Managing Scalability in Object Storage Systems for HPC Linux Clusters

Brent Welch
Panasas, Inc
6520 Kaiser Drive
Fremont, CA 94555
Tel: 1-510-608-7770
e-mail: welch@panasas.com

Garth Gibson
Panasas, Inc
1501 Reedsdale Street, Suite 400
Pittsburgh, PA 15233
Tel: 1-412-323-6409
e-mail: garth@panasas.com

Abstract
This paper describes the performance and manageability of scalable storage systems based on Object Storage Devices (OSD). Object-based storage was invented to provide scalable performance as the storage cluster scales in size. For example, in our large file tests a 10-OSD system provided 325 MB/sec read bandwidth to 5 clients (from disk), and a 299-OSD system provided 10,334 MB/sec read bandwidth to 151 clients. This shows linear scaling of 30x speedup with 30x more client demand and 30x more storage resources. However, the system must not become more difficult to manage as it grows. Otherwise, the performance benefits can be quickly overshadowed by the administrative burden of managing the system. Instead, the storage cluster must feel like a single system image from the management perspective, even though it may be internally composed of 10's, 100's or thousands of object storage devices. For the HPC market, which is characterized as having unusually large clusters with usually small IT budgets, it is important that the storage system "just work" with relatively little administrative overhead.

1. Scale Out, not Scale Up

The high-performance computing (HPC) sector has often driven the development of new computing architectures, and has given impetus to the development of the Object Storage Architecture. The new architecture driving change today is the Linux cluster system, which is revolutionizing scientific, technical, and business computing. The invention of Beowulf clustering and the development of the Message Passing Interface (MPI) middleware allowed racks of commodity Intel PC-based systems running the Linux operating system to emulate most of the functionality of monolithic Symmetric Multi-Processing (SMP) systems. Since this can be done at less than 10% the cost of the highly-specialized, shared memory systems, the cost of scientific research dropped dramatically. Linux clusters are now the dominant computing architecture for scientific computing, and are quickly gaining traction in technical computing environments as well.

Unfortunately, storage architecture scalability in terms of performance, capacity, and manageability have not kept pace, causing systems administrators to perform tedious data movement and staging tasks on multiple standalone storage systems to get data into and out of the Linux clusters where scalable resources are available. There are two main problems that the storage systems for clusters must solve. First, they must provide shared access to ever larger amounts of data so that the applications are easier to write and storage is easier to balance with the scaling compute requirements. Second, the storage system must provide high levels of performance, in both I/O rates and data throughput, to meet the aggregated requirements of 100's, 1000's and in some cases up to 10,000's of nodes in the Linux cluster. Linux cluster administrators have attempted several approaches to meet the need for shared files and high performance, supporting multiple NFS servers or copying data to the local disks in the cluster. But to date there has not been an effective solution to match the power of the Linux compute cluster for the large data sets typically found in scientific and technical computing.

1.1 Methods of Data Sharing in Clusters

Sharing files across the Linux cluster substantially decreases the burden on both the scientist writing the programs and the system administrator trying to optimize the performance of the system and control the complexity of cluster management. There are several approaches to providing shared data to a computing cluster:

• *Network file servers*. The NFS protocol and file server hardware impose a bottleneck between the clients that share the data and the disk resources that store it. As storage needs grow, additional file servers with their own disk resources must be added, creating storage "islands" and adding complexity for users and administrators alike.
• *Block storage* via SCSI on Fiber Channel (FC) provides good performance access for a small number of disks, but block storage is private to individual hosts making it generally unscalable. Systems like GFS [1] and GPFS [2], sometimes called SAN filesystems, can provide shared FC storage for compute nodes. However, the costs of FC adapters per node, FC switching with enough ports for all cluster nodes, as well as FC administrators, can significantly increase the cost of each node in the compute cluster. Moreover, SAN filesystems are also fundamentally block-based, and the overhead of managing block-level metadata in terms of lock messaging and distribution of block-level metadata limits scalability.
• *Peer-to-peer* systems share hard drives attached to individual compute clusters. However, the complexity and cost of providing storage redundancy prohibits permanent storage of valuable data in per node disks. Moreover, storing data in per node disks introduces complexity in load balancing because some nodes have much worse I/O performance depending on where the file was written.
• A new alternative is *Object-based Storage Architectures*. The Panasas object-based storage system is built from commodity parts, including SATA drives, standard processors and memory, and Gigabit Ethernet to provide a storage system with excellent price/performance characteristics. Its high performance file system can be shared among multiple compute clusters equally and with little overhead. It can also share the same

files with legacy engineering workstations using standard NFS and CIFS protocols. The entire storage system is available through one global shared namespace in a manner reminiscent of the Andrew File System (AFS) [3].

2. Object-based Storage Architecture

The Object-based Storage Architecture utilizes data Objects, which encapsulate variable-length user data and attributes of that data [4,5,6]. The device that stores, retrieves and interprets these objects is an Object Storage Device (OSD) [7]. The combination of data and attributes allows an OSD to make decisions regarding data layout or quality of service on a per-object basis, improving flexibility and manageability. The object interface is mid-way between the read-write interface of a block device, and the high-level interface of an NFS server. The core object operations are create, delete, read, write, get attributes, and set attributes. By moving low-level storage functions into the storage device itself and accessing the device through a higher-level object interface, the Object Storage Device enables:

- Dedicated resources to block-level management,
- Intelligent space management in the storage layer allowing the OSD to make late binding decisions on the allocation of data to the storage media,
- Data-aware pre-fetching, and caching,
- A natural paradigm for scaling the capacity and performance characteristics of the storage system.

OSD-based storage systems can be created with the following characteristics:

- Robust, shared access by many clients,
- Scalable performance via an offloaded data path,
- Strong fine-grained end-to-end security.

These capabilities are highly desirable across a wide range of typical IT storage applications. They are particularly valuable for scientific and technical applications that are increasingly based on Linux cluster computing which generates high levels of concurrent I/O demand for secure, shared files.

2.1 Implementing Files using Objects

An Object-based storage system includes metadata managers that coordinate data access from multiple clients and implement POSIX filesystem semantics over the object storage. They also provide location information and implement security policies. For scalable performance, the metadata servers are “out-of-band” of the data path between clients and storage nodes. The metadata servers retain strict control of what data clients can access because the OSDs provide secure access rights enforcement with every operation [8]. Studies indicate that even in demanding small file with random access workloads, these metadata servers handle less than 10% of the work associated with file access, with the rest going to the object storage devices [9], and that load balancing metadata service can be managed by partitioning the objects [10].

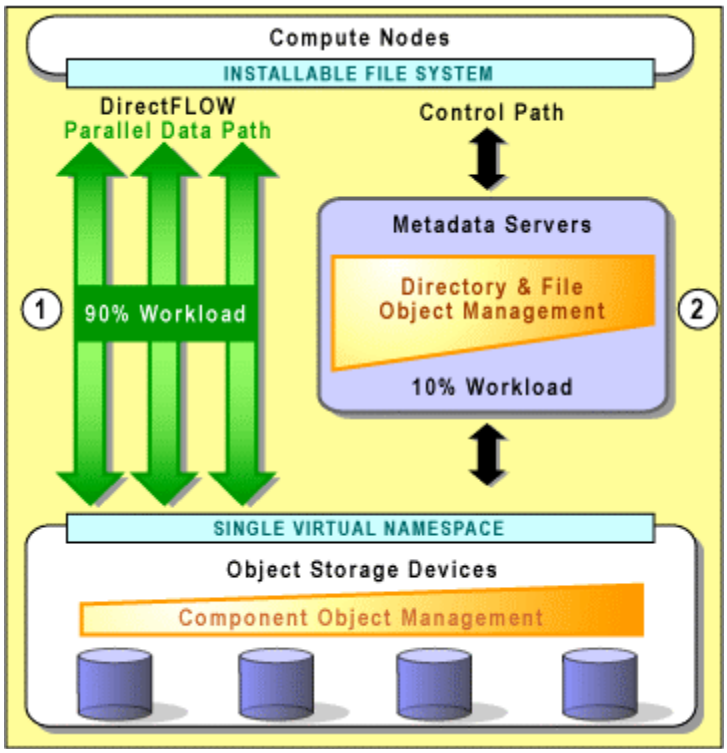

Figure 1. Object Storage Architecture

In the Panasas implementation, metadata managers are executed on server blades called DirectorBlades, and the OSDs are executed on server blades called StorageBlades. All permanent filesystem data storage is on the StorageBlades. The storage cluster can have a variable number of Director and StorageBlades depending on the workload requirements; workloads with legacy NFS and CIFS access need more DirectorBlades, while high-bandwidth, large file workloads need very few DirectorBlades.

The object interface has a core security protocol that allows safe concurrent access by multiple clients. Clients read directory objects to process filesystem pathnames. Directories map from names to object identifiers. Clients contact metadata servers to obtain location information (*maps*) and security capabilities (*caps*) that enable direct I/O access. These "maps and caps" are cached by clients so repeated access to files do not require interaction with the metadata manager. The OSDs verify security capabilities on every client access, so they enforce the policies implemented on the managers without having to know the details of POSIX or Windows ACL semantics. Our metadata managers also implement a lease-based cache consistency protocol so clients can efficiently cache file attributes and data, which provides further optimizations to file system access.

Individual files are striped across multiple StorageBlades (OSD). Striping files across objects allows high bandwidth access as well as protection from failures by using standard RAID techniques. The combination of file distribution across storage devices and multiple client access leads to very high aggregate performance to the shared storage system.

2.3 NAS Compatibility

It is also important that a new architecture such as Object-based storage be able to support NFS and CIFS compatibility for data sharing with desktops and other non-cluster computing platforms. The Panasas storage cluster does this in its DirectorBlades. DirectorBlades export standard NFS and CIFS interfaces, hiding the Object Storage Cluster from these legacy systems. Of course, a single NAS interface is a performance

bottleneck, but by deploying multiple NAS "filer head" interfaces in the storage cluster, even legacy applications see scalable performance, albeit less efficiently than through the Panasas file system protocol available on the Linux cluster nodes. Every DirectorBlade is capable of serving any file in any StorageBlade to any client. Panasas provides a global filesystem namespace, so clients only need a single mount point, which they can mount from any available DirectorBlade.

By providing a shared filesystem between the Linux computing cluster and the rest of the computing platforms, data management is dramatically simplified. For example, non-cluster hosts with tape drives can import data via multiple NFS access points in parallel, then the Linux computing cluster can access the data in place, and finally, non-cluster desktop applications can visualize and analyze the results from the computing cluster. In contrast, other approaches require explicit distribution of data to each node in the computing cluster, or management of multiple separate NAS systems.

3. Performance of Object-based Filesystems

There are four main components of the Panasas storage system: the client, the DirectorBlade, the StorageBlade, and the Shelf [11]. The client is a loadable kernel module that runs in the Linux compute nodes and implements a POSIX filesystem. It plugs into the VFS interface inside Linux. The StorageBlades have 2 SATA drives, a 1.2 GHz Pentium III processor, 512 MB memory, and 1 GE network port. The DirectorBlades have a 2.4 GHz Pentium 4 CPU, 4 GB memory, and 1 GE network port. Each shelf holds up to 11 Storage or DirectorBlades, and it has an integrated GE switch that provides up to 4 trunked GE ports out of the shelf. (A pass-through card provides 11 independent ports, but the numbers shown here use the integrated switch.) Each shelf also includes dual power supplies and a battery module, which together provide an integrated UPS function for the DirectorBlades and StorageBlades.

The bandwidth tests described below were run with 9 or 10 StorageBlades (OSD) per shelf, and in the bandwidth tests the DirectorBlades were mostly idle because most of their resources are reserved for NFS and CIFS. The goal of presenting these numbers is to give a general flavor of the scalability of the system performance. Results vary depending on the speed of the clients, the size of their I/O requests, and network topology. Typical clients in our tests have a single 2.4 GHz Pentium CPU and 1 GE network interface. Typical I/O requests in our bandwidth tests are 64 KB, and the network in our lab connects the systems and cluster under test through a high-end non-blocking Extreme Network BlackDiamond GE switch. In the bandwidth tests, files are always large enough that data must be streamed on and off the disk platters as opposed to being satisfied by cached data.

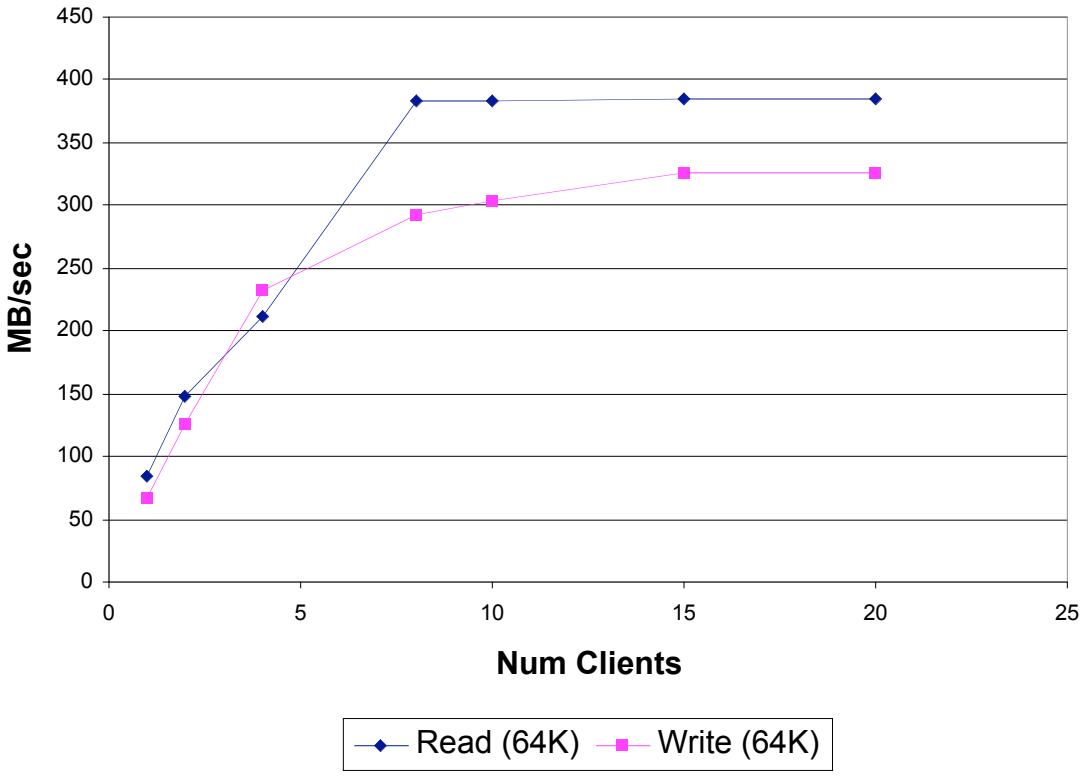

Figure 2. Bandwidth of 10 OSDs vs. the number of Clients

Figure 2 shows that aggregate bandwidth of one file per client in the same directory on a single shelf of 10 OSDs scales quite well as the number of clients increases until the shelf is providing about 380 MB/s. A single client can read from a file that is striped across 10 OSDs at about 90 MB/sec using a single GE port. As the number of clients scales up, but the storage resources remain the same, the aggregate bandwidth increases but the per-client bandwidth drops off. The less than linear scaling after about 8 clients per shelf is due to increasing load and contention at the storage devices that have to manage multiple I/O streams in parallel. There is also contention and a bottleneck in the shelf network as the data passes through the four trunked GE links providing a maximum of 4 Gb/s to each 10-OSD shelf. At saturation, each SATA drive is delivering a sustained bandwidth of just over 19 MB/sec split among the tens of clients pounding on it.

Write bandwidth follows a similar curve, with a single client writing at about 77 MB/sec and 10 clients able to write at 335 MB/sec. The write bandwidth peaks at less than the read bandwidth because the RAID engine runs at the client and so parity data flows over the network between the client and the storage nodes. For example, there is about 12% more data being written in an 8+1 RAID configuration than is reported in the bandwidth number. The advantage of moving RAID to the client is that it allows shared access to a scalable number of drives without the bottleneck of a traditional RAID controller [12]. In addition, the XOR computations RAID requires can be done efficiently using specialized

MMX instructions on the Pentium CPU, and their speed increases as client CPUs and memory systems get faster.

Figure 2 might led one to believe each shelf was a standalone 300-400 MB/s storage system. Not so. When multiple shelves are bound together by Panasas object storage software, total performance scales at 300-400 MB/s per shelf. Scaling when adding more shelves works quite well even if files are still only striped over 10 StorageBlades because the set of OSDs used to store each file are drawn from all OSDs on all shelves. The contention at any single storage device remains about the same.

Figure 3 shows several multi-shelf high bandwidth test results. For example, the test with 299 OSDs achieved 10334 MB/sec read bandwidth. There were 32 shelves and 151 clients, or about 5 clients per shelf. The per-shelf bandwidth of 322 MB/sec in this configuration is consistent with tests done with 5 clients against one shelf. (The large test used a larger application read blocksize than in the chart shown in Figure 2, so the numbers are not directly comparable.) As files are added, their data is spread across different subsets of OSDs automatically, on a per-file basis. This allows large numbers of clients to share many OSDs efficiently. Figure 3 shows that bandwidth scales quite well with the number of OSDs. In fact, because our tests are generally done with only about 5 clients per each 10-OSD shelf, Figure 2 indicates that the total bandwidth achievable from these systems can be higher. For example, the difference between the points at 116 and 118 OSDs is that the first test used 61 clients to achieve 3.1 GB/sec, while the second used 79 clients to achieve 3.9 GB/sec. That is about 50 MB/sec per client in both configurations. Finally, it is important to note that these data points were taken over time in our labs with varying configurations of clients, network topology, and software tuning, so each point is not strictly comparable. However, the overall result is that the system scales well, at about 15-20 MB/s per disk, when the number of clients is at least 25% as many as there are disks.

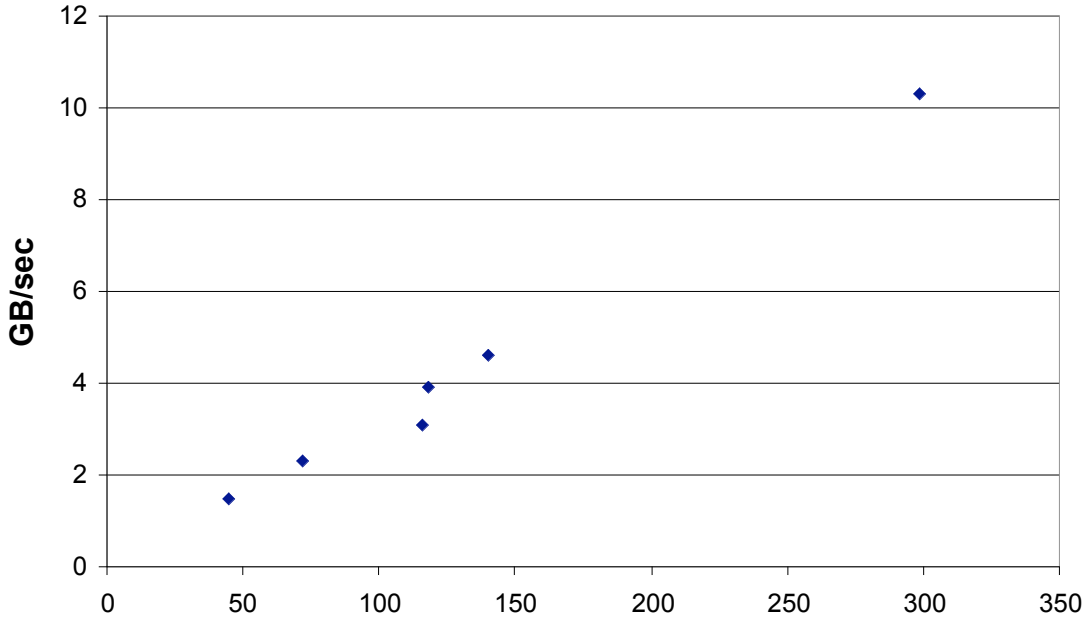

Figure 3. Scaling Aggregate Bandwidth with Number of OSDs

When files are striped very widely, contention increases because each OSD has connections to more clients. In a concurrent write test with 151 clients writing to a single, shared file striped across 198 OSDs, the write bandwidth was 2775 MB/sec. The OSDs were organized into 22 shelves with 9 OSDs each, for a per-shelf bandwidth of about 126 MB/sec.

For legacy, non-cluster computers, access is through NFS or CIFS. Because Panasas DirectorBlades offer multiple NFS servers as a single storage pool, the underlying scalability of Object Storage is made available as a scalable NAS system. Figure 4 shows two results from the industry standard SPEC SFS benchmark [13][1]. First we ran the SFS test against a 5-shelf system with 10 metadata managers and 45 OSDs, providing a single rack 90-disk system similar to the high-end of dedicated monolithic NAS filers. This delivered an excellent throughput of 50,907 ops/sec at an Overall Response Time (ORT) of 1.67 msec. The average response time as a function of load is shown in Figure 4 in the lightly shaded curve peaking at 50,000 ops/sec. ORT is a weighted average of the average response time at each of 10 load points. To show how NFS scales, we also show the results from a 30-shelf system with 60 metadata managers and 270 OSDs, which achieved 305,805 ops/sec at an ORT of 1.76 msec. While 300,000 ops/sec is much larger than any other reported SFS benchmark run to date, our real point is that a Panasas storage cluster with 6x the resources delivers 6x the workload at about the same response time profile.

[1] NFS ops/sec as measured by the SPEC benchmark. SPEC and the benchmark name SPECsfs97_R1 are registered trademarks of the Standard Performance Evaluation Corporation.

The SFS test has a uniform access requirement so each client is contacting every file server during the benchmark run. In this case each DirectorBlade is running two orthogonal functions: one is the metadata management for the object storage filesystem, the second is a client of that filesystem that is being exported via an NFS server module. The NFS server on each DirectorBlade accesses the metadata function on other directors as well as storage on all the OSDs.

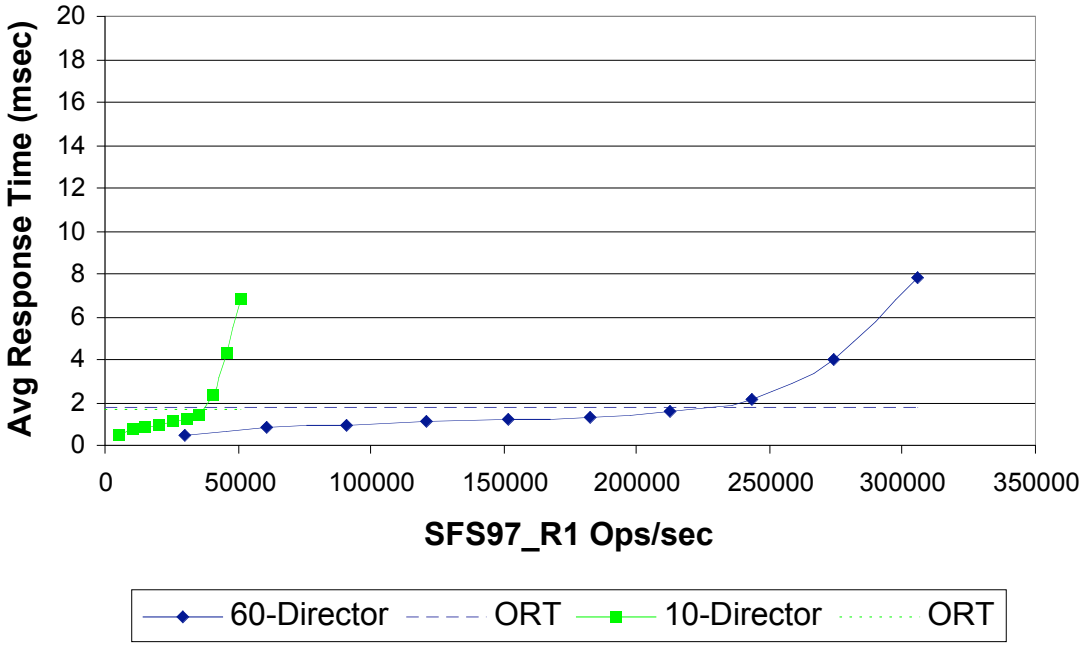

Figure 4. Results for 10- and 60-DirectorBlade Systems

4. Managing Large Scale Systems

The ideal scalable storage system is a large, seamless storage pool that grows incrementally without performance degradation and is shared uniformly by all clients of the system under a common access control scheme. As the system scales in size, however, issues arise in two general areas: traditional storage management and internal resource management. Both of these areas are affected by the distributed system implementation of the storage system itself. To external clients, the storage system should feel like one large, high-performance system with essentially no physical boundaries imposed by the implementation. Internally, the system must manage a large and growing collection of computing and storage resources and shield the administrator from chore of administering individual resources.

4.1 Traditional Storage Management

Traditional storage management issues include system configuration, monitoring system performance and capacity utilization, responding to failures, and adding and configuring hardware resources as the system grows. A scalable storage system should minimize the

burden these traditional storage management issues place on an administrator so the system can grow to very large capacities with undue operating costs.

The Panasas object-storage architecture simplifies capacity and device management by hiding the details normally associated with block devices such as LUN definition and Fiber Channel zone definitions. Instead, the filesystem is built from a collection of object storage devices (OSD) and metadata managers (Directors). The filesystem automatically stripes files across storage devices using RAID techniques to tolerate the failure of storage devices or individual objects (e.g., due to media errors). As more storage devices are added, files are striped more widely. As stripes become wider, additional parity objects are introduced to limit the size of failure domains. A unique aspect of object-based storage is that RAID configurations are configurable on a per-file basis. For example, mirroring is more efficient for small write accesses, but has high capacity overhead compared with RAID 4 or RAID 5. Ordinarily RAID parameters are managed automatically by the system. For example, the Panasas filesystem uses mirroring for directories and small files, while larger files use RAID 5. However, in simplifying management we do not want to go so far that we hurt our users' ability to optimize performance. For this reason an optional a programming interface exposes stripe width and RAID parameters so MPI IO middleware layers can create files with specific bandwidth and reliability attributes to best match application requirements.

New capacity is added simply by adding one or more new StorageBlades to the system. However, this can create a capacity imbalance among the StorageBlades, new blades less full than older blades, so the system actively rebalances capacity across StorageBlades. This is done efficiently by transparently moving component objects. For example, if files are striped across N StorageBlades, then each file is composed of N component objects, one component on each StorageBlade. When a new StorageBlade is added to the system, the system selects component objects from each of the existing StorageBlades, and moves those component objects onto the newly available StorageBlade. This reduces the capacity utilization across existing StorageBlades and fills up the new StorageBlade. The active balancer runs in the background at a low priority. The filesystem blocks the balancer from using files that are being used, and if an application happens to access a file that is currently being rebalanced, it is temporarily blocked from using the file. The application's access proceeds automatically once the balancer has finished moving the component object.

Backup and restore is obviously important for any storage system. One nice benefit of a high performance, shared storage system is that multiple backups can proceed in parallel. This helps the administrator reduce the backup window to manageable levels, even with very large systems.

Monitoring and configuration is done via a central management console that has a Web interface as well as a command line interface. Any feature accessible via the GUI is also accessible via the CLI. It usually turns out that the GUI is best for new users, offering a simple display of performance monitoring information, and a quick overview of the system. However, for large systems and more experienced administrators, it can become

tedious to configure the system one click at a time. Instead, a CLI that can be scripted (we have a TCL-based shell) can be a real time saver and an enabler for site-specific monitoring and data collection.

4.2 Internal Resource Management

Because the storage system is itself a cluster of computers working together to provide service, there are internal management issues such as resource discovery, software upgrade, failure detection, power and thermal management. These internal issues should be handled automatically by the system, yet provide the administrator with monitoring, failure reporting, and robust failure handling.

The Panasas storage system is IP-based, and its system configuration includes a block of IP addresses that is managed by an internal DHCP service. This runs on an alternate port so it will not conflict with the customer's existing DHCP infrastructure. The Panasas DHCP protocol is extended with additional information about device serial numbers, types (Storage or Director), software revision level, and physical location (shelf and slot). As blades boot up the system discovers their type, location, and software version and automatically builds its configuration database. StorageBlades are automatically added to the storage pool as they come on-line, so provisioning a running system just requires physical addition of StorageBlades.

The external view of the system is through a single DNS name. Clients mount the file system from this single name, and their filesystem accesses are automatically directed to the appropriate DirectorBlade as they access different directories. Of course, I/O access goes directly between clients and StorageBlades using the maps they get from DirectBlades during access control checks. In high availability configurations, the system DNS name is mapped to a set of IP addresses, and clients can contact any of these addresses to mount the filesystem. For NFS and CIFS load balancing, Panasas provides a delegated DNS name server to distribute legacy clients across DirectorBlades.

Software versions are maintained uniformly across the storage cluster to avoid awkward compatibility issues. Each blade boots from its own drive to avoid massive congestion on a netboot server during system startup. Software upgrade is achieved with a two-phase installation operation where all blades install a new version and reach a commit point in the first phase. Only if all blades are ready does the system commit to the new version and restart. Filesystem clients pause for the duration of the commit and restart, which is about 5 minutes regardless of the size of the storage cluster. When new blades are added to a system they are checked for hardware compatibility, and they are automatically upgraded to the same version as the rest of the system.

Power and thermal management is important for any high-density cluster installation. The Panasas blades are housed in a shelf that has dual power supplies and a battery module that together provide an integrated UPS. The UPS protects the blades against power surges and brown outs. If AC power is lost completely, the blades are signaled and use battery power to write out cached data and safely shutdown the system. By providing

an integrated UPS and power management in every shelf, the system can be aggressive about caching data in main memory without burdening the administrator with building a foolproof data center-scale UPS and scaling it up as the system grows. The StorageBlades accumulate data and metadata updates and periodically flush these in a log-like fashion. This lets the system optimize disk arm seeks and provide very high write throughput even during shared workloads. Thermal monitoring is also integrated, and the system will proactively take itself offline if external temperatures rise and cause blades to overheat. The system is able to differentiate between power or thermal failures and disk or blade failures so it can respond appropriately.

5. Conclusion

The ability of storage systems built on the Object Storage Architecture to scale capacity and performance addresses a key requirement for HPC Linux clusters. Panasas' Object-based storage cluster demonstrates scalability with 32-shelf systems providing 30x the bandwidth of a single shelf, and 30 shelf NAS benchmarks providing 6x the throughput of 5-shelf runs of the same benchmark.

While we want performance and capacity to grow linearly as resources are added to a storage cluster, we do not want administrator effort to grow anywhere near linearly. Object Storage Architectures are designed to abstract physical limitations, making virtualization easier to provide, so that larger systems can be managed with little more effort than small systems. Panasas object-based storage clusters use distributed intelligence, a single namespace interface, file-level striping and RAID, and transparent rebalancing to realize the manageability advantages of Object-based Storage.

References

[1] Soltis, Steven R., Ruwart, Thomas M., et al. *The Global File System*, proc. of the Fifth NASA Goddard Conference on Mass Storage Systems, IEEE, 1996.

[2] Schmuck, Frank, and Haskin, Roger. *GPFS: A Shared-Disk File System for Large Computing Clusters*. Proc First USENIX conf. on File and Storage Technologies (FAST02), Montery, CA Jan 2002.

[3] Howard, J.H., Kazar, M.L., Menees, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N., West, M.J, *Scale and Performance in a Distributed File System*, ACM Transactions on Computer Systems, Feb. 1988, Vol. 6, No. 1, pp. 51-81.

[4] Gibson, G. A., et. al., *A Cost-Effective, High-Bandwidth Storage Architecture*, 8th ASPLOS, 1998.

[5] Azagury, A., Dreizin, V., Factor, M., Henis, E., Naor, D., Rinetzky, N., Satran, J., Tavory, A., Yerushalmi, L., *Towards an Object Store*, IBM Storage Systems Technology Workshop, November 2002.

[6] *Lustre: A Scalable, High Performance File System*, Cluster File System, Inc. 2003. http://www.lustre.org/docs.html

[7] Draft OSD Standard, T10 Committee, Storage Networking Industry Association (SNIA), ftp://ftp.t10.org/t10/drafts/osd/osd-r05.pdf

[8] Gobioff, Howard. *Security for a High Performance Commodity Storage Subsystem*. Carnegie Mellon PhD. Dissertation, CMU-CS-99-160, July 1999.

[9] Gibson et. al. *File Server Scaling with Network-Attached Secure Disks*, ACM SIGMETRICS, June 1997, pp 272-284

[10] Brandt, S., Xue, L., Miller, E., Long D., *Efficient Metadata Management in Large Distributed File Systems*, Twentieth IEEE/Eleventh NASA Goddard Conference on Mass Storage Systems and Technologies, April 2003.

[11] http://www.panasas.com

[12] Amiri, K., G.A. Gibson, R. Golding, *Highly Concurrent Shared Storage*, Int. Conf. On Distributed Computing Systems (ICDCS2000), April 2000.

[13] http://www.spec.org/sfs97r1/