Server Consolidation with VCS 2.0
Service Group Workload Management

Table of Contents

Executive Summary

A recent trend in the IT industry has been one of "Server Consolidation". As the use of open systems has grown wildly, the complexity of managing hundreds or thousands of servers has become an impossible task. To compound matters, there is a demand for increased availability of the applications running on the servers. IT managers wish to move from large numbers of small open systems, many running at far less than capacity, to a much smaller number of large scale enterprise serve running at near max capacity (80% or better).

One possible solution is N+1 clustering, where one enterprise class server can provide redundancy for multiple active servers. This partially solves the problem, as it reduces the cost of redundancy for a given set of applications. N+1 is also simplifies failover location choices as well, as all applications running on a failed server simply move to the spare server.

N+1 clustering starts to fall short in true Server Consolidation environments. Customers require the ability to withstand multiple cascading failures, or take systems offline for maintenance and still have adequate redundancy in the server cluster. Typical application clustering packages have fallen short in this area, as the amount of flexibility is limited when it comes to choosing the proper hosts for potentially tens or hundreds of application groups.

This is a perfect place for true N-to-N clustering. N-to-N refers to multiple Service Groups running on multiple servers, with each Service Group capable of being failed over to different servers in the cluster. For example, imagine a 4-node cluster, with each node supporting 3 critical database instances. On failure of any node, each of the three instances is started on a different node, ensuring on node does not get overloaded. This is a logical evolution of N + 1, where there is not a need for a "standby system" but rather "standby capacity" in the cluster.

What is required to truly utilize the capabilities of N-to-N clustering is an advanced ability to proactively determine the absolute best node to run an application at time of failure and scalability to handle this decision for unlimited groups simultaneously.

VERITAS Cluster Server 2.0 provides a unique new feature called Service Group Workload Management to address these issues

Service Group Workload Management (SGWM) Overview

VCS 2.0 Service Group Workload Management (SGWM) is an advanced capability in VCS to proactively determine the best possible system to host an application during startup or following an application or server fault. SGWM provides necessary tools to make intelligent decisions on startup or failover location based on system capacity and finite resource availability.

SGWM Concepts

Service Group Workload Management is enabled when Service Group `AutoStartPolicy` and `FailOverPolicy` are set to "`Load`"
Load policy is made of two primary components, System `Capacity` and Service Group `Load`, and System `Limits` and Group `Prerequisites`.

System Capacity and Service Group Load

System `Capacity` sets a fixed load handling capacity to servers and a fixed demand (`Load`) for service groups. For example, imagine a 4-node cluster consisting of two 16-processor servers and two 8-processor servers. The administrator sets a `Capacity` on the 16-CPU to 200 and the 8-CPU to 100. Each Service Group running on a system has a predefined `Load` value. When a group comes online, its `Load` is subtracted from the `Capacity` of the system. The cluster engine keeps track of the `AvailableCapacity` of all systems in the cluster. `AvailableCapacity` is determined by subtracting `Load` of all groups online (a group is considered online if online or partially online) on a system from the system `Capacity`. When a failover must occur, the cluster engine determines the system with the highest `AvailableCapacity` and starts the group on that system. During a failover scenario involving multiple groups, failover decisions are made serially to facilitate the proper load based choice, however ServiceGroup online operations immediately follow in parallel.
System `Capacity` is a soft restriction. This means that the value can go below zero. During a cascading failure scenario, `AvailableCapacity` can be negative.

Static Load vs. Dynamic Load

VCS 2.0 also provides the capability for the user to run an external package to determine the actual load on a server and feed that load value to the cluster engine for dissemination to all servers in the cluster. In this way, the customer is free to implement custom or 3rd party applications to determine server load based on whatever criteria necessary. The external package can then inform VCS of load on a per server basis through a simple command line interface. When DynamicLoad is specified, VCS ignores the value of group Load.

To calculate AvailableCapacity of a given server, VCS uses the following computation:
`AvailableCapacity` of a system = `Capacity` – Current System `Load`

> Current System `Load` = Dynamic system load if dynamic system load is specified, (`DynamicLoad` > 0)
> OR
> Current System `Load` = Sum of `Load` of all groups online on that system

Limits and Prerequisites

System Limits and Service Group Prerequisites add additional capability to the load policy. The user can set a list of finite resources available on a server (`Limits`), such as shared memory segments, semaphores and others. Each Service group is then assigned a set of `Prerequisites`. For example, a database may need 5 shared memory segments and 20 semaphores. VCS load policy will first determine a subset of all systems that meet these criteria and then choose the lowest loaded system from this set. In this way, an unloaded system that does not meet all the `Prerequisites` of a group will not be chosen. As soon as the decision is made to online a group on a particular system, the `Prerequisites` of the group is subtracted from the `Limits` of the system.

Capacity and Limits Together

Capacity and limits combined make a very powerful tool for determining proper failover node. The system with all proper prerequisites and with the highest available capacity is always chosen. System `Limits` are a hard value. This means a server will not be chosen if it does not meet the `Prerequisites` of the group. `Capacity` is a soft limit. This means the system with the highest `AvailableCapacity` will be chosen, even if this will result in a negative `AvailableCapacity`. This means the cluster will never attempt to bring a service group online where it cannot possibly run due to physical resource constraints, but it will always do its best to keep applications online even at a lower than desired performance.

Overload Warning

Overload warning provides the notification and custom handling part of the load policy. When a server sustains a pre-determined load level set by `LoadWarningLevel` for a predetermined time, set by `LoadTimeThreshold`, the `loadwarning` trigger is initiated. The loadwarning trigger invokes a user defined script or application designed to carry out the proper actions. VERITAS provides sample scripts that detail simple operator warning on overload as well as a method to move or shutdown groups based on user defined priority values. For example, if load on a server running a business critical database reaches and stays above a user defined threshold, operators will be immediately notified. The loadwarning trigger could then scan the system for any service groups with a lower priority than the database (such as an internal HR app) and move the app to a lesser-loaded system or even shut the app down. The key here is the framework is completely flexible. The installer or user is free to implement any overload management scheme desired.

SystemZones

`SystemZones` provide a sub set of systems to use in an initial failover decision. A ServiceGroup will try to stay within its zone before choosing a host in another zone. For example, imagine a typical 3-tier application infrastructure with web servers, application servers and database servers. The application and database servers are configured in a single cluster, but segregated into zones. Configuring `SystemZones` means a ServiceGroup in the application zone will try to fail to another application zone server if

it is available. If not, it would then fail to the database zone based on load and limits. In this configuration, excess capacity and limits available on the database backend would essentially be kept in reserve for the larger load of a database failover, while application servers would handle the load of any groups in the application zone. During a cascading failure, excess capacity in the cluster is still available to any ServiceGroup. The `SystemZones` feature allows fine tuning application failover decisions, yet still retains the flexibility to fail anywhere in the cluster if necessary.

Load Based AutoStart

VCS 2.0 provides a new method to determine where a group should come up when the cluster initially starts. Administrators can allow the VCS engine to determine the best system to start on out of a group of systems. As with failover, a subset of systems is first created that meet all Prerequisites, then of those systems, the system with the highest AvailableCapacity is chosen.
Using load based `AutoStart` and `SystemZones` together allows the administrator to establish a list of preferred systems in a cluster to initially run a group. As mentioned above, in a 3-tier architecture, the administrator would want application groups to start first in the application zone and database groups to start in the database zone.

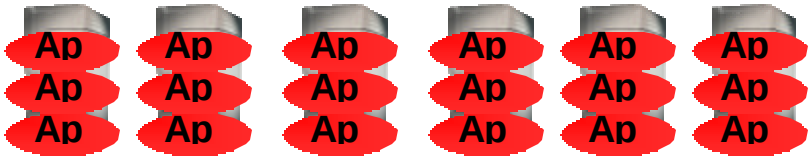
SGWM Example

The following example will show a detail a complex 9-node cluster running multiple applications and several large databases. The servers in the database zone are all large enterprise class systems. The systems are named LgSvr1, LgSvr2 and LgSvr3. The middle tier servers are less powerful systems, each running multiple applications. The middle tier servers are named MedSvr1, MedSvr2, MedSvr3, MedSvr4, MedSvr5 and MedSvr6

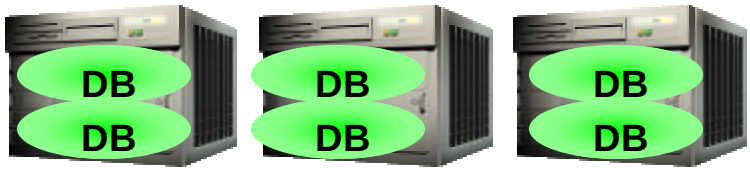
Under normal circumstances, each database server runs two critical databases. The systems are configured with adequate resources to host a maximum of 3 databases at any one time. The operator sets the Limits for these servers as Limits = {Database=3}. Each database service group then sets Prerequisites = {Database=1}. This is an example of the flexibility of the Limits and Prerequisites framework. The operator ensures there are adequate shared memory and semaphores as well as any other variables need for 3 databases to operate, then simply sets database=3.

The application servers each run three applications accessing the backend data at a time. The applications do not use any fixed resources needed by a database, however the operator still configures enough semaphores and shared memory to support one database per application server. This allows maximum cluster flexibility. During normal operation, each application server is loaded to approximately 70-80%. The following figure shows normal cluster configuration.

- **Web front end**
- **Not part of cluster**
- **IP Load balanced**



- **Application Zone**
- **Capacity 100**
- **Application load 70-80**
- **Limit = 1 database**



- **Database zone**
- **Capacity 300**
- **Load 200**
- **Limit 3 databases**

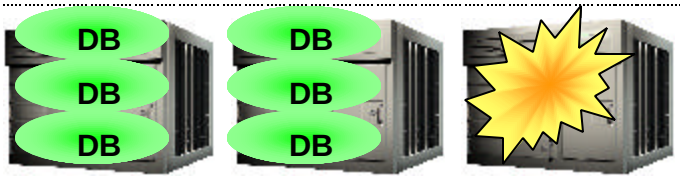
Failure Scenario

The configuration above is an ideal example of FailOverPolicy=Load and SystemZones. The database zone is capable of handling up to two database failures or one full server failure. Each server has adequate Limits to support up to three Database Service groups (with an expected performance drop with all groups running on one server). Similarly, the Application zone has excess capacity built into each machine.

One other fact to note: The Application group machines all specify Limits to support one database, even though the Application groups do not run any Prerequisites. This will allow a database to fail across SystemZones if absolutely necessary and run on the least loaded application zone machine.

For the first failure example, assume system LgSvr3 fails. The cluster engine will first scan all available systems in the zone for systems meeting proper prerequisites. In this case, LgSvr1 and LgSvr2 meet all necessary `Limits`. This will result in the following configuration for the database zone:

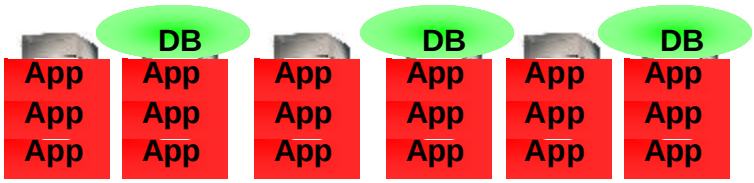
- **Application Zone**
- **Capacity 100**
- **Application load 70-80**
- **Limit = 1 database**



- **Database zone**
- **Servers at max**

Cascading Failure Scenario

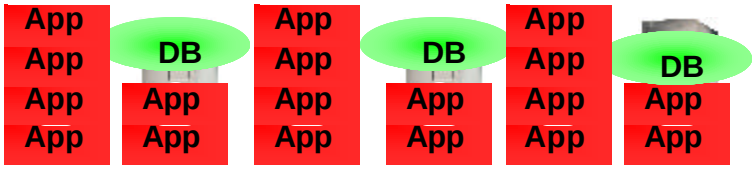
In the next failure, a second database server will fail. Since the limits of LgSvr1 are exhausted, the databases from LgSvr2 will be sent to the least loaded servers in the application zone. At this time, MedSvr2, MedSvr4 and MedSvr6 are exceeding load warning level and the loadwarning trigger will be fired.



- **Application Zone**
- **Capacity 100**
- **3 servers in overload**
- **Loadwarning trigger fired**

- **2nd database server fails**

In this case, the operator, or a comprehensive trigger can decide to reorganize application loads to better accommodate the database load until the original servers are repaired.



- **Application Zone**
- **Operator, or trigger reorganizes applications**

Summary

As shown here, Service Group Workload Management is an extremely powerful tool used to construct intelligent handling of workload in a large availability cluster. VCS 2.0 and SGWM far exceed the capabilities of not only previous versions of Cluster Server, but significantly exceed anything else available in the industry. VCS 2.0 provides the tools to safely integrate large numbers of disparate applications in one cost effective availability cluster.