Integrating new Storage Technologies into EOS

2015 J. Phys.: Conf. Ser. 664 042043

(http://iopscience.iop.org/1742-6596/664/4/042043)

Journal of Physics: Conference Series **664** (2015) 042043 doi:10.1088/1742-6596/664/4/042043

Integrating new Storage Technologies into EOS

Andreas J. Peters, Dan C. van der Ster, Joaquim Rocha
CERN IT, Geneva, Switzerland

E-mail: andreas.joachim.peters@cern.ch, daniel.vanderster@cern.ch, joaquim.rocha@cern.ch

Paul Lensing
Seagate Technologies/CERN openlab, Geneva, Switzerland

E-mail: paul.lensing@cern.ch

Abstract.
The EOS[1] storage software was designed to cover CERN disk-only storage use cases in the medium-term trading scalability against latency. To cover and prepare for long-term requirements the CERN IT data and storage services group (DSS) is actively conducting R&D and open source contributions to experiment with a next generation storage software based on CEPH[3] and ethernet enabled disk drives. CEPH provides a scale-out object storage system RADOS and additionally various optional high-level services like S3 gateway, RADOS block devices and a POSIX compliant file system *CephFS*. The acquisition of CEPH by Redhat underlines the promising role of CEPH as the open source storage platform of the future. CERN IT is running a CEPH service in the context of OpenStack on a moderate scale of 1 PB replicated storage. Building a 100+PB storage system based on CEPH will require software and hardware tuning. It is of capital importance to demonstrate the feasibility and possibly iron out bottlenecks and blocking issues beforehand. The main idea behind this R&D is to leverage and contribute to existing building blocks in the CEPH storage stack and implement a few CERN specific requirements in a thin, customisable storage layer. A second research topic is the integration of ethernet enabled disks. This paper introduces various ongoing open source developments, their status and applicability.

1. Introduction

EOS provides disk-only storage for physics data at CERN. The service was expanded to meet the capacity requirements as physics analysis storage for LHC Run II starting in 2015. The largest EOS disk pool for the ATLAS experiment manages 14.500 hard disks providing 40 PB of raw capacity. An upper limit for the maximum number of files in such a pool is defined by the in-memory namespace architecture and the available memory of namespace nodes (so-called *MGM*). For a node with 256 GB it has been successfully tested with 500 million files. Extrapolating the current average file and pool sizes, all full instances should stay significantly under the given limit at maximum 300 million files. In the recent past new use cases have evolved at CERN e.g. the CERNBox [5][6] service for *sync & share* metadata producing many small files. In the mid-term CERN needs additionally to find a replacement for the AFS filesystem which currently stores 1.6 billion files. In the meanwhile new storage technologies have evolved and matured allowing to push storage system scalability further. Given this background the

DSS group is running an R&D project to prototype a scalable metadata and data storage API library *libradosfs* based on the scalable object storage RADOS and a file IO library *libkineticio* for clustered ethernet enabled drives based on the Kinetic Open Storage platform [11].

2. *libradosfs* - a metadata server free storage library based on RADOS

libradosfs is a simple and lightweight C++ storage library. It is published under a Free Software license and is available on GitHub [4]. It provides an API to store files and directories as objects in RADOS object storage. As the main architectural difference to the Ceph filesystem implementation *CephFS*[7] it has no additional metadata service. Each instance of the library implements the metadata service. There are advantages having metadata served by a single node:

- effective caching of frequently used entries resulting in low-latency
- easier implementation of a quota system
- central place to throttle or block clients

A fundamental disadvantage is the scalability of the metadata service because all clients connect to a single machine and all metadata is served by a single service inducing a global failure mode.

2.1. Files, Directories and Inodes

libradosfs uses inodes to allow efficient renaming as it is common practice in most filesystems. Each directory is stored in a single inode object. Files may or not have an inode, which is explained further ahead. Directory inodes are looked up via full path index objects. File inodes are indexed in parent directory inodes. The file IO implementation provides a synchronous and

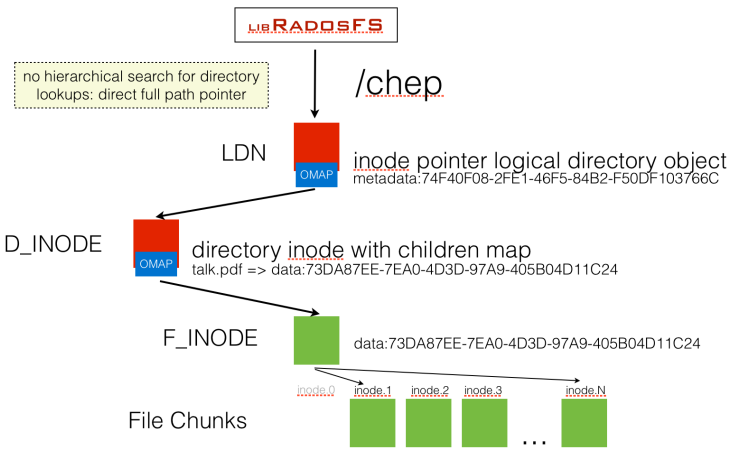

Figure 1. File Lookup

asynchronous API. For the special purpose of physics analysis the API supports vector reads. Vector reads are used to reduce latency by issuing compound read requests for a list of arbitrary *[offset,length]* pairs. Files are chunked with a per-file chunk size. This is required to keep the volume distribution in an object storage system flat and to give the possibility to implement parallel IO per file. Small files are inlined into directory objects to reduce the number of objects and to improve performance. The maximum size for inlining is configurable. Directories are written as write-ahead logs with auto-compactification. Each modification on the directory listing appends the change to the directory write-ahead log. This enables individual library instances to follow changes in a directory without the need to re-sync full directory contents.

Each library instance needs to apply the changes since the last synchronisation offset in the write-ahead log. If the ratio of change-log file entry vs. directory entry number exceeds a predefined value directories get compacted e.g. all deleted entries are removed from the change-log file. The implementation takes care that several client instances don't interfere with compactification and continue to follow at the proper offsets after compaction. *libradoss* supports extended attributes on directory and file inode objects. Additionally it allows to store extended attributes on entries stored inside the directory inode object. This allows to implement efficient queries on generic file and directory metadata in a sub-tree scanning only directory inodes. Nevertheless it offers the possibility to attach possibly large and detailed metadata/attributes to file inodes. The query interface allows to search and select files/directories based on simple comparison operations for all defined metadata keys. The file API offers store and commit syntax for files. File inodes can be created and written unattached to the namespace. Once complete they can be registered into the namespace (attached to a directory object). This behaviour is useful for applications where files are only to be published once they are completely written (file synchronisation). Symbolic links are supported for files and directories. Quota, ACL, file versioning and recycling bin support is currently under development.

2.2. RADOS Pools and Objects

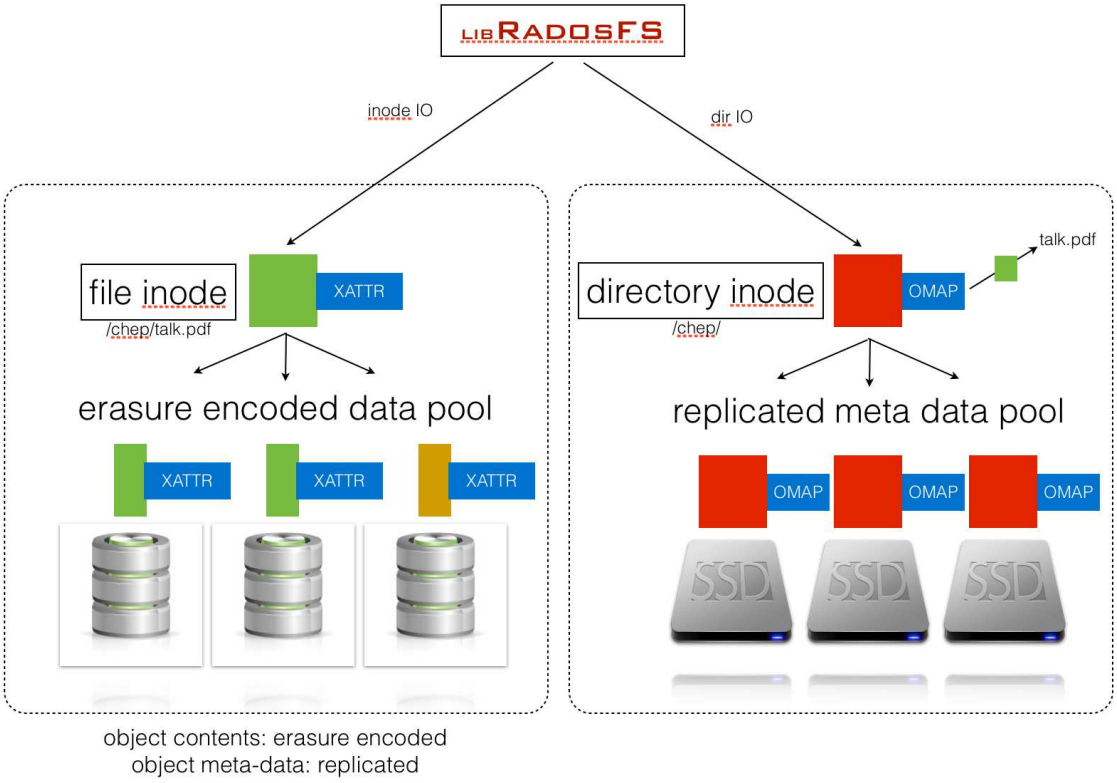

Figure 2. Pool selection

libradosfs allows to select metadata and data pools by path prefix. One can assign fast low-latency pools with a high replication factor as metadata storage while data can be stored in erasure-coded pools with high-latency disks and low capacity overhead. The metadata storage implementation requires support of RADOS kev-value maps - available only in replicated

RADOS pools - while the data storage requires only extended attribute support which is available in erasure-encoded and replicated RADOS pools.

2.3. File System Check - fsck
The file system check tool allows to check and repair inconsistencies in the metadata and object hierarchy. It verifies the bi-directional consistency from directory view inodes to file inodes and from file inodes to directory inodes. On request it repairs missing back-links and reports inconsistent metadata attributes. The *fsck* implementation is scalable:

- tunable parallelsim
- stable memory footprint
  - independent of filesystem size
  - no need to track all existing inodes in memory

3. Ethernet Enabled Disk Drives
Ethernet drives add a small operating system with CPU, memory and a network interfaces to hard drives. They can be attached directly to existing network infrastructure without the need for storage servers. This technology brings the potential to save space, cost and cooling requirements in large computer centres. There are two directions in the ethernet drive technology market. Seagate has developed the kinetic open storage platform. Kinetic drives provide a new storage API similar to well established kev-value store APIs like *memcached*, *redis* protocol etc. HGST drives allow a more generic usage providing direct access to the Linux OS on each disk drive. While the latter one offers more flexibility - it is more error prone! The memory and CPU resources on drives are very limited and it requires an extremely well tuned, slim and efficient application to get the maximum performance out of ethernet enabled drives.

3.1. Kinetic Open Storage Platform
The Kinetic open storage platform [10][11] reduces the architectural complexity of storage systems (see figure 3). The Kinetic platform is built on standard internet technology like Google protocol buffers, sorted string tables and log-structured merge trees. It fits the technology of shingled disks, provides better random write performance, less metadata overhead compared to filesystems and results in a lower total cost of ownership. The maximum object size is 1 MB. The nominal performance of currently available kinetic drives is around 50 MB/s for random/sequential write and sequential read. Drives reach up to 1000 random write object operations. The random read performance is approx. 15% less compared to traditional disk drives. Openstack Swift provides support for Kinetic drives via S3 gateway access. Integration into Ceph is under development. Currently it is still an open question if RADOS OSD daemons can run directly on the ethernet drive or drives have to be served via Proxy OSD servers.

3.2. Kinetic API
The Kinetic API (see table 1) is less feature rich than RADOS and implements mainly low-level functionality. It supports full object get/put operation but no partial get/update/append. There is no abstraction of a key-value store on top of each object, however there is support for object versions. The clustering of drives is not part of the API and has to be implemented by high-level software. A very useful functionality is a peer-to-peer push operation to migrate objects between drives.

4. *libkineticio* - software defined storage with kinetic drives
Scope of *libkineticio* is to provide a file IO interface in front of a kinetic drive cluster with minimal complexity. The library is currently still in the design and prototype phase. To guarantee

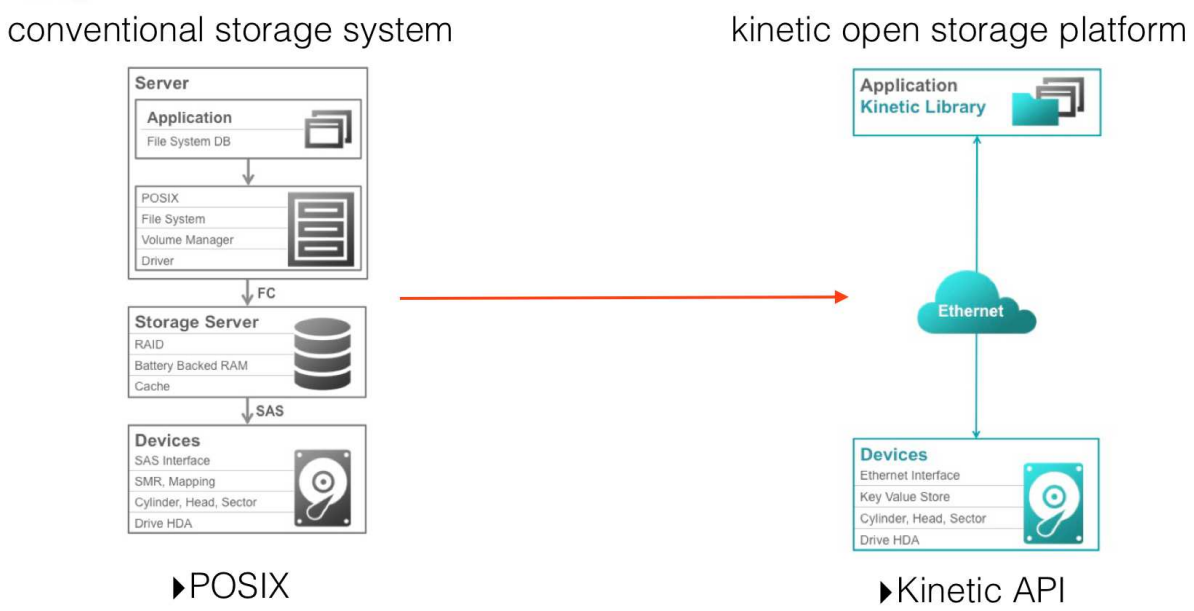

Figure 3. Architecture Transition from Posix to Ethernet Drive Technology

effectively unlimited scalability a limited number of kinetic drives will be clustered into a so-called kinetic cube (see figure 4). The storage volume is scaled horizontally by adding cubes. The cube index must be stored with the metadata of each file externally.

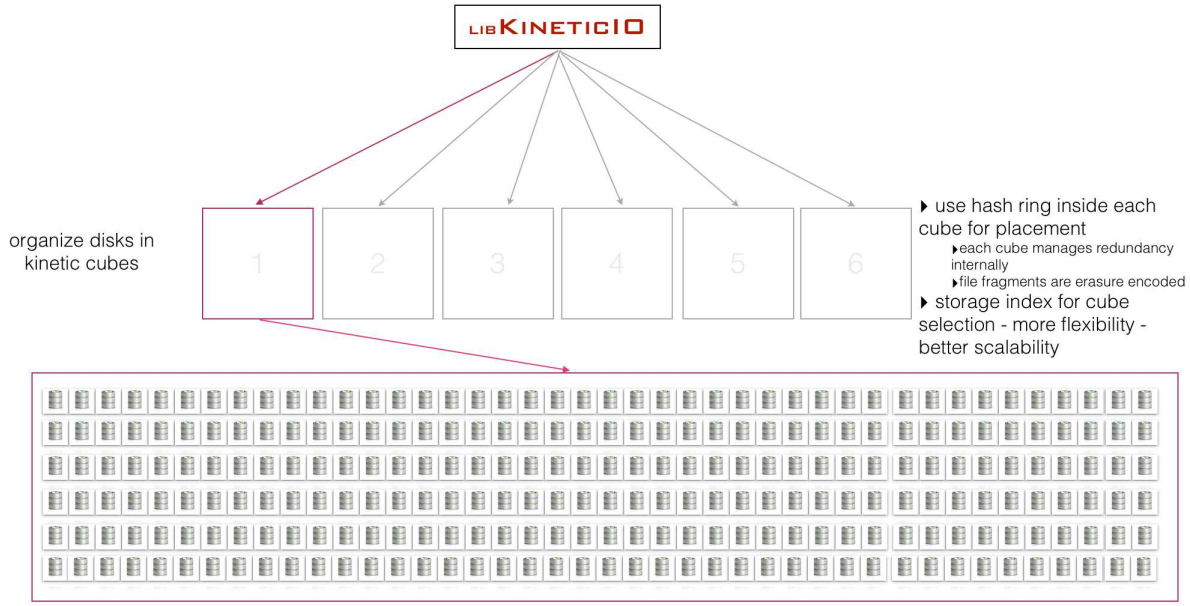

Figure 4. Cubes - Kinetic Drive High-Level Clustering

Each file is chunked into blocks, erasure encoded and placed within a cube using consistent hashing. The proposed encoding scheme is a reed-solomon RS(32,4) code with four additional local parities over eight data stripes each. In the special case of a vandermonde matrix a

API Method	
– Access Control –	
READ	can read
WRITE	can write
DELETE	can delete
RANGE	can do range
SETUP	can setup device
P2POP	can do p2p copy
GETLOG	can get log
SECURITY	can set security
– Other –	
NOOP	like ping
PUT	store object max. value size 1 MB
DELETE	delete object
FLUSH	flush outstanding PUT/DELETE to device (=sync)
GET	retrieve value + metadata
GETVERSION	retrieve version tag for object
GETNEXT	return next sorted key
GETPREVIOUS	return previous sorted key
GETKEYRANGE	return keys in range
SETCLUSTERVERSION	set cluster version
SETPIN	instant secure erase
SECURITY	set ACL
GETLOG	retrieve log
PEERTOPEERPUSH	copy KV between drives

Table 1. List of Kinetic API methods

RS(32,5) code can be used and the first parity stripe can be skipped to be stored. It can be implied by the XOR of the four local parities. The encoding procedure is shown in figure 5. The encoding scheme creates 25% storage volume overhead and provides enough redundancy to avoid the repair of any failed disks within a typical disk lifetime of three years. The encoding algorithm is said to be an MDS code for five parallel disk failures out of 40 drives. This means it is guaranteed that there is no data loss if five drives fail at the same time.

With this configuration the need for block reconstruction will be the standard mode of operation. The availability of local parities requiring simple XOR operations reduces the CPU requirement significantly.

It is envisaged to implement a tool allowing complete repair and draining of kinetic cubes and to base the CODEC on available open-source erasure encoding libraries e.g. Ceph erasure code plug-ins.

In summary the proposed encoding and clustering scheme is optimal for sequential non-sparse access and ideal as archival storage with get/put semantics. Sparse read access without drive repair can imply 8-fold to 32-fold read amplification depending on the drive failure situation and the read request size.

5. A new EOS Storage Ecosystem based on RADOS and Kinetic Technologies

Figure 6 illustrates how a complete storage infrastructure can be designed on top of ethernet drive and object storage technologies. Generic protocol gateways serving XRootd[9], WebDAV,

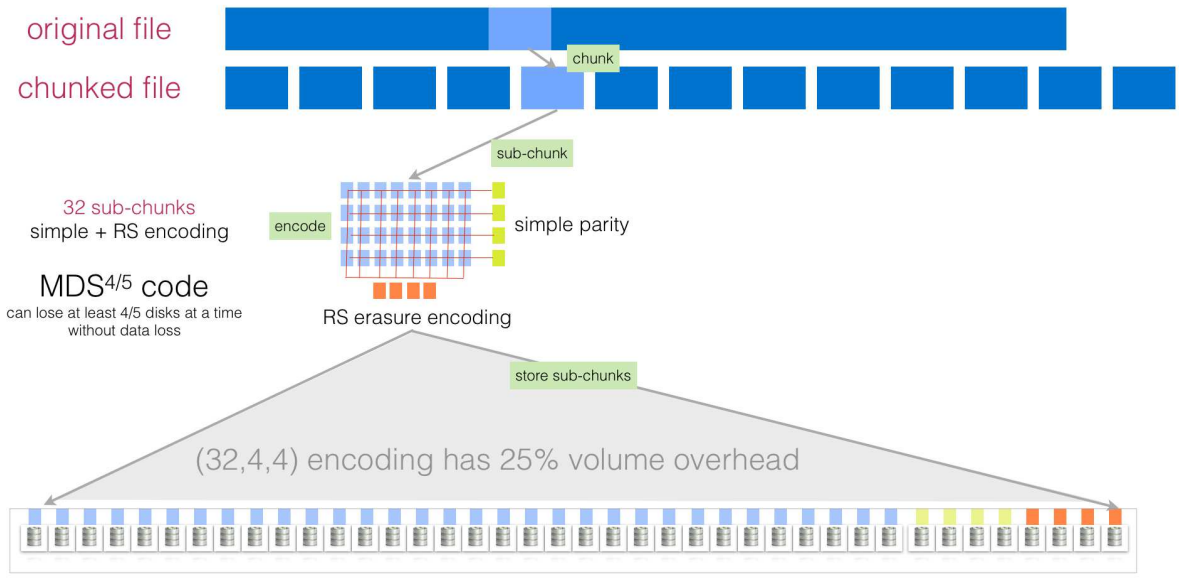

Figure 5. File Erasure Encoding in Kinetic cubes

HTTP, gridFTP, OwnCloud[8] and S3 protocol on top of a FUSE or RADOS layer provide homogeneous access to a shared backend storage. A FUSE layer allows to add non-standard extensions like more fine-grained ACLs/access policies, auditing, strong authentication via kerberos and/or certificates and allows to select use-case dependent between a full-POSIX filesystem implementation with limited scalability (CephFS) or volume optimized storage with POSIX-like semantics (*libradosfs* and *libkineticio*). The storage backend can be devided into three different categories:

- IOPS and low-latency storage with Flash and SSD devices
- Volume and POSIX semantic optimised storage with disks clustered by Ceph/RADOS
- Cost/Network/Space optimised volume storage with ethernet drives

The Ceph storage ecosystem brings additionally support for virtual machine volumes within Openstack. RADOS block devices combined with ZFS filesystems and NFS servers allow to provide a virtual NFS server infrastructure.

In the described scenario EOS converges as a layer unifying and combining many storage building blocks and their configuration into a complete storage solution for manifold use-cases.

6. Summary

Evolving technologies are changing possibilities to run cost-effective storage infrastructures. Object storage and ethernet drive technology open up new ways to build storage as a service. CERN is investing together with openlab partners into storage research to leverage these and to reshape EOS as the future **E**xabyte-scale **O**pen **S**torage at CERN.

7. Acknowledgements

We thank all members of CERN IT, InkTank/Redhat and Seagate Technologies for inspiration, collaboration and groundbreaking storage solutions.

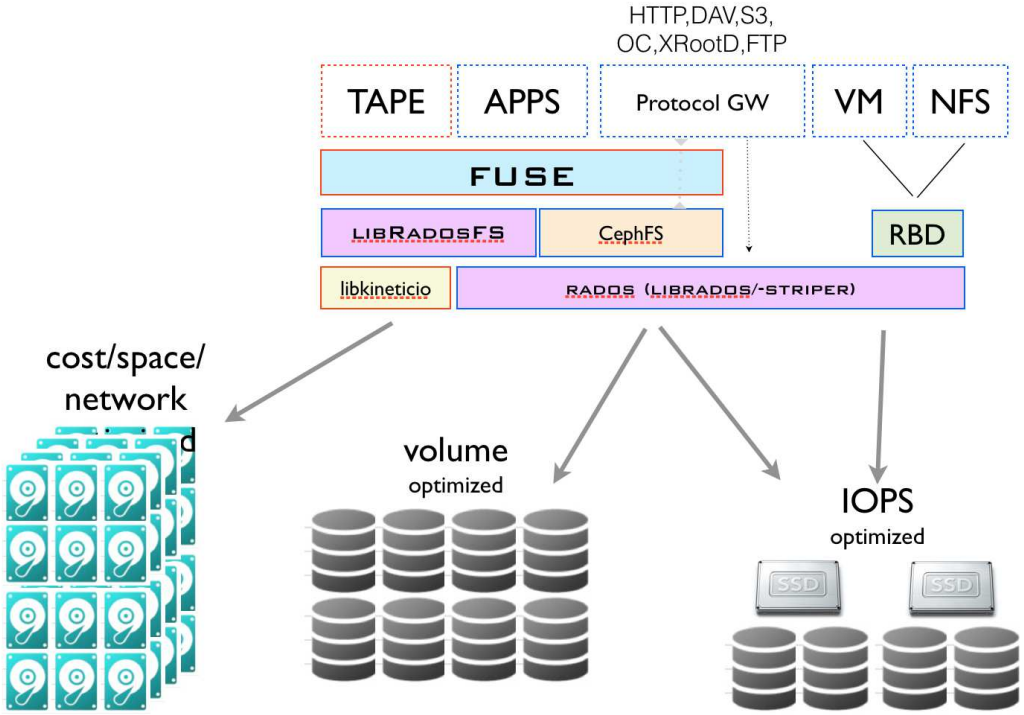

Figure 6. Future Storage Eco System based on RADOS and Kinetic Technologies

References

[1] AJ Peters and Lukasz Janyst "Exabyte Scale Storage at CERN", 2011 J. Phys.: Conf. Ser. 331 052015 doi:10.1088/1742-6596/331/5/052015
[2] "Jerasure: A Library in C/C++ Facilitating Erasure Coding for Storage Applications", Technical Report CS-08-627, Department of Electrical Engineering and Computer Science University of Tennessee Knoxville, TN 37996,
`http://web.eecs.utk.edu/~plank/plank/papers/CS-08-627.html`
[3] SA Weil, AW Leung SA, Brandt andC Maltzahn "RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters",University of California, Santa Cruz,
`http://ceph.com/papers/weil-RADOS-pdsw07.pdf`
[4] "libradosfs",
`http://github.com/cern-eos/RADOSfs`
[5] "CERNBox",
`http://cernbox.cern.ch`
[6] L Mascetti et. al. "CERNBox + EOS: end-user storage for science:, CHEP 2015 - theses proceedings
[7] SA Weil, SA Brandt, EL Miller, DDE Long and C Maltzahn "Ceph: A Scalable, High-Performance Distributed File System", Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI 06) November 2006 - See more at:
`http://ceph.com/resources/publications/#sthash.xGt0T2VR.dpuf`
[8] "OwnCloud",
`http://owncloud.org`
[9] "XRootD",
`http://xrootd.org`
[10] "Kinetic Open Storage Platform",
`http://www.snia.org/sites/default/files2/DSI2014/presentations/\`
`StorTech/Fenn-Hughes_Kinetic_Open Storage_Platform.pdf`
[11] "Seagate Kinetic - Open Storage Platform",
`http://seagate.com`