Received March 13, 2021, accepted April 15, 2021, date of publication May 11, 2021, date of current version May 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3079139

Towards HPC and Big Data Analytics Convergence: Design and Experimental Evaluation of a HPDA Framework for eScience at Scale

DONATELLO ELIA[1,2], **(Member, IEEE), SANDRO FIORE**[3], **(Member, IEEE), AND GIOVANNI ALOISIO**[1,2]

[1]Euro-Mediterranean Centre on Climate Change (CMCC) Foundation, 73100 Lecce, Italy
[2]Department of Engineering for Innovation, University of Salento, 73100 Lecce, Italy
[3]Department of Information Engineering and Computer Science, University of Trento, 38123 Trento, Italy

Corresponding author: Donatello Elia (donatello.elia@cmcc.it)

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme through the Project ESiWACE2 under Grant 823988, and in part by the European Union's Horizon 2020 Research and Innovation Programme through the Project EXDCI-2 under Grant 800957.

ABSTRACT Over the last two decades, scientific discovery has increasingly been driven by the large availability of data from a multitude of sources, including high-resolution simulations, observations and instruments, as well as an enormous network of sensors and edge components. In such a dynamic and growing landscape where data continue to expand, advances in Science have become intertwined with the capacity of analysis tools to effectively handle and extract valuable information from this ocean of data. In view of the exascale era of supercomputers that is rapidly approaching, it is of the utmost importance to design novel solutions that can take full advantage of the upcoming computing infrastructures. The convergence of High Performance Computing (HPC) and data-intensive analytics is key to delivering scalable High Performance Data Analytics (HPDA) solutions for scientific and engineering applications. The aim of this paper is threefold: reviewing some of the most relevant challenges towards HPDA at scale, presenting a HPDA-enabled version of the Ophidia framework and validating the scalability of the proposed framework through an experimental performance evaluation carried out in the context of the Centre of Excellence in Simulation of Weather and Climate in Europe (ESiWACE). The experimental results show that the proposed solution is capable of scaling over several thousand cores and hundreds of cluster nodes. The proposed work is a contribution in support of scientific large-scale applications along the wider convergence path of HPC and Big Data followed by the scientific research community.

INDEX TERMS Extreme-scale data challenges, HPC and big data convergence, high performance data analytics (HPDA), performance evaluation, scientific data analysis.

I. INTRODUCTION

Over the last two decades, scientific discovery has increasingly been driven by the large availability of data from a multitude of sources, including high-resolution simulations, observations and instruments, as well as an enormous network of sensors and edge components [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Dongxiao Yu.

Thanks to the data deluge that started at the beginning of this century, data-intensive science has emerged as the fourth scientific paradigm [2], [3] paving the way towards the Big Data revolution, which broke up around 2010 and led to a new awareness of the multifaceted complexity and relevance of data. As part of this process, the term Big Data which originally referred to just a few orthogonal dimensions such as volume, velocity and variety [4], i.e. the most obvious and quantitative aspects of data, was later on further complemented and enriched with new dimensions like veracity, variability and value [5], able to capture the most qualitative and intrinsic properties of data. Big data also meant a new set of challenges that the software community had to address to ensure efficient management of such data at all levels and throughout the entire data lifecycle [6]. As a result, the Big Data revolution led through the years to the birth of an incredibly vast software ecosystem able to foster a data-centric paradigm for scientific discovery, while complementing and enriching the well-established simulation-centric paradigm that is mainly adopted by the HPC community [7].

In such a dynamic and growing landscape where the size, rate and complexity of data continue to expand, advances in Science are dependent on the actual capacity of analysis tools to effectively handle and extract valuable information from this ocean of data. By projecting these challenges a bit further, it is clear that the definition of novel approaches and strategies is now necessary to efficiently deal with extreme-scale data-centric scenarios [1]. Additionally, from an infrastructural standpoint, it is of paramount importance that scientific analysis can soon take advantage of the large computing infrastructures that are expected to come to light in the next couple of years, such as exascale machines [8]–[10].

Therefore, in this context, it becomes critical to empower scientists with *High Performance scientific Data Analytics (HPDA)* solutions capable of fully addressing scientific discovery by leveraging technological advances in HPC, as well as data management techniques and tools designed to tackle Big Data challenges. In fact, in order to fully support scientific discovery at extreme scale, High Performance Computing (HPC) and data-intensive analytics are both deemed as fundamental and complementary aspects of this process [11].

However, software development in the fields of Big Data and HPC has mainly been carried on in a disjoint way, leading to a certain divergence in terms of resulting software ecosystems [12]. In spite of that, over the last few years, the HPC and Big Data communities have increasingly been looking at each other, in order to: (i) pass the technologies and advances of one field on to the users of another field, (ii) study viable pathways of convergence between the two approaches, (iii) identify common challenges, and (iv) provide proper guidelines and recommendations, with the ultimate goal of enabling *extreme-scale* and *High Performance Data Analytics*.

In the field of scientific data analysis, the Ophidia framework[1] [13] tries to address the challenges of large-scale multi-dimensional scientific data management and analytics. The framework provides a complete environment targeting the Big Data challenges at various layers with a multi-dimensional data model, parallel and in-memory data processing, data-driven task scheduling and service-oriented interfaces. In the last couple of years, the Ophidia platform has evolved to better support HPDA applications, with the main goal of supporting large-scale data analysis in supercomputing environments by taking advantage of the solutions from both the Big Data and HPC software ecosystems. Although some key concepts of the platform still hold, the system has undergone an internal redesign to include the new requirements related to extreme-scale scenarios on supercomputing infrastructures.

In particular, this paper describes a High Performance Data Analytics framework and how it addresses some of the main challenges of large-scale scientific analysis, with a focus on the proposed parallel run-time system and its integration with HPC cluster environments. A quantitative evaluation of a set of key analytics operators is performed to assess the scalability of the proposed HPDA solution and, in particular, of its runtime system on a hundred nodes and a few thousands cores. This provided a solid reference/baseline for future evaluation and comparison with newer versions of the HPDA framework as well as existing state-of-the-art tools (which is out of the scope of this paper). More in detail, the contribution of this article is threefold, since it:

- reviews the international context and main efforts in the field of scientific HPDA and summarizes some of the key challenges at the intersection of HPC and data analytics;
- presents a HPDA-enabled version of the Ophidia framework, in particular the following design aspects: (i) the runtime system for parallel processing and (ii) the HPC-enabled deployment and scheduling schema;
- assesses the performance and validates the scalability of the proposed HPDA framework design through an experimental evaluation carried out on the MareNostrum4 PRACE (Partnership for Advanced Computing in Europe) Tier-0 HPC cluster, in the context of the *Centre of Excellence in Simulation of Weather and Climate in Europe* (ESiWACE).

The rest of the paper is organized as follows. Section II presents relevant international initiatives and related work about the convergence between HPC and Big Data Analytics (BDA). Section III introduces the main key challenges when addressing HPDA, while the proposed HPDA framework and the related key implementation aspects to support large-scale data analytics are discussed in Section IV. Section V presents an experimental evaluation of the proposed HPDA framework regarding system scalability and the discussion of results. Finally, Section VI points out the main conclusions of the paper along with some key aspects that are worth investigating in the future.

II. INTERNATIONAL CONTEXT

This section highlights some of the most important international initiatives on the convergence between HPC and BDA (Subsection II-A) that have been held over the past decade, as well as a comprehensive overview of the relevant work in this area (Subsection II-B).

[1]Ophidia Website: http://ophidia.cmcc.it/

A. INTERNATIONAL LANDSCAPE AND MAIN INITIATIVES OVER THE LAST DECADE

Initial efforts in the area of extreme-scale scientific data processing came with the International Exascale Software Project (IESP) [14]. The IESP's vision of exascale computing was a huge international effort targeting extreme parallelism, energy consumption constraints, resilience, efficiency and programmability. However, HPC challenges towards exascale computing were primarily addressed without properly considering the emerging interest in new Big Data solutions and approaches to tackle large-scale data analysis in virtualized environments.

As a response to that, the HPC community organized the Big Data and Extreme-scale Computing (BDEC) initiative,[2] (2013-2018, now in its second stage, BDEC2, 2018-2020) an international effort to address the disruptive emergence of Big Data in the scientific and engineering domains and pave the way for the convergence between HPC and High-end Data Analysis (HDA) [7]. At EU level, in close synergy with the BDEC, the EXDCI project[3] (phase1 2015-2018, phase2 2018-2020) shifted the attention to the convergence of extreme data and computing with a new focus on machine learning, Big Data and Artificial Intelligence.

As part of the same landscape, the most recent Strategic Research Agenda (SRA) of the ETP4HPC [15] project aims to outline the European research priorities in the area of HPC technology and High Performance Data Analytics (HPDA) to support the EuroHPC Joint Undertaking (EuroHPC JU)[4] in building its 2021-2024 Work Programme. On the same line, the European Commission presented the Digital Single Market (DSM) strategy [16] in 2015 and its mid-term review [17] a few years later, where Big Data and HPC are considered core components of a broader system fostering, enabling and boosting European data economy [18]. More recently (in 2020), the *European strategy for data* [19] emphasized the importance of the connection between EU data spaces and the existing computing capacities at EU level, including top class HPC infrastructures, to support data processing and computing.

Other key national initiatives in the US [20], [21], China and Japan [22] have also highlighted the strategic value of this convergence both from an economical and a scientific standpoint.

All these efforts worldwide have strongly contributed to fostering, inspiring and supporting actions on Big Data and HPC co-existence, integration and convergence, which further stresses the importance of HPDA in Science and engineering and in turn, the strong impact it can have on society.

[2]Big Data and Extreme-scale Computing (BDEC) project: https://www.exascale.org/bdec/

[3]EXDCI/EXDCI2 project: https://exdci.eu/about-exdci

[4]EuroHPC Joint Undertaking: https://eurohpc-ju.europa.eu/discover-eurohpc

B. RELATED WORK

In several large-scale scientific and engineering applications, HPC and Big Data are currently combined into application workflows that join simulations and experiments with processing, analysis and visualization pipelines [23]–[26], which leads to a unified model where *computational* and *data science* are two orthogonal components enabling scientific discovery. The convergence of the two ecosystems is therefore key for scientific research; yet, several issues must be addressed in terms of design and skills required to overcome the great technological and cultural differences [7], [27]. Among these, some crucial differences concern execution models, job scheduling and management, storage systems and hardware architectures [11], [28].

The Big Data ecosystem provides a wide set of technologies targeting every aspect of data management and analysis. One of the most well-known solutions in this context is the Apache Hadoop framework [29] and its main modules, i.e. an implementation of the MapReduce programming model [30] and the Hadoop Distributed File System (HDFS) [31]. Various other technologies have later enriched the Apache Hadoop ecosystem [32] and solutions for On-Line Analytical Processing (OLAP) workflows have also been developed, such as Apache Kylin [33] and Apache Druid [34]. Moreover, several Big Data technologies using the main memory for data management, storage and processing have also been developed [35], and Apache Spark is one of the most popular [36], [37].

Nevertheless, general-purpose technologies are not able to fully address scientific data analysis requirements [38], for example in terms of support for domain-specific formats, algorithms and metadata. To this end, scientific-oriented data management solutions addressing multi-dimensional data analysis have also been developed, including Database Management System technologies such as SciDB [39]–[41] and Rasdaman [42], [43] or processing frameworks extending Apache Spark, like the SciSpark [44], [45] and ClimateSpark [46] frameworks addressing climate sciences requirements.

Yet, both these general-purpose and scientific-oriented data analytics solutions do not target the convergence of HPC and BDA and they have not been designed for use on HPC systems, but mainly on commodity and virtualized resources. As reported in literature [47], there have been efforts in trying to port the Big Data systems to HPC infrastructures, but their scalability can be severely limited due to the differences in the two ecosystems, and therefore requires careful tuning and optimizations.

Different approaches are currently being explored for the exploitation and integration of data analytics systems with HPC solutions, to have them complement each other while addressing scientific application needs [48]–[51].

Moreover, solutions for scientific data analysis targeting the integration with HPC systems, such as the Dask framework [52], [53], have also been developed in the Python ecosystem. Dask is natively integrated with the widely used

Python data analysis solutions (e.g., NumPy [54] and Pandas [55]) and provides the parallel computing engine for domain-specific modules (e.g., Xarray [56] and Iris [57]). Another solution targeting this integration is COMP Superscalar (COMPSs) [58], a task-based programming model able to parallelize applications over HPC infrastructure, which also provides support for Python-based data analytics applications through its PyCOMPSs module [59]. Like these solutions, the proposed HPDA framework tries to tackle the convergence of HPC and BDA to support extreme-scale data analysis, though more tailored to multi-dimensional scientific data analysis in the climate domain in terms of support for: specific data formats (i.e., NetCDF [60]), conventions (i.e., Climate and Forecast [61]), metadata management with respect to community specific vocabularies (i.e., Coupled Model Intercomparison Projects) and optimized I/O for fast read/write tasks.

Additionally, with respect to traditional solutions from the Big Data ecosystem, the proposed HPDA framework strongly integrates its in-memory analytics capabilities with the underlying HPC environment in terms of parallel paradigms (i.e., Message Passing Interface [62]), parallel shared file system, batch schedulers and hardware-optimized libraries.

III. MAIN CHALLENGES FOR EXTREME-SCALE HPDA

This section reviews some key challenges when addressing HPDA at scale, with specific focus in this paper on scientific discovery workflow, programming models/paradigms, storage solutions, resource management, including software deployability and portability.

A. PARADIGM SHIFT

In several scientific domains, the commonly adopted workflow for scientific discovery has primarily been based on server-side data access and client-side data analysis, often ending up downloading large amounts of data on the user's desktop [63]. Scientists, for example, have mainly been relying on client-side and sequential tools to deal with data analysis needs in the climate domain, like Climate Data Operators (CDO) [64], netCDF Operators (NCO) [65] and NCAR Command Language (NCL) [66], which were not designed to meet large-scale scenarios. Such tools, although being optimized for domain-specific analysis, generally provide limited amounts of parallelism, i.e. multithreading, but no direct support for scaling the processing outside the single node, and they might fail for the lack of hardware resources (usually main memory) on the client node.

This kind of workflow is simply not feasible at extreme-scale, since:

- ever-larger scientific datasets are being generated by experiments and simulations, thus requiring bigger computing facilities for data analysis as opposed to traditional desktop machines;
- the increased scale in the Big Data Vs requires novel approaches and tools featuring a high level of parallelism to efficiently perform arbitrarily complex data analysis;
- data download is becoming increasingly time- and resource-consuming, in terms of both network and storage, and thus represents an insurmountable barrier for even setting up large-scale data analysis experiments on desktop machines.

The first two points respectively relate to *capacity* and *capability* dimensions; the third one naturally refers to capacity but also more inherently to the *democratization of data analysis*, which is rapidly emerging as a key challenge in Open Science [67].

In light of the three points above, extreme-scale HPDA requires a paradigm shift based on server-side data analysis, HPC facilities, data and compute co-location as well as novel scalable and programmable HPDA tools/frameworks. Such approach greatly contributes to reducing the downloaded data and the makespan for the analysis task, but also the complexity related to software administration and computing environment, that are no longer managed on the client side but centralized on the HPC facility.

It is worth mentioning here that such paradigm shift also calls for a cultural and methodological change for scientists towards (i) new software ecosystems, (ii) inherently complex and more advanced computing environments, (iii) interdisciplinary teams and (iv) more collaborative ways of doing Science.

B. BIG DATA ANALYTICS AND HPC PARADIGMS

High Performance Computing infrastructures and data analytics ecosystems (mainly cloud-based) represent two completely different target environments to run HPDA software [11]. In the HPC case, tightly coupled approaches mostly relying on Message Passing Interface (MPI) [62] and OpenMP [68] represent viable solutions on HPC architectures, possibly together with accelerators, such as Graphics Processing Units (GPUs) or Field Programmable Gate Arrays (FPGAs). In the case of data analytics ecosystems, loosely coupled approaches, mainly based on MapReduce-like paradigms and well-known Big Data software stacks (e.g., Apache Big Data Stack [69]), can support the analysis and processing of large datasets in cloud environments. HPDA at extreme-scale could foster the development and adoption of new programming frameworks and models along the HPC and Big Data convergence pathway [7], including new application programming interfaces (APIs) and paradigms [11], [70]–[73] capable of dealing with massive parallelism through intuitive and interoperable, high-level interfaces.

HPDA-enabled runtime systems are required to support these new programming models, as they can handle large-scale analytics workloads and harness the computing power of supercomputing infrastructures for parallel processing [73].

C. STORAGE AND DATA MANAGEMENT CHALLENGES

As data grow in size and complexity, efficient storage models and data management strategies for fast I/O are essential to perform HPDA at extreme-scale. As mentioned in Subsection III-B, HPC and Big Data ecosystems are typically organized in a very different fashion. HPC clusters exploit parallel shared file systems (e.g., GPFS, Lustre) while Big Data frameworks usually rely on higher-level abstractions, loosely coupled and distributed file systems to provide high throughput access to applications (e.g., HDFS) [28]. From this point of view, different efforts concerning the definition of innovative storage architectures towards ecosystems convergence can be found in literature [74], [75].

D. DATA ANALYTICS-ORIENTED RESOURCE MANAGEMENT IN HPC

In HPC, job scheduling is handled by a distributed resource management system (DRMS), such as SLURM [76], LSF [77] and PBS [78]. These systems are highly optimized for long-running, massively parallel (mainly MPI-based) batch applications and do not support the proper flexibility required by data analytics workloads [79]. Data-driven workflows can in fact be composed of multiple loosely-coupled tasks, whereas HPC applications are much more tightly-coupled and have fixed resource requirements throughout their execution; in this case, task-level scheduling can improve the general application execution [28].

To this end, resource management solutions should be extended to also include data analytics-based workloads requirements, or else specific systems could be devised to add support for data-aware job scheduling on top of the existing batch schedulers [80], [81].

E. PORTABILITY AND DEPLOYABILITY OF THE SOFTWARE SOLUTIONS

The increasing size, complexity and heterogeneity of extreme-scale computing infrastructures require software solutions able to both scale on the cluster size and exploit different hardware technologies, along with the available computing devices (e.g., CPUs, GPUs, FPGAs, etc.). To this end, deployability and software portability are two key aspects to make sure a software solution can effectively be used on different supercomputing platforms.

Specific HPC-oriented containerization technologies are then good candidates for supporting the portability and deployability of software stacks on different HPC clusters with no major impact on performance [82], although there are still some issues and limitations to be addressed [83]. Software containers can be thought of as OS-level virtualization - in contrast with hypervisors that provide hardware-level virtualization - and have become very popular in the last few years as they provide a very convenient way to package applications and the related dependencies, with Docker containers [84] as the most widely adopted solution. However, the deployment of Docker containers in HPC environments can pose some technical and security challenges; in this respect, HPC-oriented containerization technologies, such as Singularity [85], Shifter [86] or Sarus [87], have been proposed over the last years to improve support for scientific applications needs in HPC.

IV. THE OPHIDIA HPDA FRAMEWORK

The Ophidia framework represents an open source solution[5] for scientific data analytics, joining HPC paradigms and Big Data approaches. The framework has primarily been used in the climate change domain, though it has also successfully been exploited in other domains/contexts (e.g., astronomy, seismology [88], smart cities [89]). Ophidia addresses scientific data analysis on large multi-dimensional datasets, leveraging the datacube abstraction inherited from the OLAP data warehouse systems. In the last few years, inspired by the challenges described in Section III, the framework has undergone an internal redesign with respect to the initial implementation, towards improved support of HPDA use cases. This design activity has focused on (i) a scalable runtime (Subsection IV-B) and on (ii) the integration with the HPC software ecosystem at the level of deployment and job scheduling (Subsection IV-C). Nevertheless, some core concepts have been inherited from the previous design without any change (Subsection IV-A), such as the internal storage model and the multi-layered architecture.

A. HPDA FRAMEWORK ARCHITECTURE

The latest design of the Ophidia architecture has comprehensively been presented in [90]. For the sake of clarity, a brief and high-level description of such four-layer architecture is summarized as follows, with specific focus on the aspects relevant to this paper. Figure 1 shows an updated view of the architecture.

The front-end layer is represented by the Ophidia Server [91], which **enables server-side computation by addressing the paradigm shift challenge described in Subsection III-A**. The Ophidia server provides an interoperable front-end to the rest of the Ophidia software stack deployed on the HPC infrastructure, that users can remotely connect to either via a remote lightweight client or a Python API,[6] thus achieving a *separation of concerns* between the client and the server.

The new runtime layer, described in the next section, is responsible for running all the provided analytics operators. Such architectural level provides datacube abstraction to the end-user, thus completely concealing all low-level details and the complexity related to data partitioning and distribution. Datacube operators represent high-level OLAP functionalities and include support for both (i) metadata management (sequential), and (ii) multi-dimensional (parallel) data analytics comprising domain-oriented operators, such as import and

[5]Ophidia Framework source code: https://github.com/OphidiaBigData
[6]PyOphidia: https://github.com/OphidiaBigData/PyOphidia

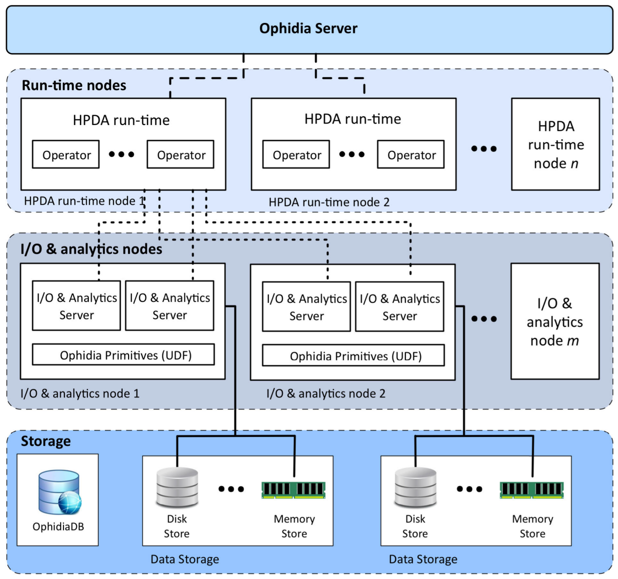

FIGURE 1. High-level architecture of the Ophidia HPDA framework, updated from [90].

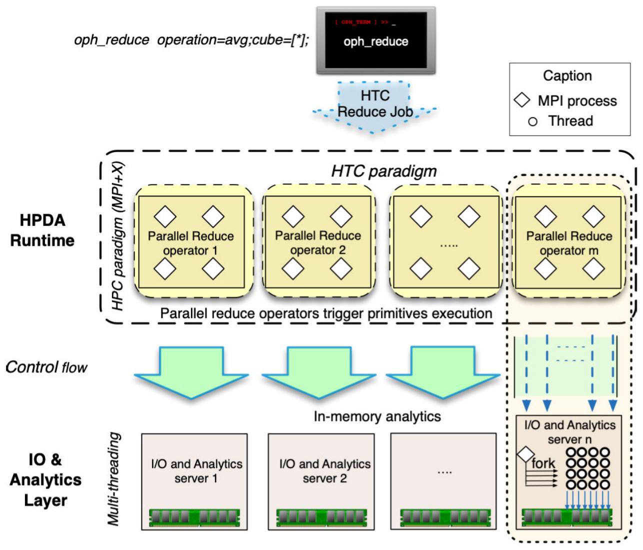
oph_reduce operation=avg;cube=[*];
oph_reduce
HTC Reduce Job
Caption
MPI process
Thread
HTC paradigm
HPDA Runtime
HPC paradigm (MPI+X)
Parallel Reduce operator 1
Parallel Reduce operator 2
Parallel Reduce operator m
Parallel reduce operators trigger primitives execution
Control flow
In-memory analytics
IO & Analytics Layer
Multi-threading
I/O and Analytics server 1
I/O and Analytics server 2
I/O and Analytics server n
fork

FIGURE 2. High-level representation of the proposed HPDA runtime system for a data reduction job (HTC reduce command). Multiple independent instances of the same operator are executed concurrently on different datacubes. Each single operator in the bag of tasks is implemented as an MPI+X parallel application executed at the level of the HPDA runtime.

export from specific formats, and domain-agnostic operators, like subsets and data aggregations.

The I/O & analytics layer hosts in-memory multi-threaded servers fully devoted to data transformation tasks. Ophidia implements a native in-memory service [90], with a query-based interface that supports the management of n-dimensional array structures in terms of both data type and functionality. This support is provided by the so-called Ophidia primitives, i.e. low-level libraries implementing specific User Defined Functions (UDF) to manipulate binary-arrays. Nesting of primitives (which will be part of the benchmark in Section V) is also supported by the I/O & Analytics servers to allow for more complex array-based transformations.

The OphidiaDB represents the Ophidia system catalog, which stores information onto the storage system about the system configuration and status, including the list of I/O & Analytics servers, the job status and history, and the physical mapping of datacube fragments.

The Ophidia storage model is designed to manage multi-dimensional data in binary arrays. The model leverages the OLAP-based *datacube* abstraction to handle multi-dimensional datasets. Each datacube is flattened to a table-like data structure and horizontally partitioned into several *fragments* (i.e., chunks), which are distributed across the available I/O & analytics nodes to enable parallel data analysis. Each fragment is composed of a set of multi-dimensional binary arrays following a data store implementation based on a NoSQL approach. A more detailed and rigorous description of the storage model is provided in [92].

In terms of data management, the new design of the HPDA framework combines approaches from the HPC and Big Data ecosystems, by building a shared-nothing storage layer for fast data access and computing and relying on the HPC-based shared layer for data I/O. **This data management architecture aims to address the challenge presented in Subsection III-C**.

The design is implemented through the integration of (i) the I/O & Analytics server, which allows performing distributed in-memory analytics by means of data locality on the node and data re-use across subsequent operations, (ii) the OLAP-based storage model leveraging an array-oriented data structure for efficient multi-dimensional data management, and (iii) the parallel file system at the storage level to address parallel I/O on the distributed set of fragments at the initial and final stages of an analytics experiment.

B. HPDA RUNTIME SYSTEM

To address HPDA challenges, Ophidia has evolved over the last few years towards a multi-level parallel execution model. The resulting new runtime system joins together the data-driven computing paradigm and the HPC-based parallel computing one. While the former supports *High Throughput Computing* (HTC) to simultaneously act on multiple datacubes, the latter exploits *High Performance Computing* (HPC) to efficiently address parallel execution at the level of a single datacube. **The runtime system targets the challenge related to the convergence of data analytics and HPC techniques to provide a scalable analytics solution (Subsection III-B)**. The runtime exploits a multi-processing (i.e., MPI) and multi-threaded (i.e., POSIX Threads) approach for the execution of parallel operators, moving computation to the I/O & analytics nodes where all data fragments are actually stored. Figure 2 provides a high-level representation of the proposed runtime system supporting the multi-level parallel execution model.

As shown in the upper part of Figure 2, the first level of parallelism (defined as *datacube-level parallelism*) implements an HTC paradigm by supporting the execution of multiple independent instances of the same operator on different input data, that is a parameter sweep approach applied to multiple files/datacubes. From a usability perspective, it allows end-users to submit an arbitrarily very high number of independent analytics tasks through a single declarative HTC statement based on user-defined filters to narrow down the set of inputs/tasks to the desired extent.

Additionally, the runtime supports a second level of parallelism defined as *fragment-level parallelism* by exploiting a MPI+X model applied to the single analytics operators. In this case, each operator is implemented as a hybrid MPI+Pthread application able to analyze multiple fragments of the same datacube in parallel, by scaling the analytics over multiple cores and nodes, according to the available resources. In such a context, the parallel operator acts as the coordinator of the whole analytics process by triggering real data crunching on the I/O & Analytics server side.

Figure 2 shows a concrete example of how the developed multi-level parallelism approach works in the case of a very common datacube reduction operation (*oph_reduce* operator, average operation) applied over the time dimension on multiple inputs.

The associated HTC reduce command is:

```
oph_reduce operation = avg;cube = [*];
```

In this case, the operator *oph_reduce* computes a temporal average (*avg*) on the time series of each input datacube (please note that *cube = [*]* means the input is *"all the cubes in the current directory"*) generating a new set of output datacubes. The HTC reduce command is automatically *expanded* by the front-end server into multiple data reduce operators based on the input *cube* argument, thus implementing the first level of parallelism of the proposed HPDA framework; then, each single operator is executed as an MPI+Pthread job, which corresponds to the implementation of the second level.

As it can easily be argued from the above example, the framework exposes a declarative interface for the execution of the analytics operators hiding most of the complexity of the architecture and the underlying computing infrastructure. For instance, in addition to our example, users might also customize the parallel execution of an operator by defining the number of threads and MPI processes to be used as well as the data fragmentation/distribution parameters or data layout during the import or subsequent data manipulation stages. This gives scientists a very high degree of flexibility through the use of the Command Line Interface (CLI), as well as programmability to developers from an API standpoint.

Delving into the details of this second level of parallelism, each MPI task associated with a parallel operator manages a specific subset of the datacube fragments and it spawns multiple concurrent threads to parallelize the management of such fragments. MPI is mainly used for inter-process (and inter-node) synchronization and result exchange, whereas the multi-thread support allows for a better exploitation of intra-node parallelism. POSIX threads have been chosen in place of other solutions, like OpenMP, since they provide a lower-level API, thus enabling a finer-grained control over the tasks in terms of shared structures, locking mechanisms and management of non-thread-safe functions. It is worth mentioning that each MPI task does not perform any direct computation, but rather it:

1) sets up the execution plan according to data distribution;
2) serves as coordinator to dispatch processing, in terms of queries, to the proper I/O & Analytics servers;
3) monitors the query execution performed by the I/O & Analytics server on each fragment;
4) performs the synchronization stages of the operator;
5) updates the relevant metadata information stored in the system catalog (OphidiaDB).

The actual computation is performed on the I/O & Analytics servers, which are directly connected to the storage in order to take advantage of data locality. From this perspective, the MPI task can be considered as an I/O bound application coordinating the whole operator process and spending most of its time waiting for query execution on the I/O & Analytics servers.

In such a scenario, the hybrid approach allows users to take full advantage of resources because several queries and connections to the I/O & Analytics servers can be handled concurrently with multiple threads within the same process, allocating hundreds of requests per physical core with very limited overhead. On the other hand, the use of MPI allows scaling up the operator over multiple nodes, thus potentially enabling the management of large datasets partitioned into hundreds of thousands of fragments.

Figure 3 shows a detailed view of the fragment-level parallelism just presented; as it can be seen in the right-side part of the figure, each POSIX thread running within the runtime system submits a query to the I/O & Analytics servers for the execution of a data analytics kernel (i.e., a primitive) on the binary arrays of a given fragment.

The following listing (Figure 4) shows a simplified pseudo-code of the template followed by the runtime for the implementation of each data parallel operator. To reduce the impact on the central components, information retrieval from the system catalog (i.e., the OphidiaDB) is performed by the master MPI process only (lines 3-5), which in turn broadcasts this information to all other MPI processes (line 7). Synchronization among the MPI processes is limited to a few blocks of code to reduce locking points. Indeed, the single process runs independently until reaching a final synchronization stage used for error checking and consolidation of metadata into the system catalog. Based on the information provided by the master, each process (i) identifies the list of fragments it is responsible for (line 8-9), (ii) spawns multiple

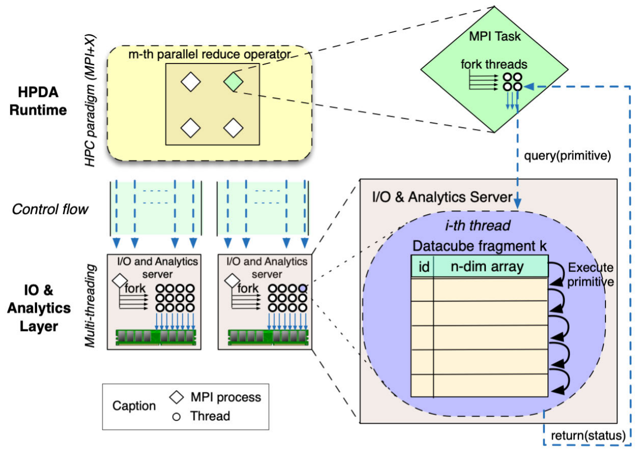

FIGURE 3. Schematization of the HPDA framework internals for the execution of a data analytics operator. The operator is implemented as an MPI+X application executed at the level of the HPDA runtime, which coordinates the parallel data analysis carried out at the level of the I/O & Analytics servers. The interaction between the HPDA runtime and the I/O & Analytics servers at the threads level is highlighted on the right-hand part of the figure.

threads to concurrently submit the query to the associated set of I/O & Analytics servers (lines 10-17) and (iii) joins the threads at the end (line 18). The master finalizes the operator by storing the output metadata in the system catalog.

It is worth mentioning that fragments are assigned to each MPI process and related threads according to a spatial data locality policy that aims to minimize the number of I/O & Analytics servers contacted by each single thread, thus reducing the overall connection overhead.

```
Input: operator name, my rank, n threads, input cube(s),
    argument list
Output: output cube
 1: initialize MPI processes
 2: if my rank == 0 then
 3:     read input cube(s) info from OphidiaDB
 4:     write output cube struct in OphidiaDB
 5:     define common query info
 6: end if
 7: MPI_Bcast(common info)
 8: compute fragment list assigned to my rank
 9: get input cube fragments location from OphidiaDB
10: fork n threads
11:     compute output cube fragments location
12:     for servers assigned to i-th thread do
13:         connect to j-th server
14:         for fragments in j-th servers do
15:             submit query on k-th fragment
16:         end for
17:     end for
18: join threads
19: update OphidiaDB with output cube fragments info
20: if my rank == 0 then
21:     save output cube metadata in OphidiaDB
22: end if
23: finalize MPI process
```

FIGURE 4. Pseudo-code related to the runtime system execution of the parallel operators.

C. HPC-ORIENTED DYNAMIC DEPLOYMENT

The framework architecture described in Subsection IV-A is designed to be flexible and modular enough to support various types of deployments covering both cloud and HPC environments.

In particular, in the current design, the HPDA framework provides full support for *on-demand dynamic deployment of I/O & Analytics servers* on different HPC infrastructures. In this respect, the framework is currently exploited by climate scientists at CMCC on the Zeus HPC cluster with 1.2 PetaFLOPS (10^{15} FLoating point Operations Per Second) of peak performance, running at the CMCC SuperComputing Center.[7] The framework has also been tested on larger European supercomputing infrastructures for benchmarking purposes, such as the MareNostrum4 supercomputer at the Barcelona Supercomputing Center (BSC) (see Section V).

According to its deployment schema, an Ophidia instance requires the instantiation of a front-end server, a HPDA runtime system and a set of I/O & Analytics servers. The deployment mechanism transparently interacts at run-time with the underlying batch scheduling system in order to acquire the necessary resources for the various components.

[7]CMCC SuperComputing Center: https://www.cmcc.it/it/super-computing-center-scc

To support compatibility with different scheduling systems, the feature has been implemented with a set of configurable submission and deployment recipes (i.e., bash scripts), which control the startup, status and shutdown of the framework components. So far, the deployment has been successfully tested on Slurm and LSF, two of the most widely used batch scheduling systems in HPC. **The deployment mechanism described above allows tackling the challenges related to framework deployability (Subsection III-E)**.

A user-friendly interface is provided by the front-end server to manage the deployment. For instance, the following command can be submitted to instantiate a new cluster with 10 in-memory I/O & Analytics servers:

```
oph_cluster action = deploy;nhost = 10;
                         name = clusterA;
```

Whereas the undeployment can easily be triggered by running the following command:

```
oph_cluster action = undeploy;
                         name = clusterA;
```

During a typical analysis experiment, the user can run the *deploy* command at the beginning of the session and the *undeploy* one at the end. It is noteworthy that the front-end

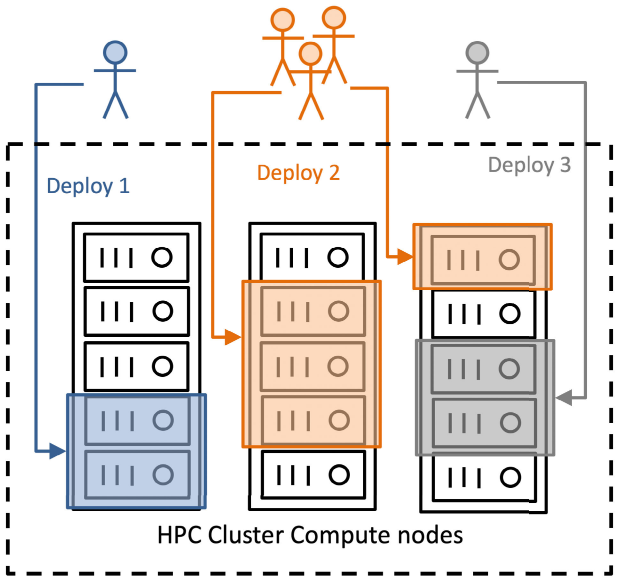

FIGURE 5. Example of a multi-tenant deployment scenario: multiple instances of the HPDA framework software stack can be deployed by different users/teams on the same HPC cluster and managed independently at the same time for different experiments.

server supports multi-tenancy, which enables users (either individuals or teams) to deploy multiple I/O & analytics cluster instances on top of the same HPC infrastructure and to interact concurrently and independently with all of them in an isolated manner, as shown in Figure 5.

Concerning resource management and job scheduling, as stated in the challenges section, HPC job schedulers are not suited for running fine-grained tasks typical of a data analytics workload, but they are mainly suited for running monolithic parallel (MPI-based) applications, such as model simulations. To improve support for data-driven analytics workloads consisting of a myriad of small tasks (*analytics operators*), the proposed framework defines a loosely-coupled, two-level scheduling mechanism by (i) leveraging the HPC job scheduler as a resource manager for cluster-aware job submission and resource allocation (as just described) and by (ii) providing a task-based scheduler to dispatch the analytics tasks on the previously allocated framework resources.

This approach overcomes the job start delay that would be introduced by the batch scheduler in a classic HPC scenario. Indeed, in supercomputers, schedulers place every single job request in a job queue waiting for available resources. A scheduling latency is hence introduced by the scheduler to identify, select and assign the best possible allocation based on the job requirements and resource availability. In the proposed approach instead, besides the initial resource allocation, the system will rely exclusively on the second level (task-based) scheduler for the execution of the applications/workflows.

This approach allows easy integration and seamless execution of data-driven analytics workflows over HPC infrastructures, addressing the challenge reported in Subsection III-D.

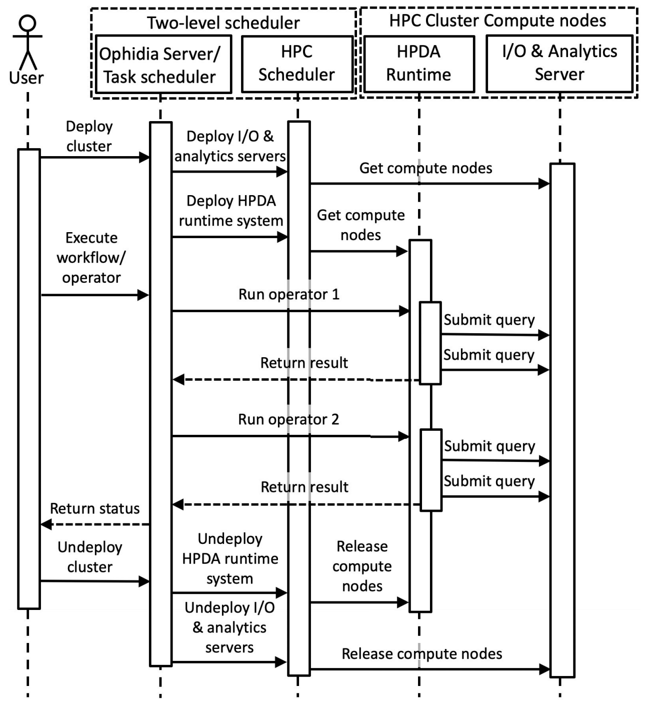

FIGURE 6. Sequence diagram of the deployment and task scheduling over a HPC infrastructure. A two-level scheduling approach allows for handling the interactions with the HPC scheduler (to allocate framework resources) and the framework scheduler (to manage analytics tasks) during a data analysis session.

An additional step forward in this solution implies the definition of HPC-specific resource usage policies to more efficiently support data analytics jobs requirements, thus improving the initial allocation step. These policies could successfully integrate the type of usage here proposed as part of the regular workload of HPC machines. The definition of such policies is beyond the scope of this work and will be taken into consideration in future work.

The overall deployment and operators execution flow is shown in the sequence diagram in Figure 6. As depicted in the figure, both the runtime system and the I/O & Analytics servers are initially instantiated (for the time required to run the analytics session) by the user via the front-end server using the deployment mechanism previously presented. After the initial deployment of the components, the user can run an entire analytics session submitting multiple operators to the runtime system on the computing resources previously allocated. As it can be seen in the figure, the task-based scheduler manages the submission of jobs to the HPDA runtime system while completely bypassing the HPC batch scheduler.

V. EXPERIMENTAL EVALUATION AND RESULTS

To analyze the scalability and performance of the proposed HPDA framework, an experimental evaluation has been conducted over a large-scale HPC cluster. A grant of 90k core-hours allocation provided by PRACE[8] through

[8]Partnership for Advanced Computing in Europe (PRACE): https://prace-ri.eu

the *PRACE HPC (Call 18) resources for Centres of Excellence in Computing Applications (CoE)* in the context of the ESiWACE project,[9] has been used to run a comprehensive set of tests.

More specifically, the proposed tests focus on the evaluation of the strong and weak scalability of the provided HPDA runtime system with respect to a set of real-world analytics operations (e.g., statistics over time series, climate indices, subsetting, regression, etc.). The experimental evaluations aim to assess:

- to what extent the proposed HPDA in-memory parallel runtime scales both in the strong and weak sense;
- the behaviour of the system with different array lengths;
- how far the use of *multi-processing* versus *multi-threading* approaches in the runtime system affects the performance of data analytics operators.

The undertaken tests focus on the execution of single operators in order to get a better understanding of the runtime system behaviour at the level of intra-task execution (i.e., the fragment-level parallelism). The benchmark moves beyond the scalability limits of a few hundreds of cores that have already been assessed in previous work [90], [92] with former versions of the framework. Moreover, initial experiments targeting inter-task behaviour at the level of the workflow (already performed on former versions of the framework as reported in [93]) will be carried out in future work after the full characterization of the framework scalability at the level of single operator.

The results of this benchmarking activity will be used to identify potential performance bottlenecks to address for enhanced scalability. Additionally, they will represent a solid baseline for comparison with (i) future versions of the proposed HPDA framework as well as (ii) existing state-of-the-art tools in the same research area.

The following subsection describes the test environment and deployment used for measuring the results (Subsection V-A), the experiments design (Subsection V-B), the workload used (Subsection V-C), the methodology followed to collect results (Subsection V-D), the experimental results (Subsection V-E) and a final discussion (Subsection V-F).

A. TESTING ENVIRONMENT

The resources granted through the aforementioned PRACE application have been allocated on the *MareNostrum4* supercomputer of the *Barcelona Supercomputing Center (BSC)*,[10] which is one of the 7 Tier-0 HPC systems available in Europe. MareNostrum4 is the fourth generation of the BSC most powerful supercomputer consisting of multiple blocks. The general-purpose block used for this activity, consists of 3 456 nodes with a total of 165 888 processors and 390 TB of memory and has a peak performance of 11.15 PetaFlops. Each compute node used in the tests is equipped with 2 Intel Xeon Platinum 8160 (24 cores) at 2.1 GHz processors and

[9]ESiWACE project: https://www.esiwace.eu
[10]MareNostrum at BSC https://www.bsc.es/marenostrum/marenostrum

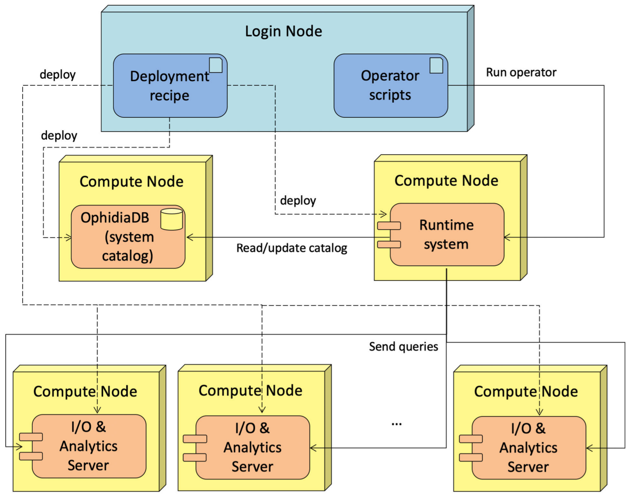

FIGURE 7. Deployment of the HPDA framework components used for the experimental evaluation. The deployment consists of: the system catalog, the runtime system and one or more I/O & Analytics servers. Each component is deployed on a separate cluster compute node (yellow boxes). Deployment recipes and the scripts for operators used in the test are submitted to the batch scheduler from the cluster login node (blue box).

96 GB or RAM memory (12 × 8 GB DDR4-2667 DIMMS), whereas the operating system is SUSE Linux Enterprise Server 12 SP2.[11]

In order to perform the experiments, the proposed HPDA framework (version 1.6.0), has been deployed at BSC using a *stand-alone cluster configuration* that only uses the architectural components responsible for computing. Such testing setup allows for focusing on the single analytics operator tests, while still retaining the same performance of the complete setup. The deployment diagram in Figure 7 depicts the setup used on the MareNostrum infrastructure. All the needed components are deployed on the cluster compute nodes before starting the tests. This deployment consists of a single *OphidiaDB* and *runtime* instance and one or more *I/O & Analytics servers*, according to the test being executed; each component is running on a separate compute node. Data are partitioned on multiple fragments that are evenly distributed among the available I/O & Analytics servers.

The deployment has dynamically been adapted to the different tests by increasing and reducing the number of I/O & analytics nodes, as described in Section IV-C, through a simple deployment recipe in the form of a Slurm submission script. Once the deployment was ready, the actual experiment tests were executed by means of other scripts submitted from the login node of the MareNostrum cluster. To avoid side-effects related to resource contention from other processes on the node, all compute nodes have been allocated in an exclusive fashion. As a technical note, the framework software stack and dependencies have been compiled using

[11]MareNostrum technical information: https://www.bsc.es/marenostrum/marenostrum/technical-information

the Intel *ICC compiler (intel/2017.4)* with the *-xHost* flag, which enables all possible optimizations available on the target platform [94].

B. EXPERIMENTS DESIGN

"Although there are diverse types of scientific big data systems, few benchmarks are available at present" [95]. In addition, none of them seem to properly address the climate-oriented features supported by the proposed HPDA framework, hence a proto-benchmark has been defined for the experimental evaluation. To this end, different key operations have been selected to evaluate the runtime execution in various relevant tests scenarios.

As the HPDA runtime system provides a wide set of analytical operators and array-based primitives that, in turn, can also be nested to perform more complex analyses, it was decided to focus only on some of the platform's most used analytics functionalities. The final selected operations are representative of real-world scientific use cases in the climate domain and cover a significant spectrum of processing types. In most cases, various primitives have been combined (i.e., nested, see Section IV-A) to be applied as a more complex atomic operation.

The selected operations can be briefly described as follows (the label is just a reference to the operations in the following sections):

- REGRESSION: it computes the slope of the regression trend line and it is based on the gsl_fit_linear function from the GNU Scientific Library (GSL) [96]. This operation is particularly interesting because it is the only one within the test suite that involves some kind of data distribution from the runtime to the I/O & Analytics servers;
- SUMMER DAYS: it computes the number of days when the average temperature is above a given threshold (25 °C) on a yearly basis. It is based on the climate index with the same name [97];
- SUBSET: it computes the average, standard deviation, minimum and maximum values from a subset of the original time series; the subset limit varies according to the time series length;
- DTR: it computes a set of 14 statistics (average, variance, skew, kurtosis, quartiles, etc.)[12] on the whole time series of daily temperature ranges (DTR), i.e. the difference between the daily maximum and minimum temperatures;
- T90P: it computes the number of days above the 90th percentile (evaluated on the whole time series) on a yearly basis. It is loosely based on the climate index TN90p [97].

All the selected operators have been tested under the same scenarios in order to evaluate the aforementioned key points.

[12]See the Ophidia documentation for the full list of statistics computed: http://ophidia.cmcc.it/documentation/users/primitives/OPH_GSL_STATS.html

Subsection V-D describes the various test scenarios executed and the results measured.

C. INPUT DATA

To evaluate the performance of the proposed framework with real climate data, a dataset from the Coupled Model Intercomparison Project Phase 6 (CMIP6) [98], [99] in NetCDF-CF format [60], [61] has been used as input for the different operators; in particular, the following historical dataset produced at CMCC with the CMCC-CM2-VHR4 model named [100]:

$$CMIP6.HighResMIP.CMCC.CMCC-CM2-VHR4.\\ hist-1950.r1i1p1f1.6hrPlevPt.ta.gn$$

The dataset is distributed under an Open Data license and can be freely downloaded through the Earth System Grid Federation (ESGF) [101] CMIP6 data portal.[13] The dataset includes a single 4-dimensional (i.e., $time \times plev \times lat \times lon$) floating point variable called *ta* (i.e, air temperature):

- each variable point has a size of 4 bytes;
- the spatial grid is at 1/4 degrees (i.e., $1152\ lon \times 768\ lat$), which is among the highest resolutions currently available;
- the number of pressure levels (*plev*) is 7;
- the time dimensions span from January 1950 to December 2014, with a time frequency set at 6-hours (i.e., 4 time steps per day). A single year (1950) was used for the tests though.

Concerning the time dimension, the dataset is split into multiple files, each one containing the time steps for a single month (i.e., about 120 time steps).

The size of each (compressed) file of the dataset is roughly around 1.5 GiB. Before executing the tests, all files were preliminarily merged into a single NetCDF file with *ncrcat* NCO operator and then rotated with *ncpdq* NCO operator [102] (with the variable re-arranged as $lat \times lon \times plev \times time$) to reduce the import time.

It is important to mention that, since the system requires loading data for each batch of tests, data were randomly generated for all the tests running on more than one I/O & Analytics server to reduce the time required for data import operations, which is out of the scope of this benchmark. This was done by following the same structure of the CMIP6 dataset just presented, by simply extending the initial time series length (i.e., 1460) to multiple years. In particular, the synthetic data were generated using a *first order auto-regressive model* (AR(1)) with a configurable time step which allows for preserving the performance behaviour of a real dataset (see the strong scalability on single node test in Subsection V-E1); moreover the AR(1) model enables the production of sufficiently consistent temperature values [103].

Table 1 summarizes the data used in the various tests.

[13]ESGF data node search interface for CMIP6 at LLNL: https://esgf-node.llnl.gov/search/cmip6/

TABLE 1. List of data used for each test run in the benchmarks.

TEST NAME	NOTES ABOUT THE INPUT DATASET
TEST1a	The dataset is a portion of a CMIP6 dataset produced at CMCC (historical experiment); it refers to 1950 only (time length 1460). The monthly files were merged prior to running the test.
TEST1b	The synthetic dataset was generated using the AR(1) model and following the same structure of the one in TEST1a, but with double time length (2920 time steps, 2 years).
TEST2	The synthetic dataset was generated using the AR(1) model and following the same structure of the one in TEST1b, with the time dimension set to 96 years (140160 time steps).
TEST3a/b	The synthetic dataset was generated using the AR(1) model and following the same structure of the one in TEST1b, and only the time dimension was extended according to the scale of the test, e.g.: 1 node = 2920 time steps, 2 node = 5840 time steps, etc.
TEST4	The synthetic dataset was generated using the AR(1) model and following the same structure of the one in TEST1a, with the time dimension set to 32 years (46720 time steps).

D. METHODOLOGY

The entire set of planned tests can be summarized as follows:

1) strong scalability on a single node;
2) strong scalability on multiple nodes;
3) weak scalability on multiple nodes;
4) comparison of multi-threaded versus mixed parallelism approach.

The results of the strong scalability tests on a single node represent the *baseline for comparison* with the other two scalability tests executed over multiple nodes.

All the operators defined in Subsection V-B were run five times for each test and the average execution time in seconds was reported. Additionally, before starting to record the execution times, warm up runs were performed for each deployment of the I/O & Analytics servers to overcome the initial transitory phase. All datasets were loaded or generated as a single datacube and evenly distributed across the I/O & Analytics servers deployed on the cluster before the execution of the tests, so that each I/O & Analytics server would manage the same amount of data and fragments.

The processing rate, also called throughput, was computed starting from the average execution time, with (1).

$$R(size, processes) = \frac{size}{T(size, processes)} \quad (1)$$

Such metric represents a proxy for the speedup and it is easy to derive, with the same formulation, for both weak and strong scalability scenarios [104]. It is measured in GiB/s as the ratio between the data size involved in the execution and the wall-clock time on the given data size with the number of processing elements.

Besides the throughput absolute value, the throughput percentage rate with respect to the theoretical ideal value (i.e., linear scalability) was evaluated in the results for a better understanding of the scalability level that was actually achieved. The theoretical ideal value is computed as the baseline value multiplied by the number of processing units, which, according to the specific test, correspond to either the involved number of cores or nodes.

TABLE 2. Results (average execution time in seconds) from the *TEST1a*.

Threads	1	2	4	6	12	24	48
Regression	240.42	122.25	61.76	42.1	21.45	11.57	5.77
Summer days	205.86	104.17	53.84	35.42	18.42	9.47	5.01
Subset	128.05	66.99	34.38	23.45	11.29	5.85	3.0
DTR	802.2	407.47	206.16	138.16	69.97	36.37	19.55
T90P	506.8	254.86	129.86	87.96	44.82	22.14	13.04

E. RESULTS

The following subsections report the experimental results in terms of average execution time and throughput, according to the test plan summarized in Subsection V-D. All plots were generated using the Python Matplotlib library [105].

1) STRONG SCALABILITY ON SINGLE NODE

This first set of tests *aims to evaluate the scalability of the selected operators over a fixed data size while increasing the number of threads* used for the execution from 1 to 48; i.e. the maximum number of cores available on a *single node*.

The number of threads was (almost) doubled on every run following the sequence: *1 - 2 - 4 - 6 - 12 - 24 - 48*. This sequence was chosen in order to keep the number of threads constantly even and balanced on the two node sockets.

Two different input datasets were used for this set of tests (see Table 1 for details):

- single dataset of about 33.7 GiB of size (simulation output from a CMIP6 experiment);
- single dataset of about 67.4 GiB of size (synthetic data generated with time series doubled in size with respect to the previous case).

Considering these two different datasets was useful to understand the behaviour of the runtime system with (i) real and synthetic data, (ii) different data sizes per node and (iii) different array length.

In both cases, data were loaded on a single I/O & Analytics server. For the sake of readability, these tests are referred to as TEST1a (with 33.7 GiB dataset) and TEST1b (with 67.4 GiB dataset) respectively.

Table 2 shows only the average execution time related to the set of configurations considered in the first case based on the real CMIP6 dataset. The execution times measured for TEST1b are almost double with respect to TEST1a measurements. According to their different types and complexity, the execution times of the various operators are quite different, with DTR being the slowest, as it needs to compute 14 different statistics, and SUBSET being the fastest, as it applies computation only to one fourth of input data.

The scalability of operators on a single node is close to linear in (almost) all cases. As it can be seen in the log-log

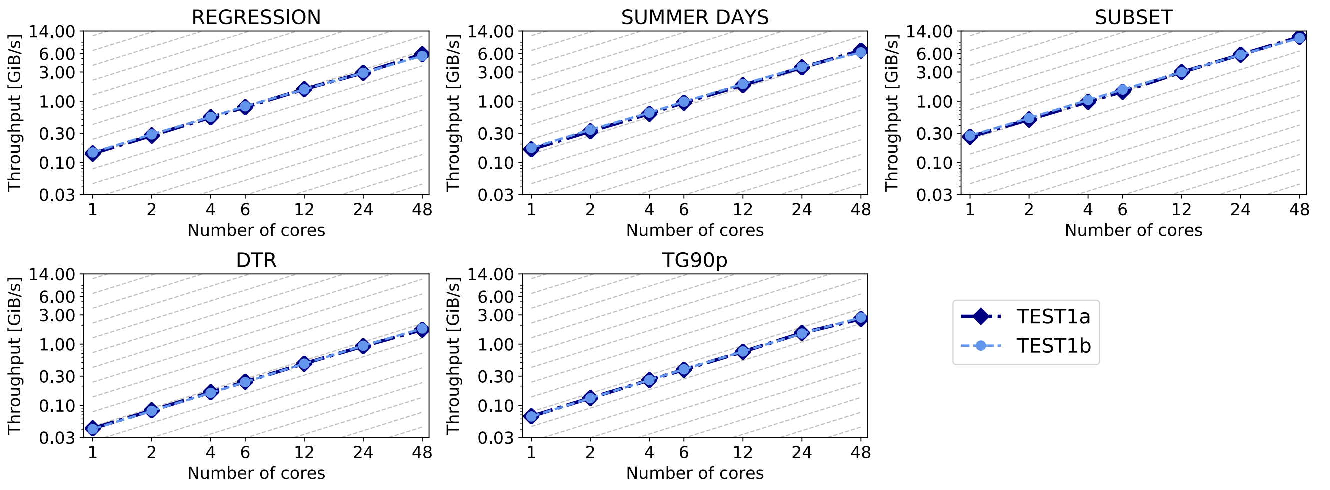

FIGURE 8. Log-log plot about the throughput for each operator run in TEST1a (with the CMIP6 dataset) and TEST1b (with synthetic data). The scalability of operators on a single node is close to linear in almost all cases (the diagonal grid lines provide a guide for comparison with respect to the ideal case). Results from TEST1a and TEST1b almost perfectly overlap, showing that there is no particular difference in the execution time when using synthetic data in the tests in place of real data.

plots in Figure 8 - which shows the throughput (computed as defined in (1)) for both tests - all the evaluated operations follow roughly the same behaviour, being very close to linear scalability until at least 12-24 threads. The diagonal grid lines provide a guide for comparison with respect to linear scalability. When 48 threads are used, there is a small reduction of performance due to a higher concurrency on resources and threads synchronization effects, although the throughput value in all cases is over 80% for TEST1a (and over 75% for TEST1b) of the linear scalability value. Moreover, the throughput values from TEST1a and TEST1b almost perfectly overlap, showing that there is no particular difference in the execution time when using synthetic data in the tests in place of real data.

This first batch of tests, although executed on limited resources, are important for the benchmark because they: (i) represent the baseline for comparison for the subsequent tests executed over multiple nodes and (ii) prove that synthetic data do not affect the scalability test results; only synthetic data were hence used in the multi-node tests.

2) STRONG SCALABILITY ON MULTIPLE NODES

This second strong scalability scenario *aims to evaluate the scalability of the runtime system while increasing the resources used to process a fixed data size over multiple nodes*.

In this case, the total data size was fixed throughout the various runs while increasing the number of threads. However, with respect to TEST1a/b, data were (evenly) distributed on the I/O & Analytics servers deployed over 48 (instead of 1) nodes. A single randomly-generated dataset of about 3.2 TiB (see Subsection V-C for details) was used throughout the full set of tests, resulting in about 67.4 GiB per node (as in the single node strong scalability case, TEST1b). The number of threads used to run the operators increased from 48 (i.e., the cores available on a single node) up to 2304 (i.e., the cores available on all 48 nodes). The number of threads was (almost) doubled on every run based on the following sequence: *48 - 96 - 192 - 288 - 576 - 1152 - 2304*. This sequence was chosen so that the cores would constantly be multiple of 48 and well-balanced, while achieving full resource utilization. In the rest of this document, such tests are referred to as TEST2.

TABLE 3. Results (average execution time in seconds) from *TEST2*.

Threads	**48**	**96**	**192**	**288**	**576**	**1152**	**2304**
Regression	450.54	233.86	116.26	80.04	42.73	24.55	14.31
Summer days	372.02	189.97	99.43	67.73	35.66	18.28	9.82
Subset	237.21	123.05	64.76	45.34	23.68	12.26	6.8
DTR	1864.95	930.97	463.78	323.0	160.27	82.25	44.17
T90P	1229.16	614.3	306.39	203.07	105.46	55.49	30.0

Table 3 shows the average execution time in seconds related to this test. The execution times are very similar to those in TEST1b. This outcome was expected since the data size per thread follows the same sequence in both tests; on the other hand, it also shows that the HPDA runtime is able to effectively scale out on multiple threads and nodes without any significant impact on performance.

Figure 9 provides a graphical representation of the scalability of the various operations. In this log-log plot, the throughput in GiB/s is presented; also in this case, the diagonal grid lines provide a guide for comparison with respect to linear scalability. It was decided to consider 48 threads as baseline, instead of a single thread, due to the enormous amount of time that would be required for the execution of the tests with few

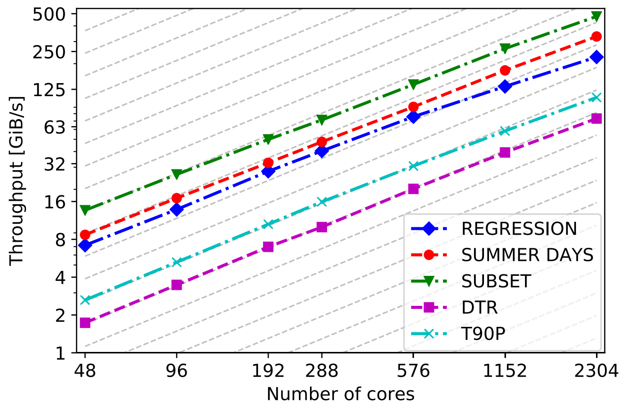

FIGURE 9. Log-log plot about the throughput for each operator run in TEST2. The scalability of the operators remains good in almost all cases even when using the maximum number of threads considered (the diagonal grid lines provide a guide for comparison with respect to the ideal case).

cores, which is not relevant at this stage, since it has already been proven to efficiently scale in TEST1a/b.

As it can be seen from the plot, scalability remains good even when using the maximum number of threads considered. In particular, the throughput value is at least at 83% and 76% of theoretical linear scalability (evaluated from the baseline case) up to 576 and 1152 threads, respectively. After this point, concurrency and synchronization effects cause a little deviation with respect to the ideal case, although general scalability remains good (over 72% of the linear scalability value) in all cases except for the REGRESSION test, which is around 65% (see next subsection for more details about the decrease in performance of this test).

It is worth noting that the highest decrease in scalability was observed in the fastest operation, also due to the major impact of synchronization and sequential stages on the smaller execution time (e.g., SUBSET with 2304 cores takes only 6 seconds), while the lowest decrease was observed in the slowest operation (e.g., DTR with 2304 is around 88% of the linear scalability value).

3) WEAK SCALABILITY ON MULTIPLE NODES

The third set of tests *aims to evaluate the scalability of the parallel runtime system while increasing the input data size linearly with the resources (i.e., the number of nodes considered); only the data size per core remains fixed.*

These experiments show the runtime system performance under weak scalability conditions: the number of threads used to run the operators was doubled for each run, together with the node count and the data size, from 48 (i.e., the cores available on a single node) up to 6144 (i.e., the cores available on 128 nodes). Data were loaded and distributed as a single dataset on the number of I/O & analytics nodes considered for each run. It is important to note that, although the amount of data per node remains fixed across different configurations, the data array length and the number of rows are multiplied

TABLE 4. Results (average execution time in seconds) from *TEST3a*.

Nodes	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**
Threads	**48**	**96**	**192**	**384**	**768**	**1536**	**3072**	**6144**
Data size [GiB]	**67.4**	**134.8**	**269.6**	**539.2**	**1078.4**	**2156.8**	**4313.6**	**8627.2**
Regression	9.83	9.74	10.39	10.69	11.26	10.75	15.6	32.61
Summer days	8.62	8.06	8.76	8.41	8.88	9.27	10.05	15.89
Subset	5.27	4.95	4.9	5.03	4.87	5.62	5.79	12.22
DTR	36.19	37.55	39.04	39.9	42.04	42.26	45.64	51.97
T90P	21.53	23.61	23.57	25.69	26.82	30.47	32.23	38.33

and divided by two, respectively, any time the number of nodes is doubled. When using a single node, for example, fragments are composed of 129k arrays with 2920 elements each; in a 2-node set up, the fragments consist of 65k arrays with 5840 elements each, whereas with 128 nodes, each fragment consists of around 1k arrays with 374k elements each.

The dataset was randomly generated, according to the structure described in Subsection V-C, to match the requirements of each configuration with the data size per core fixed at about 1.4 GiB, for a total data size ranging from 67.4 GiB to 8.4 TiB). In the rest of this document, this test is referred to as TEST3.

Furthermore, since the proposed runtime system allows for orchestrating operators execution by means of a hybrid MPI+X parallel approach (as described in Section IV-B), these tests were run by using a mixture of the two types of parallelism, with the following configurations:

- *TEST3a* with the following MPI+X parallelisms setup: *1 (cores) - 48 (threads); 1 - 96; 2 - 96; 4 - 96; 8 - 96; 16 - 96; 16 - 192; 32 - 192;*
- *TEST3b* using thread-level parallelism only (from *48* to *6144* threads).

For the sake of brevity, Table 4 shows only the average execution time in seconds related to the various configurations considered for TEST3a. Again, the case with 48 threads (full single node utilization) is considered as a baseline for comparison with the larger setup.

In this set of tests, scalability can be considered good in most cases up to 64 nodes, i.e., above 79% of the theoretical linear scalability (evaluated from the single node baseline case) except for the T90P and REGRESSION cases which stop at 67% and 63% respectively. Scalability decreases noticeably in most cases when using 128 nodes (i.e., 6144 cores), which corresponds to the most challenging test case for the runtime system, with 6144 threads allocated on the same single runtime node. This can be clearly seen in Figure 10, which shows the throughput for both TEST3a and TEST3b for each selected operator subject to testing.

As for the previous test, this behaviour becomes more relevant when the single-node operator execution time is

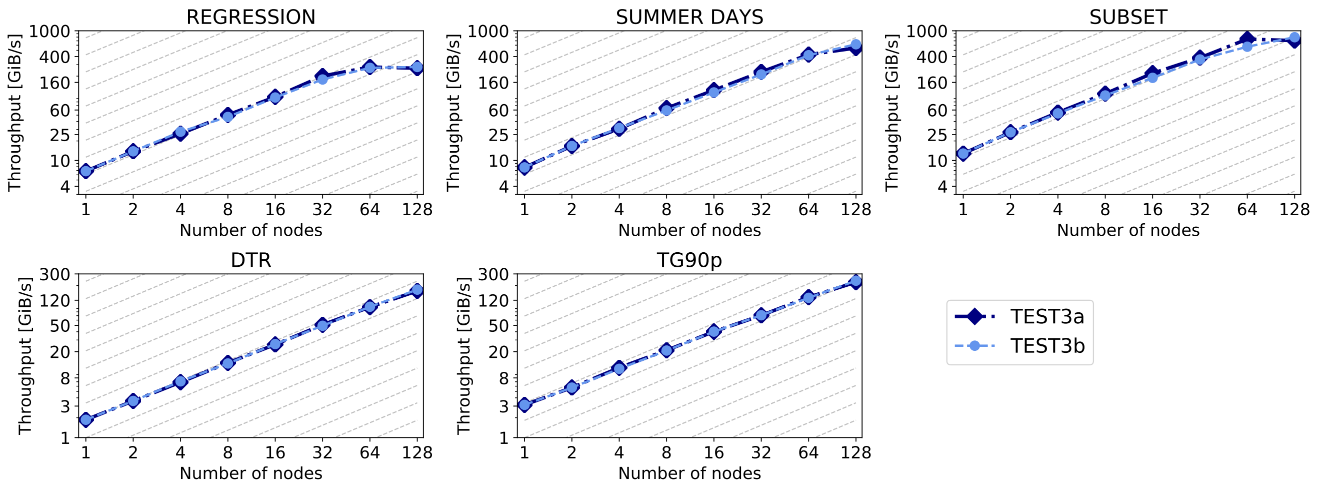

FIGURE 10. Log-log plot about the throughput for each operator run in TEST3a (with the MPI+X parallelism) and TEST3b (with thread-only parallelism). Scalability can be considered good up to 64 nodes except for the T90P and REGRESSION, while it decreases noticeably in most cases when using 128 nodes (the diagonal grid lines provide a guide for comparison with respect to the ideal case). Results from TEST3a and TEST3b almost perfectly overlap, showing that there is no particular difference in the two approaches.

small and, thus, the effects of contention and synchronization have a higher impact on the overall execution time (i.e., up to 6144 threads distributed on 128 I/O & analytics nodes have to be managed concurrently). In the case of the REGRESSION test, it is clear that when moving from 64 to 128 nodes, the time basically doubles. The reason for this behaviour is that the regression primitive requires both the array data and the time dimension values to perform the computation. In the current implementation, the dimension array is centrally managed by a single process to limit the memory footprint and, thus, when required for the computation, as in the case of regression, it needs to be transferred over the network from the runtime node to each I/O & Analytics server. Consequently, as the array length and the number of threads increase in the performed test, also the quantity of data to be transferred over the network rises exponentially by a factor of 4, from 1 MiB in the case of a single node up to 17 GiB in the case of 128 nodes. Therefore, the additional time measured in the case of 64 and 128 nodes is mainly due to this effect.

It is clear that following this trend any further increase in the array length and the number of threads will severely affect the execution time, leading to a clear performance degradation. However, it should be underlined that this is one of the few cases where this effect is noticed; still, the limit is only reached under experimental conditions, since time series with nearly 380k elements are basically unfeasible in real scenarios. Nevertheless, the test helped to understand to what extent the limitation can impact on performance and it will be used for future optimizations of the proposed HPDA framework.

Another interesting result that can be observed from the table is that the speedup is superlinear in the majority of the tested operators. This effect is due to the way data size is increased throughout the various configurations, as explained earlier in this section. The HPDA in-memory engine is in fact highly optimized to work on array-based data, hence having longer time series and fewer rows leads to improve the overall execution time thanks to data locality and caching effects. As expected, this superlinear speedup behaviour did not manifest in the strong scalability tests since the array length was fixed throughout the tests.

With respect to the table, the plots also show the results for the thread-only parallel configuration (i.e., TEST3b); as it can be observed, there is no significant difference in the execution time of the two cases; the comparison between thread-only and hybrid parallelism is further explored in the next test case.

4) COMPARISON OF MULTI-THREADS AND MIXED PARALLELISM APPROACHES

This final set of tests *aims to compare the performance of data analytics operators when exploiting multi-thread and multi-process parallelism approaches*.

For these tests, both the data size and the number of resources used (i.e., cores and nodes) was fixed throughout the runs and only the configuration of MPI process and Pthreads was changed, yet always retaining the same amount of total cores used (i.e, *number of MPI procs* $\times$ *number of threads*).

The number of nodes used in this case was set to 32 (with a total of 1536 cores available) and the data size to 1.05 TiB (see Subsection V-C for details) which provides a robust setting for this test. Data were loaded and distributed evenly over the 32 I/O & analytics nodes, resulting in 33.7 GiB per node as in TEST1a. For each run, the product of threads and MPI processes was constantly set to 1536, following this sequence: *1 process* $\times$ *1536 threads,* 2 *processes* $\times$

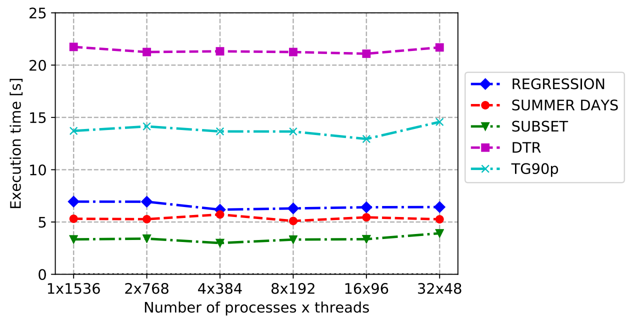

FIGURE 11. Semi-log (X-axis) plot about the execution time for each operator run in TEST4. The execution time is not significantly affected when threads are exchanged for MPI processes or vice versa.

768 threads, *4 processes* × *384 threads*, *8 processes* × *192 threads*, *16 processes* × *96 threads*, *32 processes* × *48 threads*. In the rest of this document, such tests are referred to as TEST4.

The semi-log plot in Figure 11 shows the results from this set of tests while changing the mix of MPI processes and POSIX threads used for computation. The *x axis* in the plot reports the configuration in terms of *number of processes* × *threads* used in each test. As also highlighted in the previous test, exchanging threads for MPI processes or vice versa does not significantly affect the execution time. Anyway, it is important to mention that using more MPI processes in the proposed HPDA runtime system would require allocating more nodes for the execution of the runtime system, thus resulting in a higher consumption of compute resources and power without any significant benefit.

F. DISCUSSION

This benchmark has highlighted different aspects of the proposed HPDA framework while scaling over multiple cores and nodes and with different data sizes.

The main results of the benchmark can be summarized as follows:

- the performance pattern for a particular operation is the same with both synthetic and real data;
- on a single node, the parallel runtime system provides good levels of performance, at least over 75% of the linear scalability value in all cases;
- when distributing the processing on multiple nodes, the performance of the proposed runtime system remains good in the majority of cases until 3k cores (i.e., over 70% with regard to linear scalability) in both the weak and strong scalability scenarios considered, although some limitations emerged during the tests;
- with a fixed data size, the tested primitives perform better when working on longer time series and show a superlinear behaviour, which proves the effectiveness of the array-based in-memory engine;
- multi-threaded and MPI+X parallelization approaches allow less resource usage in terms of cores (and in turn nodes) without any significant impact on performance. Consequently, given the proposed runtime system, it is best to rely on a higher number of threads rather than on MPI processes, that should only be exploited to scale over multiple nodes when it comes to larger scale scenarios.

Overall, the proposed HPDA runtime system and deployment mechanisms have proven to scale effectively over a large number of threads and nodes in a supercomputing environment, overcoming by one order of magnitude the scalability limits that affected previous releases [92]. Moreover, they gave us better insight into the framework behaviour alongside its new runtime system, and also helped us identify aspects that need to be further improved and optimized in the future, thus bringing important feedback to the software roadmap. Besides evaluations strictly related to scalability, the benchmarking results provide a strong baseline and reference for future releases of the framework as well as for experimental comparison with other tools in the HPDA area.

VI. CONCLUSION AND FUTURE WORK

This paper highlighted some of the challenges at the crossroads of HPC and BDA, which are key points to enable current and future HPDA at extreme scale.

The main design aspects of the Ophidia framework have been reviewed in light of such challenges and its redesign targeting HPDA has been presented. Focus has been placed on (i) the parallel runtime system, highlighting the different levels of parallelism designed for large-scale analysis, and on (ii) the main aspects concerning the integration of the framework with HPC ecosystem technologies and its deployment mechanism.

A large-scale experimental evaluation on a PRACE Tier-0 HPC cluster showed that the proposed system is able to scale on several thousands of computing units (i.e., cores) and on hundreds of nodes. An in-depth analysis has been made in order to properly validate the design principles and the implementation aspects by understanding the behaviour of the proposed runtime at intra-task execution level. To this end, an effort has been made to select a set of representative scientific operations and input datasets. The framework has shown good (intra-task) scalability in both the strong and weak sense for most of the benchmarked operations in multi-terabyte analytics scenarios. Such experimental results also provide a solid reference/baseline for future evaluation and comparison with newer versions of the HPDA framework as well as related work in the same research area.

The design and implementation aspects discussed in this work represent a contribution towards the long-way convergence path between Big Data and HPC for extreme-scale scientific data analytics that the scientific community is currently facing.

Future work will focus on:

- extreme-scale benchmarks (from tens to hundreds of thousands of single operators) to analyze and compare the scalability of the proposed HPDA framework with

respect to state-of-the-art tools on larger scientific use cases and workflow applications developed in the context of the ESiWACE Centre of Excellence as a set of large-scale HPDA demonstrators;
- improvements at the level of the runtime system to leverage data-driven resource management and job scheduling, as well as proactively applying optimizations to task execution and at the level of query planning;
- power efficiency improvement, mainly by optimizing data movement across nodes;
- including learning techniques to further extend the provided HPDA framework towards machine and deep learning as well as AI support to tackle intelligence-driven data-centric scenarios.

Finally, to further address deployability, portability and exploitability, a key development already ongoing relates to the use of HPC-enabled containerization technologies to target new paradigms like *HPDA as a Service* (HPDAaaS). That will be key to addressing the overarching framework vision of joining cloud computing (as enabling virtualization technology), HPC (as software ecosystem and computing environment) and HPDA (as data-centric paradigm) to deal with extreme-scale Big Data applications on peta/exascale supercomputing infrastructures.

ACKNOWLEDGMENT

The authors kindly acknowledge PRACE for awarding access to MareNostrum4 at Barcelona Supercomputing Center (BSC), Spain, as well as the support by the BSC Team. Moreover, the authors would like to acknowledge Antonio Aloisio for his editing and proofreading work on this article.

REFERENCES

[1] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, "Scientific data management in the coming decade," *ACM SIGMOD Rec.*, vol. 34, no. 4, pp. 34–41, Dec. 2005, doi: 10.1145/1107499.1107503.

[2] T. Hey, S. Tansley, and K. Tolle. (Oct. 2009). *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research. [Online]. Available: https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/

[3] G. Bell, T. Hey, and A. Szalay, "Beyond the data deluge," *Science*, vol. 323, no. 5919, pp. 1297–1298, Mar. 2009, doi: 10.1126/science.1170411.

[4] D. Laney, "3D data management: Controlling data volume velocity, and variety," Meta Group Res. Note, META Group, Stamford, CT, USA, 2001, vol. 6, no. 70.

[5] Y. Demchenko, C. D. Laat, and P. Membrey, "Defining architecture components of the big data ecosystem," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, May 2014, pp. 104–112, doi: 10.1109/CTS.2014.6867550.

[6] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014, doi: 10.1109/ACCESS.2014.2332453.

[7] M. Asch *et al.*, "Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry," *Int. J. High Perform. Comput. Appl.*, vol. 32, no. 4, pp. 435–479, Jul. 2018, doi: 10.1177/1094342018778123.

[8] (Mar. 2019). *The Argonne National Laboratory Supercomputer Will Enable High Performance Computing and Artificial Intelligence at Exascale by 2021*. Accessed: May 12, 2021. [Online]. Available: https://www.energy.gov/articles/us-department-energy-and-intel-build-first-exascale-supercomputer

[9] J. Dongarra, S. Gottlieb, and W. T. C. Kramer, "Race to exascale," *Comput. Sci. Eng.*, vol. 21, no. 1, pp. 4–5, Jan. 2019, doi: 10.1109/MCSE.2018.2882574.

[10] M. Anderson, "Will China attain exascale supercomputing in 2020?" *IEEE Spectr.*, vol. 57, no. 1, pp. 52–53, Jan. 2020, doi: 10.1109/MSPEC.2020.8946314.

[11] D. A. Reed and J. Dongarra, "Exascale computing and big data," *Commun. ACM*, vol. 58, no. 7, pp. 56–68, Jun. 2015, doi: 10.1145/2699414.

[12] B. Magro, J. Pike, and W. Stemple. (Jun. 2017). *High Performance Data Analytics: A Q&A With Subject Matter Experts. CIO Magazine*. [Online]. Available: https://www.cio.com/article/3204527/analytics/high-performance-data-analytics-a-qanda-with-subject-matter-experts.html

[13] S. Fiore, A. D'Anca, C. Palazzo, I. T. Foster, D. N. Williams, and G. Aloisio, "Ophidia: Toward big data analytics for escience," in *Proc. Int. Conf. Comput. Sci.*, Barcelona, Spain, 2013, pp. 2376–2385, doi: 10.1016/j.procs.2013.05.409.

[14] J. Dongarra *et al.*, "The international exascale software project roadmap," *Int. J. High Perform. Comput. Appl.*, vol. 25, no. 1, pp. 3–60, Feb. 2011, doi: 10.1177/1094342010391989.

[15] *ETP4HPC Strategic Research Agenda 2020 (SRA 4)—European HPC Research Priorities 2021-2024*. Accessed: May 12, 2021. [Online]. Available: https://www.etp4hpc.eu/pujades/files/ETP4HPC_SRA4_2020_web.pdf

[16] (May 2015). *A Digital Single Market Strategy for Europe—Communication From the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions*. Accessed: May 12, 2021. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52015DC0192&from=EN

[17] (May 2017). *Digital Single Market Mid-Term Review: Commission Calls for Swift Adoption of Key Proposals and Maps Out Challenges Ahead*. Accessed: May 12, 2021. [Online]. Available: https://ec.europa.eu/digital-single-market/en/news/digital-single-market-mid-term-review

[18] *Building a European Data Economy*. Accessed: May 12, 2021. [Online]. Available: https://eur-lex.europa.eu/content/news/building_EU_data_economy.html

[19] *European Data Strategy*. Accessed: May 12, 2021. [Online]. Available: https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/european-data-strategy

[20] B. Obama. (Jul. 2015). *Executive Order–Creating a National Strategic Computing Initiative. The White House, US*. Accessed: May 12, 2021. [Online]. Available: https://obamawhitehouse.archives.gov/the-press-office/2015/07/29/executive-order-creating-national-strategic-computing-initiative

[21] D. Kothe, S. Lee, and I. Qualters, "Exascale computing in the United States," *Comput. Sci. Eng.*, vol. 21, no. 1, pp. 17–29, Jan. 2019, doi: 10.1109/MCSE.2018.2875366.

[22] S. Matsuoka, H. Sato, O. Tatebe, M. Koibuchi, I. Fujiwara, S. Suzuki, M. Kakuta, T. Ishida, Y. Akiyama, T. Suzumura, K. Ueno, H. Kanezashi, and T. Miyoshi, "Extreme big data (EBD): Next generation big data infrastructure technologies towards yottabyte/year," *Supercomput. Frontiers Innov.*, vol. 1, no. 2, pp. 89–107, 2014, doi: 10.14529/jsfi140206.

[23] K. Kinkade. (Dec. 2015) *NERSC, Berkeley Lab Explore Frontiers of Deep Learning for Science*. NERSC News. Accessed: May 12, 2021. [Online]. Available: http://www.nersc.gov/news-publications/nersc-news/science-news/2015/nersc-berkeley-lab-explore-frontiers-of-deep-learning-for-science/

[24] D. George and E. A. Huerta, "Deep learning for real-time gravitational wave detection and parameter estimation: Results with advanced LIGO data," *Phys. Lett. B*, vol. 778, pp. 64–70, Mar. 2018, doi: 10.1016/j.physletb.2017.12.053.

[25] J. M. Wozniak, R. Jain, P. Balaprakash, J. Ozik, N. T. Collier, J. Bauer, F. Xia, T. Brettin, R. Stevens, J. Mohd-Yusof, C. G. Cardona, B. V. Essen, and M. Baughman, "CANDLE/supervisor: A workflow framework for machine learning applied to cancer research," *BMC Bioinf.*, vol. 19, no. 18, p. 491, Dec. 2018, doi: 10.1186/s12859-018-2508-4.

[26] S. E. Haupt and B. Kosovic, "Big data and machine learning for applied weather forecasts: Forecasting solar power for utility operations," in *Proc. IEEE Symp. Ser. Comput. Intell.*, Dec. 2015, pp. 496–501, doi: 10.1109/SSCI.2015.79.

[27] D. S. Katz, L. C. McInnes, D. E. Bernholdt, A. C. Mayes, N. P. C. Hong, J. Duckles, S. Gesing, M. A. Heroux, S. Hettrick, R. C. Jimenez, M. Pierce, B. Weaver, and N. Wilkins-Diehr, "Community organizations: Changing the culture in which research software is developed and sustained," *Comput. Sci. Eng.*, vol. 21, no. 2, pp. 8–24, Mar. 2019, doi: 10.1109/MCSE.2018.2883051.

[28] S. Jha, J. Qiu, A. Luckow, P. Mantha, and G. C. Fox, ''A tale of two data-intensive paradigms: Applications, abstractions, and architectures,'' in *Proc. IEEE Int. Congr. Big Data*, Jun./Jul. 2014, pp. 645–652, doi: 10.1109/BigData.Congress.2014.137.

[29] Hadoop. *Apache Software Foundation*. Accessed: May 12, 2021. [Online]. Available: https://hadoop.apache.org

[30] J. Dean and S. Ghemawat, ''MapReduce: Simplified data processing on large clusters,'' *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: 10.1145/1327452.1327492.

[31] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, ''The Hadoop distributed file system,'' in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, May 2010, pp. 1–10, doi: 10.1109/MSST.2010.5496972.

[32] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, ''A survey of open source tools for machine learning with big data in the Hadoop ecosystem,'' *J. Big Data*, vol. 2, no. 1, p. 24, Dec. 2015, doi: 10.1186/s40537-015-0032-1.

[33] Kylin. *Apache Software Foundation*. Accessed: May 12, 2021. [Online]. Available: http://kylin.apache.org

[34] F. Yang, E. Tschetter, X. Léauté, N. Ray, G. Merlino, and D. Ganguli, ''Druid: A real-time analytical data store,'' in *Proc. ACM SIGMOD Int. Conf. Manage. Data*. New York, NY, USA: Association Computing Machinery, Jun. 2014, pp. 157–168, doi: 10.1145/2588555.2595631.

[35] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang, ''In-memory big data management and processing: A survey,'' *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1920–1948, Jul. 2015, doi: 10.1109/TKDE.2015.2427795.

[36] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, ''Spark: Cluster computing with working sets,'' in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.* Berkeley, CA, USA: USENIX Association, 2010, p. 10.

[37] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, ''Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,'' in *Proc. 9th USENIX Symp. Netw. Syst. Design Implement.* San Jose, CA, USA: USENIX Association, Apr. 2012, pp. 15–28. [Online]. Available: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia

[38] A. Ailamaki, V. Kantere, and D. Dash, ''Managing scientific data,'' *Commun. ACM*, vol. 53, no. 6, pp. 68–78, Jun. 2010, doi: 10.1145/1743546.1743568.

[39] M. Stonebraker, P. Brown, A. Poliakov, and S. Raman, ''The architecture of SciDB,'' in *Proc. 23rd Int. Conf. Sci. Stat. Database Manage.* Berlin, Germany: Springer-Verlag, 2011, pp. 1–16, doi: 10.1007/978-3-642-22351-8_1.

[40] P. G. Brown, ''Overview of sciDB: Large scale array storage, processing and analysis,'' in *Proc. Int. Conf. Manage. Data (SIGMOD)*. New York, NY, USA: ACM, 2010, pp. 963–968, doi: 10.1145/1807167.1807271.

[41] M. Stonebraker, P. Brown, D. Zhang, and J. Becla, ''SciDB: A database management system for applications with complex analytics,'' *Comput. Sci. Eng.*, vol. 15, no. 3, pp. 54–62, May 2013, doi: 10.1109/MCSE.2013.19.

[42] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann, ''The multidimensional database system RasDaMan,'' *ACM SIGMOD Rec.*, vol. 27, no. 2, pp. 575–577, Jun. 1998, doi: 10.1145/276305.276386.

[43] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann, ''Spatio-temporal retrieval with RasDaMan,'' in *Proc. 25th Int. Conf. Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1999, pp. 746–749. [Online]. Available: http://www.vldb.org/conf/1999/P74.pdf

[44] R. Palamuttam, R. M. Mogrovejo, C. Mattmann, B. Wilson, K. Whitehall, R. Verma, L. McGibbney, and P. Ramirez, ''SciSpark: Applying in-memory distributed computing to weather event detection and tracking,'' in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct. 2015, pp. 2020–2026, doi: 10.1109/BigData.2015.7363983.

[45] B. Wilson, R. Palamuttam, K. Whitehall, C. Mattmann, A. Goodman, M. Boustani, S. Shah, P. Zimdars, and P. Ramirez, ''SciSpark: Highly interactive in-memory science data analytics,'' in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 2964–2973, doi: 10.1109/BigData.2016.7840948.

[46] F. Hu, C. Yang, J. L. Schnase, D. Q. Duffy, M. Xu, M. K. Bowen, T. Lee, and W. Song, ''ClimateSpark: An in-memory distributed computing framework for big climate data analytics,'' *Comput. Geosci.*, vol. 115, pp. 154–166, Jun. 2018, doi: 10.1016/j.cageo.2018.03.011.

[47] N. Chaimov, A. Malony, S. Canon, C. Iancu, K. Z. Ibrahim, and J. Srinivasan, ''Scaling spark on HPC systems,'' in *Proc. 25th ACM Int. Symp. High-Performance Parallel Distrib. Comput.* New York, NY, USA: Association Computing Machinery, May 2016, pp. 97–110, doi: 10.1145/2907294.2907310.

[48] Z. Zhang, K. Barbary, F. A. Nothaft, E. Sparks, O. Zahn, M. J. Franklin, D. A. Patterson, and S. Perlmutter, ''Scientific computing meets big data technology: An astronomy use case,'' in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct. 2015, pp. 918–927, doi: 10.1109/BigData.2015.7363840.

[49] S. Caino-Lores, J. Carretero, B. Nicolae, O. Yildiz, and T. Peterka, ''Toward high-performance computing and big data analytics convergence: The case of spark-DIY,'' *IEEE Access*, vol. 7, pp. 156929–156955, 2019, doi: 10.1109/ACCESS.2019.2949836.

[50] A. Scionti *et al.*, ''HPC, cloud and big-data convergent architectures: The LEXIS approach,'' in *Complex, Intelligent, and Software Intensive Systems*, L. Barolli, F. K. Hussain, and M. Ikeda, Eds. Cham, Switzerland: Springer, 2020, pp. 200–212, doi: 10.1007/978-3-030-22354-0_19.

[51] Y. Georgiou, N. Zhou, L. Zhong, D. Hoppe, M. Pospieszny, N. Papadopoulou, K. Nikas, O. L. Nikolos, P. Kranas, S. Karagiorgou, E. Pascolo, M. Mercier, and P. Velho, ''Converging HPC, big data and cloud technologies for precision agriculture data analytics on supercomputers,'' in *High Performance Computing*, H. Jagode, H. Anzt, G. Juckeland, and H. Ltaief, Eds. Cham, Switzerland: Springer, 2020, pp. 368–379, doi: 10.1007/978-3-030-59851-8_25.

[52] M. Rocklin, ''Dask: Parallel computation with blocked algorithms and task scheduling,'' in *Proc. 14th Python Sci. Conf.*, K. Huff and J. Bergstra, Eds. Austin, TX, USA: SciPy, 2015, pp. 130–136, doi: 10.25080/Majora-7b98e3ed-013.

[53] Dask Development Team. (2016). *Dask: Library for Dynamic Task Scheduling*. Accessed: May 12, 2021. [Online]. Available: https://dask.org

[54] C. R. Harris *et al.*, ''Array programming with NumPy,'' *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[55] W. McKinney, ''Data structures for statistical computing in Python,'' in *Proc. 9th Python Sci. Conf.*, S. van der Walt and J. Millman, Eds. Austin, TX, USA: SciPy, 2010, pp. 56–61, doi: 10.25080/Majora-92bf1922-00a.

[56] S. Hoyer and J. J. Hamman, ''Xarray: N-D labeled arrays and datasets in Python,'' *J. Open Res. Softw.*, vol. 5, p. 6, Apr. 2017, Art. no. 10, doi: 10.5334/jors.148.

[57] Met Office. *Iris: A Python Library for Analysing and Visualising Meteorological and Oceanographic Data Sets, Exeter, Devon, 2010–2020*. Accessed: May 12, 2021. [Online]. Available: http://scitools.org.uk/

[58] J. Conejero, S. Corella, R. M. Badia, and J. Labarta, ''Task-based programming in COMPSs to converge from HPC to big data,'' *Int. J. High Perform. Comput. Appl.*, vol. 32, no. 1, pp. 45–60, Jan. 2018, doi: 10.1177/1094342017701278.

[59] J. Á. Cid-Fuentes, P. Álvarez, R. Amela, K. Ishii, R. K. Morizawa, and R. M. Badia, ''Efficient development of high performance data analytics in Python,'' *Future Gener. Comput. Syst.*, vol. 111, pp. 570–581, Oct. 2020, doi: 10.1016/j.future.2019.09.051.

[60] R. Rew and G. Davis, ''NetCDF: An interface for scientific data access,'' *IEEE Comput. Graph. Appl.*, vol. 10, no. 4, pp. 76–82, Jul. 1990, doi: 10.1109/38.56302.

[61] J. Gregory, ''The CF metadata standard,'' *CLIVAR Exchanges*, vol. 8, no. 4, p. 4, 2003.

[62] (Jun. 2015). *MPI: A Message-Passing Interface Standard, Version 3.1. Message Passing Interface Forum*. [Online]. Available: https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf

[63] G. Aloisio, S. Fiore, I. Foster, and D. Williams, ''Scientific big data analytics challenges at large scale,'' in *Proc. Big Data Extreme-scale Comput. (BDEC)*, 2013, pp. 1–3.

[64] U. Schulzweida. (Oct. 2020). *CDO User Guide—Version 1.9.9*. [Online]. Available: https://code.mpimet.mpg.de/projects/cdo/embedded/cdo.pdf

[65] C. S. Zender, ''Analysis of self-describing gridded geoscience data with netCDF operators (NCO),'' *Environ. Model. Softw.*, vol. 23, nos. 10–11, pp. 1338–1342, Oct. 2008, doi: 10.1016/j.envsoft.2008.03.004.

[66] UCAR/NCAR/CISL/TDD, Boulder, CO, USA. (2019). *The NCAR Command Language (Version 6.6.2) [Software]*. [Online]. Available: http://dx.doi.org/10.5065/D6WD3XH5

[67] S. Fiore, D. Elia, C. Palazzo, A. D'Anca, F. Antonio, D. N. Williams, I. Foster, and G. Aloisio, ''Towards an open (Data) science analytics-hub for reproducible multi-model climate analysis at scale,'' in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 3226–3234, doi: 10.1109/BigData.2018.8622205.

[68] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, 1998, doi: 10.1109/99.660313.

[69] G. C. Fox, J. Qiu, S. Kamburugamuve, S. Jha, and A. Luckow, "HPC-ABDS high performance computing enhanced apache big data stack," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2015, pp. 1057–1066, doi: 10.1109/CCGrid.2015.122.

[70] D. Talia, "A view of programming scalable data analysis: From clouds to exascale," *J. Cloud Comput.*, vol. 8, no. 1, p. 4, Dec. 2019, doi: 10.1186/s13677-019-0127-x.

[71] R. Badia, E. Ayguade, and J. Labarta, "Workflows for science: A challenge when facing the convergence of HPC and big data," *Supercomput. Frontiers Innov., Int. J.*, vol. 4, no. 1, pp. 27–47, Mar. 2017, doi: 10.14529/jsfi170102.

[72] H. Asaadi, D. Khaldi, and B. Chapman, "A comparative survey of the HPC and big data paradigms: Analysis and experiments," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Sep. 2016, pp. 423–432, doi: 10.1109/CLUSTER.2016.21.

[73] D. Petcu *et al.*, "On processing extreme data," *Scalable Computing: Pract. Exper.*, vol. 16, no. 4, pp. 467–489, Jan. 2016, doi: 10.12694/scpe.v16i4.1134.

[74] P. Xuan, J. Denton, P. K. Srimani, R. Ge, and F. Luo, "Big data analytics on traditional HPC infrastructure using two-level storage," in *Proc. Int. Workshop Data-Intensive Scalable Comput. Syst. (DISCS)*. New York, NY, USA: Association Computing Machinery, 2015, pp. 1–8, doi: 10.1145/2831244.2831253.

[75] J. Liao, B. Gerofi, G.-Y. Lien, T. Miyoshi, S. Nishizawa, H. Tomita, W.-K. Liao, A. Choudhary, and Y. Ishikawa, "A flexible I/O arbitration framework for netCDF-based big data processing workflows on high-end supercomputers," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 15, p. e4161, Aug. 2017, doi: 10.1002/cpe.4161.

[76] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Germany: Springer, 2003, pp. 44–60, doi: 10.1007/10968987_3.

[77] IBM. *Specturm LSF*. Accessed: May 12, 2021. [Online]. Available: https://www.ibm.com/docs/en/spectrum-lsf/10.1.0

[78] R. L. Henderson, "Job scheduling under the portable batch system," in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph, Eds. Berlin, Germany: Springer, 1995, pp. 279–294, doi: 10.1007/3-540-60153-8_34.

[79] A. Reuther, C. Byun, W. Arcand, D. Bestor, B. Bergeron, M. Hubbell, M. Jones, P. Michaleas, A. Prout, A. Rosa, and J. Kepner, "Scheduler technologies in support of high performance data analysis," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2016, pp. 1–6, doi: 10.1109/HPEC.2016.7761604.

[80] C. Byun, J. Kepner, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Hubbell, P. Michaleas, J. Mullen, A. Prout, A. Rosa, C. Yee, and A. Reuther, "LLMapReduce: Multi-level map-reduce for high performance data analysis," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2016, pp. 1–8, doi: 10.1109/HPEC.2016.7761618.

[81] M. Mercier, D. Glesser, Y. Georgiou, and O. Richard, "Big data and HPC collocation: Using HPC idle resources for big data analytics," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 347–352, doi: 10.1109/BigData.2017.8257944.

[82] A. J. Younge, K. Pedretti, R. E. Grant, and R. Brightwell, "A tale of two systems: Using containers to deploy HPC applications on supercomputers and clouds," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2017, pp. 74–81, doi: 10.1109/CloudCom.2017.40.

[83] R. S. Canon and A. Younge, "A case for portability and reproducibility of HPC containers," in *Proc. IEEE/ACM Int. Workshop Containers New Orchestration Paradigms Isolated Environments HPC (CANOPIE-HPC)*, Nov. 2019, pp. 49–54, doi: 10.1109/CANOPIE-HPC49598.2019.00012.

[84] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, p. 2, Mar. 2014.

[85] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PLoS ONE*, vol. 12, no. 5, pp. 1–20, 2017, doi: 10.1371/journal.pone.0177459.

[86] D. M. Jacobsen and R. S. Canon, "Contain this, unleashing Docker for HPC," in *Proc. Cray User Group*, 2015, pp. 33–49.

[87] L. Benedicic, F. A. Cruz, A. Madonna, and K. Mariotti, "Sarus: Highly scalable Docker containers for HPC systems," in *High Performance Computing*, M. Weiland, G. Juckeland, S. Alam, and H. Jagode, Eds. Cham, Switzerland: Springer, 2019, pp. 46–60, doi: 10.1007/978-3-030-34356-9_5.

[88] S. Fiore, C. Palazzo, A. D'Anca, D. Elia, E. Londero, C. Knapic, S. Monna, N. M. Marcucci, F. Aguilar, M. Płóciennik, J. E. M. D. Lucas, and G. Aloisio, "Big data analytics on large-scale scientific datasets in the INDIGO-DataCloud project," in *Proc. Comput. Frontiers Conf.* New York, NY, USA: ACM, May 2017, pp. 343–348, doi: 10.1145/3075564.3078884.

[89] S. Fiore, D. Elia, C. E. Pires, D. G. Mestre, C. Cappiello, M. Vitali, N. Andrade, T. Braz, D. Lezzi, R. Moraes, T. Basso, N. P. Kozievitch, K. V. O. Fonseca, N. Antunes, M. Vieira, C. Palazzo, I. Blanquer, W. Meira, and G. Aloisio, "An integrated big and fast data analytics platform for smart urban transportation management," *IEEE Access*, vol. 7, pp. 117652–117677, 2019, doi: 10.1109/ACCESS.2019.2936941.

[90] D. Elia, S. Fiore, A. D'Anca, C. Palazzo, I. Foster, and D. N. Williams, "An in-memory based framework for scientific data analytics," in *Proc. ACM Int. Conf. Comput. Frontiers*. New York, NY, USA: ACM, May 2016, pp. 424–429, doi: 10.1145/2903150.2911719.

[91] C. Palazzo, A. Mariello, S. Fiore, A. D'Anca, D. Elia, D. N. Williams, and G. Aloisio, "A workflow-enabled big data analytics software stack for escience," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Amsterdam, The Netherlands, Jul. 2015, pp. 545–552, doi: 10.1109/HPCSim.2015.7237088.

[92] S. Fiore, D. Elia, C. Palazzo, F. Antonio, A. D'Anca, I. Foster, and G. Aloisio, "Towards high performance data analytics for climate change," in *High Performance Computing*, M. Weiland, G. Juckeland, S. Alam, and H. Jagode, Eds. Cham, Switzerland: Springer, 2019, pp. 240–257, doi: 10.1007/978-3-030-34356-9_20.

[93] A. D'Anca, C. Palazzo, D. Elia, S. Fiore, I. Bistinas, K. Bottcher, V. Bennett, and G. Aloisio, "On the use of in-memory analytics workflows to compute eScience indicators from large climate datasets," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2017, pp. 1035–1043, doi: 10.1109/CCGRID.2017.132.

[94] Barcelona Supercomputing Center. (2020). *MareNostrum4 User's Guide*. [Online]. Available: https://www.bsc.es/support/MareNostrum4-ug.pdf

[95] R. Han, L. K. John, and J. Zhan, "Benchmarking big data systems: A review," *IEEE Trans. Services Comput.*, vol. 11, no. 3, pp. 580–597, May 2018, doi: 10.1109/TSC.2017.2730882.

[96] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi, *GNU Sci. Library Reference Manual*, 3rd ed. Bristol, U.K.: Network Theory Ltd., Jan. 2009.

[97] ETCCDI. *Climate Change Indices, Definitions of the 27 Core Indices*. Accessed: May 12, 2021. [Online]. Available: http://etccdi.pacificclimate.org/list_27_indices.shtml

[98] V. Eyring, S. Bony, G. A. Meehl, C. A. Senior, B. Stevens, R. J. Stouffer, and K. E. Taylor, "Overview of the coupled model intercomparison project phase 6 (CMIP6) experimental design and organization," *Geosci. Model Develop.*, vol. 9, no. 5, pp. 1937–1958, May 2016, doi: 10.5194/gmd-9-1937-2016.

[99] R. Petrie *et al.*, "Coordinating an operational data distribution network for CMIP6 data," *Geosci. Model Develop.*, vol. 14, no. 1, pp. 629–644, Jan. 2021, doi: 10.5194/gmd-14-629-2021.

[100] E. Scoccimarro, A. Bellucci, and D. Peano. (2018). *CMCC CMCC-CM2-VHR4 Model Output Prepared for CMIP6 HighResMIP Hist-1950*. [Online]. Available: https://doi.org/10.22033/ESGF/CMIP6.3818

[101] L. Cinquini, D. Crichton, C. Mattmann, J. Harney, G. Shipman, F. Wang, R. Ananthakrishnan, N. Miller, S. Denvil, M. Morgan, Z. Pobre, G. M. Bell, C. Doutriaux, R. Drach, D. Williams, P. Kershaw, S. Pascoe, E. Gonzalez, S. Fiore, and R. Schweitzer, "The Earth system grid federation: An open infrastructure for access to distributed geospatial data," *Future Gener. Comput. Syst.*, vol. 36, pp. 400–417, Jul. 2014, doi: 10.1016/j.future.2013.07.002.

[102] C. Zender. (2021). *NCO User Guide*. [Online]. Available: http://nco.sourceforge.net/nco.pdf

[103] K. Breinl, T. Turkington, and M. Stowasser, "Simulating daily precipitation and temperature: A weather generation framework for assessing hydrometeorological hazards," *Meteorolog. Appl.*, vol. 22, no. 3, pp. 334–347, Jul. 2015, doi: 10.1002/met.1459.

[104] K. Moreland and R. Oldfield, "Formal metrics for large-scale parallel performance," in *High Performance Computing*, J. M. Kunkel and T. Ludwig, Eds. Cham, Switzerland: Springer, 2015, pp. 488–496, doi: 10.1007/978-3-319-20119-1_34.

[105] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

DONATELLO ELIA (Member, IEEE) received the M.Sc. degree in computer engineering from the University of Salento, Italy, in 2013, where he is currently pursuing the Ph.D. degree in engineering of complex systems. In 2013, he joined the Advanced Scientific Computing (ASC) Division, Euro-Mediterranean Centre on Climate Change (CMCC) Foundation. He has been involved in various European projects, like the FP7 and Horizon 2020 Programmes. He has authored or coauthored various articles in refereed journals and conference proceedings. His main research interests include data science, HPC, cloud computing, big data, data-intensive analytics, and scientific data management. He is a member of the IEEE Computer Society.

SANDRO FIORE (Member, IEEE) received the Ph.D. degree in innovative materials and technologies from the University of Lecce, in 2001. He is currently an Associate Professor with the Department of Information Engineering and Computer Science (DISI), University of Trento, where he also leads the High Performance Climate Informatics Laboratory and a Lecturer with the School of Innovation. Formerly (since 2006) at the Advanced Scientific Computing Division, Euro-Mediterranean Centre on Climate Change (CMCC) Foundation, he was the Division Director, from 2014 to 2018, and the Head of the Data Science Research Team, from 2018 to 2020. Since 2011, he has been the Visiting Scientist of the Lawrence Livermore National Laboratory and more recently with the CS Department, The University of Chicago. He is the author of about 100 scientific articles, an Editor of the book *Grid and Cloud Database Management*, and a coauthor of *The International Exascale Software Project Roadmap*. His research interests include data science and learning, scientific data management, big data, and artificial intelligence applied to climate change in extreme-scale HPC environments. He has been involved in several national and EU projects. Among others, some on exascale challenges at European and International level include IESP, EESI, EXDCI, and BDEC. He is a member of ACM.

GIOVANNI ALOISIO is currently a Full Professor with the University of Salento, Italy, where he leads the HPC Laboratory. At the Euro-Mediterranean Centre on Climate Change (CMCC) Foundation, he is a member of the CMCC Strategic Council and the Director of the CMCC Supercomputing Center. He is the author of more than 250 articles in refereed journals on high-performance computing, grid and cloud computing, and distributed data management. He has also contributed to the International Exascale Software Project Exascale Roadmap. His expertise concerns high-performance computing, grid and cloud computing, and distributed data management. He has been involved in several EU projects. He is a member of the ENES HPC Task Force. He has been the Chair of the European Panel of Experts on Weather Climate and Solid Earth Sciences WG.