Title needs to be defined:

Collaborative Storage Clusters, Federated Storage, Global Namespaces and Metadata

Motivation

Expected Development of Storage

Computing models for the HL-LHC era anticipate a growth of storage needs of at least two orders of magnitude. . At the same time, global collaborations are building experiments (for example the SKA) which will produce comparable, or greater quantities of data on the same timescale. Apart form the classical data intensive sciences such as HEP and Astronomy also Biology produces in the future data on the same scale [EBI reference needed].
Data storage, management and preservation mandates long-term commitment beyond the lifetime of a typical research project. In addition the reliable operation of large scale data facilities show a clear economy of scale. CERN, as an example, has scaled up their storage services over xx years by a factor of yyy without adding manpower resources. Observations in the WLCG infrastructure also indicate that small facilities are more prone to availability issues than large centres [ref ATLAS ops].
Due to various reasons, such as latency considerations, data protection, network accessibility and funding sources , a single storage facility is neither a realistic nor a desirable option. In Europe and Asia the number of potential contributing centres is on the order of 10-20. These established facilities will typically use different technologies to provide storage to users and it is important that the contribution of the more than 100 small centres can be integrated into a more coarsely clustered service seamlessly. Fig [1] shows a potential scenario for a regional federation.

A distributed heterogeneous system of independent storage systems is difficult to be used efficiently by user communities and couples the application level software stacks with the provisioning technology at sites. Federating the data centres provides a logical homogeneous and consistent reliable resource for the end users. At this scale it is a necessity to be able to index and organize data by a multitude of metadata attributes. The complexity and use cases vary widely between different science communities, but all are using a subset of the technical metadata related to the namespaces of the underlying storage systems. As a result an extensible metadata management system linked to the

namespaces is necessary to use the infrastructure efficiently. This system must allow, similar to the Lambda architecture [Ref], the use of different technologies to cover data analytics tasks with widely different requirements related to latency and complexity.
Our community has gained significant experience with managing storage, user namespaces and metadata independently. While such an approach can be used, significant operational overhead due to unavoidable inconsistencies between the loosely coupled systems require frequent expert intervention. The complexity of maintaining consistency manually increases with the number of managed files/objects and storage systems involved.

The global aggregation of participating storage services into a single logical system offers an attractive approach to meeting this challenge. It would allow the maximum available storage capacity to be integrated into the system, would match the international nature of the collaborations involved, and would bring benefits such as redundancy and locality including the ability to cluster regional storage more tightly. The system must be able to integrate heterogeneous storage systems, in order to include the large number of sizeable repositories already deployed. As the ensemble must appear to users as a single system, it must offer a single namespace with predictable consistency characteristics with respect to the storage, which today's approach of operating an independent catalogue cannot bring. A similar level of deep integration with meta-data services must also be achieved to allow successful navigation and discovery within what will be an exascale storage system. In addition not all space owned by the different providers will be shared and the envisioned system has to allow for Here we describe a system designed to meet these challenges and in consequence provide a foundational multi-science storage solution.

Fig [xxx] illustrates the proposed architecture for a Virtual Storage Cloud based on technology currently in use at CERN and several other sites. Placing, access and replication policies can be defined at different level of the hierarchy.

Figure 1 Two regional storage clouds federating storage on a regional and global level. The namespaces and policies are managed at local, regional and global level. Clients support a multitude of protocols such as xrootd, HTTP(s) .

Figure 2 Showing a multi level VST network. The VST network is used to implement policies at different levels and gather state information. XROOTD and HTTP based protocols allow transparent redirection at the global, cloud and local level.
Figure 3 A potential implementation of a virtual storage cloud system using RadosFS to build a scalable namespace. The system can be extended by adding further layers of hierarchy. Note that at the leaf level storage services based on different technologies are integrated.

Global namespace and local storage consistency

Today's global data management systems, as used by the LHC experiments, typically employ loosely coupled meta-data systems, replica catalogues (providing a logical namespace), and storage services. While federation systems can be used to alleviate potential inconsistencies between file catalogues and storage they don't resolve the underlying problem and do not alleviate "dark data".

The proposed solution has more predictable consistency characteristics regarding the catalogue and the storage. It allows the implementation of advanced placement logic while preserving the autonomy of individual repositories and ensuring that the advantage of locality is fully exploited.

Role of Metadata

The explosion of data also represents an explosion of metadata, with the volume of metadata reaching levels currently seen for data itself. This will require new management and query technology to be developed. To make the problem tractable, different types of metadata can be handled in different ways, in order to adapt to their different characteristics. [Can we talk about a framework?]

Metadata can be classified according to different criteria, in particular the provenance of the data

- storage systems
    - information about the service itself, eg resource availability
    - information about the data the service holds, eg accounting
        - § In these cases, the metadata will be harvested from the participating storage and presented through a single interface
- applications and communities
    - semantic tagging, access control
    - information derived from the data itself

It is also important to distinguish between different uses of the metadata, with querying patterns spanning a continuum from "real time" interrogation, synchronous with other operations, to offline access for extended queries and analytics. It should be noted that in some domains, final results are heavily dependent on analysis of the metatdata.

Concrete Use Cases

HEP use case

The LHC experiments currently manage catalogues which track, in some cases, upwards of a billion files, and which are reaching their scalability limits. An architecturally innovative solution to meet the expected increase in data rates is required, and should exploit this necessarily disruptive opportunity to incorporate the lessons learned about integration with metadata systems and participating repositories.

Non HEP use cases

Different scientific communities bring different requirements on data and metadata, so a lower level service available to all must carefully choose the correct level of abstraction to offer, upon which application specific functionality can be built. The solution described here will allow incorporation of modular functionality, for example for offline metadata mining, [data placement?] which can be optimised for particular applications.

Architectural Considerations

In a worldwide-distributed data management system such as the WLCG several functionalities have been identified as crucial to achieve scalability and a well-defined split of responsibilities between the different sub-systems and between the different contributing partners involved in their operation:

- Tree-structured organization of storage resources which allows for increased scalability since each sub-tree can autonomously provide the scope of
  - Data tracking to guarantee data safety through replication policies
  - Accounting
  - Monitoring
  - Metadata queries
- Awareness of the physical infrastructure to provide optimized data placement and access
- An abstract storage interface to ease the integration of different existing storage solutions as well is the evolution to future systems
- Automated facilities to insure cross-component consistency and perform internal repair tasks are needed to achieve good availability with reduced operational effort (reduced human interventions)
- A common authentication and authorization scheme including not only a syntactical description of access control, but also the agreement on some (basic) semantic meaning, which is shared across all implementation components.

In particular we propose an architecture in which each individual storage system represents a leaf node in a data management tree. For example: several storage systems may be aggregated by a “site node”. A group of site nodes can be aggregated via a cloud node.

Furthermore several clouds may be aggregated by a global node. The system should be implemented in a way, that the tree height and the storage grouping of the management tree is configurable. In the resulting setup the individual storage systems are autonomous leaf nodes that store metadata and actual data while the intermediate and top-level nodes are responsible for storing and maintaining aggregated meta-data information.
To further improve the scalability it is in addition desirable to organize also the user defined data collections in a tree structure (hierarchical namespace). This does not exclude the possibility to reduce the usage to eg the commonly used S3 container model (flat namespace). While the natural entity for a leaf storage system is a file (or an object for cloud/object storage), these do not have necessarily to be visible on a global scale in all higher levels. Here the knowledge of a storage container (data collection/directory sub-tree), its size and its replication policy is usually sufficient to implement higher level data management functions. This container-based organisation of such a system would easily allow to take advantage of the fact that most containers reach read-only status already after a short period of active writing or modification.

Choice of Protocol for Data and Metadata Access

We suggest focusing on XROOT and WebDAV over HTTPS as the protocols to handle meta-data operations and to use XROOT and HTTP as the data transport protocols. The PUT/POST semantics of the HTTP protocol fits well the proposed store (data upload) and commit (meta data registration) semantics. To move data efficiently between storage systems will require that a common peer-to-peer copy operation is implemented by all major systems. Today gridFTP is a deployed standard in WLCG, XRootD is also widely used by certain experiments (e.g. ALICE) as it supports a full-featured 3rd party copy, for simplification HTTP should become the only transfer protocol required in the longer-term future. One shortcoming in WebDAV is that there is no support for multi-part PUT requests, which are needed to parallelize uploads or to resume uploads after transfer errors. The lack of this operation could be compensated using a pull architecture in which a P2P transfer is initiated close to the target storage system. The WAN access in this scenario is a (resumeable) GET request and the LAN access executes the PUT.
Since P2P functionality is neither standardized nor implemented by commercial HTTP based storage providers we would need to provide an HTTP proxy with P2P support to bridge transfers for some of the commercial storage solutions.
Additional meta-data registration and query functionality can be implemented using a REST API (tbd).

Metadata Service – Prototype Implementations

HTTPS Front-End with Object-Storage Back-End

The main focus of the proposed metadata service is horizontal scalability. This means in particular that the metadata service can be divided in a stateless server front-end and a persistent metadata back-end. Additional front-end instances can be deployed to match eg the expected number of client connections. As a back-end we propose to focus on eg the RADOS object-store. RADOS allows configuring arbitrary reliability and scalability of OOPS (object operations). The achievable OOPS are proportional to the number of object storage daemons (OSDs) deployed. The latency per object operation is given by the media and network latency. The use of flash-based storage media allows achieving latencies in the 0.1 µs range.

libradosfs

As part of an R&D project the CERN IT data & storage service group has developed *libradosfs*.

libradosfs provides:

- A hierarchical namespace implemented on top of the RADOS object storage scaling by directories (can be optimized further by directory sharding)
- Metadata API (extended attributes) per directory, directory entry and per file
- A parallel query infrastructure for sub-tree meta-data queries using simple key-value comparisons
- File-system integrity check and recovery
- A flexible user, group and tree quota accounting

libradosfs can be integrated into the front-end meta-data HTTPS servers exposing all required meta-data functionality for the data management system. As HTTPS front-end we propose the XRootD server in combination with an HTTPS protocol bridge plug-in or an embedded HTTPS server like *civetweb* or *libmicrohttpd.*

A simpler, but less scalable implementation of a metadata store could be provided by any storage system that provides extended attribute support. A small to medium sized data management namespace can already be constructed using eg a ZFS file system namespace and hence avoing the need for a more demanding RADOS deployment.

Consistency

In a system with multi-billion records the operation of a full resynchronization of meta-data can only be seen as a last resort procedure in case of major system failures. During normal operation each file movement, creation or deletion needs to first be reliably recorded (eg in an intent log). These intent logs can then be processed asynchronously by active agents at the different data management layers. These agents insure, that the information stays consistent across levels and will if necessary apply corrections.

Item for further discussion:

- Different consistency levels in one system (eventual and strong) to combine the additional requirements from Massimo (**Home-Directories Catalogue**)

with the grid scale requirements (eg trading more scalability for weaker/delayed consistency).

Prototyping

1. Russian T1s and T2s are collaborating closely and could profit from a clustered or federated approach. These sites are especially suited for participating in a prototyping activity since they plan a new deployment of resources with little history at scale.
2. CERN is in the process of clustering or federating several local storage services. This includes services that require large namespaces for the use as home directories for the large user communities and local projects. This prototype will allow a comparison with currently used solutions in the identical environment.

Appendix

Home-Directories Catalogue (Massimo and Jan AUG-2015)

This is a description of the cataloging capabilities for a AFS replacement. On the short (medium) term the present EOS catalogue is functionally adequate and there is enough room to prepare the “full solution”. The time scale for the full solution is LS2 (2018).

Functional requirements

1. The system should handle 10^{11} objects describing the files in the system
    a. AFS handles 3 10^{9} files (20% growth over the last 12 months; 35% over the last 3 years)
2. The system should provide a namespace view of the system to support filesystem-access
3. The system should support extended attributes (cfr sys and usr attributes in EOS)
4. The system should have a quota system and accounting system (space and #objects)
5. The system should have ACL system which can be integrated with the CERN LDAP infrastructure and the CERN Egroup system

Operational requirements

We assume the catalogue is instantiated in a scale-out architecture, like a “cluster” of boxes using their memory and disk to hold and serve their information (catalogue metadata). The “cluster” is self-consistent (e.g. provision for read-only or stand-by copies etc.).

1. The impact of a single-box failure should (at most) result in some part of the namespace to become “read-only” transiently
2. Adding a single box should be possible by declaring the new node and (possibly) giving hints to which part of the namespace information should be hosted (transparent intervention)
3. Removing a single box should be possible by (at most) issuing a “drain” command with a few-hours latency (transparent intervention)
4. In case of a single-box failure, the namespace can be incomplete (like missing or orphaned directories). In turn directory content should be consistent (all files/dirs present)
5. The full system restart should not take more than 10 s (system completely down); no more than 100 s with the system mainly in read-only mode); no more than 10,000 s in degraded mode (catalogue consistency checks)
6. Quota and usage information should be retrieved in less then 100 s (main queries: list all quotas; list all users with usage>90%quota; inconsistent entities)
7. At 10^{11} objects the equivalent of a full find command should run at 10 kHz or better

Ideas for the architecture

A prototype of the catalogue has been developed in DSS. The main features are:

- Extensible front-end of “stateless” gateways
  - Caching part of the metadata information
  - Implementing a hash-based segmentation of the name space (internal redirection)
- Common back-end (object store)
  - Prototyped within DSS
    - To be demonstrated at 10^{10} level (current implementation scales with #directories)