SQL SERVER FAILOVER CLUSTERING

KAVURI.ROSHAN[1], SUSMITHA SRIPADA[2], CH.SRINIVASULU[3], NIRAJ UPADHAYAYA[4], A.GOVARDHAN[5].

[1,2,3&4]Department of CSE & IT , JB.Institute of Engineering & Technology, Hyderabad,
roshan.kavuri@gmail.com
[5]Department of Computer Science JNTUH, Kukatpalli, Hyderabad..AP. India.

ABSTRACT

SQL Server provides a solution for mission-critical applications that require the highest levels of availability. SQL Server failover clustering is part of the SQL Server high-availability technology toolset and is designed to help businesses meet their availability and uptime goals. This provides protection against both planned as well as unplanned downtime. When a server on one of the node fails SQL Server can continue serving request through other node(s).

SQL Server includes several changes to the implementation of SQL Server failover clustering, including an entirely new setup process and support for up to 16 nodes. This project discusses several new and existing concepts for SQL Server failover clustering, such as architecture, implementation, administration, and troubleshooting.

Keywords

SQL Server, Clustering, Failover, Mission critical Application, Net work Load Balancing

1. INTRODUCTION

1.1 Introduction to SQL Server Failover Clustering

Organizations rely on mission-critical servers to run their businesses. As a result, server downtime can be very expensive. A heavily used e-mail or database server can easily cost a business thousands or even tens of thousands of dollars in lost productivity or lost business for every hour that it is unavailable. For every benefit and advantage an organization gains by an IT solution, technical and business decision makers should also consider how to manage the inevitable downtime that accompanies these solutions. Hence such a database server is to be used so that data is always in a consistent manner and serves thousands of requests with a much improved performance. Therefore an organization or a business environment is need of such an environment that has a server which serves the requests continuously i.e., high availability is always maintained and the business meet its uptime goals without ending up in loss.

A Windows Server failover cluster aims to provide high availability for services or applications that run within the failover cluster. It contains a group of independent servers that work together to increase the availability of applications and services. Failover clustering can protect against hardware and software failures by failing over resources from one server (or cluster node) to another as required. Failover is the process of taking a clustered service or application offline on one node and bringing it back online on another node. This process is typically transparent to the users, who should experience a minimal disruption of service when a failover occurs.

1.2 Existing System

In the existing system only one database is maintained at a datacenter. Though SQL Servers are used, they are used only with Network load balancing concept. So whenever the server at a datacenter fails, all the traffic is routed to the other datacenter manually by running commands in the foundry. Only after the server is recovered from the failure it is allowed to serve the users.

Disadvantages:

- Though SQL Servers are used with NLB concept
- Clustering environment is not used because of which failure of server had a greater impact in providing high availability.
- All the traffic is routed manually by writing commands.

1.3 Proposed System

In the proposed system SQL Server is used in cluster environment because of which a failure of a server doesn't have any impact because the traffic is automatically routed to the other server which handles the user request. Not only routing the traffic to the other server but also deals with the load balance which in turn give a better performance than considering individual servers.

Advantages:

The advantages of clustering over NLB include the following:

Most important thing to remember is, MSCS maintains the session state but NLB doesn't. For example in case of two node cluster (MSCS with two nodes) if a SQL database is hosted on it and a transaction is going on, there will be no effect on the transaction even if one of the nodes goes down. While the fail over happens the transaction would definitely stop but resume as soon as the other node takes over. Where as in the case of NLB, if a node goes down, any session associated with it would end and the client has to reconnect and establish a new session. For example OWA, if the node to which my OWA connections is established goes down, I need to reconnect as my session would time out after the node serving my session fails.

2. MODULES

2.1. AUTOMATIC INSTALLATION OF SQL SERVER

This module includes the automation of SQL Server installation. Automating SQL server is done using Power Shell scripting language functions. The basic advantage one can have by automating the installation process is admin who has all the privileges to install the SQL server need not spend his time in clicking on the next button to process the installation process. By running a simple .ps1 file he/she can install the SQL Server and if any kind of error occurs he/she can be easily notified by a warning message which appears on the console window. The implementation of SQL Server automatic installation can be done by using the basic Power Shell functions which include

The process of explaining automatic installation is as follows:

Initially use **Get-content** function for calling the .exe file present in some folder and then use **Test-path** function to test whether the .exe file is present in the specified folder if it is present then process continues else by using **write-warning** function it displays a warning message that the .exe file is not present in the specified process and stops the installation.

Other main function is clicking on the button to move to the next page in the wizard and selecting particular sections from options present in the wizard. For this we generally make use WASP (Windows Automation Snap in for Power Shell) is a snap in for windows automation. The different functions that can be used under this sections are as follows:

Select-Window - pick windows by process name or window caption—Used for selecting SQL Server installation center/set up roles etc.

Select-Control - pick controls (children) of a specific window, by class and/or name and/or index i.e. selecting okay, next buttons

Send-Click - send mouse clicks (any button, with any modifier keys) i.e., assigning action to these buttons

Start-Sleep Cmdlet (which is probably more useful in scripts than it is as a command-line command) enables you to pause Windows Power Shell activity for a specified period of time. i.e., while setting up support roles it takes certain time to complete process so in that situation we can specify the time so for that amount of time it waits to perform that specified activity and by combining this with write-warning if any error occurs we will be notified with that error/warnings can be neglected.

In certain step we have to specify the path of shared feature directory by using

Set-Location cmdlet is roughly equivalent to the cd command found in Cmd.exe: it enables you to specify a new working location in a namespace.
For selecting a particular section in the wizard like say installation in step 2 we have to use 2 functions one is

Set-Window Position

(set any one of (or all of) top, **left**, width, height on a window ... or maximize/minimize/restore) and pipeline that result to **select-Window** (pick windows by **process name(i.e. installation)** or window caption).

2.2 REPORT GENERATION

The steps that are to be followed to generate reports are as follows:
The process is done in two stages

<u>STAGE1:</u>

- Write a power shell script and run it in.
- Schedule the process. Initially write a Power Shell script that connects to the database, fetches the required data from a table and pass the IP address to the API and get the required fields as output which is not only printed in the excel sheet as output but also

updates the data back into the database. This process is automated by creating a schedule for this process.

- Schedule the process. Initially write a Power Shell script that connects to the database, fetches the required data from a table and pass the IP address to the API and get the required fields as output which is not only printed in the excel sheet as output but also updates the data back into the database. This process is automated by creating a schedule for this process.

The process runs as follows

- A script is written in Power Shell.
- A script is written in Power Shell.

- This power shell script fetches the required fields---session id, IP address and the name of the application a user has launched into a notepad.
- The IP address is passed to the API which in turn returns the location i.e., the country and the state from where the user has launched the application.
- The output is printed in to an excel sheet.

STAGE2:

The second stage is to schedule the process so that the process automatically runs at a particular time, which not only generates the output but also the additional data which is generated is updated back into the database. The scheduling of a process is done as follows.

- Click on the Control panel-->Selected Tasks-->Add selected task
- Click on the Ne w option and select the Scheduled Task.
- All the scheduled tasks along with the new task appear in a window. Specify the name of the task.
- In order to run the task, specify the path where the script is saved.
  Now schedule the process by specifying the start time, how much amount of time the
- The process automatically runs and if any kind of deadlock situation occurs and if the process is in running state then by specifying the time limit the process automatically stops at the amount of time(by default it is- 72 hrs).

2. 3. SUMARY;

Therefore in the above explained way, a process can be scheduled so that it automatically runs at the specified amount of time which not only generates the output but also database is updated automatically. The advantage one can have is it reduces manual work which in turn reduces human burden of doing the same work repeatedly daily

3. PRILIMINARIES

Before choosing SQL Server certain queries are to be considered. These queries include

What is SQL Server
Top reasons for choosing SQL Server
How much memory and disk space
How much will a server cost

SQL Server can be defined as--Microsoft SQL Server is a relational database server, developed by Microsoft: it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet).

SQL server is chosen because Microsoft SQL Server is a comprehensive database server and information platform offering a complete set of enterprise-ready technologies and tools that help people derive the most value from information at the lowest total-cost-of-ownership. Enjoy high levels of performance, availability, and security; employ more productive management and development tools; and deliver pervasive insight with self-service business intelligence (BI).

A complete and integrated platform, Microsoft SQL Server brings it all together to get more value out of existing IT skills and assets, increase the productivity and agility of IT departments, and quickly build flexible, innovative application. The top reasons for choosing SQL Server include:

- Most Secure
- TCA
- Six nines availability.

CLUSTERING

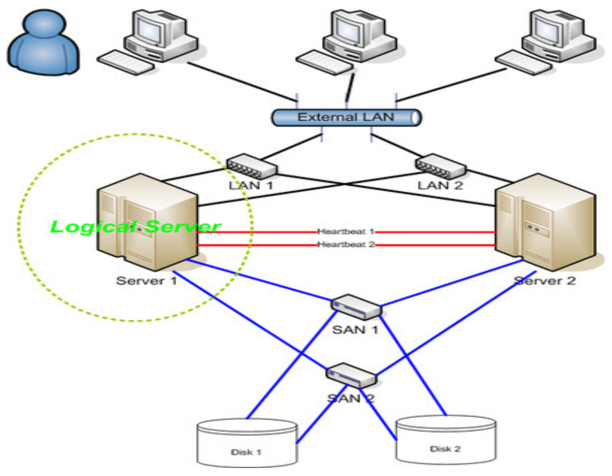
These days, high availability is in demand. No company would afford to lose a customer when the services they offer to be unreachable, especially since that downtime will almost undoubtedly cost them money. High Availability services offered in most load sharing and balancing software and clustering on NT or Linux clusters offers the solution to this need for a lack of downtime.

A cluster is two or more interconnected servers (sometimes called nodes) that create a solution to provide higher availability, higher scalability or both. The advantage of clustering servers for high availability is seen if one node fails, another node in the cluster can assume the workload of the failed node, and users see no interruption of access. The advantages of clustering servers for scalability include increased application performance and a greater number of users that can be supported. One can think of a cluster of servers as if they were a single, unified computing resource. With the total redundancy of multiple servers, the cluster can help achieve greater system uptime.

Clustering can be implemented at different levels of the system, including hardware, operating systems, middleware, systems management and applications. The more layers that incorporate clustering technology, the more reliable, scalable and manageable the cluster. Cluster operating systems divide the tasks amongst the available servers. Clusters of systems or workstations, on the other hand, connect a group of systems together to jointly share a critically demanding computational task. Theoretically, a cluster operating system should provide seamless optimization in every case.

3. SYSTEM ARCHITECTURE

The users connect to the servers via external LAN to the server. The servers share the SAN (Storage Area Network) so that the data of one server is available to the other server. The Heartbeat is responsible for checking the status of other servers. So whenever there is failure with a particular server the HB notifies to the corresponding server and if it has the required data then it processes the request of the user and this is left unknown to the user.



4. IMPLEMENTATION

4.1. Introduction to Power Shell

The technology that is used in automating the SQL server installation process or report generation module is Power Shell scripting language. Windows Power Shell is Microsoft's task automation framework, consisting of a command-line shell and associated scripting language built on top of, and integrated with the .NET Framework. Power Shell provides full access to COM and WMI, enabling administrators to perform administrative tasks on both local and remote Windows systems.

In Power Shell, administrative tasks are generally performed by Cmdlets (pronounced command-lets) specialized .NET classes implementing a particular operation. Sets of Cmdlets may be combined together in scripts, executables (which are standalone applications), or by instantiating regular .NET classes (or WMI/COM Objects). These work by accessing data in different data stores, like the file system or registry, which are made available to the Power Shell runtime via Windows Power Shell providers.

There are four kinds of commands Windows Power Shell can execute:

Cmdlets, which are .NET programs designed to interact with Power Shell
Power Shell scripts (files suffixed by .ps1)
Power Shell functions
Standalone executable programs

Cmdlets:

Cmdlets are specialized commands in the Power Shell environment that implement specific functions. These are the native commands in the Power Shell stack. Cmdlets follow a <verb>-<noun> naming pattern, such as Get-Child Item, helping to make them self-descriptive. Cmdlets output their results as objects, or collections thereof (including arrays), and can optionally receive input in that form, making them suitable for use as recipients in a pipeline. But, whereas Power Shell allows arrays and other collections of objects to be written to the pipeline, Cmdlets always process objects individually. For collections of objects, Power Shell invokes the Cmdlets on each object in the collection, in sequence.

Cmdlets are specialized .NET classes, which the Power Shell runtime instantiates and invokes when they are run. Cmdlets derive either from Cmdlet or from PS Cmdlet, the latter being used when the cmdlet needs to interact with the Power Shell runtime. These base classes specify certain methods - Begin Processing (), Process Record () and End Processing () - which the Cmdlets implementation overrides to provide the functionality. Whenever a cmdlet is run, these methods are invoked by Power Shell in sequence, with Process Record () being called if it receives pipeline input. If a collection of objects is piped, the method is invoked for each object in the collection. The class implementing the Cmdlet must have one .NET attribute – Cmdlet Attribute - which specifies the verb and the noun that make up the name of the cmdlet.

4.2 Pipeline:

Power Shell, like Unix/Linux based shells, implements a pipeline. This pipeline enables the output of one cmdlet to be piped as input to another cmdlet.

Power Shell differs from Unix/Linux in that structured .NET objects are passed between stages in the pipeline instead of typically unstructured text. Using objects eliminates the need to explicitly parse text output to extract data.

Scripting:

Windows Power Shell includes a dynamically typed scripting language which can implement complex operations using Cmdlets imperatively. The scripting language supports variables, functions, branching (if-then-else), loops (while, do, for, and for each), structured error/exception handling and closures/lambda expressions, as well as integration with .NET. Variables in Power Shell scripts have names that start with $; they can be assigned any value, including the output of Cmdlets. Strings can be enclosed either in single quotes or in double quotes: when using double quotes, variables will be expanded even if they are inside the quotation marks. According to the variable syntax, if the path to a file is enclosed in braces preceded by a dollar sign (as in ${C:\foo.txt}), it refers to the contents of the file. If it is used as an L-value, anything assigned to it will be written to the file. When used as an R-value, it will be read from the file. If an object is assigned, it is serialized before storing it.

4.3. Example Commands:

Get-Content: Get the content of the item at the specified location. Get-Content reads the Content one line at a time and returns an object for each line.

- Syntax: Get-Content [[-path] | -literal Path] string[]
  [-Read Count Int64] [-Total Count Int64]
  [-include string []] [-exclude string[]] [-filter string]
  [-force] [-credential PS Credential]
  [-Encoding] [-Delimiter String] [-Wait]
  [-Use Transaction] [Common Parameters]

- Start-Sleep: Suspend shell, script, or run space activity for a specified period of time.
  Syntax: Start-Sleep [-seconds] int [Common Parameters]

- Get-Child Item: Get the items and child items in a folder or registry key.
  Syntax: Get-Child Item [[-path] string[] | [-literal Path] string[]]
  [[-filter] string] [-include string[]] [-exclude string []] [-name]
  [-recourse] [-force] [-Use Transaction] [Common Parameters]

4.4. WASP:

WASP is a Power Shell snap in for Windows Automation tasks like selecting windows and controls and sending mouse and keyboard events. We have automation Cmdlets like Select-Window, Select-Control, Send-Keys, Send Click, Get-Window Position, Set-Window Position,

Set-Window Active, Remove-Window...etc. Our goal is to enable you to accomplish most Windows GUI Automation scripting from inside Power Shell, without resorting to specialized (and expensive) scripting tools. Just to be clear, don't expect any "click to record" functionality but do expect to be able to automatically tile windows, send mouse clicks and keystrokes, and in general, automate those tasks that you would normally not be able to do from a console.

5. CONCLUSION

The proposed failover clustering technique is quite useful setup in the business environment because in an organization they have to the business environment because in an organization they have to serve many users with 24/7 availability. In such case if there is any failure then the server cannot process the requests of users And also cannot meet its uptime goals and therefore the Organization ends up at loss. Hence failover clustering setup can help the organization by providing solution to mission-critical applications. Therefore failover clustering can help an organization not only meets its uptime goals but also the end user can access the applications without any problem.

6. FUTURE SCOPE OF THE PROJECT

In future this project is very useful in every business environment to provide high availability to the users. The entire users request can be handled and availability can be provided to them and performance can be maintained by tuning the database and by Preview tool which helps in improving the performance of CPU..

7. REFERENCES

[1] Windows Power Shell in Action, Second Edition-Bruce Payett

[2] Windows Power Shell in Action, Second Edition-Bruce Payett

[3] Learn Windows Power Shell in a Month of Lunches-Don Jones

[4] Power Shell in Depth-Don Jones, Richard Siddaway, Jeffery Hicks

[5] Windows Power Shell for Developers-Douglas Finke

[6] Power Shell and WMI-Richard Siddaway

[7] Windows Power shell Scripting And Tool making-Don Jone

[8] Windows Power Shell 2.0 Bible-Thomas Lee, Karl Mitschke, Mark E. Schill, Tome [10]Tanasovski]

[9] Automating Microsoft Windows Server 2008 R2 with Windows Power Shell 2.0-Matthews Sarah Dutkiewicz

[10] Windows Power Shell 2.0 Best Practices-Ed Wilson

[11] http://powershellcommunity.org/Forums/tabid/54/aft/7969/Default.aspx

[12] http://technet.microsoft.com/en-us/library/ee177032.aspx

[13] http://wasp.codeplex.com/wikipage?title=automation%20cmdlets&referringTitle=Home

[14] http://wasp.codeplex.com/

About the Author:

Mr. Roshan Kavuri has obtained his B.E Degree from Andhra University during 1988-92, and M.Tech (CSE) from JNT University, Kukatpally, Hyderabad. He is having nearly 20 years experience in industry as well as a faculty of Computer Science and Information Technology departments. He is pursuing his PhD from JNTU Hyderabad. His area of research includes Computer Architecture, Parallel Computing, Operating Systems and Computer Networks. Presently he is working as Associate Professor in JB Institute of Engineering Technology, Hyderabad since 2004.

.