evolve

Leading the Big Data Revolution

D2.1 Plan for converged hardware platform dimensioning

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

DDN Storage, BULL, IBM, FORTH, OnApp, Institute of Communications and Computer Systems, MemoScale, webLyzard technology, LOBA, Thales Alenia Space, Space Hellas, CybeleTech, Neurocom Luxembourg, MemEX, Tiemme SPA, Virtual Vehicle, AVL List GmBH, BMW AG, KOOLA

The Deliverable 2.1 is based on the contributions from the following persons.

Contributors

Name	Organization
Nam Nguyen	ATOS
Jean-Thomas Acquaviva	DDN
Per Simonsen	MEMOSCALE

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

Peer reviews

Name	Organization
Sotirios Xydis	Space Hellas
Angelos Bilas	FORTH

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

Revision history

version	date	reviewer
V0	05/20/ 2019	Huy Nam Nguyen
V1	05/22/ 2019	Jean-Thomas Acquaviva
V2	26/05/ 2019	Angelos Bilas
V3	05/31/ 2019	Sotirios Xydis
V4	05/31/ 2019	Jean-Thomas Acquaviva

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

Executive Summary

This document represents the deliverable D2.1 *Plan for converged hardware platform dimensioning*. It presents the plan for sizing the HPC architecture with its main features and the plan for how it will evolve with over the duration of the project based on technology availability, the work in the project, towards the final configuration that will be made available at the end of the project.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

Index of Contents

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

1 HPC Platform

The main objective of HPC is either to accelerate applications or to run problems that require shorter execution times or more compute, memory, and I/O resources than what is available on smaller systems. For this purpose, it is necessary for applications to include components that are parallel across multiple compute nodes. Therefore, it is important to understand how applications run across physically different servers and how to design and administer a system of discrete and potentially heterogeneous physical components.

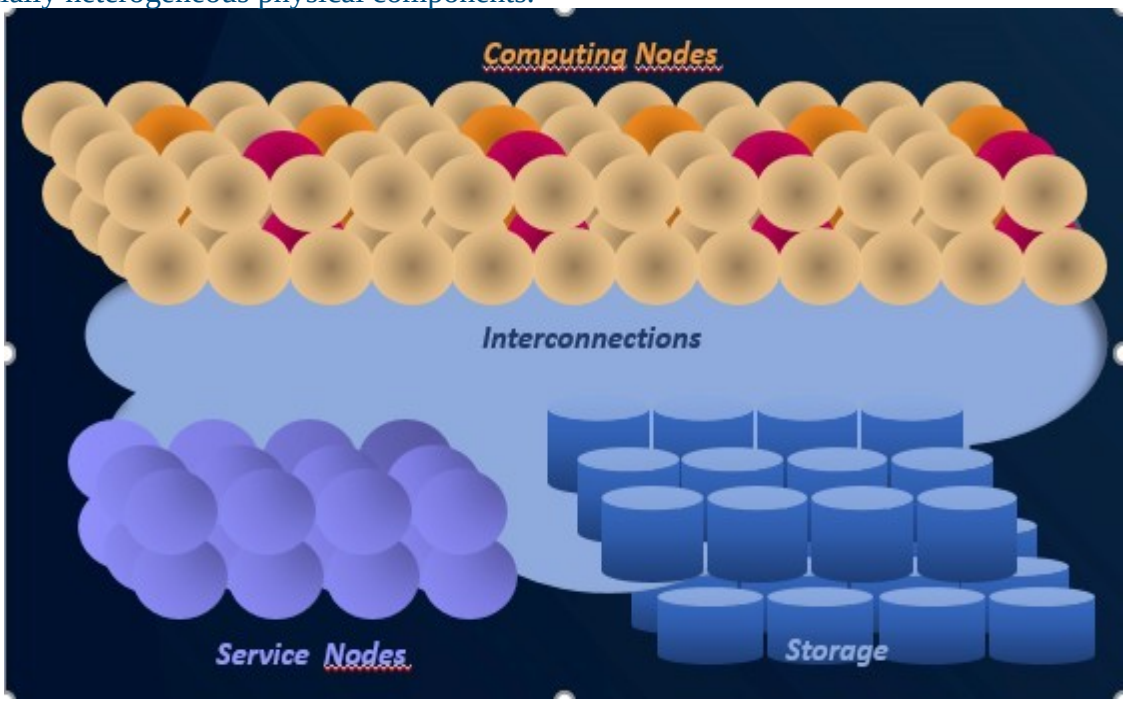


Figure 1: Typical HPC Architecture.

1.1 Hardware

Figure 1 illustrates the main aspects of high performance systems and which are considered in the design of the EVOLVE platform:

- Computing: The compute nodes characterized by a local interconnection of CPUs, eventually enhanced with acceleration technologies (GPU or FPGA) and addressing consistently the same address space. These computing devices are characterized by their components, e.g. number of cores with respect to CPU and GPU or the number of logic elements for FPGAs, together with their embedded memory. The organization and capacity of memory, e.g. shared memory, global memory, etc., and also plays an important role in the performance of compute nodes.
- Communication: The network to connect the nodes so they can communicate to share data, the state of the solution to the problem, and possibly the instructions that need to be executed. High performance systems, typically include a network dedicated to the computing aspect, along with other networks dedicated to management services, control operations, data storage, and I/O.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

Factors that come into play in the overall performance include the interconnection topology and communication protocol.

- Storage: While the local storage in each node can be as simple as an SDD device to hold the OS, the application and the data, clusters of storage devices should be made available to the whole system for the purpose of checkpoint, archiving, etc. In order to separate the performance and capacity aspects, the insertion of flash storage, e.g. DDN/IME burst buffer, that streamline the application I/O and perform data cache, represents a plus.

1.2 Software

On the software side, most distributions provide the basic tools for making a cluster work and for administering the tools to which it is necessary to add the tools and libraries for the parallel applications (e.g., compilers, programming models libraries and any specific additional libraries needed by the application).
The software is organized into layers: The first layer is the rudimentary systems software you need and nothing extra. The second layer adds administrative tools to make it easier to operate the cluster, as well as tools to reduce problems when running parallel applications. The third layer adds more sophisticated cluster tools and adds the concept of monitoring, so you can understand what's happening. Next, we discuss these layers in more detail.

1.2.1 Software layer 1: Configuration

The first layer of software contains the minimum software to run parallel applications, i.e. the OS together with a set of MPI libraries such as Open MPI or MPICH. These are the libraries necessary for creating parallel applications and running them on the cluster. In order to run MPI, SSH is another piece of software needed in this basic layer.

1.2.2 Software layer 2: Architecture and Tools

The second layer of software provides tools to assist the administration of cluster problems of the kind: Running commands on each node , Configuring identical nodes , Alignment of time on each node, Job scheduling and resource management.
These issues arise particularly when trying to scale the system size.

1.2.3 Software layer 3: Deep Administration

The third level of tools goes deeper into HPC administration and begins to gather more information about the cluster, so you can find problems before they happen. Examples of these tools are: Cluster management tools (e.g. resources reservation, task tracking, ec.), Monitoring tools (e.g. Bull Performance toolkit including HPC toolkit, MPIAnalyzer, etc.), Environment Modules (an environment consists of a set of compatible products eg. Fortran compiler/C compiler, libraries, debugger, etc.) and Multiple networks.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

2 Design/Evolution of the Evolve Platform

2.1 Design/Evolution Strategy

In order to support development in other WPs/Tasks of EVOLVE at the same time, we adopt a pipeline approach to the design and assembly of the hardware platform: A preliminary V0 version of the platform was made available to the partners, already from M0; afterwards, a V1 version of the platform will be built for M6 and this version will evolve continuously with upgrades of the different computing/memory/storage components as described in the previous paragraph. Table 1 illustrates this overall schedule.

2.1.1 Evolution of hardware

It is important to note that the evolution strategy of the platform hardware is dictated by the following considerations:
In order to keep this evolution as smooth as possible and hence to minimize disruption to partner developments it has been decided to keep constant the interconnection network which represents the backbone of the HPC system, i.e. Infiniband FDR Interconnect and Interconnection Topology.
Improvement of storage will be done mainly on the basis of capacity increase, while the storage architecture (scratch storage, SSD storage, DDN/IME burst buffer) and APIs will be kept unchanged, as much as possible.
Continuous increase of performance will be performed via the upgrade of components, such as CPUs (#cores, frequency) together with the organization and amount of their associated memory (levels of cache, memory bus, etc.)
Performance will be improved also via the use of new acceleration technologies (GPU and FPGA). This introduction will be done in discrete mode (communication via PCIe) and hence will not change drastically the programming model throughout the development cycle.

2.1.2 Evolution of software

From its first version, the Evolve platform is able to support a large range of programming models varying from OpenMP, MPI, OpenCL, CUDA, etc. and hence the evolution of the basic software layer will be mainly oriented towards upgrading performance and adding specific features in conjunction with new components.
Finally, it is important to mention that the dynamic evolution of the platform will be done according to the profiling of pilots in order to establish a priority of criteria in terms of acceleration technologies, local memory capacity, etc. To support this purpose, a set of monitoring tools (e.g. PMC, MPI monitoring) will be provided to the partners to accomplish their profiling task.

2.2 Evolution Plan

The following Table 1 and Table 2 describe the main milestones in the evolution of the Evolve platform hardware. The data are indicative and may be subjected to changes related to specific requirements or constraints.

2.2.1 Computation evolution plan

Version	Time	Main Features	Improvements
V0	M0	Intel/SNB, Nividia/P10	Preliminary version

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

V1	M6	CPU Intel/SNB [O(TFlops)] GPU Nvidia/P40 Memory Hierarchy: [O(TB)]	Upgrade of GPU, Integration of DDN/IME
V2	M18	CPU Intel/SKL, FPGA Intel/Stratix10	CPU upgrade to Intel/SKL Integration of FPGA
VF	M30	CPU Intel/SKL, GPU Nvidia/V100, FPGA Intel/Stratix10	Increase of CPU performance, Memory capacity and update of acceleration technologies

Table 1: Plan for the Evolution of the Computing Hardware.

2.2.2 Storage Evolution plan

Version	Time	Main Features	Improvements
V0	M0	Lustre parallel server	Parallel file system performance
V1	M6	IME 120: 4 servers with 12 SSD Software IME1.2	Storage tiering with Flash layer
V2	M18	IME 240 4 servers with 24 NVMe Software IME1.4	Faster tiering increased capacity. Support for container technology
VF	M24	IME 240 4 servers with 24 NVMe Software IME 1.4 + Evolve features	Advanced monitoring capabilities. Native support of the Evolve software stack. Compression and encryption technologies.

Table 2: Plan for the Evolution of Storage.

Regarding storage the evolution of the platform is twofold. First, from a raw hardware perspective, with the shift from the SSD to more advanced NMVe technology. NMVe allows higher bandwidth (typically a factor of 2) and more importantly a latency improved by a factor of 10x.
The shift from IME 120 to IME 240 will thus improve the per device bandwidth, reduce the latency which can be of high interest for workload dominated by sparse read access. The applications used within the scope of Evolve may take large benefit from this improve read capabilities. Additionally, the IME 240 host more devices, each of them being of larger capacity (1TB) than the 256 GB provisioned in the IME 120.
The second improvement is on the internal software stack of the storage. As IME is mostly a software defined storage, the improvements from one software version to the other will impact significantly the storage system. From the initial 1.1 to the 1.4, fault tolerance capabilities will be greatly improved, prefetch and data movements will be improved considerably as well. An important effort will be made on the monitoring capabilities of the system and its integration with the job scheduler and more generally the resource allocation middleware.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

3 Tiered Architecture

3.1 Overview of a Tiered architecture

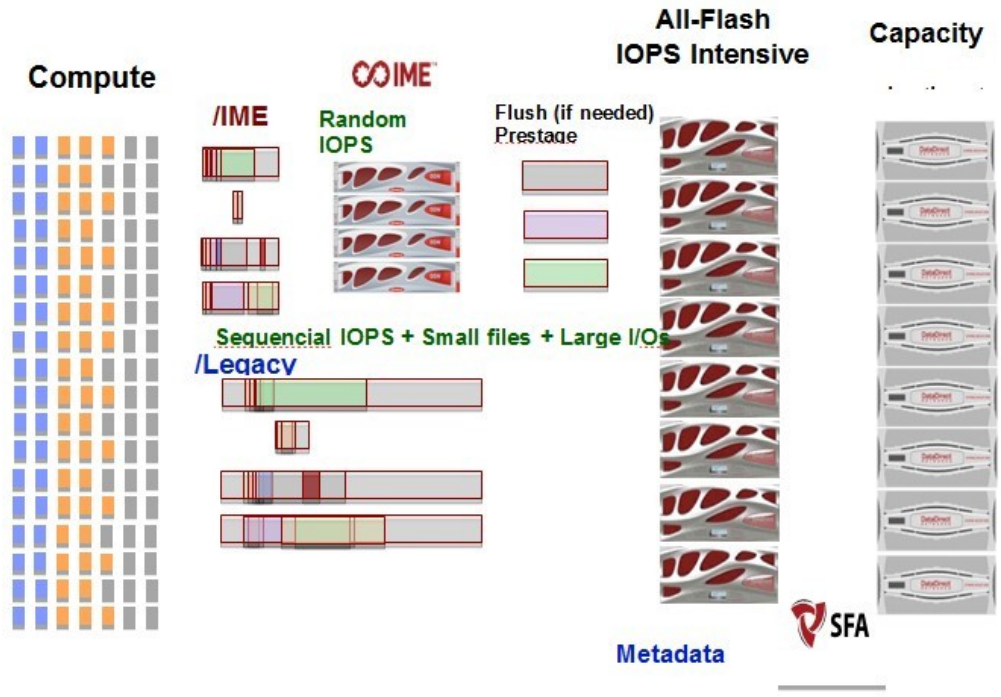


Figure 2: Tiered storage architecture of EVOLVE.

At the opposite of a per component approach in Evolve the storage is envisioned as a tiered architecture (Figure 2). Depending on the workload characteristic a distinct hardware component will be involved in order to provide the performance target.

3.2 Key Benefits of Tiering

Tiering implies an additional complexity at the system architecture, from an infrastructure stand point but also for user since data movement and location become more complex. The benefit is illustrated in Error: Reference source not found. These measurements have been realized on the exact same platform, a system using 24 NMVe with 2 EDR link. The write access pattern is characterized by three metrics:

1. I/O size, are the write access made by small or large chunck. Here Large stands for 1MB, medium 64 KB and small 32 KB
2. I/O predictability, is the pattern random or sequential
3. I/O sharing, is the I/O request targetting a file also accessed by other processes. In this case we only consider share everything (single shared file) or share nothing (each processes write in its own file)

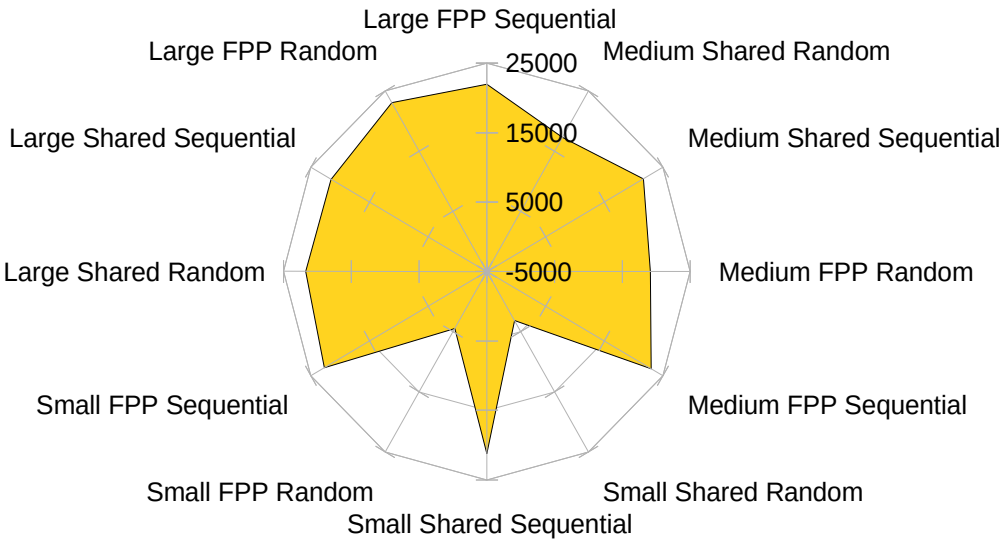
This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

The measurements on the Lustre file system shows that despite the usage of fast NVMe the system is sensitive the both the sharing pattern and to the predictability of the IO. Random small I/O present an efficiency of around 1%. Therefore, while Lustre is the system used for extreme scale up to hundred of Petabyte, it appears that the capacity component of the storage system is not suitable to handle all kind of workloads.

Figure 3: Performance space for typical I/O subsystems.

Running the exact same experiment on an IME system display a different picture. The surface of efficiency of IME is much better, showing the resilient of this tier in respect of complicated I/O patterns.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061



However, IME is flash-based, hence not very cost-effective for large capacities. Thus, EVOLVE implements a tiered architecture, as sketched in Figure 4, which offers a fast and resilient layer of storage based on flash technology, where the capacity tier is 'shielded' from complicated I/O.

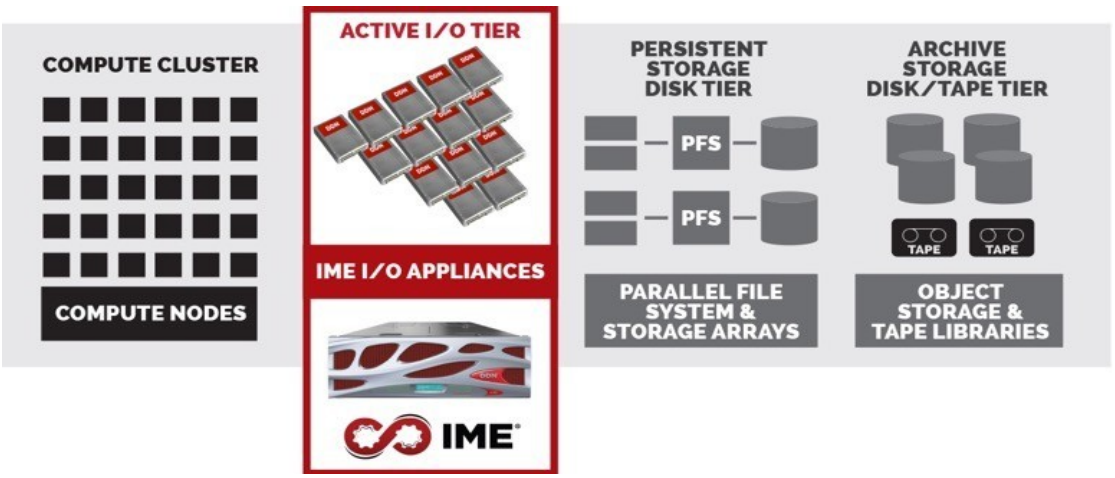


Figure 4: EVOLVE tiered storage architecture.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

4 Interacting with IME

The information provided in this section are an introduction to IME usage. More detailed process and explanations can be found in the reference documents:

a) IME developer guide
b) IME installation and administration guideline

Both reference document are available on the Evolve-H2020 website, in the foloder dedicated to WP2: https://consortiumarea.evolve-h2020.eu

4.1 Evolve I/O Subsystem

The I/O subsystem that is designed for EVOLVE is shown in Figure 5.

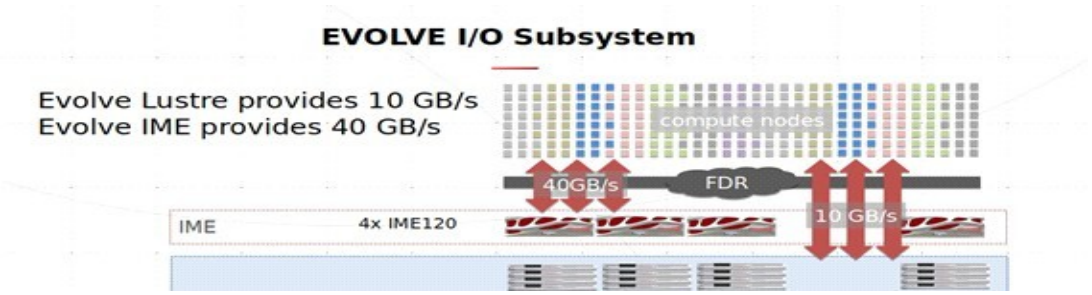


Figure 5: EVOLVE I/O subsystem.

4.2 Selecting the right file system

As discussed in Section Evolution Plan, depending on the I/O pattern the performance response of a file system varies. A general guideline being that Lustre is highly sensitive to both random accesses (either read or write) and write on shared files and hence, for these kind of patterns IME should be evaluated as an alternative to Lustre.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

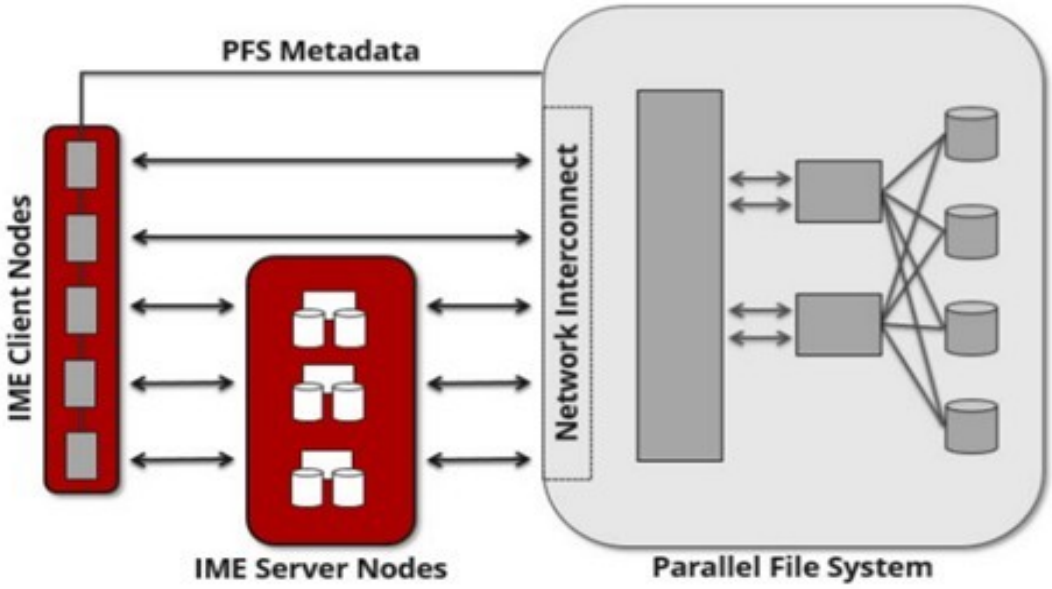


Figure 6: Filesystem configuration for EVOLVE IME.

One important conceptual point is that IME is an I/O accelerator not a file system *per se*. Metadata operations, such as file creation, are still handled by Lustre, even if they go through the IME layer (Figure 6). For metadata traffic IME is a proxy towards Lustre.
Access to the storage layer of IME is simple. IME appears as an additional mountpoint in the compute node. The following example (Figure 7) is taken from a larger IME configuration but otherwise similar to what is used in the Evolve testbed. The IME mount point appears listed as **imefs** and can be used as any other mount point.

```
[ddnsupport@mon02 IO500]$ df -h
Filesystem                      Size  Used Avail Use% Mounted on
/dev/sda3                       273G  216G   44G  84% /
tmpfs                            63G  136K   63G   1% /dev/shm
/dev/sda1                       488M   33M  431M   7% /boot
,192.168.156.30@o2ib:/scratch   2.8P  2.2P  664T  77% /scratch
,192.168.156.30@o2ib:/seq       1.2P  892T  240T  79% /seq
192.168.160.104:/home/          3.4P  2.7P  775T  78% /home
192.168.160.101:/data/          5.3P  3.9P  1.4P  74% /data
imefs                           2.8P  2.2P  664T  77% /ime
```

Figure 7: IME-based mount point in a typical server.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

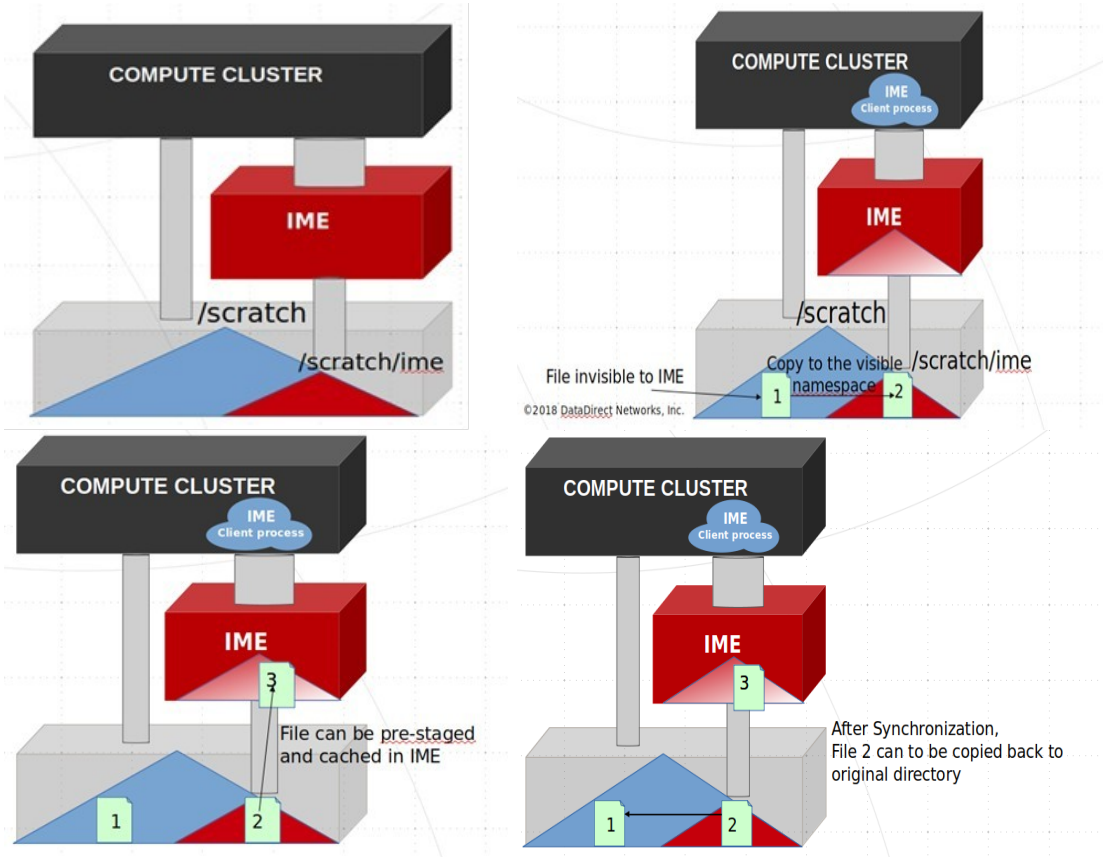


Figure 8: Configuration options for IME storage acceleration.

4.3 Mountpoint and data visibility

Imefs allows accelerated access to a portion of the parallel system name space. The fraction of the accelerated name space depends on the way the IME servers have been configured. In Figure 8 all data accesses made to the sub-tree /scratch/ime are intercepted by the IME client and accelerated due to the storage of the Flash native IME servers.

4.4 Controlling data movements

Data movement can be monitored and controlled with command line tools. The system supports an array of commands, the most generic one and easier to use being **ime-ctl**. Man pages are available for a full description of the command, however a simple description is:

- Express data movement, from backend file system to IME, from IME to backend file system. Where stage-in means coping data from backend file system to IME. Stage-out or Synchronization mean copying data out of IME to backend file system, and purge means removing data from IME without copy to the backend file system.
- Monitor file status, list the file cached in the IME and their status: copy of the back-end file system or newer version of an existing file of the file system, number of Byte of delta between the two versions and so on.
- Configure IME, several performance knobs are exposed to the end-user in order to help the optimization of a specific workload.

These commands can be used on-line, scripts or integrated within an epilogue / prologue of the job scheduler.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

5 Interacting with Containers

The Evolve platform allows end user not only to run traditional HPC codes as detailed in the section 2 but also to use containers.

6.1 Working around root access

Container is a technology coming from the Cloud where root access is not as strictly controlled than in HPC. Therefore most if not all Docker command implies a root access. In order to work around this constraints and still allow end-user to run Docker command without compromising root access, the platform support a Docker groups. All Evolve partners are added to the docker group. Only the command prefix by # need to be run as root.

```
= Create docker groups
# sudo groupadd docker
= Adding user to the Docker group
#sudo gpasswd -a my_user docker
Adding user my_user to group docker:
=  Refresh groups belonging to take modification into account
% groups
my_user adm cdrom sudo dip plugdev lpadmin sambashare
%newgrp docker
%groups
docker adm cdrom sudo dip plugdev lpadmin sambashare my_user
```

6.2 Fetching Docker image

In the container world the code is not built locally on the platform, it is considered as portable hence built on a remote platform and fetch for the execution phase. In such a case, a remote access from the node is mandatory since the container will be refreshed just prior to its execution. In order to implement this workflow application partners need to set-up a docker repository. The repository is accessed dynamically to fetch the latest version of the image. With respect to classic cluster configuration, it means that the executing node need to have access to the internet. The example bellow is taken from the CybeleTech application:

```
% docker login -u teddy.debroutelle registry.cybeletech.fr
Password:
WARNING! Your password will be stored unencrypted in /home/user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
% docker pull registry.cybeletech.fr/cybeletech/image/image:dev
dev: Pulling from cybeletech/image/image
Digest: sha256:51d8ce7862eaf74f9aab4a3209de45a401b5553046a986521254117f8dbc85fb
Status: Image is up to date for registry.cybeletech.fr/cybeletech/image/image:dev
```

Once the code has been executed, it is considered as good practice to remove the configuration file created during the login stage.

```
% docker logout  registry.cybeletech.fr
Removing login credentials for registry.cybeletech.fr
```


This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

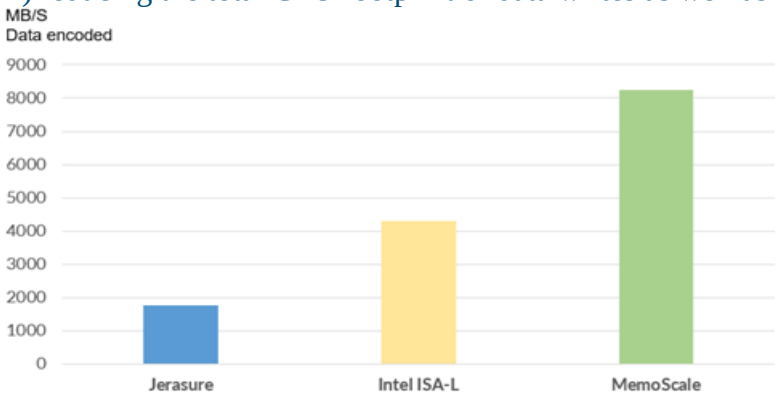
```
% rm /home/user/.docker/config.json
```


This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

6 Optimizing fault tolerance, throughput and storage capacity

6.1 Erasure coding and fast general compression

The storage software architecture will be upgraded with highly optimized libraries for compression and erasure coding functionality(for fault tolerance). MemoScale's fast compression for floating point and general data will be used to explore the potential of further increase the speed of the data transfer between compute nodes and the storage servers by compressing the data before transmissions.
The MemoScale erasure coding library will provide up to 2x improvements in erasure coding encoding speed per CPU core compared to the fastest alternative erasure coding library (Intel ISA-L) reducing the total CPU footprint of data writes as well as increasing the throughput.



Erasure coding encoding benchmarks. Processor used: Intel Xeon E5-2676 v3 2.4 Ghz. 10 data blocks, 4 redundancy blocks, 4 MB block size. Jerasure and Intel ISA-L are alternative erasure coding libraries.

Figure 9: Performance efficiency of MemoScale erasure coding.

7.2Satellite Image Compression

Three of the pilot users (Thales, Space Hellas, and Cybeletech) will process and store satellite images on the platform. The amount of storage capacity needed for satellite images can potentially grow to a significant size during the course of the project. A single satellite image is up to almost 1 GB in size, and the pilot users plan to analyze long time series of images over increasingly larger geographical areas eventually covering the whole of Europe. MemoScale will investigate if their lossless image compression technology can be used to compress satellite images further to increase the amount of satellite images which can be stored on the platform with between 37% - 275%.

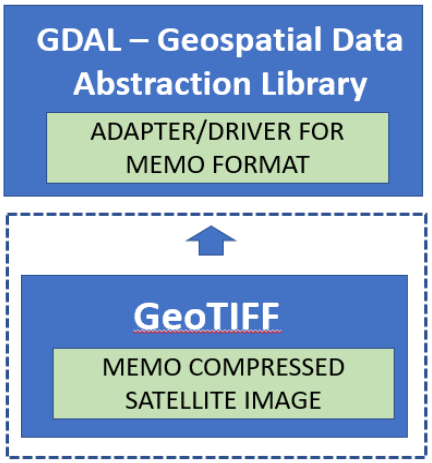


Figure 10: MemoScale adapter for image compression.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

Satellite images are stored inside an image container called GeoTIFF and are read with a library called GDAL (Geospatial Data Abstraction Library). MemoScale will replace images stored inside of the GeoTIFF file format with highly compressed images. To enable the users of the platform to use the compressed format, MemoScale will provide a driver/adapter for the GDAL library so that users can access the compressed images in a transparent way without interference to their work flow (Figure 10). Depending on the compression/decompression speed needed by users in the project, MemoScale will consider accelerating the compression algorithm with FPGA or GPU.

6.2 Encryption

MemoScale, DDN and Atos/Bull are researching feasible ways of implementing encryption on the platform given the existing storage infrastructure. Encryption needs to be implemented in a way which does not have a large negative impact on performance of the HPC system. Currently, it looks like implementing encryption on the capacity storage tier is the most promising approach.

8 Conclusions

The evolution plan described in this document takes into account the state-of-the-art of HPC components and their availability and their contribution to the optimal support of the pilots proposed in the project. This plan is mostly conservative and hence takes only limited risks in order to avoid disruptions in the development of partners. It is worth to mention that we have paid a specific attention to the storage aspect in order to increase the overall performance. Obviously this plan must be in line with the profiling and implementation tasks of those use cases.

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061

Consortium

DDN Storage
www.ddn.com

BULL
www.atos.net

IBM
www.ibm.com

FORTH
www.ics.forth.gr

OnApp
www.onapp.com

Institute of Communications and Computer Systems
www.microlab.ntua.gr

MemoScale
www.memoscale.com

webLyzard technology
www.weblyzard.com

LOBA
www.loba.pt

Thales Alenia Space
www.thalesgroup.com

Space Hellas
www.space.gr

CybeleTech
www.cybeletech.com

Neurocom Luxembourg
www.neurocom.eu

MemEX
www.memexitaly.it

Tiemme SPA
www.tiemmespa.it

Virtual Vehicle
www.v2c2.at

AVL List GmBH
www.avl.com

BMW AG
www.bmw.com

KOOLA
www.koola.io

This project has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 825061