IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 6, Ver. IV (Nov – Dec. 2014), PP 21-23
www.iosrjournals.org

Automated system for deployment of websites and windows services to the production servers

Drumil V. Deshpande
Computer Science and Engineering, Visvesvaraya National Institute of Technology, Nagpur,India.

Abstract: *This paper is discussed on automated system for deployment of websites and windows services to the production servers. The aim of this paper is to develop and implement an automatic system for deployment of websites and windows services to the production servers.[1]-[2] In addition to that it also foresees the issues regarding re scheduling of errored windows services. The system is implemented using windows sql server as a database, asp.net as the front end technology and c# the used programming language.The paper presents a method to reduce human effort required in rescheduling running windows services on a server and also deployment to the production servers.*

Keywords: *Automated deployment, production servers , windows services , websites.*

I. Introduction

Deployment of websites and windows services is required in the projects that deal with creation or maintenance of windows applications. This process of deployment is often done manually in the IT and computer industry. Typically the applications requiring this procedure have a model[3] in which different websites and services are running on different servers and these websites and or services have to be updated regularly from a central server connected to all the production servers.These type of application typically also store a backup version belonging to each production server on the central server.Also, the applications using this model and running a service on a production server might want to re schedule the service if it errored and terminated due to some reasons.This paper concerns with creation of such an automated system that could handle all these issues that is, it should be able to deploy the websites and services running on respective production servers , take backup of data with respect to each production server and reschedule the terminated services when and where ever required.

II. System Overview

The system is divide into sub-systems which carry out tasks in synchronisation.

2.1 Database: The first sub system deals with storage of the required information. The information stored in the database might vary from application to application but they will generally consist of information such as server_id , website_id , service_id ,server_name , src_location , dest_location , backup_location , server_desc , files_to_exclude. The database is by the system to determine where the data is to be updated , where the backup is to be taken etc. Additional information such as last updated time for each server , server description may also be stored. Also a database need to store information for the services that are running on the servers. This database might contain entries such as service_status , service_id. The database also maintains the network load on the network for each server for optimization of the deployment.

2.2 User Interface and functionalities: The user part of the system contains an application with the required functionalities. The user can choose one or many server(s) to be deployed. The system also provides the user to save and view logs of transactions. Another functionality provided by the system is to re schedule the prematurely terminated services, the user can thus reschedule any number of services that have been terminated abnormally. Finally, the user can change the destination and source location of central, production server as well as the backup location for each server.

III. System Implementation

3.1 Database implementation: The database as mentioned can be stored at the central server or at any other location. The system is implemented by storing data at central server as it reduces the time required to fetch the information such as source and destination addresses from the database server. The database is implemented using Microsoft sql server 2008[4]. A database script is generated to create the tables in database. The database consists of multiple tables to store different information such as information for deployment (viz. source , destination location of all production servers , server ids , website and service ids etc.) , file exclusion(viz. files to be excluded , server id etc.). The system provides a feature to exclude certain files which user would rather want not to be replaced on the production server. An example of such files can be .config files. The database

also stores the load on network for each production server. The issues pertaining when to store then load on network are discussed in next sub-section.

3.2 Implementation of system (user functionalities) : An important issue in deployment of the production server is how frequently should the servers be deployed. The methods available range from using time scheduling to load scheduling. A methodthat would not end up using bandwidth for useless updates nor would increase the load on the network was required. The proposed algorithm used to solve this problem was to keep a check on the dirty flag and the current load on the network as well as the history stored for previous load on network.

3.2.1 Dirty Flag: The dirty flag stored in database indicated whether the production server needed to be deployed or not. A background process (assignFlag) running on the central server was used to check if the data at the source location of each server was modified or not. If the data was modified the dirty flag is set to true indicating that the production server needed to be updated.

3.2.2 Network Load[5]: Another background process (checkForDeployment) running on the central server checked whether a production server needed to be deployed or not. If any server needed to be deployed, the current network load to that production server is checked. Given that the network load on the production server was low, a third background process (autoDeploy) running on the central server deployed that production server. If the network load on the production server that is ready for deployment is not low then the process autoDeploy would wait until the user manually issued a deployment or the load on the network to that production server became low. The time at which the network load might be low can be calculated using the past data.The network load can be stored in the database depending upon the observation made about the traffic on the respective production server. The system then can be scheduled to save the network load depending upon this observation. A counter method to this is to store the network load whenever there is a significant change in the network load. The average network load for an interval can then be stored.

3.3 Overview of Implementation: Following pseudo code represents the overview of the system:

```
assignFlag(); //to check for any server data updated.
queueserver_id readyQ = checkForDeployment(); //to check for any server ready for deployment.
While(readyQ != empty)
{
If(readyQ.top  has low network traffic)
      {
            autoDeploy(readyQ.top);
      }
      readyQ.pop();
}
```

Here, readyQ is a queue containing server_ids of the production servers that are ready for deployment. This represents the automated part of the system that deploys data to the production servers without any human interaction. The user is however also given a functionality to deploy ant server at his will. The serve is deployed if the dirty flag is set to true when user issues a deployment command for that server.

IV. Results and Analysis:

Following are the scenarios that occur:

4.1 Best Case: The best case scenario is the one in which all the production servers are autoDeployed and there is no need of human interaction. The system therefore reduces all of human effort and tie required in this case. Considering thatone human did the work of deployment and it took him about (2 minutes + time taken to update data over network) to locate the source and server for deploying each production server.Also considering that the user required about 30 seconds to reschedule each errored process. For 10,000 production servers and 1,000 errors per day, the system saves about 20,500 minutes of time per day. That is about 14.236 days of time is saved in the best case scenario considering the mentioned scale.(Neglecting the time to reschedule all services using this applications as this time is far less than time to reschedule manually. Also note that the time taken to update the data on serve is independent of this system being used or not).

4.2 Average Case: The average case scenario is when the there are some production servers that the user has to manually deploy. The average case thus would be that N/2 production servers are being auto deployed whereas remaining N/2 servers are deployed manually, where N denotes the number of servers. In this case considering the scale of application mentioned above and considering that it takes a man about 5 minutes to

manually select 5000 servers using the user interface of this application to deploy, the system would save (10000*2) -5+500.That is about 20495 minutes, that is about 14.232 days of time.

4.3 Worst Case: The worst case scenario is the one in which all the production servers have to be deployed by the user that is the auto deploy feature of the system is unable to deploy any server due to load on the network. Again using the above mentioned example the time saved to deploy servers would be 2*10000, as the time required to select all the production servers for deployments using this application is negligible (the user has to just check select all box to select all the production servers for deployment).Hence total time save would be 20500 minutes that is 14.236 days. Note that the time saved by the system in best and worst case is same, and average case is also almost same. However the time required for completion of deployment in worst case would be greater than best case as in worst case the network load is high). As the network load is independent of usage of this system the time saved and human effort saved[6] is almost constant throughout any scenario.

References:

[1]. Automate your deployment process, http://www.rackspace.com/blog/automate-your-deployment-process/
[2]. Jez Humble, Chris Read, Dan North, The Deployment Production Line.
[3]. Architecture model,https://developer.ibm.com/urbancode/products/urbancode-deploy/features/architecture/
[4]. Microsoft Sql server 2008, http://msdn.microsoft.com/en-us/library/ff929050(v=sql.10).aspx
[5]. Common causes of network slowdown, http://www.techrepublic.com/article/common-causes-of-network-slowdowns/
[6]. The human impact of Automation, Harley Shaiken.