Inca as Monitoring

Kavin Kumar Palanisamy
Indiana University Bloomington

Abstract

Grids are built with multiple complex and interdependent systems to provide better resources. It is necessary that the service they provided is integrated and consistent. Managing such complex systems is very difficult. Although, there a number of tools to provide system-level information of the usage of the Grid resources we propose a user-level Grid monitoring system.

1. Introduction

Inca is a monitoring system that detects Grid infrastructure problems periodically by performing automated, user-level testing of Grid software and services. It enables consistent testing by a Grid user running under a standard user account using a standard GSI credential with centralized test configuration. It uses a GUI (incat) that will manage the results of the tests.

SDSC and TeraGrid developed Inca in mid 2003 as a user-level resource monitoring system. It has several advantages over other user-level monitoring tools. It provides automation and extensibility. There has been several versions released and the latest being 2.5 released in 2009. As per the report of 2009, 7 grids in the USA, Europe and Australia use Inca 2.5. Prewritten Perl and Python API’s provide reporters, which are executables that measure some aspect of the system and output the result in XML. All the communications are done securely through SSL.

Section 2 will cover the components of Inca and its features. In Section 3 we discuss Inca as monitoring in TeraGrid and in Section 4 we conclude with Personal Discussion.

2. Components

The Inca architecture is very simple. There is the agent, reporter manager, and depot. The agent and reporter manager organize the execution of tests and resources. Depot stores all the results in an archive, which can be queried by a data consumer when a Cyber Infrastructure operator needs results.

Figure 1. Inca architecture

The input to Inca is a set of reporter repositories that consists of reporters and a configuration file. The administrator creates this configuration file using the incat GUI. The reports are generated in XML.

The Inca administrator can use a reporter from the repository or generate his own reporter for testing the performance of the grid. Initially for a resource the administrator creates a configuration deployment file using incat and sends it to the agent. The agent fetches the reporters from repository and creates a reporter manager for each resource. It sends the reporter from the repository and instructions to the reporter managers, which operates over the resource and send the details to the depot. The Cyber Infrastructure Operators query the depot to get the information of their resources that is displayed in a friendly format.

Let's discuss the components that contribute to Inca. All the components are described in Figure 1. The more important of them is the Reporter, Agent and Depot.
**Reporter**

A reporter is an executable program that measures some aspect of the system or installed software. It operates out of Inca. There are more than 150 plus reporters currently in Inca and more than 30 API’s written in both Perl and Python. Each reporter executes these scripts inbuilt with Reporter libraries on every resource using the reporter manager and outputs the result in XML schema.

A very good example of the functioning of the reporter is, it pings the pre-WS global GRAM server using the script Figure 2 and the local machine outputs the XML Figure 3. The script consists of Reporter Libraries, like in the below example, the script consists of Inca:: Reporter::SimpleUnit and Inca::Reporter::GridProxy. Most of the scripts are kept less than 30 lines of code.

```perl
#!/usr/bin/env perl
use Inca::Reporter::SimpleUnit;
my $reporter = new Inca::Reporter::SimpleUnit(
    name => 'grid.globus.gramPing',
    version => 2,
    description => 'Checks gatekeeper is accessible from local machine',
    url => 'http://www.globus.org',
    unit_name => 'gramPing');
$reporter->addDependency('Inca::Reporter::GridProxy');
$reporter->addArg('host', 'gatekeeper host');
$reporter->processArgv(@ARGV);
my $host = $reporter->argValue('host');
my $out = $reporter->loggedCommand('globusrun -a -r $host', 30);
if ($out){
    $reporter->unitFailure('globusrun failed: $!');
} elsif($out ! ~/GRAM Authentication test successful/){
    $reporter->unitFailure('globusrun failed: $out');
} else {
    $reporter->unitSuccess();
}
$reporter->print();
```

**Figure 2. Perl Script**

The library Inca::Reporter::SimpleUnit gives the basic XML body structure. The $out = Slogged command gives the logged command to the reporter to execute at system-level command after logging to the system. The Inca::Reporter::GridProxy adds a proxy dependency to the reporter. The short-proxy is established by Inca for the reporter and is used for its running time.

The reporter has the XML schema as in Figure 3 and it consists of a set of header tags that give the context of the reporter execution, optional tags, log tags, body and exit status tags.

```xml
<?xml version='1.0'? >
<rep:report
xmlns:rep='http://inca.sdsc.edu/dataModel1/report_2.1'>
<gmt>2007-02-22T18:05:37Z</gmt>
<hostname>client64-51.sdsc.edu</hostname>
<name>grid.globusporters.gramPing</name>
<version>2</version>
<workingDir>/Users/ssmallen</workingDir>
<reporterPath>grid.globus.gramPing-2</reporterPath>
<args>
    <arg>
        <name>help</name>
        <value>no</value>
    </arg>
    <arg>
        <name>host</name>
        <Value>localhost</Value>
    </arg>
</args>
<body>
    <unitTest>
        <ID>gramPing</ID>
    </unitTest>
</body>
<exitStatus>
    <completed>true</completed>
</exitStatus>
```

**Figure 3.XML**

**Agent**

The agent is a server written in Java and it implements the configuration specified using Incat. The agent expands the resource group and macro references to determine the series of execution on every resource using resource manager by SSH or Globus. The reporter manager contacts its agent to get the reporter, dependencies and other executables and sends the final report to depot. The agent will transfer any updates done by the Inca administrator to the resources. The agent periodically pings the report manager and maintains a cache. In case of failure the agent looks for the failed system and restarts it.

**Reporter Repository:**
There are currently 150 plus reporters and the repository holds the necessary packages, libraries and catalog files. All its contents are accessible by a URL and are shared with many Inca deployments. The catalog files give the dependencies and commands to be deployed by resources. The packages in the catalog file are versioned and it automatically allows Inca to distribute reporter code updates. The reporters are themselves versioned or they are performance benchmark reporters. The versioned reporters are unit and version test reporters and cover software such as Grid middleware and tools, compilers, data tools, etc., The performance reporters measure data transfer and execute Grid benchmarks.

**Incat:**

The administrators configure user-level monitoring of the Grid resources using the Incat GUI. It is implemented in Java Swing. The Incat looks like the one shown in figure 4. The administrator can give the location of his reporter repositories and incat loads the reporters at that repository. Each reporter has its own properties say, it can specify its own resources to run on, input arguments, email notifications, runtime environment, etc., The set of reports generated in XML when reporters are run over resources is called the report series and these series can be grouped into suites and can be deployed across other Inca deployments. Resources can be grouped into resource groups and input and runtime environment attributes can be attached to these resource macros that can have many values and is executed on the resource once for each macro value.

**Reporter Manager:**

The reporter manager is a lightweight process written in Perl for managing the Inca reporters on a single resource. The reporter manager, running under a regular account, executes reporters on-demand and sends it to the depot. System usage information is sent to the depot with every report.

**Depot:**

The Depot implemented in Java archives the reports of each resource and data is stored in Hibernate that supports many relational databases. The Inca administrators can specify pattern tests to run on reports and get e-mail notifications when the results of these tests change. A Web services is available in addition to predefined queries to allow query access to data.

**Data consumer:**

The Data consumer Figure 5 is a web application that has the jetty server and queries the depot for user-friendly web pages using JSP pages. The XML returned is formatted to HTML by XSL style sheets. The admin can make use of these style sheets or by writing their own JSP tags and pages.

**Secure Communication:**

All the communications are securely done using SSL. In case of requirement of proxy credential, the reporter manager will fetch the short-term proxy from a MyProxy server, then destroys the proxy once the reporter has completed. Generating and using such short-term proxies is more secure and easy than long-term credentials.

**Co-ordinating Inca Deployments:**

Inca supports shared control and storage workload across multiple Inca components. Running two Inca deployments independently across wide resources could result in duplicate testing. Instead we eliminate redundancy by allowing multiple deployments to coordinate suite execution. This co-ordination favors scalability and fault tolerance in Inca system

3. **Inca monitoring on TeraGrid:**

Monitoring in TeraGrid included the validation and verification of Coordinated TeraGrid Software and Services (CTSS) on every Inca resource. Currently, CTSSv3 is being used and it contains approximately 30 software packages facilitating grid tools and services, data management tools and application tools. Inca is employed by TeraGrid along with CTSSv3 to collect job
usage information from Globus 2 GRAM logs and expired host and CA certificates and CA CRLs.

Figure 5. Data Consumer

Currently there are 150 plus reporters executing an average of 109 series on each of TeraGrid’s 18 resources. The Inca configuration of TeraGrid makes a wide usage of macros and resource groups and uses more than 80 series and 82 resource macros to manage a total of 2000 series executing over resources of Inca. Figure 6 shows the components and distribution of Inca in TeraGrid all at SDSC (San Diego Supercomputer Center) and the reporters from the repository can be accessed at the URL inca.sdsc.edu.

The XML pages at the Data consumer have more than six status pages displaying Inca monitoring information and Figure 7 shows one of the six pages, which has a detailed view of CTSSv3 test results page. The packages are in the columns and TeraGrid resources in header rows and their results are shown in the body. A red row defines error or problem. Different graphs and charts giving more insight to every resource can be also obtained to get a better understanding of the problems.

Figure 6.

4. Discussion:

I chose Inca because I have a personal interest in monitoring systems. SDSC and TeraGrid under the lead of Shava Smallen are working hard to make Inca more scalable and fault tolerant in every version they release and Inca is getting more sophisticated. I had a chance to view the videos of the workshops conducted by Shava Smallen and it gave me a thorough knowledge like hands on experience of Inca. I personally think that the scripts in Perl and Python for every reporter can be re-written in Java with more agile functions. The number of reporters...
keeps in reporter repository will keep increasing and more scalability will make Inca one of the best user-level monitoring systems.

References:

[1] Shava Smallen, Kate Ericson, Jim Hayes, and Catherine Olschanowsky, “Article Title”, San Diego Supercomputer Center, 9500 Gilman Drive, La Jolla, CA, pp. 1-10.

