Project 5c : Remote Service Governor

Name	Roll Number	Phone Number	Email address
Peeyush P. Deshpande	MT2011037	9880450584	peeyush.deshpande@iiitb.org
Jatin Chaudhary	MT2011050	8861047701	jatin.chaudhary@iiitb.org
Monika P	MT2011081	8095844904	monika.p@iiitb.org
R.K. Karthik	MT2011114	8123671945	rk.karthik@iiitb.org
Vikas Korjani	MT2011171	8861047512	vikas.korjani@iiitb.org

Team Leader: Peeyush P. Deshpande

Contents

1 Introduction

Remote Service Governor is an application that provides the ability to manage remotely located resources. It is a tool to display statistics that describe the performance of target resources in real time (or near-real time). Such an application can be used to monitor the health of resources, identifying the over-used and under-used resources and controlling the services on these resources.

The remote service governor gets the information about the governed systems, such as number and types of processes running, % CPU used by each process, the time for which they are executing.It is used to provide performance tuning so as to improve services pro-actively, reduce bottlenecks, improve QOS, optimize investments etc. [1]

The application provides better asset/resource utilization by providing information and control over how the resources are actually being used and ensuring their availability. Using the application we can keep track of under/over utilized resources. These abilities improve decision making,balance workloads which is vital for efficiently responding to critical performance and reliability. The application aids in decision execution by providing controls to manage services on the remote resources.

2 Project Description

2.1 Objective

To design and implement a system where a governor process/application manages multiple remotely-located resources or services automatically,in real-time (or near-real time). The governor should follow rules or policies that an administrator can choose and change from time to time. It should be able to receive instructions in the form of e-mail or tweets [2] and send reports to an admin the same way.

2.2 Description

System governance basically involves two categories of tasks:

- System Monitoring
- System Administration

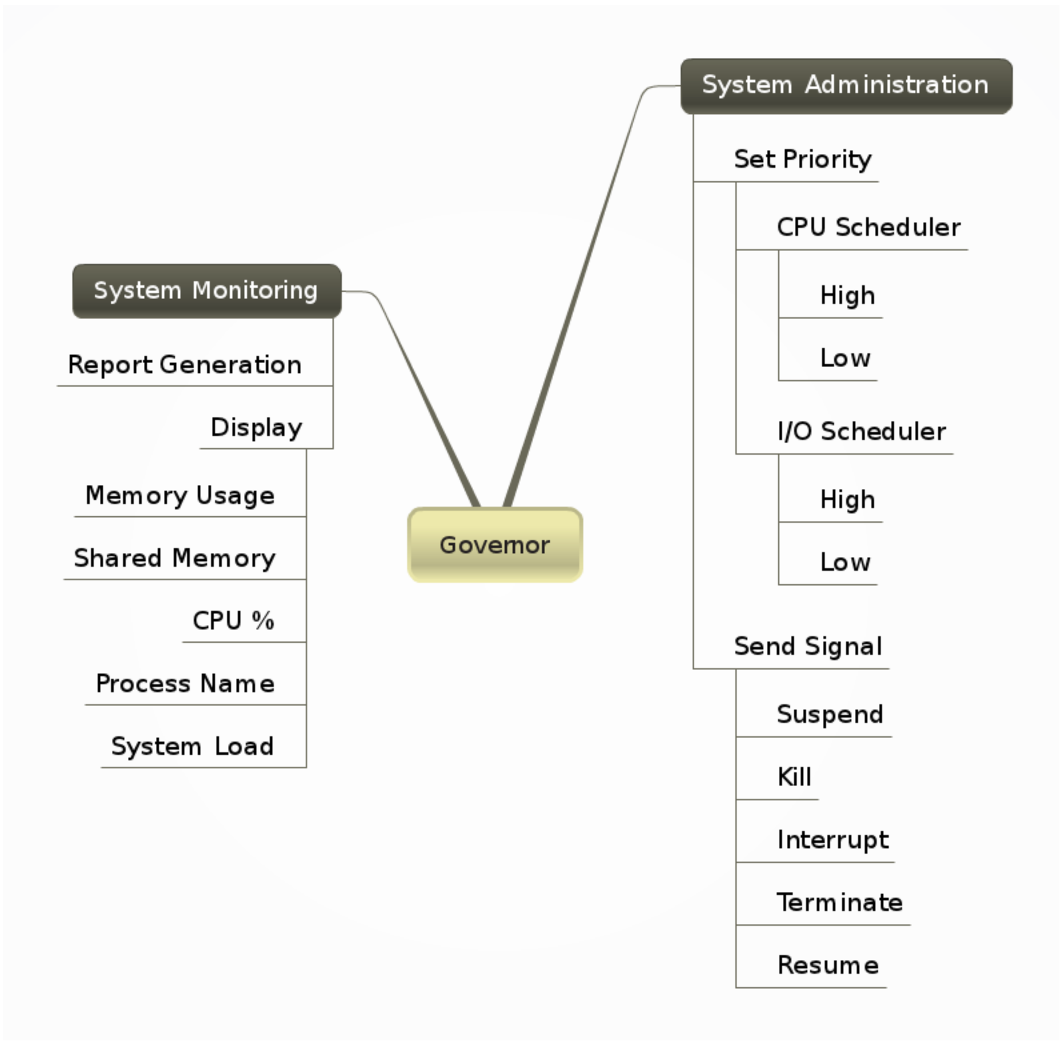

Figure 1: System Monitoring and Administration

As shown in Figure 1, System Monitoring deals with the monitoring of key system parameters like memory usage, shared memory, %CPU utilization, process name, system load etc. Apart from displaying these critical parameters, the generation of reports about these parameters is also an important part of system monitoring. Based on the reports or the analysis of the monitored parameters, the user can take required actions.

System Adminisitration may involve altering the priorities of the processes or sending signals to perform some actions on the processes. These actions may include suspending the process, killing the process, interrupting the process, terminating the process or resuming the process.

The Remote Resource Governor will help the user to perform system monitoring and administration on remote resources. Therefore, the tasks that the governor will perform are also divided into the following two categories:

2.2.1 Remote Resource Monitoring

Remote Resource Monitoring involves remotely monitoring performance parameters such as processor and memory utilization of the resources, listing name and number of processes running on remote system, report generation etc.

2.2.2 Remote Resource Administration

Remote Resource Administration involves basic governance activities like terminate/kill a process, start/stop/resume/prioritize services on remote resources and shutdown/reboot the resource.

3 Gap Analysis

A wide variety of applications which perform remote resource management are present. As of now, these remote resource monitoring/administration applications require the user/admin to operate from a console/system which

is connected to the remote resources present on the network or through a special connection made over the internet. The connection made over the internet is not very secure. Further, it also requires the administrator to log-in to a console/machine which is an accessibility issue. These applications have very low degree of automation and are at times complex as well.
In this project, the limitation of accessibility is overcome by providing control and feedback through e-mails or tweets. The application also handles instructions in the form of e-mails or tweets from the admin/approved user and provides feedback reports in the same manner. [3]

4 Proposed Solution

Remote Service Governor designs and implements a system where a governor process/application manages multiple remotely-located resources or services automatically, in real-time (or near-real time).
As shown in Figure 2, the remote service governor will check the status of processes on connected systems periodically and take decisions about the actions to be performed on them as per the rules set by the administrator.

The system has three components namely,

1. Governor systems

    - Remote process management
    - Policy management
    - Policy based decision control.
    - Report generation

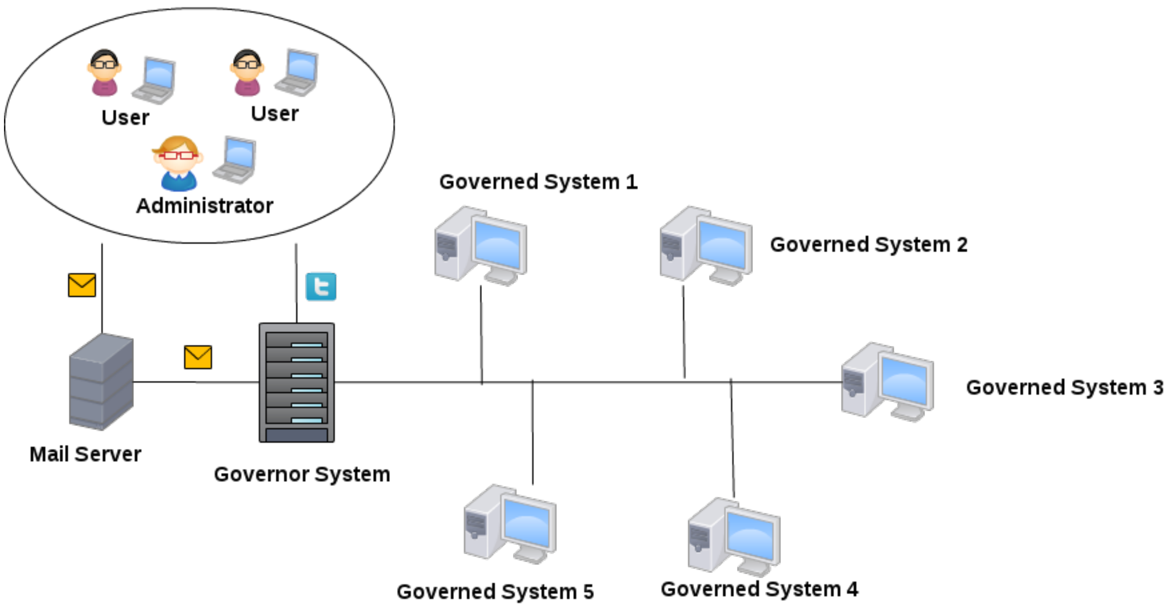

Figure 2: Proposed Solution

2. Governed System

   - Local process management
   - Request handling
   - Task completion acknowledgement

3. Users

   - Request generation
   - Reports management

4.1 System Architecture

The Remote Service Governor is implemented as a client-server architecture where the governor system acts as a client and the governed systems act as servers. The administrator can set the rules/policies to be followed by the governor system. Administration requests can be made through mail/tweets, in turn the governor application can send reports in the same way.

The policies like shutdown/rebooting any resource at predefined time, suspending/killing any process if its % CPU utilization and memory utilization exceeds beyond certain limit etc. can be handled using the application.
The governor system has an application running on it, which communicates with the similar application running on governed systems through Remote Method Invocation (RMI), requesting the statistics of the processes running on the governed system.
The governed systems application in turn sends back the required details. The governor system application analyses the statistics and takes decisions as per the policies set by the administrator. The governor follows rules or policies that an administrator can choose and change from time to time.

4.1.1 Governor System Modules

The Governor System Architecture is as shown in the figure below:

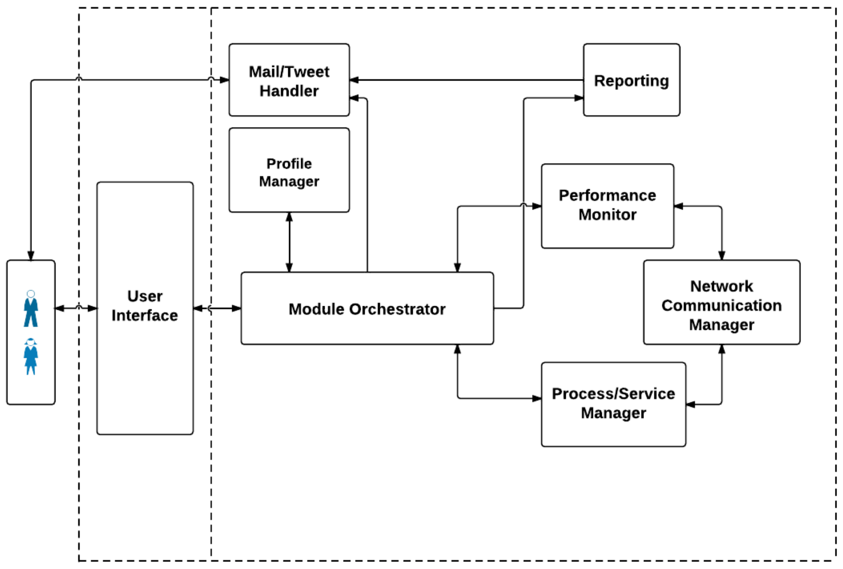

Figure 3: Governor System Architecture

- **User Interface**
  As the application enables users to perform monitoring and administration through various modes like mails, tweets, console, etc, designing GUI for this application is one of the important modules.

- **Profile Manager**
  The governor systems will be used by different kinds of users which may or may not be having access to all the feature of the application. For effectively providing such authorization "User Profile" maintenance is used.
  With the help of this module we will be able to add new users, delete users, define their level of access to the remote resources, create user groups and managing them etc.

- **Network Communication Manager**
  Network Manager module will connect and manage the resources on the network and will control the access mechanism by which users will be able to access these resources and perform operations.

- **Performance Monitor**
  This module enables the user to collect the performance parameters from available remote resources so as to make decisions on the tasks to be performed to improve the performance of overall system.
  Performance parameters like number of processes, amount of memory that is free or in use, %CPU usage of all the processes can be monitored with this module.

- **Process/Service Manager**
  This is the most important module from a remote resources administration point of view.
  This module enables user to gain access over the processes running on remotely located resources. Process Manager sends request to the governed system controller to perform specific actions and the governed system controller in turn calls the appropriate methods in the method container.

- **Mail/Tweet Handler**
  This is one of the new features proposed over existing remote service governors which enables users/resources to receive and respond to commands through mails or tweets.
  Due to this functionality, a user need not be physically present in front of governor system to perform certain operations on remote resources.

- **Reporting**
  This module generates reports that are made available to the administrator/user.

4.1.2 Governed System Modules

The Governed System Architecture is as shown in the figure below:

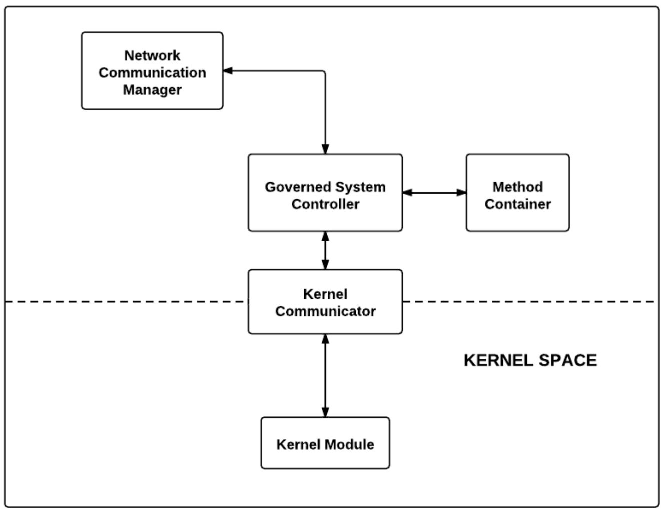

Figure 4: Governed System Architecture

- **Network Manager**
  Network Manager module will enable the communication between the governor system and governed system.

- **Method Container**
  The method container contains all the methods that the governor will invoke for performing monitoring and administration services. This will be useful for performing monitoring and administration of activities which are at application level. The methods can be subdivided into 3 categories:

  1. Monitoring Methods
  2. Administration Methods
  3. Notification Methods

- **Kernel Module**
  The kernel module will handle kernel related work. This will be our key for communicating directly with the OS. This will be useful for performing monitoring and administration of activities which are at system level.

- **Kernel Communicator Module**
  This module will enable communication flow between the kernel module and the Governed System Controller.

- **Governed System Controller**
  This module is responsible for over all integrity of the application. This will handle all the request that the governor application will make. The appropriate methods will be invoked through this module. This module will invoke the methods in the method container and well as the methods in the Kernel module through communication via the Kernel Communicator module. After the completion of the task, it will send back the result/response to the Network Manager so that the same can be communicated back to the Governor App.

4.2 System Requirements

4.2.1 Hardware Requirements

- Minimum 4 laptop/PC with at least 1 GB RAM and 10 GB capacity secondary storage each.
- Ethernet connectivity between machines.

4.2.2 Software Requirements

- Operating Systems : Any Linux operating system for development.
- JDK 6 with Eclipse/Net Beans IDE for development.
- Standard C compiler

5 Development Plan

The development stages are as follows:

- **Stage 1**
  - Understanding the various mechanisms to access processes and perform basic governance activities like kill/terminate processes, start/stop processes, change priority of processes. This would involve understanding the proc file in detail.
  - Understanding how to configure the mail servers to be able to send and receive mails. In particular, we would be looking at Apache James (Java Apache Mail Enterprise Server) and explore its options.
  - Understanding how client-server architecture can be implemented on a local network.
  - Study of socket programming using Java.
  - Study of Linux file system and kernel structures.

- **Stage 2**
  - Establish communication between two peers in the network using socket programming, with one peer acting as a client(governor system) and other as a server (governed system).
  - Creating basic Java applications for governor and governed systems to get information about the processes running on the governed system. This will be done using the Remote Method Invocation (RMI). The governor system will create objects which will be passed over network to the governed systems (server). The object will be used to call and execute the methods on the governed system to get the details of the processes running on it and send back the result.
    In this stage we will explore the use of RPC (Remote Procedure Calls) and its variants like SOAP and JSON RPC. [4]
- **Stage 3**
  - Extend the basic applications developed for governor and governed systems in Stage 2 so as to enable governor system to perform actions like start/stop processes, change priority etc. on the processes in the governed system.
  - Develop a mechanism to set rules/policies on the governor system that will help governor system to take decisions about the actions to be performed on the processes running on the governed systems. The rules/policies will be maintained as files in the governor system. These files will be accessed each time the decision has to made about the processes running on the governed systems.
  - Creating kernel modules to access the system data structures, tables and resources using C language.
  - Automate the process where governor system will query governed system, get the details about the processes running on it, take decisions as per the rules/policies set and perform required action

  on it. The query request/reply and the actions to be taken will be done using RMI.

- **Stage 4**
  - Develop GUI for the basic applications developed in Stage 2 and Stage 3, to enable executions of operations/actions in a user friendly manner.
    The GUI would be developed by using Java Swing components.
- **Stage 5**
  - Develop a mechanism by which administrator will be able to set rules/policies on the governor system by logging into the governor system.
  - Develop and deploy an application on mail server that allows administrator to send mails or tweets to/receive mails or tweets from the governor system. Requests sent using mails/tweets will also be maintained in files in the governed system. Using these files, the governor system should be able to perform required governance activities on the governed system.
  - Implement a mechanism so as to enable governor system to send reports, in the form of mails/tweets, to the administrator about the various actions executed on the governed systems.
- **Stage 6**
  - Integrate the modules as a part of a large system that is capable of taking instructions from the administrator and perform required operations. The system should be capable of getting the statistics of the processes on the governed systems, take decisions by analysing them and send back the report to the administrator.
  - Perform Integration testing on the developed system so as to ensure proper functioning of all the modules.

6 Milestones

Milestone	Task	Due Date
MS1	Initial Project Goals doc draft.	January 17, 2012
MS2	Address review comments and finalize Project Goals doc.	January 27, 2012
MS3	Understanding the various methods to access processes Learning how to set up mail server/twitter REST API. Learn the implementation of socket programming. Corresponds to development plan stage 1.	February 5, 2012
MS4	Establish communication between peers, creating basic Java application to get information about processes running on remote system. Corresponds to development plan stage 2.	February 20, 2012
MS5	Extend application to enable governor system to perform actions on governed system. Develop mechanism to set policies on governor system. Automate the process where governor system requests certain actions to be performed on governed systems. Corresponds to development plan stage 3.	March 15, 2012
MS6	Develop GUI for the basic applications developed in Stage 2 and Stage 3. Corresponds to development plan stage 4.	March 26, 2012
MS7	Develop a mechanism by which administrator will be able to administer the remote resources through mails/tweets and governor system will in turn send back the results. Corresponds to development plan stage 5.	April 5, 2012
MS8	Integrate the modules as a part of a large system and perform Integration testing. Corresponds to development plan stage 6.	April 14, 2012

References

[1] M. Metzker and V. Danciu, “Towards end-to-end management of network qos in virtualized infrastructures,” in *4th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM)*, pp. 21–24, oct. 2010.

[2] “Twitter api,” [Online]. Available: https://dev.twitter.com//.

[3] M. Kalochristianakis and E. Varvarigos, “Open source integrated remote systems and network management with openrsm,” in *4th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM)*, pp. 33–36, oct. 2010.

[4] “Remote procedure calls,” [Online]. Available: http://www.webbasedprogramming.com/Java-Unleashed/ch16.htm.