EPJ Web of Conferences **116**, 07001 (2016)
DOI: 10.1051/epjconf/201611607001

Using IKAROS as a data transfer and management utility within the KM3NeT computing model

Christos Filippidis[1,a], Yiannis Cotronis[2], and Christos Markou[1]

[1] NCSR Demokritos, Greece
[2] University of Athens, Greece

Abstract. KM3NeT is a future European deep-sea research infrastructure hosting a new generation neutrino detectors that – located at the bottom of the Mediterranean Sea – will open a new window on the universe and answer fundamental questions both in particle physics and astrophysics. IKAROS is a framework that enables creating scalable storage formations on-demand and helps addressing several limitations that the current file systems face when dealing with very large scale infrastructures. It enables creating ad-hoc nearby storage formations and can use a huge number of I/O nodes in order to increase the available bandwidth (I/O and network). IKAROS unifies remote and local access in the overall data flow, by permitting direct access to each I/O node. In this way we can handle the overall data flow at the network layer, limiting the interaction with the operating system. This approach allows virtually connecting, at the users level, the several different computing facilities used (Grids, Clouds, HPCs, Data Centers, Local computing Clusters and personal storage devices), on-demand, based on the needs, by using well known standards and protocols, like HTTP.

1. Introduction

Large-scale scientific computations tend to stretch the limits of computational power and parallel computing is generally recognized as the only viable solution to high performance computing problems. I/O has become a bottleneck in application performance as processor speed increases, leaving storage hardware and software struggling to keep up. Parallel file systems have be developed in order to allow applications to make optimum use of available processor parallelism. The most important factors affecting performance are the number of parallel processes participating in the transfers, the size of the individual transfers and of course the access patterns. The I/O access patterns are generally divided into the following subgroups [1]:

(1) Compulsory;
(2) Checkpoint/restart;
(3) Regular snapshots of the computation's progress;

[a] e-mail: cfjs@outlook.com

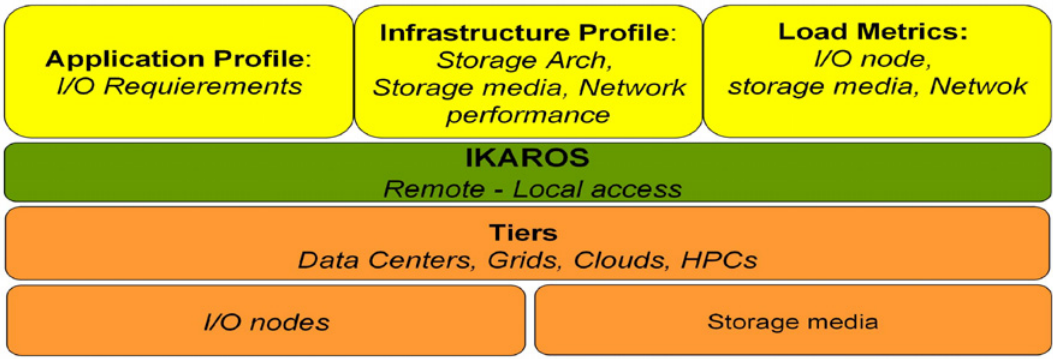

Figure 1. IKAROS Framework.

(4) Out-of-core read/writes for problems exceeding available memory;
(5) Continuous output of data for visualization and other post-processing.

The factors affecting performance are increasing if we consider the overall data flow (remote-local access) within an international collaborative scientific experiment, like KM3NeT. Such experiments will generate datasets that increase exponentially in both complexity and volume, making their analysis, archival, and sharing one of the grand challenges of the 21st century. These experiments, in their majority, adopt computing models consisting of Tiers (each Tier is made up of several computing centres and provides a specific set of services) and for the different steps of data processing (simulation, filtering, calibration, reconstruction and analysis) several software packages are utilized.

In order to confront the aforementioned problems and limitations we introduced IKAROS as a framework that enables us to create ad-hoc nearby storage formations and is able to use a huge number of I/O nodes in order to increase the available bandwidth (I/O and network) [3]. It unifies remote and local access in the overall data flow by permitting direct access to each I/O node, regardless of the tier. In this way we can handle the overall data flow at the network layer, limit the interaction with the operating system and minimize disk and network contention. This approach enables us to virtually connect, at user level, the several different computing facilities used (Grids, Clouds, HPCs, Data Centers, Local computing Clusters and personal storage devices), on-demand, based on the needs, by using well known standards and protocols, like HTTP.

2. IKAROS Framework

IKAROS [3] provides a dynamically coordinated I/O architecture for I/O accesses according to infrastructure topology/profile, load metrics, and the I/O demands of each application. At Fig. 1 we show an overview of the IKAROS framework: the input parameters that we feed at IKAROS (application I/O requirements, load metrics, infrastructure profile/topology) and the resources that IKAROS manages (I/O nodes and storage media) in all the tiers of the computing model. Currently we feed the appropriate input parameters to IKAROS manually. IKAROS allows data in a file to be striped across multiple disk volumes on multiple heterogeneous nodes and provides the utility for the storage system to access and transfer a data file in parts and in parallel mode, without a specific order according to a client request.

2.1 IKAROS overall data flow scenario (remote-local access)

In this scenario we analyse a data transfer from a remote storage server to the local parallel file system, which is actually a typical use case in a tiered system computing model, e.g: LHC and KM3NeT

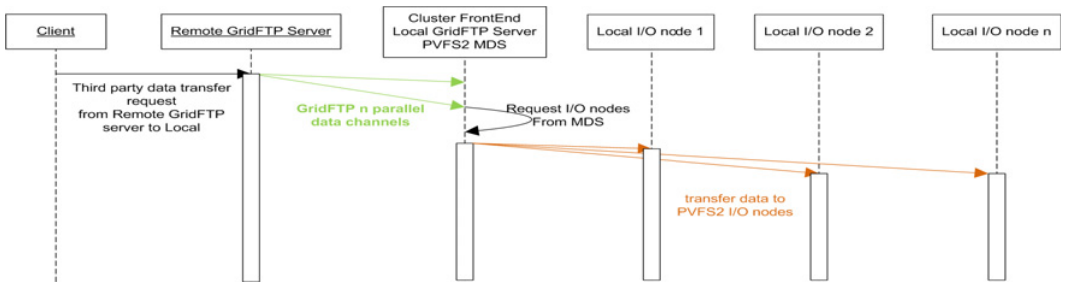

Figure 2. Overall data flow, GridFTP + PVFS2.

experiments. In Fig. 2 we show the implementation of this action by combining GridFTP and PVFS2. The client initiates a third party data transfer in order to transfer the data file from the remote GridFTP server to the local parallel file system in this case we are using PVFS2. We implement the local GridFTP server and the PVFS2 MDS at the frontend machine of the local computing cluster. Due to the client request, the remote GridFTP server starts sending the data file to the local GridFTP server by using N parallel data channels. The local GridFTP server moves the data chunks to the PVFS2 I/O nodes. The combined use of GridFTP and PVFS2, in order to implement the overall data flow, forces us to initiate many independent transfers incurring much overhead to set up and release connections. This approach can significantly impact performance due to the unnecessary network and disk contention.

The network and disk contention mainly appears due to the lack of proper coordination between the two systems, GridFTP/PVFS2. There is no guarantee that the remote access protocol and the local parallel file system, in this case GridFTP and PVFS2, have the same stripe size and the same stripe mapping.

A solution could be to use the GridFTP striped server technique, which is not exactly the case we analysed in Fig. 2. In the striped server scenario the data file will be striped at several remote GridFTP servers and the local I/O nodes will have to act both as PVFS2 I/O nodes and GridFTP servers. In this scenario, there is no guarantee that the GridFTP servers will stripe the data across the data nodes in the same sequence as PVFS2 does across the I/O servers.

In Fig. 3 we analyse, again, a data transfer from a remote storage server to the local parallel file system, but now we are using IKAROS for the overall data flow in order to avoid the unnecessary network and disk contention. Techniques like the reverse read implementation of the write request, introduced by IKAROS [3], combined with reverse HTTP tunneling techniques can help us provide proper coordination between remote and local access and achieve better performance.

This approach permits us to mainly route the data at the network level and minimize the usage of the operating system. In Fig. 3 the IKAROS client requests from the MDS system the available I/O nodes and decides the file partition distribution schema, based on the input parameters. Using this information, the client triggers each local I/O node, which participates in the transfer. Then each I/O node, on the basis of the previous trigger, requests from the remote storage server the corresponding data chunk. In this way we actually apply only coordinated parallel data transfers in contrast with the GridFTP/PVFS2 case in which we must manually synchronize the stripe size and the stripe mapping between them. In [5] we present measurements which shows that IKAROS approach outperforms the combined GridFTP+PVFS2 system.

3. KM3NeT–EGI flowchart (first use-case)

The KM3NeT computing model is based on the LHC computing model. The general concept consists of a hierarchical data processing system, commonly referred to as Tier structure and consists of several

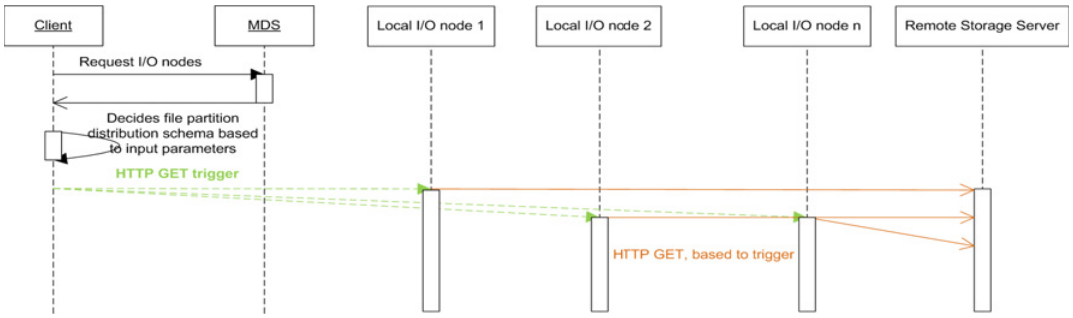

Figure 3. IKAROS overall data flow.

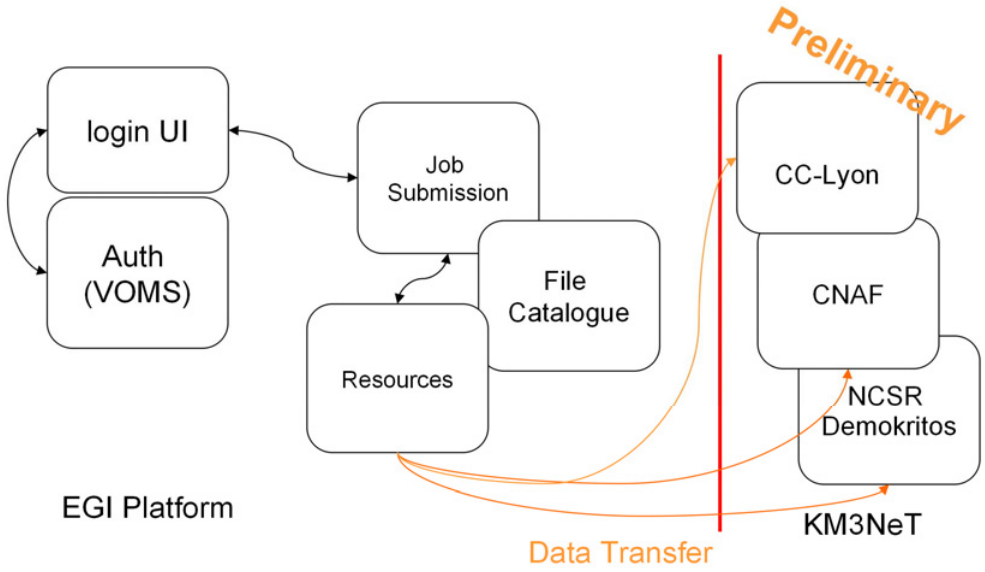

Figure 4. KM3NeT-EGI flowchart, first use-case.

Data Centers and Grid Computing infrastructures, like EGI. One of the main tasks is the efficient distribution of the data between the different computing centres – CC-IN2P3, CNAF will act as central storage, i.e. the resulting data of each processing step is transferred to those centres. The data storage at the centres is mirrored. In Fig. 3 we show the preliminary flowchart between KM3NeT and the resources hosted by EGI [4]. The data output for each process is transferred from the Grid worker node to the Grid storage element at the Napoli ReCaS Grid site and then to CC- IN2P3 at Lyon by using the GridFTP protocol or directly to user destination (laptop, local computer cluster, CC-Lyon) by using the eT-IKAROS [3] utility and the concept we demonstrated at Sect. 2.

References

[1] Schmuck, F., Haskin, R.: GPFS: a shared-disk file system for large computing clusters. In: Proceedings of the FAST 2002 Conference on File and Storage Technologies. IBM Almaden Research Center, San Jose, CA (2002)

[2] Dongarra, J., Beckman, P. et al. The International Exascale Software Roadmap," , Volume 25, Number 1, 2011, International Journal of High Performance Computer Applications, ISSN 1094-3420. Exascale Nearby Storage, Cray Position paper

[3] Filippidis, C., Cotronis, Y., & Markou, C. IKAROS: an HTTP-based distributed File System, for low consumption & low specification devices. Journal of Grid Computing, Springer, **11**, Issue 4 , pp 681–698 (2013)

[4] EGI Web site, `http://egi.eu` (2015)

[5] Filippidis, C., Cotronis, Y., & Markou, C. IKAROS: The IKAROS Metadata service as a Utility. 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), `http://doi.ieeecomputersociety.org/10.1109/UCC.2014.75`