An Effective Storage Mechanism for High Performance Computing (HPC)

Fatima El Jamiy, Abderrahmane Daif, Mohamed Azouazi and Abdelaziz Marzak
University Hassan II Mohammedia, Faculty of Sciences Ben M'sik,
Laboratoire Mathématiques Informatique et Traitement de l'Information MITI, Casablanca, Morocco

***Abstract*—All over the process of treating data on HPC Systems, parallel file systems play a significant role. With more and more applications, the need for high performance Input-Output is rising. Different possibilities exist: General Parallel File System, cluster file systems and virtual parallel file system (PVFS) are the most important ones. However, these parallel file systems use pattern and model access less effective such as POSIX semantics (A family of technical standards emerged from a project to standardize programming interfaces software designed to operate on variant UNIX operating system.), which forces the MPI-IO implementations to use inefficient techniques based on locks. To avoid this synchronization in these techniques, we ensure that the use of a versioning-based file system is much more effective.**

***Keywords*—Big data; High Performance Computing; Storage; Distributed File System; BlobSeer**

I. Introduction

Industrial research and development on parallel file systems that can provide outstanding performance is prompted by the need to treat raising volume of data in technological and business applications that usually require high Input/output throughput [1]. Among these, we can cite Physical Simulation, processing a big volume of data sets to extract knowledge and business email services. In this article, we will present two important parallel file systems while addressing their major limitations and propose a new File System based on versioning and inspired from PVFS and Lustre File systems. Our choice of these parallel file systems is mainly due to their extensive use. Although both systems have many differences in their design such as Locking, Semantics, Caching and Striping Pattern, they have the same fundamentals of Striping Width and metadata management. The main purpose of this document is to emphasize the strengths of the two systems and present a prototype of a new file system based on the BlobSeer storage system that provides high Input/output throughput while ensuring simultaneous access data for distributed file systems.

II. Main Problem and Approach

HPC is traditionally defined by parallel scientific applications that are becoming more and more intensive on data and whose I/O (input-output) performances become quickly a problem, causing a bottleneck that has a negative impact on the overall application performance [2].

It has been found that most software components of parallel computing systems are in place such as operating systems, local storage systems, and message passing systems. However, one area is devoid of components to the production level for clusters, it is the one of systems parallel I/O [3].

The HPC system architecture is divided into several support layers (Figure 1) that provide much functionality:

- Abstractions to data structure, cell (eg netCDF parallel, Adios)

- Manage the organization of the data access

- Sustain a logical space (eg Lustre and PVFS); it manages the storage hardware and provides a single view, while focusing on simultaneous and independent access.

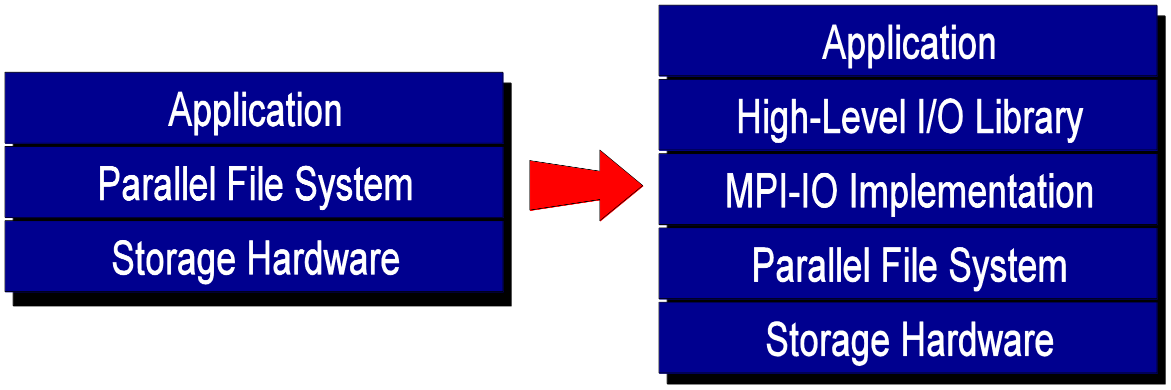

Fig. 1. High Performance Computing Architecture

A. PVFS and LUSTRE

PVFS is a scalable high performance parallel file system for HPC systems. Distributing data through several nodes named Input/Output nodes is the main method used to offer a high access to data kept in the file system by numerous users. The expansion of data that way enable applications to use many directions to access data and thus eliminate bottlenecks while the total throughput is also improved [4]. Although the traditional mechanism of access to the file is convenient and allows all applications to access stored files on many different types of file systems, there is an overload in access through the core [5].

B. LUSTRE

Lustre is an open source project (GPL) distributed file system based on objects. Its name is the mixture of the words "Linux" and "cluster". The system architecture consists of metadata server (MDS), the target object storage servers (OST) and customers. The metadata servers maintain data about all files in Lustre [6].

In fact, to guarantee the atomicity of access to non-contiguous and overlapping areas, parallel file systems use access patterns less powerful and less effective such as POSIX semantics that forces MPI-IO Implementations (Message Passing Interface) to use inefficient techniques based on locks [7]. To avoid this synchronization in these techniques, we ensure that the use of a versioning-based file system is much more effective. Our prototype will be greatly inspired from PVFS and Lustre and It will be mainly an evolution of PVFS, it will re-take the same principles of operation (cell, volume, persistent caching, replication,). Its first goal is to provide high Input/output throughput while providing simultaneous data access.

C. Blobseer Architecture

BlobSeer is a large-range distributed storage service that meets the advanced management data from a large mass of data requirements [8]. It is based on the use of versions to manipulate simultaneously a large binary objects (BLOBs) in order to effectively exploit parallelism in data and sustain a high throughput despite the massively data access parallel [9] .

The client controlled and handled the Versioning in the system where Each BLOB (Binary large object) has its own unique key. When writing or adding, the data to be written is divided into a number of small pieces and are written in the data providers listed by the provider vendor manager [10]. New versions are then produced in each write or append, however storage space is conserved because what is kept is just the dissimilar patch. During a read operation, the latest version number is first obtained from the manager version. All pieces that match this version are identified by the client and then perform parallel read operation [11]. BSFS does not have master-slave architecture and thus released from the single point of failure. The biggest benefit is that metadata are highly distributed between metadata providers. So there is no fear of point failures that will stop the availability of metadata and slow down all operations that depend on it. Another feature that adds to the attraction of this architecture is the versioning technique. Indeed, the amount of treatments accomplished in the parallel file system is optimized because the concurrency control algorithm [12].

D. Approach:

The principal point where Lustre and PVFS have differences is in the method they use to split the metadata. In fact the metadata management is an important element in offering scalability and performance, and in this context PVFS does not give any assurance that quality will be fulfilled since all the tasks related to that are distributed across the servers' without taking in consideration that factor. Whereas on the other side, Lustre reach an elevated availability but it does not enhance performance. To attain this, two servers are used in a consolidation plan.

When concurrent requests occurred, constancy and stability are provided by Lustre, whereas PVFS does not bear that implementation plan, this possible only if we have non overlapping areas to access. PVFS does not provide POSIX semantics [14]. The atomicity of writes is guaranteed in non-overlapping areas and even in non-overlapping, non-contiguous regions. It does not implement a lock infrastructure [15].

The atomic mode ensures that data written to a process is immediately visible to another process (like POSIX semantics by default). ROMIO currently uses file locking to implement the functionality of the atomic mode MPI-IO [16]. Locks in PVFS are not supported and therefore the atomic mode is not supported too [17, 18, 19].

The implementation of PVFS does not include all the features of MPI-IO specification. Eventually we reached a point where it was obvious to us that a new design is required. Our conception embodies the principles of MPI-IO components missing for PVFS that we consider key to an efficient, robust and high performance parallel file system.

The efficient design oriented version of BlobSeer enables a lock-free data access, and thus promotes scalability under high concurrency. A high I/O debit data is offered by Blobseer because of his particular characteristics and decentralized data and metadata management. This realization aim to come up with a new vision that demonstrate the way Blobseer can be employed as effective backend storage by expanding it to a distributed file system for HPC systems.

We will configure our new file system and evaluate its performance against the PVFS performance and Lustre on a set of Data-intensive computing benchmarks and real systems.

III. Comparison with Related Work

There have been many works aiming to develop and enhance the performance of parallel systems but until now they cannot bring the results sought by this kind of systems. Hadoop, an open source framework designed to carry out processing on massive data volumes, on the order of several petabytes (or several thousand TB) and written in Java has been improved with PVFS. PVFS is a widely used parallel file system that allows a high performance data access for the operations I / O that are adjacent and non-contiguous without guaranteeing atomicity.

Continuing with Hadoop, another experiment has been done to enhance it. It was integrated with Blobseer to allow a high access data and avoid synchronization but it is still not enough to take over all the requirements.

In another work, the authors propose a lock pattern for a non-contiguous access strictly aimed at reducing the scope of the locked region to areas that are really accessed. However, this approach does not prevent the serialization for the overlapping and simultaneous Input/Output.

IV. Conclusion

Our work confirm that using a new layer created with the different principles of conception cited above is apt to improve the efficiency of data storage layer and thus that of the whole HPC applications. With the new layer of the file system BlobSeer (BSFS), we will propose a new file system inspired from the two principal distributed file systems Lustre and PVFS. The next step is the implementation of this new file system on the Grid'5000 infrastructure and evaluates its performance.

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simpli_ed Data Processing On Large Clusters, Commun. ACM, vol. 51, 2008.

[2] Mahadev Satyanarayanan, James J. Kistler, Puneet Kumar, Maria E. Okasaki, Ellen H. Siegel, and David C.Steere, "Coda: A highly available file system for a distributed workstation environment," IEEE Trans. Comput., 1990.

[3] A. Ching, K. Coloma, and A. Coudhary, Challenges for Parallel I/O in GRID Computing. Publisher's address: American Scientific Publisher, 2006, ch. 6, Grid I/O.

[4] K. Shvachko, "Hdfs scalability: The limits to growth", The USENIX Magazine , 2010.

[5] J. Dean and S. Ghemawat. MapReduce: A Flexible Data ProcessingTool. CACM, 53(1):72–77, 2010.

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system.In SOSP, 2003.

[7] C. Yan, X. Yang, Z. Yu, M. Li, X. Li, IncMR: incremental data processing based on MapReduce, in: Proc. Int'l Conf. Cloud Computing, CLOUD, 2012.

[8] S. Wu, F. Li, S. Mehrotra, and B. C. Ooi. Query Optimization for Massively Parallel Data Processing. In SOCC, 2011.

[9] S. Babu. Towards automatic optimization of MapReduce programs. In SOCC, 2010.

[10] J. Shafer, S. Rixner, and A.L. Cox, "The hadoop distributed filesystem: Balancing portability and performance",IEEE international symposium, 2010.

[11] J. Dittrich, J.-A. Quian´e-Ruiz, S. Richter, S. Schuh, A. Jindal, and J. Schad. Only Aggressive Elephants are Fast Elephants. PVLDB, 2012.

[12] William Gropp, Ewing Lusk, and Anthony Skjellum. Using MPI : Portable Parallel Programming with the Message Passing Interface. MIT Press, 1994.

[13] Peter Aarestad, Avery Ching, George Thiruvathukal, and Alok Choudhary. Scalable approaches for supporting MPI-IO atomicity. In Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid,May 2006.

[14] Randal E. Bryant, Data-intensive scalable computing for scientific applications, Comput. Sci., 2011.

[15] S. Weil, S. Brandt, E. Miller, D. Long, C. Maltzahn, "Ceph: A Scalable, High Performance Distributed FileSystem," In Proc. of the 7th Symposium on Operating Systems Design and Implementation, Seattle, WA, 2006.

[16] Avery Ching, Alok Choudhary, Kenin Coloma, Wei Keng Liao, Robert Ross,and William Gropp. Noncontiguous access through MPI-IO. In Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003.

[17] Philip H. Carns, Walter B. Ligon III, Robert B. Ross, and Rajeev Thakur. PVFS: A parallel file system for Linux clusters. In Proceedings of the 4th Annual Linux Showcase and Conference, pages 317–327, Atlanta, GA, 2000. USENIX Association.

[18] Avery Ching, Alok Choudhary, Wei-keng Liao, Rob Ross, and William Gropp. Noncontiguous I/O through PVFS. In CLUSTER '02 : Proceedings of the IEEE International Conference on Cluster Computing,CLUSTER '02, pages 405–,Washington, DC, USA, 2002. IEEE Computer Society.

[19] K. Shvachko, H. Huang, S. Radia, and R. Chansler, " The hadoop distributed file system," In 26th IEEE (MSST2010) Symposium on Massive Storage Systems and Technologies, 2010.