HNAS Storage Pool and HDP Best Practices

By Francisco Salinas and Nathan King
Contributors

The information included in this document represents the expertise, feedback, and suggestions of a number of skilled practitioners. The author would like to recognize and sincerely thank the following contributors and reviewers of this document:

- Al Hagopian
- Bent Knudsen
- Edward Gouin
- Michael Nguyen
- Phil Wood
- Troy Pillon
- Victor Abyad
- HNAS Engineering

Contact

Hitachi Data Systems
2845 Lafayette Street
Santa Clara, California 95050-2627
https://portal.hds.com

North America: 1-800-446-0744
Table of Contents

Introduction........................................................................................................................................1

HNAS storage concepts.....................................................................................................................1
  System drives................................................................................................................................1
  System drive groups .......................................................................................................................1
  Storage pools .................................................................................................................................2
  Storage pool stripesets ...................................................................................................................3
  File systems and chunks.................................................................................................................3
  Tiered file system ..........................................................................................................................6

Dynamic write balancing ..................................................................................................................7
  Dynamic write balancing overview ...............................................................................................7

Using dynamic write balancing .......................................................................................................7
  Grouping system drives ..................................................................................................................7

Dynamic read balancing ..................................................................................................................8
  Dynamic read balancing overview ...............................................................................................8
  Dynamic read balancing caveats and additional use cases ............................................................9

Using dynamic read balancing .......................................................................................................11
  Snapshot considerations ................................................................................................................12
  Dynamic read balancing timing and performance load .................................................................13

Best Practices ..................................................................................................................................13
  Storage Pools .................................................................................................................................13
  Expanding a storage pool ..............................................................................................................14
  Storage pool and EVS affinity .........................................................................................................17
  SD Superflush .................................................................................................................................17
  File systems ......................................................................................................................................18
  File system utilization .....................................................................................................................18
  File system - Allocate on Demand (auto-expand) versus Allocate Now .......................................18
  Storage pool considerations for Allocate on Demand and Allocate Now .....................................20
  File system block size selection ....................................................................................................20
    4 KB file systems ..........................................................................................................................20
    32 KB file systems .......................................................................................................................20

Best Practices for Using HDP with HNAS

July 2014

Hitachi Data Systems
Hitachi Dynamic Provisioning (HDP) Considerations ........................................................................... 20
Mandatory guidelines for HDP use with HNAS ......................................................................................... 20
HDP oversubscription ............................................................................................................................... 21

HDP best practices ................................................................................................................................... 22
System drive groups ................................................................................................................................. 22
HDP Pool sharing ......................................................................................................................................... 22
Recommended settings ............................................................................................................................. 22
Ratio of DP-Volumes to RAID groups ......................................................................................................... 23
Introduction

The Hitachi NAS Platform (HNAS) is a versatile, intelligent, and scalable multi-protocol solution. HNAS can be configured in many ways to meet a variety of requirements. Understanding the principles of how storage is used within HNAS will help you maximize the solutions capabilities and reduce the likelihood of improper configuration. This document details the best practices for configuring and using HNAS storage pools, related features, and Hitachi Dynamic Provisioning (HDP). The document assumes that you are familiar with HDP and HDS storage.

HNAS storage concepts

In order to best configure HNAS, it is important to understand the core concepts of HNAS storage. The next several sections describe these concepts.

System drives

The HNAS system uses the RAID technology on the HDS storage subsystems as a foundation for data protection. The backend storage subsystem is configured using an appropriate data protection scheme (that is, RAID6), and the resulting logical disks are presented as logical units (LUNs) for the HNAS system to use. These LUNs are referred to as system drives (SDs) within HNAS and are combined to create storage pools.

Note: Throughout this document, the terms logical disk, LDEV, LU, and LUN are referred to as LUNs. In this context, LUN refers to a logical disk defined within a RAID group. LDEV is the term used to define storage on Hitachi Virtual Storage Platform, Hitachi Unified Storage VM storage subsystems, and previous generations. LU is the equivalent term used in the Hitachi Unified Storage 100 series storage subsystems and previous generations.

System drive groups

HNAS uses system drive groups (SDGs) to help optimize performance and resiliency. SDGs assist HNAS in understanding the relationship between RAID groups (RGs) and SDs. See Figure 1

SDGs tell HNAS which SDs reside on the same physical RAID group. With this knowledge, HNAS can efficiently write to those SDs to minimize head movement (commonly referred to as thrashing), thus optimizing performance. HNAS can also improve resiliency, ensuring that multiple copies of critical file system metadata are not hosted on the same RAID group.
Each file system maintains a write cursor, and each SD group maintains its own write cursor per file system. The write cursor cycles through the file system linearly, chunk by chunk, and then wraps back around. The individual SDG write cursors move through an entire SD before moving to the next SD in the SDG.

Typically, there is one LUN configured on each RAID group, which mandates one SD for each SDG per HNAS best practices. If more than one LUN is defined within a RAID group, HNAS will map those SDs to a single SDG. HNAS will also configure all LUNs that originate from an HDP Pool into a single SDG.

There are two types of system drive groups, serial and parallel.

- Serial SDGs are used when multiple LUNs are hosted on a single RAID group.
- Parallel SDGs are used for Flash/SSD RAID groups and are also applicable for HDP Pools.

Starting in HNAS release 11.2, HNAS provides programatic grouping of LUNs from Hitachi storage subsystems. Prior to HNAS release 11.2, grouping is performed manually by the administrator.

Storage pools

The HNAS system combines SDs to create a virtualized pool of storage (which is also known as a span in the HNAS CLI command set). Storage pools contain file systems that an administrator can share through the NFS, SMB, FTP, and iSCSI protocols.

Storage pools consist of one or more SDs (a minimum of two is strongly recommended, with four or more being the best practice). Data is striped evenly across all SDs in a stripeset; and a storage pool can have multiple stripesets. When a storage pool is expanded, a new stripeset is created. This allows HNAS to leverage the aggregate throughput of many physical disks and provide optimal performance.

A storage pool provides a logical container that can contain one or more file systems. HNAS supports dynamic expansion of file systems. When expanding file system capacity, it may also be necessary to expand the storage pool itself. In general, GSS strongly recommends expanding the
storage pools using the same number of SDs with the same characteristics as in the first stripeset. For example, a storage pool consisting of four SDs, should be expanded by four SDs. Ideally, the expansion should originate from the same storage subsystem. This will ensure consistent performance for all file systems residing within a storage pool (see Figure 2).

![Figure 2: Adding capacity to a storage pool](image)

**Storage pool stripesets**

Storage pools consist of one or more stripesets. A stripeset is created from a group of SDs. When a storage pool is first created, it has one stripeset. When a storage pool is expanded, a new stripeset is created consisting of the specified new SDs.

**Note:** Stripesets are automatically managed and created by HNAS when a storage pool is created or expanded. They can be displayed by using the HNAS CLI `span-list -sds` command.

In Figure 2, the storage pool’s original capacity (shown in orange) was 6.32 TB, which was striped across four SDs of 1.58 TBs each. This is the initial stripeset of that storage pool. Data is striped evenly in segments of 4 MBs across each SD in that stripeset. When the four new SDs are added, they form a new stripeset (shown in green).

**Best Practice:** GSS strongly recommends that the initial storage pool stripeset contain at least four SDs. More should be used to provide better performance; however, this should be balanced with expansion requirements. Following this recommendation allows for sufficient queue depth to service IO requests. The absolute minimum is two SDs, but again, four or more should be used. When possible, use even numbers of SDs. Do not create multiple LUNs from the same RAID group to achieve these recommendations. Also, do not create more than the recommended number of DP-Volumes from an HDP Pool to satisfy this best practice (see the HDP best practices section for more details). HNAS supports a maximum of 512 SDs. Its scalability will be limited if you create numerous small SDs.

**File systems and chunks**

File systems are provisioned from storage pools, and they can grow, independently of one another, according to guidelines set by the administrator. File systems contain directories, HNAS virtual volumes, files, iSCSI LUNs, metadata, and snapshots.

File systems are created within storage pools. Each file system can be dynamically expanded and offers two block sizes: 4 KB and 32 KB.
File systems are made up of chunks and expanded by chunks (see Figure 3). At a minimum, a file system is made up of at least one chunk and only whole chunks are used. The size of chunk defines the minimum unit by which a file system expands.

Prior to HNAS release 11.2, there was a limit of 1023 chunks per file system and 16384 chunks per storage pool. Chunk size is determined at storage pool creation by the SMU Web Manager or administrator via the CLI. The table below shows examples of different chunk sizes (pre-11.2) and the limits on the file system and storage based on chunk size selection.

<table>
<thead>
<tr>
<th>Chunk Size</th>
<th>Max File System Size</th>
<th>Max Storage Pool Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 GB</td>
<td>10 TB</td>
<td>160 TB</td>
</tr>
<tr>
<td>50 GB</td>
<td>50 TB</td>
<td>800 TB</td>
</tr>
<tr>
<td>256 GB</td>
<td>256 TB</td>
<td>1 PB</td>
</tr>
</tbody>
</table>

Starting with HNAS release 11.2, there have been several changes to the chunk limits within a storage pool. A storage pool and file system can now have a maximum of 60,000 chunks. The file system capacity limit has not changed. To achieve these new limits, HNAS creates sequential “runs” of several chunks. Multiple consecutive chunks can be accounted for using a single entry in the chunk table.
When an HNAS system is upgraded to 11.2, it is possible for existing storage pools to take advantage of new limits. HNAS will automatically amalgamate sequential chunk runs into a single entry when possible. Therefore, it may be possible that an older storage pool or file system, which could not previously expand, can be expanded after upgrading to HNAS release 11.2. After upgrading, use the `filesystem-scalability` CLI command to determine file system scalability. For example:

```
HNAS:~ $ filesystem-scalability storagepool1
----------- Current ----------- | ----------- Expected -----------
Chunks Runs Ratio Cap/GiB | Chunks Cap/GiB | Chunks Cap/GiB | Name
------------------- | ----------------- | ----------------- | -----
696 688 1.01116 | 1031 1344 | 6056 7896 | fs1
714 308 2.31818 | 1429 1863 | 12154 15847 | fs2
669 9 74.3333 | 1683 2194 | 16893 22026 | fs3
696 4 174 | 1715 2236 | 17000 22165 | fs4
```

HNAS attempts to maintain the highest ratio possible in a chunk run to allow a file system to reach its maximum growth potential. For more details, see the `chunk`, `filesystem-scalability` and `span-tune-allocator` man pages.

The table below shows examples of different chunk sizes (11.2 and newer) and the limits on the file system and storage based on chunk size selection.

<table>
<thead>
<tr>
<th>Chunk size</th>
<th>Max file system size</th>
<th>Max storage pool size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GB</td>
<td>60 TB</td>
<td>60 TB</td>
</tr>
<tr>
<td>5 GB</td>
<td>256 TB</td>
<td>292 TB</td>
</tr>
<tr>
<td>18 GB</td>
<td>256 TB</td>
<td>1 PB</td>
</tr>
</tbody>
</table>

In HNAS release 11.2, HNAS will automatically determine a chunk size based on the initial capacity of the storage pool when it is created.

**Note:** When downgrading from HNAS release 11.2, any storage pools or file systems that have greater than 16384 or 1023 chunks, respectively, will not be useable.

**Best Practice:** It is important to take into account the maximum desired size of the file system and storage pool when creating the initial storage pool. The default storage pool chunk size is 1/3750 of the initial capacity. A user-defined chunk size can be specified from the HNAS CLI. When using HNAS release 12.0 or newer, the Web Manager GUI defaults to a chunk size of 18 GiB. The HNAS release 12.0 CLI `span-create` command still uses the algorithm described above.

When creating storage pools that consist of multiple stripesets and new file systems on those storage pools, it is important to take note of the file system size. A file system should be large enough so that it spans all the stripesets in a storage pool. This ensures optimal performance by spreading IO load across all of the SDs in the storage pool. As shown in Figure 4, depending on
the chunk size and run bias, a file system may not initially utilize chunks across all the stripesets (such as File System B). It is possible to adjust the run bias at the storage pool level using the `span-tune-allocator --run-bias` CLI command; however, this is not generally recommended and the run bias should only be modified under direction from Hitachi Support.

![Figure 4 - Chunk runs](image)

**Tiered file system**

Tiered File System (TFS) is a feature that automatically and intelligently separates data and metadata onto different tiers of storage. In a TFS, the underlying storage pool can support up to two tiers and is called a Tiered Storage Pool (TSP). All file systems created from a TSP are (and can only be) tiered file systems and leverage the intelligent metadata separation. A TSP defines the two tiers as Tier 0 and Tier 1. This is not to be confused with the storage tiers referenced in the *Hitachi NAS Platform Storage Subsystem Administration Guide*, because a Storage Pool Tier can still leverage any form of storage array tier.

For example, SSD is usually referred to as Tier 0, FC drives and SAS drives are generally referred to as Tier 1, and both NLSAS and SATA are often referred to as Tier 2 Storage. A Storage Pool Tier can use any of these storage array tiers.

The Storage Pool Tier 0 is the tier that supports the metadata transactions, which tends to be very random, small IO. As such, it is highly recommended that the Storage Pool Tier 0 be composed of high-speed disk technologies like SSD or Flash. The second half of the Storage Pool, Tier 1, will continue to hold the user and application data. The Storage Pool Tier 1 has the potential to use cost-effective disk technologies depending on the overall workload profile and the level of metadata offloading. This new ability to create Tiered Storage Pool combinations creates the availability of new price points for storage when mixing the technologies together. Since metadata is nearly always required for file system operations, separating it onto fast storage can greatly improve the responsiveness of the Hitachi NAS Platform. Isolating metadata IO from data IO can often improve the performance when compared to file systems that only implement a single tier of storage.

Figure 5 illustrates the layout of a tiered file system.
Best Practice: During normal operation, one tier of a tiered storage pool might become filled before the other tier. In such a case, one tier of the storage pool can be expanded. In general, the expansion should follow the same storage pool expansion practices outlined in this document i.e. if the tier originally consisted of 4 SDs, then it should be expanded by another 4. Note that, when expanding a tier, you must make certain that the SD being added to the tier has the same performance characteristics as the SDs already in the tier (for example, do not add NLSAS based SDs to a tier already made up of SSD/flash).

For more information see the Tiered File System Best Practices Guide (MK-92HNAS038).

Dynamic write balancing

Dynamic write balancing overview

Dynamic write balancing (DWB) allows HNAS file systems to take advantage of new storage added to a storage pool. It ensures that data is written to all SDs in a storage pool. This allows for increased performance as the storage pool grows, and widens storage bandwidth available for writes.

Note: A file system must be expanded onto the newly added stripeset for DWB to utilize the new SDs.

Best Practice: Expand all file systems in a storage pool by the same proportion as you expanded the storage pool. This helps balance workload across the storage.

Using dynamic write balancing

In order for DWB to function, SDs must be placed into SDGs. In HNAS versions prior to 11.2, SD grouping is configured through the System Management Unit (SMU) Web Manager or through the HNAS CLI. In 11.2 and higher, SDs are grouped automatically when a storage pool is created or expanded through CLI or Web Manager.

Grouping system drives

The grouping of system drives is performed using the following command: `sd-group-auto` from the HNAS CLI command set from HNAS release 11.2 onwards.
**Note:** For HDP pools and SSD/flash storage use parallel SDGs. The previous best practice of configuring DP-Vols in single SDG with HNAS release 11.2 and above is no longer valid.

## Dynamic read balancing

### Dynamic read balancing overview

Dynamic read balancing (DRB) is a utility designed to redistribute static data such that it is evenly balanced among all SDGs that a file system is using. For example, say a single file system, which is 90% utilized, is configured to use all the capacity on a storage pool that consists of 4 SDs. More read performance is required, so the storage pool and then the file system are expanded. After the expansions, the majority of the data still resides on the original SDs. Since the data is mostly static, it will not make use of the new SDG’s and not be able to benefit from the additional performance the storage pool is now capable of. This is where DRB helps. DRB will move the static data to the new SDG’s thereby improving read performance. For active datasets, DRB is not required; the SDGs will become naturally balanced over time through dynamic write balancing. The figure below illustrates the described example.

**Note:** In the example below, each SDG contains 1 SD.

![File System before storage pool and capacity expansion](image1)

![File System after storage pool and capacity expansion](image2)

![File System after running DRB](image3)

The dynamic read balancing utility examines a file system and its chunks, as well as the SDGs the chunks reside on. Dynamic read balancing determines the utilization rate of the file systems’ SDGs and how much data needs to be moved. Based on this information, Dynamic read balancing then partially disables dynamic write balancing, moves data, and forces all writes to the chunks on the newly added SDGs until they are balanced. This does two things:

1. It ensures data is distributed to the correct SDGs.
2. It also forces data to be written by the client to the new SDGs. This speeds up the balancing process.
As Dynamic read balancing is writing data to the new SDGs, it does not block client access to any files. Dynamic read balancing only moves the data stream portion of the files. Also, it keeps track of its progress during each checkpoint by saving a “resume point” on disk. This ensures that it does not rebalance incorrect data if it is paused or stopped.

**Note:** Dynamic read balancing balances based on the utilization rate, not the amount of data on the SDG.

**IMPORTANT:** Dynamic read balancing checks to ensure that the last expansion was done solely onto new storage. If not, then the Dynamic read balancing will fail and produce the following message in the event logs:

```
File System: Dynamic read balancing of file system fs1 has failed. Reason: File system is not suitable for read balancing - the last expansion was not solely onto new storage (chunks hosted on a previously utilized stripeset were added).
```

When expanding a file system, the `filesystem-expand --on-stripeset` CLI command (HNAS release 11.2 or newer) can be used to tell HNAS which stripeset to assign chunks from. For example, suppose a file system is to be expanded to take advantage of new SDs recently added to the storage pool on which it resides; however, the storage pool has available chunks prior to adding the new SDs. After the expansion, DRB is to be run on the file system. To ensure DRB will run, the file system must only be expanded onto the new stripeset created when the storage pool was expanded. The referenced CLI command is used to ensure the requirement for DRB is met.

**Note:** Using the `--on-stripeset` option more than a few times per file system may reduce the scalability of the file system because the chunk allocator may not be able to maintain long runs.

**Best Practice:** In order for dynamic read balancing to efficiently operate, the file system should be expanded sufficiently. For example: A file system of 100 GiBs in size resides on a storage pool that consists of four SDs with a chunk size of 5 GiB. The storage pool is expanded with four additional SDs. Ideally, the file system should be expanded by 50% or 50 GiB (10 chunks). Smaller expansions are possible; however, the minimum recommended expansion to use dynamic read balancing is 20% of the file system size. If that file system was only expanded by 5 GiB, dynamic read balancing may not be able to rebalance the data on the SDGs effectively. Also, dynamic read balancing will not run if the expansion is too small.

After dynamic read balancing has finished, it will log an entry in the HNAS event log. For example:

```
File System: Dynamic read balancing of file system FS1 has completed successfully.
```

**Dynamic read balancing caveats and use cases**

The dynamic read balancing utility is not intended to be a general purpose tool. Its main use case is when the data is primarily static and more read performance is required. See [Figure 6](#).

![Figure 6: Appropriate scenario for using dynamic read balancing](image-url)
Dynamic read balancing caveats:

- **IMPORTANT:** Dynamic read balancing cannot be used with parallel SDGs. This applies to HDP DP-volumes and SSD/flash LUNs.

- In releases prior to 11.3.3450.13 there is a known issue and dynamic read balancing may not balance properly if the capacity of the newly expanded file system is ≥ 128 TB on 32 KB HNAS file systems and ≥16 TB on 4 KB HNAS file systems.

- It is not possible to choose which SDGs are the targets for dynamic read balancing.

- Dynamic read balancing can only be run on the last expansion of a file system.
  - In HNAS releases prior to 11.2, when planning multiple expansions of a storage pool, dynamic read balancing should be run after each individual expansion as it is not possible to control which stripeset the file system expands onto.
  - In HNAS release 11.2 and newer, the `filesystem-expand --on-stripeset` HNAS CLI command can be used to control which stripeset the file system expands onto. Therefore, when planning multiple storage pool expansions, it is not necessary to run dynamic read balancing after each individual expansion. Note that using this command can limit the maximum capacity of that file system.

Dynamic read balancing usage scenarios:

1. Expanding a file system that uses all capacity in the storage pool (after expanding the storage pool) to improve read performance.

2. Expanding a file system that does not use all capacity in the storage pool (after expanding the storage pool) to improve read performance.

In the second scenario, and for HNAS deployments using versions *prior* to HNAS release 11.2, follow the procedure below to ensure that the file system is suitable for balancing.

1. Before adding more SDGs to the target storage pool, create a dummy file system using all free space on the storage pool.

2. Expand the storage pool.

3. Sufficiently expand the production file system. (See the recommendations in the File systems and chunks section.)

4. Run dynamic read balancing.

5. If more file systems reside on the storage pool that will require dynamic read balancing, repeat steps 3 and 4.

6. Delete the dummy file system. Note: If multiple file systems exist on the storage pool, do not remove the dummy file system until all required file systems have been expanded and balanced with dynamic read balancing.

---

**Figure 7:** Special case for ensuring SDG balancing
For deployments using HNAS release 11.2 or newer, when expanding a file system in any of the above scenarios, use the HNAS CLI filesystem-expand command and specify the --on-stripeset option. To determine the stripeset number, use the span-list -s CLI command. Look for the “Set” field.

<table>
<thead>
<tr>
<th>Span instance name</th>
<th>OK?</th>
<th>Free</th>
<th>Cap/GiB</th>
<th>Chunks</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storagepool</td>
<td>Yes</td>
<td>100%</td>
<td>8192</td>
<td>8192 x 1073741824</td>
<td>90%</td>
</tr>
<tr>
<td>Set 0: 4 x 1024GiB = 4096GiB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD 20 (rack '93050290', SD '0002')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD 21 (rack '93050290', SD '0003')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD 22 (rack '93050290', SD '0004')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD 23 (rack '93050290', SD '0005')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set 1: 4 x 1034GiB = 4096GiB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD 30 (rack '93050290', SD '0043')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD 31 (rack '93050290', SD '0044')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD 32 (rack '93050290', SD '0045')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD 33 (rack '93050290', SD '0046')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Large deletes of file system data while dynamic read balancing is running could result in an unbalanced SDG utilization.

**Note:** When running dynamic read balancing, you may see the following warning messages in the HNAS event log. This warning is expected when dynamic read balancing is run because it temporarily disables dynamic write balancing.

Warning: assert FSA/T2_FSA_MAP/dwb_failed_to_provide_tier0_cursor_b from FsaM1 (MFB1)...

**Using dynamic read balancing**

Dynamic read balancing is a command line utility. The two commands needed to use dynamic read balancing are:

```
fs-sdg-utilization
fs-read-balancer
```

**Note:** Remember to select the correct EVS context when running these commands.

The `fs-sdg-utilization` command is used to report how the specified file system is distributed on the SDGs. In an ideal case, when expanding a file system that is near 100% utilization, you will see output similar to the following:

```
fs-sdg-utilization FS1
    SD Group: 15, 16, 17, 28
Space Allocated to FS: 1000 GB
    Space Utilization: Live: 99.315% (993 GB), Snapshot: 0.000% (0 B), Total: 99.315% (993 GB)  Existing SDG
    SD Group: 6, 7, 8, 9
Space Allocated to FS: 1000 GB
    Space Utilization: Live: 0.001% (12.2 MB), Snapshot: 0.000% (0 B), Total: 0.001% (12.2 MB)  Newly added SDG
```
The **fs-read-balancer** command is used to start, stop, and monitor the progress of the utility.

```
fs-read-balancer begin FS1
fs-read-balancer progress FS1
   11.409% (moved 56.7 GB / 497 GB)
fs-read-balancer progress FS1
   36.903% (moved 183 GB / 497 GB)
```

After completion, the SDGs should be balanced, as shown in the example output.

```
fs-sdg-utilization FS1
   SD Group: 15, 16, 17, 28
   Space Allocated to FS: 1000 GB
   Space Utilization: Live: 49.768% (498 GB), Snapshot: 0.000% (0 B),
                       Total: 49.768% (498 GB)

   SD Group: 6, 7, 8, 9
   Space Allocated to FS: 1000 GB
   Space Utilization: Live: 49.564% (496 GB), Snapshot: 0.000% (0 B),
                       Total: 49.564% (496 GB)
```

**Snapshot considerations**

The dynamic read balancing utility rebalances data by re-writing files to new disk space. Because of this, it is important to appropriately plan how the utility is used because of the implications for snapshot space usage. Any existing snapshots will grow significantly as data is moved by dynamic read balancing and have the potential to fill up the file system.

The obvious approach is to delete and/or disable snapshots before the rebalancing. However, this can impact other aspects of the environment such as:

- Tape backup
- Replication
- Performance

There are also some other options available to prevent snapshot growth problems:

1. **Allow snapshots to cycle normally**
   This is the least disruptive method, as it preserves the data protection strategy (that is, backup, replication). However, this needs to be weighed against the expected snapshot growth and whether or not there will be sufficient capacity.

2. **Delete snapshots after running dynamic read balancing.**
   This is similar to the first approach, with the advantage of space being recovered...
faster. If incremental replication or backup is being used on the file system, this will force the next job to be a full (level 0) backup or replication.

3. Run **kill-snapshots** before or after dynamic read balancing.

   This is the quickest method to purge snapshots; however, it requires that the file system be unmounted. Read the man page before running this command.

**Note:** When performing incremental backup or replication, deleting all snapshots will force the next backup/replication job to be a *full replication*.

**Dynamic read balancing timing and performance load**

This section provides reference information to be used for setting customer expectations with regard to how long it will take dynamic read balancing to balance a file system of X TBs and the expected load on the disk storage.

The following table provides reference information for elapsed balancing times based on tests conducted by GSS.

<table>
<thead>
<tr>
<th>FS Size (Block Size)</th>
<th>Client Load</th>
<th>Time</th>
<th>Data Moved (GB)</th>
<th>Data Set</th>
<th>Drive Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 TB (32 K)</td>
<td>Idle</td>
<td>103 Minutes</td>
<td>497</td>
<td>Many small files (Avg. size 30 K)</td>
<td>SATA 1 TB</td>
</tr>
<tr>
<td>1 TB (4 K)</td>
<td>Idle</td>
<td>560 minutes</td>
<td>510</td>
<td>Many small files (Avg. size 30 K)</td>
<td>SAS 450 G 15 K</td>
</tr>
<tr>
<td>1 TB (4 K)</td>
<td>Idle</td>
<td>555 minutes</td>
<td>506</td>
<td>Few large files (Avg. size 30 MB)</td>
<td>SAS 450 G 15 K</td>
</tr>
<tr>
<td>1 TB (32 K)</td>
<td>Idle</td>
<td>57 minutes</td>
<td>500</td>
<td>Few large files (Avg. size 30 MB)</td>
<td>SAS 450 G 15 K</td>
</tr>
</tbody>
</table>

The following table shows the number of operations and throughput reported by a HNAS 3200 on a file system that was being read balanced.

<table>
<thead>
<tr>
<th>FS Block Size</th>
<th>FS Load</th>
<th>Ops Read</th>
<th>Ops Write</th>
<th>Read MBs</th>
<th>Write MBs</th>
<th>Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 K</td>
<td>13%</td>
<td>8591</td>
<td>4295</td>
<td>16</td>
<td>16</td>
<td>Small files</td>
</tr>
<tr>
<td>32 K</td>
<td>7%</td>
<td>7263</td>
<td>2518</td>
<td>78</td>
<td>78</td>
<td>Large files</td>
</tr>
</tbody>
</table>

**Best Practices**

Best practices are identified throughout the document. This section discusses best practices not covered elsewhere.

**Storage Pools**

Storage pool layout is a critical design point. A storage pool laid out poorly can result in lower than expected performance and scalability. Adhering to the advice provided in this section will lead to optimal storage pool layout.
While a storage pool can be created with a single SD, this is strongly discouraged. A storage pool should always contain at least two SDs. The best practice is to have the initial stripeset configured with a minimum of four SDs to provide sufficient storage queue depth to service I/O requests.

**Note:** Do not create multiple LUNs from the same RAID group to satisfy these recommendations. Also, do not create more than the recommended number of DP-Volumes from an HDP Pool to satisfy this best practice (see the HDP best practices section for more details).

The number of SDs per stripeset should be balanced against performance and incremental expansion requirements. Larger stripesets tend to provide better performance, but are not as flexible when it is time to expand. Smaller stripesets provide better expansion granularity and flexibility, but may not perform as well as larger stripesets.

If a storage pool will never be expanded, all SDs can be configured into a single stripeset provided there are no more than 32 SDs. When more than 32 SDs are used, the SDs should be divided up evenly. For example, to create a storage pool with 48 SDs, create the initial storage pool with 24 SDs, and then expand it with the other 24.

**Note:** The maximum number of SDs per stripeset is 32.

If there are minimal performance requirements, and granular expansion is desired, two SDs per stripeset may suffice in some circumstances; however GSS would not generally advocate such a design.

When possible, storage pool stripesets should contain an even number of SDs.

**Note:** Never create a storage pool or expand a storage pool with only a single SD. It may create a significant storage bottleneck regardless of the underlying disk configuration.

### Expanding a storage pool

Expanding a storage pool, with one or more SDs, results in the creation of an additional stripeset. When SDs are added to a storage pool, they should be the same size, type, and speed as the existing SDs. This recommendation also applies to expansions of the same tier within a tiered storage pool. Storage pools should be expanded in a consistent way to ensure stable performance.

For example, Figure 8 shows two storage pools, Storage Pool 1 was expanded consistently and the other was not.
Figure 8 - Storage pool expansion example 1

Storage Pool 1 was expanded by the same number of SDs used in the initial stripeset. The performance of Storage Pool 2 may be inconsistent due to multiple expansions with fewer SDs than the initial stripeset. In addition, creation of new file systems after any of the expansions on Storage Pool 2 may result in those file systems residing on fewer SDs, which can limit performance of the new file systems.

The expansions of Storage Pool 2 may cause undesired performance when the file system(s) on the existing stripeset(s) are nearing full capacity utilization. For example:

- A storage pool has a single file system and consists of one stripeset with four SDs.
- The file system capacity utilization reaches 98% before the storage pool and then file system itself is expanded.
- If the expansion is with fewer SDs than were used in the initial stripeset, the new SD(s) can become a bottleneck as the majority of new data written to the file system will reside within the last stripeset.

Figure 9 illustrates the example.
**Note:** The previous example assumes all of the new capacity was allocated to the existing file system.

**Note:** The absolute minimum recommended expansion unit for a storage pool is two SDs.

Another scenario where inconsistent expansions can lead to undesired performance is when new file systems are created after a storage pool has been expanded. Consider a situation where all of the capacity in a storage pool has been previously allocated to other file systems and a new file system is required. If the storage pool is expanded by a single SD or with fewer SDs than the initial stripeset, the performance of the new file system may be lower than other file systems on the existing storage pool. The reason is that the new file system will reside wholly on the new stripeset with fewer SDs.

The guidelines in this section for stripesets and storage pool expansion also apply to storage pools created using SDs from an HDP pool. Following the advice and practices ensures that adequate Q-depth is provided to the underlying disk across the HNAS logical constructs. Even though HDP spreads IO to all the underlying storage in the HDP pool, sufficient Q-Depth across the stripesets is needed to achieve the best performance.

It is strongly recommended that an HNAS storage pool maintain a 1-to-1 relationship with HDP pools. Following this practice will allow for better storage pool expansion options in future HNAS releases.

Starting with HNAS 11.2, it is no longer possible to use dynamic read balancing on file systems that reside on storage pools whose underlying storage is DP Vols from an HDP pool. See the [Dynamic read balancing](#) section for more information.

**Note:** A storage pool can contain a maximum of 64 stripesets; therefore, it can only be expanded up to 63 times.
**Storage pool and EVS affinity**

When an EVS on an HNAS node mounts a file system from a storage pool, it reserves SD Q-Depth to perform IO. A storage pool may be accessed by multiple HNAS nodes. When a storage pool is shared, the per-SD Q-Depth is divided among all the nodes that have EVSs accessing it.

**Note:** The HNAS per-SD Q-Depth is set at 32, regardless of underlying RAID group size or disk type within a specific array.

For example, if an EVS on HNAS node 1 mounts a file system in storage pool 1 and an EVS on HNAS node 2 mounts a file system in storage pool 1, the Q-Depth of the SDs in storage pool 1 will be reduced to 16. This configuration can potentially reduce performance, especially if the workload is skewed across the two nodes. The best practice is to design where a storage pool will be shared across multiple HNAS nodes. **Figure 10** illustrates the recommended configuration.

**Figure 10 Storage pool and EVS affinity.**

**SD Superflush**

HNAS uses a setting called Superflush to control the maximum write IO size it submits to the underlying Hitachi storage subsystem. This can improve write efficiency and peak performance by performing full stripe writes. The larger Superflush IO allows HNAS to use the available Q-Depth more efficiently and the storage systems tend to deliver higher performance with larger writes. Superflush has been observed to significantly improve write performance, so the best practice is to always enable it.

The HNAS file system always writes user data to free space. For Superflush to occur, multiple free file system blocks must exist contiguously and the sum of the blocks must be equal to the Superflush size. Newly configured file systems will have large amounts of contiguous free space. As a file system ages, the amount of contiguous free space available will vary, affecting the percentage of writes that can be superflushed.

Superflush is configured manually using the HNAS CLI `sd-set` command. Superflush is also set when using HCS 7.6.1 onwards. Use the rules in the following table to configure Superflush:
<table>
<thead>
<tr>
<th>HDS Storage Type</th>
<th>Recommended Superflush Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSP Class</td>
<td>3x128</td>
</tr>
<tr>
<td>HUS100 Class</td>
<td>Nx64, where N is the number of data drives in the RAID Group, that is, RAID6 6+2, N=6. <strong>Exceptions</strong> When N &gt;9, use a value of 3x128. <strong>Note:</strong> Any RAID groups used by HNAS should be formatted with 64 KB RAID stripe chunk size</td>
</tr>
</tbody>
</table>

**Warning:** Setting Superflush to values larger than those listed above can reduce the percentage of Superflush and negatively impact performance.

**File systems**

**File system utilization**

Do not allow file systems to become 100 percent full. In general, maintain at least 10% free space on the file system. When expanding a file system, it is recommended to add enough capacity to reduce the file system utilization to 80% or lower. This will help improve performance by making it easier for HNAS to find available free space when performing writes. This also improves performance in general because there is likely to be larger sections of contiguous free space for Superflush.

**File system - Allocate on Demand (auto-expand) versus Allocate Now**

When creating an HNAS file system, there are two options: Allocate on Demand and Allocate Now. Either option can be successfully used. This section discusses the tradeoffs of each option.

The diagram in Figure 11 illustrates the Allocation on Demand (asynchronous auto-expansion) model. Whole chunks of capacity are allocated from a storage pool to a file system when initially created. The initial allocation, and automatic growth limit is predetermined. As the file system becomes utilized, more chunks are added.

**Note:** The growth limit can be modified at any time.

- Prior to HNAS release 11.2, an asynchronous auto-expansion takes enough space to ensure that the file system is no more than 80% full, and it runs whenever used space exceeds 80%.
- From HNAS release 11.2 onwards, an asynchronous auto-expansion takes enough space to ensure that the file system is no more than 80% full, but it does not run until the file system is 85% full; the effect is to perform fewer, larger expansions.
- From HNAS release 12.1 onwards, and only on HDP pools, HNAS limits the size of asynchronous allocations to roughly 32 GiB. Synchronous (allocate now) expansions are not limited, and neither are expansions on non-HDP storage.

Allocation on Demand (Asynchronous auto-expansion): Chunks from the storage pool are allocated as required.
Pro
- Allows for more flexibility if file system capacity requirements are not well understood at creation or if capacity requirements change.
- Chunks are only used as required. Prevents possible over allocation of capacity that cannot easily be undone.
- The storage pool has available space to create future file systems.
- HNAS file system thin provisioning presents the growth limit (capacity) to the clients, not the actual size.
- It allows the chunk allocator to take advantage of new information -- for example, if one deletes file system A while file system B is still auto-expanding, the chunk allocator can back-fill previously used chunks by including them in file system B to balance load optimally across the stripesets.
- When using HDP in HNAS release 12.1 and onwards, small expansions will complete faster compared to larger expansions.

Con
- In storage pools with multiple stripesets, new expansions may not use chunks from other stripesets in the storage pool.

![Small Allocation (at creation)](image)

**Figure 11 - Allocate on Demand**

Allocate Now (Synchronous auto-expansion): All chunks from the storage pool are allocated on file system creation, that is, all capacity is allocated up front.

Pro
- The file system can still be expanded (provided there are unused chunks in the storage pool).

Con
- Once chunks are allocated, they cannot be reclaimed without first deleting the file system.
Storage Pool considerations for Allocate on Demand and Allocate Now

When using Allocate on Demand, ensuring the initial allocation is sufficient to span all stripesets in the storage pool (if multiple are present) will allow for use of all SDs in the storage pool, which generally provides the best performance. The advice holds true for Allocate Now.

File system block size selection

4 KB file systems

4 KB file systems provide the best capacity efficiency with small files, but should only be used with 10k/15k/SSD HDDs.

This size is also best suited for small block random I/O, whereas 32 KB file systems are better suited for large block I/O.

32 KB file systems

32 KB file systems should always be used with slower, dense drives, such as 7.2k SAS HDDs.

This file system size is best suited for large block/sequential I/O.

Hitachi Dynamic Provisioning (HDP) Considerations

Note: The information in the following sections is applicable to HNAS releases up to v12.0. If using a firmware version beyond v12.0, check for a newer revision of this document as practices will evolve due to improved HDP integration in future HNAS releases.

Mandatory guidelines for HDP use with HNAS

HDP is a feature of Hitachi Unified Storage (HUS), Adaptable Modular Storage (AMS2000), and Virtual Storage Platform (VSP) class storage arrays that HNAS can leverage. HDP is supported for use with HNAS with two mandatory guidelines:

- The DP Pool must not be oversubscribed. Ideally use a dedicated DP pool for HNAS.
  - For VSP and VSP class storage subsystems, set oversubscription to 95%.
  - For HUS100 or AMS2000 series, use full capacity mode.
- DP-Volumes must never be re-sized. HNAS volumes cannot be re-sized, either online or offline. To add storage to HNAS, create more SDs, after adding additional RAID groups to the HDP Pool.
  - Do not create more than 2 DP-Volumes per RAID group added to the HDP Pool (except when using SSD or Hitachi Accelerated Flash (HAF)).

HDP oversubscription

HDP oversubscription (thin provisioning) is not currently supported with HNAS at the time of this writing. Currently, HNAS does not have required visibility into the HDP Pool to properly integrate with HDP.

HNAS expands the file system when usage reaches a set threshold or when the administrator manually performs the expansion. This allocates space from the underlying HNAS storage pool and the operation will succeed even if no space is available on the HDP Pool. However, eventually HNAS will issue a SCSI write operation to new space. If the HDP Pool is unable to allocate the physical page, a failure condition is triggered and cascades:

- The SCSI write fails
- The HNAS Pool fails
- The HNAS file systems fail

Figure 13 illustrates this situation: 64 TB of capacity is allocated to HNAS and only 20 TB of real capacity is available in the storage. When HNAS attempts to write past 20 GB to the storage and the HDP Pool has exhausted all available space, an error is received causing HNAS to take the storage pool offline.

Figure 13 Oversubscription failure condition.
HDP best practices

System drive groups
HNAS release 11.2 auto-groups DP-Volumes from the same HDP pool into a single parallel SDG.

Prior to HNAS release 11.2, the best practice was to group each SD alone in its own SDG (which is no longer applicable in HNAS release 11.2 and higher). As a result, changing the SD grouping method requires unmounting of file systems.

Note: SDs grouped into a parallel SDG can only belong to a single tier.

HDP Pool sharing
The HDP Pool should be dedicated to the Hitachi NAS cluster. Sharing the pool with block clients is strongly not recommended and strongly discouraged.

Recommended settings
- On HUS 100, and AMS2000 class storage systems, use Advanced Wide Striping Mode (AWS) and Full Capacity Mode.
- On HUS-VM/VSP storage systems, ensure System Option Mode (SOM) 917 and 896 are set to ON.
- SOM917 facilitates the balance of pages across Parity Groups rather than Pool Volumes.
  - SOM 896 ensures that pages are formatted in the background rather than at allocation.

Figure 14. HDP DP-Vol HNAS relationship.
Ratio of DP-Volumes to RAID groups

The ratio of DP Volumes to physical RAID groups is important. Undersubscribing can starve the storage and potentially reduce performance. Oversubscribing can overload the storage and potentially increase response times and reduce performance. In the HUS100 and AMS2000 series storage subsystems, whole RAID groups are assigned to an HDP Pool. In VSP class storage subsystems, LDEVs are assigned to DP-Vols and referred to as “pool volumes.” The best practice is to allocate all defined LDEVs in a RAID group to an HDP Pool.

The best practice is to maintain a ratio of 1:1 (see Figure 15), however 2:1 (DP-Vol/RG) may be used with SAS 10K drives. Note that HNAS supports up to 512 LUNs, and while creating many smaller DP-Vols may increase flexibility, it can also significantly reduce an HNAS cluster’s ability to scale to maximum storage capacity.

In some cases, a larger ratio may be required. Those cases are:

- When using Hitachi Accelerate Flash or Solid-State Drives, an 8:1 ratio provides best overall Q-Depth to support the drives maximum IOPS capability.
- When using storage-based replication on VSP class systems. In these cases, the ratio may need to be greater than 2:1 to meet LDEV size requirements for replication.

![Diagram of HNAS Storage Pool andraid groups](image)

*Figure 15. DP Volumes to RG Ratio.*