DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING AND SCIENCE

DOCTORATE SCHOOL IN INFORMATION AND COMMUNICATION TECHNOLOGIES

XXXI CYCLE

UNIVERSITY OF MODENA AND REGGIO EMILIA
"ENZO FERRARI" ENGINEERING DEPARTMENT

Ph.D. DISSERTATION

A distributed HPC infrastructure to process very large scientific data sets

Candidate: Giuseppe Fiameni

Advisor: Prof. Sonia Bergamaschi

Director of the School: Prof. Sonia Bergamaschi

DOTTORATO DI RICERCA IN
COMPUTER ENGINEERING AND SCIENCE

SCUOLA DI DOTTORATO IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXXI CYCLE

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA
DIPARTIMENTO DI INGEGNERIA "ENZO FERRARI"

TESI PER IL CONSEGUIMENTO DEL TITOLO DI DOTTORE DI RICERCA

Una infrastruttura di calcolo distribuita per il processamento di dati scientifici di grandi dimensioni

Tasi di: Giuseppe Fiameni

Relatore: Prof.ssa Sonia Bergamaschi

Il Direttore della Scuola: Prof.ssa Sonia Bergamaschi

Keywords:

High Performance Computing

Digital Infrastructure

Data Processing

Interactive Computing

Abstract (English)

Access to research infrastructure at national and international level is becoming ever more important in an increasing number of scientific fields. Many research activities as well as infrastructures play an important role for both the business and the public sector. They are important components to meet the majority of scientific challenges of our time. Advancements into climate changes study, sustainable and safe societal development, public health, food production, democracy, labour market, etc. also require advanced services and resources. Thus, the goal of this thesis work is twofold; present a distributed HPC infrastructure realized though the federation of different compute and data resources across Europe; provide insights about emerging technologies being crucial for the processing of large data sets.

As starting point, a set of common technical specifications has been derived to provide a high-level specification of the overall architecture and give details of key architectural elements that are essential for realizing such infrastructure, including scientific cases, new technologies and new processing methodologies. The work has been mainly fuelled by the need to provide a scalable solution, harness new memory technologies, such as those based on non-volatile chips, provide easy access to data, and improve user experience by fostering the convergence between traditional High Performance and Cloud Computing utilization models. Nowadays the main access model for large scale HPC systems is based on the scheduling of batch jobs. This approach is not connected with a requirement from the computational science community, but it reflects the predominant issue in the management of the HPC systems: the maximization of resource utilisation. Conversely, the situation differs

when taking into consideration personal workstations or shared memory servers, where timesharing interactive executions are the norm. Our design, which proposes a new sort of paradigm called “Interactive Computing” refers to the capability of a system to support massive computing workloads while permitting on-the-fly interruption by the user. The real-time interaction of a user with a program runtime is motivated by various factors, such as the need to estimate the state of a program or its future tendency, to access intermediate results, and to steer the computation by modifying input parameters or boundary conditions. Within the neuro-science community, one of the scientific cases taken in consideration in this work, most used applications (i.e. brain activity simulation, large image volume rendering and visualization, connectomics experiment) imply that the runtime can be modified interactively so that the user can gain insight on parameters, algorithmic behaviour, and optimization potentials. Besides that, in many application fields, the computational scientists are starting to use interactive frameworks and scripting languages to integrate the more traditional compute and data processing application running in batch, e.g. the use of R, Stata, Matlab/Octave or Jupyter Notebook just to name few. The work has been supported by Human Brain Project (www.humanbrainproject.eu) and Cineca (www.hpc.cineca.it), the main supercomputing centre in Italy.

Sommario (Italian)

L'accesso ad infrastrutture di ricerca a livello nazionale e internazionale sta diventando sempre più importante in un numero crescente di settori scientifici. Molte attività di ricerca svolgono un ruolo importante sia per le imprese che per il settore pubblico. Sono componenti importanti per affrontare la maggior parte delle sfide scientifiche del nostro tempo. I progressi compiuti nello studio sui cambiamenti climatici, lo sviluppo sociale sostenibile, la sicurezza, la salute pubblica, la produzione alimentare, la democrazia, il mercato del lavoro, ecc. richiedono la disponibilità di servizi e risorse avanzate. L'obiettivo di questo lavoro di tesi è duplice; presentare un'infrastruttura distribuita HPC realizzata attraverso l'aggregazione di diverse risorse di elaborazione e dati presenti in Europa, fornire informazioni sulle tecnologie emergenti che sono fondamentali per l'elaborazione di grandi quantità di dai.

Come punto di partenza, è stata derivata una serie di specifiche tecniche comuni per fornire una specifica di alto livello dell'architettura generale e fornire dettagli sugli elementi architettonici chiave essenziali per la realizzazione di tali infrastrutture, inclusi casi scientifici, nuove tecnologie e nuove metodologie di elaborazione. Il lavoro è stato principalmente alimentato dalla necessità di fornire una soluzione scalabile, sfruttare nuove tecnologie di memoria, come quelle basate su chip non volatili, fornire un facile accesso ai dati e migliorare l'esperienza degli utenti promuovendo la convergenza tra le tradizionali prestazioni elevate e il cloud Modelli di utilizzo del calcolo. Al giorno d'oggi il modello di accesso principale per i sistemi HPC su larga scala si basa sulla pianificazione dei lavori batch. Questo approccio non è collegato a un requisito della comunità scientifica computazionale, ma riflette il problema predominante nella gestione dei sistemi HPC: la massimizzazione dell'utilizzo delle risorse. Viceversa, la

situazione si differenzia quando si prendono in considerazione workstation personali o server di memoria condivisi, dove le condivisioni interattive di timesharing sono la norma. Il nostro design, che propone un nuovo paradigma chiamato "Interactive Computing" si riferisce alla capacità di un sistema di supportare massicci carichi di lavoro informatici consentendo al tempo stesso l'interruzione immediata da parte dell'utente. L'interazione in tempo reale di un utente con un runtime del programma è motivata da vari fattori, come la necessità di stimare lo stato di un programma o la sua tendenza futura, di accedere a risultati intermedi e di indirizzare il calcolo modificando i parametri di input o il limite condizioni. All'interno della comunità neuroscientifica, uno dei casi scientifici presi in considerazione in questo lavoro, le applicazioni più utilizzate (es. Simulazione dell'attività cerebrale, rendering e visualizzazione di grandi volumi di immagini, esperimenti di connettività) implicano che il runtime possa essere modificato interattivamente in modo che l'utente può acquisire informazioni su parametri, comportamento algoritmico e potenziali di ottimizzazione. Oltre a ciò, in molti campi applicativi, gli scienziati computazionali stanno iniziando a utilizzare framework interattivi e linguaggi di scripting per integrare l'applicazione di elaborazione dati e di elaborazione più tradizionale in esecuzione in batch, ad es. l'uso di R, Stata, Matlab / Octave o Jupyter Notebook solo per citarne alcuni. Il lavoro è stato supportato da Human Brain Project (www.humanbrainproject.eu) e Cineca (www.hpc.cineca.it), il principale centro di supercalcolo in Italia.

List of Publications

- Fiameni G., Zanella R., Rorro M., “***A performance study of machine and deep learning frameworks on Cineca HPC systems***”, Parallel Computing Conference, 12-15 September 2017, Bologna, Italy
- Fiameni G., Carmona Manuel Juan, Graziano G. “***Big-Data in Climate Change Models - A novel approach with Hadoop MapReduce***”, The 4th International Symposium on Big Data Principles, Architectures & Applications, BDAA2017, 17-21 July, Genoa, Italy
- Fiameni G., Zhu S. Simonini G., Bergamaschi S., “***SOPJ: A Scalable Online Provenance Join for Data Integration***”, The 4th International Symposium on Big Data Principles, Architectures & Applications, BDAA2017, 17-21 July, Genoa, Italy
- Fiameni G., D'Antonio M., D'Onorio De Meo P., Cacciari C., “***Characterization of genomic data using graph databases***”, Parallel Computing Conference, 12-15 September 2017, Bologna, Italy
- Fiameni G., Caraceni S., Caroli C., Carpenè M., D’Antonio M. et al “***I-Media-Cities, a searchable platform on moving images with automatic and manual annotations***”, International Society on Virtual Systems and MultiMedia, Dublin 2017.

Table of Contents

List of Figures

Chapter I.

Outline

This dissertation explores various topics concerning digital infrastructures, high performance computing technologies and future trends. It is organized in a cumulative style. The major findings are presented in form of peer-reviewed conference and publications that I contributed to as well as developments done as part of my job position at Cineca[1]. For a complete list of all publications, however, please refer to the "List of Publications".

The following chapters will provide a brief summary of the results as follows.

- **Chapter II** – introduces the work.
- **Chapter III** – presents the architecture of the Fenix infrastructure, a federation of resources and services to serve the European neuroscience community as part of the Human Brain Project[2]. I reported the requirements that led to the design of the infrastructure as collection of use cases, list of services, and resources made available to the community. The design and the realization of the infrastructure is the result of a joint collaboration activity funded by the European Commission under the ICEI (Interactive Computing E-Infrastructure for the Human Brain Project) project. This initial chapter helped me introduce the successive parts of this thesis which focus on HPC trends, emerging Big Data technologies, and Deep Learning workloads.

[1] www.cineca.it
[2] www.humanbrainproject.eu

- **Chapter IV** – reports about the trends that are pushing the evolution of HPC systems and which challenges are foreseen to make further steps. The chapter also includes a study concerning the adoption of virtualization technology, based on container, to support HPC workloads.
- **Chapter V** – presents a brief summary of the Quantum Computing technology as it is expected to revolutionize the way we conceive computing.
- **Chapter VI** — covers Deep Learning frameworks and networks. It also includes benchmark results obtained using Cineca HPC systems using synthetic benchmark.
- **Chapter VII** – copes with the adoption of the Apache Hadoop framework to solve a use case in the field of climate change. Furthermore, it reports about the use of Apache Spark into a HPC environment.
- **Chapter VIII** —concludes the thesis and presents perspectives for future research opportunities.

Chapter II.

Introduction

Access to research infrastructure at national and international level is becoming ever more important in an increasing number of scientific fields. This work summarises overarching tendencies in relation to the needs of European research community. The starting point is the needs that are driven by basic and researcher initiated research. At the same time, it should be emphasised that many research activities as well as infrastructures play an important role for both the business sector and the public one. Infrastructures are also important components to meet the major scientific challenges of our time. Research of relevance to societal challenges related to climate changes, sustainable and safe societal development, public health, food production, democracy, labour market, etc. also require advanced services and resources.

The observation, the measurement, the experimentation, the supercomputing, the storage and sharing of data, all suppose to use big instruments with technical performances beyond the existing ones and integrating interdisciplinary as a source of innovation. These tools constitute a mandatory condition for future discoveries as well as the product of the past scientific and technological advances. In parallel to these major programs, a number of instruments shared amongst many actors on various sites have been developed in recent years: new modes of microscopy and imaging, new high throughput screening devices, virtual experiments, social, environmental and health databases, corpus of digitized texts with their operating tools.

The needs for research infrastructure – large research facilities, laboratory environments, experimental workshops, complex digital research systems and comprehensive databases – are increasingly rapidly within most research fields. Technical developments and ever more complex scientific questions are simultaneously driving this development forward. This applies not only within environment and climate research, humanities, social sciences and major parts of medical research. Fundamental knowledge about our universe, the characteristics of materials, the function of cells and internal characteristics of matter demands advanced instruments.

Complex questions also require data and observations from several sources to be combined. Moreover, a common feature of all research fields is that the need to store, transfer and analyse large amounts of data is increasing very rapidly. In many cases, the development means that barriers between research disciplines is breaking down, and that the need for international collaboration is increasing. Advanced research infrastructure also constitutes a resource for industry, and it is in many cases a prerequisite for collaboration between industry and academia.

Traditionally, the needs for calculation by high-performance computers, accessible via peer-review process, as that provided by PRACE (Partnership for Advanced Computing in Europe)[3], have dominated the use of e-infrastructure, with strong research within fields such as materials sciences and fluid mechanics. Growing needs are driven by new technologies and infrastructures that generate considerable data amounts within successful research areas, such as life sciences and image analysis.

Successful research and innovation needs access to first class research infrastructures, and development of top-class research infrastructures is one of the key areas to enforce the European Research Area. In the overall context of research infrastructures, e-Infrastructures play a more and more important role. Today, almost

[3] www.prace-ri.eu

all large-scale research activities include or are supported by several e-Infrastructure components. Major scientific breakthroughs are increasingly achieved by an international, cross-disciplinary team transferring, storing and analysing vast data collections and performing advanced simulations using different types of computing facilities.

Much effort has been spent on bringing research infrastructures and e-Infrastructures together for their mutual benefit, but it might still be difficult for many of the stakeholders to grasp the needs of users and the advantages of using existing e-Infrastructures and related services.

More and more scientific user communities share the demand for new solutions for the steadily increasing amount of scientific data that arise from scientific instruments (i.e. higher resolutions), an increased amount of sensors in scientific field studies (i.e. constantly created real-time datasets), or better computing power (i.e. more granular and realistic simulations). As a consequence, the term ‘big data’ emerged as a kind of new research field that aim to address the aforementioned challenges and to provide reasonable solutions in scientific ecosystems (i.e. research labs, universities, shared scientific instrument collaborations, etc.).

In this dissertation, I present the design of a new research infrastructure to support the European neuroscience community being part of the Human Brain Flagship project. The infrastructure, named Fenix[4], and receiving funds from the European Union's Horizon 2020 research and innovation programme through the ICEI project through, will provide computing and storage capability to address interactive use cases. This work also reports about technology trends in the field of HPC, memory

[4] https://fenix-ri.eu/

technology, deep learning and quantum computing as driving forces to tackle future large scientific challenges.

Chapter III.

The Fenix Infrastructure

Fenix is based on a consortium of five European supercomputing centres, which have agreed to deploy a set of infrastructure components (IaaS) and integrated platform (iPaaS) services federating users access, data repository and resources allocation mechanism. The key infrastructure services provided by each site are scalable computing services, various data storage services and, as a new element introduced with this project, Interactive Computing Services.

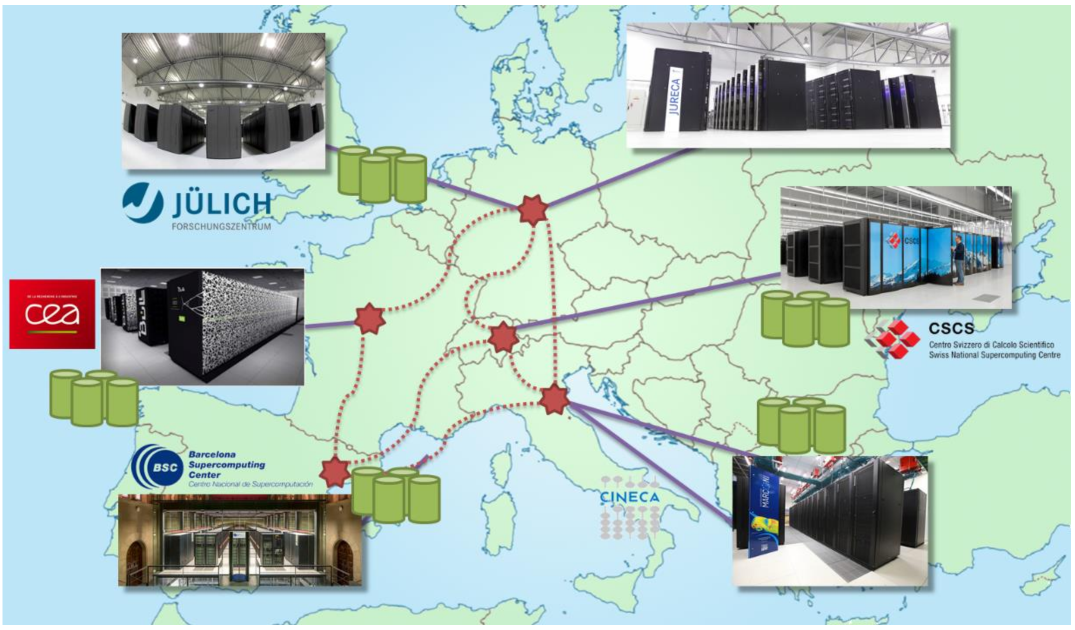

JÜLICH
FORSCHUNGSZENTRUM
JURECA
cea
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación
CINECA

Figure 1 – Centres participating to the Fenix distributed infrastructure.

Each of the involved centres is currently operating HPC systems with a throughput of floating-point operations in the multi-PFlop/s range. The compute capabilities will increase over the next couple of years such that each of the five sites will provide at least 10 PFlop/s, resulting in an aggregate compute capability of at least 50 PFlop/s.

Each of the sites will provide the following services:

- Scalable Compute Services;
- Interactive Compute Services;
- Active Data Repositories based on fast memory and active storage tiers;
- Archival Data Repositories; and
- Information/catalogue services.

All services will be tightly integrated through a high-speed network of the scalable compute systems (as shown Figure 1).

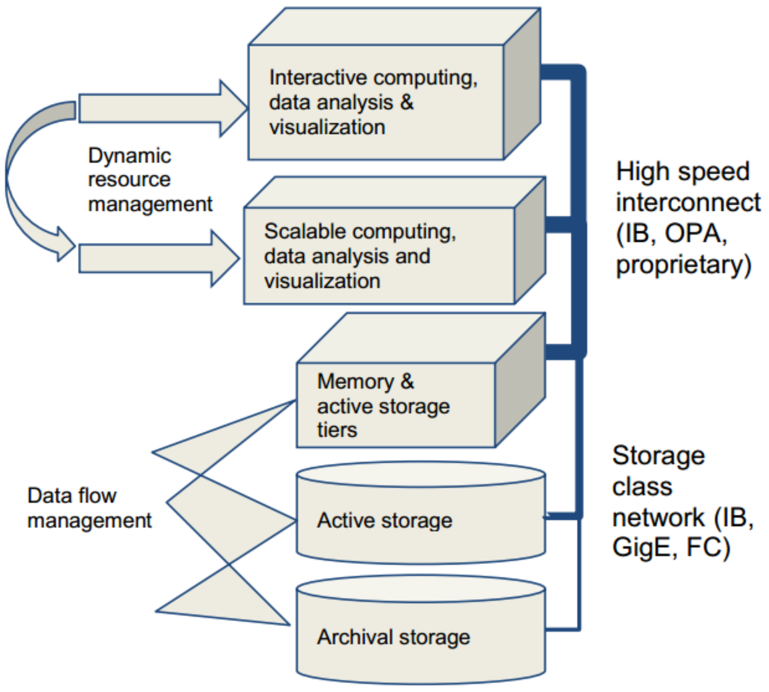

Interactive computing, data analysis & visualization
Dynamic resource management
Scalable computing, data analysis and visualization
High speed interconnect (IB, OPA, proprietary)
Memory & active storage tiers
Data flow management
Active storage
Storage class network (IB, GigE, FC)
Archival storage

Figure 2 – Fenix services.

The local services need to be interoperable to facilitate, e.g., easy transfer of data between federated Archival Data Repositories. Whenever interoperability is required, our strategy is to deploy the same or similar technologies at all sites. At the same time the project aims to keep the level of integration low in order to reduce operational dependencies between the sites (to avoid, e.g., the need for coordinated maintenance and upgrades) and to allow for the site local infrastructures to evolve following different technology roadmaps.

Furthermore, to allow for these distributed computing and storage resources to be perceived as a coordinated, distributed infrastructure, various additional federated infrastructure services need to be deployed (as shown in Figure 2):

- Authentication and Authorization Infrastructure (AAI) services;
- File catalogue and location services;
- User and resource management services; and
- Data transfer services.

Use cases

This section reports on use cases emerged from the neuroscience community and that are expected to exploit the infrastructure services. Based on the information collected from end users, the architectural specification of the Fenix infrastructure has been designed. These details are also necessary to identify the capacity requirements and to identify which elements of the infrastructure are best suited for each use case. They are presented from less articulated to more complex ones.

GUI-based interaction with extreme scale network models

Different software components have been developed as part of the Brain Simulation Platform[5] (BSP) for the simulation of large, biologically realistic networks

[5] https://www.humanbrainproject.eu/en/brain-simulation/brain-simulation-platform/

representing large regions of the Central Nervous System (CNS), like entire model animal brains (such as robotic rodents) or mouse and human CNS regions such as the cerebellum or hippocampus. For the development of these simulations, iterative visualization of network generation and simulation is necessary.

The BSP is an internet-accessible collaborative platform, which comprises a suite of software tools and workflows for collaborative brain research to allow researchers to reconstruct and simulate detailed multi-level models of the brain, displaying emergent structures and behaviours. BSP users can define and launch simulations from Jupyter[6] notebooks through any web browser. Most of these simulations, including the simplest one, require HPC resources and the management of non-trivial workloads via interactive notebooks.

Currently, the bottleneck of computational demands for this class of experiments is the network simulation, using tools like NEST[7], Neuron[8], Arbor[9] or TVB[10]. To estimate the scale of the problem, the biggest NEST simulation ever executed on the K[11] supercomputer in 2013 used approximately 1.1 PByte of memory while reaching up approximately 10 PFlop/s of peak performance. To move from 109 neurons to 1010 neurons using the current class of software architectures would take order of 10 PByte and 100 PFlop/s [16].

Conversely, for morphologically detailed simulations, initial estimates using the "Arbor" simulator indicate that peak performance for a GPU-based architecture occurs with 10k cells/GPU. Since 10,000 NVIDIA TESLA P100 GPUs accumulate 50

[6] http://jupyter.org/
[7] www.nest-simulator.org
[8] www.neuron.yale.edu/neuron
[9] arbor.readthedocs.io
[10] www.thevirtualbrain.org
[11] www.r-ccs.riken.jp/en/k-computer/about

PFlop/s, to sustain such activity it would need 10,000 sockets * 10,000 cells/socket * (1~1000) kBytes/cells = 1~1000 * 109 Bytes, for 100 * 106 cells, which is on the order of the size of the human hippocampus. In practice, from the computational point of view, such simulations require a variety of hardware resources and capacity that hardly are available today. Besides CPU power, these applications require post-processing capabilities, such as visualization tasks that can be handled with desktop systems equipped with a moderate amount of memory. However, the amount of resources needed for such visualization tasks, especially expressed in terms of memory capacity, is highly dependent on the details of the simulation but existing applications are usually flexible enough to balance between different levels of the memory hierarchy.

<u>Large scale simulations of Hippocampus model</u>

In silico experimentation within brain region models is a core target of the European neuroscience community. It allows linking results from experimental neuroscience with model predictions for discovery and validation. The models that will be made available are based on a close bidirectional interaction with anatomical and physiological data. The models focus on the cerebellum, the hippocampus, and the basal ganglia.

This use case describes the execution of an in silico experiment of a biophysically detailed model and the execution of a pre-defined analysis by a community user against models released to the community. Scientists can now devise in silico experiments that they could not do before in the absence of the required storage and compute resources for downloading and executing potentially large models. At the same time, the resulting artefacts remain within the infrastructure and become easily available for reuse in other contexts, i.e. analysing, visualisation, sharing with the community etc.

The focus here is on models of synaptic plasticity of hippocampal synapses (see Figure 3), and how they can be integrated into cellular level microcircuit models using data-driven subcellular pathways and/or rule-based effective implementation. The emphasis is on the mechanisms underlying associative memory processes and spatial navigation, integrated into a user-friendly user interface allowing an easy community engagement.

Hippocampus data are the most interesting simulation targets for the following reasons:

- Contain few millions neurons.
- Are strongly involved in higher brain functions, such as learning, memory, spatial navigation.
- Implicated in Alzheimer's disease, temporal lobe epilepsy, cognitive aging, post-traumatic stress disorder, transient global amnesia, schizophrenia, depressive and anxiety disorders.

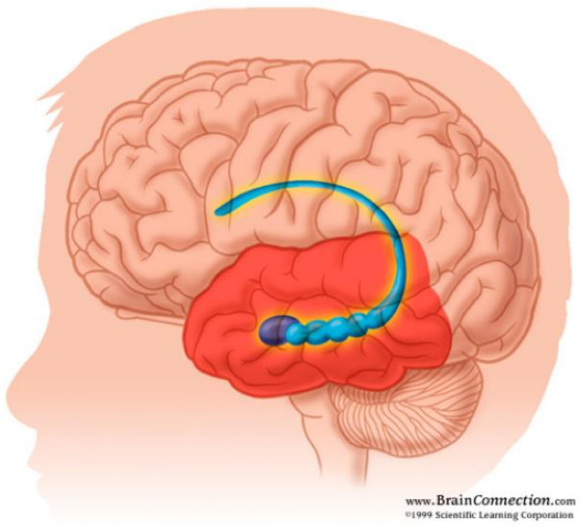


Figure 3 - Location of the hippocampus in the brain (Image courtesy of the Brain Connection project).

Some raw numbers follow:

- 450000 neurons, ~$1 \cdot 10^8$ membrane segment, 20 ODE[12]/seg
- $2 \cdot 10^9$ ODEs + synapses
- 1 second of simulation time (approximately 5h on a massive parallel system using 32000 processors)
- ~2TB of input, up to ~3TB of output.

A basic execution workflow for this use case includes the following steps. The user:

- selects the Small Circuit or Brain Area Circuit in silico Experiment function of the Brain Simulation Platform;
- selects a detailed circuit model from an online databank called NIP[13];
- selects target regions he/she wants to stimulate;
- for each selected target region, he/she defines the stimulus he wants to apply;
- defines the particular parameters of each stimulus (e.g. start, duration);
- selects what he wants to record from the circuit (e.g. soma voltage of a particular subset of neurons);
- defines global parameters for the simulation (e.g. time steps);
- defines additional parameters related to the allocation of the compute resources for the simulation (HPC center and system, HPC project, number of nodes, memory, …);
- defines the analysis he/she wants to perform from a predefined set and configures this analysis;

[12] Ordinary Differential Equation

[13] https://nip.humanbrainproject.eu/

- defines additional parameters related to the allocation of the compute resources for the analysis (HPC centre system, HPC project, number of nodes, memory, …);
- the simulation and the analysis are executed on the different compute resources defined by and accessible to the user. The circuit is available on this compute center at this stage;
- investigates the simulation results and the circuit interactively through a Jupyter notebook;
- visualizes the simulation on a visualization web service;
- registers and stores simulation results in the knowledge graph; the Human Brain Project has developed a central data repository where to store relevant data sets and associated meta information for sharing purposes.

Figure 4 shows the entire workflow and all its steps.

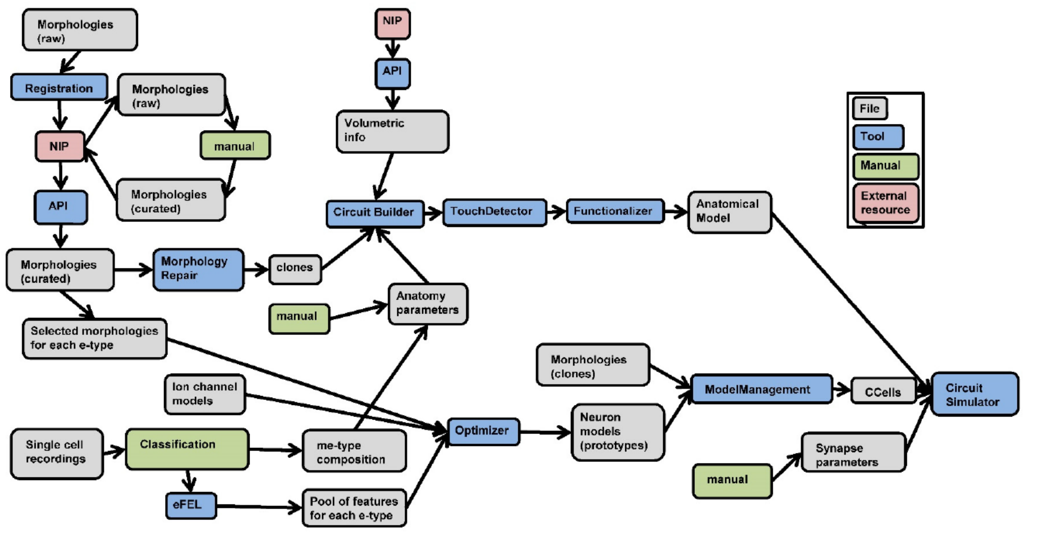


Figure 4 – Pre-processing and model generation pipeline as described in Hippocampus model generation.

<u>**Decoding of brain cyto-architecture using large scale simulations.**</u>

Mapping and understanding the cytoarchitectonics [21] of the human brain is a challenge that started back at the beginning of the 20th century with famous neuroanatomists who segregated the cortex of a post-mortem human brain sample into a few dozens of areas from the observation of the laminar structure of the cortical ribbon and of its cellular organization using optical microscopy. The most famous atlas was developed in 1905 by Korbinian Brodmann and remains today widely used by neuroscientists even if it suffers from several limitations. First, because it was developed from a single sample, it cannot capture the inter-subject variability of the cytoarchitecture maps; second, boundaries of the areas have been drawn from visual observations, and may not reflect the real boundaries of functional areas. The community is fighting to go beyond Brodmann areas and during the last decade, several teams attempted new strategies to map the brain cytoarchitectonics. Several teams have tried to establish maps from the acquisition of large cohorts of in vivo human healthy volunteers. The success of such challenges rely on the capability of modern neuroimaging methods to probe the variations of the cellular organization of brain tissues in vivo. Unfortunately, the cellular organization of brain tissues (gray and white matter) can be extremely complex, and today, few is known about the diffusion MRI (Magnetic resonance imaging) signature of the plethora of possible cellular environments met in the brain. The study of MRI signatures requires the tuning of several sequence parameters that impact the nature of the diffusion contrasts obtained at the end. Obviously, one cannot achieve an exhaustive scanning of the sequence parameter space in vivo.

This specific use case aims at replacing in vivo diffusion of MRI scans by in silico diffusion enabling to reach a much higher level of completeness of the parameter space sampling. To do so, it first focuses on white matter (WM) cyto-architecture being simpler than grey matter (from a cyto-architectural point of view) and requires to:

- **Step #1** - create an exhaustive bunch of in-silico realistic white matter virtual tissue samples by numerical simulations of cellular membrane geometries;
- **Step #2** - simulate the diffusion process of water molecules in every realistic in-silico WM tissue sample using a Monte-Carlo approach;
- **Step #3** - simulate the diffusion MRI signature of every WM tissue sample for an exhaustive set of diffusion MRI sequence parameters achievable on actual preclinical and clinical MRI systems;
- **Step #4** - train a deep neural network to build a decoder/regressor of the WM microstructure;
- **Step #5** - use the decoder to establish an atlas of the WM microstructure.

Dedicated software tools have already been developed by several teams to address these first three tasks. However, there is a clear need to extend simulations to a larger facility in order to run the plethora of needed simulations and obtain results for white matter tissues. If successful, the extension to grey matter (cortex and deep nuclei) will be straightforward but will need even more computational resources, due to the higher level of complexity of the cellular environment in grey matter.

Diagram 1 provides a detailed flowchart for **step #1** consisting of simulating a dictionary of virtual white matter tissues. Each virtual tissue is designed from a set of geometrical parameters including:

- the number of white matter fiber populations (from 1 to 3),
- the properties of each fiber population including its volume fraction, the main direction of the population, the dispersion and tortuosity of its fibers, the statistics of the axon diameter, the statistics of the g-ratio characterizing the myelin sheath, myelin g-ratio, the statistics of the Ranvier nodes, the permeability of axons,
- the properties of the glial cell population including their mean diameter, the statistics of the number of branches per cell, and the statistics of the diameter of these branches.

A graphical user interface will be developed to facilitate the prescription of tissue parameters, and a 3D viewer will be developed to visualize 3D renderings of virtual tissues.

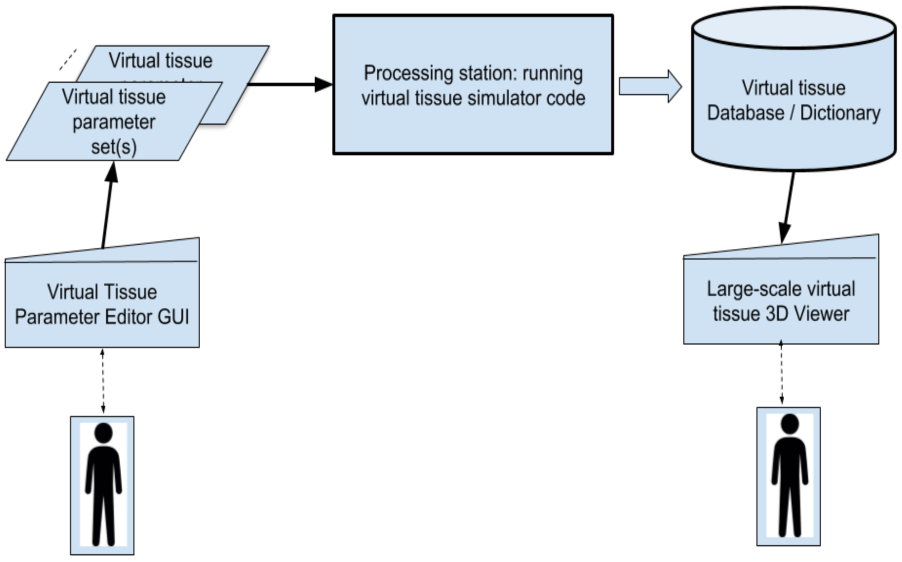

Virtual tissue parameter
Virtual tissue parameter set(s)
Processing station: running virtual tissue simulator code
Virtual tissue Database / Dictionary
Virtual Tissue Parameter Editor GUI
Large-scale virtual tissue 3D Viewer

Diagram 1 - Flowchart of the construction of a large dictionary of virtual brain white matter tissues.

Step #1 details

This step consists in generating realistic geometries of white matter: each generated voxel of size 100 x 100 x 100 µm will contain from 500 to10 000 axons (depending on the mean diameter of axons and on the packing density). Each axon is represented as a set of spheres, which is the basic unit of the analysis algorithms. An upper bound of 500MB for each geometry to store the position and the radius of all spheres within a voxel is estimated. The geometry generation algorithm can be decomposed in two steps:

- creation of overlapping axons in the voxel according to the required biophysical parameters (diameter, packing density, angular dispersion, etc.);

- solving the overlapping between axons using the decomposition of axons into spheres and applying repulsion forces between overlapping spheres.

An upper bound of 30 minutes to generate a given geometry is estimated on a NVIDIA Tesla K40 GPU and it corresponds to the worst possible case with very small mean axons diameter (0.1µm) and high packing density (0.8) for which the number of spheres is maximal. In most of the cases (diameters > 0.5µm and volume fraction inferior to 0.7), the geometry generation will take less than a minute using the same graphic card.

Diagram 2 provides the complete flowchart of this step required to establish a huge database of Monte-Carlo simulations of the diffusion process of water molecules within each white matter virtual tissue belonging to a virtual tissue dictionary. Biophysical parameters characterizing the diffusion process in brain tissues have to be fed into the Monte-Carlo simulator as well as the individual virtual tissue sample. Trajectories followed by random walkers are then stored for each tissue sample. The number of random walkers has to be tuned with respect to the complexity of the geometry of cell membranes populating every virtual tissue, typically on the order of 10^5 particles. Temporal constraints are imposed by the specifications of the simulated diffusion MRI sequence (echo time and temporal resolution of gradient waveforms).

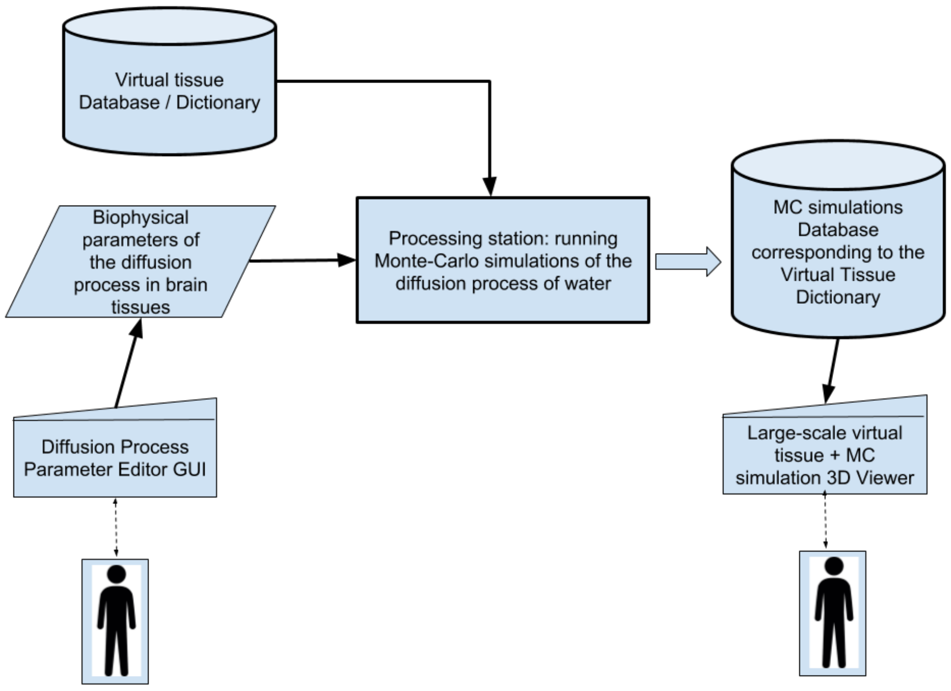

Virtual tissue Database / Dictionary
Biophysical parameters of the diffusion process in brain tissues
Processing station: running Monte-Carlo simulations of the diffusion process of water
MC simulations Database corresponding to the Virtual Tissue Dictionary
Diffusion Process Parameter Editor GUI
Large-scale virtual tissue + MC simulation 3D Viewer

Diagram 2 - Large-scale Monte-Carlo simulations of the Brownian motion of water molecule corresponding to the Virtual Tissue dictionary established in Diagram 1.

Step #2 details

This step consists in performing a Monte-Carlo simulation of the Brownian motion of 2.10^5 random walkers during 300ms with a time step of 10µs for each geometry generated in step #1. The trajectories of all the random walkers have to be stored, leading to a size of 97 GB per simulation.

Although the application gives space to further optimization, a runtime of 1h30 on a NVIDIA Tesla K40 GPU to perform the simulation for one voxel has been estimated.

Diagram 3 provides the flowchart of step #3 required to establish the huge dictionary of (virtual tissues/diffusion MRI signature) required to learn the decoder mentioned in step #4. The diffusion MRI signature will consist of a few thousands of simulated NMR contrasts corresponding to Pulsed Gradient Spin Echo (PGSE) and

Cosine Trapezoidal Oscillating Gradient Spin Echo (CT-OGSE) sequences achievable on an actual clinical 3T MRI system. Each of these sequences offers the possibility to tune parameters affecting the diffusion sensitization such as the diffusion gradient magnitude, the diffusion direction, the diffusion pulse width and separation for the PGSE sequence or the diffusion pulse frequency and number of lobes for the CT-OGSE sequence.

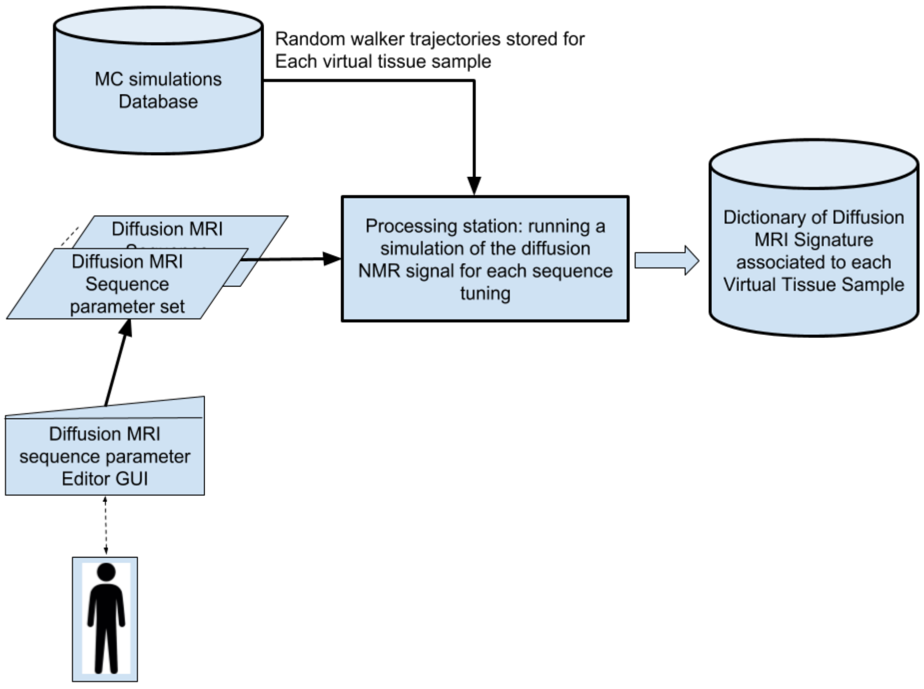

MC simulations Database
Random walker trajectories stored for Each virtual tissue sample
Diffusion MRI Sequence
Diffusion MRI Sequence parameter set
Processing station: running a simulation of the diffusion NMR signal for each sequence tuning
Dictionary of Diffusion MRI Signature associated to each Virtual Tissue Sample
Diffusion MRI sequence parameter Editor GUI

Diagram 3 - Large-scale simulations of the diffusion-weighted MRI signal over a large set of sequence tunings from the Monte-Carlo simulations obtained in Diagram 2 in order to establish a dictionary of (virtual tissues / diffusion MRI signatures).

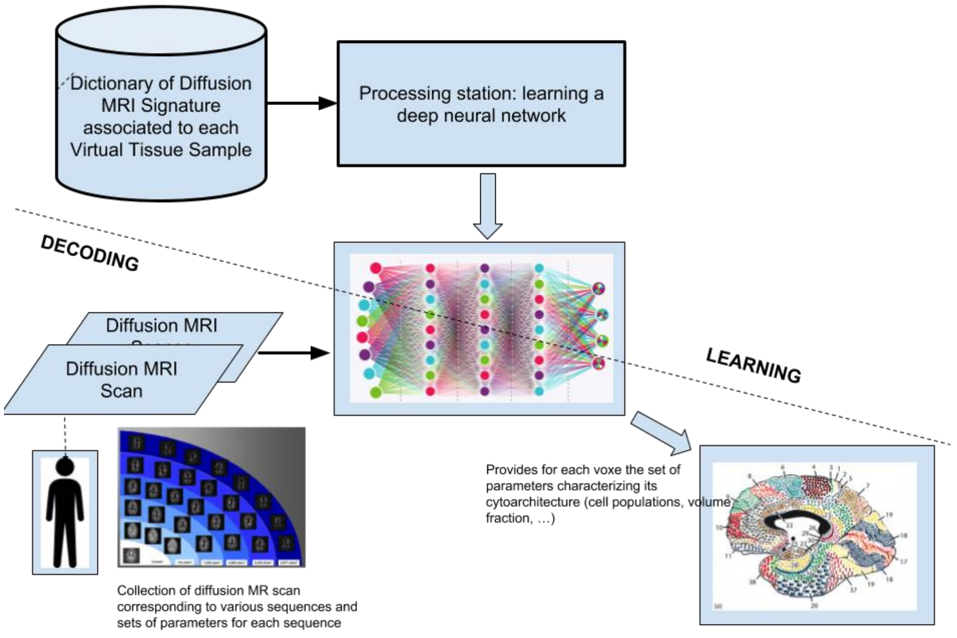
Step #3 details

This task consists in synthesizing the NMR (Nuclear Magnetic Resonance) signatures of each generated geometry by applying diffusion sequences with varying parameters on the previously generated random walkers trajectories (see step #2).

I estimated that 3000 NMR signatures have to be computed for each geometry to fully explore the parameter space. Each geometry has a size of 10KB, thus leading to a total of 30MB for each generated geometry.

The computation of NMR signatures can be easily parallelized on GPU, leading to an estimated runtime of ~1.2s per signature, and thus a total runtime of 1h for each geometry.

Diagram 4 depicts how the former dictionary of virtual white matter tissue samples/diffusion MRI signatures enables to train a machine learning tool, like a deep neural network, in order to create a decoding tool able to recognize/extrapolate the set of quantitative features characterizing the cyto-architecture at each voxel of the brain, from a real and individual set of diffusion MRI scans, corresponding to various sequences and sequence settings. The input database used to train the DNN (Deep Neuronal Network) is composed of around ~10^{10} entries resulting from the previous large-scale simulations.



Dictionary of Diffusion MRI Signature associated to each Virtual Tissue Sample
Processing station: learning a deep neural network
DECODING
LEARNING
Diffusion MRI Scan
Diffusion MRI Scan
Provides for each voxe the set of parameters characterizing its cytoarchitecture (cell populations, volume fraction, …)
Collection of diffusion MR scan corresponding to various sequences and sets of parameters for each sequence

Diagram 4 - use of the simulated "virtual sample/diffusion MRI signature" dictionary to training a deep neural network and use of the trained network to decode the brain cyto-architecture of individuals in vivo.

Step #4 details

The last step of this use case is to train a neural network with all the generated NMR signatures. For each signature, the parameters of the employed diffusion sequence and the biophysical parameters of the generated geometry are known. The aim of the training is that, when a real NMR signature is fed to the network with known diffusion sequence parameters, the network will be able to estimate the underlying biophysical parameters. This task brings the need for "Deep-Learning oriented" resources, such as GPUs, to train a neural network with all previously simulated data, i.e. NMR signatures.

The large majority of collected requirements bring important needs especially in terms of hyper-scaling and data handling. The Fenix architecture reflects these needs by providing low level services in order to permit community build upon and create sustainable research environments. The infrastructure is optimised for handling and processing large volumes of data efficiently and to make them easily identifiable. Much care has been put to keep the infrastructure generic and compatible enough to support other workloads emerging from different communities. To achieve community requirements, it will provide the following major components:

- Interactive computing nodes with large memory, equipped with high performance GPUs, and interconnected by a low-latency, high-performance network. Its purpose is to improve pre and post treatment like meshing creation, steering computation, remote visualisation but also for running MPI applications. Some of the compute nodes will have particularly large memory (several Terabytes per node) to overtake usual limitations in neural simulation by allowing to compute larger brain regions.
- Intensive I/O will be handled by an Active Data Repository, a high-performance file-system based on flash storage technologies. The purpose of this system will be to handle high data throughputs, and in particularly, adapted to speed-up non-sequential data extraction, which is often needed to process experimental data.
- A large long-term capacity is needed to store experimental data as well as simulation outputs, and to make them publicly accessible and sharable. Using the Fenix terminology, this long-term capacity implements the Archival Data Repository (ARD), and will be completed by additional servers to enable data movements across multiple storage tiers. Regardless of its name, this storage area is not designed to implement a pure repository as the management of information like metadata, persistent identifiers, etc. is encapsulated into community services. The Archival Data Repository will only ensure long-term

archiving of data (bit-stream preservation) and provide basic components to build a full data repository on top.

- A Cloud Computing infrastructure to make it possible to setup community services to access data: web interfaces, data processing service, data exploration, workflow orchestrator, visualization etc.

The sections that follow provide further details about infrastructure characteristics.

Data Infrastructure and Management

In contrast to other communities that already operate a large-scale federated data infrastructure, neuroscience has to cope with a diverse set of data sources with their specific formats, modalities, spatial and temporal scales, coverage, sample sizes, etc.. This includes neuro-imaging data from high-resolution microscopes and MR-imaging, electro-physiological data from multi-electrode array measurements, data from brain simulations on HPC systems or neuromorphic architectures. No fixed relationship exists between the data sources. Rather, the scientific approaches and workflows are a much faster moving target compared to, e.g., high-energy physics. Thus, a tiered approach as realised in the world-wide CERN Grid will not work for neuroscience, and a different organising principle has to be chosen for HBP. The setup of the Fenix data infrastructure is guided by the following considerations.

- Data is brought in close proximity to the data processing resources at different compute and data infrastructure service providers in order to take advantage of high bandwidth active data repositories as well as data archival services.
- Federating multiple data resources enables easy replication of data at multiple sites. This capability can be exploited to improve data resilience, data availability as well as data access performance.

As anticipated above, the Fenix sites plan to provide different classes of data repositories:

- **Archival Data Repositories**. Data stores that are optimized for capacity, reliability and availability. Archival data repositories are meant to be used for storing large data products that cannot be easily regenerated, in particular those for which long-term accessibility is crucial.

- **Active Data Repositories.** Data stores that are located close to computational or visualisation resources such that high performance access to data is enabled (in terms of high bandwidth and/or high IOPS rates). Active data repositories should be used for storing temporary slave replica of large data objects, for improving access performance, with the master copy of the data kept in an archival data repository.

The various Archival Data Repositories will be federated using standard access protocols. In this context, federation means that users of the infrastructure are offered with a unified view on loosely coupled storage resources, which, in particular, has the following features:

- Integrated AAI with single sign-on;
- Data transfer services; and a
- Data location service.

The Archival Data Repository will provide role-based access control (RBAC) mechanisms for authorising access to data. This approach is sufficiently flexible for staying compliant with most of the community's data management policies. Conversely, the Active Data Repository will not be federated and will hold temporary, private copies of data products. This allows using POSIX compliant parallel file-systems as storage volume. A storage architecture concept that would realise the presented concept is shown in Figure 5. The Archival Data Repository is assumed to be realised by an object store using an OpenStack SWIFT front-end to facilitate data sharing over HTTP protocol. A data mover will allow data copying between the two repositories, the object store and the parallel file-system. The decision to keep data separated, also in terms of underneath technology solutions, will provide high-performance access to data from both the High Performance Computing resources as well as the Interactive ones.

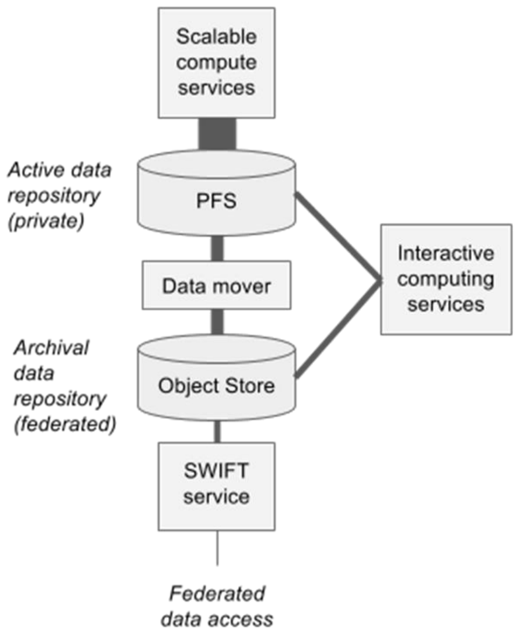

Scalable compute services
Active data repository (private)
PFS
Data mover
Interactive computing services
Archival data repository (federated)
Object Store
SWIFT service
Federated data access

Figure 5 – The Fenix multiple tiers data management model.

Interactive Computing Services

Nowadays the largely widespread usage model for large-scale HPC systems is based on the scheduling of batch jobs. This approach is not needed to meet a requirement from the computational science communities, but it reflects the predominant issues in the management of the HPC systems, the maximization of resource utilisation, and the handling of concurrent workloads. However, the situation differs when taking into consideration personal workstations or shared memory servers, where time-sharing interactive executions are the norm.

As an extension to batch-oriented systems, Fenix aims at providing resources that can be used interactively to the cost of obtaining nearly lower performance while keep ensuring high resource utilization. Thus, Interactive Computing refers to the capability of a system to support distributed computing workloads while permitting on-the-fly interruption by the user. The real-time interaction of a user with a program runtime is motivated by various factors, such as the need to estimate the state of a program or its future tendency, to access intermediate results, and to steer the computation by modifying input parameters or boundary conditions. Within neuro-science applications, i.e. brain activity simulation, large image volume rendering and visualization, connectomics experiment, this implies that the runtime can be modified interactively so that the user can gain insight on parameters, algorithmic behaviour, and optimization opportunities. The commonly agreed central components of interactive computing are, on the front-end, a sophisticated user interface to interact with the program runtime and, on the back-end, a separated steerable, often CPU and memory consuming application running on an HPC system.

A typical usage scenario for interactive computing regards the visualization, the processing, and the reduction of large amounts of data, especially where the processing cannot be standardized or implemented in a monolithic workflow. The data can be generated by simulation or harvested from experiment or observation; in both cases, during the analysis the scientist performs an interactive process of successive reductions and production of data views that may include even complex processing like convolution, filtering, clustering, etc. This kind of processing could be easily parallelized to take advantage of HPC resources, but it would become clearly counterproductive to break-down a user session into separate interactive steps interspersed by batch jobs as their scheduling would delay the entire execution degrading the user experience. Besides that, in many application fields, the computational scientists are starting to use interactive frameworks and scripting languages to integrate the more traditional compute and data processing application running in batch, e.g. the use of R, Stata, Matlab/Octave or Jupyter Notebook just to name few. In this way, the time spent in this activity is a non-negligible component in the “time to science”.

In Fenix, users will be presented with the possibility to load and visualize data resulting from simulations or collections, series or experimental data, in an interactive way. To properly support interactive supercomputing, the system must be able to properly handle all the connections and protocols needed to dynamically attach multiple visualization and steering front-ends to a running application and to enable transparent staging of data across multiple storage tiers, from distributed storage participation to node-central memory, to ensure front-ends receive a continuous stream of data. To support interactive workloads, the infrastructure will include high-end servers equipped with a) large capacity of high-end volatile memory, optionally integrated with non-volatile volumes (e.g. NVMe or DDR-like interfaces); b) GPUs for visualisation and image manipulation; c) high-performance low-latency interconnect to facilitate fast access to repositories as well as computing resources.

User Management and AAI

For the Fenix infrastructure, a central identity accreditation system will be set up. The goal of such a system is to provide users with secure, trustworthy and convenient access to services and resources made available by the infrastructure. Ideally, users would only need to use a single set of credentials to access distributed resources through authenticating towards trusted locations, these being either the home organization identity provider, a community portal, or the Human Brain central accreditation system. To implement this vision, the central accreditation system will be able to a) manage and authenticate local (homeless) users, b) broker identities managed by external organizations, i.e. EduGain, which manifest their interest in federating with the infrastructure, and c) support existing Identities Directory. Through supporting standard authentication protocols, such as OpenID Connect or SAML, the central system will voucher for the user's identity trusted by the infrastructure to assert certain basic users profile attributes. This scheme has already successfully explored within other initiatives, such as the EUDAT Collaborative Data Infrastructure[14].

One of the promises of the Fenix AAI is to enable domain scientists to make effective use of the virtual research environment without having to deal with the intricacies that are typically associated with the use of HPC systems. As also highlighted by the AARC (Authentication and Authorisation for Research and Collaboration) project[15] guidelines, an omnipresent aspect of dealing with computing resources efficiently is connected with the authentication and authorization of users. When I talk about seamless access, it is intended for any user, registered on a "trusted" IdP (Identity Provider) and granted to consume a certain amount of resources, to access federated resources using his/her credentials (i.e. username/password, X.509, etc..) without going through any further registration process. The list that follows presents the main principles that have influenced the design of the AAI:

- The federation concerns with the aggregation of multiple hosting sites providing services and resources to users. Member sites contribute to the federation by accepting a common collaboration agreement that request them to operate service according to federation goals and performance. They need to maintain their independence and keep functioning regardless the status of the federation. Each member may request the acceptance of a local access policy besides that requested by the federation. Existing users, those who are already registered on

[14] https://www.eudat.eu/services/b2access
[15] https://aarc-project.eu/

one of the federated sites, become member of the federation by default. As such, they are automatically allowed (authentication) to access the federation and consume resources if granted through an allocation project. If a user has multiple accounts across different sites, all his/her profiles must be mapped to a unique Fenix identity for accounting and security reasons.

- The Fenix AAI is conceived as the bounding of two separate services, a Central IdP which is responsible to proxy authentication requests coming from federated IdPs, and an Attributes Provider which is responsible to control authorization requests through budget allocation, groups and roles management. Each federated site needs to expose its user-base through its own IdP, supporting standard identification protocols, i.e. SAML v2 or OpenID Connect (OIDC). Sites are responsible for the operation of their own IdP and free to decide which authentication mechanisms to support for their users, i.e. username/password, X.509, 2FA, RSA, etc.
- To be part of the federation, each federated IdP should be able to release a common set of user profile attributes in response to any authentication requests, successful or unsuccessful. These attributes should meet the requirements set by the REFEDS Research and Scholarship Entity Category (R&S)[16].
- The Fenix Central IdP needs to be able to proxy authentication requests coming from external IdPs, such the HBP OIDC server or the eduGain federation[17].
- Since the Fenix infrastructure plans to deliver various classes of services, the AAI needs to cope with this heterogeneity and thus support various access

[16] https://refeds.org/
[17] https://edugain.org/

interfaces, either based on Internet protocols, such as HTTP (Web Portal, SWIFT, etc.), or SSH.

- In order to access resources, users will be requested to accept the Fenix access policy and any extensions appended by individual sites. Global and local policies should not include conflicting terms or create ambiguity. The fact that a user has accepted the Fenix general policy will be recorded at the Central IdP.
- Access to resources is granted through the creation of projects/budgets managed by a central accreditation service called FURMS.
- The federation will also support "service account" to permit machine-to-machine interaction. However, for traceability reasons, these accounts need to be attached to a real person who is responsible to manage the service.

Thus, according to the design principles presented above, and the outcomes of the AARC project, Fenix foresees the deployment of two services covering the following high-level functions respectively:

- Fenix Central IdP
- Users identification and authentication
- Federation of multiple IdPs by proxying authentication requests
- Validation of user profiles
- Policy registry and management of principles of engagement
- FURMS (Fenix User and Resource Management Service) – central accreditation service
- Group/budgets membership management
- Authorization attributes provider
- SSH public keys management

- Managing site specific Usage Agreements
- Reporting and metering

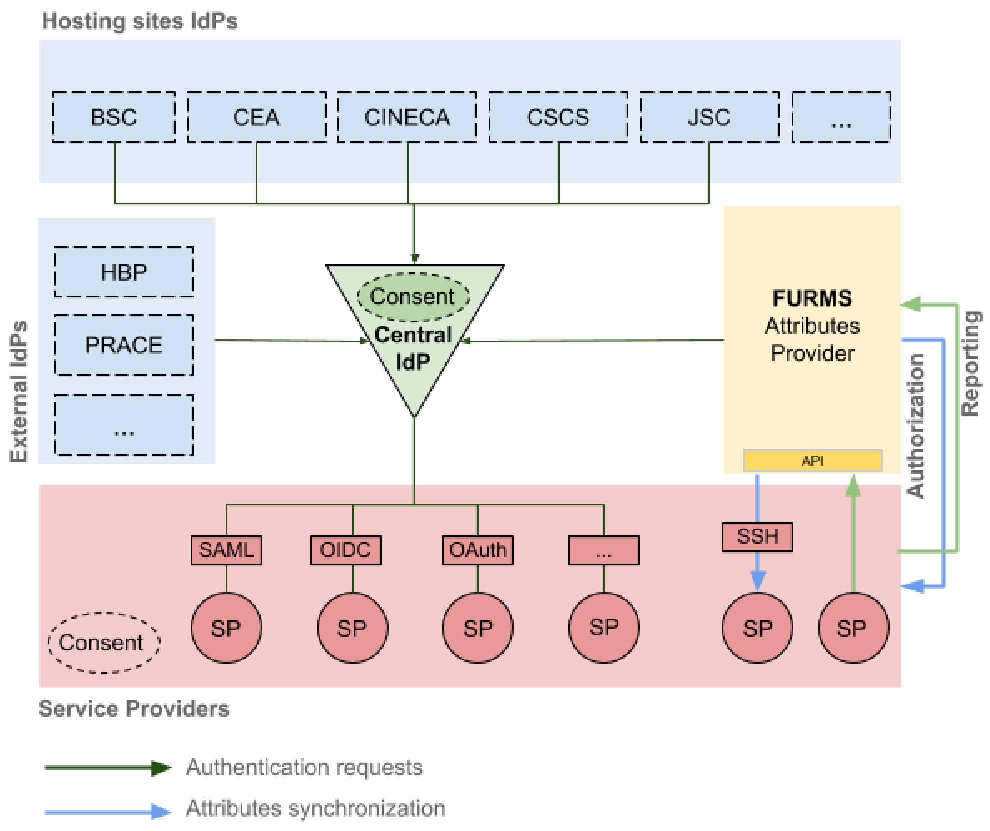


Figure 6 – Architecture of the Fenix Authentication and Authorization Infrastructure.

The Fenix Central IdP is responsible to proxy authentication requests among hosting site's IdPs. At the same time each site is deputed to maintain its own IdP, plus as many SPs as the number of offered services. Through this central proxy, a user already registered on a site or an external IdP, such as eduGAIN ot the HBP one, will be able to access services and resources provided by other Fenix sites without registering twice and using its home IdP to authenticate. The central proxy is also responsible to control the validity of a user profile, including the acceptance of the Fenix access policy, and to provide a set of APIs to permit hosting sites, or other services, to retrieve information about users profile basic attributes.

The Fenix Central IdP does not enforce access rights but simply forwards authenticated user' attributes to the hosting site which, applying local policies, decides if that user can access services or not. The Central IdP will be also configured to release LoA (Level of Assurance) information for each federated IdP, as well as the IdP the user is authenticating from. If needed, it can also retrieve attributes from external Attribute Authorities to enrich users' profile.

In parallel, FURMS will act essentially as a Service Provider (SP) of the Central IdP with the task of managing and issuing authorization records, such as information about available budget on sites. Furthermore, in order to serve multiple communities, the concept of Fenix Community has been introduced. A Fenix Community is a virtual organisation of scientists, for which the Fenix Resource Providers have committed to make part of the Fenix Infrastructure available. The approach is very similar to the virtual organisations used in grid computing and specifically the WLCG (Worldwide LHC Computing Grid).

While the federation of services supporting web protocols is a widely discussed topic which many solutions have been developed for, the access to resources via secure shell logins (SSH) presents different challenges. Our approach to address this requirement follows the IAM (Identity Access Management) model where user public SSH keys are stored on a remote IAM service and recall when needed. Each Fenix user will be responsible to create it public and private keys and upload only the public part to the IAM service. While trying to access a HPC system via SSH, the user will present his/her private key and leave the ***sshd daemon*** contact the IAM service to validate that the private key pairs with the public one. This handshake is made possible thanks to the "AuthorizedKeysCommand" option of the OpenBSD SSH implementation[18] which

[18] https://man.openbsd.org/sshd_config

serves to specify which service or program to look up for the user's public key. For security reasons only "RSA 2048" or "Elliptic Curves" encryption algorithms will be allowed to create key pairs. In the case any site would not agree to use the ***AuthorizedKeysCommand*** option to look up for the user's public SSH key on demand, a simple synchronization mechanism between the central IAM and a local database may be implemented. However, on a long-term vision, we foresee SSH access dropped off in favour of web applications, i.e. Jupyter notebooks, to access HPC resources even to execute very large workloads.

Open data

The discussion about open data has been going on for a long time at various levels, local, national and within the EU. At EU level, the European Open ScienceCloud (EOSC) is being discussed, which has resulted in a declaration[19] that many countries have endorsed. Fenix promotes a culture of open data, encouraging users adopt policy, guidelines to facilitate data sharing, and interoperability and making available needed technological solutions.

A prerequisite for open data is functioning and coherent e-infrastructure for storage, reuse, access and analysis of data. At the same time, open data are an important prerequisite for "data-driven research". Major inputs are also needed to make existing data accessible, which includes careful documentation of how data have been generated and the information the data contain, which is often referred to as metadata. Standardised metadata are, in turn, a prerequisite for research data living up to the principle of FAIR data[18], that is to say data that are:

- **Findable** – easy to find and discover.

[19] https://ec.europa.eu/research/openscience/pdf/eosc_declaration.pd

- **Accessible** – openly accessible without charge or other restrictions.
- **Interoperable** – comply with widely used standards and data formats.
- **Reusable** – available to be used and reused.

The fact that data are open and managed according to the FAIR principle should not be interpreted as an absence of restrictions. The principle for EOSC is that access to data shall be "as open as possible, as closed as necessary". Research into neuroscience, medicine and health in particular, data on individuals play a crucial role. These are data collected by researchers where private individuals provide the information, data from registers, patient data, biobanks, genome sequencing and quality registers. In these cases, technical solutions to protect data and the integrity of individuals are needed. Respect for the integrity of individuals and research that is conducted according to applicable legislation and ethical guidelines are both given. The General Data Protection Regulation (GDPR) entails stronger protection of the integrity of individuals, at the same time as fulfilling the needs of research. It is important to emphasise that research is ever more dependent on horizontal data, where individuals are monitored over long periods. As research is a dynamic process, this means that it must be possible to assemble databases with broad consent from the individuals providing information to research. The increased need for long-term studies of individuals also raises the potential conflict between the requirement to protect the integrity of sensitive personal data and the requirement to make data available for peer review on scientific publication. To safeguard the quality of research, this issue needs further consideration in the future. This is an area that is developing very rapidly, and continuous discussion is needed about the prerequisites for and ethical attitude of research.

Chapter IV.

Evolution of High Performance Computing

The purpose of this section is to document the changes that are taking place in HPC design, and outline some of the likely developments expected in the near future. The initial part then leads into a discussion of how software is evolving, the challenges that are being faced, and some of the new directions being explored to address them. It concludes with a discussion of the implications for researchers developing new codes; what should they think about in designing their new codes, and how can they avoid being locked into any one particular architecture.

High-performance computing has evolved remarkably over the past 20 years, and that progress is likely to continue. HPC is a strategic tool to foster competitive science while promoting innovation in different disciplines. After having being used for more than 30 years in climate research, numerical weather prediction, astrophysics or chemistry, HPC is now contributing to all scientific fields including biology, life sciences, materials sciences to finally reach social sciences and humanities. Into the industry field, HPC has been widely used in oil & gas exploration, aeronautics, automotive and finance, and it is now becoming crucial for ensuring personalized medicine, develop nanotechnologies or enable the exploitation of renewable energies. Moreover, HPC is becoming a tool of growing importance for public decision making by creating simulated scenarios to prevent and react to natural risk events, such as earthquakes, thunderstorms, floodings, volcano eruptions, biological hazards and (cyber)-terrorism attacks. However, in recent years, progresses in HPC have been achieved through greatly increased hardware complexity with the rise of multicore and many-core processors, sometimes affecting to much the ability of developers to achieve the full potential of new architectures. Into the Moore's law regime, also moving from vector to parallel processing, we have observed relatively smooth transitions from Gigascale (1985), Terascale (1997) to Petascale (2008), while the next move that should take the community to Exascale (between 2019 and 2023) is bringing new challenges still hard to solve, such as energy efficiency and the high cost of moving data.

Another important factor connected with the evolution of HPC systems is the convergence between numeric simulation and big data workloads due to the high volume of data coming from next generation scientific instruments, e.g. satellites, telescopes, microscopes, and sequencers.

Processing vast amount of structured or unstructured data in a profitable manner is not possible for human beings any longer. This limitation fuelled the rise of a new paradigm, the High Performance Data Analytics (HPDA), which, combining processing and data manipulation techniques (assimilation, interpretation, extraction, prediction), capitalizes on artificial intelligence (AI) and machine learning (ML) methods to extract new values from data.

As a result, many scientific communities are now combining HPC and HPDA methods into large-scale analysis pipelines to process massive volume of data generated by simulations, experiments or observations, and being able to - in-situ or in-transit - infer further outputs. This convergence will permit the classification and identification of relevant data features at processing time avoiding the data to be first stored and then post-processed. In the near future, this could also lead to the development of smart AI-driven computational steering techniques. In combination to this, the modern AI, the one combining 25-year-old Deep Neuronal Network (DNN) algorithms with modern hardware (GPU or FPGA) will benefit from converged HPC/AI architectures for addressing new scale out challenges including:

- developing more complex and deeper networks (like CapsNet Capsule neural networks [63]);
- use more complex and multidimensional data and increase training sets;
- develop explainable AI (XAI) methods[20].

Impact on multi Petascale and pre Exascale architectures

Because of energy cap and high cost of moving data, the road to Exascale needs to create a new paradigm, not only technological but also organizational. More precisely, to pave the way towards Exascale we need to:

- rethink and support data workflows from end to end;
- federate converged HPC/data infrastructures through (dynamic) high bandwidth network services (like Software Defined Networks);
- ensure the co-existence of stream and batch models, provide smart resources managers able by secured containerization techniques to support efficiently multiple software stacks including HPDA and AI;
- enable efficient exchange of data between simulations and analytics going beyond POSIX limitations;
- develop new multidisciplinary skills in HPC, data management and AI and foster new insights from scientific and industrial disciplines towards data discovery.

[20] https://www.cmswire.com/digital-experience/what-is-explainable-ai-xai/

With several architectural options still remain open for the Exascale area in terms of processor, accelerators, memory and I/O subsystems, the multi-petascale and the pre-Exascale production systems planned to be deployed by the EuroHPC Joint Undertaking and the Member States around 2020/21 will be based on 2 different kind of hardware architectures for converged (HPC+AI) nodes:

- (multi, many) core based architectures;
- accelerated architectures using more or less tightly coupled GPU (or FPGA) with (multi, many) core CPU.

Pre-Exascale systems of approximately 300 PFlops of peak performance are expected to couple around 45,000 nodes (for multi-many core based partitions) and 3,000 nodes (for accelerated ones).

Based on the requirements coming from scientific and industrial communities, such systems, based on balanced architectures, in terms of compute capacity per node, memory capacity/bandwidth per node as well as networking performance, will be used for both capability[21] and capacity[22] workloads. They will also serve to accommodate a mix of HPC and HPDA/AI workflows orchestrated by next-generation resource managers able to support coexistence of batch and interactive nodes.

Specific HPC options

This section explains the main hardware and software options to build a credible roadmap towards Exascale. As no significant breakthroughs in hardware technologies are expected in the next 2-3 years, the focus is on pre-exascale and multi-petascale

[21] Execution of large-scale applications spanning potentially to all the system.
[22] Execution of coupled multi scale, multi physics applications, ensemble, optimisation and uncertainties quantification studies.

systems as the fulfilment of their requirements is a necessary condition to make the successive step.

HPC processing units

There are three main elements which are characterizing recently deployed HPC systems; a) accelerators such as GPUs attached to generic processors, b) presence of standalone processors (multi-core) and c) many-core cards. While multi-core processors provide a generic approach for general HPC workloads, the accelerators and the many-core ones support a higher number of threads at a lower performance, which imposes certain limitations to get good performance from them. The most relevant ones is the high level of scalability that an application must have to profit from the availability of so many cores without incurring in race conditions.

Although the approaches based on accelerators or many-core provide very profitable GFlops/watt ratios, HPC standalone processors usually offer lower performance but without requiring applications to be modified.

Considering the size of the pre-exascale machines, the options to provide a pure HPC standalone-processor machine is limited, as with the current technologies it will require an excessive power consumption to provide a 300 PF peak performance machine. Furthermore, the current options for many-core processors after the decommissioning of the Intel Xeon Phi product line are currently very limited. Intel is working on a new many-core architecture for exascale that includes dataflow engines[23] but the releases roadmap has not been confirmed yet.

Another improvement path relies into making use of resources more dynamic to adjust hardware behaviour, such as voltage/frequency scaling and (de)-activation of

[23] https://www.nextplatform.com/2018/08/30/intels-exascale-dataflow-engine-drops-x86-and-von-neuman/

functional units, according to application performance needs. Even with these improvements, the architecture might remain too complex for large HPC applications. Of course programming environment should be able to provide a standard interface to hide this complexity. At the hardware level the appropriate features should be available to support a hybrid approach as MPI + X where X could be either OpenMP v4.0 or other libraries as well as PGAS + X.

Memory

Although the performance of CPUs and HPC accelerators has improved drastically over the last years, the delivered peak of compute nodes has not followed the same positive trend. The gap between theoretical and delivered performance is directly connected to memory access whose throughout has not registered any significate improvements; this phenomenon is often referred to as “memory wall”. It is evident that memory systems play a critical part in HPC systems, directly affecting their performance, power consumption, and cost of ownership. Recent requirements from applications and constraints from technologies are shifting HPC systems towards more complex, deep memory hierarchies. First, requirements for memory capacity continue to increase dramatically in response to simulation, machine learning, and enterprise applications [1], [2]. Second, the continued imbalance of computing performance to memory performance constricts overall application performance. Third, in HPC specifically, the plateauing performance of both I/O subsystems and interconnection networks is forcing applications to consider alternative scenarios like in situ visualization and analytics that utilize large on-node memory to reduce expensive inter-node data movement. Finally, the conventional memory technology – dynamic random-access memory (DRAM) – is approaching its limits in terms of density, power, and cost; hence, researchers are investigating new technology options [3], [4], [5] : non-volatile memory (NVM), such as 3D-XPoint [14], phase-change memory (PCM [6]), 3D NAND flash [7], and spin-transfer-torque magnetic RAM (STT-MRAM [8]). Meanwhile, high-performance volatile memory, like Hybrid Memory Cube (HMC [9]), Wide-I/O 2 (WIO2 [10]), high-bandwidth memory (HBM [11]), and GDDR6 [12], continues to be actively developed and deployed.

These solutions offer much larger bandwidth (> 1TB/s for HBM2 vs 160 to 200 GB/s for DDR based systems) at same latency level but offering a smaller capacity than today DRAM memory modules since package size is limited, and thus, if external memory is still required, an extra level in the memory hierarchy needs to be added.

In general, delivering improvements in bandwidth and latency can have major impacts on code efficiency. Thus, the challenge is to find the right balance between future memory characteristics (bandwidth, latency, size, power consumption, integration and cost) and usage (explicit data placement, automated placement or even caching). Upcoming Non-Volatile Memory (NVRAM) technologies are opening new opportunities for HPC systems. New NVRAM will feature much larger byte-addressable capacities as DRAM; hundreds of GBs vs tens of GBs allowing new approaches where only compute nodes are populated with HBM. Their performance will be much better than current FLASH based NVRAM technology approaching DRAM levels. Furthermore, their endurance should be comparable with DRAM at least in combination with some hidden wear-levelling technology. They could be used in HPC systems, both as main memory and ultra-fast I/O. It is clear than these new technologies will provide new options of memory speed and size within each node but the way to access to this memory hierarchy may increase the complexity of the codes and limit its portability. Not surprisingly, this strategy of coupling different memory layers together provides considerably more flexibility, but it also presents new risks and challenges. First, this large, multidimensional design space is expensive to evaluate with existing tools for functional or cycle-accurate simulations. Additionally, applications, code generation toolchains, and other architectural components must adapt to each memory technology under consideration. For example applications must be explicitly ported to use these new memory features, as they are not transparently managed by the hardware or the operating system. Second, these new memory systems must be designed for a specific set of applications in terms of computational intensity, working-set size, memory access patterns, parallelism, etc. If the specific application requirements and scaling predictions are inaccurate, then the ultimate design could miss the optimal balance. Third, in this period of expanding options, many of these memory systems will be unique in that they are immature and the first implementation of this memory architecture. Designer experience, existing toolchains, and performance estimates will be less obtainable and require more efforts. As a result, architects and customers are struggling to design HPC memory systems that effectively balance multiple factors of cost, performance, capacity, and power. Moreover, many diverse technologies, such as NAND flash, HBM2, GDDR6, are being rapidly improved [13], [14], [15] and must be evaluated frequently as new parameters become available. In this regard, efficient and flexible design tools for memory systems for analyzing and optimizing these options are gaining in importance.

HPC Systems Interconnect

Application performance relies on parallelism and much depends on the efficiency of the network interconnecting the compute nodes into a single system. The HPC system interconnection network must scale together with the compute nodes and the storage performance. The HPC networks bandwidth is planned to grow from a 100

Gb/s today, to 200, then 400 Gb/s in the coming years thanks to the development of new generations of SerDes (Serializer/Deserializer) circuits[24].

The minimum required bandwidth should be around 2Gb/s per MPI task, but anything up to this number will be better as it will permit to share in the same network using QoS channels the MPI communications and the parallel file-system I/O. About the latencies, the current technologies are providing around 0,5-0.9µs, adding to this number the latency of each of the hops required to get the final destination of the package, about message rate, most of the network technologies used at HPC should provide at least 100-150 million packages per second, and finally such interconnects should be able to offload MPI traffic, provide adaptive routing mechanisms and ensure end-to-end reliability of communications.

Possible options for a machine with more than 5000 nodes are either to have a non-blocking network, or define several islands with blocking (pruning) options among them. The necessity of a full non-blocking network is relative to the amount of applications using an important part of the system in a single execution and their bandwidth requirements. If the expected use of the system is with smaller executions, the most cost-effective solution is to have islands of enough cores to fulfil at least 90% of the executions and then a blocking connectivity 1:2 or 1:3 oversubscription factors between the islands. Thus, the network topology needs to be synchronized with a resource manager with the capability to manage the different islands throughout the allocation of jobs. The performance lost due to the blocking factor is application dependent, and there will be application with no lost due to a 1:3 blocking factor, when others can be severe affected by the same blocking, so an analysis of the kind of jobs executed will be an important study to take the final decision.

[24] http://emlab.uiuc.edu/jose/Theses/Yang.pdf

Storage

The main interest for HPC users is the maximum capacity and parallel performance of the file-system, in this case, the number of users using local disks is reduced. However, having a fast local storage is a good approach for specific I/O than cannot be afforded by the parallel file-system. In that sense and following the latest memory technology developments (see section above) a tiered storage solution incorporating a mix of local storage, burst buffers and HDD and finally a vast cold storage on HDD/tapes supported by a fast and smart parallel file system need to be considered.

Any usable parallel file-system needs to support MPI-IO [26], and it will also be very recommendable to have a checkpointing/restart at the system level, as one of the main problems we will face off in the future is the reliability of the applications due to high possibility of failure of one component of the system involved in large executions. In this case and for performance issues, the techniques of Fault-Tolerance in MPI [26] and from an application point of view (or ideally the system itself) will be very important to restart big executions, to manage this restarts a fast and persistent local file-system should be required, so technologies like NVRAM can be a good approach. The idea of using NVRAM in high performance computing was already investigated, especially in the area of data-intensive architectures, where usage of only DRAM is costly and power-intensive [28]. NVRAM was also already used for checkpointing of distributed parallel applications. Dong et al. [29] proposed to use hybrid local and global checkpointing using phase-change memories (PCM). Narayanan and Hodson [30] proposed to use NVRAM to make whole-system checkpoints by keeping all data in NVRAM. Gao et al. [31] created their own checkpointing system that creates partial checkpoints during application execution. It utilizes runtime idle periods to copy data from DRAM to NVRAM in order not to interfere with application execution.

<u>Virtualization and Container</u>

Virtualization is an important tool for improving user's experience while accessing HPC systems, including exploitability, reliability and security. At the node level, containers can be set-up to facilitate system administration. VMs and containers provide a flexible way to tailor the run-time environment for each user and application. They will also enforce better security as applications will be insulated from system software and other applications running on the system.

At the network level, virtualization support will allow a better Quality of Service (QoS). It will arbitrate between concurrent users, applications, data flows, and their respective priorities. Another important aspect is that it could help to improve system resiliency with an easier implementation of checkpoint/restart at the system level. It is interesting to develop virtualization at all levels of the HPC systems and in a coherent way.

For the pre-exascale and exascale machines, the hardware, the operating system and the network should be able to support the standard containers and virtualization systems, e.g. Docker[25], Singularity[26], Shifter[27], etc. and with a level of security enough for the use of these technologies in a HPC environment. Packaging and running applications via containers has the potential to enable extreme mobility in computational modelling and form an integral part of the effort to enable transnational access to HPC resources and collaboration in research. It offers a method of simplifying and standardising the building and execution of applications on diverse hardware

[25] https://www.docker.com/
[26] https://singularity.lbl.gov/
[27] https://github.com/NERSC/shifter

platforms without compromising on performance so that researchers can develop applications on their laptop or local HPC system and easily and quickly get that application running on different HPC system in other institutions or countries. An important additional benefit of this is that containerisation can also effectively capture and preserve the exact tools and environment in which the results of an analysis or simulation were derived and so help to advance the implementation of Open Science policies.

In the context of executing applications via containers, a number of factors unique to HPC need to be addressed. These include the ubiquitous use of the Message Passing Interface standard as a means of parallelising codes, the use of specialised hardware such as GPUs and the widespread use of parallel file-systems. Here we will examine the challenges each of these factors presents as well as how the alternative technologies address them where relevant.

MPI is a standard API commonly used in parallel HPC applications to pass data between processes and control the concurrent execution of many independent tasks. Numerous implementations of the standard exist such as Intel MPI, Cray MPI, MVAPICH, OpenMPI, each providing a different set of runtime libraries and configuration options. When containerising an MPI application we first need to consider how it can be made to work in terms of the correct start-up and execution of all containers involved and secondly how to make it as portable as possible so that it can run on multiple systems with minimal (if any) changes.

There are generally two ways to implement MPI from within a container: a) the entire MPI stack is packed within the container. This is the most commonly used model and can be made working with different technologies. At first glance it appears to offer the advantage of fully encapsulating the environment; however critical information such as how network addressing is set up on the host cluster and specific information on features and tunings of the HPC network is dependent on the system where the containers are going to be run. This significantly affects the portability and potential performance of the container; b) the MPI stack is split partially between the host and the container. This is the preferable approach as the MPI stack does not need to be built specifically for a target host or resource but simply requires a compatible version to be present on the hosting cluster. It also alleviates much of the networking complexities as the MPI processes in the containers can be started through the batch system or local MPI start-up method. In this scenario for example, the batch system could call the regular system installed *mpiexec*, which will then launch the containers using the set of nodes allocated to it.

Recent versions of Singularity and OpenMPI have been modified to work well together (as does Shifter and Cray MPI). Singularity detects when an OpenMPI application is being built and automatically adds all library dependencies into the image. Additional factors to consider are how MPI processes communicate between containers within the same node. Depending on which namespaces are in use, i.e. the IPC (Inter-Process Communication) namespace, the optimal use of shared memory for communications may not be available in which case the network must be used which will increase message passing latency.

GPUs and CUDA

Driven by the insatiable market demand for real-time, high-definition 3D graphics, the programmable Graphic Processor Unit or GPU has evolved into a highly parallel, multithreaded, many-core processor with tremendous computational horsepower and very high memory bandwidth. More specifically, the GPU is well-suited to address problems that can be expressed as data-parallel computations - the same program is executed on many data elements in parallel - with high arithmetic intensity - the ratio of arithmetic operations to memory operations. In November 2006, NVIDIA introduced CUDA, a general purpose parallel computing platform and programming model that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU. NVIDIA CUDA is the dominant programming framework, but any language that allows the code running on the CPU to poll a GPU shader for return values, can create a GPGPU (General-purpose computing on graphics) framework. Other common programming standards for GPU parallel computing include OpenCL[28] (vendor-independent) and OpenACC[29].

To enable a container to run a CUDA application using the host GPU, there are two approaches which can be used: a) install the GPU drivers and CUDA libraries within the container. The disadvantage here is that driver versions must match on the host and container; b) use a GPU enabled container technology. The *nvidia-docker* wrapper is able to create Docker containers with a CUDA runtime which can interoperate with the host GPU drivers thereby eliminating the need to explicitly match versions on the host and image. Singularity (version 2.4 or above) enables the use of the host GPU drivers and CUDA libraries to be automatically bound to a container at runtime thus eliminating the need to install them in the container image.

Parallel cluster file systems such as Lustre[30], BeeGFS[31] and IBM Spectrum Scale (GPFS)[32] are a key component of almost all HPC systems and enable a consistent

[28] https://www.khronos.org/opencl/
[29] https://www.openacc.org/
[30] http://lustre.org/
[31] https://www.beegfs.io/content/
[32] https://www.ibm.com/support/knowledgecenter/en/STXKQY_4.2.3/com.ibm.spectrum.scale.v4r23.doc/bl1ins_intro.htm

view of a shared persistent file system on all nodes with high performance being a primary design principle. All of the container technologies presented above allow containers to mount a directory structure from the host into the runtime container so that applications can read and write data to this filesystem without requiring specific software to be installed. Thus, for example, a user's home directory on GPFS can be mounted at runtime in a container and the application can read its input data from there and write output as the job proceeds. Differences in the use of the Network Namespace between Docker and the other container implementations suggest that there may be a performance overhead on I/O for Docker.

However the layered UnionFS[33] image format of Docker is not compatible with parallel file systems and so native Docker images must be stored on a separate ***ext4*** file-system, possibly duplicated on each node. The other technologies all use flat images that are essentially a single large file in the case of Singularity or a tar file of a directory structure which is subsequently extracted in the case of Charliecloud[34]. The single large image file used by Singularity is particularly well suited to parallel file systems which are typically optimised for this type of file. Also, for applications such as Python workflows which involve accessing a very large number of small module files, having the container image as a single file enables much higher performance in terms of startup time because the metadata servers are not overwhelmed by many nodes simultaneously performing metadata lookups on hundreds of small files. Benchmarks[35] have shown that as the number of concurrent Python interpreter increases with the size of a job, the start-up time increases due to file system contention but that this can be offset by using Singularity containers to reduce the number of file accesses (Figure 7).

[33] http://unionfs.filesystems.org/
[34] https://github.com/openhpc/ohpc/issues/712
[35] https://paperpile.com/c/C6IbYQ/Tt8t

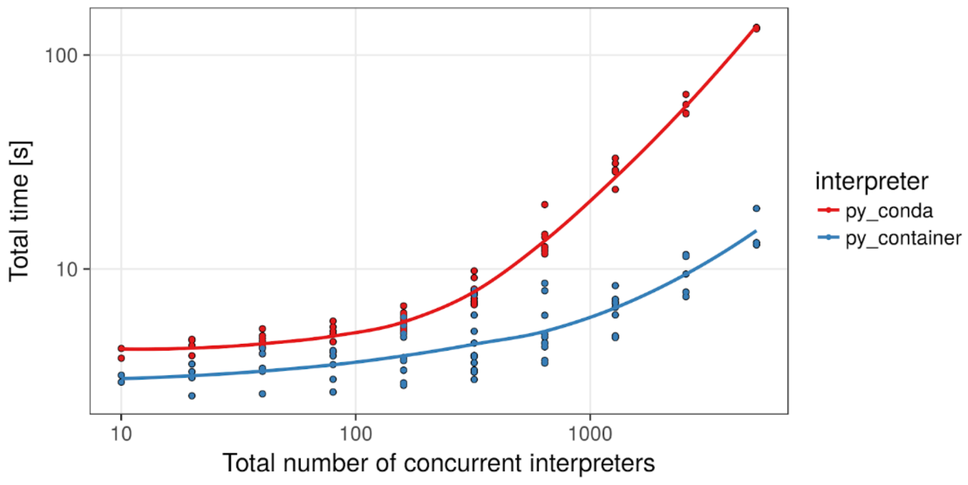


Figure 7 – Start-up time of python script on container vs regular file system install.

The first test and comparison reported below is a proof of concept test to run a MPI benchmark across multiple nodes. These tests are intended to examine and demonstrate the feasibility of running HPC applications using containers, with performance considerations coming later into the document.

For the purpose of testing we selected a very simple hybrid micro-benchmark which can be run with multiple configurations (OpenMP, OmpSs, MPI+OpenMP and MPI+OmpSs) to study how parallel applications may behave with containers and verify the BSC's DLB (Dynamic Load Balancing)[36] library is able to handle load imbalances from inside. We decided to launch our tests twice using Docker, Singularity and Shifter, one without DLB and one enabling it, using two nodes with two MPI processes each and four threads per MPI process, so in total we were running eight threads of our benchmark on each node. In the Singularity and Shifter cases, when submitting the jobs we just invoked from the login node ***mpirun*** specifying a ***hostfile*** and desired bindings,

[36] https://www.bsc.es/research-and-development/software-and-apps/software-list/dlb-library-dynamic-load-balancing

treating containers as wrappers of our program. With Docker, however, we deployed two containers as virtual nodes connected through an overlay network and ran our tests from inside since Docker's aim is to make containers as isolated and autonomous as possible.

In Figure 8 we can see the execution flow of the same executable with Docker, Singularity and Shifter respectively. In red it is represented each OmpSs' task execution (the parallel code) and in pink all the sequential code (the code outside parallel regions) that each MPI process performs, which consists of MPI calls and the sequential code of our micro-benchmark. We can also observe that the execution presents a considerable load imbalance between MPI processes 1.1, 1.2 and 1.3, 1.4 because there are threads doing nothing useful, which is represented in black since they are idle, between pink or red regions.

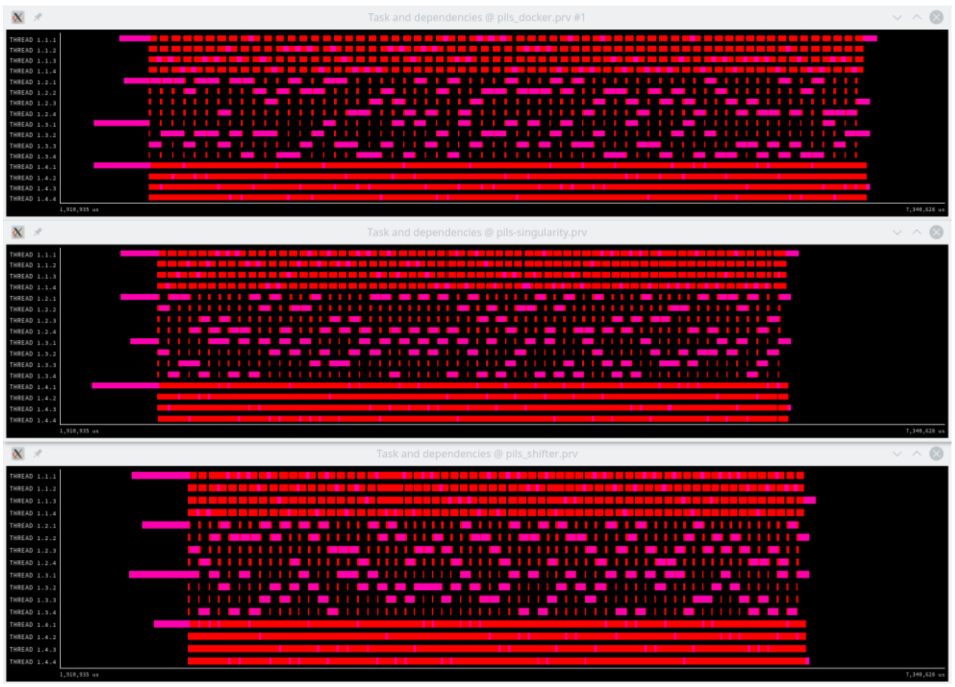

Task and dependencies @ pils_docker.prv #1
Task and dependencies @ pils-singularity.prv
Task and dependencies @ pils_shifter.prv

Figure 8 - Paraver trace without DLB using Docker (top), Singularity (middle), Shifter (bottom).

Essentially the executable behaves the same with all three implementations; nevertheless, it is evident that Docker takes some more time to complete a task than Singularity or Shifter. Considering the fact that Docker containers operate using an overlay network it makes sense there exists extra overhead due to MPI communications.

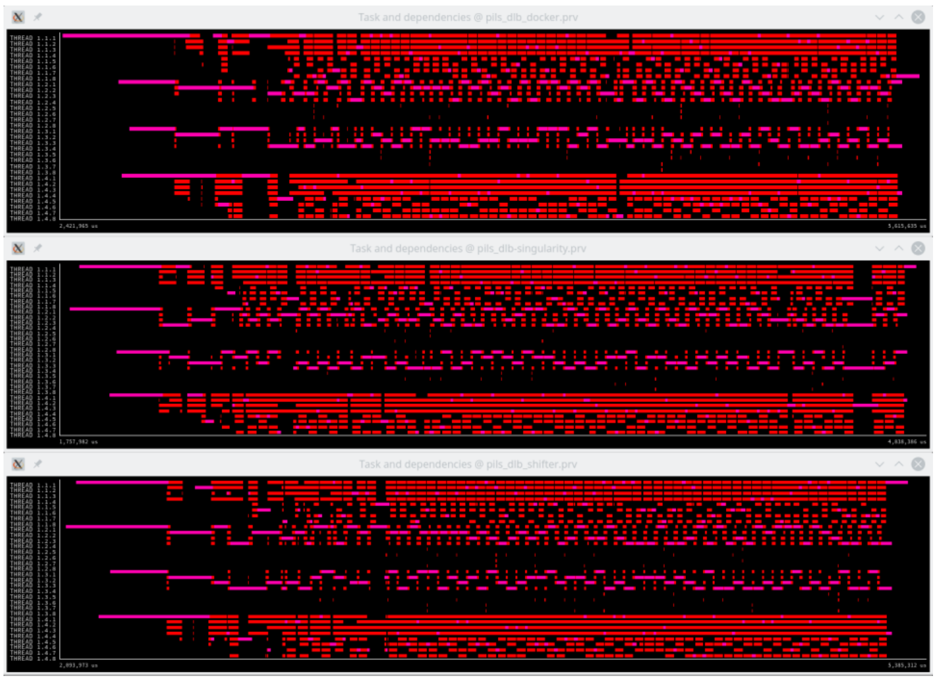

Task and dependencies @ pils_dlb_docker.prv
Task and dependencies @ pils_dlb-singularity.prv
Task and dependencies @ pils_dlb_shifter.prv

Figure 9 - Paraver trace with DLB using Docker (top), Singularity (middle), Shifter (bottom).

Figure 9 shows the flow of test application with dynamic load balancing. The three traces (Docker, Singularity and Shifter) are very similar and we can clearly appreciate the effect of enabling DLB library. Besides, each MPI process now has eight threads instead of two. The application possesses the same amount of hardware resources as in Figure 8, but with DLB each process is able to borrow the assigned CPUs of the other MPI process in the same node, so at the end one MPI process is able to potentially run on eight CPUs and therefore it needs eight threads now. If we take a closer look it seems evident how when the less loaded MPI processes 1.2 and 1.3 finish their work, instead of blocking themselves and going idle their CPUs DLB assigns that resources to the other two processes which still have computations to do. Thanks to a more efficient hardware usage because DLB avoids having CPUs in idle state, we were able to get a speedup of around 30% with respect to the same execution but without DLB enabled presented above.

Starting from these results, one may conclude that: a) it is viable to launch MPI applications with Docker, Singularity and Shifter. The benchmark presents very similar behaviour with Docker, Singularity and Shifter, thus it is logical to believe that these three implementations do not interact significantly with the application's execution. Docker brings some overhead that may be due to the implementation of the network layer. Applications seem to be easily scalable with Singularity and Shifter, with Docker however we would need a more complex deployment involving explicit network and MPI setup and configuration.

Chapter V.
Deep Learning

With the data doubling every year, data intensive applications are increasing as well as the demand of high-end resource capacity to analyse collected data sets. The explosion of analysis applications have become a major driver for revising system architecture and tools leading to the proliferation of software components and frameworks which may require multi-node and multi-core systems to scale-up and provide good performance. To this respect, Machine Learning, and in particular Deep Learning [46], is a field that is rapidly taking over a variety of aspects in our daily lives. In the realm of deep learning, it lies the Deep Neural Network (DNN), a construct inspired by the interconnected nature of the human brain. Trained properly, the expressiveness of DNNs provides accurate solutions for problems previously thought to be unsolvable, simply by observing large amounts of data. Deep learning has been successfully implemented for a plethora of subjects, ranging from image classification through speech recognition [48] and medical diagnosis, to autonomous driving and defeating human players in complex games. Although neural networks have attracted the attention of the data mining community since many years, they only rose in success once the available computational power permitted training on workstations by exploiting their inherent parallelism. As datasets increase in size and DNNs in complexity, the computational intensity and memory demands of deep learning increase proportionally. Training a DNN to competitive accuracy today essentially requires a cluster of machines with high-performance computing architectures. To harness the computational power available in such systems, different aspects of training and inference (evaluation) of DNNs have been modified to increase their underlying concurrency.

Consequently, the first successful modern DNN, such as AlexNet [49], managed to outperform existing classification methods by a factor of two, and current prevalent DNNs [47] outperform AlexNet by an additional factor of ~2.9.

However, the high popularity of these methods has resulted in numerous open-source software tools becoming accessible to the public and popular across different scientific disciplines. Conversely, with the growth of applications and tools, it is becoming difficult for researchers to estimate how much resource is needed to run their analyses and select appropriate software and hardware components. For instance training deep learning models is a time-consuming process and many software tools are being designed to exploit special hardware features such as multi-core CPUs and many-core GPUs to shorten the training time.

In this section the results of a comparative study of state-of-the-art deep learning tools benchmarked onto Cineca HPC systems is presented. The comparison takes in consideration different aspects including the impossibility to benchmark all tools available on the market, the existence of tools supporting hardware accelerators and the availability of precedent studies [19] [20]. Our results show that tested tools are able to leverage underneath system capabilities to achieve significant performance and that no single software exists that outperforms others opening space to further optimisation. Although these workloads stress all system components, such as CPU cores, memory, and storage volumes, inter-node communications might introduce a significant overhead affecting overall performance. The update of deep network weights requires many collective communications. In many circumstances if you have a large data set it is better first to consider writing a custom, optimised, in-memory version of the code rather than investing effort in trying to distribute the load on a multi-node cluster. This is also because most of the commonly available tools leverage traditional BSD sockets interface to communicate which prevent the use of RDMA (Remote Direct Data Access) present in modern low-latency networking technologies. Furthermore, only few tools make use of library supporting low level processor instructions for linear algebra, i.e. SSE, AVX, AVX2, AVX512, etc. Many machine-learning workloads are memory bandwidth bound and tools, especially those based on Java, do computation directly on compressed data and thus no easy use of inner math kernels is possible. Finally, despite the diffused belief, many tools show substantial limits while handling very large data sets resulting into the application to crash or exceed memory capacity, e.g. stack overflow.

Performance has been measured using the **convnet-benchmarks** suite[37] that tests all publicly accessible implementations of convolutional networks and includes models for object localization/detection from images/videos which won the Large Scale Visual Recognition Challenge on different years[38].

Frameworks have been selected according to their ability to support three distinct approaches for solving a deep learning task and are based on three distinct low-level libraries, Intel MKL DNN, Intel MKL, NVIDIA cuDNN respectively. This part of the document reports about running performance of deep learning tools on different types of neural networks and different hardware architectures, including Intel Broadwell, Intel Knights Landing, and NVIDIA Tesla K80, Tesla P100. Results show that tools are able to make good use of platform characteristics and reach significant performance. Studied networks are listed below:

[37] https://github.com/soumith/convnet-benchmarks
[38] http://image-net.org

- **AlexNet [22]**. Developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton, it is considered the first work that popularized Convolutional Networks in Computer Vision. The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other.
- **GoogLeNet** [23]. The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google. Its main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M).
- **Overfeat** [24]. This integrated framework, based on CNN, is winner of ILSVRC 2013 on localization task and obtained remarkable results also in classification and detection tasks. The novelty of the approach resides on the application of a CNN on multiple locations of the image and the ability to predict position and size of the bounding box surrounding objects.
- **VGG** [25]. Visual Geometry Group (VGG) of Oxford University studied the effect of the convolutional network depth on image recognition tasks accuracy. This group ranked number one on localisation and second on classification tasks ILSVRC 2014. For this benchmark, model A is taken in consideration.

Tested frameworks

Caffe

Caffe application can read prototxt files containing the definition of the network and data sources, while minimization algorithm parameters are set on a secondary text file. Python API is also available and supports model loading, I/O, visualization and model training instrumentation. In this work we are considering two different caffe development branches:

- **Caffe (bvlc)** is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and community contributors[39]. Release 1.0.0 exploits recent Pascal series GPUs thanks to CUDA 8.0 and CUDNN 6.0. Multi GPU (data parallel model) training is supported.
- **Caffe (Intel)** is a fork of the main distribution dedicated to improving Caffe performance when running on Intel Xeon processors (HSW, BDW, Xeon Phi). Computational backend is based on Intel Math Kernel Library for Deep Neural Networks (mkl-dnn)[40]: an open-source intel project that provides fast implementation of common functions for neural networks carefully designed for Intel multicores and manycores. OpenMP is used for shared memory parallelism, while hardware vectorization is supported up through latest AVX512. For this benchmark, we used mkl-dnn 0.9 and mkl/2017 and Caffe 1.0.0.

<u>Theano</u>

Theano is a Python library for the definition and optimization of mathematical functions involving n-dimensional arrays. Among different building blocks, a relevant number of layers for the implementation of neural networks are present. After the definition of the net, the library builds a C++ source that will be used for the actual training and inference phases.

[39] https://github.com/intel/caffe
[40] https://github.com/01org/mkl-dnn

- **Theano (native)** is developed at Universit de Montral4. Release 0.9.0 is based on CUDNN 5.1, while master branch can exploit GPU architectures through libgpuarray 0.6.8, CUDA 8.0 and cudnn 6.0. Although beyond the purpose of this paper, it is worth noting that multi-gpu support is achieved through the adoption of Nvidia NNCL open-source project.
- **Theano (MKL)** is a branch of the main distribution able to leverage Intel mathematical library (Intel MKL). Release 1.1 (based on native 0.9.0) is considered.

<u>TensorFlow</u>

TensorFlow is an open source software library for numerical computation using data flow graphs5. It is capable to leverage NVIDIA CUDA Deep Neural Network library (cuDNN), a GPU-accelerated library of primitives for deep neural networks6. Thanks to a close collaboration between Google and Intel, optimizations aiming at code optimization for Intel CPUs and manycores are already available on Tensorflow main branch7. Code refactoring has been carried out in order to allow SIMD vectorization (both AVX2 and AVX512 can be exploited, depending on the actual hardware); moreover a number of computational kernels are based on MKL and mkl-dnn.

<u>Neon</u>

Neon is the deep learning framework of the Intel Nervana initiative committed to achieve the best performance on all types of hardware8. The benchmarks used version 2.0.0+ that comes with Intel MKL support, which enables multi-threading operations on Intel CPU. A multi-GPU support seems to be available on Nervana Cloud only. Neon is highly optimised for Maxwell GPUs. Although it might get speedups over CPUs, note that on a pre-Maxwell GPU it might not experience the fastest performance. The benchmarks use the recommended settings on Intel architectures, that are: ***KMP***

AFFINITY=compact,1,0,granularity=fine, and the AVX512 compiler flag has been enabled on the Intel Knights Landing system. Moreover the number of threads are set to the number of cores through the OMP NUM THREADS variable. Although Neon provide python code with option for setting the batch size and other parameters, the default batch size is hardcoded in the codes.

Obtained results

This part of the document summarises the results obtained by previously introduced frameworks when running on hardware described in Table 1. Both learning and inference tasks can be characterised by a simple measure that describes the capability of the underlying network to ingest a number of images on a unit of time. Generally, when training a model, a stochastic gradient descent (SGD) approach is used to decrease the value of a given cost function. The original convnet-benchmark approach is to provide the forward-backward time (averaged over 10 runs) for a fixed batch size. This running time is related to one single iteration of a SGD, when the same batch size is selected. Different hardware can require different data sizes to better exploit their inner parallel structures. We chose to report the highest number of images per second obtained, alongside with the relative batch size. Forward time is related to the velocity of performing inference tasks, only the evaluation of a function, not its gradient, is involved.

Name	CPU	# Nodes	RAM (GB)	Accelerators
MARCONI A1	Intel Broadwell 2x Intel Xeon E5-2697 v4@2.3GHz 18 cores each	1512	128	n/a
MARCONI A2	Intel Knights Landing 1x Intel Xeon	3600	96 + 16 (MCDRAM)	n/a

	Phi7250 @1.4GHz 68 cores			
MARCONI A3	Intel SkyLake 2x Intel Xeon 8160 @2.1GHz 24 cores each	3216	192	n/a
GALILEO	Intel Haswell 2 x Intel Xeon 2630 v3 @2.4GHz 8 cores each	524	128	78 NVIDIA K80 2 NVIDIA P100

Table 1 – Cineca systems description

Concerning CPU architectures (Marconi A1 and A2), the most significant performance boosts are achieved by Intel branches of Caffe and Theano with respect to their native counterparts. For example, Theano/Intel is able to process a number of images which is 3.2 times larger in forward, 5 times larger in forward-backward evaluations, for a batch of 512 images; a similar behaviour has been observed with Caffe. For this reason, only the performance of Intel branches in A1 and A2 is considered. Optimal number of OpenMP threads seems strictly related to the effective number of cores. Thus results are obtained by fixing OMP NUM THREADS to 36 on A1 and 68 on A2.

Library	**System**	**Imgs/s (fwd/back)**	**Imgs/s (fwd)**
Caffe-intel	Marconi (A1)	60 [bs: 2048]	358 [bs: 2048]
	Marconi (A2)	69 [bs: 2048]	493 [bs: 2048]
Caffe (native)	Galileo (K80)	107 [bs: 512]	319 [bs: 512]
	Galileo (P100)	458 [bs: 1024]	1450 [bs: 1024]
TensorFlow	Marconi (A1)	131 [bs: 1024]	424 [bs: 1024]
	Marconi (A2)	214 [bs: 1024]	709 [bs: 1024]
	Galileo (K80)	258 [bs: 512]	753 [bs: 512]
	Galileo (P100)	652 [bs: 512]	1769 [bs: 512]
Theano-intel	Marconi (A1)	112 [bs: 1024]	326 [bs: 1024]
	Marconi (A2)	174 [bs: 1024]	582 [bs: 1024]
Theano (native)	Galileo (K80)	80 [bs: 512]	226 [bs: 512]
	Galileo (P100)	311 [bs: 512]	1261 [bs: 512]

Neon	Marconi (A1)	120 [bs: 4096]	370 [bs: 4096]
	Marconi (A2)	189 [bs: 2048]	603 [bs: 1024]
	Galileo (K80)	61 [bs: 1024]	226 [bs: 128]
	Galileo (P100)	1131 [bs:1024]	3371 [bs: 1024]

Table 2 - Performance summary of a number of frameworks, considering Overfeat neural network. Both forward (fwd) and forward-backward (fwd/back) number of images per second are considered, between square brackets is the batch size.

Table 2 shows that TensorFlow performance is both remarkable and well balanced among all considered architectures. Neon seems to be highly optimised for recent NVIDIA Pascal architecture, i.e. Tesla P100 card, with a factor of at least 2 with respect to others. On the other side, it performs poorly on older cards. Vectorized hardware, i.e. Intel Knights Landing, can be fruitfully exploited by TensorFlow (after enabling Intel optimisation described in Subsection 2.4.3) and benchmarks show comparable performance with respect to NVIDIA Tesla K80 GPU. From Figure 10 and Figure 11 it is clear that the performances are not heavily impacted by the selection of a suitable batch size.

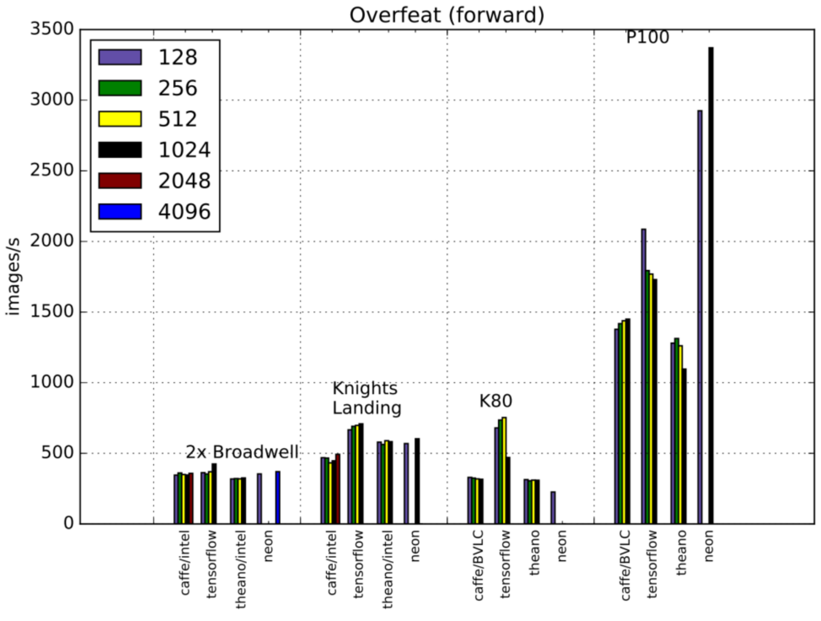


Figure 10 - Performance distribution, when varying the batch size: forward pass on Overfeat network is considered.

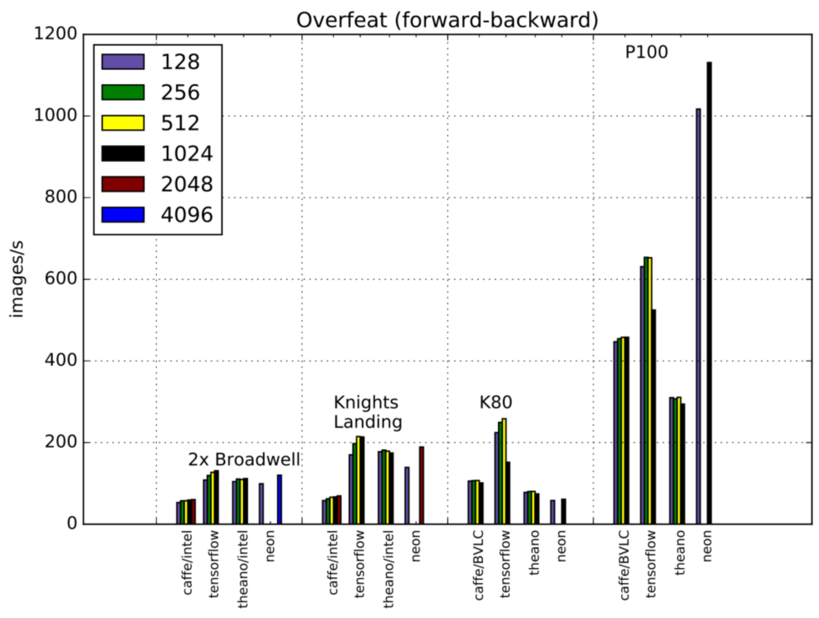


Figure 11 - Performance distribution, when varying the batch size: forward-backward pass on Overfeat network is considered.

A further series of tests has been conducted in order to find the optimal conditions for Tensorflow 1.10.0 on the partition A3 of Marconi. As a first step, tests were performed on one node only and in a second time, they were then extended to multiple nodes. An A3 node comprises two sockets each with 24 Skylake cores Intel Xeon 8160. Images from the synthetic version of the database ImageNet[41] were utilized both during the training and inferring phases. The convolutional neural network used has been RESNET50 [64], which was run for 100 iterations. Tests were launched specifying the following SLURM[42] options:

- *--nodes=N,*
- *--ntasks-per-node=n,*
- *--ntasks-per-socket=n/2,*

[41] http://www.image-net.org/
[42] https://slurm.schedmd.com/

- *--cpus-per-task=48/n,*
- *--walltime=01:00:00*

where N refers to the requested number of nodes and n indicates the number of cores requested per node and corresponds, when asking only one node, to the total number of desired mpi tasks. Furthermore, the ambient variable OMP NUM THREADS is set to 48/n, equivalent to the quantity specified in *--cpus-per-task*. We verified that not specifying the number of tasks per socket and the number of cpus per tasks leads to performance degradation. The parameter kmp affinity was set to *granularity=fine,verbose,compact,1,0,* while horovod fusion threshold was chosen as 64 GB.

A detailed parameter scan has been carried out varying the following quantities:

1. batch size per processor: 32, 64, 128, and 256 images;
2. number of mpi tasks (omp threads): 1 (48), 2 (24), 4 (12), and 8 (6);
3. number of inter threads[43]: 1, 2, 4, 8, and 16;
4. kmp blocktime[44]: 1 and 20.

In the performed tests the value of intra threads3 was set to the number of omp threads. We verified that the simulation failed when indicating more mpi tasks due to memory limitation. Results are reported in Figure 12, which shows the number of processed images per second versus the number of MPI tasks for different values of *kmp blocktime,* batch size and number of inter threads. The best result has been obtained using 2 MPI tasks, 24 OMP THREADS with 24 intra threads and 1 inter thread, batch

[43] Thread pool used by TensorFlow to run concurrently operations that are independent in the graph.
[44] Time, in milliseconds, that a thread should wait, after completing the execution of a parallel region, before sleeping.

size/proc = 256 (global batch size = 512), and kmp blocktime = 20. In this case 97.79 images per second were processed, while the same setup, but a smaller kmp blocktime, leads to 95.22 images per second. In general, it can be observed that a smaller value (1 or 2) for the inter threads parameter is more beneficial in term of performances. For batch size of the order of 128 and 256, the highest number of processed images per second is obtained selecting 2 MPI tasks. For smaller batch sizes, results indicate that good performances can be achieved even with an higher number of mpi tasks (and lower number of omp threads). It is interesting to underline the performances obtained considering a batch size of 64 with 8 mpi tasks (6 omp e intra threads) for a global batch size of 512, 2 inter threads and kmp blocktime = 1. In this case, Tensorflow could process up to 91.79 images per second. This result is encouraging especially with the perspective of using more nodes and so smaller batch size per mpi task. We underline that, since only 100 iterations were considered, data on the accuracy are not reported because meaningless.

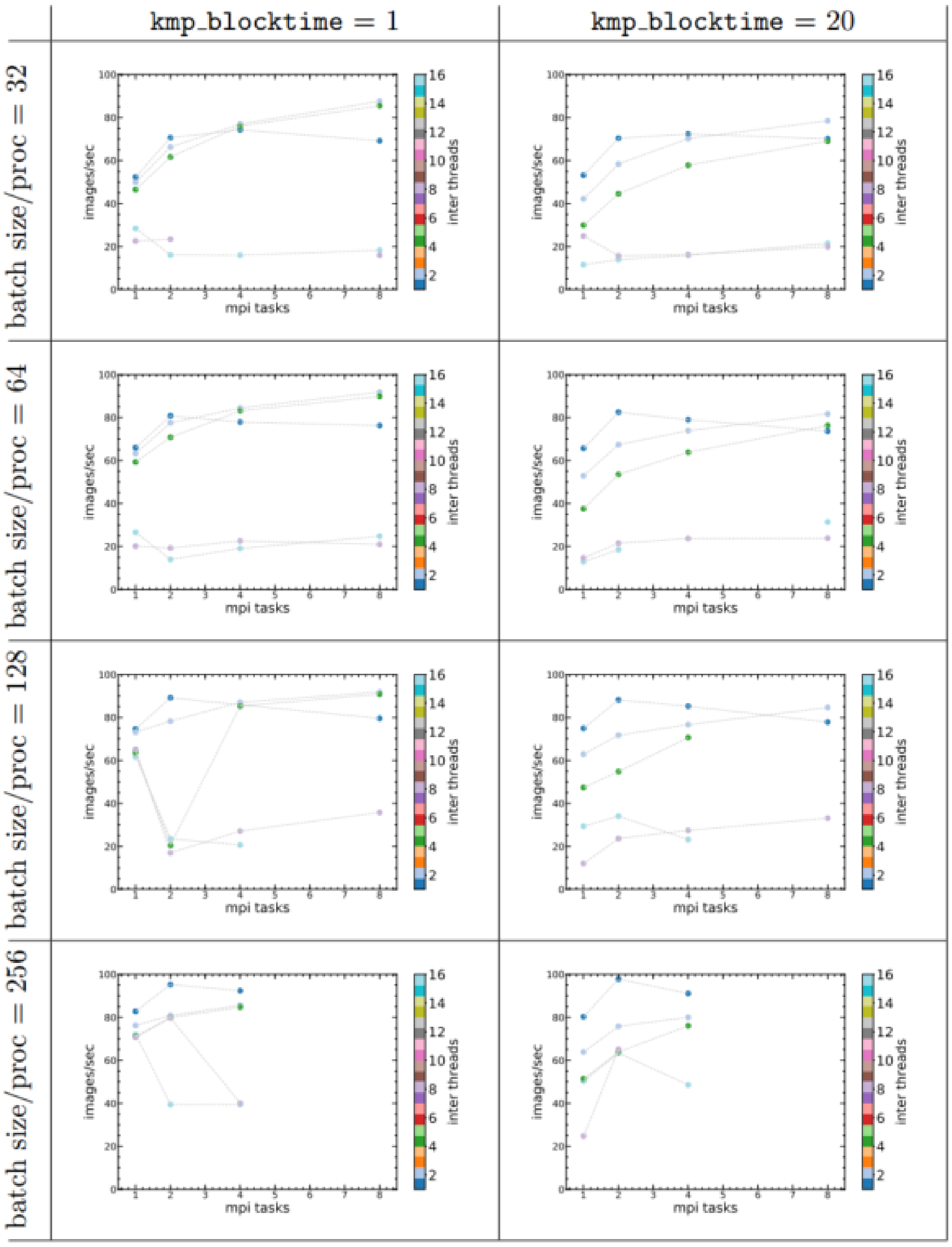


Figure 12 - Intranode test results for kmp blocktime = 1 (left) and 20 (right). The plots report the images processed per second as a function of the number of mpi tasks (and implicitly the number of omp and intra threads, since these have been set as the ratio between 48 and the number of mpi tasks). The batch size per processor is increasing from the top towards the bottom. Different colors have been used for the number of inter threads: 16 (cyan), 8 (violet), 4 (green), 2 (light blue), and 1 (dark blue).

The previous tests indicate as optimal setup in term of performances the following:

1. batch size per processor: 256 images;
2. number of mpi tasks per node (omp e intra threads per node): 2 (24);
3. number of inter threads: 1;
4. kmp blocktime: 20.

This setup was then utilized to perform tests on more than one node. In particular a scalability test using 2, 4, 8, 16, 32, and 64 nodes (and as a consequence 4, 8, 16, 32, 64, and 128 MPI processes) has been carried out. Results are illustrated in Figure 13, where the number of processed images per second is reported versus the number of MPI tasks. The plot shows that the scaling is linear.

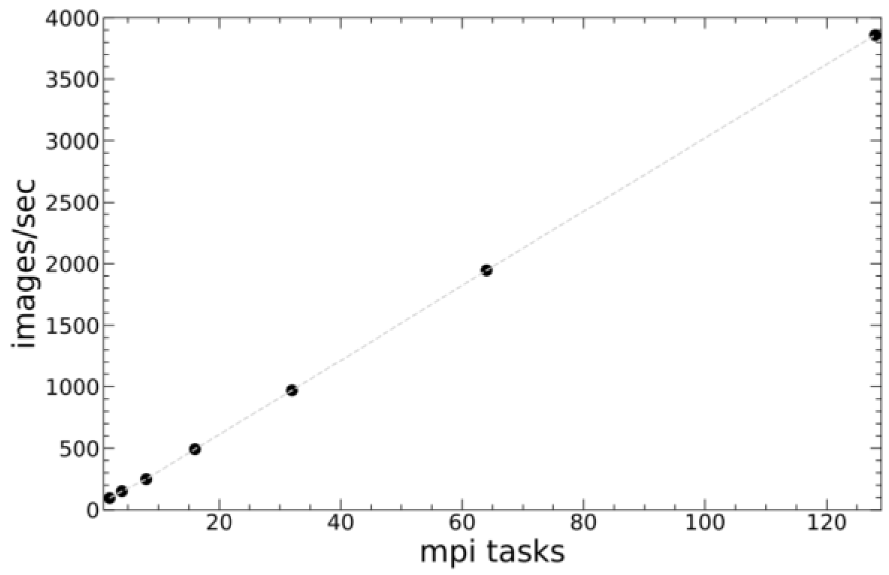


Figure 13 - Processed images per second versus number of mpi tasks. A batch size per processor of 256 images has been considered. The other parameters were chosen according to the tests performed on one node: 2 mpi processes per node, 24 omp threads, 1 inter thread, and kmp blocktime = 20.

The same scalability test has been performed also for a case with a smaller batch size per processor. In particular, the following setup has been considered:

1. batch size per processor: 64 images;
2. number of mpi tasks per node (omp e intra threads per node): 8 (64);

3. number of inter threads: 2;
4. kmp blocktime: 1,

because on one node it led to more than 90 images processed per second. Results are shown in Figure 14. Also in this case, the scaling over more than one node is quite linear, with the performances dropping a little when using 64 nodes.

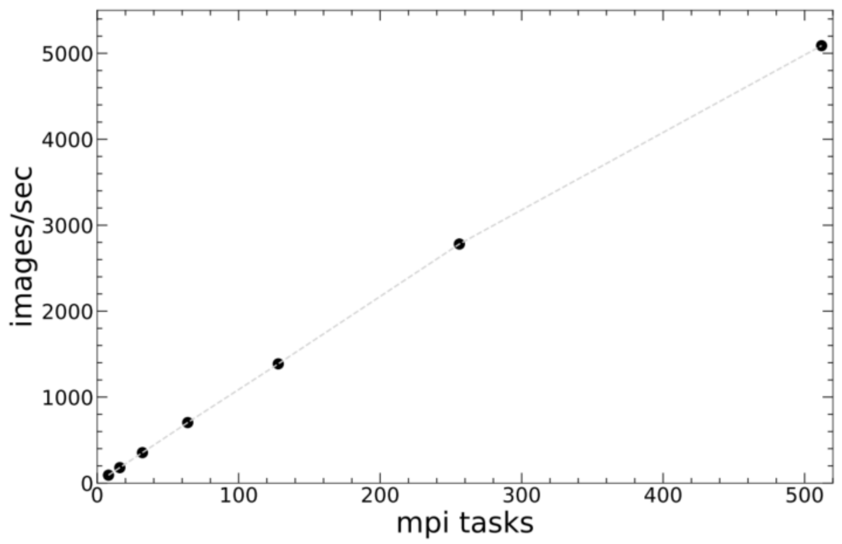


Figure 14 - Processed images per second versus number of mpi tasks. A batch size per processor of 64 images has been considered. The other parameters were chosen according to the tests performed on one node: 8 mpi processes per node, 6 omp threads, 2 inter threads, and kmp blocktime = 1.

Chapter VI.

Apache Hadoop and Apache Spark

To perform even a simple analysis over a large amount of data in a reasonable amount of time, it is desirable to have some mechanisms that are able to distribute processing tasks over distributed resources without introducing much complexity or re-engineering the whole application. Furthermore, as multiple science fields have started to use analytics for filtering results between coupled simulations (e.g. materials science or climate) or extracting interesting features from high throughput observations (e.g. telescopes, particle accelerators), the incentives for the deployment of large scale data analytics tools on High Performance Computing systems are growing.

The framework that is attracting attention for its capability to distribute processing of petabyte-class big data is Apache Hadoop. Together with its distributed file system HDFS[45] and its implementation of the Map-Reduce programming model, Apache Hadoop is able to bring the computation to the places where data is stored. Hadoop [34] and, successively Spark [34], provide a high level and productive programming interface for large-scale data processing and are able to operate on datasets which are larger of the memory capacity of a single compute node. Through specialized runtimes they reach good performance and resilience on data center systems for a robust ecosystem of application specific libraries [36][37][38].

[45] Hadoop Distributed File System

They implement the Map-Reduce model [39] using an abstraction layer in which programs are expressed as data flow graphs. The nodes in the graph are of two types: map operations, which are purely local, and reduce operations, which can involve communication between multiple nodes. The traditional Map-Reduce framework [39] is limited to acyclic graphs, preventing efficient representation of iterative methods, and it uses data redundancy to provide resiliency. In contrast Spark can handle cyclic and acyclic graphs, and provides resiliency through resilient distributed datasets [40] (RDD), which carry sufficient information to re-compute their contents. In particular, the ability to express iterative algorithms accelerated Spark's adoption that continues to attain new problem domains.

To leverage on these scalability characteristics and enable fast processing of scientific data sets, we developed a prototype software package able to process binary data through spawning Map-Reduce tasks while introducing minimum computational overhead and without modifying existing application code. The code was developed to analyse climate data but it can be generalized to cover other use cases, including neuroscientific ones.

The package is formed by the combination of two tools, ***Pipistrello***, a Java utility that allows users to execute Map-Reduce tasks over any kind of binary file, and ***Tina*** a lightweight Python library that building on top of ***Pipistrello*** is able to process scientific dataset, including NetCDF[46] files. We benchmarked the combination of this two tools using a test Apache Hadoop Cluster (4 nodes) and a relatively small data set (200 GB), obtaining encouraging results. When using larger clusters and larger storage space, *Tina* and

[46] https://www.unidata.ucar.edu/software/netcdf/

Pipistrello should be able to scale-up and analyse hundreds of Terabytes of scientific data in a faster, easier and possible efficient way.

Pipistrello

Pipistrello[47] is a Java utility very similar in philosophy to Hadoop-Streaming that allows users to run Map-Reduce jobs over any kind of binary files. Just as Hadoop-Streaming, Pipistrello requires mapper and reducer scripts to comply with a set of conventions which are fundamental to build the common processing chain ”input — mapper — reducer — output” and to get the four steps be correctly linkable together. Most of the magic of Pipistrello lies inside its implementation of the Mapper *HdfsToLocalTranslator* classes. The class Mapper uses the class *HdfsToLocalTranslator* to get the local path of the data corresponding to the input that the MapTask needs to run over. The class *HdfsToLocalTranslator* is in charge of interacting with the *NameNode*, without starting any new JVM process, to get the name of the local file corresponding to a given HDFS file. Once the name is leaked inside the *Mapper* class, it can be given to the mapper script as an argument13. After the mapper script is executed, the *Mapper* class will put its output (a file) onto the HDFS volume. It will also read the ***stdout*** of the mapper script (a filename) and set it as the value of the (*key,value*) pair that the Map-Reduce framework will send to the reducer. Map-Reduce will start a *ReduceTask* in any available node with enough resources to run it. At this point, (key, value) pairs are going to be fed to the Reduce class. The Reducer class will open a text file files to reduce.txt and write all the values (filenames), one for each line. At the same time, the Reducer class will download these files from the HDFS to make them available locally. Finally, the Reducer Class will launch

[47] https://pipistrello.readthedocs.io

the reducer script, giving it as an argument the file files to reduce.txt and its output (a file) is uploaded to HDFS.

<u>Tina</u>

Tina[48] is lightweight Python library that sits on top of:

- Iris[49] to handle NetCDF data;
- Snakebite[50] to efficiently communicate with HDFS;
- Pipistrello to launch Map-Reduce jobs over a scientific dataset.

Despite its promises for good scalability and performance, Apache Hadoop retains some drawbacks that prevent NetCDF5 files to be analysed out-of-the-box for the following reasons:

- Apache Hadoop is designed and highly optimized to read and write text data while weather and climate scientists work with binary data in the form of NetCDF files;
- HDFS stores the data by splitting every file in blocks and distributing those blocks among different nodes. NetCDF files cannot be easily split without corrupting the data;
- Java is the most used programming language for MapReduce applications while climate scientists codes are written in every programming language except Java.

[48] https://tina.readthedocs.io
[49] https://scitools.org.uk/iris/docs/latest/iris/iris/fileformats/netcdf.html
[50] https://github.com/spotify/snakebite

Preliminary tests were performed by generating 4-D arrays of synthetic data (doubles representing temperature values) with different sizes using a 4 nodes Apache Cluster (16GB RAM - 8 cores each). The arrays were then distributed among several NetCDF data sets in chunks of 1284 (which sums to a global size of ∼ 2 GB). For the sake of testing, only two different array sizes were tested, (128 × 2) distributed over 16 files and (128 × 3) distributed over 81 files. The analysis made to test the performance of the Apache Hadoop cluster, running Tina and Pipistrello Map-Reduce jobs, consisted on computing the time-height-average temperature of each of the whole arrays. Figure 15 shows speed-up for the analysis of these arrays. By comparing those numbers, it is evident that, for the array sizes tested and the chunking of the data, Pipistrello and Tina provide good results. The Map-Reduce analysis done for the smaller array takes 1=3 of the time the best local approach (a local Map-Reduce approach, as the big amount of data does not fit in memory), while, for the larger array, it takes only 1=2:5 of the time. Pipistrello and Tina are experimental software. Further enhancements are still needed to make it as stable and usable as Luigi and Mrjob are. The bright side, however, is that this work has traced the path and paved the road to future development. In the immediate future, the software will be used for the re-analysis of the Mediterranean Sea biogeochemistry produced in the frame of the EU Copernicus Marine Environment Monitoring Services [50] and more broadly to any large coupled Ocean-Atmosphere climatic dataset that will be produced targeting the Mediterranean region in the framework of CORDEX phase 2 FPS (Flagships Pilot Studies) over the European region [51].

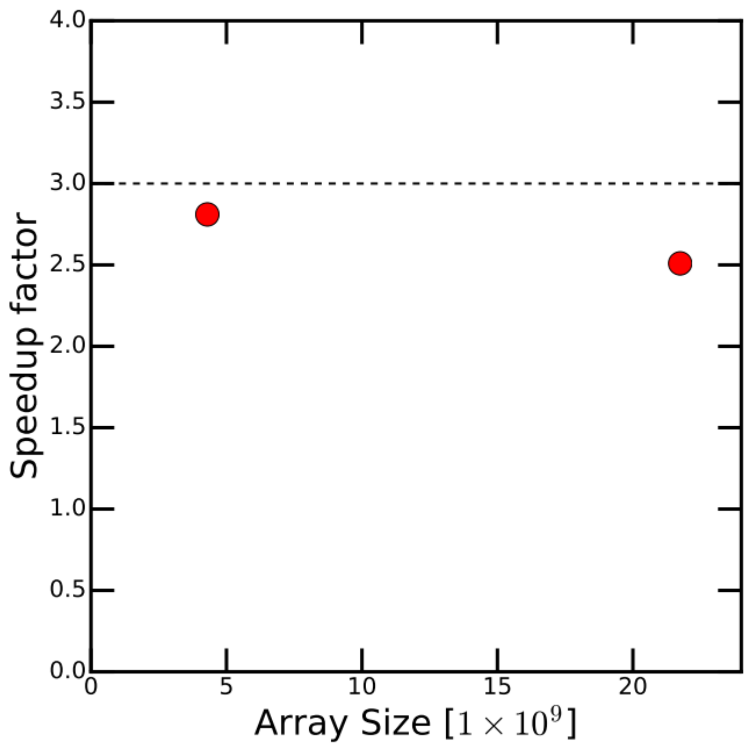


Figure 15 - Speedup reached for each case (red points) and the ideal speedup (3, for our 3-node cluster) marked by the dotted line.

Using Apache Hadoop and Spark in a HPC context

Hadoop and Spark are designed to implement stateless operation for resilience and parallelism purposes by opening and closing the files involved in each individual data access. While doing this operation they assume that the computing nodes are equipped with local disks which usually do not affect the overall job time for opening or creating a file. However, what is valid for a standard data centre does not apply for HPC systems where file metadata operations are a common scalability bottleneck. In a distributed data centre, environment disk I/O is optimized for latency by using local disks and the network between nodes is optimized primarily for bandwidth. In contrast, HPC systems use a global parallel file system, with no local storage: disk I/O is optimized primarily for bandwidth, while the network is optimized for latency.

Consequently, any effort on scaling these frameworks up and out on an HPC installation has first to address the data management concern.

For instance, the data movement is one of the performance determining factors in any large scale system. In Spark, data is logically split into partitions, which have an associated worker task. A partition is further subdivided into blocks. A block is the minimum unit of data movement and execution task. Figure 16 shows the Spark compute engine interaction with the block and shuffle managers, which control data movement. The ***BlockManager*** handles application level input and output data, as well as intermediate data within the Map stages. The ***ShufflueManager*** handles runtime intermediate results during the shuffle stage.

For the sake of reliability and scalability, Spark manipulates data combining global as well as local scope. Application level data (RDDs)[51] use a global naming space while intermediate data blocks generated throughout execution have a local scope. However, since objects may exceed the capacity of the node physical memory but still requested to be efficiently moved through the vertical storage hierarchy, managing logical naming schemes with underlying system architecture is a challenge. For instance, when global object is distributed across multiple storage volumes a long latency naming service may be needed to locate its physical location. Conversely, any locally named object stored in a physically shared storage may experience unexpected resource contentions while servicing requests. A current research direction in the Spark community aims at proving an efficient

[51] https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-rdd.html

global naming service able to reduce network traffic. Within HPC installations, the global, usually parallel, file system implements the global naming service. As anticipated above, data may move across two dimensions, vertical and horizontal respectively, for computing.

Vertical Data Movement

Vertical data movement refers to the movement through the entire memory hierarchy, including persistent storage. It is needed to move input data blocks into the memory for processing and for storing output data to the persistent storage. To minimize vertical movement for RDDs, Spark allows persisting data in the fast level of memory. As fast memory is capacity constrained, the Spark runtime assigns the task of moving objects across the memory hierarchy to a block manager. Whenever the working set size (input data or intermediate results) exceeds memory capacity, the block manager may trigger vertical data movement. The block manager may also decide to drop a block, in which case its later access may trigger additional vertical data movement for recomputation. Research efforts such as Tachyon [52] aim to reduce expensive (to storage) vertical data movement by replacing it with horizontal (inter-node) data movement. In network-based storage systems, a critical [53][54] component to the performance of vertical data movement is the file setup stage (communication with the metadata servers).

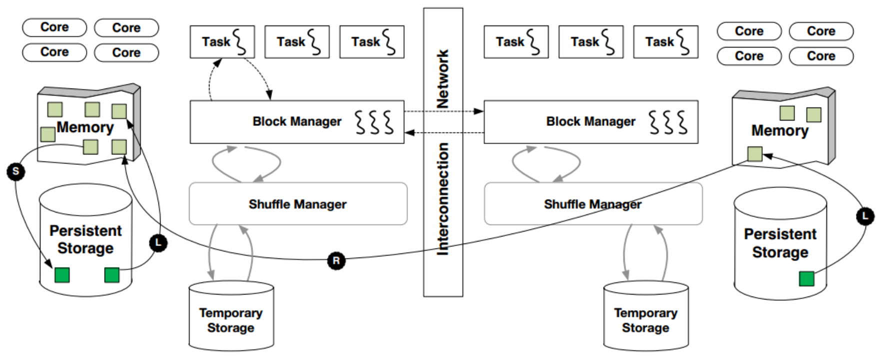


Figure 16 - Data movement in Spark and the interaction with the memory hierarchy.

Horizontal Data Movement - Block Shuffling

The horizontal data movement refers to the shuffle communication phase between compute nodes. Spark assigns the horizontal data movement to the shuffle manager and the block manager. A horizontal data movement request of a block could trigger a vertical data movement because a block may not be resident in memory. Optimizing the performance of horizontal data movement has been the subject of multiple studies [55][56][57], in which hardware acceleration such as RDMA is used to reduce the communication cost. The benefit of these techniques is of less importance on HPC systems with network-based storage because the performance is dominated by vertical data movement.

Many configuration alternatives have been developed to circumvent the performance penalty affecting file I/O behaviour. One solution is to utilize bigger and better hardware to the problem - for instance - to increase the ration between memory capacity and cores concurrency. However, as demonstrated in precedent works, while increasing the number of cores per node improves performance, it does not mitigate enough the effects of the file system as the PCI bus gets congested rapidly. Another solution relies on using *BurstBuffer* I/O subsystem[52], large NVRAM array situated close to the CPU designed to improve throughput for small I/O operations and for data pre-staging. Apart from the high cost of this solution, the question remains if it is well suited for the access patterns performed by Spark. Besides exploiting hardware upgrades, software techniques

[52] https://www.cray.com/products/storage/datawarp

can also alleviate some of the metadata performance bottlenecks. The first and most obvious solution is to use a memory mapped file system (e.g. */dev/shm*) as a secondary storage target. Subject to physical memory constraints, this eliminates a large fraction of the traffic to the back-end storage system. Although easy to implement, this solution comes with several limitations:

- the job crashes when memory is exhausted;
- since data is not written to disk it does not provide any resilience and persistence guarantees. Furthermore, for medium to large problems and long running iterative algorithms Spark will fail during execution due to lax garbage collection in the block and shuffle managers [41][42].

Unlike data centers configuration, where network performance dominates, the global file system metadata overhead in fopen limits Spark scalability to O(100) cores. If configured to use “local” file systems for the shuffle stage, Spark can reach O(10,000) cores.

Chapter VII.
Quantum Computing

Quantum computing (QC) is essentially harnessing and exploiting the amazing laws of quantum mechanics to process information [58]. A traditional computer performs operations on long strings of “bits”, which encode either a zero or a one. A quantum computer, on the other hand, uses quantum bits, or qubits. A *qubit* is a quantum system that encodes the zero and the one into two indistinguishable quantum states. The *qubit* takes its final value (0 or 1) only once read. However, since qubits behave ***quantumly***, one can capitalize on the phenomena of "superposition" and "entanglement" to build a quantum system. Superposition is essentially the ability of a system to be in multiple states at the same time, i.e. it can be “here” and “there,” or “up” and “down” at the same time. Entanglement is an extremely strong correlation that exists between quantum particles. The correlation is so strong that two or more quantum particles can be inextricably linked in a perfect unison although physically separated by a huge distance. Thus, thanks to these two characteristics, a quantum computer can execute a vast number of calculations simultaneously. To simplify, while a digital computer works with ones and zeros, a quantum computer will have the advantage of using ones, zeros and their “superpositions”.

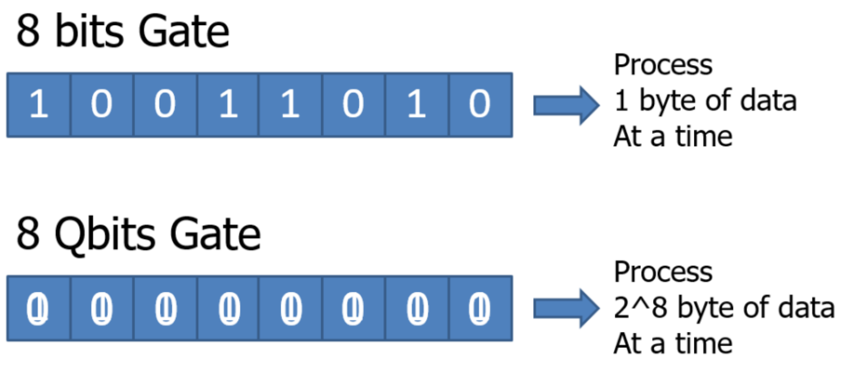


Figure 17 - Upper panel: digital computer gate can act on data one bity at a time. Lower panel: quantum computer gate can act on all possible states of the 8 Qbits at the same time.

The superposition of states of a quantum system is the key feature that allows a quantum computer to solve complex problems at a speed that is far beyond any comparison with a typical digital computer, e.g. a quantum computer can solve in principle a problem with factorial complexity into a single instruction. Essentially, a full featured quantum computer with a complete instruction set in theory would outperform any digital computer, but the reality is quite far from that assumption.

These properties make QC very attractive for HPC, where quantum computing could be used in synergy with digital computers to speed-up complex applications otherwise unaffordable on digital supercomputers. Traditional HPC systems could take care of all the instructions that cannot be executed on the quantum computer, i.e. I/O, control code, firmware, etc. while offloading resolutions of complex problems to the quantum engine by mean of a dedicated library engineered using the quantum logic. On the other hand, small quantum computers can be simulated today on large supercomputers, enabling the development of the methods and software in a user friendlier environment.

In terms of innovation trends, there are many companies with significant venture capital support investing in quantum technology. Google is venturing on the superconducting qubit technology [62], pursuing both digital and analog quantum computation, doing both theoretical and experimental research. They are pushing the field to the commercialisation of new devices and services for users. Their “manifesto” for quantum technologies was recently published in Nature [3]. Google’s Quantum Artificial Intelligence team[53], directed by Hartmut Neven, identifies three applicative priorities, such as quantum simulation, quantum-assisted optimization, quantum sampling, technical hurdles and business opportunities. The main hurdle is scaling the number of qubits while maintaining coherence; error correction is a big issue, because it will require many additional qubits. Specifically, quantum simulation applications can aim at better computational chemistry, machine learning, logistics, financial portfolio management, drug design and cyber security. However, these application areas will probably be based upon a hybrid platform that combines classical and quantum computing in a single environment to get the best from both domains.

[53] https://research.google.com/pubs/QuantumAI.html

Chapter VII.

Conclusion

The scientific community's interest in large-scale infrastructure has been continuously increasing over the last decades, with innovations in data-driven workloads and in-situ processing of simulation. It is expected that this trend will not cease in the near future, but rather persist. A review of the state of art of algorithms development in computational science shows numerous examples of initial adaptations to the challenges posed by Big Data. The whole Deep Learning movement is a clear example of how dated data analysis algorithms have become crucial in many scientific fields thanks to the availability of large computing resources and high-quality data. The same applies to data processing technologies - à la Apache Hadoop – that, originally developed to index web resources, are now permeating the scientific fields for being extremely simply to use. This dissertation's subject matter lies into the intersection of these two distinct aspects. On one hand it presents the details of a large digital infrastructure capable to process very large amount of data by federating distributed resources, on the other it reports about concrete examples on how to employ Big Data technologies in a profitable manner to solve scientific challenges. The findings of our work lead to some generic conclusions. HPC has become a prominent instrument to conduct scientific experiments, as most of the physical phenomena can be model and simulated at a very high level of accuracy. Ever growing demand of computation resources keeps pushing the optimization and development of new technologies but traditional systems architecture will soon reach expansion limits mainly due to energy efficiency cap. With this respect, Quantum Computing represents a concrete alternative, but it may require more than a decade to achieve its supremacy. Finally, along with a further investigation of scientific use cases, this work could be extended to cover other domains and evaluate new technology areas.

Acknowledgments

I am grateful to my supervisor Prof. Sonia Bergamaschi as well as all the members of the DB Group who took me over this long journey. Profound thanks go to my Cineca colleagues and members of the ICEI project. Without their help, support and contribution, it would have been impossible to complete this work.

I am grateful to my family and all my beloved friends who got me to the point in my life I am at right now, and that I like a lot.

Finally, the biggest thank is for my partner Francesca, for her support, patience, understanding and love, who helped me enormously no matter my humor was.

In memory of my father for his endless sense of curiosity, patience and gentleness.

Bibliography

[1] Huynh Phung Huynh, Andrei Hagiescu, Weng-Fai Wong, and Rick Siow Mong Goh. *Scalable framework for mapping streaming applications onto multi-GPU systems*. In Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'12), pages 1–10, 2012.

[2] Amit Sabne, Putt Sakdhnagool, and Rudolf Eigenmann. *Scaling largedata computations on multi-GPU accelerators*. In Proceedings of the 27th International ACM Conference on International Conference on Supercomputing (ICS'13), pages 443–454, 2013.

[3] Adrian M Caulfield, Joel Coburn, Todor Mollov, Arup De, Ameen Akel, Jiahua He, Arun Jagatheesan, Rajesh K Gupta, Allan Snavely, and Steven Swanson. *Understanding the impact of emerging nonvolatile memories on high-performance, I/O-intensive computing*. In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–11. IEEE Computer Society, 2010.

[4] M. H. Kryder and K. Chang Soo. *After hard drives: What comes next?* IEEE Transactions on Magnetics, 45(10):3406–3413, 2009.

[5] J. S. Vetter and S. Mittal. *Opportunities for nonvolatile memory systems in extreme-scale high performance computing.* IEES Journal of Computing in Science and Engineering, 17(2):73–82, 2015.

[6] Benjamin C Lee, Engin Ipek, Onur Mutlu, and Doug Burger. *Architecting phase change memory as a scalable DRAM alternative*. In ACM SIGARCH Computer Architecture News, volume 37, pages 2–13. ACM, 2009.

[7] Ki-Tae Park, Sangwan Nam, Daehan Kim, Pansuk Kwak, Doosub Lee, Yoon-He Choi, Myung-Hoon Choi, Dong-Hun Kwak, Doo-Hyun Kim, Min-Su Kim, et al. *Three-dimensional 128 Gb MLC vertical NAND flash memory with 24-WL stacked layers and 50 MB/s high-speed programming*. In IEEE Journal of Solid-State Circuits, 50(1):204–213, 2015.

[8] Cristian Zambelli, Gabriele Navarro, Veronique Sousa, Ioan Lucian Prejbeanu, and Luca Perniola. *Phase change and magnetic memories for solid-state drive applications*. Proceedings of the IEEE, 105(9):1790–1811, 2017.

[9] HMC Consortium. Hybrid Memory Cube Specification 2.1. http://hybridmemorycube.org/files/SiteDownloads/HMC-30G-VSR HMCC Specification Rev2.1 20151105.pdf, 2015.

[10] JEDEC JESD229-2. Wide I/O 2 (WideIO2), 2014.

[11] JEDEC JESD235A. High bandwidth memory (HBM) DRAM. JEDEC Solid State Technology Association, Nov 2015.

[12] JEDEC JESD250. Graphics double data rate 6 (GDDR6) SGRAM standard. JEDEC Solid State Technology Association, Jul 2017.

[13] W. Cheong, C. Yoon, S. Woo, K. Han, D. Kim, C. Lee, Y. Choi, S. Kim, D. Kang, G. Yu, J. Kim, J. Park, K. W. Song, K. T. Park, S. Cho, H. Oh, D. D. G. Lee, J. H. Choi, and J. Jeong. *A flash memory controller for 15us ultra-low-latency SSD using high-speed 3D NAND flash with 3us read time*. In 2018 IEEE International Solid - State Circuits Conference- (ISSCC), pages 338–340, 2018.

[14] Jin Hee Cho, Jihwan Kim, Woo Young Lee, Dong Uk Lee, Tae Kyun Kim, Heat Bit Park, Chunseok Jeong, Myeong-Jae Park, Seung Geun Baek, Seokwoo Choi, et al. *A 1.2 V 64Gb 341GB/S HBM2 stacked DRAM with spiral point-to-point TSV structure and improved bank group data control. In Solid-State Circuits Conference-(ISSCC)*, 2018 IEEE International, pages 208–210. IEEE, 2018.

[15] Young-Ju Kim, Hye-Jung Kwon, Su-Yeon Doo, Yoon-Joo Eom, YoungSik Kim, Min-Su Ahn, Yong-Hun Kim, Sang-Hoon Jung, Sung-Geun Do, Chang-Yong Lee, et al. *A 16Gb 18Gb/S/pin GDDR6 DRAM with per-bit trainable single-ended DFE and PLL-less clocking. In Solid-State Circuits Conference-(ISSCC), 2018 IEEE International*, pages 204–206. IEEE, 2018.

[16] Ján Antolík and Andrew P. Davison Arkheia, *Data Management and Communication for Open Computational Neuroscience*, Front. Neuroinform., 05 March 2018 | https://doi.org/10.3389/fninf.2018.00006.

[17] R. Shane Canon, D. Jacobsen, *Shifter: Containers for HPC,* CUG16 Proceedings, 2016.

[18] Wilkinson M., Dumontier M., Mons B. et al., *The FAIR Guiding Principles for scientific data management and stewardship*, Nature Scientific Data, https://doi.org/10.1038/sdata.2016.18.

[19] Shaohuai Shi, Qiang Wang, Pengfei Xu, and Xiaowen Chu. *Benchmarking state-of-the-art deep learning software tools*. CoRR, abs/1608.07249, 2016.

[20] Jurgen Schmidhuber., *Deep learning in neural networks: An overview,* CoRR, abs/1404.7828, 2014.

[21] KatrinAmunts, KarlZilles, *Architectonic Mapping of the Human Brain beyond Brodmann*, https://doi.org/10.1016/j.neuron.2015.12.001.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks*. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. *Going deeper with convolutions*. In Computer Vision and Pattern Recognition (CVPR), 2015.

[24] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann Lecun. *Overfeat: Integrated recognition, localization and detection using convolutional networks*. http://arxiv.org/abs/1312.6229, 2014.

[25] Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. CoRR, abs/1409.1556, 2014.

[26] Wittmann, Markus & Hager, Georg & Zeiser, Thomas & Wellein, Gerhard. (2013). *Asynchronous MPI for the Masses*.

[27] William Gropp and Ewing Lusk, *Fault Tolerance in MPI Programs*, Special issue of the Journal High Performance Computing Applications, 2002, Vol. 18, 363—372.

[28] B. Van Essen, R. Pearce, S. Ames, and M. Gokhale. *On the Role of NVRAM in Data-intensive Architectures: An Evaluation.* In Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International, pages 703–714, May 2012. doi: 10.1109/IPDPS.2012.69.

[29] Xiangyu Dong, Naveen Muralimanohar, Norm Jouppi, Richard Kaufmann, and Yuan Xie. *Leveraging 3D PCRAM Technologies to Reduce Checkpoint Overhead for Future Exascale Systems.* In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, pages 57:1–57:12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-744-8. doi: 10.1145/1654059.1654117. URL http://doi.acm.org/10.1145/1654059.1654117.

[30] Dushyanth Narayanan and Orion Hodson. *Whole-system persistence with non-volatile memories*. In Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012). ACM, March 2012. URL http://research. microsoft.com/apps/pubs/default.aspx?id=160853.

[31] Shen Gao, Bingsheng He, and Jianliang Xu. *Real-time in-memory checkpointing for future hybrid memory systems*. In Proceedings of the 29th ACM on International Conference on Supercomputing, ICS '15, pages 263–272, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3559-1. doi: 10.1145/2751205.2751212. URL http://doi.acm.org/10.1145/2751205.2751212.

[32] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*, 2004, OSDI'04: Sixth Symposium on Operating System Design and Implementation, 137—150.

[33] Rowstron, Ant and Narayanan, Dushyanth and Donnelly, Austin and O'Shea, Greg and Douglas, Andrew. *Nobody ever got fired for using Hadoop on a cluster*. 1st International Workshop on Hot Topics in Cloud Data Processing (HotCDP 2012), ACM, https://www.microsoft.com/en-us/research/publication/nobody-ever-got-fired-for-using-hadoop-on-a-cluster/.

[34] T. White. *Hadoop: The Definitive Guide.* O'Reilly Media, Inc., 1st edition, 2009.

[35] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. *Spark: cluster computing with working sets.* In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, volume 10, page 10.

[36] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica. *Graphx: Graph processing in a distributed dataflow framework*. In Proceedings of OSDI, pages 599–613

[37] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar. *MLlib: Machine learning in apache spark*.

[38] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia. *Spark SQL: Relational data processing in Spark*. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15, pages 1383–1394. ACM.

[39] J. Dean and S. Ghemawat. *MapReduce: Simplified data processing on large clusters.* 51(1):107–113.

[40] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing.* In Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI'12, pages 2–2. USENIX Association.

[41] Aaron Davidson, Andrew O. *Optimizing Shuffle Performance in Spark*. UC Berkeley 2013.

[42] Jingui Li, Xuelian Lin , Xiaolong Cui , Yue Ye. *Improving the Shuffle of Hadoop Map Reduce*. IEEE 5th International Conference on Cloud Computing Technology and Science.

[43] Xiaoyi Lu, Md. Wasi-ur-Rahman, Nusrat Islam, Dipti Shankar and Dhabaleswar K. (DK) Panda. *Accelerating Spark with RDMA for Big Data Processing: Early Experiences*. The Ohio State University, 2014 IEEE.

[44] S. R. Alam, H. N. El-Harake, K. Howard, N. Stringfellow, and F. Verzelloni. *Parallel I/O and the metadata wall*. In Proceedings of the sixth workshop on Parallel Data Storage, pages 13–18. ACM.

[45] P. Carns, S. Lang, R. Ross, M. Vilayannur, J. Kunkel, and T. Ludwig. *Small-file access in parallel file systems*. In IEEE International Symposium on Parallel Distributed Processing, 2009. IPDPS 2009, pages 1–11.

[46] LeCun, Yann & Bengio, Y & Hinton, Geoffrey. (2015). *Deep Learning*. Nature. 521. 436-44. 10.1038/nature14539.

[47] Huang, H. and Winter, J. M. and Osterberg, E. C. and Horton, R. M. and Beckage, B., *Total and extreme precipitation changes over the Northeastern United States*. 2017, J. Hydrometeorol., Vol. 18, 1783--1798, 10.1175/JHM-D-16-0195.1.

[48] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, Dan Mané. *Concrete Problems in AI Safety*. 2016, arXiv:1606.06565.

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. *ImageNet classification with deep convolutional neural networks*. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12), F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 1. Curran Associates Inc., USA, 1097-1105.

[50] S. Salon, G. Cossarini, G. Bolzon, A. Teruzzi, C. Solidoro. *Current state and recent trends of Mediterranean sea biogeochemistry derived by high-resolution reanalysis*. In: Rapp. Comm. int. Mer Medit., 41:214 and the Copernicus catalogue. 2016.

[51] W. J. Gutowski Jr. et al. *WCRP COordinated Regional Downscaling EXperiment (CORDEX): a diagnostic MIP for CMIP6*. In: Geoscientific Model Development 9.11 (2016), pp. 4087–4095. DOI: 10.5194/gmd-9-4087-2016. URL: http://www.geosci-model-dev.net/9/4087/2016/.

[52] H. Li, A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica. Tachyon. *Reliable, memory speed storage for cluster computing frameworks*. In Proceedings of the ACM Symposium on Cloud Computing, pages 1–15. ACM

[53] S. R. Alam, H. N. El-Harake, K. Howard, N. Stringfellow, and F. Verzelloni. *Parallel I/O and the metadata wall*. In Proceedings of the sixth workshop on Parallel Data Storage, pages 13–18. ACM.

[54] P. Carns, S. Lang, R. Ross, M. Vilayannur, J. Kunkel, and T. Ludwig. *Small-file access in parallel file systems*. In IEEE International Symposium on Parallel Distributed Processing, 2009. IPDPS 2009, pages 1–11.

[55] Y. Wang, X. Que, W. Yu, D. Goldenberg, and D. Sehgal. *Hadoop acceleration through network levitated merge*. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC ’11, pages 57:1–57:10. ACM.

[56] N. S. Islam, M. W. Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy, and D. K. Panda. *High performance RDMA-based design of hdfs over infiniband*. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC ’12, pages 35:1–35:35, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[57] X. Lu, M. Rahman, N. Islam, D. Shankar, and D. Panda. *Accelerating spark with RDMA for big data processing: Early experiences*. In 2014 IEEE 22nd Annual Symposium on High-Performance Interconnects (HOTI), pages 9–16.

[58] Gibney E. *Billion-euro quantum project takes shape*. Nature 545 (2017).

[59] Carleo G. & Troyer M. *Solving the quantum many-body problem with artificial neural networks*. Science 355, 602-606, doi:10.1126/science.aag2302 (2017)

[60] Haner, T. & Steiger, D. *0.5 Petabyte Simulation of a 45-Qubit Quantum Circuit*. arXiv:1704.01127 (2017).

[61] Mohseni, M. et al. *Commercialize early quantum technologies*. Nature 543, 171-174 (2017).

[62] Barends, R. et al. *Coherent Josephson Qubit Suitable for Scalable Quantum Integrated Circuits*. Phys Rev Lett 111, doi:ARTN 080502 10.1103/PhysRevLett.111.080502 (2013).

[63] Sabour, Frosst and Hinton. *Dynamic Routing Between Capsules*. 2017, https://arxiv.org/abs/1710.09829v2.

[64] K. He et al., arXiv e-prints (2015), arXiv:1512.03385.