Sigiri Scheduler for Large scale resources

User Manual V1.1

September 5, 2011

Copyright 2011 The Trustees of Indiana University
Copyright Licensing

Sigiri is an open source tool licensed under the Apache 2.0 license. A copy of the license is available at http://www.apache.org/licenses/LICENSE-2.0.html. The code is copyrighted and copyright owned by The Trustees of Indiana University. Sigiri is a product of the Data to Insight Center of Pervasive Technology Institute (http://pti.iu.edu) at Indiana University. See http://pti.iu.edu/d2i/sigiri for more information.
# Table of Contents

Introduction ................................................................................................................................. 4

Features ......................................................................................................................................... 4

Sigiri Architecture ......................................................................................................................... 4

Software Dependencies .................................................................................................................. 5

Installation Steps ........................................................................................................................... 5

Setting up the database .................................................................................................................. 5

Setting up Sigiri Web Service ....................................................................................................... 5

Setup Web Service Container with Axis2 ....................................................................................... 6

Building Sigiri Service as Axis2 Service ....................................................................................... 7

Deploying Sigiri Web service ......................................................................................................... 7

Sigiri Daemons supported .............................................................................................................. 7

Grid Resource daemon .................................................................................................................. 7

Building Grid Resource daemon .................................................................................................. 7

Running daemon for PBS .............................................................................................................. 8

Running daemon for LoadLeveler .................................................................................................. 8

Running daemon for SGE .............................................................................................................. 9

Running Grid Resource client ....................................................................................................... 9

Windows Local daemon ............................................................................................................... 10

Azure Cloud Resource daemon .................................................................................................... 11

Azure Daemon .............................................................................................................................. 12

Job Manager Web Service ........................................................................................................... 14
Introduction

Scientists who conduct mid-range computationally heavy modeling and analysis often scramble to find sufficient computational resources to test and run their codes. The science they carry out is not petascale or even terascale science but the computational needs often go beyond what can be satisfied by their university. With the maturation of Grid computing facilities and recent explosion of cloud computing data centers, mid-scale computational science has more options to satisfy computational needs. Sigiri Job Scheduler provides a simple abstraction for interaction with heterogeneous resource managers spanning grid and cloud computing, and features that make the tool useful for the mid-scale physical or natural scientist. A key aspect of the service is its support for multiple standard job specification languages and the ability for the user to directly interact with the service, removing the delay that can come through layers of services.

Features

Sigiri has two main components:

1. Sigiri Web Service:
   Sigiri can be configured as a web service by building it into an .aar file. This front-end service provides a web service single access point for platform independent clients, such as workflow clients, to submit and manage jobs to multiple large-scale systems.

2. Job Submission and Management:
   Sigiri has a Persistent Layer block, the Sigiri database and some number ‘n’ of Resource Manager Daemons. Each managed compute resource has a light-weight daemon which periodically checks the job request queue, translates the job specification to a resource manager specific script, submits the pending jobs and persists the correlation between the resource manager’s job id with internal id. The daemon also handles resource manager specific faults and propagates them to the service to notify the clients.

3. Asynchronous Job Status Notification:
   Sigiri clients can register a call back email address to receive notifications of job progression and status.

Sigiri Architecture
Software Dependencies

1. Apache ANT v1.6 or higher
2. Java Development Kit (JDK) v5 or higher
3. MySQL Database Community Server v5.1

4. Software dependencies Sigiri Web Service
   4.1. Apache Tomcat Server v5.5x or higher
   4.2. Apache Axis2

5. Software dependencies for Daemons:
   5.1. Windows Local Daemon:
       5.1.1. Microsoft Windows SDK and .NET 4 Framework
       5.1.2. MySql connector for .Net
   5.2. Azure Daemon:
       5.2.1. Microsoft Windows SDK and .NET 4 Framework
       5.2.2. MySql connector for .Net
       5.2.3. Windows Azure SDK
       5.2.4. NLog

Installation Steps

Setting up the database
After you have successfully installed and launched a MySQL database instance, please follow these steps to setup the Sigiri database.

1. Log into mySQL as an administrator:

   mysql -u root -p

2. Create a database, preferably named "sigiri"

   CREATE DATABASE sigiri;

3. Create login credentials for sigiri and grant permissions

   GRANT ALL ON sigiri.* TO 'SigiriUser'@'localhost' IDENTIFIED BY 'sigiripwd'

   In the example above, 'SigiriUser' and 'sigiripwd' are just examples. Please use a more secure password.

   If the database is not hosted on the same node as Sigiri Webservice, please specify the host onto which Sigiri Webservice is hosted:

   GRANT ALL ON sigiri.* TO 'SigiriUser'@'dedicated-node.iu.edu' IDENTIFIED BY 'sigiripwd'
Apply the script `dbscript.sql` in the directory `code/modules/common/resources`. For setting up the database run the following command:

```
cd code/modules/common/resources
mysql -u <sigiriuser> -p <sigiri_db_name> < dbscript.sql
```

Now the database is created.

**Setting up Sigiri Web Service**

Sigiri provides a web service interface, to allow interoperating with a wide variety of clients including workflows, gateway environments, script based environments, etc. Sigiri also has the flexibility of extending its Web service to support different job specification languages. Our current implementation understands both JSDL and RSL.

The Sigiri web service depends on a number of properties for correct operation. The distribution package contains sample properties file, which can be found in `code/modules/common/resources/sigiri.properties`. Please use this sample file to configure Sigiri according to the deployment environment. Below is the explanation of some important properties.

1. TempScriptFolder=JobScripts (This is the temporary location where temporary scripts for submitting jobs to LoadLeveler are stored).
2. Database Configuration Properties
   a. DriverName=com.mysql.jdbc.Driver (The database driver being used)
   b. DBUrl=jdbc:mysql://localhost:8080/sigiri (The url for database)
   c. Username=SigiriUser (Database username)
   d. Password=sigiripwd (Database password)
3. Sigiri allows job notification status to be sent to an email Id. The details can be specified as mentioned below:
   a. Gmail-UserName=gmailUser
   b. Gmail-Password=GmailPwd

Detailed explanation on other properties can be found in the sigiri.properties file. Default values can be used for the other properties.

**Setup Web Service Container with Axis2**

Assuming that you have successfully installed and launched an instance of Apache Tomcat server and successfully downloaded a copy of Apache Axis2 WAR distribution, follow these steps to deploy Axis2:

1. Unzip Axis2 WAR distribution package.
2. Copy the WAR file into the `webapps/` directory under tomcat.
3. `cp axis2-1.5.1-war/axis2.war ~/apache-tomcat-5.5/webapps/`

Apache Tomcat server will automatically deploy the WAR file and create a directory named axis2.
Building Sigiri Service as Axis2 Service
To build Sigiri as a Web Service, untar the installation tar file and run the following commands:

```
cd code/modules/webservice
ant jar.server
```

This generates sigiri.aar in modules/target/sigiri.aar.

Deploying Sigiri Web service
In the build.xml file, change the tomcat.home value to the directory where tomcat has been installed.

```
<property name="tomcat.home" value="/path/to/apache/tomcat/directory/>
```

Type the following command to copy the sigiri.aar file to the tomcat folder.

```
ant deploy
```

Now the webservice can be accessed at

```
http://localhost:port/axis2/services/SigiriService?wsdl
```

The default port in the above url is 8080.

Sigiri Daemons supported
1. Grid Resource daemon: This daemon can be used to execute jobs using job managers on Grid resources.
2. Windows Local daemon: This daemon executes jobs on a local windows machine.
3. Azure Cloud Resource daemon: This daemon can be used to execute jobs on Azure cloud resource. This daemon can be run on any Windows machine and does not need to run on the cloud.
4. Amazon Cloud Resource daemon: This daemon can be used execute jobs on Amazon EC2.

Grid Resource daemon

Building Grid Resource daemon
The module named standalone supports daemons for Grid based services.

Following job managers are supported:

1. IBM’s LoadLeveller
2. PBS -Portable Batch System
3. SGE -Sun Grid Engine

To build the code, type the following commands

```
cd code/modules/standalone
ant dist
```
Running daemon for PBS
To execute jobs using PBS job manager, ‘PBS’ needs to be passed as argument to the daemon. The daemon has to be first launched on any node in a cluster, which has PBS installed on it. We provide a script file to run the daemon on Quarry cluster at Indiana University, which has PBS job manager installed in it.

Login to Quarry cluster. Download and untar the Sigiri installation tar file. Build the standalone module as specified in the above subsection.

To start the daemon with Job Manager PBS in the resource Quarry, modify the script run-sigiri-in-quarry.sh in directory code/modules/standalone/bin.

```
cd code/modules/standalone/bin
```

Set SIGIRI_HOME to the absolute path of the standalone/releasedirectory.

Now run the script run-sigiri-in-quarry.sh.

```
/run-sigiri-in-quarry.sh PBS
```

This will start up the daemon.

To test with client code, please refer to subsection.

Running daemon for LoadLeveler
To start the daemon with Job Manager LoadLeveler in the resource BigRed, run the script in standalone/bin/run-sigiri-in-bigred.sh

To execute jobs using LoadLeveler job manager, ‘LL’ needs to be passed as argument to the daemon. The daemon has to be first launched on any node in a cluster, which has LoadLeveler installed on it. We provide a script file to run the daemon on Big Red cluster at Indiana University, which has LoadLeveler job manager installed in it.

Login to Big Red cluster. Download and untar the Sigiri installation tar ball. Build the standalone module as specified in the above subsection.

To start the daemon with Job Manager LoadLeveler in the resource Big Red, modify the script run-sigiri-in-bigred.sh in directory code/modules/standalone/bin.

```
cd code/modules/standalone/bin
```

Set SIGIRI_HOME to the absolute path of the standalone/releasedirectory.

Now run the script run-sigiri-in-bigred.sh.
This will start up the daemon.

To test with client code, please refer to subsection.

Running daemon for SGE
To start the daemon with Job Manager SGE in the resource Pragma cluster, run the script in standalone/bin/run-sigiri-in-pragma.sh

To execute jobs using Sun Grid Engine job manager, ‘SGE’ needs to be passed as argument to the script for running daemon. The daemon has to be first launched on any node in a cluster, which has LoadLeveler installed on it. We provide a script file to run the daemon on Big Red cluster at Indiana University, which has SGE job manager installed in it.

Login to Big Red cluster. Download and untar the Sigiri installation tar ball. Build the standalone module as specified in the above subsection.

To start the daemon with Job Manager LoadLeveler in the resource Big Red, modify the script run-sigiri-in-bigred.sh in directory code/modules/standalone/bin.

```
cd code/modules/standalone/bin
```

Set SIGIRI_HOME to the absolute path of the standalone/release directory.

Now run the script run-sigiri-in-pragma.sh.

```
./run-sigiri-in-pragma.sh SGE
```

This will start up the daemon.

To test with client code, please refer to subsection.

Running Grid Resource client
The client code executes a simple echo command on the Grid resource and saves the standard output to the location on the Grid resource specified as argument. An RSL script is used to submit jobs to the Grid resource.

To run the client code, please refer to the directory code/modules/client. Type the following to build the client code:

```
cd code/modules/sigiriClients
ant jar
```
To run the client for submitting a simple echo job to a Grid Resource, type the following

```
cd code/modules/ sigiriClients/bin
```

Modify the property “SIGIRI_URL” in submitJob.sh script to point to the location where you Sigiri service has been setup.

To test client with echo job on Quarry cluster with PBS job manager, run the script as follows:

```
./submitJob.sh PBS stdoutLocation
```

To test client with echo job on Big Red cluster with LoadLeveler job manager, run the script as follows:

```
./submitJob.sh LL stdoutLocation
```

To test client with echo job on Pragma cluster with SGE job manager, run the script as follows:

```
./submitJob.sh SGE stdoutLocation
```

The sample rsl scripts used to submit jobs can be found under code/modules/standalone/resources. These rsl scripts can be modified to change the job submission details:

- PBS_rsl.xml (rsl script to submit job to PBS)
- LL_rsl.xml (rsl script to submit job to LoadLeveler)
- SGE_rsl.xml (rsl script to submit job to SGE)

Sigiri also allows file movement using these resources. For Filemovement, run script fileMove.sh found in the code/modules/client/bin directory.

```
./fileMove.sh resourceName sourceType-to-local
```

Where resourceName can be Quarry, BigRed or Pragma and sourceType can be web or local.

Please ensure that you have write permissions on the destination location.

**Windows Local daemon**

To build this module, please ensure Windows SDK,.Net Framework 4.0 and MySql connector for .Net are installed as specified in Software Dependencies section.

Specify the database details in sigiri.properties file code/modules/WindowsLocal/sigiri.properties file:

```
DBUrl= host-where-db-is-installed
Username=SigiriUser
Password= sigiripwd
DBName= sigiri
```
Apply the patch sqlError.patch in the directory code/modules/WindowsLocal.

Now to build type command:

```
  cd code/modules/WindowsLocal
  MSBuild WindowsLocal.sln
```

To run the daemon, type the following:

```
  cd code/modules/WindowsLocal/bin/Debug
  WindowsLocal.exe
```

The daemon is running now and can execute commands from Sigiri Web Service on Windows local machine, where the daemon is running.

To test the daemon, client code can be found in code/module/WindowsLocal/WindowsLocalClient. RSL is used to submit the job to Sigiri. The sample RSL file used in this sample can be found in directory code/module/WindowsLocal/WindowsLocalClient/Resources as windowsLocalRsl.xml. This can be modified to change the job submission properties. The client code was built as you built the WindowsLocal.sln solution.

To run the client, type the following command:

```
  cd code/modules/WindowsLocal/WindowsLocalClient/bin/Debug
  WindowsLocalClient.exe SigiriUrl
```

Where SigiriUrl is the Url where Sigiri service has been deployed.

Alternatively, to build and run the code, Visual Studio 2010 (VS 2010) can be used. Once VS 2010 has been installed, open the solution file path/to/Sigiri-code/code/modules/WindowsLocal/WindowsLocal.sln

In VS 2010, right click on the solution and select build. The WindowsLocal project can then be run from VS2010.

**Azure Cloud Resource daemon**

The Sigiri Azure daemon will pick up jobs that are required to be launched in Windows Azure and request for workers to schedule the jobs. Current implementation supports using virtual machine roles within Windows Azure, but the proposed extension will also support worker roles to schedule and execute jobs.

Azure job submission has 2 components (in addition to the Sigiri web service, that we already setup):

1. **Azure daemon**: This can be run on any Windows Desktop machine and is used to manage the lifecycle of Azure VMs.
2. **Job Manager Web service**: This needs to be installed as a web service on the Azure VM image which would be deployed as an Azure VM role, when jobs need to be run. Since Azure VM role does not have a job manager/scheduler that will execute jobs on Azure VMs, we use this web
service to execute jobs on the VMs. The daemon, on receiving a job request from Sigiri web service, contacts this Job Manager web service, which in turn executes the job on the specified Azure VM.

We now explain how to install each of these components:

**Azure Daemon**
The Azure daemon in Sigiri runs only under Windows environment (Windows XP, Windows 7 and Windows Server). Please ensure that Windows SDK, .Net Framework 4.0, MySql Connector for .Net and Azure Cloud SDK are installed as specified in the Software Dependencies section. Also the Sigiri daemon has a client that needs to listen on a port 5566. Please ensure that the Firewall rules for the port are changed to allow communication through the port.

**Azure Account Authentication Details:**

Specify your authentication details for your Azure account in the app.config file in path/to/Sigiri-code/code/modules/AzureService/SigiriAzureStandalone/app.config.

Set the following values:

```xml
<appSettings>
    <add key="SubscriptionId" value="your-azure-subscription-id" />
    <add key="StorageAccountName" value="name-of-your-storage-account" />
    <add key="Key" value="your-storage-account-key"/>
</appSettings>
```

Replace `your-azure-subscription-id` with the subscription Id of your Azure account which can be found on the Management Portal at https://windows.azure.com.

Replace `name-of-your-storage-account` with the Name of a storage account you have created using your Azure account. This storage account will provide access to blob, queue and table storage.

Replace `your-storage-account-key` with the Access key (preferably primary key) associated with the storage account you specified.

**Azure Account Security Details:**

You also need to upload a certificate to be associated with your Azure account. Login to the Management Portal at https://windows.azure.com and add the certificate path/to/Sigiri-code/code/modules/AzureService/SigiriAzureStandalone/certificatesCertTest.pfx to the Management Portal. This is a sample self-signed certificate we have provided.

Alternatively, to add your own certificate (eg: issued by a Certificate Authority) do the following:

Copy your .pfx certificate(with private key) and .cer file to the path: path/to/Sigiri-code/code/modules/AzureService/SigiriAzureStandalone/certificates.

In the file path/to/Sigiri-code/code/modules/AzureService/SigiriAzureStandalone/app.config file, set the value for CertificateName to the name of your certificate file(without extension like .pfx) and CertificatePwd’s value to password of your certificate.
You also need to upload your certificate (.cer file) to the Management Portal in Azure as a Management Certificate.

Sigiri application-mapping:

In the file path/to/Sigiri-code/code/modules/AzureService/SigiriAzureStandalone/map.xml file, we maintain a mapping of applications to VM image names uploaded in Azure. This helps the Sigiri Azure daemon to deploy the respective VM Image which supports (has the application installed) the application you wish to start. The VM Image name corresponding to the “default” is the default VM to be used, in case a mapping is not available for a given application. You can add mappings based on the applications supported by the VM Images you have already uploaded to Azure.

Sigiri Database properties:

Specify in the file path/to/Sigiri-code/code/modules/AzureService/SigiriAzureStandalone/app.config, the Database details used by the Sigiri webservice.

Set the following values:

```xml
<appSettings>
  <add key="CertificateName" value="certTest"/>
  <add key="CertificatePwd" value="D2iazure"/>
</appSettings>
```

Where DbUrl is the host address where the database is hosted. DbUsername and DbPwd are the authentication details. DbName is the name of the database.

To build the Azure daemon code, type the following commands:

```
cd path/to/Msbuild (eg: C:/Windows/Microsoft.NET/Framework64/v4.0.30319)
MSBuild path/to/Sigiri-code/code/modules/AzureService/AzureService.sln
```

Now to run the daemon, type the following commands, (open the command prompt as Administrator): 

```
cd code/modules/AzureService/SigiriAzureStandalone/bin/Release
SigiriAzureStandalone.exe
```

Alternatively, to build and run the code, Visual Studio 2010 (VS 2010) can be used. Once VS 2010 has been installed, open the solution file path/to/Sigiri-code/code/modules/AzureService/AzureService.sln in Administrator mode.
In VS 2010, right click on the solution and select build. The SigiriAzureStandalone project can then be run from VS2010.
**Job Manager Web Service**

The code for the Job Manager Web Service can be found at the directory code/modules/AzureService/SigiriAzureDaemonService.

This code was already built, when you built the entire solution AzureService.sln using the command MSBuild AzureService.sln in the previous subsection.

This code needs to be installed in the VM image, before the VM image is uploaded to Azure. After this service is installed, the VM image is uploaded to Azure and can be deployed on demand-basis by the Sigiri Azure daemon’s life cycle manager as and when it receives requests to execute jobs on Azure VMs.

Please follow the following steps to install the web service:

Before installing the web service on the VM, please ensure that IIS is installed on the VM. Please follow the steps [here](#) to install IIS server.

To install the Job Manager web service, follow the steps below:

Create a directory sigiri in the folder C:/inetpub/wwwroot/

cd code/modules/AzureService/SigiriAzureDaemonService

Copy the following files to C:/inetpub/wwwroot/sigiri/

1. Web.config
2. Service1.svc
3. Bin folder with contents inside

Click Start->Internet Information Services (IIS) Manager.
Right-click on the machine name and select “Add Website”.
Enter a suitable “Site Name”.
Enter the “Physical path” as C:/inetpub/wwwroot/sigiri
Click ok.

When deploying the VM Image as an Azure VM role, the ServiceConfiguration.cscfg file provided has a default values for the VM role’s remote desktop access:

- Username as debuguser
- Password as Debug123

In order to change these values for username and password, you can change the values in path/to/Sigiri-code/code/modules/AzureService/SigiriAzureStandalone/Publish/ServiceConfiguration.cscfg

**Sigiri Azure Client**

To test the daemon, client code can be found in code/module/AzureService/AzureServiceClient. RSL is used to submit the job to Sigiri. This can be modified to change the job submission properties.

To build the client code, type the following commands:

```plaintext
cd path/to/Msbuild(eg: C:/Windows/Microsoft.NET/Framework64/v4.0.30319)
MSBuild path/to/Sigiri-code/code/modules/AzureService/AzureSigiriClient.sln
```
To run the client, type the following command:

```
cd code/modules/AzureService/AzureSigiriClient/bin/Debug
AzureServiceClient.exe SigiriUrl
```

Where SigiriUrl is the URL where Sigiri service has been deployed.

Azure daemon also supports file movement to and from Azure blob store.

Alternatively, to build and run the client code, Visual Studio 2010 (VS 2010) can be used. In VS2010, open the solution file path/to/Sigiri-code/code/modules/AzureService/ AzureSigiriClient.sln
Right-click on the solution and select build. The AzureSigiriClient project can then be run from VS2010.