Remote Software Toolkit

A GUI for the Grid

Jeff M Larkin, Innovative Computing Laboratory, UT
<larkin@cs.utk.edu>
Eric T Meek, Innovative Computing Laboratory, UT
<meek@cs.utk.edu>

Revision History

Abstract

The concept of distributing software across a heterogeneous set of computers is simple; however, in practice remote installation and administration are daunting tasks. Currently several proprietary solutions exist addressing this problem; however, for general users the requirement of a homogeneous environment quickly diminishes their usefulness. The Remote Software Toolkit (ReST) proposes a suite of tools as a single solution to the problem of software administration across a heterogeneous environment. Consisting of three primary components -- Remote Software Installer (ReSI), Remote Software Explorer (ReSE), and Remote Software Monitor (ReSM) -- ReST provides a familiar graphical interface to basic tasks such as software installation, administration, and monitoring.

Table of Contents

Project Description

When referring to a grid, the national power grid is most commonly cited as an example. It sets the standard for usefulness and reliability among grids. It demonstrates the goals of the grid computing community. It is used as an example, or a "holy grail" because of the it's enormous success. However, mirroring the success of the power grid in the grid computing environment has proved quite a difficult task.

Many of the concepts and techniques used in developing grid computing environments have been taken directly from the concepts and techniques used in developing the power grid. Autonomous usage and "on demand" computa-

tions are two example of the computational grids mirroring concepts first seen in the power grid. However, an arduous, tedious and laborious infrastructure setup is an area where the computational grid should not mirror the power grid.

The excitement grid computing has generated in the scientific computing community is phenomenal. As grid computing has become a widespread area of research the groundwork has been laid the for many new and thought provoking ideas. However, as with most new ideas and technologies, the usefulness of grid computing has been far surpassed by its potential. The idea of using computing resources that are geographically separated and architecturally diverse is simple, but the logistics of using such a system is complex. Resources in such an environment generally fall under different administrative domains and rarely have the luxuries of cluster computers (i.e. shared filesystems, common software base, etc.). These logistical problems limit grid computing to users who are very adept with computers or have a support staff adept with computers. For this reason, among others, software distribution and administration is an arduous task in grid computing environments.

The Remote Software Toolkit (ReST) aims to address these difficulties by providing a familiar, graphical interface to basic tasks, such as software installation, administration, and monitoring. ReST provides an extensible toolkit that can be used and customized by software providers for a wide variety of software packages. The Remote Software Toolkit consists of three primary components: Remote Software Installer (ReSI), Remote Software Explorer (ReSE), and Remote Software Monitor (ReSM).

Remote Software Installer - ReSI

For consumer software installation, Installation "Wizards" have become the de-facto standard; but in a heterogeneous computing environment, such a luxury is almost nonexistent. Software installation in a heterogeneous environment often require a user to login to numerous machines and perform roughly the same tasks on each. Further compounding the problem, most source distributed software installation requires both a lengthy and confusing configuration, compilation, and installation process. The Remote Software Installer (ReSI) provides a familiar, wizard-like interface to source and binary distributed software that can be used to install software in a heterogeneous environment easily and automatically. At the core of the Remote Software Installer (ReSI) is the ReST package model. By defining the structure of the ReSI software packages, the ReST package model provides software maintainers a way to define their software's requirements and options so that they can be presented to a user in a simplified manner, shifting the burden of software installation from the user to the software provider. For more information about ReSI, please read the ReSI section later in this document.

Remote Software Explorer

The difficulties of using a grid environment don't end once the software has been installed. Without careful notes during installation it is easy to forget where software has been installed, the options used to configure the installation and sometimes even which machines are available. These reasons as well as others provided the motivation behind the development of the Remote Software Explorer. The Remote Software Explorer, or ReSE, provides an interface similar to a filesystem or network explorer for managing computers and software packages. ReSE gives users a quick glance of the hardware and software resources available in a friendly and useful interface. Future versions of ReSE will even include "drag-and-drop" features that mirror those available on the user's workstation. For more information about ReSE, please read the ReSE section later in this document.

Remote Software Monitor

Once the software environment has been installed and configured, it is critical for users be able to easily monitor the health of the system. Many software packages aready provide products that allow for such monitoring, but so far no generic toolkit for such service monitoring exists. The ReSM aims to provide a unified interface to these monitors. By providing developers with a flexible toolkit on which to base their monitoring software, users will be able to oversee their grid applications from one common interface. For more information about ReSM, please read the ReSM section later in this document.

The ReST Framework

Communcations

In order to create a viable communcations framework several constraints needed to be met. The viability of the Remote Software Installer (ReSI) totally depends on its ability to communicate in a heterogenious environment. The Grid Software Explorer (ReSE) also requires communications in a heterogenious envirement as well as optional ability to communication with various software packages. Unlike the previous two applications, the Remote Software Monitor (ReSM) currently only communicates with a single sensor monitoring a specific process. Considering these diverse communication needs of the installer, explorer and monitor no single scheme could be chosen.

Augueably the most important decision in building the ReST framework was choosing the Remote Software Installer's built-in communications scheme. Considering the diverse yet strict set of requirements along with the wide variety of choices available to interconnect computers the choice quickly became clear, ssh.

The ssh protocol was an ideal choice as the built-in communications scheme for several reasons. First, among computers likely used as a software grid, the availability of ssh is almost assured. Security concise organizations generally only provide users ssh connections to remote computers. Second, choosing ssh as the communications scheme, allows ReSTs to leverage an existing ssh environment a user might have setup. However, as in most circumstances one size generally doesn't fit all. To accommodate the various communications needs of software providers, a ReST communications plugin has been defined allowing software providers/users to extend ReST to suit their own communications needs.

One important constraint of the Remote Software Explorer is the ability to communicate in an heterogenious environment. By leveraging the work meeting the same constraint in the Remote Software Installer the work in designing the ReSE communications scheme was significantly reduced. Using the same scheme, software independent package control could be done without any further work. However, integrating the Explorer into the software grid requires the development of a ReST Software Plug-in.

Graphical User Interface

Installer, Explorer, Monitor, as the names imply, familiarity is one of the main goals in the design of the GUI for each tool. For modern day graphical user interfaces, software installers, utilities to interact with the computer and system monitors are fundamental elements. Each element serves a necessary purpose in simplifying the computing environment for the average user. If any one element is missing, the complexity of the system exponentially increases.

In the scientific computing and grid software communities the complexity due to the lack of these type of tools is very evident. Access to most scientific and grid software is limited to users with an in-depth knowledge of the operating systems to be used. Installation wizards are a common convenience for most software and are the basis for the design of the Remote Software Installer. Reducing the complexity and depth of knowledge required to install even the most basic software package was a fundamental key. Accomplishing this required a simple yet extensible installer, able to handle the most basic to the most complex software packages with minimal effort on the part of the user. In continuing this thrust, the ReST Explorer aims to provide the user a familiar interface to interact with Remote Software. Based on the common theme used in file system browsers, the ReST Explorer allows the user to display/control elements previously installed with the ReST Installer.

Packages

When designing a new package specification it is critical that the specification offers features that are both not currently available in other package managers and worthwhile enough to convince developers to adopt the packages. Since no package specifications seemed to meet the requirements of ReST and new package specification was designed to allow maximum flexibility to package developers while maintaining ease of use. The ReST package specification gives developers a way to completely describe the installation process for their software and support both source and binary packages. Often times the software user is not fully aware of options available to them during installation or common problems that may occur. The ReST package specification encourages developers, who have an intimate understanding of the software, to package their software so that the user will be able to fully configure and install the software with minimal effort

The most important part of a ReST package is the package.xml file, which describes the software contained in the package, the installation process, options that are available to the user, and what actions the user will be able to perform on the remote server once the package is installed. By encapsulating this information in an XML file, we are able to maximize flexibility while keeping machine-readability. The package.xml file is broken down into 8 parts, some of which are optional: header, 6 "steps" (preparation, configuration, compilation, installation, completion, and uninstallation), and actions. Each of these sections is described below.

Header

As should be expected, the header section contains basic meta-data about the software package. Information such as the software name, version, web page, and license is located in the package header. Additional information about the packager can also be placed in the package, if it is packaged by someone other than the original author. The header may also contain URIs to the actual package sources and patches, which may be contained in the package itself or could also be a URL. Additionally the package could contain configuration file stubs, which are listed in the header, along with information on how to fill-in additional information needed to complete the configuration file. An example of a package header appears below.

Example 1. A ReST Package Header

```
<!-- Basic information about the software package -->
<header>
  <name>NetSolve</name>
  <version>2.0</version>
  <description>NetSolve is a grid middleware package</description>
  <uri>http://icl.cs.utk.edu/netsolve/</uri>

  <!-- Basic information about the packager -->
  <packager>
    <name>Jeff M. Larkin</name>
    <uri>mailto:larkin@cs.utk.edu</uri>
  </packager>

  <!-- Package source(s). We can do both remote and local files  -->
  <packagesrc>NetSolve-2.0.tgz</packagesrc>

  <!-- Configuration files that need editing -->
  <configfile packagefile="server_config"
              remotefile="NetSolve-2.0/server_config"
              description="NetSolve Server Configuration File">
    <sub name="agent" description="The NetSolve Agent hostname"
                 default="netsolve.cs.utk.edu"/>
  </configfile>
</header>
```

The 6 Steps

The process of installing a package is described in five steps, and uninstallation is described in one, optional step. Each of the first five steps are essentially the same, consisting of zero or more commands, each of which consists of zero or more options. The steps simply provide a way of logically grouping the commands. Here is a description of each of the first five steps in the order that they appear in the file (also the order that they will be executed).

1. *Preparation* - Commands used to prepare for future steps, (i.e. decompressing source archives)
2. *Configuration* - Any pre-compilation steps. (i.e running a GNU configure script)
3. *Compilation* - Source packages should be compiled during this step. (i.e. make all)
4. *Installation* - Files are copied to their final location during this step. (i.e. make install)
5. *Completion* - Post-installation clean-up occurs during this final step. (i.e. removing source directories)

Many considerations must be made when installing the software on the remote systems. The most common, and perhaps most difficult, consideration is the existence of shared filesystems and a common architecture, like that found on computing clusters. On such machines certain operations may only need to be performed once and others may need to be performed on all machines. Likewise files may only need to be copied once to be used on all machines.

STUFF ABOUT FILE COPYING AND UNINSTALLATION

Actions

Actions can be thought of as a glue between ReSI and ReSE. Once a package is installed on a group of machines the user will presumable want the ability to use the software, which is where actions are useful. The packager is able to define a series of actions that the user may want to do, such as starting and stopping servers or running tests. The user is presented the opportunity to execute these actions at installation time, or use the explorer to execute an action later on a given location. Actions are defined similarly to commands, but are run after the package has been completely installed. An example actions section appears below.

Example 2. Package Actions

ReST Plugins

Communications Plugin

Software Plugin

ReST Plug-in definitions: Discovery - Software Locations Software services provided Maintenance - Status Usage -

ReST Components

Remote Software Installer

Designing software

Remote Software Explorer

Remote Software Monitor

Future Work

Everything that we haven't done already.

Conclusions

ReST is the coolest project ever!

References

A.

Example A.1. Example Package XML

```
<?xml version="1.0" encoding="UTF-8"?>
<package xmlns="http://icl.cs.utk.edu/ReST/Package"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://icl.cs.utk.edu/ReST/Package
                             http://www.cs.utk.edu/~larkin/icl/ReST/restpackage.xsd">

  <!-- Basic information about the software package -->
  <header>
    <name>NetSolve</name>
    <title>NetSolve Installer</title>
    <version>2.0</version>
    <description>NetSolve is a grid middleware package</description>
    <uri>http://icl.cs.utk.edu/netsolve/</uri>

    <!-- Basic information about the packager -->
    <packager>
      <name>Jeff M. Larkin</name>
      <uri>mailto:larkin@cs.utk.edu</uri>
    </packager>

    <actions>
      <action name="Start Server" tooltip="Start a NetSolve server.">
        <command value="/bin/bash ./start_server.sh" statusmsg="Starting Server"
                 errormsg="Failed to start server."/>
      </action>
      <action name="Kill Server" tooltip="Kill a NetSolve server.">
        <command value="/bin/bash ./kill_server.sh" statusmsg="Killing Server"
                 errormsg="Failed to kill server."/>
      </action>
      <action name="Restart Server" tooltip="Restart a NetSolve server.">
        <command value="/bin/bash ./kill_server.sh" statusmsg="Killing Server"
                 errormsg="Failed to kill server."/>
        <command value="/bin/bash ./start_server.sh" statusmsg="Starting Server"
                 errormsg="Failed to start server."/>
      </action>
      <action name="Start Agent" tooltip="Start a NetSolve Agent.">
        <command value="/bin/bash ./start_agent.sh" statusmsg="Starting Agent"
                 errormsg="Failed to start Agent"/>
      </action>
      <action name="Kill Agent" tooltip="Kill a NetSolve Agent.">
        <command value="/bin/bash ./kill_agent.sh" statusmsg="Killing Agent"
                 errormsg="Failed to kill Agent"/>
      </action>
      <action name="Restart Agent" tooltip="Restart a NetSolve Agent.">
        <command value="/bin/bash ./kill_agent.sh" statusmsg="Killing Agent"
                 errormsg="Failed to kill Agent"/>
        <command value="/bin/bash ./start_agent.sh" statusmsg="Starting Agent"
                 errormsg="Failed to start Agent"/>
      </action>
    </actions>

    <configfile packagefile="server_config"
                remotefile="NetSolve-2.0/server_config"
                description="NetSolve Server Configuration File">
      <sub name="nproc" description="Number of processors"
           default="2" type="string"/>
      <sub name="agent" description="The NetSolve Agent hostname"
           default="netsolve.cs.utk.edu" type="string"/>
      <sub name="scratch" description="Scratch Directory"
           default="/tmp/" type="string"/>
      <sub name="mpihosts" description="Number of MPI Hosts"
           default="4" type="string"/>
      <sub name="workloadmax" description="Maximum allowable workload"
           default="-1" type="string"/>
      <sub name="testing" description="Testing PDF"
           truevalue="" falsevalue="#" type="boolean" default="true"/>
      <sub name="qsort" description="QuickSort PDF"
           truevalue="" falsevalue="#" type="boolean" default="true"/>
      <sub name="area" description="Area PDF"
           truevalue="" falsevalue="#" type="boolean" default="true"/>
      <sub name="mandelbrot" description="Mandelbrot PDF"
```

```
           truevalue="" falsevalue="#" type="boolean" default="true"/>
      <sub name="blas_subset" description="BLAS Subset PDF"
           truevalue="" falsevalue="#" type="boolean" default="true"/>
      <sub name="lapack_subset" description="LAPACK Subset PDF"
           truevalue="" falsevalue="#" type="boolean" default="true"/>
      <sub name="lapack" description="LAPACK PDF"
           truevalue="" falsevalue="#" type="boolean" default="false"/>
      <sub name="lapack_extended" description="LAPACK Extended Drivers PDF"
           truevalue="" falsevalue="#" type="boolean" default="false"/>
      <sub name="scalapack" description="SCALAPACK PDF"
           truevalue="" falsevalue="#" type="boolean" default="false"/>
      <sub name="sparse_iterative_solve" description="Sparse Iterative Solvers PDF"
           truevalue="" falsevalue="#" type="boolean" default="false"/>
      <sub name="sparse_direct_solve" description="Sparse Direct Solvers PDF"
           truevalue="" falsevalue="#" type="boolean" default="false"/>
      <sub name="arpack" description="ARPACK PDF"
           truevalue="" falsevalue="#" type="boolean" default="false"/>
      <sub name="testingglobus" description="Globus Testing PDF"
           truevalue="" falsevalue="#" type="boolean" default="false"/>
      <sub name="restrictions" description="Maximum allowable workload"
           default="" type="text">* 10</sub>
    </configfile>
    <configfile packagefile="MPImachines"
                remotefile="NetSolve-2.0/MPImachines"
                description="NetSolve MPI Hosts File">
      <sub name="hosts" description="List of MPI Hosts" type="text" default="">
enterprise
enterprise
enterprise
enterprise
      </sub>
    </configfile>
    <configfile packagefile="netsolve.env"
                remotefile="netsolve.env"
                description="NetSolve Environment Variables">
      <sub name="agent" description="NetSolve Agent"
           default="netsolve.cs.utk.edu" type="string"/>
    </configfile>
    <!-- Package source(s). We can do both remote and local files  -->
    <packagesrc>NetSolve-2.0.tgz</packagesrc>
    <packagesrc>config.guess</packagesrc>
    <packagesrc>start_server.sh</packagesrc>
    <packagesrc>start_agent.sh</packagesrc>
    <packagesrc>kill_agent.sh</packagesrc>
    <packagesrc>kill_server.sh</packagesrc>

    <installerattributes>
        <backgroundimage>http://www.cs.utk.edu/~meek/icl/GSAP/netsolve_bg.png</backgroundimage>
        <icon>http://icl.cs.utk.edu/favicon.ico</icon>
    </installerattributes>
  </header>

  <!-- Things to do before anything else -->
  <preparation>
    <command value="gunzip -f NetSolve-2.0.tgz" grouped="true"/>
    <command value="tar -xf NetSolve-2.0.tar" grouped="true"/>
    <command value="cd NetSolve-2.0/" grouped="false"/>
  </preparation>

  <!-- Configuration of the package before compilation -->
  <configuration>
    <!-- This is the configure line -->
    <command value="./configure" grouped="true">
      <!-- One of the possible configure options -->
      <option name="lapack" type="text" default="/usr/local/lib/libpack.a"
              truevalue="--with-lapack="/>
      <option name="blas" type="text" default="/usr/local/lib/libblas.a"
              truevalue="--with-blaslib="/>
      <option name="petsc" type="text" default=""
              truevalue="--with-petsc="/>
      <option name="petsclibdir" type="text" default=""
              truevalue="--with-petsclibdir="/>
      <option name="aztec" type="text" default=""
              truevalue="--with-aztec="/>
      <option name="azteclib" type="text" default=""
              truevalue="--with-azteclib="/>
      <option name="superlu" type="text" default=""
              truevalue="--with-superlu="/>
      <option name="superlulib" type="text" default=""
              truevalue="--with-superlulib="/>
```

```
      <option name="ma28" type="boolean" default="false"
              truevalue="--with-ma28"/>
      <option name="itpack" type="boolean" default="false"
              truevalue="--with-itpack"/>
      <option name="arpacklib" type="text" default=""
              truevalue="--with-arpacklib="/>
      <option name="mpi" type="text" default=""
              truevalue="--with-mpi=" falsevalue="--without-mpi"/>
      <option name="scalapack" type="text" default=""
              truevalue="--with-scalapacklib="/>
      <option name="blacslib" type="text" default=""
              truevalue="--with-blacslib="/>
      <option name="mldk" type="text" default=""
              truevalue="--with-mldk="/>
      <option name="rpclib" type="text" default=""
              truevalue="--with-rpclib="/>
      <option name="rpcinc" type="text" default=""
              truevalue="--with-rpcinc="/>
      <option name="octave-include" type="text" default=""
              truevalue="--with-octave-include="/>
      <option name="gpg" type="text" default="/usr/bin/gpg"
              truevalue="--with-gpg=" falsevalue="--without-gpg"/>
      <option name="buildgpg" type="text" default=""
              truevalue="--with-buildgpg="/>
      <option name="nws" type="text" default=""
              truevalue="--with-nws="/>
      <option name="ibp" type="text" default=""
              truevalue="--with-ibp="/>
      <option name="kerberos" type="text" default=""
              truevalue="--with-kerberos"/>
      <option name="proxy" type="choice" choices="nestolve,globus" default=""
              truevalue="--with-proxy "/>
      <option name="ouputlevel" type="choice" choices="debug,view,none" default="none"
              truevalue="--with-outputlevel "/>
      <option name="infoserver" type="text" default=""
              truevalue="--enable-infoserver"/>
    </command>
  </configuration>

  <!-- Source Compilation -->
  <compilation>
    <command value="make" grouped="true">
      <option type="boolean" truevalue="standard" name="Standard" enabled="true"/>
      <option type="boolean" truevalue="all" name="All"/>
      <option type="boolean" truevalue="server" name="Server"/>
      <option type="boolean" truevalue="agent" name="Agent"/>
      <option type="boolean" truevalue="C" name="C"/>
      <option type="boolean" truevalue="Fortran" name="Fortran"/>
      <option type="boolean" truevalue="matlab" name="Matlab"/>
      <option type="boolean" truevalue="octave" name="Octave"/>
      <option type="boolean" truevalue="mathematica" name="Mathematica"/>
      <option type="boolean" truevalue="gridrpc" name="GridRPC"/>
      <option type="boolean" truevalue="pdfgui" name="PDF Gui"/>
      <option type="boolean" truevalue="tools" name="Tools"/>
      <option type="boolean" truevalue="wrappers" name="Wrappers"/>
      <option type="boolean" truevalue="tester"  name="Tester"/>
      <option type="boolean" truevalue="regress" name="Regression Test Suite"/>
      <option type="boolean" truevalue="clean" name="Clean"/>
      <option type="boolean" truevalue="configclean" name="Configclean"/>
      <option type="boolean" truevalue="CLEAN" name="Clean every architecture"/>
    </command>
  </compilation>

  <!-- Package Installation -->
  <installation>
    <!--<command value="make install"/>-->
  </installation>

  <!-- Clean-up what is no longer needed -->
  <completion>
    <command value="cd ../"/>
    <command value="rm -rf NetSolve-2.0.tar" grouped="true"/>
    <command value="rm -rf NetSolve-2.0.tgz" grouped="true"/>
  </completion>

</package>
```

B. Screenshots