Summary

This paper broadly talks about co-locating of compute and storage nodes in HPC architecture to overcome the I/O challenges in scientific applications. In a conventional HPC architecture compute nodes and storage nodes are located far apart and are interconnected via shared network. The bottleneck of the system is the gap between compute node and I/O which continues to widen up with increasing storage. The idea of co-locating distributed storage nodes with compute nodes seems to be an eye catcher, as it allows the applications to perform their computation or manipulate their intermediate results and checkpoints, the data only needed to be transferred over the network for archival purpose only. This paper also discusses the improvement of metadata performance and data movement and how to overcome them. This paper identifies these challenges and proposed a distributed file system FusionFS to address them. The objective of the paper is to access FusionFS's scalability towards exascale computing systems (10^{18} ops/sec).

To tackle the above-mentioned challenges, implementation of FusionFS is proposed in the paper. FusionFS disperse all its metadata to the available nodes to achieve the maximum concurrency. Every FusionFS Client optimizes the file writes with local writes, thus reducing the network traffic and making the I/O throughput highly scalable. FusionFS uses the POSIX interface with FUSE framework to run the applications directly without any modifications. FusionFS provides ffs_open(), ffs_close(), ffs_read(), and ffs_write() API's similar to POSIX. This paper has given some insights on two of the most prominent features of the FusionFS metadata management while using the parameters like NameSpace, Data Structure, Persistence, Consistency and Fault Tolerance and File Manipulation and its operations like File Open, File Read, File Write, File Close, File Movement.

To understand FusionFS's scalability, an event driven simulator, FusionSim is developed and tested by FusionFS traces. FusionSim simulation demonstrates that the FusionFS scales linearly and has the capability to deliver 329 TB/s over 2 million nodes surpassing 2.5 TB/s aggregate I/O throughput on 16k nodes of PVFS and HDFS. Two of the building blocks of the FusionSim are ROSS and CODES. ROSS is a discrete-event simulation system uses the time warp algorithm and features reverse computation for optimistic simulation whereas CODES is a simulation system built on ROSS used for investigating and co-designing of exascale systems. With these two techniques in mind FusionSim is developed to simulate FusionFS behavior on exascale systems or large-scale systems.

How is this work different than the related work

Major distributed file systems like NFS, PVFS, HDFS, GFS have compute and storage node located at a distance apart interconnected via shared link to perform I/O operations. Even though the network bandwidth is high enough to provide fast data transfer but when the day comes for the exascale systems it would not be adequate for data intensive applications. Cost of the hardware and overall efficiency of the systems are the two major factors in deciding the future of the HPC systems. With the ever-increasing data in cluster computing the cost of the storage and hardware for radical performance is increasing exponentially. N number of hardwares running m number of process or threads to get extremely high parallelism, concurrency and resilience out of HPC systems is a real deal.

The conventional file systems already implemented still struggling to improve the metadata performance and overall throughput of the HPC systems while running on different cores, configuration, RAM, number of nodes etc. Most of the file systems are tightly coupled with execution framework which means that scientific applications not using these frameworks must be modified to use these non-POSIX file systems. For those that offer a POSIX interface, they are not designed for metadata-intensive operations at extreme scales.

FusionFS address these problems and provides ability to the HPC systems to co-locate their compute and storage nodes. This will solve many existing HPC problems. FusionFS has completely distributed metadata

management based on the implementation on Zero Hop Distributed Hash Table (ZHT) which increases the concurrency and provides scalability towards large number of nodes or exascale systems. ZHT is dynamic and improves the metadata management and file movement over the nodes. FusionFS shows linear scalability if the number of nodes increases. The file read/write and aggregate throughput will also be increased as the number of nodes increases. FusionFS can work under intensive workload at extreme scale.

Identify the top 3 technical things this paper does well

- Comparison of FusionFS metadata performance and I/O throughput under intensive workload at large scale with other major file systems like HDFS, GFS, PVFS etc.
- Simulation of FusionSim simulator to give the general idea how the FusionFS will work in exascale systems.
- Performance improvement for real-world intrepid applications on FusionFS i.e. Plasma Physics, Astrophysics, Turbulence etc.

Identify 3 things the paper could do better

- Paper forgets to mention the types of storage we will be using in FusionFS file system while implementing on exascale systems.
- Paper could have mentioned any techniques to compress files before storing them as it would have increased the I/O throughput.
- Paper could have discussed or provided extensive research or detailed discussion about FusionSim clearly demonstrating how FusionFS will be implemented on exascale system.

If you were to be an author of a follow-up paper to this paper, what extensions would you make to improve on this paper?

Most of the experiments were conducted on the POSIX interface that means Fuse overhead is considered in the reported results. As an author of the paper I would have used the FusionFS libraries which provide API's to conduct the same experiments to compare the results of both. Moreover, file compression is one of the efficient features, we can use any of the file compression techniques before storing it may prove effective in large scale systems.