no external dependencies. They are used because GPT does not provide the sort of automatic dependency installation that the ports collection does. Without this sort of dependency management, installing Globus would almost certainly be too difficult if administrators had to install it from packages as they would have to install as many of 61 packages in a specific order. The problem presented by the bundles is that while they are a reasonable breakdown of the functionality of GT2, they all contain some packages such as `globus_core`. Since each file in the system must be managed by at most one port, the bundles can not be separate ports.

The simplest way to work around this is to use a single port that installs all the bundles. This is the approach we took in the current port. Another approach would be to use the ports collection's dependency management system to install the individual packages as individual ports. This should be a feasible approach. Meta ports could be used to represent the bundles easing upgrading in the face of security advisories and allowing more selective inclusion of Globus tools. One concern with this approach is dealing with packages that install binaries in `bin`, `sbin`, or `libexec` and must be configured with more than one flavor. Some method of insuring that only one flavor installs the primary executable will need to be determined or both flavors will always have to be installed. Adding a feature to the ports system that allowed ports to depend on other ports with specific build options enabled would be very helpful here.

Flavors may be another point of contention. Support for third-party compilers such as Intel's `icc` would be useful, but the ports collection does not handle this sort of thing well. Currently an option could be used to control the base flavor but if there was a desire to install multiple compiler flavors, the only option would be to create slave ports for other compilers leading to an explosion of ports. This solution would be time consuming and unsatisfying. In this

regard GPT is superior to most existing build and packaging systems including the ports collection.

On a more philosophical level the Globus Toolkit's practice of providing its own version of standard libraries such as OpenSSL and SASL is not in keeping with the policy of the ports collection that the ports or base system version of a library be used where ever possible. Using a single version of a library simplifies maintenance of FreeBSD specific patches and reduces the overall size of the system by avoiding duplication. Unfortunately, in the case of the Globus Toolkit, this is likely to be impossible, primarily due to GPT's use of flavors.

5 Future Work

At this point, what Globus needs most on FreeBSD is more user testing and small bits of cleanup. Most of the existing problems are not particularly serious, but may require time consuming work. For example writing GRIS data collectors is mostly simple shell scripting, but the shell scripts have somewhat strange interfaces so there is a learning curve.

In addition to these sorts of cleanup tasks, more work could be done to make FreeBSD an attractive platform for grid computing. There are two direct ways to attack this problem. One is to improve the integration of Globus with the system by making it easier to install and making it easier for base services such as SSH and FTP to take advantage of enhancements such as GSI authentication and the GridFTP extensions. Much of this can be accomplished by adding or improving existing ports. Adding support or improving existing support for GSSAPI in applications in the base system could enable those applications to use GSI authentication [RFC2743].

The second way is to enhance Globus to take advantage of unique operating system features. One idea would be to investigate adding sendfile support to GridFTP. If it were possible to do GridFTP parallel transfers via a zero-copy mechanism, performance could be significantly improved. This might require enhancing sendfile or building a new GridFTP client/server. Another option would be adding support for using Name Service Switch (NSS) to determine the mapping between local user names and x.509 distinguished names for use in GSI authentication. With the current file based approach, maintaining those mappings across a cluster of machines is a potentially time consuming task. Adding network database support via NSS is something FreeBSD (and other OSes with a NetBSD derived NSS implementation) are uniquely capable of doing because NetBSD had the foresight to include external access to NSS databases via the nsdispatch() function. Having a NSS replacement for `/etc/grid-security/grid-mapfile` could significantly decrease administrative overhead in clusters of grid enabled machines.

A third, less direct method of making FreeBSD more attractive to grid users is to work towards solving problems that are high priorities in the grid computing community. For example, many grid projects have to deal with transferring huge amounts of data over long distances. Research into improved TCP congestion control techniques such as HSTCP and XCP (and into methods to ease that research) could pay big dividends in terms of attracting these types of users [Floyd, RFC3649, Falk]. Other areas to look at include trusted computing [Watson] and storage technologies.

6 Conclusions

FreeBSD clients should be able to interact with any GT2-based grid as first class citizens. Most services also seem to work, but further testing is warranted and GRIS support needs some work. GT3 programs are expected to generally work, but have not yet been tested. While there are still significant rough edges, FreeBSD shows promise as a platform for Globus-based grid computing that should continue with GT4.

References

[Allcock] W. Allcock, *GridFTP: Protocol Extensions to FTP for the Grid*, GFD-R.020, GGF. http://www.ggf.org/documents/GWD-R/GFD-R.020.pdf

[Berman] F. Berman and G. Fox and A. Hey (eds.), *Grid Computing: Making the Global Infrastructure a Reality*, Wiley, 2003.

[Bletzinger] M. Bletzinger, *A User Guide to the Grid Packaging Tools*, National Center for Super Computing Applications (NCSA) University of Illinois, 2003. http://www.gridpackagingtools.com/book.html

[DeFanti] T. DeFanti, I. Foster, M. Papka, R. Stevens, and T. Kuhfuss, *Overview of the I-WAY: Wide area visual supercomputing,* International Journal of Supercomputer Applications, 10(2):123– 130, 1996.

[Falk] A. Falk, D. Katabi, "XCP Protocol Specification", unpublished draft, February 12, 2004. http://www.isi.edu/isi-xcp/docs/xcp-spec-04.txt

[Floyd] S. Floyd, S. Ratnasamy, and S. Shenker, *Modifying TCP's Congestion Control for High Speeds*, Rough draft, May 2002. http://www.icir.org/floyd/papers/hstcp.pdf

[FreeBSD1] *The FreeBSD Handbook*, The FreeBSD Documentation Project, 2004. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook

[FreeBSD2] *The FreeBSD Porter's Handbook*, The FreeBSD Documentation Project, 2004. http://www.freebsd.org/doc/en_US.ISO8859-1/books/porters-handbook

[Foster1] I. Foster and C. Kesselman, *Globus: A Metacomputing Infrastructure Toolkit*, Proceedings of the Workshop on Environments and Tools for Parallel Scientific Computing, SIAM, Lyon, France, August 1996.

[Foster2] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, *A Security Architecture for Computational Grids*, Proceedings of the Fifth Conference on Computer and Communications Security, San Francisco, CA, 1998, pp. 83–92.

[Foster3] I. Foster and C. Kesselman (eds.), *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Fransisco, 1999

[Foster4] I.Foster, *What is the Grid? A Three Point Checklist*, Grid Today, 2002. http://www.gridtoday.com/02/0722/100136.html

[Gentzsch] W. Gentzsch, *Response to Ian Foster's "What is the Grid?"*, Grid Today, 2002. http://www.gridtoday.com/02/0805/100191.html

[GGF] http://www.globalgridforum.org

[GRAM] http://www-unix.globus.org/developer/resource-management.html

[MDS] http://www.globus.org/mds/

[RFC2743] J. Linn, *Generic Security Service Application Program Interface, Version 2, Update 1*, RFC2743, IETF, 2000. http://www.ietf.org/rfc/rfc2743.txt

[RFC3649] S. Floyd, *HighSpeed TCP for Large Congestion Windows*, RFC3649, IETF, 2003. http://www.ietf.org/rfc/rfc3649.txt

[Seed] T. Seed, *Transfer-queue Over Globus (TOG): How To*, EPCC, 2003. http://gridengine.sunsource.net/download/TOG/tog-howto.pdf

[SGE] http://gridengine.sunsource.net

[Tuecke] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt, *Open Grid Services Infrastructure (OGSI) Version 1.0*, GFD-R-P.15, GGF. http://www.ggf.org/documents/GWD-R/GFD-R.015.pdf

[Watson] R. Watson, W. Morrison, C. Vance, B. Feldman, *The TrustedBSD MAC Framework: Extensible Kernel Access Control for FreeBSD 5.0*, Proceedings of USENIX '04: FREENIX, USENIX, June, 2003. http://www.trustedbsd.org/trustedbsd-usenix2003freenix.pdf.gz

[WSRF] http://www.globus.org/wsrf