Joining the petabyte club with direct attached storage

Andreas Haupt, Kai Leffhalm, Peter Wegner and Stephan Wiesand

Deutsches Elektronensynchrotron DESY, Platanenallee 6, 15738 Zeuthen, Germany

E-mail: stephan.wiesand@desy.de

Abstract. Our site successfully runs more than a Petabyte of online disk, using nothing but Direct Attached Storage. The bulk of this capacity is grid-enabled and served by dCache, but sizable amounts are provided by traditional AFS or modern Lustre filesystems as well. While each of these storage flavors has a different purpose, owing to their respective strengths and weaknesses for certain use cases, their instances are all built from the same universal storage bricks. These are managed using the same scale-out techniques used for compute nodes, and run the same operating system as those, thus fully leveraging the existing know-how and infrastructure. As a result, this storage is cost effective especially regarding total cost of ownership. It is also competitive in terms of aggregate performance, performance per capacity, and - due to the possibility to make use of the latest technology early - density and power efficiency. Further advantages include a high degree of flexibility and complete avoidance of vendor lock-in. Availability and reliability in practice turn out to be more than adequate for a HENP site's major tasks. We present details about this Ansatz for online storage, hardware and software used, tweaking and tuning, lessons learned, and the actual result in practice.

1. Computing environment and usage

The scientific computing environment at DESY's Zeuthen site comprises four major facilities:

- *A batch farm with 696 CPU cores* for trivially parallelizable tasks like Monte Carlo production, physics analysis, and post-processing of data from parallel computing. It is mainly used by the astroparticle physics experiments IceCube, CTA, Veritas and Magic, theorists working on phenomenology, and accelerator research.
- *A cluster with 1024 CPU cores and InfiniBand interconnect* for parallel computing, mainly used by lattice gauge theory, astroparticle theory, and accelerator research.
- *A WLCG computing element with 712 CPU cores* as part of DESY's Tier2 centre for the ATLAS, CMS, and LHCb experiments.
- *Batch nodes for the national analysis facility with 512 CPU cores* as part of a facility distributed over both DESY sites, Hamburg and Zeuthen, for german physicists participating in these LHC experiments and those working on the future international linear collider ILC.

2. Disk storage resources

Online storage for these facilities is provided in three ways:

- *An AFS cell 55 TB in size* for local users of the batch farm and the parallel cluster. In addition, there are servers for use from the national analysis facility, which has a dedicated AFS cell distributed over both DESY sites.
- *Several Lustre instances with a total size of 305 TB*, of different sizes from 40 TB to 100 TB. Most of these are for the local batch farm and the parallel cluster, but some are dedicated to the national analysis facility.
- *970 TB of dCache pools*, belonging to a smaller instance of 270 TB with tape backend and a larger one without. The latter instance is dedicated to LHC data and other datasets mainly accessed from the grid and the national analysis facility, while the smaller one is general purpose and mainly used by the local activities supported.

2.1. Accessibility
AFS is suitable for access over wide area networks, and both cells are available globally, with limitations regarding the local cell. All dCache storage is accessible globally through the SRM protocol. Part of the Lustre Storage has recently become available this way as well, by means of the Berkeley Storage Manager BeStMan.

2.2. Use cases
AFS is mostly used for home directories, workgroup space and experiment software. It is also used for storing files for which Lustre and dCache are not well suited, in particular datasets containing many small files. Lustre and dCache are both meant to be used for bulk data like experiment data sets, lattice QCD configurations and ntuples. The distinction is that dCache is regarded as managed long term storage, while Lustre merely provides scratch space.

3. Filesystem characteristics
The reason for serving disk space using three different filesystem solutions is that they have very different characteristics, and each has unique strengths and weaknesses for certain applications.

3.1. AFS
AFS data is organized in chunks called volumes. A volume is confined to a single filesystem on a server and appears as a subtree in the filesystem hierarchy on the client.

Volume location information is maintained on a set of database servers, with automatic replication and client failover. This high level metadata has a low volume and transaction rate, and constitutes no bottleneck for a site of our size. Per file metadata, like the filename, path, owner and permissions, resides within the volume on the fileserver. Consequently, there is no metadata bottleneck, provided that data can be distributed over several volumes and a sufficient number of fileservers is available. Access to the actual data also scales well in this case.

Volumes can be migrated between fileservers. This process is transparent to the clients - access isn't interrupted, and the client needn't be reconfigured.

Data has however to be distributed manually, which doesn't work very well in practice. In addition, AFS access by clients is slow compared to the other filesystems we use.

AFS supports Kerberos authentication, making it possible to serve data to untrusted clients. Software on the client can access data in AFS using the usual POSIX methods. The filesystem is not 100% POSIX compliant though.

3.2. Lustre
A Lustre instance consists of a single metadata server (MDS) and any number of object storage servers (OSSs), each serving one or more object storage targets (OSTs). Files are

distributed across OSTs automatically, making access to the data scale very well. In addition, the performance of clients and storage servers is the best of the three filesystems.

Metadata performance is limited by the fact that a single server has to handle all metadata for each and every file in the instance. File sizes have to be retrieved from the OSSs before the information can be transferred to the client. For datasets stored in small files, this causes a significant slowdown of certain operations like detailed directory listings. For very small files, as much space is required on the MDS as on the OSSs. Backing up the metadata takes longer with each additional file. Consequently, Lustre works well and very performant for large files, but is not a good solution for large datasets of small files.

While it is possible to add OSSs and OSTs to an existing instance, and also to remove empty ones, there is no way to rebalance the data or to empty an OST while guaranteeing continuous access to the affected files by the clients.

Lustre provides POSIX access on the client, optionally including local or distributed locks if so configured.

Users are authenticated by their numerical ID as transmitted by the client, thus clients must be trusted.

3.3. dCache

A dCache instance comprises a single head node storing file metadata and locations, and any number of pool nodes providing the actual storage. This layout is similar to that of Lustre, thus the characteristics are similar as well. Files are distributed over the pool nodes automatically, making the achievable aggregate throughput scale with the number of nodes. dCache can also be configured to keep several copies of a file on different nodes, which helps performance if there are hot files that are accessed by many clients. Files can be migrated to a different pool. Like the migration of AFS volumes, this is transparent to clients. Both replication and migration can be configured to happen automatically when hot spots are detected.

dCache has a number of other unique features. It can be configured with an HSM backend, thus serving as an online cache for tape storage.

A variety of access options is available, among them SRM, GridFTP, xrootd and WebDAV, besides the native access protocol dcap. Most of these provide strong authentication.

On the other hand, a client providing POSIX access to a mounted filesystem is not available today. While support for NFS 4.1 is being worked on and can already be tested, there is no production release with this feature yet, and client support is in its infancy on most platforms.

Once written, files stored in dCache cannot be modified. The only way to modify a file is to remove and rewrite it.

Like Lustre's MDS, the head node can become a bottleneck for metadata operations unless files are sufficiently large. If using an HSM backend, it becomes even more important to avoid small files.

4. Direct Attached Storage

For implementing AFS, dCache and Lustre, we utilize storage directly attached to the fileservers, pool nodes, and object storage servers, without the use of a storage area network. Today, we generally use an internal RAID controller on a PCI-Express card mounted in the server, with SAS or enterprise SATA drives either mounted in the same enclosure or in an external JBOD attached via one or more SAS cables. Compared to large storage devices distributed to servers through a storage area network, this approach has a number of significant advantages for our use cases:

- *Lower investment cost.* The total cost per Terabyte is lower by at least a factor of two, even if components of high quality are used.

- *Scalable performance.* All three filesystems distribute data over servers, and clients connect directly to the storage nodes to fetch or write data. By attaching the storage directly to the server nodes, there is no bottleneck like a central RAID controller. With every additional storage node, controller processing power and data bandwidth from disks to controllers and controllers to servers is added in proportion to the additional space.
- *Simplicity and uniformity.* The hardware used for storage nodes, and the operating system running on it, are identical to those used for compute nodes and other kinds of servers. This allows leveraging all existing know-how, tools, and methods for operating the storage. In particular, monitoring the storage simply means monitoring a few additional computers.
- *Incremental growth.* Storage can be extended in small steps, a single storage brick at a time if required. Each purchase happens at the current market price, and yields the current state of technology with respect to performance, space density, and electrical power efficiency. There is no need to obtain and operate unused capacity. With each purchase, the hardware configuration can be tailored to the actual use case.
- *No vendor lock-in.* While it is of advantage to have a homogeneous hardware base, and we prefer using hardware from only one or two vendors for extended periods, there's nothing stopping us from switching with every round of purchases. This can be important if a vendor discontinues a product line, develops a quality or engineering problem, or no longer offers competitive pricing.
- *Rapid purchase and deployment.* Since we're using commodity hardware purchased frequently, the procedures for selection of the right hardware configuration, purchase, deployment and burn-in are well established and fast. Hardware from our current preferred vendor typically arrives on site within two weeks after the decision to purchase storage.

4.1. Typical hardware configurations
There's a wide variety of possible configurations for the same basic hardware models. The most important choices are those of hard disk technology, hard disk size, interconnect to the clients, and amount of storage attached to a single node. Examples:

- *AFS fileserver for home directories.* For this application, performance and reliability are most important. Since we keep home directories small, it is no problem to choose a relatively expensive configuration with low density. A typical choice is a two height units server with twelve internal 146 GB SAS drives spinning at 15000 r.p.m, 24 GB of RAM, 2 CPUs, and two bonded Gigabit Ethernet links.
- *AFS fileserver for bulk data.* This could be the same configuration, but either with 600 GB 10k SAS drives, or even larger SATA disks, depending on the required random access performance.
- *dCache pool node.* Typically, the same 2U server, equipped with 2 TB SATA disks.
- *Lustre OSS accessible from the farm and the cluster.* A single height unit server with one or two JBODs of twelve or fifteen disks large SATA drives each. Infiniband connection to the cluster, plus two bonded GbE links for access by other clients.

Ever since it became available, we generally used RAID-6 for data arrays. We usually create a single array from all disks in an enclosure, without hot spares.

4.2. Operating system
All storage servers run Scientific Linux 5, x86-64. Like all our Linux systems, they are fully updated with security patches. Updates to the next minor release are performed after a testing period. As the backend filesystem, we use ext3 on AFS servers and XFS on dCache pool nodes. Lustre brings its own kernel with its own filesystem, actually an advanced version of ext3.

5. Performance

A single storage brick can sustain up to a few hundred Megabytes per second streaming I/O when accessed from a few fast clients. Lustre can even deliver this rate to a single client over a fast interconnect, and both Lustre and dCache easily achieve wire speed over Gigabit Ethernet. AFS single client performance is limited to about half of this, and a fileserver will not saturate two such links with multiple clients.

Throughput is generally better with fewer clients, especially with SATA drives. The most important tuning measure therefore is to make sure that sufficiently many server nodes are available and the data is distributed evenly over those. Our dCache grid storage can currently deliver an aggregate throughput of almost four Gigabytes per second, without significant load on the pool nodes. The current limit is set by the network.

5.1. Tuning

Other than this, we are applying very basic tuning only:

It used to be very important not to use the default I/O scheduler in the Linux kernel but to choose either the "noop" scheduler, leaving the work of reordering I/O requests to the RAID controller, or the "deadline" one. However, with recent kernels the default "completely fair queuing" has improved and is now about as fast as the others. It generally depends on the I/O access pattern which scheduler performs best. Since Scientific Linux 5, the scheduler can be changed for each block device on the fly.

The XFS and ext3 filesystems typically perform very similarly, with the notable exception that deletion of large files is orders of magnitude faster in XFS. It is mainly for this reason that we use XFS on the dCache pools.

It is good practice to align partitions to the stripes of the underlying RAID, and to pass information about the RAID geometry to mkfs when creating the filesystems.

Raising the read-ahead value, typically to four to 64 times the default, can help streaming read performance, especially when the number of concurrent streams is small.

6. Reliability

The reliability of direct attached storage turned out to be sufficient for our use. From our experience over the past three years, we estimate that with current technology there are some 15 minor hardware problems per year and Petabyte. Most of these incidents are hard drive replacements, which don't cause any service downtime. Almost all of the other cases are DIMMs with excessive single bit failure rates, either host or controller memory, requiring scheduled downtime. In addition, there are one to two serious incidents causing unscheduled downtime. These are either machine check exceptions on the host, resulting in a kernel panic, or RAID controller failures making the storage inaccessible and requiring a reset.

We experienced a single storage unit with serious problems. It went down repeatedly due to simultaneous failures of multiple disks, but would continue to work after a power cycle. It was demoted to read-only use after the second outage. The cause of the problem could not be isolated. After replacement of practically every part in the server and the attached JBOD, and several drives, our vendor finally replaced the complete system.

In none of these cases, even the most severe one, any data was lost.

7. Operational aspects

Although we are a relatively small team, the use of standard commodity hardware makes it possible that several members are familiar with all aspects of operating the hardware, and able to deal with any kind of failure, efficiently. Test and spare units are cheap, thus they are available. It is possible to keep a large fraction of hardware running beyond the end of its service contract. In cases of severe failure affecting important data, whole disk sets can be moved into a spare server or JBOD, resulting in very short downtimes.

8. Lessons learned
We have been running direct attached storage on commodity hardware for a decade. Along the way, we gained experience leading to the current state of smooth operation:

- *Purchase complete systems from a single vendor.* All components of a storage brick, including the hard disks, should be purchased as a whole unit from a single source responsible for the functionality of the complete system. This avoids interoperability problems, and finger-pointing in cases of serious problems. Hardware should be certified for Enterprise Linux.
- *Keep firmware up to date.* Many components in today's commodity systems run field-replaceable firmware. This includes the host's BIOS, the baseboard management controller, backplanes, enclosures - and hard drives. Like all software, this firmware has deficiencies that can be rectified as experience is gained in the field. It is our experience that applying these updates does improve the reliability of the hardware. In particular, the firmware of RAID controllers and hard drives seems to learn how to deal better with aging disks or rare failure scenarios. Applying the updates is usually simple. Yet, it does cause a significant amount of work. Some updates can be applied online, but many require a host reboot to become effective. We adopted the policy of applying firmware updates opportunistically, whenever a storage unit requires a downtime for other reasons or during site maintenance windows.
- *Scrub hard disks and check RAID parities weekly.* It is important to discover defective blocks on hard drives, and have them rewritten or remapped, as timely as possible. This functionality should be provided by the RAID controller, which should also provide a log of problems encountered and access to the drives' error counters. These should be monitored. From our experience, they are more reliable in predicting drive failures than SMART tests. Today, the majority of drive replacements at our site happen before the hardware actually fails completely.
- *Use RAID-6.* The additional parity provided by this RAID mode significantly reduces the risk of data loss. This can be important especially if a batch of disks develops problems within a short time window. In one such case, we would have lost an array at least twice had we run it at RAID-5.

9. Summary
There is no single filesystem solution fulfilling all our requirements, hence we have to deploy three of them. dCache is our workhorse, providing high aggregate throughput and global access through a variety of methods. Lustre is popular with users, very performant, and becoming more versatile with grid access. AFS serves those purposes for which the others are not suitable, in particular home directories and data sets that require smaller file sizes.

All filesystems are deployed on universal storage units built from commodity hardware with direct attached storage. The result is a suitable solution for all our use cases, with good performance, sufficient availability, and a very competitive total cost of ownership. The latter is not only due to the relatively low investment cost, but especially due to reduced overhead for administration, maintenance, monitoring and troubleshooting, because even significant amounts of storage become just an extension of the site's server park. Additional storage can be provisioned incrementally just in time, which makes it possible to use the latest technology early and thus results in space density and power efficiency comparable to those of more expensive enterprise storage solutions.