FlexCache for hotspot remediation

ONTAP 9

NetApp
February 12, 2026

Table of Contents

FlexCache for hotspot remediation

Remediating hotspotting in high-performance compute workloads with ONTAP FlexCache volumes

A common problem with many high-performance compute workloads, such as animation rendering or EDA, is hotspotting. Hotspotting is a situation that occurs when a specific part of the cluster or network experiences a significantly higher load compared to other areas, leading to performance bottlenecks and reduced overall efficiency due to excessive data traffic concentrated in that location. For example, a file, or multiple files, is in high demand for the job running which results in a bottleneck at the CPU used to service requests (via a volume affinity) to that file. FlexCache can help alleviate this bottleneck, but it must be set up properly.

This documentation explains how to set up FlexCache to remediate hotspotting.

> Beginning July 2024, content from technical reports previously published as PDFs has been integrated with ONTAP product documentation. This ONTAP hotspot remediation technical report content is net new as of the date of its publication and no earlier format was ever produced.

Key concepts

When planning hotspot remediation, it's important to understand these essential concepts.

- **High-density FlexCache (HDF)**: A FlexCache that is condensed to span as few nodes as the cache capacity requirements allow
- **HDF Array (HDFA)**: A group of HDFs that are caches of the same origin, distributed across the cluster
- **Inter-SVM HDFA**: One HDF from the HDFA per server virtual machine (SVM)
- **Intra-SVM HDFA**: All HDFs in the HDFA in one SVM
- **East-west traffic**: Cluster backend traffic generated from indirect data access

What's next

- Understand how to architect with high-density FlexCache to help remediate hotspotting
- Decide on FlexCache array density
- Determine the density of your HDFs and decide whether you will be accessing the HDFs using NFS with inter-SVM HDFAs and intra-SVM HDFAs
- Configure HDFA and the data LIFs to realize the benefits of using intracluster caching with ONTAP configuration
- Learn how to configure clients to distribute ONTAP NAS connections with client configuration

Architecting an ONTAP FlexCache hotspot remediation solution

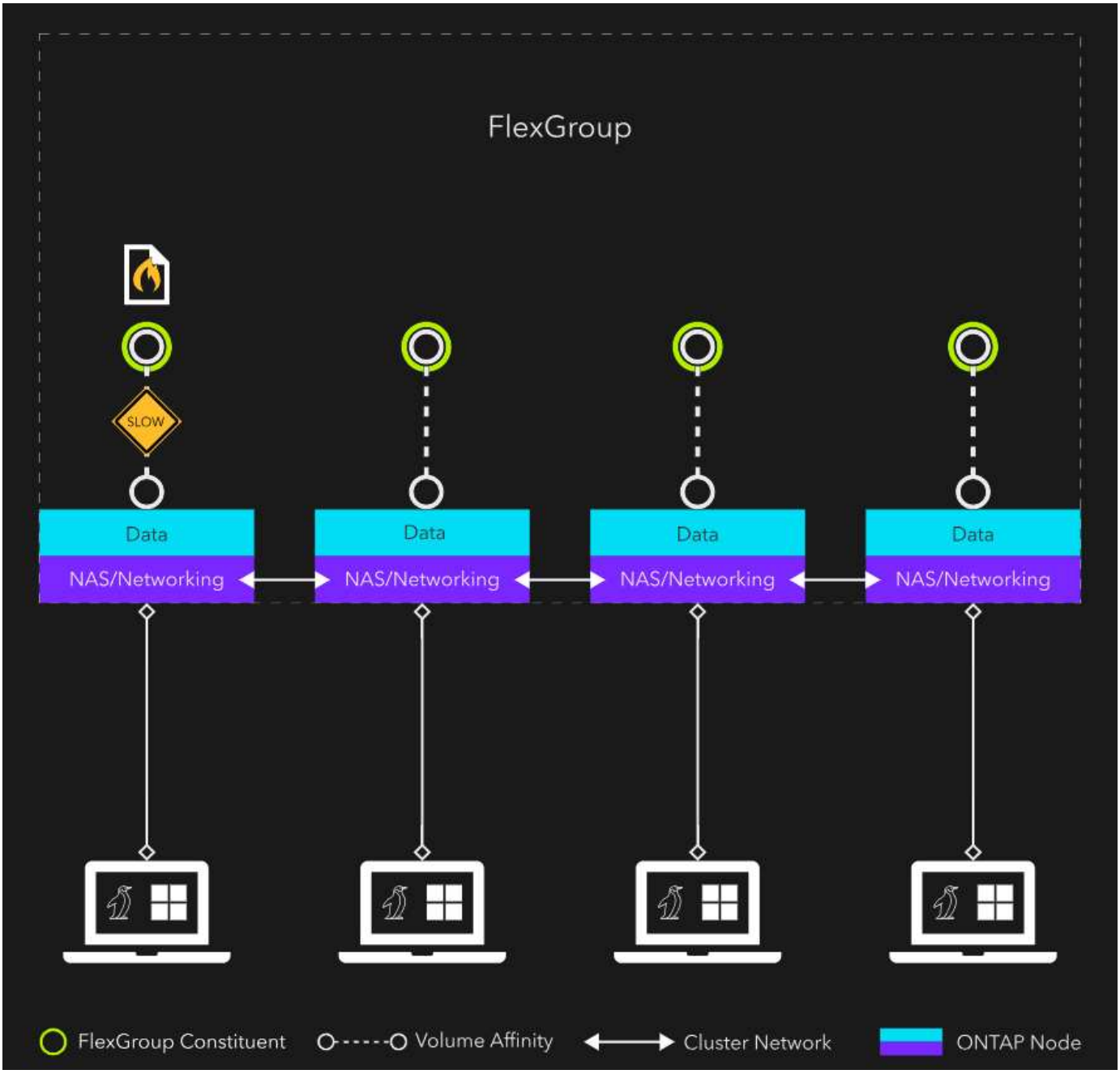
To remediate hotspotting, explore the underlying causes of bottlenecks, why auto-provisioned FlexCache isn’t sufficient, and the technical details necessary to effectively architect a FlexCache solution. By understanding and implementing high-density FlexCache arrays (HDFAs), you can optimize performance and eliminate bottlenecks in your high-demand workloads.

Understanding the bottleneck

The following image shows a typical single-file hotspotting scenario. The volume is a FlexGroup with a single constituent per node, and the file resides on node 1.

If you distribute all of the NAS clients' network connections across different nodes in the cluster, you still bottleneck on the CPU that services the volume affinity where the hot file resides. You also introduce cluster network traffic (east-west traffic) to the calls coming from clients connected to nodes other than where the file resides. The east-west traffic overhead is typically small, but for high-performance compute workloads every little bit counts.

Figure 1: FlexGroup single-file hotspot scenario

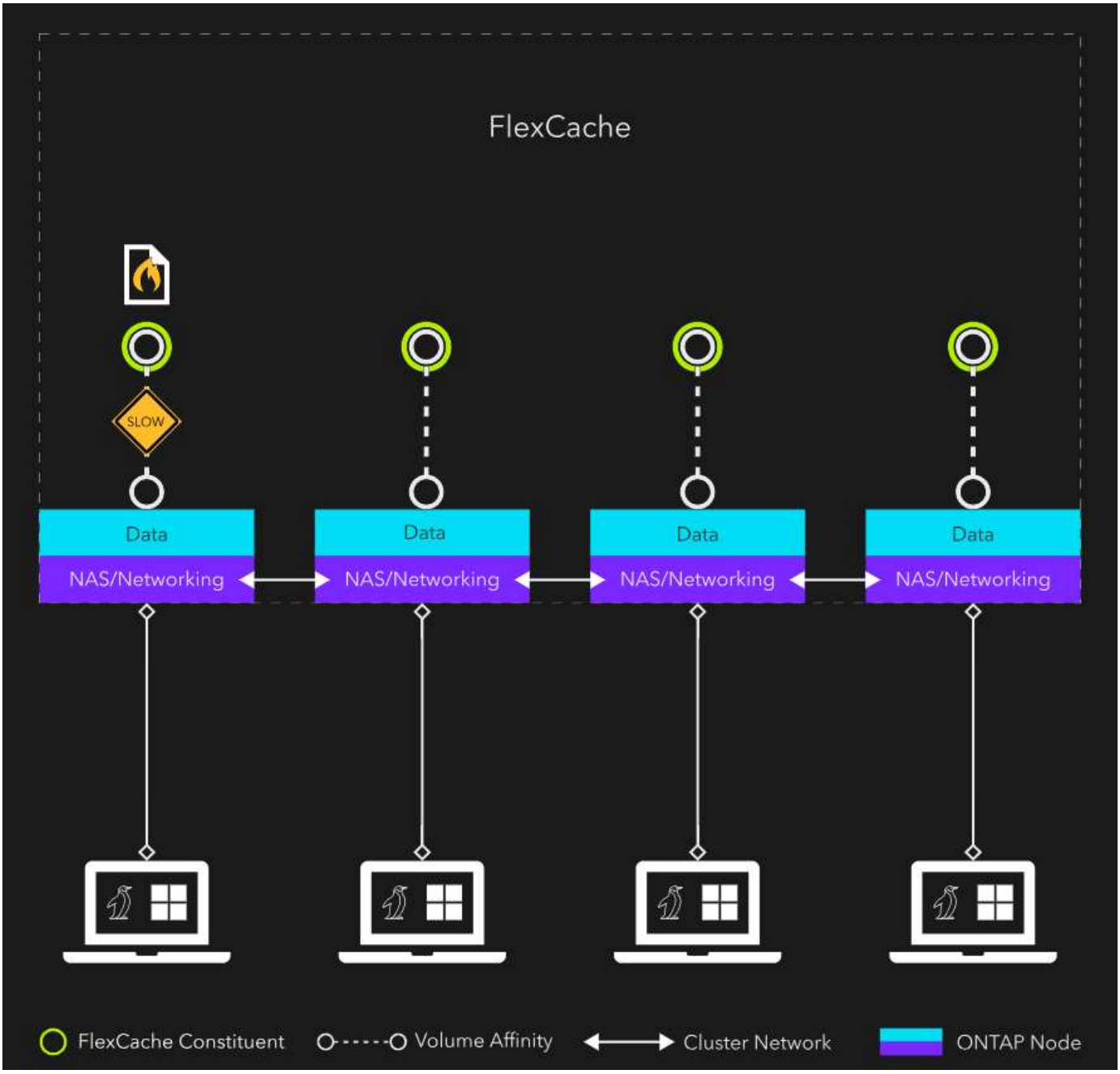


Why an auto-provisioned FlexCache isn't the answer

To remedy hotspotting, eliminate the CPU bottleneck and preferably the east-west traffic too. FlexCache can help if set up properly.

In the following example, FlexCache is auto-provisioned with System Manager, NetApp Console, or default CLI arguments. Figure 1 and figure 2 at first appear the same: both are four-node, single-constituent NAS containers. The only difference is that figure 1's NAS container is a FlexGroup, and figure 2's NAS container is a FlexCache. Each figure profiles the same bottleneck: node 1's CPU for volume affinity servicing access to the hot file, and east-west traffic contributing to latency. An auto-provisioned FlexCache hasn't eliminated the bottleneck.

Figure 2: Auto-provisioned FlexCache scenario



Anatomy of a FlexCache

To effectively architect a FlexCache for hotspot remediation, you need to understand some technical details about FlexCache.

FlexCache is always a sparse FlexGroup. A FlexGroup is made up of multiple FlexVols. These FlexVols are called FlexGroup constituents. In a default FlexGroup layout, there are one or more constituents per node in the cluster. The constituents are "sewn together" under an abstraction layer and presented to the client as a single large NAS container. When a file is written to a FlexGroup, ingest heuristics determine which constituent the file will be stored on. It might be a constituent containing the client's NAS connection or it might be a different node. The location is irrelevant because everything operates under the abstraction layer and is invisible to the client.

Let's apply this understanding of FlexGroup to FlexCache. Because FlexCache is built on a FlexGroup, by default you have a single FlexCache that has constituents on all the nodes in the cluster, as depicted in figure

1. In most cases, this is a great thing. You are utilizing all the resources in your cluster.

For remediating hot files, however, this isn't ideal because of the two bottlenecks: CPU for a single file and east-west traffic. If you create a FlexCache with constituents on every node for a hot file, that file will still reside on only one of the constituents. This means there's one CPU to service all access to the hot file. You also want to limit the amount of east-west traffic required to reach the hot file.

The solution is an array of high-density FlexCaches.

Anatomy of a high-density FlexCache

A high-density FlexCache (HDF) will have constituents on as few nodes as the capacity requirements for the cached data allow. The goal is to get your cache to live on a single node. If capacity requirements make that impossible, you can have constituents on only a few nodes instead.

For example, a 24-node cluster could have three high-density FlexCaches:

- One that spans nodes 1 through 8
- A second that spans nodes 9 through 16
- A third that spans nodes 17 through 24

These three HDFs would make up one high-density FlexCache array (HDFA). If the files are evenly distributed within each HDF, you will have a one-in-eight chance that the file requested by the client resides local to the front-end NAS connection. If you were to have 12 HDFs that span only two nodes each, you have a 50% chance of the file being local. If you can collapse the HDF down to a single node, and create 24 of them, you are guaranteed that the file is local.

This configuration will eliminate all east-west traffic and, most importantly, will provide 24 CPUs/volume affinities for accessing the hot file.

What's next?

Decide on FlexCache array density

Related information

Documentation on FlexGroup and TRs

Determine ONTAP FlexCache density

Your first hotspot remediation design decision is to figure out FlexCache density. The following examples are four-node clusters. Assume that the file count is evenly distributed among all the constituents in each HDF. Assume also an even distribution of frontend NAS connections across all nodes.

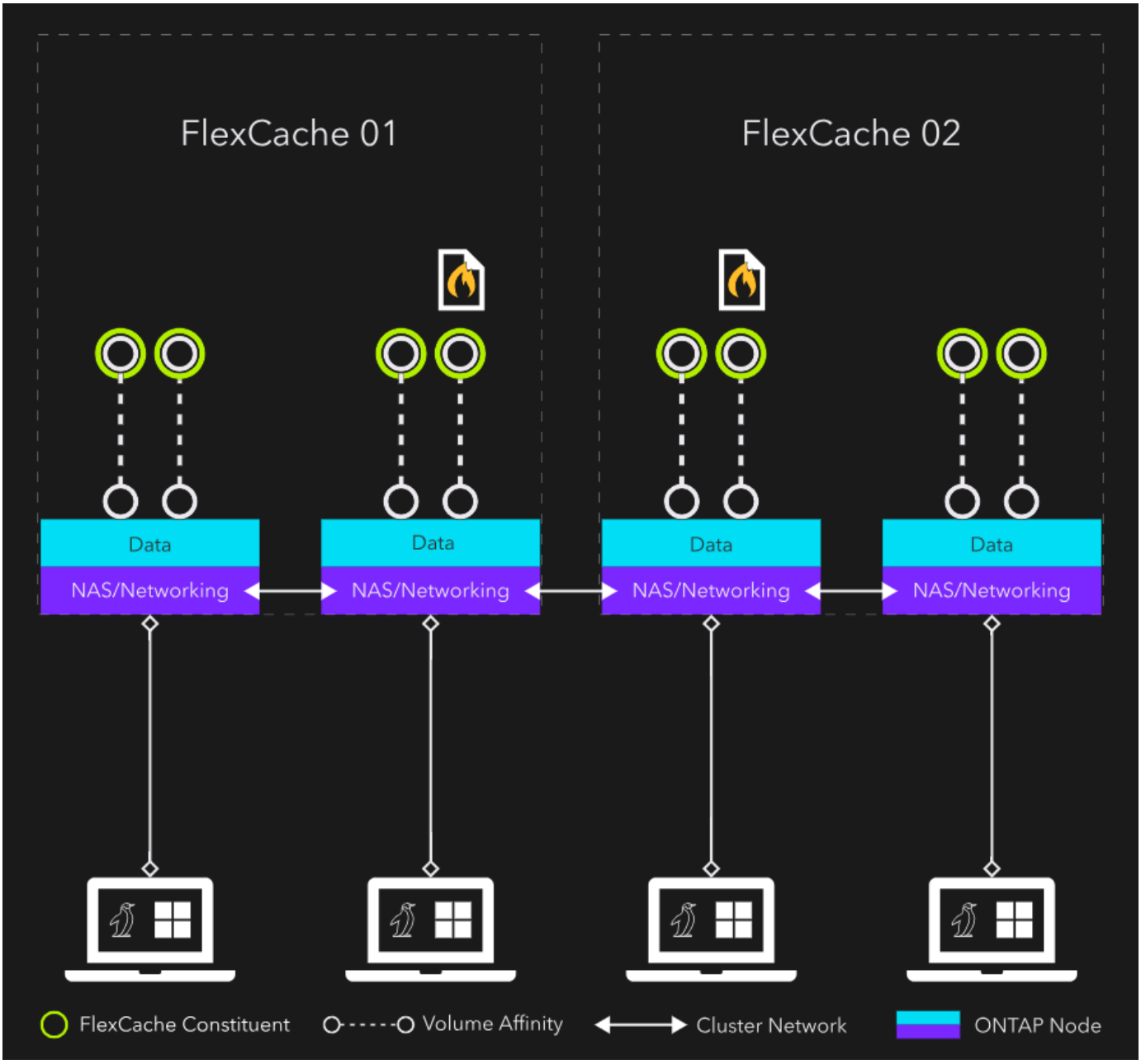
Although these examples aren't the only configurations you can use, you should understand the guiding design principle to make as many HDFs as your space requirements and available resources allow.

HDFAs are represented using the following syntax: `HDFs per HDFA x nodes per HDF x constituents per node per HDF`

2x2x2 HDFA configuration

Figure 1 is an example of a 2x2x2 HDFA configuration: two HDFs, each spanning two nodes, and each node containing two constituent volumes. In this example, each client has a 50% chance of having direct access to the hot file. Two of the four clients have east-west traffic. Importantly, there are now two HDFs, which means two distinct caches of the hot file. There are now two CPUs/volume affinities servicing access to the hot file.

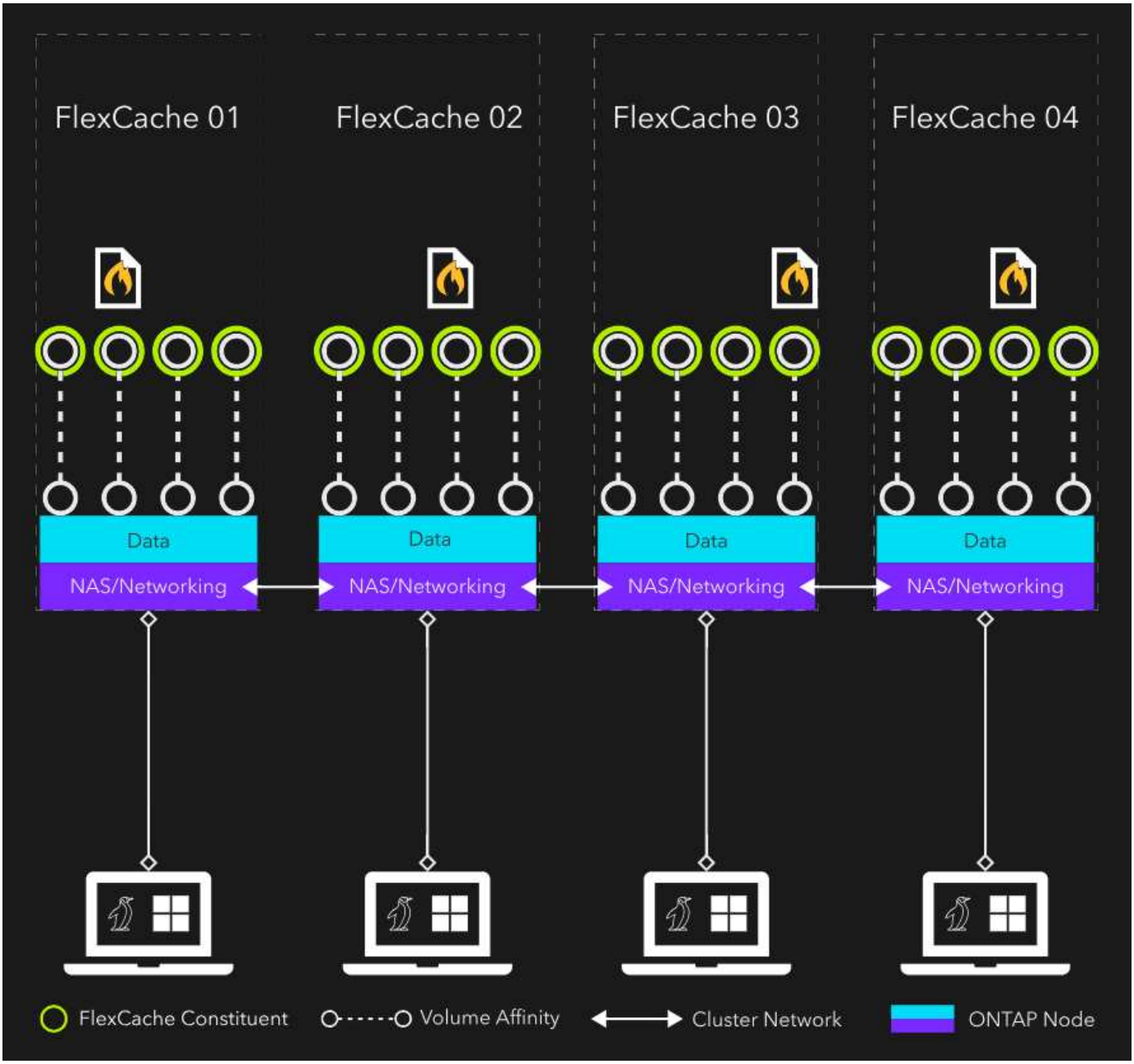
Figure 1: 2x2x2 HDFA configuration



4x1x4 HDFA configuration

Figure 2 represents an optimal configuration. It is an example of a 4x1x4 HDFA configuration: four HDFs, each contained to a single node, and each node containing four constituents. In this example, each client is guaranteed to have direct access to a cache of the hot file. Since there are four cached files on four different nodes, four different CPUs/volume affinities help service access to the hot file. Additionally, there is zero east-west traffic generated.

Figure 2: 4x1x4 HDFA configuration



What's next

After you decide how dense you want to make your HDFs, you must make another design decision if you will be accessing the HDFs with NFS with inter-SVM HDFAs and intra-SVM HDFAs.

Determine an ONTAP inter-SVM or intra-SVM HDFA option

After you determine the density of your HDFs, decide whether you will be accessing the HDFs using NFS and learn about inter-SVM HDFA and intra-SVM HDFA options.

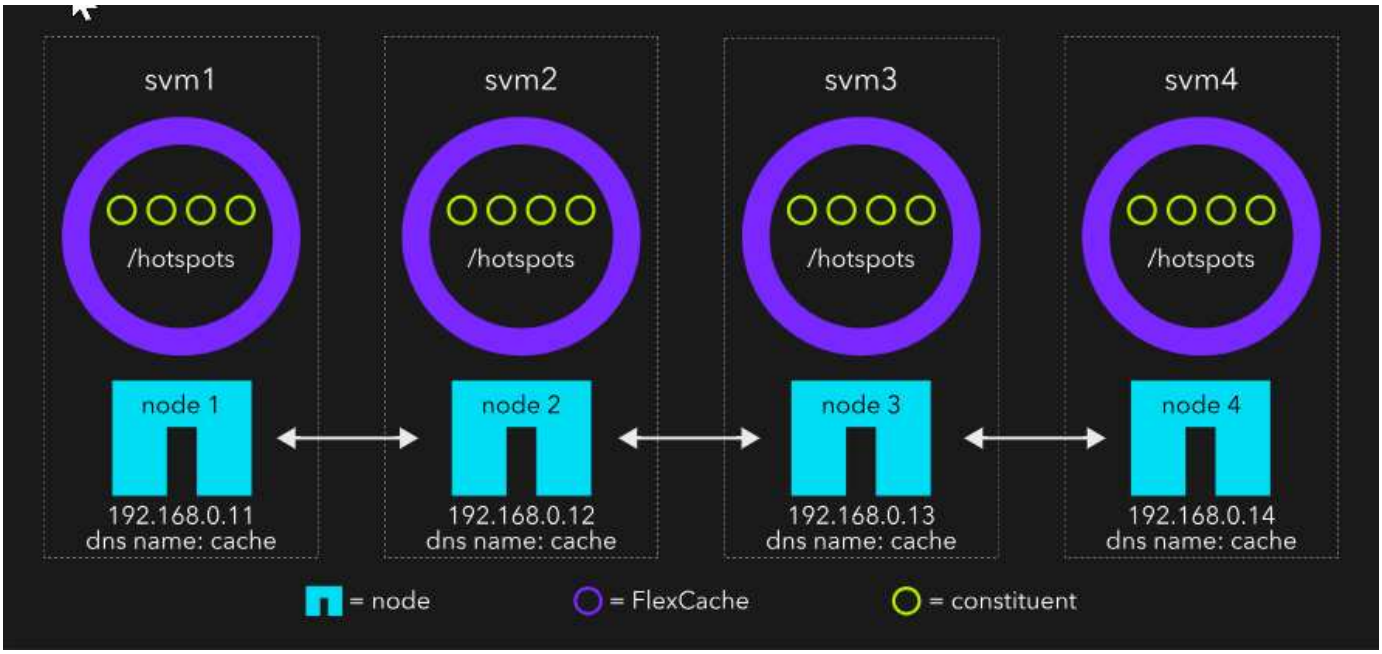
If only SMB clients will be accessing the HDFs, you should create all HDFs in a single SVM. Refer to Windows client configuration to see how to use DFS targets for load balancing.

Inter-SVM HDFA deployment

An inter-SVM HDFA requires an SVM be created for each HDF in the HDFA. This allows all HDFs within the HDFA to have the same junction-path, allowing for easier configuration on the client side.

In the figure 1 example, each HDF is in its own SVM. This is an inter-SVM HDFA deployment. Each HDF has a junction-path of /hotspots. Also, every IP has a DNS A record of hostname cache. This configuration leverages DNS round-robin to load balance mounts across the different HDFs.

Figure 1: 4x1x4 inter-SVM HDFA configuration

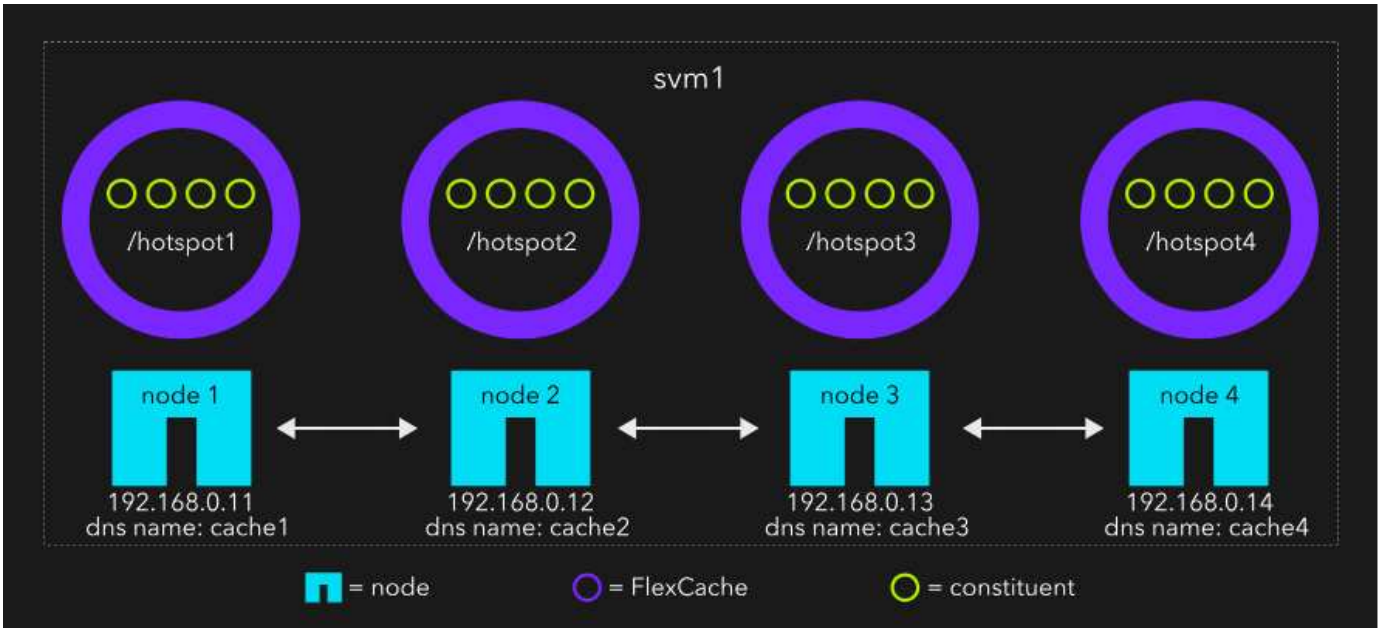


Intra-SVM HDFA deployment

An intra-SVM requires each HDF to have a unique junction-path, but all HDFs are in one SVM. This setup is easier in ONTAP because it requires only one SVM, but it needs more advanced configuration on the Linux side with `autofs` and data LIF placement in ONTAP.

In the figure 2 example, every HDF is in the same SVM. This is an intra-SVM HDFA deployment and requires the junction-paths to be unique. To make load balancing work appropriately, you'll need to create a unique DNS name for each IP and place the data LIFs the hostname resolves to only on the nodes where the HDF resides. You'll also need to configure `autofs` with multiple entries as covered in Linux client configuration.

Figure 2: 4x1x4 intra-SVM HDFA configuration



What's next

Now that you have an idea of how you want to deploy your HDFAs, deploy the HDFA and configure the clients to access them in a distributed fashion.

Configure HDFAs and ONTAP data LIFs

You'll need to configure the HDFA and the data LIFs appropriately to realize the benefits of this hotspot remediation solution. This solution uses intracluster caching with the origin and HDFA in the same cluster.

The following are two HDFA sample configurations:

- 2x2x2 inter-SVM HDFA
- 4x1x4 intra-SVM HDFA

About this task

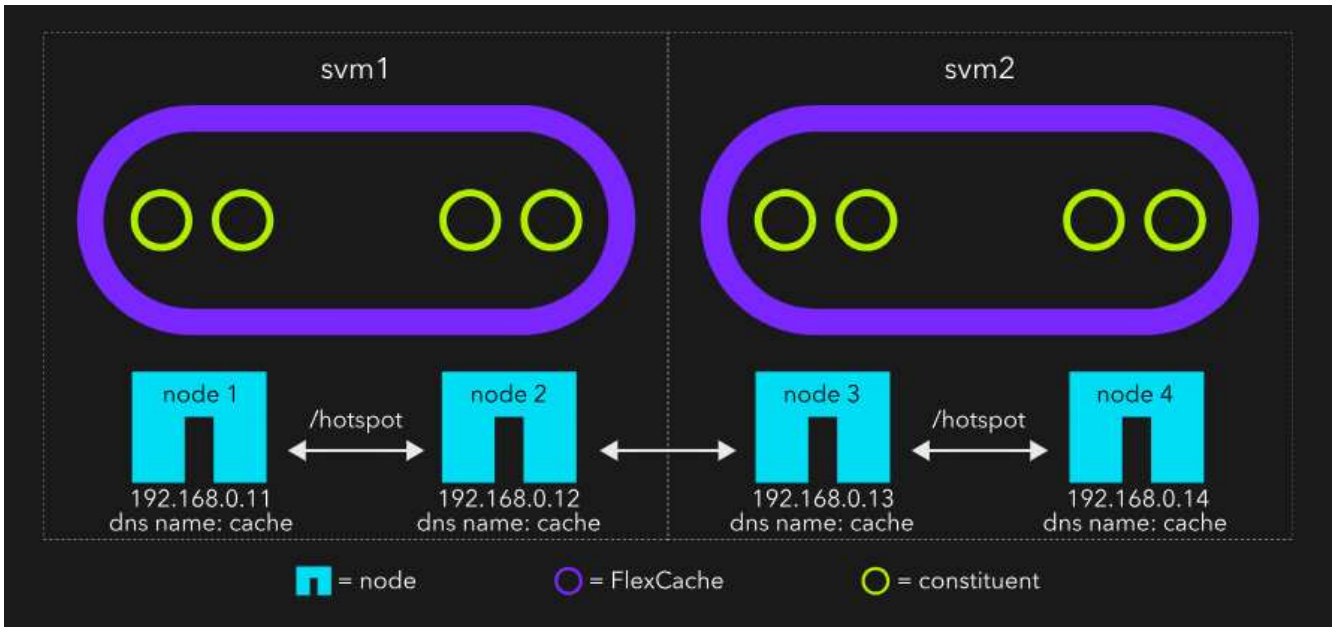
Perform this advanced configuration using the ONTAP CLI. There are two configurations you must use in the `flexcache create` command, and one configuration you must make sure isn't configured:

- `-aggr-list`: Provide an aggregate, or list of aggregates, that reside on the node or subset of nodes you want to restrict the HDF to.
- `-aggr-list-multiplier`: Determine how many constituents will be created per aggregate listed in the `aggr-list` option. If you have two aggregates listed, and set this value to `2`, you will end up with four constituents. NetApp recommends up to 8 constituents per aggregate, but 16 is also sufficient.
- `-auto-provision-as`: If you tab out, the CLI will try to autofill and set the value to `flexgroup`. Make sure this isn't configured. If it appears, delete it.

Create a 2x2x2 inter-SVM HDFA configuration

1. To assist in configuring a 2x2x2 inter-SVM HDFA as shown in Figure 1, complete a prep sheet.

   Figure 1: 2x2x2 Inter-SVM HDFA layout



SVM	Nodes per HDF	Aggregates	Constituents per node	Junction path	Data LIF IPs
svm1	node1, node2	aggr1, aggr2	2	/hotspot	192.168.0.11,192.168.0.12
svm2	node3, node4	aggr3, aggr4	2	/hotspot	192.168.0.13,192.168.0.14

2. Create the HDFs. Run the following command twice, once for each row in the prep sheet. Make sure you adjust the `vserver` and `aggr-list` values for the second iteration.

```
cache::> flexcache create -vserver svm1 -volume hotspot -aggr-list
aggr1,aggr2 -aggr-list-multiplier 2 -origin-volume <origin_vol> -origin
-vserver <origin_svm> -size <size> -junction-path /hotspot
```

3. Create the data LIFs. Run the command four times, creating two data LIFs per SVM on the nodes listed in the prep sheet. Make sure you adjust the values appropriately for each iteration.

```
cache::> net int create -vserver svm1 -home-port e0a -home-node node1
-address 192.168.0.11 -netmask-length 24
```

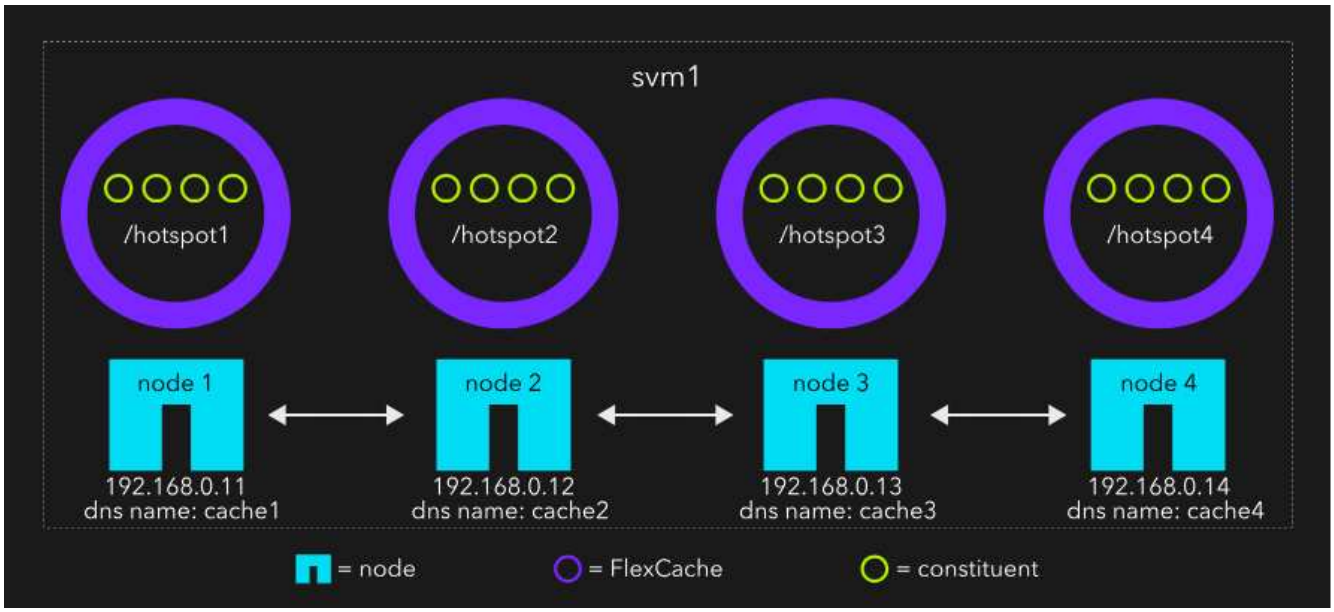
What's next

Now you need to configure your clients to utilize the HDFA appropriately. See client configuration.

Create a 4x1x4 intra-SVM HDFA

1. To assist in configuring a 4x1x4 inter-SVM HDFA as shown in figure 2, fill out a prep sheet.

Figure 2: 4x1x4 intra-SVM HDFA layout



SVM	Nodes per HDF	Aggregates	Constituents per node	Junction path	Data LIF IPs
svm1	node1	aggr1	4	/hotspot1	192.168.0.11
svm1	node2	aggr2	4	/hotspot2	192.168.0.12
svm1	node3	aggr3	4	/hotspot3	192.168.0.13
svm1	node4	aggr4	4	/hotspot4	192.168.0.14

2. Create the HDFs. Run the following command four times, once for each row in the prep sheet. Make sure you adjust the `aggr-list` and `junction-path` values for each iteration.

```
cache::> flexcache create -vserver svm1 -volume hotspot1 -aggr-list
aggr1 -aggr-list-multiplier 4 -origin-volume <origin_vol> -origin
-vserver <origin_svm> -size <size> -junction-path /hotspot1
```

3. Create the data LIFs. Run the command four times, creating a total of four data LIFs in the SVM. There should be one data LIF per node. Make sure you adjust the values appropriately for each iteration.

```
cache::> net int create -vserver svm1 -home-port e0a -home-node node1
-address 192.168.0.11 -netmask-length 24
```

What's next

Now you need to configure your clients to utilize the HDFA appropriately. See client configuration.

Configure clients to distribute ONTAP NAS connections

To remedy hotspotting, configure the client properly to do its part in preventing CPU bottleneck.

Linux client configuration

Whether you chose an intra-SVM or inter-SVM HDFA deployment, you should use `autofs` in Linux to make sure clients are load-balancing across the different HDFs. The `autofs` configuration will differ for inter- and intra-SVM.

Before you begin

You'll need `autofs` and the appropriate dependencies installed. For help with this, refer to Linux documentation.

About this task

The steps described will use an example `/etc/auto_master` file with the following entry:

```
/flexcache auto_hotspot
```

This configures `autofs` to look for a file called `auto_hotspot` in the `/etc` directory any time a process tries to access the `/flexcache` directory. The contents of the `auto_hotspot` file will dictate which NFS server and junction-path to mount inside the `/flexcache` directory. The examples described are different configurations for the `auto_hotspot` file.

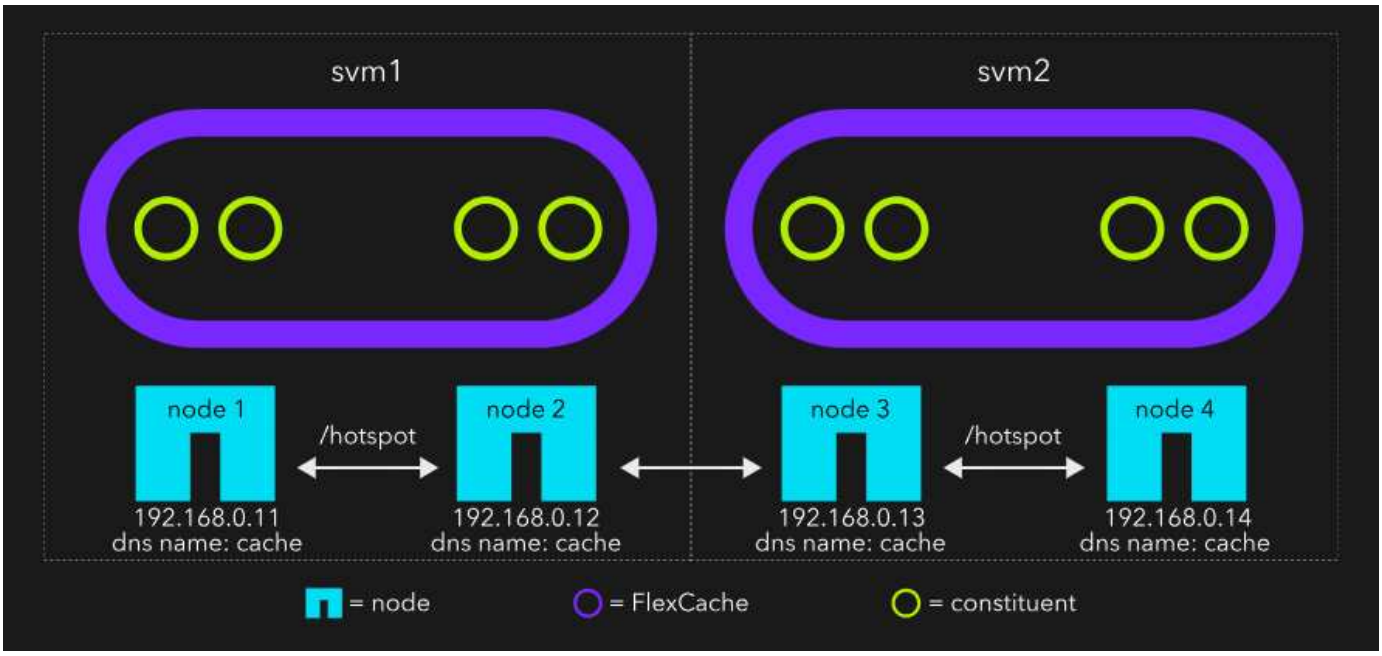
Intra-SVM HDFA autofs configuration

In the following example, we'll create an `autofs` map for the diagram in figure 1. Because each cache has the same junction-path, and the hostname `cache` has four DNS A records, we only need one line:

```
hotspot cache:/hotspot
```

This one simple line will cause the NFS client to do a DNS lookup for hostname `cache`. DNS is setup to return the IPs in a round-robin fashion. This will result in an even distribution of front-end NAS connections. After the client receives the IP, it will mount the junction-path `/hotspot` at `/flexcache/hotspot`. It could be connected to SVM1, SVM2, SVM3, or SVM4, but the particular SVM doesn't matter.

Figure 1: 2x2x2 inter-SVM HDFA



Intra-SVM HDFA autofs configuration

In the following example, we'll create an `autofs` map for the diagram in figure 2. We need to make sure the NFS clients mount the IPs that are a part of the HDF junction-path deployment. In other words, we don't want to mount `/hotspot1` with anything other than IP 192.168.0.11. To do this, we can list all four IP/junction-path pairs for one local mount location in the `auto_hotspot` map.

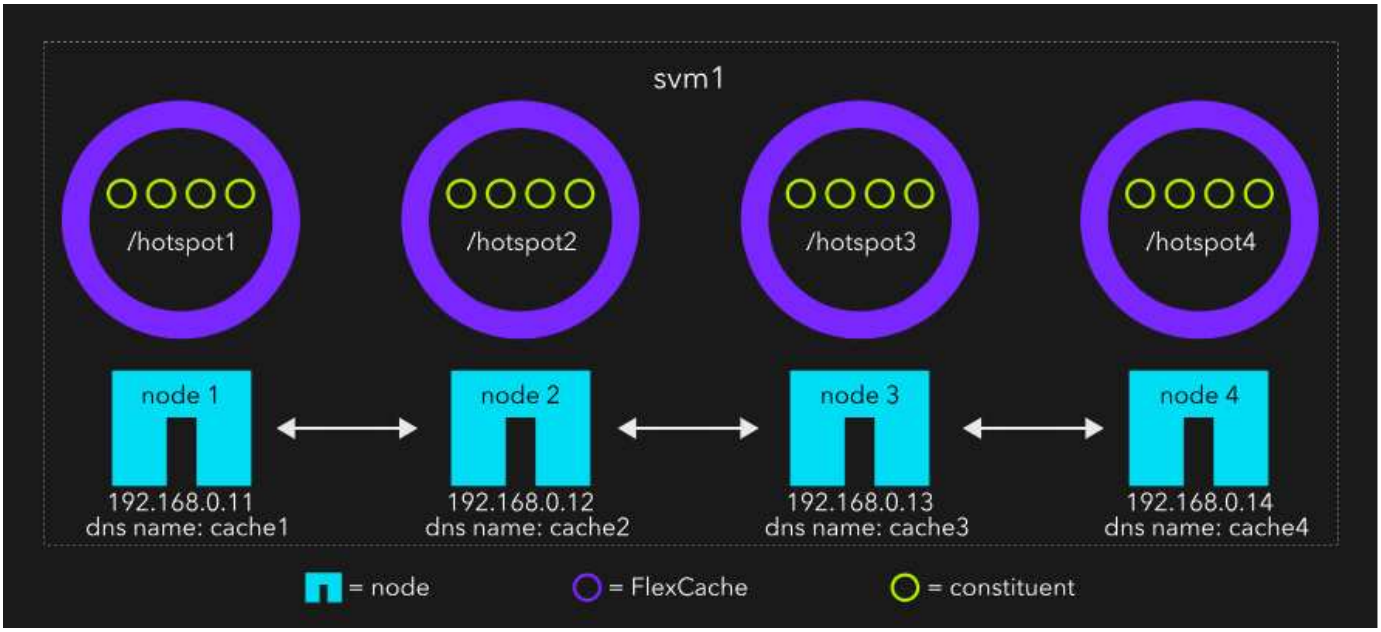
The backslash (`\`) in the following example continues the entry to the next line, making it easier to read.

```
hotspot     cache1:/hostspot1 \
            cache2:/hostspot2 \
            cache3:/hostspot3 \
            cache4:/hostspot4
```

When the client tries to access `/flexcache/hotspot`, `autofs` is going to do a forward-lookup for all four hostnames. Assuming all four IPs are either in the same subnet as the client or in a different subnet, `autofs` will issue an NFS NULL ping to each IP.

This NULL ping requires the packet to be processed by ONTAP's NFS service, but it doesn't require any disk access. The first ping to return is going to be the IP and junction-path `autofs` chooses to mount.

Figure 2: 4x1x4 intra-SVM HDFA



Windows client configuration

With Windows clients, you should use an intra-SVM HDFA. To load balance across the different HDFs in the SVM, you must add a unique share name to each HDF. After that, follow the steps in Microsoft documentation to implement multiple DFS targets for the same folder.

Copyright information

Trademark information