

Storage Efficiency

White Paper

With the explosive growth of computer storage requirements being experienced by most companies, IT organizations are looking for ways to provide the storage capacity their applications need without having to endlessly purchase more and more raw capacity. Especially with the escalating expenses associated with rack space, power and cooling, keeping a reign on the number of disk drives that you require is critical to managing your costs. However, you don't want to throttle your business by denying applications or users the space that they need.

IceWEB Storage Systems offer an innovative collection of technologies that dramatically reduce your physical storage requirements while still giving your servers, applications and users the capacity they need. Together these are IceWEB's Storage Efficiency features, and are comprised of thin provisioning, linked clones, in-line compression and in-line block deduplication. Let's take a look at each of these technologies in turn, and then we'll show you how effective these can be when used together to reduce your overall storage requirements.

Thin Provisioning

Thin provisioning is simple in concept, although there are a number of different implementations from different vendors. Within the IceWEB OS, thin provisioning means that when an iSCSI volume or a fibre channel LUN (we'll use "volume" from here on to represent both types of block volumes) gets presented to a server, it is presented as a fixed size drive to the server, but no space reservation is made on the array. For example, you may want to create a 500GB G:\ drive for a Windows server. A "fully provisioned" volume checks to see that there is 500GB available and reserve that much space within the array. Many arrays not only reserve the space, they "allocate" or specify which blocks are to be used for that volume. In the IceWEB Storage System, blocks of physical space within the array are only allocated when needed to accommodate a write to a volume. Thus, a thin provisioned volume simply bypasses the reservation step. If your 500GB G:\ drive only has 12GB of data written to it, only 12GB of blocks are allocated, and since there is no reservation, the remaining 482GBs are available to other volumes or snapshots as needed.

To set a volume to be thin provisioned, select "No" for the Initial Reservation field when creating the volume. Figure 1 shows how to check thin provisioning efficiency for an IceWEB volume. In the figure, VolA is configured at 100GB, but only 27.5KB is allocated within the pool.

As noted above, all thin provisioning implementations are not the same. Many systems must pre-allocate blocks, so the operation of "extending" a volume can be costly, involving

thresholds and “growth extents”. Managing these systems can be complex, and there can be a significant performance impact to using thin provisioning. Not with the IceWEB Storage System. Blocks are only allocated as needed by writes, and the only difference with thin provisioning and full provisioning is the reservation step. Thus, there is no performance penalty for using thin provisioning; the data path operations are identical.

So why not use thin provisioning for everything? There are two management caveats that are important to understand. The first is the obvious one: you can over-provision your array. In fact, you probably want to over-provision your array to get the most from thin provisioning. This means that the total “visible” capacity that you have configured for your servers exceeds the physical capacity of the array. Since typical volumes run about 40% to 70% full, it is okay if the over-provisioning is 150%, but is probably going to be a problem if it’s 400%. The consequence of over-provisioning is that you can run out of physical space, and if that happens, writes to volumes that should have enough space start to fail. Applications don’t typically like this very much, and you can have a variety of problems depending on the application. Obviously the solution here is to have a good alert system to tell you if you start running low on physical storage, and to manage how much over-provisioning you have. With some good management practices, you can save back 20% to 40% of your capacity. Best practices are to keep over-provisioning around 120% to 150% unless you know that you have many mostly empty volumes sitting around (...or a few really full volumes). Within the IceWEB Management Console, an alert is automatically generated once a pool is 80% full, and a critical alert once the capacity is at 95%. These alerts are controlled by the volume-check fault trigger, and the levels are configurable.

Key Takeaways:

1. Thin provisioning saves you capacity with no performance impact.
2. Best practice: no more than 120% to 150% over-provisioning
3. Pay close attention to alerts which inform when you are running out of space

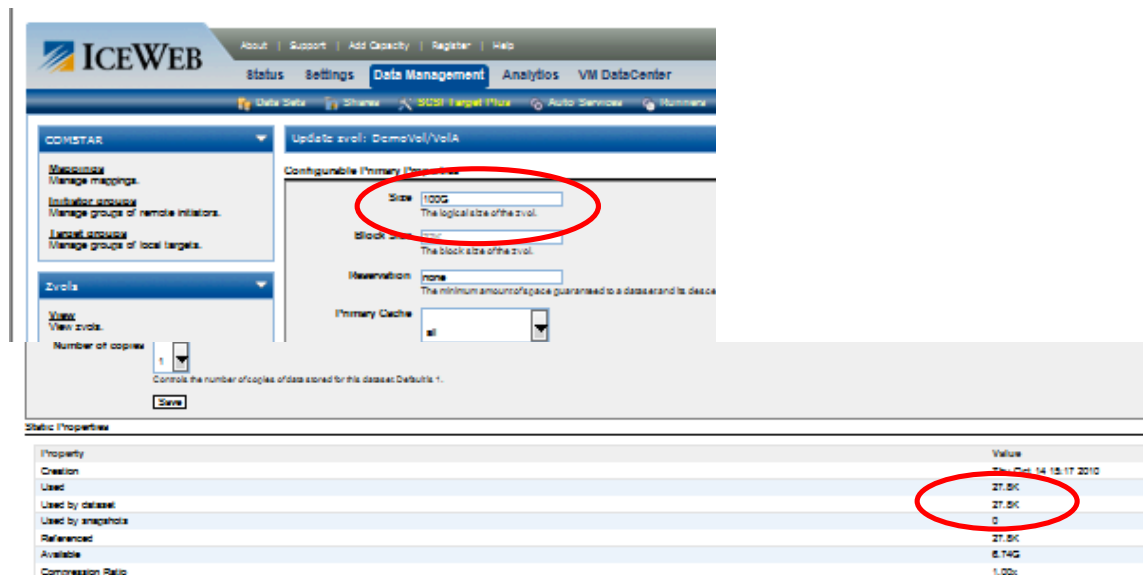


Figure 1 - Volume Capacity Screens

The other caveat to thin provisioning is that (counter to your intuition) allocated volume size (or “used” on the status screen) is not the same as the size of the file system that’s on that volume. This is because once a block of data is written, it can never be de-allocated (or “released”) from the array. Even if a file system deletes a file, as far as the array knows those blocks still have live data and so the array must preserve them. Thus, thin provisioned volumes only grow and never shrink. The only way to “reclaim” space within a volume from a bunch of deleted files is to create a new thin provisioned volume and copy all the data to it, and then delete the old one. This may lead to confusion, as you could look and see a G:\ drive that’s only 5% full, and then go to the array and see the corresponding volume at 95% full. This just means that at some time, the file system has written to 95% of the blocks. By the way, there is nothing wrong or worrisome about a volume that is 100% allocated; it’s no different than if you had just fully provisioned it in the first place. You are not “running out of space” in the volume; that is only a concern if you are running out of space in the file system that’s mounted to it.

So, should you use thin provisioning? There is no performance impact, and the only downside is administrative; you need to watch your physical capacity consumption, and also understand that file system fullness and volume allocation are not the same thing. The upside is that you certainly save yourself some capacity.

Linked Clones

Linked clones are another powerful technology that is available within the IceWEB OS to dramatically cut down on physical capacity requirements. Linked clones allow the creation of many volumes from a “master” volume, and those new volumes use the data blocks from the master copy until such time as they are modified.

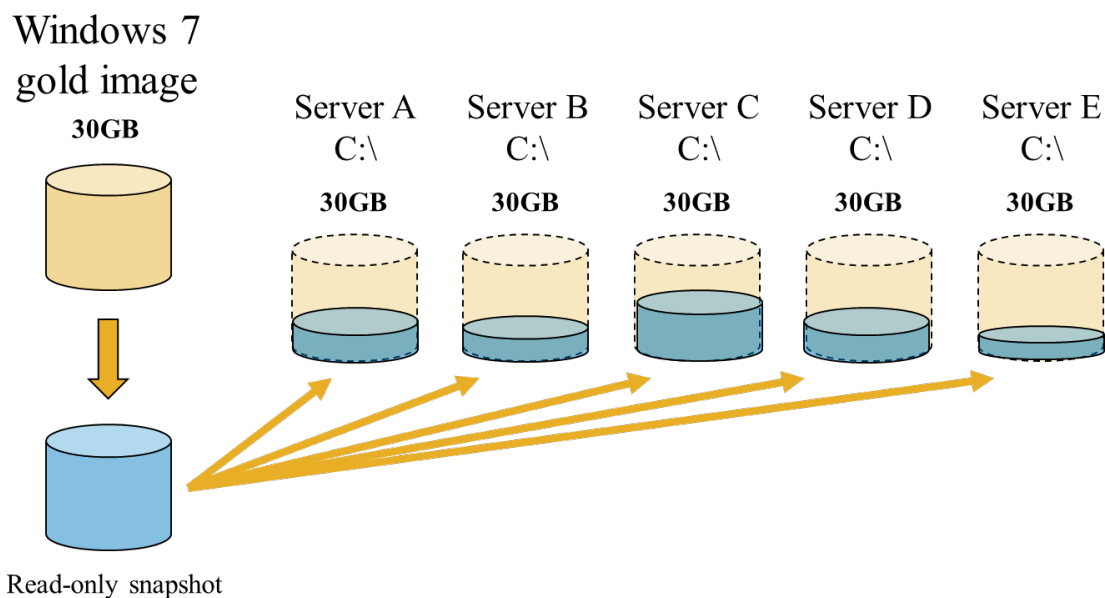


Figure 2 - Linked Clones used an OS drives for servers

Figure 2 shows an example. Say you wish to create a large number of virtual machines all running Windows 7 and a web application ... each server requires a C:\ drive which contains the OS and application binaries. Since all these C:\ drives are essentially the same, this is a good candidate for linked clones. The procedure is to create a “gold” volume and install Windows 7 and the application. You then take a snapshot of the volume to ensure that you have a “read only” version of the volume from which to create the clones. You then create as many clones as required. Each clone appears as a volume in its own right, and can be used as the C:\ drive for a server or virtual machine. All the clones reference the blocks of data in the base snapshot for reads. If a block of data is written to a particular clone, a new physical block is created and the clone references that block going forward. In this example where C:\ drives would typically have less than 20% of their blocks modified, linked clones deliver around 80% efficiency; that is, 80% of the data blocks are shared across all the volumes. If the C:\ volume is 30GB, this would save roughly 24GB of capacity per server. You can see that over 20 or 30 servers this would amount to quite a bit of capacity savings!

There is no performance impact to using linked clones. The IceWEB OS read and write data paths are identical whether the I/O is to a normal volume or a linked clone. As with thin provisioning, the only “impact” is administrative; you need to be aware of which volumes are linked clones and which are the base snapshots. Deleting the base volume for a number of clones would be a bad thing.

Be aware that the space savings are a “one time” event. The shared blocks are determined at the time of the clone creation, and all subsequent writes to the clone are forever separate. For example, if you created 20 linked clones containing an application binary, and then later updated that binary on each server, each volume clone would then have its own copy of the updated binary (no savings). So over time, the number of shared blocks will be fewer if the clone volumes are being continually updated.

This leads to a common question about linked clones: If you want to update a binary or an OS, how do you do that update once and propagate the changes out to all the clones? The procedure is to apply the updates to the “gold” volume and take a new snapshot, and then create new clones which become the new C:\ drives for the servers next time they boot. Once a server is using the new clone, the old one can be discarded. However, any changes that were written to the old C:\ drive are lost. Therefore, in this scenario it’s important to keep persistent data on a different drive. Under these conditions, it becomes easy to apply an update once, and propagate it out to a large number of servers. Each time this occurs, you “reset the base” with regard to the shared blocks between the clones, so re-cloning can be an effective way to gain back a significant amount of capacity.

In-line Compression

Compression and deduplication are two storage efficiency technologies that are used within the data path when data is written to the array. Both these data reduction technologies are included in the IceWEB OS. We will discuss compression here and deduplication in the next section.

You are probably already familiar with the concept of compression to some degree. It’s used extensively with your photos (every time you post a photo to Facebook, it is compressed), other images and videos. Every time you “zip” a file you are compressing it. The basic concept is that repeating patterns of “1s” and “0s” are reduced by using a counter. Five hundred “0s” in a row would be represented by 500(0) rather than writing all 500 “0s”. Inline

compression uses this technology to reduce the size of all the objects that are written to the array. The degree to which the data is “compressible” depends on how many repeating patterns the data has. Compression works on files of any size and composition, as contrasted with deduplication where a whole block (e.g. 8KB) must be identical to be a match. Obviously the compression algorithms used with an array must be “lossless”; in other words, we are not throwing away any of your data to make the data set smaller.

There are a number of algorithms for compression. The two main algorithms are LZJB (Lossless compression for ZFS by Jeff Bonwick) and GZIP (GNU Zip) with a “deflate” factor of 1 through 9. Generally, there is a tradeoff between the CPU consumption and the resulting compression ratio obtained; the higher the number, the better the compression, but with an increase in CPU consumption. With a storage system, you generally don’t want to risk running a very high CPU consumptive algorithm as I/O latency is almost always important. The best practice is to use LZJB, as it was designed with this use in mind. However, for deep archive volumes, you may want to turn on gzip9 to get a high compression ratio for files that won’t get read very often.

What kind of compression ratio can you expect? First of all, a compression ratio is reported on file systems within IceWEB OS. This is simply the compressed size / uncompressed size, so a 200MB file compressed to 100MB is a compression ratio of 2.0. Table 1 shows typical compression ratios you could expect against some common types of data:

Table 1 - Typical compression ratios for various applications

File shares containing Office documents	1.2 to 1.6
File shares containing virtual machine images	1.5 to 3.0
Messaging databases	1.2 to 1.8
Structured databases	1.5 to 2.1
Uncompressed images	1.9 to 3.5
Uncompressed videos	2.2 to 3.8
HTML files	1.6 to 2.0
Executables	2.0 to 2.4

The performance impact of compression is the CPU consumption of the algorithm and some additional amount of memory usage. This may lead to additional latency for each I/O. The resulting impact of compression is that the system takes slightly longer to respond to each I/O, and that the system as a whole may not be able to handle as many total I/Os (typically expressed as the I/O Per Second, or IOPS rate). Normally there is a latency penalty of around 2% to 5%, but if the system gets really busy it can be more. However, a beneficial consequence of using compression is that files are smaller as a result, so they take less I/Os to read and write. For example (Figure 3 below), a 2.0GB file with a 2.4 compression ratio results in a file that is 820KB, which works out to 148 fewer I/Os every time that file is read or written. Thus, compression can reduce the overall I/O to your system, so even though individual I/Os are taking a little longer, there are fewer of them. The IceWEB OS has built in intelligence to ensure that there is some benefit to compression before using it. If an object doesn’t “compress” by at least 12.5%, the OS writes it non-compressed. The rationale here is that you don’t want to take the compression penalty if there’s no reduction in I/Os.

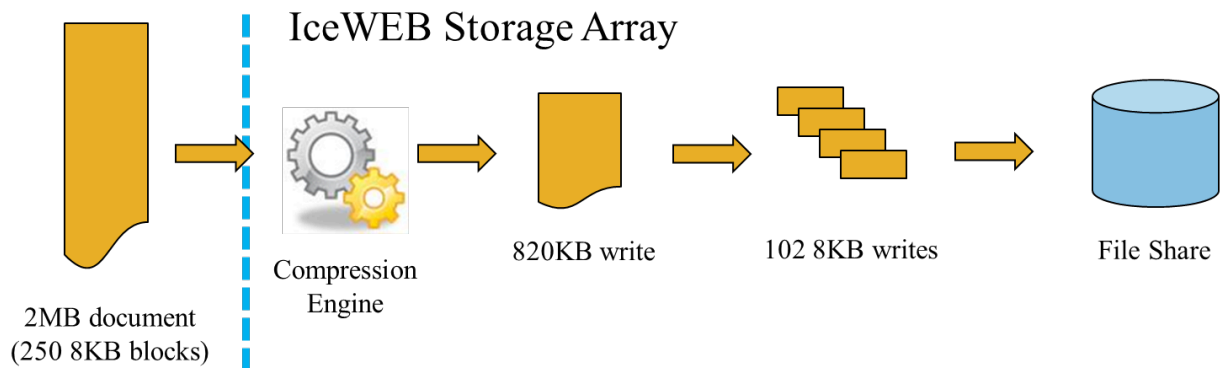


Figure 3 - Writing a 2MB file with compression enabled

When should you use compression? Compression must be set for each volume or file system within a pool. The best practice is to use it for all volumes/file systems unless you know of a good reason why it won't be useful (for example, the pool is being used to store compressed backup images). The potential latency overhead is more than offset by the reduction in I/Os, so generally compression results in a net performance gain. Unfortunately, the only thing you may see through monitoring is the increased latency times, so be aware that this is not the whole story and don't assume this means that performance is worse. If your compression ratio is anything reasonable (1.2:1 or higher) you are getting a performance benefit (as well as a capacity benefit) from using the feature.

Key Takeaways:

1. Compression can result in performance gains as well as a reduction in capacity
2. It is almost always worth using, especially with file shares
3. IceWEB Storage Systems have fast enough CPUs to offset the additional processing requirements

In-line Block Deduplication

Deduplication has become the new "buzz word" in storage, but most implementations of deduplication are within backup products and the deduplication is used to efficiently store backup sets. Typically backup deduplication is done at the backup-set level, and is effective in environments where you have many servers or desktops saving basically the same files over and over again to a backup device. For example, how many copies of excel.exe do you really need to save when backing up 500 desktops? The main value proposition for customers is the reduction in overall capacity requirements for data protection.

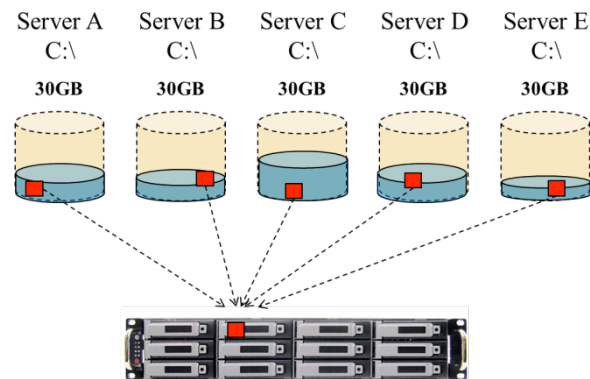


Figure 4 - Multiple volumes reference the same data block within the array

IceWEB's inline deduplication works for primary storage, and it works at the array's "block" level. For example, if data written to the array is in 8KB blocks, the IceWEB deduplication is looking for 8KB blocks of data that match, bit for bit. The technology details are straightforward; each time some data is written by an application, it is broken into a discrete set of blocks. Each block has a checksum calculated (this is always done for data integrity reasons), and, if deduplication is enabled, the system checks to see if there exists another block of data with the same checksum. If there is, we say "great, we have that block already" and increment a reference counter for the block indicating that there are multiple references to it, and then make the "pointer" for the new block point to the one that's already on disk. Now, we can simply disregard the block of data that was written and acknowledge the write.

Two questions are probably running through your head right now: How do we know that if the checksums match that the data also matches? And, what's the performance overhead for all of this?

The IceWEB OS uses a very aggressive hash algorithm for the compare (SHA256), and the chances of a hash collision (which means the checksums match, but the blocks don't) is 1 out of 10^{77} . That's a very safe number, and our entire modern computing infrastructure is based on much lower odds against data corruption than that. Within an array, it's obviously very important for a storage vendor to be able to say "your data is 100% safe", and this is the position IceWEB takes with SHA256. However, if you are ultra-conservative, you can select an additional "verify" step which actually compares the two blocks, bit-by-bit to ensure they really match. Needless to say, there is some performance impact to the "verify" option, and IceWEB's recommendation is to not use "verify" with SHA256. If you wish to use verify, we recommend using a "looser" hash called "FLETCHER" which is not as certain but much faster. FLETCHER in combination with verify is also "100% safe," but typically has more performance overhead...which brings us to the second question about performance.

Ideally, all arrays want to be "spindle limited"; meaning they are serving up I/Os as fast as the disk drives can go (with all the caches figured in as well). This means that the onus is on the storage network and the storage operating system to introduce as little latency as possible into the equation. In practice the network and software overhead is typically about 10% of the overall I/O latency time, and anything over this is regarded as a "bottleneck" and must be addressed. This is the concern about deduplication and compression; that they increase the latency to 15% or 20% and "slow down" the overall system. The extra time introduced by deduplication is the time it takes to do the compare, but is hopefully offset by the benefit of not having to do an I/O for matches. The checksum calculation is done very quickly (and as

noted above is always done regardless), and the hash compare is also quite fast. A check of the deduplication table (“DDT”) is on the order of 1 microsecond, so out of a 12 millisecond I/O operation this is not a problem. The hash compare can be longer, however, if the DDT is not all in memory. In this case, an I/O to disk would occur and the latency impact is quite significant. This can be avoided by having enough memory (RAM) to ensure that deduplication is operating at that 1 microsecond level. Where this is achieved, deduplication usually results in a net performance improvement and is directly proportional to the number of blocks that are matched.

So, the abundance of memory (and its management) is the main issue in determining the impact of deduplication. For today’s modern arrays which can scale to 100s of TB, this is a pretty big issue as the size of the DDTs is directly proportional to the amount of data. The formula for the size of the DDT is:

$$((\text{Size of pool} / \text{average block size}) * (270 \text{ bytes})) / \text{estimated dedupe ratio}$$

For example, a 2TB pool using 8KB blocks with a 20% dedupe rate is $(2,000,000 / 8 * 270) / 1.25 = 5.4\text{GB}$, so it is roughly 2.2GB of memory for each TB of capacity in the pool that’s being deduplicated. Memory is also used for caching and other things, so for a large pool (>10TB) it is hard to configure systems with enough main memory for the entire deduplication table. An option for high capacity pools is to provide a PCIe flash drive or a solid state drive (both referred to as “SSDs”) for the “overflow” needed for the DDT for larger configurations. Although not as fast as main memory, both these options allow better access speeds than standard disks. The IceWEB array should be configured with a SSD for any pool capacity over 8TB. To use the SSD for the DDT, configure the SSD as a “L2ARC” drive within the IceWEB Management Console.

What kind of benefit can you expect from deduplication? As with compression, it is very dependent on the type of data you are storing. The best case for deduplication is multiple servers (virtual machines, physical servers, or a combination) running the same OS and application set. For example, 30 VMs all running a web application dedupes very well, as most of the data is the same. Keep in mind that deduplication works at the storage pool level. Therefore, storing all the virtual machine images for 30 servers within a set of volumes within the same pool is likely to result in a very high deduplication rate, probably over 80%. The other extreme would be data that’s not at all the same between servers. For example, there’s no reason to think that an SQL database on Windows will deduplicate well with a Linux development system. The more that volumes or file systems for “like” systems can be stored in the same pool, the better it is. Table 2 lists the typical space savings for a variety of application and file types:

Table 2- Typical deduplication efficiency for various applications

Virtual desktops running the same OS / Apps	90%
Virtual machines running the same OS / Apps	70%
Virtual machines running the same OS	60%
File shares for office documents	40%
Database copies	40%
Messaging	30%
Sharepoint	30%
Databases	20%
Document archives	20%

Image / video files	10%
---------------------	-----

Thus, best practice for deduplication is to group “like” volumes / file shares into a pool, and if there’s any reason to assume that the volumes do in fact contain similar data, turn deduplication on. Monitoring consists of seeing what deduplication rate you get, and watching the latency for I/Os corresponding to each volume. If the total amount of deduplicated capacity exceeds 8TB, you need to add additional memory up to 128GB, which is the maximum for the IceWEB 3000. If the deduplicated capacity exceeds 40TB, then you must have an SSD staged as a level 2 cache.

Better Together...

Can you use both compression and deduplication together? Yes, you can enable both technologies and these results in a “let the best technology win” mechanism. Again, deduplication is best enabled at the pool level and compression is set for each object (volume or file system) within a pool. Figure 5 shows a flow chart on how the two technologies are applied to a single write. A couple things to note: Compression is done first, and there is a check for each write to see if the resulting compression ratio is at least a 12.5% reduction. Although we do endure the overhead of the compression work on the write either way, at least we don’t need to subsequently uncompress for future reads if we are not seeing a benefit. If compression is not on for that IceWEB OS object, or the compression check fails, then IceWEB OS checks for duplicate blocks. The IceWEB OS does not attempt to deduplicate compressed blocks, as this is unlikely to result in enough hits to be worth the overhead.

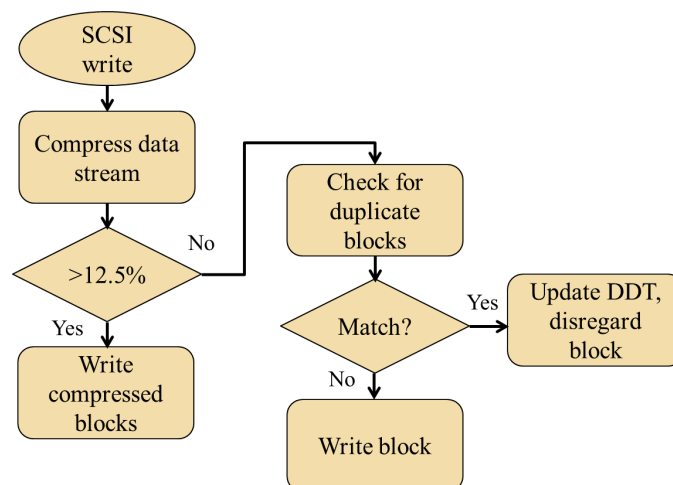


Figure 5 - Flow chart for writes with compression and deduplication enabled

Let’s look at a hypothetical environment and see how the various storage efficiency technologies work together to lower your \$/TB costs. Our environment has the following servers/applications:

- 500GB Exchange database (3 mail stores on iSCSI volumes)
- 250GB SQL Server database (2 iSCSI volumes)
- 500GB SQL Server database (2 iSCSI volumes)
- 20 virtual machines running Windows 7 and a web application (20 30GB C:\ volumes and 20 200GB D:\ volumes)

- 30 Linux development / test workstations (30 20GB C:\ volumes and 50 other volumes of various sizes)
- 1.5TB NFS file system for shared documents

All the volumes in this environment would constitute about 12.9 TB (usable) which requires about 16TB raw capacity. However, if all volumes are thin provisioned, linked clones are used for the C:\ drives, deduplication/compression for the file share and persistent data drives, and compression for the databases, we could reasonably expect to see the following capacity savings:

- Thin provisioning:
 - 312GB for the database and messaging volumes, assuming 75% utilization
 - 525GB for the file share, assuming 65% utilization
 - 3.6TB for all the user volumes, assuming 60% utilization
- Linked clones
 - 1.1TB for the C:\ drives for the Windows / Linux servers
- Deduplication
 - 450GB for deduplicated data across the user volumes and C:\ drives
- Compression
 - 462GB, assuming a compression ratio of 1.6 for the databases and messaging
 - 487GB, assuming a compression ratio of 2.0 for the NFS file share

This results in capacity savings of around 7TB, reducing the amount of raw storage requirements by around 10TB. This puts the total raw capacity at around 6TB, saving about \$20,000 (at \$2000 per raw TB). Or to put it another way, the original \$ per usable TB was \$2,480 (\$32,000 / 12.9), and with the use of the storage efficiency technologies this would be reduced to \$1,085 per usable TB (\$14,000 / 12.9). Obviously “your mileage may vary”, but even a conservative estimate puts the savings at over 50%.

Summary

The combination of thin provisioning, inline compression and inline deduplication can easily result in a storage efficiency rate of 50% or better for companies running a normal mix of business applications and/or virtual machines. The end benefit is a substantial reduction in dollars per usable TB. Within the IceWEB Storage System, your capacity costs can be reduced to under \$500 per usable TB. There is potential for some performance impact, but the IceWEB systems are configured with the proper processors, memory and cache to offset these effects and deliver overall IOPS performance per drive that is consistent with other vendors not offering these features.