
# SDP Memo 037: IO and Storage Software

Document number……SDP Memo 37
Document Type……MEMO
Revision……..
Author……Peter J. Braam
Release Date……2017-12-20
Document Classification…… Unrestricted

| Lead Author | Designation | Affiliation |
|---|---|---|
| Peter J Braam | | |
| Signature & Date: | Peter J Braam<br>Peter J Braam (Apr 29, 2018) | |

## SDP Memo Disclaimer

The SDP memos are designed to allow the quick recording of investigations and research done by members of the SDP. They are also designed to raise questions about parts of the SDP design or SDP process. The contents of a memo may be the opinion of the author, not the whole of the SDP.

## Table of Contents

# List of Figures



# List of Tables



# List of Abbreviations

# Introduction

This document provides an overview, specifically at the time of writing - Q1 2018 - of expected developments in storage and IO software and best practices for IO software implementation, as they may be relevant to the SKA SDP. Several themes shape the document:

1. A proliferation of new storage and memory technologies necessitates re-working basic technologies, such as the lowest levels of storage software, and handling multiple tiers in the storage software stack. Products have barely started to reach the market, and there is presently very little industrial activity for HPC IO.
2. Cloud storage systems pose high promise for scalability, but have not yet been proven effective for the highly bursty and concurrent HPC IO patterns, likely to also be seen in the SDP.
3. Addressing application IO performance problems with specific software choices, and the structure of future HPC storage software, with a perspective on the role of cloud storage software.

This memo is organized in the following sections:

1. A review of storage tiers
2. Historical overview of addressing IO problems with software
3. Structure of new storage software stacks.
4. Recommendations for reducing risk.

For easy reference we include in the table summarizing the recommendations in this memo.

| Area | Observation |
|---|---|
| Solution Identification | This memo makes no recommendation of any specific product. Vendors will deliver systems with integrated storage solutions. Few new choices, if any, are presently becoming available, and it will be necessary to observe the market for several more years. |
| Node local storage | Has always been and will likely remain vastly superior to any form of shared networked storage for performance. |
| Cloud Storage SW | Normal concurrent, bursty HPC loads are insufficiently tested. Excellent scalability with custom SW development. File systems over objects defeat the purpose. High promise, much uninformed discussion. |
| Container volumes served from network file system | Excellent performance and most manageability from networked FS. No node to node synchronization. |
| Economic models | Refine from mere capacity prices. |
| Metadata | A strongly coherent directory, regardless of its implementation, poses major threats to scalability. Such directories can often be |

| | |
|---|---|
| | avoided when entry names can be computed instead of retrieved. A decentralized metadata system with weak coherency (Dropbox being an example) can have excellent scalability.<br><br>Directory sizes commonly become problematic when they exceed 10M entries. There is little or no experience with more than a few 100 file system mount points in the directory tree. |
| Exa-scale FS selection | Lustre's Dominance may not change for a significant time period. |
| General Level of risk with Storage Software is High | Both read and write rates envisaged for SDP are record breaking numbers, which likely can only be achieved after overcoming significant difficulties. |
| Implementation approach | 1. All IO is performed exclusively through a customized IO library. In the case of SDP this library will facilitate the layout of critical data (visiblities, image grids etc).<br>2. IO benchmarks, not burdened by compute operations, are available to tune and adjust the IO library. Adjusting IO using real applications is severely hindered by the fact that IO usually merely consumes a single digit percentage of the runtime, while more than 90% is spent in computing operations. |
| Steps for Review and Risk Mitigation | To review an approach to IO validate:<br>1. Proper alignment with RAID, partition and device boundaries to minimize protocol overhead and eliminates false sharing.<br>2. Aggregation to enable reasonable IO transfer sizes, which can deliver a high percentage of hardware capabilities.<br>3. Collections of files or objects can be created, traversed and deleted with adequate speed, and file system directories with many entries (e.g. >100K) are avoided.<br>4. Scalability w.r.t. to the number of nodes is proven.<br>5. The data network and data distributed allow full and even utilization of all network links.<br>6. Data is cached or staged for optimal re-use.<br>7. Small all-to-all data exchanges leverage specialized staging software. |

Table 1: Recommendations

# References

## Reference Documents

| Reference Number | Reference |
|---|---|
| | DAOS high level design<br>https://wiki.hpdd.intel.com/display/DC/Resources<br>https://github.com/daos-stack |
| | HDF5 group<br>https://www.hdfgroup.org/solutions/hdf5/ |
| | ADIOS<br>https://dl.acm.org/citation.cfm?id=1551618 (or see http://www.lofstead.org/) and<br>https://www.olcf.ornl.gov/center-projects/adios/ |
| | Campaign Storage<br>http://storageconference.us/2017/Papers/CampaignStorage.pdf<br>http://www.lanl.gov/newsroom/video/video-stories/supercomputing-storage.php<br>https://www.snia.org/sites/default/files/SDC/2016/presentations/keynote_general/Gary_Grider_MarFS_Scalable_Near-POSIX_File_System_over_Cloud_Objects_HPC_Cool_Storage.pdf |
| | DOE Exascale RFP -<br>http://procurement.ornl.gov/rfp/CORAL2/ |

# Storage Tiers

Storage systems in recent years are seeing an increasing number of hardware **tiers.** Different tiers generally leverage different hardware, and are characterised by different price and performance characteristics. The ranges of performance and pricing variations are extreme as indicated in table 1.

| | HBM | RAM | NVM | Flash | Disk | Tape |
|---|---|---|---|---|---|---|
| $ cost / GB | 0.3-3x RAM | 10 | ?? | 0.2 | 0.02 | 0.01 |
| $ cost / GB/s | 0.3-3 | 10 | ?? | 500 | 2,000 | 10,000 |
| capacity TB/ | 0.1 | 1 | 5 | 30 | 1,000 | many PB |

| | | | | | | |
|---|---|---|---|---|---|---|
| node | | | | | | |
| read BW / node | 750/4 modules | 60/4 dimms | 50/4 dimms | 30/10 devs | 8/100 disks | 0.3/drive |

Table 2: Tier bandwidth and capacity pricing

Hardware vendors indicate that new developments in solid state storage hardware will continue, and that current limitations, e.g. the limited write bandwidth achieved initially with XPoint (roughly 10x below RAM memory write speeds), ***do not*** represent fundamental limitations of the technology and will likely be addressed.

Somewhat in contrast, cluster network bandwidth is presently around 200 Gb/sec per node, and will continue to face challenges to be a good match for the fastest storage devices.

Node local storage has had and likely will have performance advantages that are unlikely to be met by any form of shared networked storage solutions.

## Economic Models for IO Tier Selection

The use of tiers is a tradeoff between cost and capacity and performance characteristics, at the time and scale of the deployment of the SDP.

Quantitative information needs to be obtained and analyzed to design a competitive storage solution. These include future price trends, budgets, required capacity bandwidth and latency and required re-write durability. Energy consumption and reliability can also be important.

As an example of a simple extrapolation, it has been predicted that around 2022 the cost of solid state storage will be lower than that of rotating media on the basis of capacity (on the basis of bandwidth offered, solid state storage has been cheaper for a long time). In the presence of multiple storage tiers, networking and service/server infrastructure to handle additional tiers can present significant costs.

## Tiering Software

### Implementations and Functionality

Informal approaches to tiering are commonplace and take the form of copying data to local storage on compute nodes.

To manage a namespace coherent across multiple nodes, tiering software typically addresses the following concerns. First, it enables small granularity IO through a file system interface into a storage system on a node. Secondly, it efficiently transports many small fragments to network storage devices through aggregation. When data needs to be staged

into fast tiers prior to use by applications, typically the scheduler leverages a data movement process to achieve data readiness for the job without locking resources. Archiving data to slower tiers is simpler and can be done after jobs complete. Sometimes managing tiers is done while maintaining a global namespace which includes all file names. Such solutions are deemed very desirable, but are complex to maintain and have not yet become successful in the market.

When data synchronization is limited to server to client propagation, more tiering solutions are available based on container principles. A general perspective on tiering with containers is shown in Figure 1. The container offers a fine granularity file system interface to an application on the faster storage tier. A streaming mechanism efficiently transports the contents or differentials of contents from the container to slower tiers.

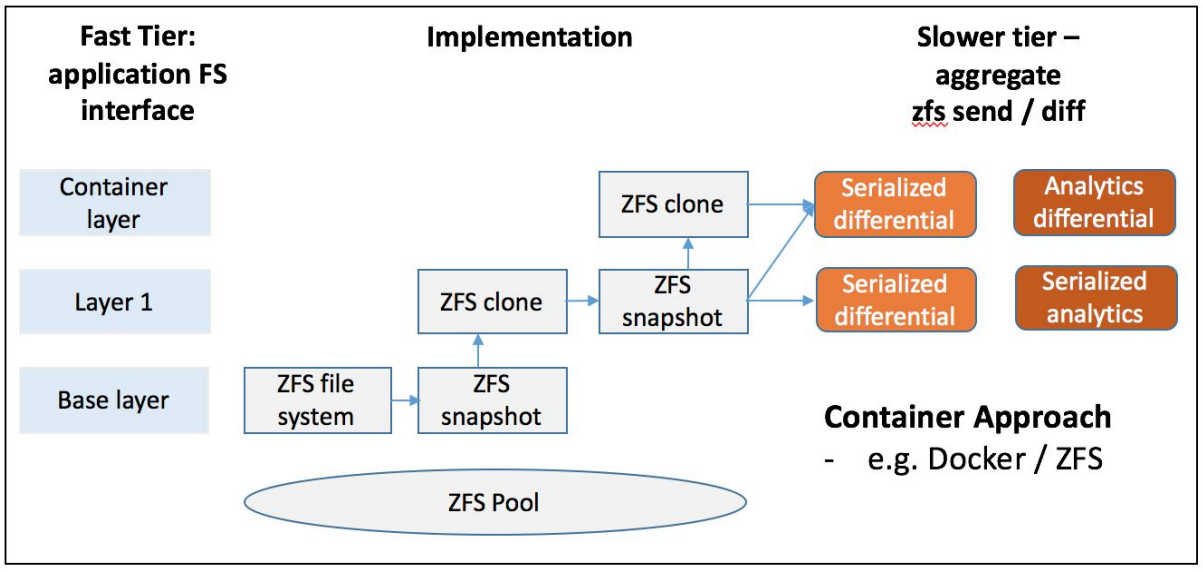


Figure 1: Faster and slower tiers interacting with containers.

Since the 1990's HSM software has offered transparent file system tiers, but these are unlikely to adapt to present day high performance systems. Among these solutions are HPSS, DMF and SAM QFS. The early 2010's saw development of tiering software for flash and disk in a single node, and this has become widely used in appliances shipped by vendors. Due to required capacity of tiers by SDP, which is likely to exceed anything that can be attached to a single node, it is not directly applicable as a scalable storage solution. Newer networked tiering software is being rolled out, e.g. DDN's IME and Cray® XC™ series DataWarp, and actively being explored in major compute installations. Many systems such as DAOS and Lustre mention that tiering will be automatically handled at a future point in time. The validity of such claims needs to be evaluated.

Several important HPC container projects, notably "Shifter" from NERSC, are implementing a tiering system leveraging containers on compute nodes. The system stores the container images as files in a cluster file system. The cluster file system's network IO provides the large granularity transport of data to the faster nodes. On the faster nodes, these files in the

parallel file system are mounted as loopback devices. Each such file is mounted, using it as a device for a local file system offering that file system as a container on one compute node. This provides the compute nodes with extremely fast write-back access for data and metadata stored in the container, and offers many advantages of the data management offered through the cluster file system.

Presently such systems have little or no client - to - client data synchronization.

If a container system is used, applications may be unable to synchronize data node-to-node. This can become a major problem if such a feature becomes desirable at a future point.

### The Campaign Storage Approach

Los Alamos National Laboratory has developed an approach to tiering called campaign storage. Campaign storage considers off the shelf storage systems at different tiers and its functioning is based on two principles:

1. Each layer only allows specific use patterns. For example, the slowest archival layers may only be used with specific tools that aggregate data and metadata, the mid-tier systems can only accept data when it is written with specialized software that arranges data locality without overhead.
2. A simple job scheduler runs IO jobs for ingest into and staging from slower tiers to faster tiers. It schedules open-source distributed parallel data movement software (pftool) that handles refined data placement, integrity. Such distributed parallel data movement software is not widely available.

The advantages of campaign storage are that it avoids lock in to complex vendor systems, but it requires planning.

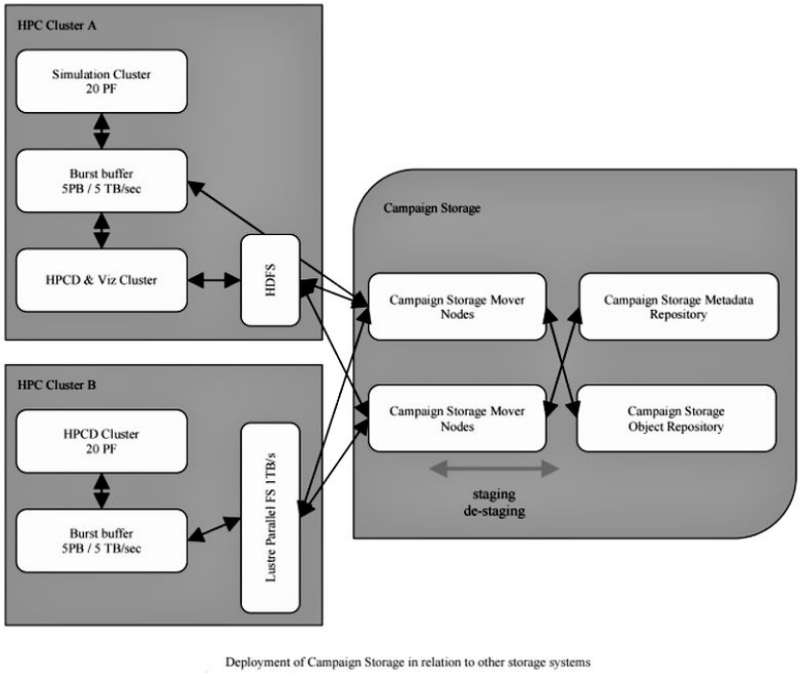


Figure 2: Campaign Storage Deployment

# IO Libraries to Improve Performance

By 2007 the vast majority of large computers (e.g. the top 20 in the top 500) ran a cluster file system as their primary storage system for compute jobs. Large IBM systems mostly used GPFS and other systems used Lustre. Despite this success of cluster file system technology, IO performance of applications was dramatically different from benchmark performance. This section describes what developments in software overcome these problems.

Application IO efficiency (achieved vs offered by hardware and software with fine tuning of resources) is often a single digit percentage. Scalable performance is often limited to just a dozen client nodes.

Cluster file systems implement many performance and scalability optimizations, a few of which we discuss first in this section. Subsequently IO libraries successfully eliminated most other problems. The most popular IO library is HDF5 (named after the accompanying file format), but many discoveries and approaches described here were first made in the context of ADIOS.

## Optimizations in Cluster File Systems

Optimizations in cluster file systems initially addressed the capability to handle massive amounts of clients reliably. Avoidance of thundering herd problems and the elimination of superfluous remote procedure calls were most central. This was followed by numerous asynchronous support mechanisms, such as read-ahead, pre-creation of files, and delayed deletes. To a significant degree these optimizations were required because the VFS layer in the client operating system used relatively inefficient mechanisms to access server data.

Newer, object oriented, cloud storage mechanisms are being considered for HPC loads, and subject to intense, often uninformed discussion. Such systems work with extremely large scale clusters, both server and client side. However:

Compared with cloud workloads, HPC IO workloads are extremely bursty and synchronous, and they may involve up to 100M application threads performing IO concurrently. Cloud systems have not been adequately tested with such loads, and few facilities have the ability to do so.

Yet, cloud systems are expected to be the future, and may be usable. We issue the following guidelines:

If the SDP wishes to use object stores instead of cluster file systems as their primary storage systems, then:

1. This cannot be done through a simple file system layer over the object store, and currently file systems over cloud object stores have not been optimized for HPC use.
2. Efficient IO libraries currently do not layer on systems other than a cluster file system.
3. The SDP applications must be programmed to leverage the cloud storage system.

Notwithstanding these recommendations, cloud object stores may ultimately prove much easier to manage, and more reliable than cluster file systems, but this has not yet taken place in high end HPC, and progress in this area must be monitored. However, object store's benchmark performance for simple loads is now hardware limited.

## Aggregation and alignment

Small IO is very harmful for performance. Even for flash devices, small writes are an order of magnitude slower than IO with large transfer sizes. Some applications can explicitly manage data aggregation and perform IO with large transfer sizes, others rely on caches in the IO system to perform such aggregation. IO libraries such as ADIOS offer extremely aggressive write caching. Aggregation is generally addressed by using sufficiently large buffers on which IO is not performed synchronously with updates to the buffer. These optimizations are normally portable, but can be hindered by a lack of available memory.

A second basic issue is alignment of data. When a single unit of data required by an application program is distributed over multiple devices unnecessarily this can lead to so called read-amplification, and to numerous extra RPC's. Alignment of data is system independent, and not always tuneable - unless IO software is well planned, portability may be hindered.

## Metadata Interactions

Many cases exist where metadata overhead found in cluster file systems dominates the IO execution profile, at the expense of IO throughput and latency.

A primary culprit in this area is file creation and opening. A mechanism to address this is a collective open, where a single file handle at the file system level can be used by a large process group. All IO libraries offer forms of collective file create and opening. When files are used in this manner, the internal layout of the file may replace what would be offered by a large collection of open files in a file system.

A second issue is traversal of metadata, in Unix parlance seen in the “ls -l” command. At the file system level numerous attempts have been made to improve this with widely varying mechanisms, such as distributed metadata, combining inode and directory entry information and read-ahead. Traversing collections while using the POSIX interface is limited by single threaded semantics and locks. IO libraries have taken a different approach to this by embedding the object collections as trees inside the file data, thereby bypassing POSIX semantics. In object systems a secondary database is used to hold collections. Few comparisons exist between these approaches, but generally simply limiting the number of open files can eliminate such issues.

A strongly coherent directory, regardless of its implementation, poses major threats to scalability. Such directories can often be avoided when entry names can be computed instead of retrieved. A decentralized metadata system with weak coherency (Dropbox being an example) can have excellent scalability.

Directory sizes commonly become problematic when they exceed 10M entries.

Horizontally scalable databases are not known to have brought benefits to HPC metadata handling.

## Non-implemented IO optimizations

Most critical use patterns have been addressed in cluster file system software and/or IO libraries, but, aside exceptions remain.

A particular issue that remains is that many pathological uses of storage systems exist, and may be accidentally introduced into applications. While the IO system could detect and warn about pathological uses, this is only done externally, by using logging and IO analysis systems (see for example the products from Ellexus Mistral).

### Staging

Recent progress on the most complex - all-to-all - data exchanges has found that creating dynamic staging nodes which aggregate data from many nodes before distributing it to other nodes, have been successful. However, these approaches have not yet led to mainstream developments in IO libraries. This may not be relevant to the SDP.

Most importantly - IO libraries lack a facility to perform the data management associated with staging between tiers, as discussed in the previous section.

# The structure of future IO software for scientific computing

Future HPC storage software systems are likely to refine current approaches with a 2-3 layer structure:

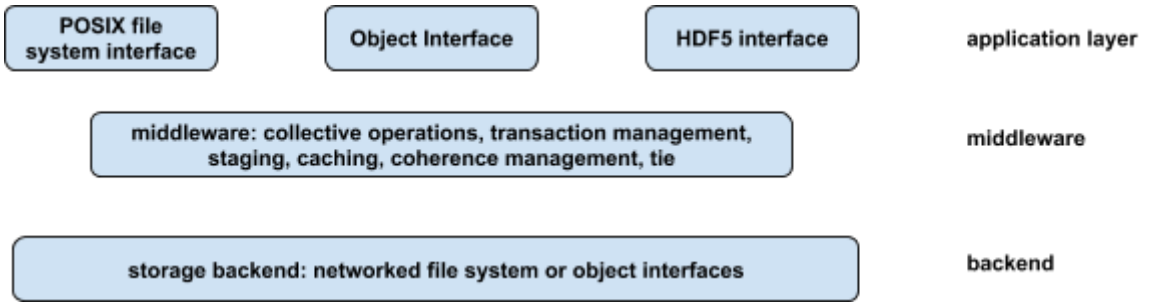


Figure 4: module decomposition of future IO stacks

In a file system like Lustre or GPFS internal structures like these are present, but the system is only usable as a monolithic entity. Future developments may run Lustre on 3rd party object servers, introducing at least a partial layering. DAOS separates at least the application layer from the two lower layers. Presently, no middleware layer has full functionality, leading to many ad-hoc approaches to handle data management problems, such as adding tiering mechanisms or staging for re-shuffle operations.

## The Future IO Application Interface

Three different IO interfaces are likely to be offered at the application level. Possibly all interfaces offer access to the same data simultaneously:

1. A (near) POSIX file system interface
2. A object and key value store interface
3. The HDF5 interface

Presently object interfaces are made available through separate systems, while HDF5 is typically layered on the file system. On the world's very large systems, Lustre holds the dominant position, IBM Spectrum Scale has a good presence, and at the very top Lustre forks for China's Taihu Systems and for Fujitsu's K system exist.

Lustre's dominance may not change for a significant time period.

Although Lustre's dominance was expected to decrease with the arrival of exa-scale the formulations found in the Coral 2 RFP's indicate that it may continue to be used.

## The Storage Layer

Future implementations are considering a layered construction where API implementations are layered on a lower storage layer. Candidates for lower storage layers are:

1. A scalable cluster file system
2. An object store, such as DAOS, CEPH (or numerous commercial variants), cloud object stores such as Azure Blob or Amazon S3.

DAOS is one of the few (the only?) open source project implementing a new underpinning for exa-scale IO. It has a beautiful design, but despite many years of development, it appears not yet to be in use. Its implementation is extremely traditional, using C libraries developed at Argonne and related laboratories, not more modern languages or interfaces (such as transactional memory). For the foreseeable future, and hinted at in the DOE 2021/2022 exa-scale procurement RFI, Lustre may remain or become the storage layer of choice for the first exa-scale IO systems, contrary to expectations earlier in the 2010 decade.

# Risk Avoidance for SDP Storage and IO

## Scope

Purchasing and planning data storage systems is presently a complex undertaking due to the availability of many tiers.

SDP planning must use refined cost models taking into account capacity and bandwidth requirements, to meet those of the ingest, imaging and transient data processing, and data management (e.g. for the use of staging) IO software.

The SDP systems envisage a continuous influx of data (writes of data) from correlation systems at a rate comparable with 1TB/sec, and consumption of data (reads from a storage system) at a 10TB/sec rate.

Both read and write rates envisaged for SDP are record breaking numbers, which likely can only be achieved after overcoming significant difficulties.

In the author's experience with world class HPC IO systems, all elements of the system - network, switches, storage devices, busses, processors and all software,have lead to problems.

Sending reasonably sized IO packets (bigger than 1MB / packet) almost certainly will be necessary, and the correlation system must support this.

In practice, to avoid congestion, precise routes for data traffic have often been implemented through low level configuration of switches and routers.

The overhead of IP protocols has had to be addressed with detailed tuning of kernel level software. Top performance IO systems have ***never before*** used IP networking, but always relied on RDMA IB networks.

## Implementation Choices

Without doubt, but rarely done, the most successful use of IO software by major research laboratories has been enabled through 2 simple mechanisms:

1. All IO is performed exclusively through a customized IO library. In the case of SDP this library will facilitate the layout of critical data (visiblities, image grids etc).
2. IO benchmarks, not burdened by compute operations, are available to tune and adjust the IO library. Adjusting IO using real applications is severely hindered by the fact that IO usually merely consumes a single digit percentage of the runtime, while more than 90% is spent in computing operations.

SDP should create an IO reference library and IO benchmark prototype to guide production software development.

## Critical Design Review for IO architecture

Ultimately, only a few adjustment to parameters in IO will normally lead to good performance. The following callout captures these.

To review an approach to IO validate:

1. Proper alignment with RAID, partition and device boundaries to minimize protocol overhead and eliminates false sharing.
2. Aggregation to enable reasonable IO transfer sizes, which can deliver a high percentage of hardware capabilities.
3. Collections of files or objects can be created, traversed and deleted with adequate speed, and file system directories with many entries (e.g. >100K) are avoided.
4. Scalability w.r.t. to the number of nodes is proven.
5. The data network and data distributed allow full and even utilization of all network links.
6. Data is cached or staged for optimal re-use.
7. Small all-to-all data exchanges leverage specialized staging software.