Journal of Physics: Conference Series **513** (2014) 042018 doi:10.1088/1742-6596/513/4/042018

Forming an ad-hoc nearby storage, based on IKAROS and social networking services

Christos Filippidis1, **Yiannis Cotronis**2 **and Christos Markou**1

1 Institute of Nuclear & Particle Physics, "NCSR" Demokritos, Greece
2 Department of Informatics & Telecommunications, University of Athens, Greece

E-mail: filippidis@inp.demokritos.gr

Abstract. We present an ad-hoc "nearby" storage, based on IKAROS and social networking services, such as Facebook. By design, IKAROS is capable to increase or decrease the number of nodes of the I/O system instance on the fly, without bringing everything down or losing data. IKAROS is capable to decide the file partition distribution schema, by taking on account requests from the user or an application, as well as a domain or a Virtual Organization policy. In this way, it is possible to form multiple instances of smaller capacity higher bandwidth storage utilities capable to respond in an ad-hoc manner. This approach, focusing on flexibility, can scale both up and down and so can provide more cost effective infrastructures for both large scale and smaller size systems. A set of experiments is performed comparing IKAROS with PVFS2 by using multiple clients requests under HPC IOR benchmark and MPICH2.

1. Introduction

World wide scientific experiments are generating datasets which are increasing exponentially in both complexity and volume, making their analysis, archival, and sharing one of the grand challenges of the 21st century. A supercomputer can be defined as a device for turning compute-bound problems into I/O-bound problems. This addresses the fundamental shift in bottlenecks, as supercomputers gain parallelism at exponential rates, while the performance of the storage infrastructure increases at a significantly lower rate [1]. This implies that the data management and data flow between the storage and compute resources is becoming the new bottleneck for large-scale applications. The support for data intensive computing [2] is critical to advancing modern science as storage systems have experienced a gap between capacity and bandwidth [1].

It is obvious that storage systems in future exascale systems will be their Achilles heel, unless storage systems are re-architected to ensure scalability to millions of nodes and potentially billions of concurrent I/O requests. We can clearly show the problem if, for example, we try to extend a supercomputing infrastructure such as the Blue Gene/P to scale to a million of nodes. We have only to consider the booting parameter at the Blue Gene/P machine, On 256 processors, it takes 85 seconds to boot the allocation; on the full 160K processors, it takes 1090 seconds. Unfortunately, it appears that the machine boot time grows linearly with the number of nodes, translating to potentially over 25K seconds (7+ hours) boot-time at 1M node scales [6].

Exascale systems will require I/O bandwidth proportional to their computational capacity. Traditional globally shared file systems have limitations when used with large-scale systems, because:

(i) bandwidth does not scale economically to large-scale systems,
(ii) I/O traffic on the high speed network can impact on and be influenced by other unrelated jobs,
(iii) I/O traffic on the storage server can impact on and be influenced by other unrelated jobs.

To avoid those limitations, one approach is to configure multiple instances of smaller capacity, higher bandwidth storage closer to the compute nodes (nearby storage) [3]. The multiple instances can provide aggregated exascale size bandwidth and capacity and can avoid much of the impact on other jobs. A possible disadvantage to this approach is that it does not provide the same file system semantics as a globally shared file system. In particular, it does not provide file cache coherency or distributed locking, but there are many use cases where those semantics are not required. Other globally shared file system semantics are required, such as a consistent file name space, and must be provided by a nearby storage infrastructure. In cases where the usage or lifetime of the application data is constrained, a globally shared file system provides more functionality than the application requirements while at the same time limits the bandwidth which the application can use. Nearby storage as described above reverses that, providing more bandwidth but not providing globally shared file system behavior [3].

Except from the above I/O systems challenges, we have to consider additional issues in order to build a large scale distributed infrastructure, such as: the meta-data management and the way that individual users, groups or bigger consortiums produce, transfer, publish and share data. It seems that Grid computing infrastructures are not so efficient in enabling individual users and small groups to fully exploit their capabilities. A more dynamic way to manage and share data is needed, similar to the dynamic allocation of other resources, such as CPU hours and storage space. Cloud computing and Web 2.0 technologies could probably provide that, due to their user friendly approach and their distributed computing paradigm driven by economies of scale. The prospect of needing only a credit card to get on-demand access to 100,000+ computers in tens of data centers distributed throughout the world, resources which can be applied to problems with massive, potentially distributed data, is exciting! [1]. We choose to build a hybrid model that can fully exploit existing Web and Cloud infrastructures, such as Facebook, and at the same time maintain characteristics such as interoperability, non trivial qualities of service and multi-institutional/multi-domain cooperation inherited by Grid computing infrastructures, such as EGI/gLite.

In section 2, we examine related work. In section 3 we present an ad-hoc nearby storage based on IKAROS and Facebook, In section 4 we present the comparison results between IKAROS and PVFS2 by using multiple clients requests under HPC IOR benchmark and MPICH2. Finally, in section 5 we present the conclusions.

2. Related Work

There have been many shared and parallel file systems proposed since the 1980s, such as the Network File System (NFS), Andrew File System (AFS), GPFS, PVFS, Lustre, Panases, Microsoft's Distributed File System (DFS), GlusterFS, OneFS, POHMELFS, and XtreemFS. While the majority of these file systems expose a POSIX-like interface providing a global namespace, and many have been adopted in cluster-, grid-, and even supercomputing, the biggest

critique of these file systems is their vision that compute resources should be completely agnostic of the data locality on the underlying storage system. All of these file systems assume that the storage nodes/servers are significantly fewer than the client/compute nodes which will access the file system, resulting in an unbalanced architecture for data-intensive workloads [1]. A variety of distributed file systems have been developed to address this unbalance, such as GFS, HDFS, Sector, CloudStore, Ceph, GFarm, MooseFS, Chirp, MosaStore, PAST, Circle, and RAMCloud. However, many of these file systems are tightly coupled with execution frameworks (e.g. Hadoop), which means that scientific applications not using these frameworks must be modified to use these underlying non-POSIX-compliant file systems. For those that offer a POSIX-like interface, they lack distributed meta-data management. And for those few (e.g. Ceph, PAST, Circle) that also have distributed meta-data management, they fail to decouple data and meta-data management making the data locality difficult. Taking into account the above considerations, it is evident that what is really required is an architecture which will be optimized both for LAN and WAN environments, one that can easily work, at the same time, within a tight coupled infrastructure such as a computer cluster, as well as a more loose environment such as a Grid infrastructure. Extending the above, we can say that we need an architecture which will allow us to define, for the same data, meta-data in several different ways, depending on the network environments. IKAROS [4,5] supports multiple types of meta-data for the same data and is capable to scale its subsystems independently, based to the needs. Due to the fact that we use HTTP as the basic mechanism to build our architecture we can, easily, adopt protocols, techniques and utilities that have been proven to be very useful in a world wide web scale, capable to interact with millions of nodes.

3. An ad-hoc nearby storage based on IKAROS and Facebook

Given the current state of I/O and storage systems in petascale systems, incremental advances in individual aspects are unlikely to provide the required capabilities in exascale systems. I/O architectures, when designed as separate and independent components from the compute infrastructure, have already been shown not to be scalable as needed. Indeed, I/O has , traditionally, been considered as a separate activity which is performed before or after the main simulation or analysis computation, or even periodically for activities such as checkpointing, but still as a separate overhead. File systems, which have mainly been adapted from the legacy (sequential) file systems with overly constraining semantics, are not scalable. Traditional interfaces in file systems and storage systems, or even in some cases higher-level data libraries, are designed to handle the worst-case scenarios for conflicts, synchronization, and coherence and mostly ignore the purpose of the I/O by an application, which is an important source of information for scaling I/O performance when millions of cores simultaneously access the I/O system [7].

By placing higher bandwidth storage close to the computation using it, more aggregate bandwidth can be provided than with an external globally shared file system. By placing the storage close to the compute nodes, the I/O traffic can avoid interference with unrelated jobs. Globally shared file systems are sufficient for most existing HPC systems, but the bandwidth demands of exascale systems require a more scalable (in both bandwidth and cost) solution. However, when creating a nearby storage infrastructure, it is not only scalability in both bandwidth and cost what we seek, but we also need flexibility and dynamic management. That is why we must be able to form ad hoc nearby storage utilities, by taking on account a user or an application request, a domain or a Virtual Organization policy. By enabling our infrastructure to interact with other services, such as social networking services, we are able to create storage formations, on the fly, concluding to more efficient infrastructures. In this way we can scale the platform subsystems independently based to the demands.

We are using existing social networking services, such as Facebook, in order to dynamical manage, share and publish meta-data. In this way we do not have to build our own utilities for searching, sharing and publishing and in the same time we are enabling users to dynamically use the infrastructure, by creating ad hoc storage formations. This concludes in a model than can scale both up and down and so can provide more cost effective infrastructures for both large scale and smaller size groups and consortiums. This approach gives us the opportunity to build more efficient ad-hoc nearby storage: multiple instances of smaller capacity, higher bandwidth storage closer to the compute nodes. To achieve that we are using IKAROS and social networking services, currently Facebook. At the same time, we are not constrained by Facebook, because IKAROS is using a JSON (JavaScript Object Notation) format for populating meta-data, which is very common within Web 2.0 technologies. Individual users or groups can dynamically change their meta-data management schema based on their needs. We use Facebook as a use case to show how easily IKAROS can connect with existing, successful, infrastructures. Facebook is a social networking service that scales to millions of users.

3.1. IKAROS and Facebook interactions

IKAROS creates meta-data as JSON objects and keeps a local copy to the meta-data nodes. Based on the IKAROS architecture [5] each node is a peer and can act at the same time as a client node, as an I/O node and as a meta-data node. Additionally, we can provide our meta-data to a social networking service, such as Facebook, or to our custom meta-data management system. Figure 1 shows the interactions between IKAROS and Facebook.

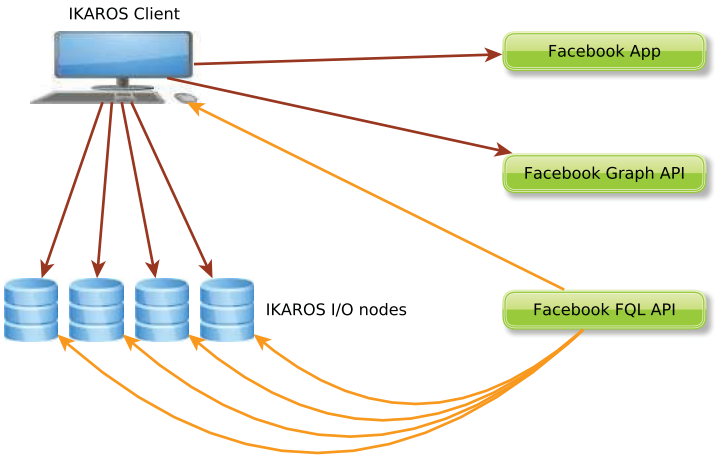

Figure 1. IKAROS and Facebook interactions

IKAROS interacts with the Facebook Graph API and the FQL(Facebook Query Language). The Graph API presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph and the connections between them. FQL enables users to use a SQL-style interface to query the data exposed by the Graph API. It provides some advanced features not available in the Graph API, such as using the results of one query in another one. In order to grant access to these components, we have to create our own Facebook App and provide to IKAROS the App ID/API Key and a user token or an App secret respectively (the Facebook Apps are using the oAuth security model). After that we have to provide, to this App, access to the following "Extended Permissions": "create note", "user notes" and "friends notes", from

Figure 2. IKAROS meta-data, JSON, object as a Facebook "note"

our profile. Finally, we can publish and share our meta-data as notes to our Facebook profile through the Graph API and query them through FQL. These two actions follow the IKAROS workflow [5] and we call them at the stages where an interaction is needed with the meta-data utility. Figure 2 shows an IKAROS meta-data, JSON, object as an Facebook "note". We are using the Facebook App "IFS M" with App ID/API Key: "513587365332568".

4. Experimental Results

In this section, we present the results of a number of experiments designed to measure the performance behavior of IKAROS. A set of experiments is performed comparing IKAROS with PVFS2 by using multiple clients requests under HPC IOR benchmark and MPICH2. With this set of experiments, we intent to extend our previous comparisons between IKAROS, PVFS2 and HDFS, based to low consumption, low technical specification devices [5]. Here we compare IKAROS with PVFS2 using storage devices/servers with more advanced technical specifications. We could not include HDFS to this set of experiments because Hadoop is not using a standard POSIX like interface and the HPC IOR benchmark tool cannot interact with it.

The system used for testing was the ZEUS computer cluster at the Institute of Nuclear and Particle Physics of NCSR "Demokritos". The ZEUS computer cluster is running applications mainly for the CMS - LHC CERN experiment and for the KM3Net consortium. It also participates to the NCSR Demokritos Mobile Grid Infrastructure [4], where the IKAROS module is a building block of the architecture. The work-flow of our applications forces us to run a huge number of embarrassingly parallel tasks at geographically distributed computing centers. For these types of applications it is crucial to focus on operations such as, concurrent writes and partially reads of the output, both locally and remotely. The ZEUS computer cluster provides an infrastructure composed by ten AMD Opteron CPU based systems, each with 8 CPU cores and 16 GB of RAM, four 800Mhz CPU based SOHO-NAS devices, each with 256 MB of RAM, 1000

Mbps Ethernet controller and 3 TB of storage capacity and ten 200Mhz CPU based SOHO-NAS devices, each with 32 MB of RAM, 100 Mbps Ethernet controller and 2 TB of storage capacity. The ZEUS computer cluster provides a 1000 Mbps full duplex link between the nodes and an 2.5 Gbps WAN connectivity.

4.1. Comparing IKAROS with PVFS2

We present the results of an comparison experiment between IKAROS and PVFS2 data transfer with a variety of file sizes, using an infrastructure composed by 10 AMD opteron CPU based systems, each with 8 CPU cores and 16 GB of RAM. In each run of the experiment we measured the time to perform each request and averaged the results over the course of the run. Each data point represents the average of 10 runs of the same experiment.

Figures 3 and 4 shows the performance of IKAROS compared to PVFS2 with multple clients on a READ operation. We use two block sizes (100MB and 1GB), the aggregate file size varies from 1000 MB to 40 GB. IKAROS has a slightly better performance in most cases.

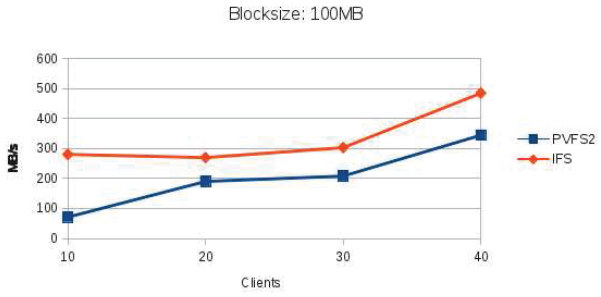

Figure 3. Comparing IKAROS with PVFS2, READ operation, Block size: 100MB, Max Aggregate size: 4GB, I/O node number : 10

Figure 4. Comparing IKAROS with PVFS2, READ operation, Block size: 1GB, Max Aggregate size: 40GB, I/O node number : 10

Figure 5. Comparing IKAROS with PVFS2, WRITE operation, Block size: 100MB, Max Aggregate size: 4GB, I/O node number : 10

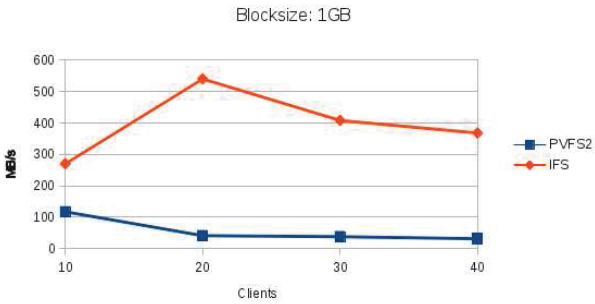

Figure 6. Comparing IKAROS with PVFS2, WRITE operation, Block size: 1GB, Max Aggregate size: 40GB, I/O node number : 10

Figures 5 and 6 shows the performance of IKAROS compared to PVFS2 with multple clients due to the WRITE operation. We use two block sizes (100MB and 1GB), the aggregate file size varies from 1000 MB to 40 GB. IKAROS outperforms in all cases. We must, also, highlight that

when we are using 1 GB block size, PVFS2 appears to break down. The above measurements clearly show that the IKAROS decentralized "HTTP GET" [5] design manages to overcome the performance problems which PVFS2 exhibts during WRITE operations, while at the same time performs equally well during READ operations.

5. Conclusions

Parallel file systems such as PVFS2, Lustre and GPFS are High-Performance systems, highly specialized, and often targeted to a single operating system and hardware platform. IKAROS on the other hand is open and supports heterogeneous systems. PNFS is closer to our work trying to be universal and transparent while focusing on interoperability taking advantage of the NFS wide spread implementations. Nevertheless, pNFS needs a considerable configuration effort requiring a kernel rebuild against the pNFS utilities. HDFS and Gfarm have a more WAN approach, which is one of our goals, but it seems that they can be more effective when the application work flow permits to associate the Computational Grid with the Data Grid, by running the computations at the file servers. IKAROS can be categorized as a system between a global shared file system and a "nearby" storage. By design, IKAROS is capable to increase or decrease the number of nodes of the I/O system instance on the fly, without bringing everything down or losing data.IKAROS is capable to decide the file partition distribution schema by taking into account requests from the user or an application, or a domain or Virtual Organization policy. In this way it is possible to form multiple instances of smaller capacity higher bandwidth storage utilities capable to respond in an ad-hoc manner. This approach, focusing on flexibility, will scale both up and down and so can provide more cost effective bandwidth for both large scale and for smaller size systems.

6. References

[1] I. Raicu, I. Foster, P. Beckman. Making a Case for Distributed File Systems at Exascale, LSAP11, June 8, 2011.
[2] T. Hey, S. Tansley, and K. Tolle. The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research 2009
[3] The International Exascale Software Roadmap," Dongarra, J., Beckman, P. et al., Volume 25, Number 1, 2011, International Journal of High Performance Computer Applications, ISSN 1094-3420. Exascale Nearby Storage, Cray Position paper
[4] Filippidis C., Cotronis Y.,Markou C.,(2012) Design and Implementation of the Mobile Grid Resource Management System. Computer Science, 13 (1). pp. 17-24. ISSN 1508-2806
[5] Filippidis C., Cotronis Y.,Markou C.,(2013) IKAROS: an HTTP-based distributed File System, for low consumption and low specification devices,Journal of Grid Computing
[6] I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, B. Clifford. Toward Loosely Coupled Programming on Petascale Systems, IEEE SC 2008
[7] The International Exascale Software Roadmap," Dongarra, J., Beckman, P. et al., Volume 25, Number 1, 2011, International Journal of High Performance Computer Applications, ISSN 1094-3420.