

With the explosive growth of computer storage 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 hardware. Especially with the escalating expenses associated with rack space, power, cooling and management, keeping a reign on your storage is critical to managing costs. However, you also don't want to throttle your business by denying applications or users the space that they need.

The IceWEB Storage System offers an innovative collection of "storage efficiency" technologies that together dramatically reduce your physical storage requirements while still giving your servers, applications and users the capacity they need. This paper looks 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 (see Read the Fine Print below). Within the IceWEB OS, thin provisioning means that when an iSCSI volume or Fibre Channel LUN (we'll use the term "volume" from here on to represent both) gets presented to a server, it appears as a fixed "disk" that the server has available, but no space reservation is made for that volume on the array. For example, you may want to create a 500GB G: drive for a Windows server. A "fully provisioned" volume reserves 500GB within an array... a thin provisioned volume simply bypasses the reservation. So, if your 500GB G: drive only has 12GB of data written to it, only 12GB of blocks are allocated within the array and the remaining 482GBs are available to other volumes or snapshots as needed. Most block volumes are typically 60 to 75 percent full, and file shares typically run at 75 to 80 percent of quota, so you can expect to save somewhere between 20 and 40 percent of your physical capacity by using thin provisioning.

With thin provisioning, you can "over allocate" your array, effectively having more space available to your servers than you actually have in the array. The IceWEB OS keeps track of this for you, and ensures that you don't accidentally run out of physical capacity. With some applications (in particular, databases), it's appropriate to set up "arbitrarily large" drives to ensure that the application does not run out of space and avoid the dreaded "disk full" problems. All-in-all, thin provisioning is a very effective method to get a high utilization rate from your storage array.

### READ THE FINE PRINT

Not all thin provisioning implementations are the same. Many systems must pre-allocate blocks, so the operation of "extending" a volume can be costly, involving thresholds and "extents." Managing these systems is confusing, and can result in a significant performance impact. 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 during volume creation. Thus, there is no performance penalty for using thin provisioning with the IceWEB Storage System.

## 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. These are very beneficial for environments where many servers are running the same OS and/or applications. Linked clones can deliver capacity savings of up to 90% in some cases.

There is no performance impact for 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. Thus, linked clones can be used wherever they make sense.

## In-line Compression

You are probably already familiar with data compression. It's used extensively with your photos (every time you post a photo to Facebook, it's compressed), 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. So, 1500 "0s" in a row would be represented by 1500(0) rather than writing all 1500 "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" has to do with the number of repeating patterns in the data. The compression algorithms used with an array must be "lossless"; in other words we are not throwing away any of your data (unlike Facebook with your photos...).

What kind of compression ratio can you expect? First of all, a compression ratio is reported on files and file systems within the array. This ratio is simply the uncompressed size divided by compressed size. Table 1 shows the compression ratios you could expect with some common types of data:

**Table 1. Typical Compression Ratios for Common Data Types**

|   |            |
|---|------------|
| 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 |

Data must be compressed when written to the array and uncompressed when read. IceWEB Storage Systems are designed to accommodate this additional processing by incorporating high-speed processors and an innovative cache design to ensure that using compression does not add significant overhead to I/O operations.

In addition, for blocks of data that are not very compressible -- that is, the compression algorithm does not result in significant space savings -- the system automatically skips compression for that data stream.

The idea is that we don't want to waste time compressing and uncompressing a block of data that doesn't offer much benefit. For blocks of data that do compress, we can eliminate I/Os, thus not only saving you capacity, but also reducing the overall I/O load on the system resulting in an overall performance increase for your array. Given all of this, there really is no reason not to be using compression for all your data; there is minimal downside, and a very significant upside with most applications.

## In-line Block Deduplication

Deduplication has become the new "buzz word" in storage, but most implementations of deduplication is used to efficiently store backup sets within backup products. 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 engine 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 is checked to see if there is another block of data that matches it exactly within the array. If there is, we say "great, we have that block already", adjust pointers, and then disregard the block of data that was just written and acknowledge the write.

Three questions are probably running through your head right now: How do we know that the blocks match, how often do the blocks match, and what's the performance overhead for all of this?

The IceWEB OS uses a very aggressive checksum/hash algorithm for the compare and then verifies the matches. Since deduplication operates at a storage pool level, it can compare blocks between a wide variety of servers (and virtual machines). This is important, because if you have 100 servers running the same operating system and/or applications, much of the data is duplicate, and you will achieve a high rate of matches. Ultimately, the effectiveness depends on the servers and applications using that pool and common sense can be used to predict success. For example, 100 virtual machines running the same web application deduplicate very well, but an Exchange database with a Linux development server does not.

What is the "downside" to using deduplication? The overhead of deduplication is that it consumes a lot of memory in the array. If more memory is needed than what you have, it may slow down your application I/Os. To prevent this, IceWEB configures the array with a large amount of memory appropriate for most environments. When very large amounts of data are to be deduplicated, solid state drives are configured as part of the system to augment memory. On the plus side, when deduplication is successful (that is, many matches are found), the number of blocks that must be written to disk are dramatically reduced, resulting in a lower I/O load and a faster array. Thus, just like compression, for environments where deduplication is likely to be successful, deduplication can result in a performance improvement along with capacity savings.

### WHAT CAN YOU EXPECT FROM DEDUPLICATION?

The effectiveness of inline deduplication is dependent on the similarity of application and server data. The following table shows the effectiveness of deduplication against common IT environments.

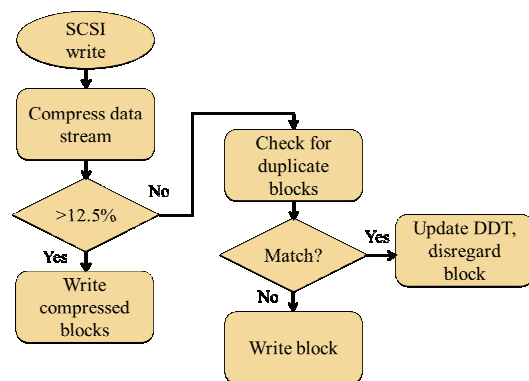
**Table 2. Effectiveness of deduplication against common IT environments**

|   |     |
|---|-----|
| 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% |

## Better Together...

Can you use both compression and deduplication together? Yes, you can enable both technologies against the same data sets. This results in a "let the best technology win" mechanism. Compression is done first, and there is a check for each write to see if the resulting compression is worth it (a greater than 12.5% benefit). If not, we then check for duplicate blocks. If a match is found the Deduplication Data Table is updated and the block is disregarded (avoiding the I/O).

**Figure 1 - Data Writes with Both Compression and Deduplication Enabled**



The combination of thin provisioning, linked clones, inline compression and inline deduplication can easily result in a storage efficiency rate of 50 percent or better for companies running a normal mix of business applications. The end benefit is a substantial reduction in dollars per usable TB. With other arrays, you may pay a significant performance penalty for using these technologies, but the IceWEB systems are configured with the proper processors, memory and cache to offset these effects and deliver the overall I/Os and throughput per drive that you would expect.