Grid Infrastructure Deployment using SmartFrog Technology

Ritu Sabharwal
Hewlett-Packard (STSD), Bangalore, India
ritu@hp.com

Abstract

The Globus Toolkit is properly configured open source software for setting up grid nodes across multiple heterogeneous platforms. The process of grid enabling a machine is a long one involving Globus Toolkit installation, certificate requests generation, signing certificate requests by CA (Certificate Authority), installation of signed certificates, configuring various grid services and some basic testing to ensure that the setup is correct. Generally, a grid deployment is done across a large number of distributed machines. In such cases, the process of grid enabling becomes tedious and time-consuming more so in a heterogeneous environment. This paper proposes a solution for grid infrastructure deployment across multiple heterogeneous distributed machines in parallel using SmartFrog (Smart Framework for Object Groups) technology. SmartFrog is a framework for configuring and automatically activating distributed applications. SmartFrog helps in abstracting the configuration for grid enabling process and its runtime environment automatically triggers installations across distributed machines. The initial results we achieved on experimental setups are encouraging.

1. Introduction

Grid computing [1], simply stated, is distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple, yet large, and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. There are no dedicated resources as in parallel computing; any resource (anywhere on the global network) available can be used for computation.

The Globus Toolkit [2] provides software tools to build computational grids and grid-based applications. The Globus Toolkit is both an open architecture and open source toolkit.

An ad hoc grid deployment can be done manually by following the Globus Toolkit installation instructions from Globus homepage [3]. After the installation of Globus Toolkit, grid environment must be properly configured and setup. This includes the generation of the certificate requests for the host, LDAP and user, configuration and starting various grid services like gatekeeper, gridFTP, LDAP server and creation of a map file for mapping user identities. Host certificate is required to run the gatekeeper and GridFTP server on the machine. LDAP certificate is required for running the LDAP server, which provides authenticated access to MDS (Monitoring and Discovery Service). Once the configuration and setup is complete, some basic testing is done to ensure all the grid services are working properly. For testing grid configurations on a machine, a user certificate is required. However, as the grid grows, i.e., the number of machines increases, the overall procedure of manual installation and configuration becomes cumbersome. The whole process of grid enabling a machine has to be repeated for all the resources, which requires lot of time and effort. In the same way, upgrading the software will be equally difficult. The problem also exists for applications. As adaptive enterprises evolve and data centers become larger, deploying, configuring and upgrading applications will become more challenging.

This paper proposes an efficient programmatic approach that can be reused with little effort and scales well for solving the problem of multiple grid deployments. The solution is based on the SmartFrog (Smart Framework for object Groups) technology [4]. The next section discusses related work in the area of grid deployments. Section 3 gives a brief description of the SmartFrog framework. The following sections describe the solution overview, system workflow, experimental results and future steps.

2. Related work

There are various ways of reducing the difficulties of manual installations. A simple approach is to use an interactive shell script for installing the Globus Toolkit source bundles. The script guides the user through the grid installation but still requires setting up of grid environment interactively. Going one-step ahead, the

shell script can take care of both installation and grid environment setup. This approach lacks in the fact that the user is required to provide grid configuration data interactively. It also does not scale well because the script has to be run interactively for each node.

HP Labs' GridWeaver project [6, 7] with University of Edinburgh is an application of SmartFrog Framework that applied LCFG [8] and SmartFrog for grid installations. This is suitable for UNIX platforms because LCFG is a configuration tool for UNIX systems. GridWeaver, is an ongoing research work that is not available publicly or commercially for use. There are other sources available like rpm that only do the installations. Some part of ProLiant Essentials [9] also deals with deployment of applications and images. Administrative tools like SD [10] in HP-UX use Product Specification File (PSF) for describing the configuration data. SmartFrog description language is richer with features like data organization as *attribute:value* pairs, inheritance, templates, parameterization, variable linking and component binding.

3. SmartFrog (Smart Framework for Object Groups)

SmartFrog (Smart Framework for Object Groups) is a technology for describing distributed software systems as collections of cooperating components, and then activating and managing them [5].

The SmartFrog Framework consists of a *description language* for describing component collections and component configuration parameters, a *deployment infrastructure* that activates the application description, a *component model* that manages the components to deliver and maintain running systems and a *set of components* that provide various application services.

SmartFrog has wide applicability across domains ranging from utility computing to large-scale system configuration. SmartFrog and its components are implemented in Java™, though SmartFrog components can easily be written to encapsulate software components based on other technologies.

3.1. SmartFrog language

The SmartFrog description language allows one to create a declarative description of the system needed. The description includes things like *which software components are part of the system, their configuration parameters, and how they should connect to other components in the system, and the workflow associated with the lifecycle of the components and the system as a whole.*

The SmartFrog language is a prototype-based language, which supports templates. The prototype approach makes it very easy for system configurations to be specialized for a specific context, without losing the default configuration, or indeed any other changes in the chain of modifications. The template mechanism also supports extension mechanism; in addition, templates allow multiple configuration descriptions to be composed into one.

3.2. Component model

The component model supported by SmartFrog is a simple, extensible set of interfaces providing access to key management actions – such as instance creation, configuration, termination, and so on. SmartFrog considers a whole system to be defined as a collection of applications running over a distributed collection of compute resources. Each application is, in turn, a collection of components defined statically via an application description or generated dynamically at run-time according to the requirements determined at that time.

Using the SmartFrog language, a component description is given to the framework to create and manage a running component associated with that description.

An important aspect of the SmartFrog component model is the lifecycle. The lifecycle is implemented as a simple state machine. The transitions in the state machine are associated with actions implemented (if required) by the components. The transition actions are implemented by the invocation of methods on the component, during which the component may take any appropriate action.

3.3. Deployment infrastructure

The SmartFrog deployment infrastructure is a distributed network of components that interprets system descriptions, realizes the systems' subcomponents in the correct order and binds them together. The deployment infrastructure continues to manage the components while the system is running, and is also responsible for the clean, properly sequenced shutdown of the system. If any component fails, the deployment infrastructure can detect this and can be configured to take restorative action, or to shutdown the system cleanly.

The SmartFrog framework is designed to form the basis for a fully distributed configuration and programming environment. As such, the system must be able to deal with deploying components into many Processes (Java Virtual Machines or JVM) on many different hosts. The deployment infrastructure uses

```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/services/sfinstaller/sfinstaller.sf

sfConfig extends SFInstaller {
        host "host1";
        user "user";
        passwordFile "passwd.txt";
        ftpLocalFiles ["C:\\.release.tar.gz"];
        ftpRemoteFiles ["/root/release.tar.gz"];
        telnetLogfile "SFinstaller.log";
        telnetCommands [ "tar -xvzf release.tar.gz",
            "export SFHOME/root/smartfrog/dist",
            "export PATH=$SFHOME/bin:$PATH",
            "nohup sfDaemon &"
        ];
        emailAttachments ["SFinstaller.log"];
            emailMessage ("SmartFrog installation status
            on node:"  ++ ATTRIB host ++  ":
            Successful.\n Please see attached telnet log
            file for details.");
}
```

Figure 1. SFInstaller component

RMI as communication and advertising mechanism, though it is designed so that it can be used or replaced with other mechanisms.

4. Solution overview

This solution does not require any pre-requisite software to be present on the machines for grid deployments.

SmartFrog uses RMI for communication mechanism, which requires a SmartFrog daemon to be running on each of the machines associated with the grid deployment. This requires that SmartFrog be installed prior to running the distributed application of grid enabling. Hence, the solution is divided into two parts: first part installs SmartFrog and starts the SmartFrog daemon; the second part takes care of Globus Toolkit installation and configurations. Individual SmartFrog components named SFInstaller, GTKInit and GTKPost are written that provide the functionality of SmartFrog installation and daemon starting, Globus Toolkit installation and configuring grid environment on a single remote machine respectively. SmartFrog description language is used for writing these component descriptions. These components are described below.

Typically Grid infrastructure deployments are done for large number of machines. Hence, the approach used for installation and configurations for a single machine has to be extended to scale up to hundreds of machines. By using the inheritance feature of SmartFrog language, an application is written by extending the individual components. The detail implementation of this application is explained in the next section. This application is deployed on a driver machine which is responsible for triggering grid infrastructure deployment.

5. System workflow

This section describes the SmartFrog components and the certificate signing procedure for authentication of various machines. We will then describe about the workflow for extending this solution for multiple machines.

SmartFrog provides inbuilt components for some basic services like ftp, telnet etc. that are used by the components involved. Each component is described below along with the implementation details.

5.1. SFInstaller component

SmartFrog can be easily installed and the SmartFrog daemon can be started on a remote machine using the SFInstaller component. This is the first component to be deployed on the driver machine.

The SFInstaller component uses inbuilt components like scp/ftp, SSH/telnet and mailer. The SFInstaller component first copies the SmartFrog release files from the driver machine to the remote machine using the scp/ftp component. It then uses the SSH/telnet component for logging into the remote machine and installs SmartFrog by extracting the release files and starts the daemon. After the daemon is started properly, it sends an email to the driver giving intimation about successful SmartFrog installation and starting of the daemon. Figure 1 shows the sample configuration file for SFInstaller component using ftp, telnet and mailer component.

5.2. GTKInit and GTKPost components

The GTKInit and GTKPost components use an install script for installing Globus Toolkit and configuring the grid environment on a machine. The install script provides various options to the user.

- *doinit*: This option sets up access to Globus Toolkit source/binary bundles and triggers the installation process in the user specified location. It provides the facility to log the build/install activities. After a successful installation, it issues certificate requests for user(s), host and LDAP on the machine. These requests are then automatically mailed to the Certificate Authority.
- *dopost*: This option is triggered once the signed certificates are available on the machine. It first installs the user(s), host and LDAP certificates. Then it proceeds to make necessary changes in the /etc/services and /etc/xinetd.d/ to configure the

```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/services/gtk/gtkinit.sf

sfConfig extends GTKInit {
        host "host1";
        user "user";
        passwordFile "passwd.txt";
        ftpLocalFiles ["C:\\gridsetup"];
        ftpRemoteFiles "/tmp/gridInstaller"];
        gtkInstallerScript "/tmp/gridInstaller";
        gtkRepositoryHost "test.abc.com";
        defaultEmail "abc@xyz.com;
}
```

```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/services/gtk/gtkpsot.sf

sfConfig extends GTKPost {
        sfProcessHost "host1";
        gtkInstallerScript "/tmp/gridInstaller";
        gtkRepositoryHost "test.abc.com";
        defaultEmail "abc@xyz.com;
}
```

Figure 2. GTKInit & GTKPost components

gatekeeper and gridFTP components. It also starts the LDAP server and configures the MDS component. After this configuration, it updates or creates the local grid-mapfile (/etc/grid-services/grid-mapfile) with the user(s) who have valid certificates. It optionally updates the local grid-mapfile with entries from the global or grid specific mapfile. This enables other grid users to access the local machine's resources. After updating the map file, it tests the functionality of personal gatekeeper, global gatekeeper, gridFTP, anonymous and authenticated LDAP queries. Finally, it produces a test report along with any error information.

The GTKInit and GTKPost components also use the inbuilt SmartFrog components like scp/ftp and RunShell. RunShell component provides the functionality of executing the shell scripts on Linux based systems and batch files on Windows systems. The GTKInit component copies the install script to the remote machine using scp/ftp component and executes it with *doinit* option using RunShell component after giving it execute permissions. The GTKPost component executes the install script with *dopost* option using RunShell component after giving it execute permissions. Figure 2 shows the sample configuration file for these components.

5.3. Certificate signing

The process of certificate signing by CA (Certificate Authority) cannot be automated through a

```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/services/sfinstaller/sfinstaller.sf
#include "org/smartfrog/sfcore/workflow/components.sf"

sfConfig extends Parallel {
   Setup1 extends SFInstaller {
            host "host1";            user "user1";
            passwordFile "passwordfile1.txt";
            ftpLocalFiles ["C:\\.release.tar.gz"];
            ftpRemoteFiles ["/root/release.tar.gz"];
            telnetLogfile "host1.log";
            telnetCommands [ "tar -xvzf release.tar.gz",
                    "export SFHOME/root/smartfrog/dist",
                    "export PATH=$SFHOME/bin:$PATH",
                     "nohup sfDaemon &"];
           emailMessage ("SmartFrog installation on
           node:"  ++ ATTRIB host ++   ": Successful.\n");
   }
   Setup2 extends SFInstaller {
            host "host2";            user "user2";
             …….. // other sfinstaller attributes for host2
   }
     ……..// similar components for other hosts
}
```

```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/services/gtk/gtkinit.sf
#include "org/smartfrog/sfcore/workflow/components.sf"

sfConfig extends Parallel {
   Setup1 extends GTKInit {
            host "host1";            user "user";
            passwordFile "passwd.txt";
            ftpLocalFiles ["C:\\gridsetup"];
            ftpRemoteFiles "/tmp/gridInstaller"];
            gtkInstallerScript "/tmp/gridInstaller";
            gtkRepositoryHost "test.abc.com";
            defaultEmail "abc@xyz.com;
   }
   Setup2 extends GTKInit {
            host "host2";            user "user2";
             …….. // other sfinstaller attributes for host2
   }
   ……..// similar components for other hosts
}
```

```
#include "org/smartfrog/components.sf"
#include "org/smartfrog/services/gtk/gtkpost.sf
#include "org/smartfrog/sfcore/workflow/componentsl.sf"

sfConfig extends Parallel {
   Setup1 extends GTKPost {
            sfProcessHost "host1";
            gtkInstallerScript "/tmp/gridInstaller";
            gtkRepositoryHost "test.abc.com";
            defaultEmail "abc@xyz.com;
   }
   Setup2 extends GTKPost {
            sfProcessHost "host2";
          …….. // other sfinstaller attributes for host2
   }
    ……..// similar components for other hosts
}
```

Figure 3. Parallel deployment application

programmatic approach. Each machine can be configured to send the certificate requests to different CA for signing. Also, different CA has different signing policies for certificate signing which cannot be made universal. So, the GTKPost component is deployed after the certificates are signed by the CA and placed in some location in the machine.

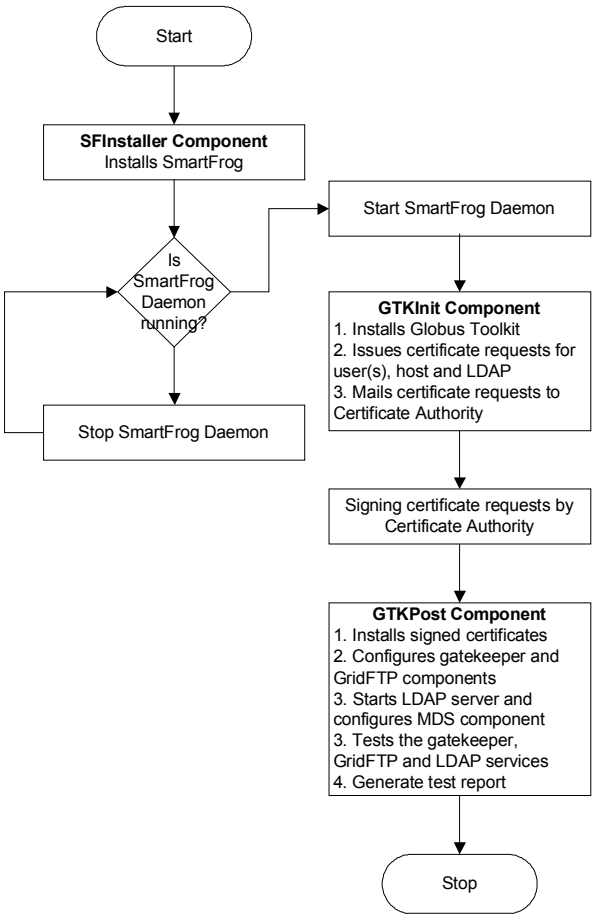

Figure 4. Flow Diagram for multi-machine grid deployments

5.4. Multi-machine deployment

The application uses a complex workflow, in this case *Parallel*, for enabling the deployment and configuration across multiple machines. The components within *Parallel* workflow are deployed independently on each machine without affecting the deployment on some other machine. Hence, separate applications are written using *Parallel* workflow for deploying SFInstaller, GTKInit and GTKPost components on multiple machines. The various attributes in these components can be configured for individual machines. Figure 3 shows the sample code for these applications.

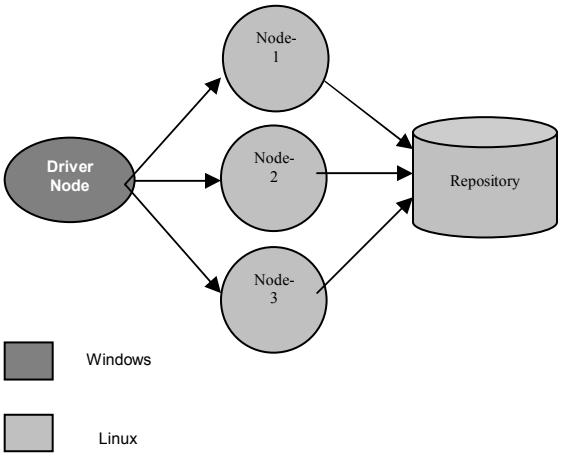

Figure 5. Experimental setup

A driver machine (possibly a desktop) is needed for initiating the whole process. This machine is responsible for triggering grid infrastructure deployment. SmartFrog is installed and the daemon is started on this machine and the SmartFrog applications for SFInstaller, GTKInit and GTKPost are deployed from this machine. The source/binary Globus bundles are kept in a separate repository. Figure 4 shows the flow diagram for this solution.

Advantageously, the SmartFrog components used in this solution are reusable with minimal configuration changes. Upgrading of grid infrastructure does not require re-installation of SmartFrog. Hence, the solution is much simpler even though it is two-stepped procedure.

6. Experimental results

The solution has been tested for grid infrastructure deployment with 5 Linux/x86 and Windows machines. We have used Globus Toolkit 2.4.3 for our experiment. The experimental setup is shown in Figure 5.

The manual installation of Globus Toolkit 2.4.3 and configuration of grid environment on each machine took 2 hours, which equals to 10 hours for the whole grid deployment. Using this solution, the setup was completed in 2 hours which is a speedup of 5 times. This solution is simpler, faster and an automated process that does not require human intervention and is less prone to errors. The resulting description can also be customized to different situations keeping all the knowledge and history of changes. The Quality of sService (QoS) issues are yet not addressed. SmartFrog can be used for other distributed installations like virus

updates, patches etc. This solution is going to be used at Indian Institute of Science (IISc), a research institute in Bangalore, India, for grid deployments across its various departments.

7. Conclusion

In this paper we have presented an automated and faster solution for grid infrastructure deployment across multiple heterogeneous distributed machines to form a computational grid. SmartFrog technology is exploited for this purpose. This solution is scalable and can be used for setting up real world grids.

In future, we will be working to add the following features: distributed logging facility, generating a setup report, better error handling for the install script, improved resource management, proper cleanup of resources in case of failure, management and monitoring of the deployed components using SmartFrog etc. by enhancing the configurations of the existing components. Testing on multiple platforms like HP-UX, Solaris and Windows will also be done.

8. References

[1] Grid Computing. http://www-1.ibm.com/grid

[2] The Globus Toolkit. http://www-unix.globus.org/toolkit/

[3] Globus Toolkit 2.4 Installation Instructions. http://www.globus.org/toolkit/downloads/2.4.3/

[4] SmartFrog. http://www.smartfrog.org , http://sourceforge.net/projects/smartfrog

[5] P. Goldsack, Julio Guijarro, Antonio Lain, Guillaume Mecheneau, Paul Murray and Peter Toft, "SmartFrog: Configuration and Automatic Ignition of Distributed Applications", HP Labs, Bristol, UK, May 29 2003. http://www.hpl.hp.com/research/smartfrog/papers/SmartFrog_Overview_HPOVA03.May.pdf

[6] G. Beckett (EPCC), K. Kavoussanakis (EPCC), G. Mecheneau (HP Labs), Peter Toft (HP Labs), P. Goldsack (HP Labs), P. Anderson (School of Informatics, The University of Edinburgh), J. Paterson (School of Informatics, The University of Edinburgh), C. Edwards (School of Informatics, The University of Edinburgh), "GridWeaver: automatic, adaptive, large-scale fabric configuration for Grid Computing". www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/130.pdf

[7] Peter Toft, "Large-Scale, Adaptive Fabric Configuration for Grid Computing", HP Labs, Bristol. http://www.epcc.ed.ac.uk/gridweaver/docs/GridWeaver-GGF8-PGM-RG-Workshop.pdf

[8] LCFG. http://www.lcfg.org

[9] ProLiant Essentials. http://h18004.www1.hp.com/products/servers/proliantessentials/index.html

[10] SD. http://www.software.hp.com/products/SD_AT_HP/faqs/general.html