Hyper-V VM Fleet for S2D

Dan Lovinger
Windows Server Foundation
4/14/2016 – TP5 v 0.4

1 Introduction

This document describes a mechanism for running a distributed set of VMs – the 'fleet' – to provide functional or performance stress against a Storage Spaces Direct deployment. This work originated from the Windows Server 2016 TP2-aligned Ignite Conference demo from Jose Barreto and Claus Joergensen, then extended for the Intel Developer Forum 2015 demo[1] presented in mid-August 2015.

The VM Fleet is specifically adapted to performance analysis work, and allows an analyst to inject near real-time changes to the load as simply as editing and saving a script. This same mechanism can be used to create a simple demonstration environment that loops through a set of scripts.

This work is a prototype in progress, and presumes a hyper-converged deployment where the fleet VMs run on the same hardware as the Storage Spaces Direct cluster. There are certain assumptions that will require adaptation to run in generalized environments, which will be highlighted throughout the document.

Note: the 0.3 version, aligned to Windows Server 2016 TP5, has a few incremental updates:

- test-clusterhealth adds SMB disconnect event aggregation, an improved crashdump sorting pattern, , SMB CSV instance multichannel checks, Mellanox RoCE disable event checks, StorPort unresponsive device checks, and now parallelizes all checks for much faster response time.
- group-scoped start/stop-vmfleet (see new -group <list>)
- create-vmfleet sets default cluster owner node for vms
- create-vmfleet supports taking a dynamic vhdx as base and converting it to fixed in-place at destination (see -fixedvhd)
- run.ps1 updated to unify the human output/xml capture cases

0.4 added more features:

- bugfix: master.ps1 now takes connection credentials as parameters; this allows master to be edited on the fly, it is no longer templated
- credential templating moves to launch.ps1, which is now generated into the VMs
- pause handling now uses an epoch ask/response from the VMs
- check-pause uses the epoch to directly test whether a given VM has indeed acknowledged the current pause request, and is definitive with respect to all running VMs
- new automated sweep mechanic (see Section 4.1)

VM Fleet is now part of DISKSPD. See Section 6.

[1] See Claus Joergensen's blog for a description of the demo, here: http://blogs.technet.com/b/clausjor/archive/2015/08/18/microsoft-and-intel-showcase-storage-spaces-direct-with-nvm-express-at-idf-15.aspx

2 VM Fleet

The current scripting contains a few assumptions based on the environment it was developed in.

- that the VMs do not need external network connectivity
- that the central control point is within CSV
- location of central control point within CSV

The basic design is to have create a fleet of VMs which autologin and launch a master control script which connects back to a known location in CSV, courtesy of a loopback through an internal vswitch to their host. This script then launches the most-current load run script present, and monitors for updates and/or fleet pause requests.

SCRIPT	NOTES
LAUNCH-TEMPLATE.PS1	Per VM autologin script template: launches master.ps1, below, in a loop. ***Contains plaintext credentials when injected into the VMs.***
MASTER.PS1	Master control script for the VM. Copies in a toolset, runs load, monitors for master control and load run script updates, watches for the pause and sweep epochs.
CHECK-PAUSE.PS1	From the control console, checks how many pause acknowledgements have been received/host node.
CLEAR-PAUSE.PS1	From the control console, clears a pause flag.
SET-PAUSE.PS1	From the control console, sets the pause flag.
CREATE-VMFLEET.PS1	Creates the per-node internal VM switches and deploys the VM Fleet VMs from a pre-created VHD master image.
SET-VMFLEET.PS1	Adjusts the number of VPs and memory size/type per VM.
DESTROY-VMFLEET.PS1	Removes all vmfleet VM content.
CHECK-VMFLEET.PS1	From the control console, checks the operational state of VMs hosted throughout the cluster.
START-VMFLEET.PS1	From the control console, launches all VMs currently in **OFF** state.
STOP-VMFLEET.PS1	From the control console, shuts down all VMs currently not in **OFF** state.
RUN.PS1	A standalone load run script. This specific form is simply an example, and can be anything.
RUN-100R.PS1 **RUN-9010.PS1** **RUN-7030.PS1**	These are example scripts used to set up performance demonstration environments (IDF'15, Gartner Datacenter Infrastructure Conference). The demo.ps1 script, described below, causes the VM fleet to alternate between these.
RUN-SWEEPTEMPLATE.PS1	Template file for automated sweeps.
START-SWEEP.PS1	Control script for automated sweeps. See Section 4.1.
WATCH-CLUSTER.PS1	This is an example of text-console performance monitor tracking across a cluster. It displays the CSVFS IOP, bandwidth and latency counters, and aggregates them per-node and whole-cluster.
UPDATE-CSV.PS1	This script is used to manage the tenant CSV volumes per a naming convention, described in Section 3.2 below.
DEMO.PS1	An example script to run a looped demo load with Storage Quality of Service. Run alongside *Watch-Cluster*. This assumes a specific set of QoS policies created ahead of time: SilverVM, GoldVM and PlatinumVM. See Section 5 for example definitions.
SET-STORAGEQOS.PS1	A wrapper for Set-VMHardDiskDrive, which takes a predefined Storage QoS Policy and applies it to all VMs within the hyperconverged cluster.

2.1 Master Control

To see the master control, load runner and pause in action, connect to one of the VMs. The color splash should help make the running operation self-describing. If an issue occurs, simply ^C back to the powershell prompt (note -noexit in the launch parameters) and debug/restart the launch script, or simply shut down and reboot the VM.

3 Create The Fleet VM Image

The VM fleet has been built focusing on the Server Core image for the guest VMs. The only requirement is that the administrator password has been set.

To construct the image, create a VM on a Server Core VHDX or install a Server Core VM using ISO media. Once installed, launch the VM and follow the prompts to specify the administrator password. This password will be specified later to the create-vmfleet script which deploys the VMs. Then tear down the VM – the resulting VHDX will be used as the base for deployment in Section 3.3.

Using a fixed VHDX for the VM is recommended to eliminate warmup-like interactions with the filesystem on dynamic VHDX extension. The Convert-VHD cmdlet can be used to do this if starting from a dynamic VHDX. if there is a specific measurement or deployment goal, though, it is reasonable to deploy with dynamic/unseasoned content and drive through the warmup effects that that mode of operation implies.

It should be possible to use full SKUs. The most immediate change is that they will need to use shell startup items to run the launch script that is injected by the specializer. This mechanism is not used on Core since it lacks the shell.

Support and/or documentation for Nano is possible in a future update.

3.1 Create the CSV Structure and Fleet VMs

The VM Fleet assumes a specific directory structure in CSV for control information & collecting results.

DIRECTORY	CONTENT
C:\CLUSTERSTORAGE\COLLECT	--
C:\CLUSTERSTORAGE\COLLECT\CONTROL	VM Fleet Scripting
C:\CLUSTERSTORAGE\COLLECT\CONTROL\TOOLS	Content to be copied into each VM for load execution (e.g., DISKSPD, others)

The VM Fleet also assumes that there is a set of one or more CSV created per cluster node for its VMs, with virtual disks (and as a result, CSV's) named following the pattern of **<node name>[-suffix]**. These CSVs are mounted in C:\ClusterStorage per the friendly name. This convention simplifies a few tasks:

- moving CSVs within the cluster and back to their nominal owner node
- for a given host, finding its nominally owned CSV

This has proven effective in eliminating the need for additional configuration documentation, such as an XML description of the mappings.

VMs are named following a similar pattern: **vm-[-group/virtual disk suffix]-<nodename>-<number>**

Lastly, a directory is assumed inside of the VMs themselves.

- C:\run

3.2 Create the CSV

The following fragment is an example of creating CSVs following the node naming convention, assuming that the S2D storage pool is named 's2d'. Any appropriate CSV filesystem type, size or resiliency can be used.

```
Get-ClusterNode |% { New-Volume -StoragePoolFriendlyName s2d -FriendlyName $_ -
FileSystem CSVFS_ReFS -Size 1TB }
New-Volume -StoragePoolFriendlyName s2d -FriendlyName collect -FileSystem
CSVFS_ReFS -Size 1TB
```

The update-csv.ps1 script complete the deployment by adjusting the CSV mountpoints, the renamecsvmounts option below. It also has options to verify that integrity is disabled in the CSVs (if desired) and also handles the basic pre-test operation of redistributing or moving the CSVs back to their home nodes.

- -renamecsvmounts: change the csv mount points (C:\ClusterStorage*XXXX*) to match the CSV virtual disk name.
- -disableintegrity: recursively disable integrity streams for all files/directories in all CSVs. Not needed with default options in Windows Server 2016 TP4, but can continue to be used for confirmation.
- -movecsv: (default: always) moves all CSVs back to their home node.
- -shiftcsv: shift all ***current*** CSV node ownerships one node over (lexical by node name). Use this to create cases where redirected CSV access is occurring.

Note that shiftcsv is with respect to the current state of the cluster.

Use update-csv at this time to rename the mounts and move the CSVs into place.

```
update-csv -renamecsvmounts:$true
```

When complete, copy the VM fleet scripting into C:\ClusterStorage\collect\control.

3.3 Create the Fleet VMs

The create-vmfleet script performs the following steps. It is idempotent, i.e. it can be rerun if failures occur to complete the specified deployment.

- deploys one internal vmswitch per node with the IPv4 APIPA IP 169.254.1.1; this will be the connectivity for the VMs back through the host
- copies a gold/base VHD per VM
- instantiates a VM over that VHD
- instantiates a clustered VM role for the VM
- specializes the VM/VHD
    - sets up autologin of the administrative account
    - installs VM fleet launch scripting
    - creates a sample load file for DISKSPD
    - creates an identification file naming the VM (c:\vmspec.txt)

To prepare for deployment, provide access to the VHD prepared as specific in Section 3. Create-vmfleet has the following switches:

- basevhd : the path to the prepared VHD
- vms : the number of vms per node per csv (group) to create
- group : specify an explicit group; else (default) it is inherited from the suffix of the CSV virtualdisk friendlyname, i.e.: <nodename>-**<suffix>**

- adminpass : password for the VM-local administrative user
- admin : (default: administrator) name of the VM-local administrative user
- connectpass : password for the user to establish the loopback connection to the host
- connectuser: name of the user to establish the loopback connection to the host
- stopafter : (not normally needed) used for triage, halts deployment at a specific step for each VM
- specialize : specifies whether specialization should
    - auto : (default) be done as each VM is deployed
    - none : not be performed
    - force : always be performed, even if the VM is already deployed

Note that specialization requires that the VM be offline. If the VM is online, the specialization process cannot mount the VHD to inject the content.

If an alternate network configuration is desirable, update create-vmfleet and ensure master.ps1 is modified as needed so that the VMs can establish their loopback connection to their host to CSV.

The default VM sizing follows the defaults for the New-VM cmdlet. To set a specific VM sizing, use the set-vmfleet.ps1 script. Its options follow the Set-VM cmdlet:

- ProcessorCount : VP count per VM
- MemoryStartupBytes : memory reserved at VM startup
- MemoryMaximumBytes : maximum memory per VM
- DynamicMemory : (default: no, i.e. StaticMemory) whether dynamic memory is enabled

The Azure A-series VM sizes provide a baseline for VM sizing.

https://azure.microsoft.com/en-us/pricing/details/virtual-machines

4 Run the VM Fleet

At this point, the VM fleet should be ready for operation.

In addition to the pause/stop/start scripts, the basic control mechanism involves the VMs watching for an updated run*.ps1 script (note the wildcard) in the control directory. The master script checks every five seconds for pause or run script updates, so any changes should propagate in near real-time to the fleet.

4.1 Automated Sweeps

As of version 0.4, VM Fleet has an initial set of mechanics for automated performance sweeps. This is based on an epoch ask/response similar to how pause works.

- master tracks a “go” epoch and listens for a “done” response from run scripts
- when “done” is received, master drops a done flag file indicating the “go” epoch
- start-sweep uses this interaction to step the fleet through a sequence of run scripts, ensuring that the fleet has successfully completed each step before moving to the next
    - generated off of run-sweeptemplate.ps1
    - place into run-sweep.ps1 for each step

To use this, look in start-sweep for this block:

```
############################
############################
## Modify from here down
############################
############################
```

Below this point is a substitution list (actually, hashtable) which enumerates a set of substitutions to make in the template run file for the sweep, by default run-sweeptemplate.ps1. Fill in the list with defaults and loop the rest.

If additional controls are needed on DISKSPD, or any other tooling, simply have the substitution list match what needs to be rewritten in the template run file and update the top of the **do-run** function to allow for control of the necessary parameters.

The default template will drop DISKSPD XML results into the collect/control/result folder, which must already exist. It is currently a flagged error if any VM indicates early completion of a step, so ensure that the result folder does not have pre-existing results.

Start-sweep takes a parameter which inserts an arbitrary string into the result file name, “addspec”. Use this to differentiate distinct sweeps that vary external parameters such as the number of active VMs in the fleet, vCPU counts, CSV placement, and so forth.

Enjoy!

5 Storage QoS

Storage Quality of Service is a new capability for Windows Server 2016. To use this with the VM Fleet, define one or more MultiInstance polices. Examples:

```
New-StorageQosPolicy -Name SilverVM -MinimumIops 100 -MaximumIops 500 -
PolicyType MultiInstance
New-StorageQosPolicy -Name GoldVM -MinimumIops 500 -MaximumIops 5000 -
PolicyType MultiInstance
New-StorageQosPolicy -Name PlatinumVM -MinimumIops 500 -MaximumIops 10000 -
PolicyType MultiInstance
```

These names correspond to those used within the demo.ps1 demonstration script. The individually specify a range of 20x (500 – 10,000) IOP controls to put on the VMs. To then apply these policies to the VMs, use the set-storageqos.ps1 script.

6 Further Resources

For information on Storage Spaces Direct (S2D), including deployment instructions: http://aka.ms/s2d

To download a DISKSPD binary and its documentation: http://aka.ms/diskspd

VM Fleet is now part of DISKSPD. DISKSPD is OSS, hosted on GitHub: https://github.com/microsoft/diskspd

The binary distribution may lag the OSS repo from time to time. The most up to date versions of DISKSPD and the VM Fleet will be found at GitHub.