FLASH IMPLICATIONS IN ENTERPRISE STORAGE ARRAY DESIGNS

ABSTRACT

This white paper examines some common practices in enterprise storage array design and their resulting trade-offs and limitations. The goal of the paper is to help the reader understand the opportunities XtremIO found to improve enterprise storage systems by thinking with random, rather than sequential I/O in mind, and to illustrate why similar advancements simply cannot be achieved using a hard drive or hybrid SSD/hard drive array design.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE OF CONTENTS</td>
<td>3</td>
</tr>
<tr>
<td>EXECUTIVE SUMMARY</td>
<td>4</td>
</tr>
<tr>
<td>STORAGE ARRAY DESIGN PRIMER</td>
<td>5</td>
</tr>
<tr>
<td>THE I/O BLENDER</td>
<td>6</td>
</tr>
<tr>
<td>STORAGE ENLIGHTENMENT</td>
<td>7</td>
</tr>
<tr>
<td>DEDUPLICATION</td>
<td>7</td>
</tr>
<tr>
<td>SNAPSHOTS</td>
<td>9</td>
</tr>
<tr>
<td>THIN PROVISIONING</td>
<td>10</td>
</tr>
<tr>
<td>DATA PROTECTION</td>
<td>11</td>
</tr>
<tr>
<td>THE INEVITABLE CONCLUSION</td>
<td>12</td>
</tr>
<tr>
<td>HOW TO LEARN MORE</td>
<td>12</td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

Enterprise storage arrays are highly sophisticated systems that have evolved over decades to provide levels of performance, reliability, and functionality unavailable in the underlying hard disk drives upon which they're based. A significant aspect of the engineering work to make this possible is designing the array hardware and software stack to extract the most capability out of hard drives, which are physical machines with mechanically moving parts and vastly different performance profiles for sequentially (good performance) versus randomly (poor performance) accessed data. A good enterprise storage array attempts to sequentialize the workload seen by the hard drives, regardless of the native data pattern at the host level.

In recent years, Solid State Drives (SSDs) based on flash media and not bound by mechanical limitations have come to market. A single SSD is capable of random I/O that would require hundreds of hard drives to match. However, obtaining this performance from an SSD, much less in an enterprise storage array filled with them, is challenging. Existing storage arrays designed to sequentialize workloads are simply optimized along a dimension that is no longer relevant. Furthermore, since an array based on flash media is capable of approximately two orders of magnitude greater performance than an array with a comparable number of hard drives, array controller designs must be entirely rethought, or they simply become the bottleneck.

The challenge isn't just limited to performance. Modern storage arrays offer a wide variety of features such as deduplication, snapshots, clones, thin provisioning, and replication. These features are built on top of the underlying disk management engine, and are based on the same rules and limitations favoring sequential I/O. Simply substituting flash for hard drives won't break these features, but neither does it enhance them.

XtremIO has developed a new class of enterprise data storage system based entirely on flash media. XtremIO’s approach was not simply to substitute flash in an existing storage controller design or software stack, but rather to engineer an entirely new array from the ground-up to unlock flash’s full performance potential and deliver array-based capabilities that are counterintuitive when thought about in the context of current storage system features and limitations. This approach requires a 100% commitment to flash (or more accurately a storage media that can be accessed randomly without performance penalty) and XtremIO’s products use flash SSDs exclusively.

This white paper examines some common practices in enterprise storage array design and their resulting trade-offs and limitations. The goal of the paper is to help the reader understand the opportunities XtremIO found to improve enterprise storage systems by thinking with random, rather than sequential I/O in mind, and to illustrate why similar advancements simply cannot be achieved using a hard drive or hybrid SSD/hard drive array design.
STORAGE ARRAY DESIGN PRIMER

At their core, storage arrays perform a few basic functions:

- Aggregate several physical disk drives for combined performance
- Provide data protection in the event of a disk drive failure (RAID)
- Present several physical disk drives as a single logical volume to host computers
- Enable multiple hosts to share the array

Since physical disk drives perform substantially better when accessed sequentially, array designs hold incoming data in cache until such time as enough data accumulates to enable a sequential write to disk.

When multiple hosts share the array the workload is inherently randomized since each host is acting independently of the others and the array sees a mix of traffic from all hosts.

Storage arrays worked well enough for many years but have become increasingly challenged due to advancements in server design and virtualization technology that create what has come to be known in the industry as the “I/O Blender”. In short, the I/O Blender is an effect that causes high levels of workload randomization as seen by a storage array.
THE I/O BLENDER

CPUs have historically gained power through increases in transistor count and clock speed. More recently, a shift has been made to multi-core CPUs and multi-threading. This, combined with server virtualization technology, allows massive consolidation of applications onto a single physical server. The result is intensive randomization of the workload as seen by the storage array. Imagine a dual socket server with six cores per socket and two threads per core. With virtualization technology this server can easily present shared storage with a workload that intermixes twenty-four unique data streams (2 sockets x 6 cores per socket x 2 threads per core). Now imagine numerous servers on a SAN sharing that same storage array. The array’s workload very quickly becomes completely random I/O coming from hundreds or thousands of intermixed sources. This is the I/O Blender.

Storage array designs, which depend on the ability to sequentialize I/O to disk, very quickly face performance challenges in these increasingly common server environments. Common techniques to address this challenge are:

- **Increase array cache sizes:** It is not uncommon for enterprise arrays to support 1TB or more of cache. With more cache capacity there is a better chance to aggregate disparate I/O since more data can be held in cache, opportunistically waiting for enough random data to accumulate to allow sequential writes. The downside to this approach is that cache memory is very expensive and read performance is not addressed (random reads result in cache misses and random disk I/O).

- **Spread the workload across more spindles:** If the workload becomes more randomized, then more disk IOPS are needed to service it. Adding more spindles to the array yields more potential IOPS. The downsides to this approach are numerous – cost, space, power consumption, inefficient capacity utilization (having much more capacity than the applications require, simply because so many spindles are needed for performance), and the need to plan application deployments based on how their I/O requirements will map to the array.

- **Add a flash cache or flash tier:** This approach adds a higher performing media as a tier inside the array. There are numerous benefits to this approach, however array controllers designed to work with hard drives are internally optimized, both in hardware and software, to work with hard drives. Adding a flash cache or tier simply allows the performance potential of the array controller to be reached with fewer drives, but it doesn’t improve the overall capabilities of the array.

Of note, when flash is added as a cache, it will experience write cycles more frequently (data is moved in and out of the cache as it becomes “hot” or “cold”), necessitating expensive SLC flash that provides longer endurance.

The efficacy of caching and tiering can vary based on application requirements. Even if only 10% of I/Os go to spinning disk, the application will often become limited by the response time for those transactions due to serialization. Maintaining consistently high and predictable performance is not always possible in caching/tiering designs.
Attempting to resolve the I/O Blender problem with these methods helps, but what happens next year when CPU core counts double again? And again the year after that? Eventually the size of the array RAM cache and flash cache tiers need to become so large that effectively all data (excepting purely archival information) will need to reside on them. With array controller designs sized and optimized for the performance of hard drives, the full potential of flash will not be realized.

This fact alone is enough to warrant a ground-up new storage system design. But so far we have only discussed the drivers around performance. Enterprise storage systems also have many advanced features for data management. Yet just as performance can be enhanced by a ground-up flash approach, so may advanced array features.

STORAGE ENLIGHTENMENT
In this section we’ll examine four common array-based data management features: deduplication, snapshots, thin provisioning and data protection. For each feature we’ll examine how it works, how hard disk drives impact the performance of the feature, and why a simple substitution of flash media for hard drives doesn’t unlock the full potential of the feature. While we’ve chosen four examples here for illustrative purposes, the implications are common across nearly all array-based features including replication, cloning, and more.

DEDUPLICATION
Deduplication has predominately been deployed as a backup technology for two reasons. First, it is clear that multiple backup jobs run over a period of days, weeks, and months will deduplicate very well. But more importantly for this discussion, deduplication has never been suitable in primary storage applications because it has a negative impact on performance, always for reading data, and in most implementations, on writing it.

Deduplication affects access performance on primary data because it leads to logical fragmentation of the data volume. Whenever duplicate data is eliminated and replaced with a pointer to a unique data block, the data stream for the duplicate blocks is no longer sequentially laid out on disk.
But how does deduplication affect write performance? Most deduplication implementations are post-process operations. Incoming data is first written to disk without being deduplicated. Then, the array (either on a schedule or by administrator command) processes the data to deduplicate it. This processing necessitates reading the data and writing it back to disk in deduplicated form. Every I/O operation consumed during the deduplication processing is unavailable for servicing host I/O. Furthermore, the complex processing involved in deduplication is computationally intensive and slows the performance of the array controller. This has the additional drawback of requiring a large amount of capacity to "land" the data while waiting for the deduplication process to reduce it.

When flash is substituted for hard drives, performance improves as the random I/O deduplication creates is more favorably handled by SSDs. And the extra operations of the deduplication post-processing could be completed more quickly. But the post-processing operation itself requires multiple writes to disk – the initial landing of data on disk, and then the subsequent write for the deduplicated data. This consumes IOPS that would otherwise be available for host I/O, but more importantly the extra writes negatively impact flash endurance (see sidebar on previous page), which is an undesirable outcome. Other problems relating to the computational burden deduplication places on the controller, the inability of many deduplication implementations to operate globally across all array volumes or to scale across any array capacity are also not addressed by merely swapping from disk drives to flash.

DEDUPLICATION PERFORMANCE IMPACT

Once data is deduplicated, it is no longer possible to perform sequential reads as any deduplicated block requires a disk seek to the location of the stored block. In this example, what would have required a single disk seek followed by a sequential data read now requires 21 disk seeks to accomplish. With a disk seek taking roughly 15ms, a single read has just increased from 15ms to 315ms, far outside acceptable performance limits.
SNAPSHOTS

Snapshots are point-in-time images of a storage volume allowing the volume to be rolled back or replicated within the array. While some snapshot implementations literally make full copies of the source volume (split mirror snapshots), we will focus our discussion on space-efficient snapshots, which only write changes to the source volume. There are two types of space-efficient snapshots; copy-on-write and redirect-on-write.

In a copy-on-write snapshot, changes to the source volume invoke a copy operation in the array. The original block is first copied to a designated snapshot location, and then the new block is written in its place. Copy-on-write snapshots keep the source volume layout intact, but “old” blocks in the snapshot volume are heavily fragmented.

This results in two penalties. First, there are two array writes for every host write – the copy operation and the new write. This severely impacts write performance. Secondly, reads from the snapshots become highly fragmented, causing disk I/O both from the source volume and random locations in the snapshot reserve volume. This makes the snapshots unsuitable for any performance oriented application (for example, as a test/development copy of a database volume).

The other type of space-efficient snapshot is redirect-on-write. In a redirect-on-write snapshot, once the snapshot is taken, new writes are redirected to a snapshot reserve volume rather than copying the existing volume’s data. This avoids the extra write penalty of copy-on-write designs.

However, now reads from the source volume must occur from two locations; the original source volume and the snapshot reserve volume. This results in poor read performance as changes in the source volume accumulate and as more snapshots are taken since the data is no longer stored contiguously and numerous disk seeks ensue.

As with deduplication, a simple substitution of flash in place of hard drives will yield some benefits, but the full potential of flash cannot be achieved. In a copy-on-write design, there are still two writes for every new write (bad for flash endurance). A redirect-on-write design can take better advantage of flash, but challenges come in other areas. For example, mixing deduplication with snapshots is extremely complex from a metadata management perspective. Blocks of data may exist on disk, or only as pointers. They may be in a source volume or in a snapshot. Figuring out which blocks need to be kept in the face of changing data, deleting data, and creating, deleting, and cloning snapshots is non-trivial. In deduplicating storage arrays, the performance impact when snapshots are utilized can be acute.
A final area of concern in snapshot technology is cloning. A clone is a copy of a volume creating using a snapshot, but that may be subsequently altered and can diverge from the source volume. Clones are particularly challenging to implement with hard-drive based storage because changes to the source volume and the clone must both be tracked and lead to fragmentation of both volumes.

Cloning is an especially complex feature that often does not live up to performance expectations (and thus intended use cases) in disk-based arrays. As previously noted, merely substituting flash media doesn’t solve the inherent way in which snapshots or clones are implemented (e.g. copy-on-write or redirect-on-write) or the underlying metadata structures, and thus does not solve the underlying problems.

**THIN PROVISIONING**

Thin provisioning allows volumes to be created of any arbitrary size without having to pre-allocate storage space to the volume. The array dynamically allocates free space to the volume as data is written, improving storage utilization rates. Storage arrays allocate free space in fairly large chunks, typically 1MB or more, in order to preserve the ability to write sequentially into the chunks. However, most operating system and application I/O does not align perfectly to the storage system’s allocation boundaries. The result is thin provisioning “creep”, where the storage system has allocated more space to the volume than the OS or application thinks it has used. Some more advanced thin provisioning arrays have post-processing operations that periodically re-pack data to reclaim the dead space, but this results in performance issues, both from the repacking operation itself, and the resultant volume fragmentation and random I/O.

Substituting flash for hard drives provides no benefit because the underlying allocation size in the storage array hasn’t changed. Allocation sizes are not simple parameters that can be tuned. They are deeply woven into the storage system design, data structures, and layouts on disk.

**Thin Provisioning**

Problems resulting from large allocation sizes – wasted space (creep) and volume fragmentation leading to additional disk seeks during read operations.
DATA PROTECTION
The RAID algorithms developed for disk-based storage arrays protect data in the event of one or more disk drive failures. Each algorithm has a trade-off between its protection level, its performance and its capacity overhead. Consider the following comparison of typical RAID levels used in current storage systems:

<table>
<thead>
<tr>
<th>RAID Algorithm</th>
<th>Relative Read Performance</th>
<th>Relative Write Performance</th>
<th>Relative Capacity Overhead</th>
<th>Data Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAID 1 – Mirroring</td>
<td>Excellent</td>
<td>Good</td>
<td>Poor</td>
<td>Single disk failure</td>
</tr>
<tr>
<td>RAID 5 – single parity</td>
<td>Good</td>
<td>OK</td>
<td>Good</td>
<td>Single disk failure</td>
</tr>
<tr>
<td>RAID 6 – double parity</td>
<td>Good</td>
<td>Poor</td>
<td>OK</td>
<td>Single disk failure</td>
</tr>
</tbody>
</table>

Regardless of which RAID level is chosen, there is no way to achieve the best performance, the lowest capacity overhead, and the best data protection simultaneously. Much of the reason for this is rooted in the way these RAID algorithms place data on disk and the need for data to be laid out sequentially.

Using standard RAID algorithms with flash media is possible, but it doesn’t change these trade-offs. There are clear gains to be realized by rethinking data protection algorithms to leverage random-access media that would be all but impossible to replicate using hard drives. In fact, a storage system designed for flash can provide nearly optimal read performance, write performance, capacity overhead, and data protection simultaneously.

Another important consideration in the data protection design is how to handle flash “hiccups”, or periods where the flash drive does not respond as expected due to internal operations. RAID schemes created for hard drives sometimes have “on-the-fly” drive rebuilding and/or journaling to handle hard drive “hiccups”. However, these hiccup management techniques are implemented within the constraints of the core RAID algorithm and would not function properly with flash media.
THE INEVITABLE CONCLUSION

What we have shown is that while utilizing flash in existing storage systems will lead to some performance gains, the true potential of flash media cannot be realized without a ground-up array design specifically engineered for its unique random I/O capabilities. The implications for an all flash storage array include:

• It must be a scale-out design. Flash simply delivers performance levels beyond the capabilities of any scale-up (i.e. dual controller) architecture.

• The system must automatically balance itself. If balancing does not take place then adding capacity will not increase performance.

• Attempting to sequentialize workloads no longer makes sense since flash media easily handles random I/O. Rather, the unique ability of flash to perform random I/O should be exploited to provide new capabilities.

• Cache-centric architectures should be rethought since today’s data patterns are increasingly random and inherently do not cache well.

• Any storage feature that performs multiple writes to function must be completely rethought for two reasons. First, the extra writes steal available I/O operations from serving hosts. And second, with flash’s finite write cycles, extra writes must be avoided to extend the usable life of the array.

• Array features that have been implemented over time are typically functionally separate. For example, in most arrays snapshots have existed for a long time and deduplication (if available) is fairly new. The two features do not overlap or leverage each other. But there are significant benefits to be realized through having a unified metadata model for deduplication, snapshots, thin provisioning, replication, and other advanced array features.

XtremIO was founded because of a realization not just that storage arrays must be redesigned to handle the performance potential flash offers, but that practically every aspect of storage array functionality can be improved by fully exploiting flash’s random-access nature. XtremIO’s products are thus about much more than higher IOPS and lower latency. They are about delivering business value through new levels of array functionality.

HOW TO LEARN MORE

For a detailed presentation explaining XtremIO’s storage array capabilities and how it substantially improves performance, operational efficiency, ease-of-use, and total cost of ownership, please contact XtremIO at info@xtremio.com. We will schedule a private briefing in person or via web meeting. XtremIO has benefits in many environments, but is particularly effective for virtual server, virtual desktop, and database applications.