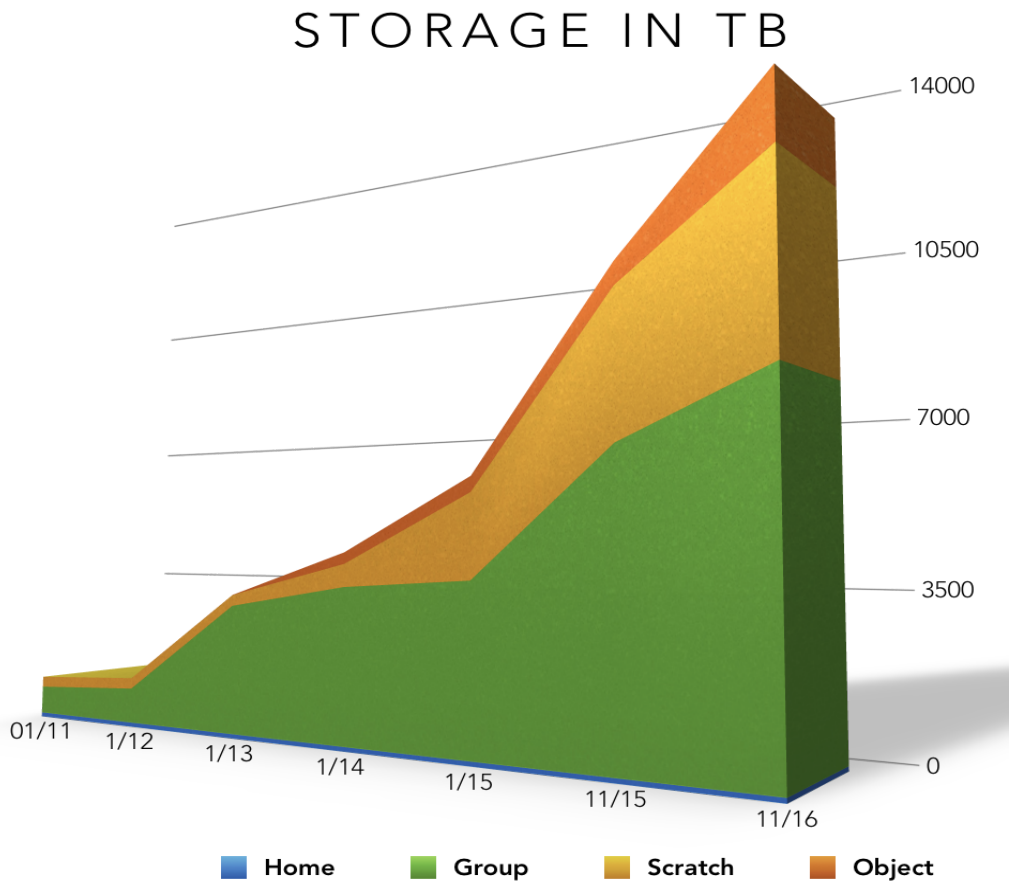
Archive Solutions at the Center for High Performance Computing

by Sam Liston (University of Utah)

The scale of the data housed at the Center for High Performance Computing (CHPC) has dramatically increased over the past 10 years. This growth is partially due to lower storage cost in terms of dollars/terabyte, and partially due to our ability to architect reliable cost-effective storage solutions. As a result of this rapid growth, the backup capacities at CHPC have not been able to scale at a proportional rate, leaving a good portion of the total data without backup. In an attempt to mitigate this shortcoming CHPC has developed a disk based archive solution. This archive, named Pando after the single organism glade of Aspen trees near Fish Lake, Utah, will provide a place for researchers to store a secondary copy of their data.



Pando has 1.02PB of capacity. It provides greater resiliency characteristics than our other storage systems and yet is offered at a lower cost; in addition it is much more accessible than a traditional backup. With Pando researchers are responsible for moving data in and out of the archive, removing some of the administrative burden in our current tape archive solution. As we

planned this solution we understood the scale of the system could eventually be vast and therefore we wanted it to have the ability to scale to an immense size. In order to do that we needed to find a solution that addressed the fundamental scaling issues that our current storage solutions, based upon traditional file systems and RAID (Redundant Array of Independent Disks) sets, have.

With the way that RAID is designed, as drive sizes increase the ability for those arrays to rebuild after a failure in a timely fashion diminishes. Similarly as maximum file system capacities increase, the time required to recover and repair a file system in the event of an error or corruption also increases, to the point where it may require many days to find and repair the errors. An alternative method to manage and organize files at massive scales is becoming necessary.

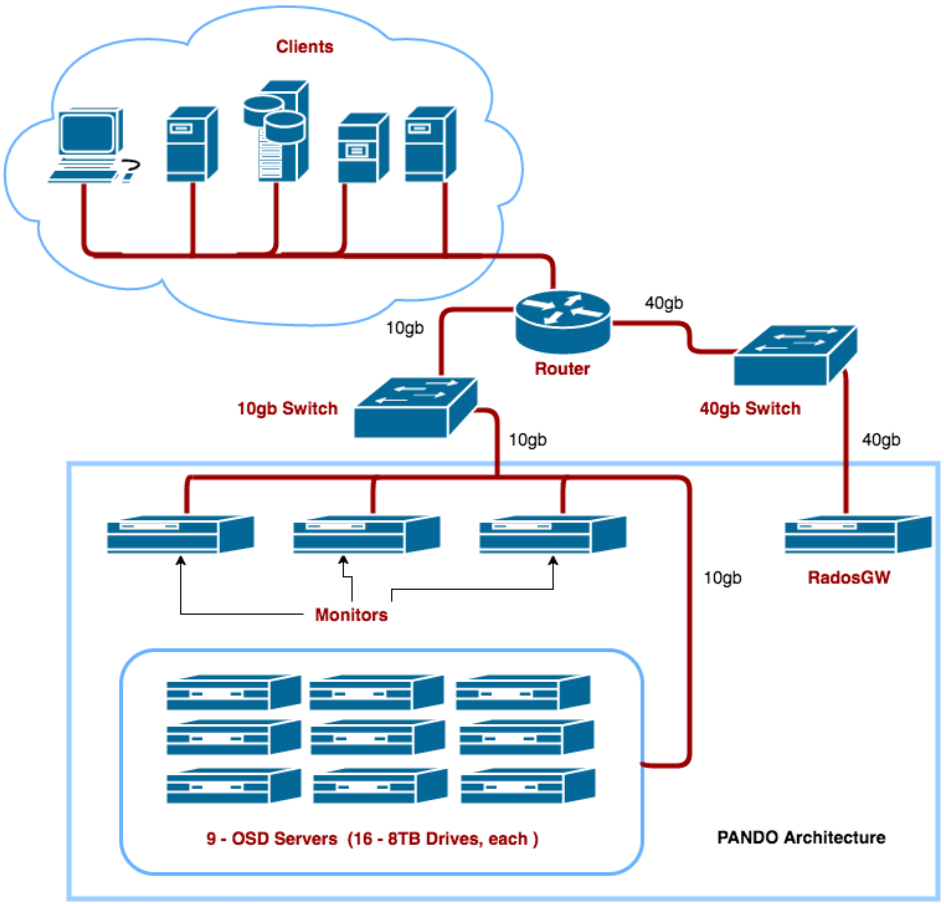
As object storage addresses several of the design shortcomings of both RAID and file systems in general. Object-based storage systems provide solutions to scaling issues inherent in very large traditional file system and RAID implementation, CHPC focused on an object-based system for our archive solution. Object storage abstracts the low-level operations away from the user or administrator, putting these operations under the control of a layer of software. Block or file level I/O are managed and facilitated by this software layer. With this abstraction layer the fundamental unit of operation becomes the object instead of the file. In object-based storage systems, both redundancy and resiliency are also moved away from the RAID controller or individual file system into the object layer. Expensive RAID controllers are no longer needed, and the worry that your very large, single file system may become corrupt and need repair, becomes moot. Every object is replicated or made redundant according to configurable parameters. One option is to have N number of replicated copies. Alternately, a particular factor of erasure coding defined as K+M, in which an object is broken up into K data chunks which are distributed across the system along with M additional resiliency chunks added for data protection and reconstruction can be chosen. In this architecture the resiliency provided by a RAID controller and its parity or mirroring protection becomes unnecessary, and in many cases becomes a hindrance to performance. Every drive in the system becomes an individual file system. If the file system on a single drive becomes corrupt, that drive is logically removed from the system, the file system is created again and then logically added back in, thereby eliminating the need for a file system check. After the freshly recreated file system is added back in, objects or object chunks are redistributed to that file system to maintain the configured level of redundancy.

In looking for a very cost-effective object-based storage solutions we discovered an open source product developed at UC - Santa Cruz called Ceph. Ceph was originally conceived as the doctoral dissertation of Sage Weil. After graduation in 2007 Sage worked to develop Ceph full-time. In 2012 Sage formed the company Inktank Storage, to provide professional services and support for Ceph. In 2014 Red Hat purchased Inktank and moved much of the development under their umbrella. Ceph continues to also be backed by a strong community, providing development, bug fixes and support.

The beauty of Ceph is in its design. First, a universal blob of object storage is created on the backend hardware. It is merely an organized mass of capacity. Second, pools are created on top of that hardware to give some structure to that blob. Inside these pools information like user keys and logs as well as the data payloads are stored. After pools are created, the backend to Ceph is ready to be expressed. There are three main methods the backend storage can be expressed: block, file, or through the Amazon S3 API. One method could be chosen or all three. It is possible to have a set of pools being expressed as block for VM access, another set of pools being expressed as files so that they can be NFS or SMB/CIFS exported to clients, and a third set of pools being expressed through S3 for archive purposes or web access.

The Pando archive project provided CHPC an opportunity to explore relatively new technologies like Ceph. The process of understanding, vetting and implementing this new technology happened over a few years. The process was quite thorough and involved building several test storage clusters, the result being a quality solution that is more resilient and redundant than our standard storage systems at a lower price per terabyte.

Ceph Hardware Configuration and the CHPC Installation



The architecture of our initial purchase is laid out in the figure above. The choices made were a reflection of the budget we had for the project. The first choice was to determine the level of redundancy; the choice was to use 6+3 erasure coding.

With 6+3 erasure coding, in order to ensure that no two chunks of data are placed on overlapping hardware requires a minimum of nine - storage servers, called Object Storage Devices or OSDs, therefore our initial implementation uses nine object storage servers.

Normally in an archive solution large drive-count storage servers are used to dramatically drive the dollars/TB very low. Storage servers with 84 and 90 drives are common. With our limited budget for this project and in an effort to meet the requirements laid out by our desired level of erasure coding we selected a smaller drive-count storage server. These nine servers contain 16 8TB drives each. There is another caveat and benefit to fewer drives per server: the Ceph community recommends having 2GB of RAM per drive in the system. With a large drive-count server, the cost for the amount of recommended RAM became prohibitive. An additional benefit to a smaller drive count is there is a smaller ratio between CPU horsepower and number of drives. Too many drives and the servers may struggle under heavy I/O load to keep up. Too few drives per server and CPU resources can be wasted. Through testing and the initial use by users we believe that our servers are slightly underused. In future expansion we would like to explore servers with 32-45 drives attached. With 6+3 erasure coding it is possible to have three whole servers full of disks fail and not experience data loss. In the event of such a failure there would be a large data storm as the remaining members immediately begin the work of redistributing the data chunks to regain the configured level of redundancy. As the drive counts per server go up so does the magnitude of this storm.

In front of the nine storage servers are three monitor nodes. The monitor nodes are the coordinators of a Ceph storage cluster. Three is minimum number of monitors for a production cluster. Beyond three, your number of monitors can scale with the size of your cluster. As monitors create and maintain quorum between the members, having an odd number of monitors is recommended. The monitor nodes keep and maintain the map of the objects in the system. This map is kept stateful and synced between all monitors. Unless the storage cluster is being expressed via CEPHFS, there is no metadata per se. When a user requests a particular object from the archive the client machine that the request was made from contacts one of the monitor node and asks for the map. With the map the client machine calculates the location of the desired object, using the CRUSH (Controlled Replication Under Scalable Hashing) algorithm. It then interacts directly with the storage server hosting that object. Once the client has the map it no longer interacts with the monitor nodes, keeping their workload minimal and pushing that workload out to a client. Without traditional metadata operations which often become the bottleneck for traditional, distributed and parallel file systems, a Ceph storage cluster can scale to extremely large numbers of objects. If orchestrating access of objects became too great for

the existing monitors, additional monitors could easily be added and the cumulative load redistributed among the members.

There are two more machines that complete the cluster. The first, an admin node provides a place to orchestra installation and configuration. The second, a RADOS (Reliable Autonomic Distributed Object Store) Gateway node used for user access to the data via a S3 interface (more below). Backend operations on the Ceph cluster are handled using the underlying RADOS software layer. RADOS can be directly interacted with, but is quite awkward and not at all user friendly. It is possible to have multiple RADOS Gateway machines, and we will look to implement this in a future expansion. In our initial load testing, it quickly became apparent that a single RADOS Gateway is the bottleneck of our current system. Under a heavy load the RADOS Gateway tops out around 5.0gb/s. At this level of throughput the backend resources are only showing around 10%-15% utilization. We theorize having 4-5 properly load balanced RADOS Gateways in front of our current Ceph cluster would allow the system to better utilize the backend resources and allow it to get around 20gb/s aggregate throughput to and from the cluster.

Ceph allows for easy expansion of the backend object storage simply by the addition of more storage. It also allows for transparent migration of data as old hardware is vacated, retired and new hardware is added and populated. In most traditional storage systems the data must be manually migrated from one generation of hardware to the next. Ceph handles this seamlessly. This feature was key in developing an archive that could be in production through a period of time that extends beyond the various vendor warranty lengths and through several generations of hardware.

We chose S3 as the method to express our archive solution. There were two main points in our reasoning that lead us to S3. First, as an archive solution the design goal was for longevity over performance, so expressing the Ceph Archive in a way that it could not be mounted on CHPC compute resources, where users could inadvertently write the output from their batch computational runs to it, became important. Second, we wanted the archive to be perceived as an island, fully separated from home, group and scratch disk spaces, in order to differentiate it from other mounted file systems. As our archive was to be a “self-serve” archive we wanted to make moving data in and out of it a deliberate process. An additional motivation towards S3 was the fact that the POSIX file method of expressing the data was not certified at a production level at the time of our implementation. Since then the file interface (CephFS) has been further developed and moved to a full production state, therefore, it is likely that in the future we will explore implementing CephFS. Implementation of the CephFS interface would require an additional server. As Ceph does not employ the use of metadata, this hardware would act as a metadata translator, offering metadata information necessary for POSIX interactions outward to the clients and Ceph calls inward to the object storage.

In order to ensures proper, balanced object disbursement and configured redundancy across the storage cluster, Ceph uses the concept of placement groups. When a Ceph cluster is first

configured, after the various pools are created, those pools are configured with a certain number of placement groups. The number of placement groups in a pool depends on the number of OSDs (object storage daemons) in the cluster. An OSD is the atomic storage piece in a Ceph cluster. Most often there is a one to one relation between number of individual drives and number of OSDs. It is possible for an OSD to relate to a RAID set of drives, but this is quite uncommon and not recommended. As the number of OSDs in a cluster increases so does the number of placement groups. To illustrate how placement groups function we will use a pool that is configured for three-way replication. Placement groups defined in this configuration will have three OSDs assigned to them, a primary and two secondaries. As an object is written the client calculates, after retrieving the map, which placement group this object should be placed in. The client then interacts directly with the primary OSD in that placement group to write out the object. Once the object is written to the primary OSD, it facilitates the replication of the object to the other two secondary OSDs. When placement groups are established data locality, particularly across configured failure domains, is considered. In the case of a failure that causes OSDs to go missing, new OSDs are assigned to placement groups that lost a member. The objects that the missing OSD contained are then replicated to the newly assigned member. Due to the relatively small capacity our initial implementation, we have only enabled failure domains at the server level, ensuring placement groups properly distribute object evenly across the storage servers. As Pando grows we will look at expanding the failure domain configurations to respect disparate racks, or rows of racks, or disparate rooms or datacenters.

Use of CHPC Pando Archive Storage

CHPC recommends three tools to use to interact with Pando.

1. **S3cmd:** S3cmd is a command-line tool for interacting with s3 storage. S3cmd moves data in and out of an s3 platform in a serial fashion, so it does not excel in data transfer performance. What s3cmd is good for is manipulating object properties in an object storage solution. With s3cmd, object ownership and permissions can be changed and sharing enabled.
2. **Rclone**: This tool developed by Nick Craig-Wood was designed to interact with a variety of cloud-based storage platforms and local storage. It acts very much like a parallel rsync for cloud resources. Rclone parallelizes flows very well. We have seen excellent throughput transferring files both in and out of Pando, as well as to public cloud platforms. Rclone excels at data transfer, but has no abilities to manipulate object ownership or permission properties.
3. **Globus:** Globus has developed an endpoint specifically to put in front of a RADOS Gateway. The Globus Ceph endpoint offers the most user-friendly interface into Pando of the three transfer tools. All the normal functions of a Globus endpoint are available in the Ceph version, including sharing.