evolve

Leading the Big Data Revolution

The EVOLVE Platform

HPC and Cloud Enhanced Testbed for Extracting Value from Large-scale Diverse Data

Antony Chazapis
Institute of Computer Science, FORTH
Heraklion, Greece
chazapis@ics.forth.gr

Jean-Thomas Acquaviva
DataDirect Networks
Paris, France
jtacquaviva@ddn.com

Angelos Bilas*
Institute of Computer Science, FORTH
Heraklion, Greece
bilas@ics.forth.gr

Georgios Gardikis
Space Hellas S.A.
Athens, Greece
ggar@space.gr

Christos Kozanitis
Institute of Computer Science, FORTH
Heraklion, Greece
kozanitis@ics.forth.gr

Stelios Louloudakis
Sunlight.io
Heraklion, Greece
stelios.louloudakis@sunlight.io

Huy-Nam Nguyen
ATOS/Bull
Paris, France
Huy-Nam.Nguyen@atos.net

Christian Pinto
IBM Research Europe
Dublin, Ireland
christian.pinto@ibm.com

Arno Scharl
webLyzard technology
Vienna, Austria
scharl@weblyzard.com

Dimitrios Soudris
School of Electrical and Computer Engineering, National Technical University of Athens
Athens, Greece
dsoudris@microlab.ntua.gr

Abstract

EVOLVE is a pan-European Innovation Action building a converged infrastructure to bring together the HPC, Cloud, and Big Data worlds. EVOLVE's platform and software stack supports large-scale, data-intensive applications, driven primarily by industry requirements set by pilot and proof-of-concept use cases from diverse fields. Given the unprecedented data growth we are experiencing, EVOLVE's infrastructure is key in enabling the cost-effective processing of massive amounts of data and the adaptation of multiple high-end technologies, in an environment that fosters interoperability and enforces increased security.

* Also with University of Crete, Computer Science Department.

 ACM ISBN 978-1-4503-8404-9/21/05. . . $15.00

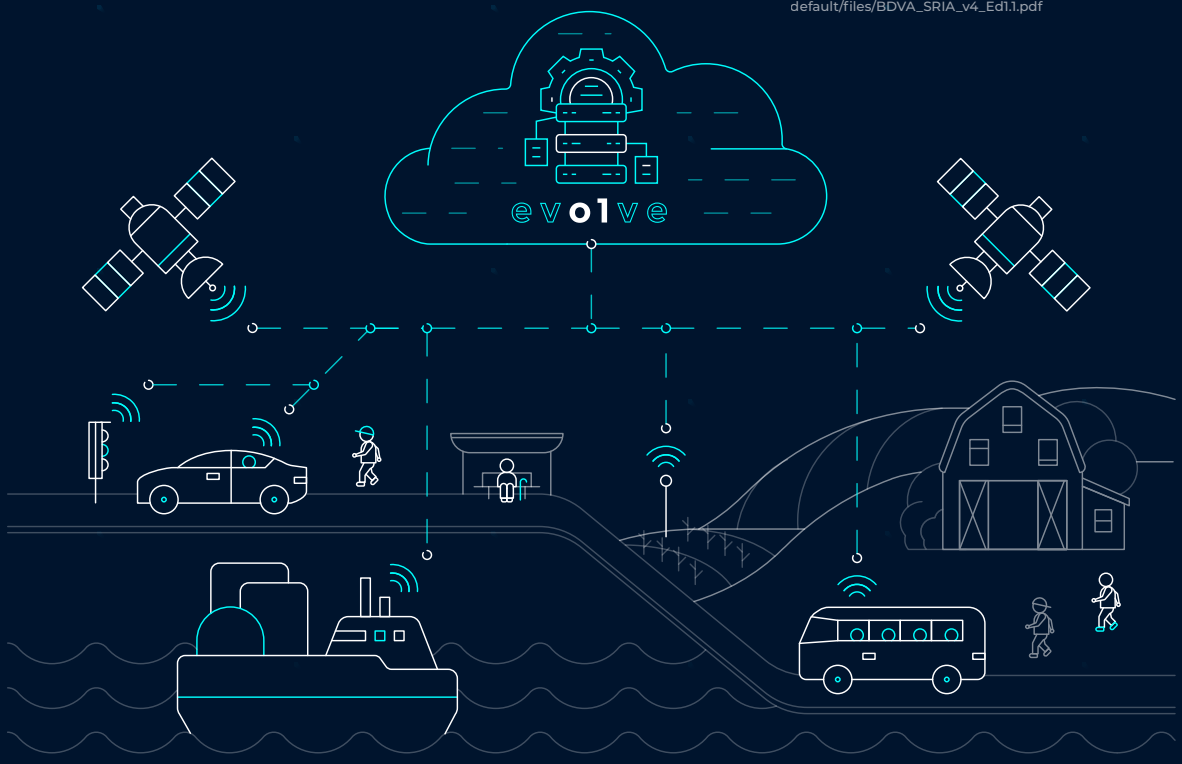
Overview

Processing large datasets is emerging as a main challenge of modern compute infrastructures. Applications require an increasing amount of processing power as data grows at an unprecedented rate [1]. Although tremendous progress has happened over the past several years on increasing productivity for data processing over commodity systems and providing new services with Big Data(BD) and Cloud technologies, the projected data deluge brings business, consumers, and the society in general at a new frontier:

How can we process massive data that require demanding computation?

Data analytics at scale requires extensive computing infrastructure, either on-premise or in the Cloud, which in turn demands extensive expertise, both to operate, as well as to express and tune application logic so it can exploit the full benefits of the available hardware. Cost (CAPEX as well as OPEX), a steep learning curve, and portability, are the main barriers to scaling out applications. This results in long turn-around times for domain experts to actually process data and therefore design new services; long turn-around times for large industry, SMEs, and startups to implement and deploy new services; and long turn-around times for users and consumers, that in many cases would prefer near-real-time responses.

[1] Big Data Value Association (BDVA). 2017. European Big Data Value Strategic Research and Innovation Agenda (SRIA), Version 4.0. (10 2017). https://www. bdva.eu/sites/default/files/BDVA_SRIA_v4_Ed1.1.pdf



EVOLVE addresses these issues by offering a novel, integrated computing environment that boosts productivity and allows interoperability, while maintaining hardware-specific performance benefits.

In EVOLVE, applications express their logic and required datasets in the form of workflows that can be automated, shared, refined, and maintained across groups of domain experts without significant IT expertise.

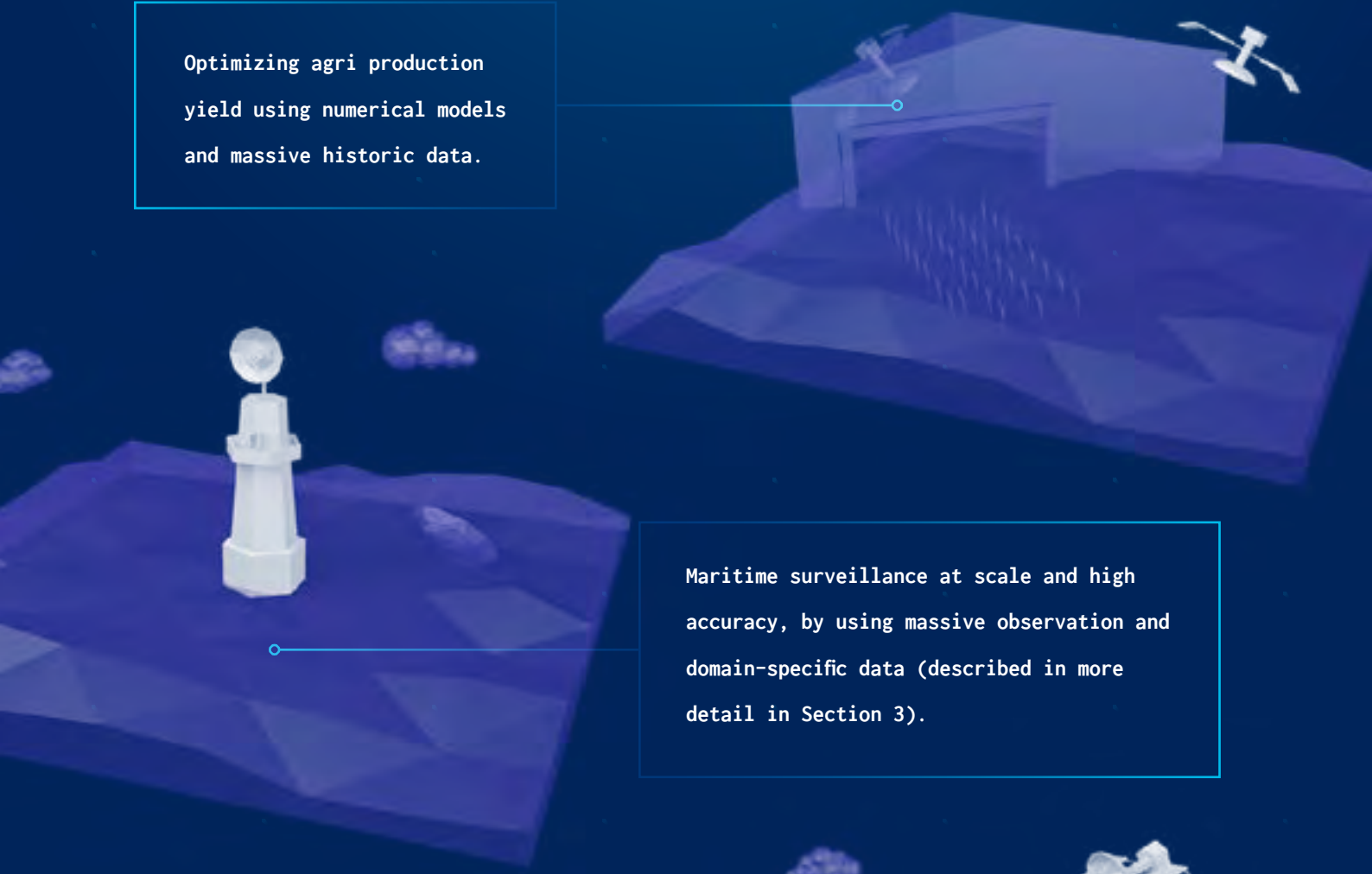
Workflows and data are manipulated through a web-based dashboard, and code – expressed in a portable notebook format – executes seamlessly on HPC hardware, using a rich and versatile set of Big Data processing frameworks. The EVOLVE testbed is shared across applications using a Kubernetes-based execution framework, where all workflow stages are containerized, facilitating ease of deployment, isolation, portability and reproducibility. At the same time, the testbed includes provisions for data protection (security and integrity), the main reason that hinders shared deployment in domains with sensitive data.

Unique EVOLVE features include:

- Complex application orchestration, fully containerized and controlled through our workflow definition language.
- A unified data management layer, that allows users to transparently access their local or Cloud-based datasets as files, both through the dashboard and in containers.
- Custom, fine-grain resource allocation and scheduling schemes, as well as accelerator sharing methods and policies – important steps forward in terms of data center efficiency.
- Tight accelerator integration with BD frameworks, such as Spark, both for batch processing and streaming applications.
- Seamless execution of HPC applications as workflow stages, with Slurm compatibility.
- Embedded visualization modules with innovative and responsive features, allowing efficient and versatile interaction with data.
- Efficient monitoring of utilization, performance, and QoS for the whole wsystem and individual workflows.

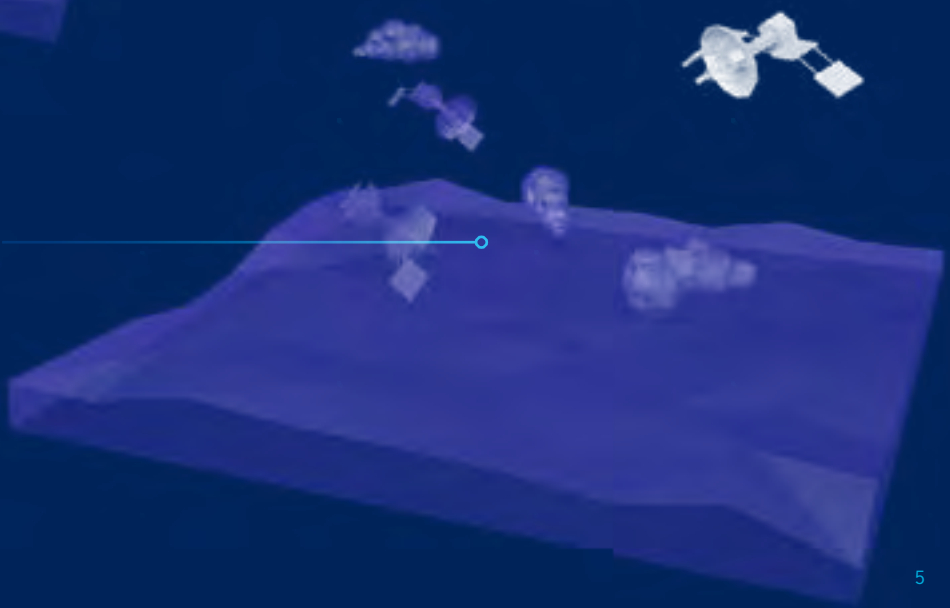
EVOLVE has developed key technologies to advance these features into a high maturity level (respective software components have also been made available as open-source projects), boosting pilots and Proof-of-Concepts (PoCs) significantly on many axes. After all, EVOLVE's innovation is driven by its real-life applications, provided by industrial partners from seven different domains, that make use of massive data and require complex processing:

Optimizing agri production yield using numerical models and massive historic data.



Maritime surveillance at scale and high accuracy, by using massive observation and domain-specific data (described in more detail in Section 3).

Radiometric correction and change detection on Sentinel-2 images.



Improvement of public bus services that dominate transportation in Europe.

Advanced vehicle routing algorithms and mobility services optimization.

Data-assisted automotive service development.

Automotive data-driven services for vehicle predictive maintenance.

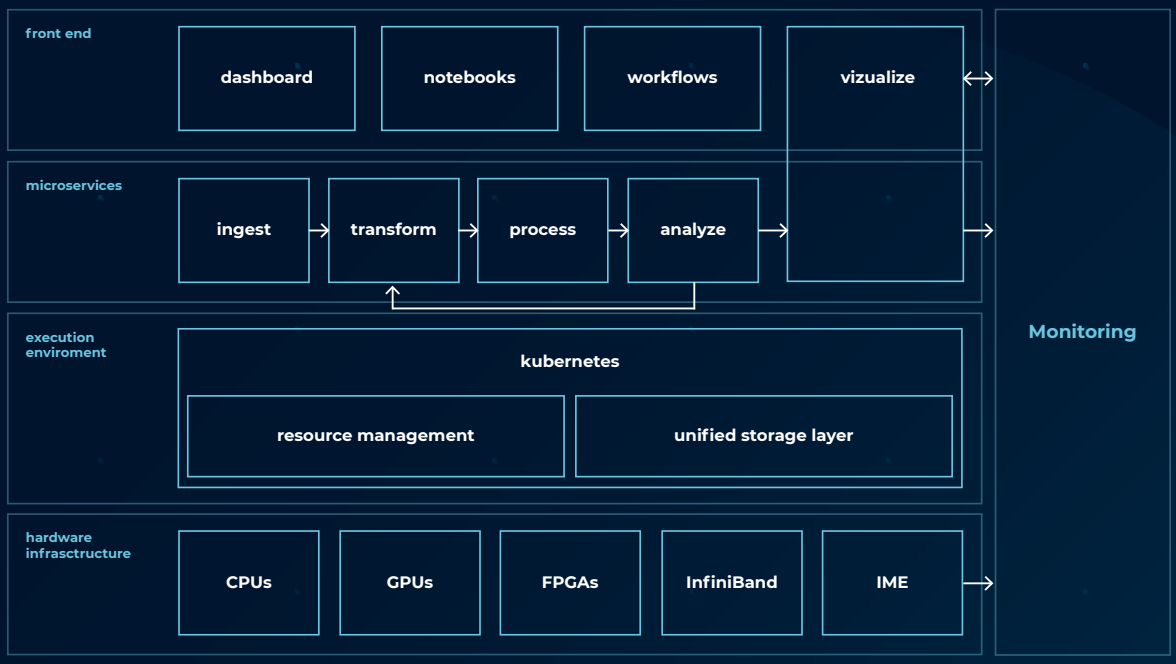
This crucial focus on applications allows EVOLVE to realistically set the bar for what is possible today and what should be the target, based on technology projections, in the future.

In addition, the project has enlisted more than a dozen external PoCs, i.e. applications which were not part of the original planning and which are now running on the EVOLVE testbed for improving their productivity, dataset sizes, as well as processing times. EVOLVE aims at building an ecosystem around HPC-enabled BD processing, to disseminate knowledge and possibilities, foster innovation, but also solicit feedback and ensure continuity.

The Evolve Platform

In the EVOLVE platform (Fig. 1), the heterogeneous Hardware infrastructure is integrated into a single Execution environment using Kubernetes [2]. As a scalable, container-based substrate, Kubernetes hides the complexity of managing massive numbers of compute nodes, and has thus become the industry-leading platform for supporting large-scale applications and complex processing pipelines dealing with enormous and diverse datasets.

The EVOLVE platform (Fig. 1)



To accomodate for the unique characteristics of EVOLVE pilots and PoCs, we have extended Kubernetes with novel workload placement and scheduling features, as well as a new storage abstraction, called the Unified Storage Layer (USL). We have also integrated a wide selection of BD, Cloud, and HPC software frameworks as Microservices, including a state-of-the art visualization service that extends to the Front end. A vertical, Monitoring layer collects runtime statistics from hardware and software components, to assist developers into understanding the performance characteristics of their applications, and to enable automated fine-tuning of container placement and scheduling.

The different layers of the EVOLVE infrastructure, as well as the EVOLVE-specific enhancements are briefly described in the following sections.

[2] Kubernetes: Production-Grade Container Orchestration. https://kubernetes.io

2.1 Hardware infrastructure

Needs for endless computing power has spurred architectural innovations in the pursuit of more compute capabilities. The current state-of-the-art in HPC is to create heterogeneous computing environments, by integrating accelerators – in the form of GPUs, FPGAs or ASICs – in the building blocks of the overall infrastructure. Following this direction, the EVOLVE hardware platform adopts the following elements:

CPUs (Intel Xeon), characterized by a variety of features including NUMA and multiple cores.

GPUs (NVIDIA Tesla K20, P40, V100), supporting several programming models, like CUDA, OpenMP, OpenAcc, and OpenCL.

FPGAs (Altera Arria 10, Stratix 10), used via native HDL (VHDL, Verilog) and Intel OpenCL SDK.

Overall, the HPC cluster used in EVOLVE, combines 10 identical compute nodes (each with 24 cores/48 threads and 128 GB of RAM), with a complementary set of 5 accelerator nodes with similar characteristics, that host the accelerators.

All nodes are interconnected via NVIDIA Mellanox InfiniBand FDR links (56 Gb/s) and run Linux. The InfiniBand backbone, which uses a fat-tree interconnect topology to minimize the number of hops between computing nodes, is employed for both storage and computing.

Storage to the cluster is provided by an external storage hierarchy, optimized both for capacity and performance. **For capacity, there is a 120TB Lustre [3] filesystem, implemented with 2 NetApp FAS2700 Series storage arrays.** To accelerate the path from compute nodes to storage devices, data is accessed through **DDN's Infinite Memory Engine (IME)** [4], a scale-out, software-defined, flash storage platform, that automatically optimizes and coordinates data movement from high-speed devices to the high-volume back end.

[3] Lustre. https://www.lustre.org/

[4] DDN. Infinite Memory Engine. https://www.ddn.com/products/ime-flash-native-data-cache/

2.2 Execution environment

2.2.1 Resource management extensions.

Kubernetes uses a discrete, pluggable service, the scheduler, to assign pods to compute nodes, based on well-defined policies. The software stack of EVOLVE includes a series of respective extensions, which we use to apply custom placement and scheduling decisions:

The Skynet

scheduler automatically calculates resources that a pod needs to meet a target performance. Unlike existing Kubernetes deployments, where users specify the amount of resources for their pods, Skynet expects users to enter target performance metrics and it uses a closed loop control mechanism to calculate an optimal resource allocation that allows pods to meet those targets. Skynet extends previous work [5], by dynamically adjusting application runtime profiles, based on metrics gathered during execution.

Volcano [6]

provides an abstraction of Kubernetes primitives to domain job schedulers of frameworks, such as Apache Spark and TensorFlow. Those primitives allow such frameworks to optimize their scheduling over Kubernetes for performance. In the back end, the scoring module of the Kubernetes scheduler has been re-designed to use Volcano primitives in its ranking of the candidate nodes for every pod request that arrives from Volcano.

The interference-aware custom scheduler [7]

is able to efficiently place applications on a cluster of heterogeneous machines. Using a universal approach for every kind of workload behaviour and duration, the custom scheduler aims to maximize resource utilization and minimize application execution delays provoked by interference phenomena. Compared to priorwork,we monitor low-level metrics, describing micro-architectural events, which are capable of providing useful information for resources under contention [8], thus pinpointing to the origin of a system's inability to serve the workloads efficiently.

All aforementioned practices outperform the default resource management schemes of Kubernetes, improving the performance of the scheduled workloads, by better balancing the usage of different components of the hardware platform.

[5] Y. Sfakianakis, C. Kozanitis, C. Kozyrakis, and A. Bilas. 2018. QuMan: Profile- Based Improvement of Cluster Utilization. ACM Trans. Archit. Code Optim. 15, 3, Article 27 (Aug. 2018), 25 pages. https://doi.org/10.1145/3210560

[6] Volcano: A Kubernetes Native Batch System. https://github.com/volcano-sh/volcano

[7] A. Tzenetopoulos, D. Masouros, S. Xydis, and D. Soudris. 2020. Interference-Aware Orchestration in Kubernetes. In High Performance Computing - ISC High Performance 2020 International Workshops, Frankfurt, Germany, June 21-25, 2020, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 12321), H. Jagode, H. Anzt, G. Juckeland, and H. Ltaief (Eds.). Springer, 321–330. https://doi.org/10.1007/978-3-030-59851-8_21

[8] D. Masouros, S. Xydis, and D. Soudris. 2021. Rusty: Runtime Interference-Aware Predictive Monitoring for Modern Multi-Tenant Systems. IEEE Trans. Parallel Distributed Syst. 32, 1 (2021), 184–198. https://doi.org/10.1109/TPDS.2020.3013948

2.2.2 Unified Storage Layer.

During EVOLVE development, we had to cope with conflicting storage abstractions provided by the available hardware and systems software, and respective assumptions used by different Cloud and Big Data tools and frameworks. Pipeline stages may use diverse software frameworks that deal with storage in a completely different manner. To tackle the heterogeneity of APIs and programming libraries to interact with data, we designed and implemented the *Unified Storage Layer* (USL) for Kubernetes.

The USL enables applications to be designed following the cloud-ready paradigm and to transparently benefit from potentially any storage solution available, ranging from high-performance file systems or key-value stores, to cloud based object storage. With the USL, workflow execution units are automatically provided with a broad range of endpoints, so that application developers can easily select where to store data depending on the unique storage characteristics of their application.

USL components are stand-alone open source software frameworks, whose integration creates a **unique software stack for data access in Kubernetes environments:**

- **Karvdash [9]: The EVOLVE dashboard serves as the USL front end, by providing a user interface for configuring Datashim and H3 storage attachments. Karvdash also wires up a private and shared dataset per user by default.**

- **Datashim [10]: The USL core, mounting the actual datasets to containers, thus unifying access to a diverse set of actual storage protocols and technologies.**

- **H3 [11]: An embedded object store library, backed by a highperformance key-value store. H3 can either be embedded into applications, or accessed through Datashim as part of the unified storage offering.**

USL simplifies workflows by completely replacing multiple data movement stages with a set of Dataset configuration directives (modelled as Kubernetes Custom Resource Definitions) that are defined as part of workflow initialization.

USL provides an abstract form for defining varying types of storage endpoints, and attaches the corresponding data collections as filesystem mounts inside containers, preserving the files-based data access pattern. These configuration directives can either be constructed by users manually, or setup through Karvdash.

The latter, as part of the default deployment, automatically configures a private "home" folder per user, as well as a cross-platform shared dataspace, and makes them accessible through a web-based front end (Fig. 2).

(Fig.2)
The **"Files"** tab in Karvdash allows users to directly interact with their **"home" datasets via their browser.**

[9] Karvdash: A dashboard service for facilitating data science on Kubernetes. https://github.com/CARV-ICS-FORTH/karvdash

[10] Datashim: A Kubernetes based framework for hassle free handling of datasets. https://github.com/datashim-io/datashim

[11] H3: An embedded object store. https://github.com/CARV-ICS-FORTH/H3

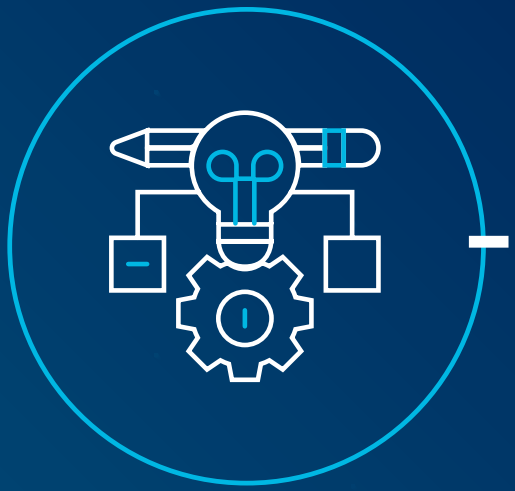
Datashim performs storage attachments by utilizing low-level, Kubernetes-compatible Container Storage Interface (CSI) plug-ins. Additionally, Datashim provides a set of caching plugins to speedup data-access.

Through an experimental evaluation we have demonstrated that colocation of data and computation can affect the data access rates and thus the overall performance of an application, especially for data-intensive workloads such as Spark jobs or training of neural networks.

To address this issue, we have implemented a cache-aware scheduling plugin for Volcano to place Datashim cache pods as close as possible to the respective Spark executors.

Proximity might not necessarily be with the actual disks that hold the data, but also with the gateway that Datashim uses for providing access to cached data. The matching is done by parsing Dataset information associated with a specific Spark job.

H3 is meant to ease application transition to high-performance key-value stores (like RocksDB [12], Redis [13], and Kreon [14]), by offering a simple object store API that translates respective commands to key-value operations, in order to decrease the latency of small data operations and facilitate the use of node-local storage devices and memory in workflow steps. H3 also implements a CSI plug-in for integration with Datashim, for allowing existing pipeline steps to seamlessly exploit the functionality without code changes.



With the seamless data abstraction provided by the USL, **the user experience of dealing with data collections becomes more familiar and friendly, and workflows become portable, simpler to develop and deploy, without sacrificing performance.**

[12] RocksDB. https://rocksdb.org/

[13] Redis. https://redis.io/

[14] A. Papagiannis, G. Saloustros, P. González-Férez, and A. Bilas. 2018. An Efficient Memory-Mapped Key-Value Store for Flash Storage. In Proceedings of the ACM Symposium on Cloud Computing (Carlsbad, CA, USA) (SoCC '18). 490–502. https://doi.org/10.1145/3267809.3267824

2.3 Microservices

Software frameworks, as runtime components for workflow execution, are packaged up in containers as microservices, which are then used as building blocks for workflow steps. Most of the microservices handle compute-intensive tasks: data ingest/extraction, flexible and efficient data transformations, data processing with integrated custom HPC kernels, and incremental one-pass analytics over bounded data streams.

During development of EVOLVE, we first identified state-of-the-art software frameworks to be included into the software stack, performed all necessary integration actions to make these frameworks easily accessible as workflow components, and demonstrated their applicability by providing example notebooks that use the high-level, user-facing software components of the Front end layer.

Special care has been taken so that provided microservices are compatible with the software components of other layers as well: the containerized executor instances of Kafka [15], Spark [16], TensorFlow [17], MPI [18], Dask [19], etc. preserve the user isolation/protection properties of the Execution framework (i.e., run in separate namespaces), use the custom Kubernetes resource management extensions where needed, and send performance metrics to Monitoring. More importantly, microservices exploit the HPC capabilities of the Hardware infrastructure:

Fast storage, accelerators, and the high-speed, low-latency network.

Especially for HPC, EVOLVE enables workflows to seamlessly incorporate MPI-based executables as processing stages, taking a concrete step towards integration of HPC and BD processing. We deploy on-demand virtual clusters — container-based HPC environments that allow MPI codes to run in Kubernetes, using the Slurm job manager [20] and a wide range of libraries and compilers, while maintaining direct access to the InfiniBand network present at physical nodes, as well as available GPUs and FPGAs. Each virtual cluster hosts a custom Slurm controller that communicates with Skynet to coordinate the actual placement of HPC jobs. Experimental results indicate that the virtual cluster construct retains the performance and scalability properties of the supporting physical infrastructure, while the unique resource allocation scheme guarantees (to a level) that HPC containers run unobstructed.

[15] Apache Kafka. https://kafka.apache.org
[16] Apache Spark. https://spark.apache.org
[17] TensorFlow. https://www.tensorflow.org
[18] MPI Forum. https://www.mpi-forum.org
[19] Dask: Scalable analytics in Python. https://dask.org
[20] Slurm Workload Manager. https://slurm.schedmd.com/documentation.html

2.4 Front end

2.4.1 User experience.

EVOLVE users interact with the platform through Karvdash — the EVOLVE dashboard — and implement their applications in notebooks, using workflows that interface with platform microservices and the visualization component of the software stack. Karvdash, is a service management software for Kubernetes, which runs in Kubernetes as a service itself. It provides a web-based graphical frontend to coordinate accesses to the platform, orchestrate service execution in containers from pre-defined templates — including Zeppelin notebooks [21], interact with collections of data that are automatically attached to application containers when launched (as part of the USL), and securely provision multiple services under one externally-accessible HTTPS endpoint. All dashboard resources are organized per-user, and each user is assigned to a corresponding, isolated Kubernetes namespace. Moreover, the dashboard provides web-based access to the private Docker registry for convenience.

In EVOLVE, we use Argo Workflows [22] as the workflow runtime. Additionally, we provide the evolve Python library [23], which allows composing workflows as code. One of the most significant features of Zeppelin is that it provides the capability to build custom interpreters. Thus, we have implemented a custom EVOLVE interpreter, for simplifying the definition and deployment of workflows with Argo.

The EVOLVE custom interpreter is based on Zeppelin's Python interpreter and embeds the evolve library. This allows the programmatic handling of workflows alongside other management and data manipulation tasks through the same Zeppelin paragraphs.

[21] Apache Zeppelin: Web-based notebook that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala and more. https://zeppelin.apache.org

[22] Argo Workflows. https://argoproj.github.io/projects/argo

[23] The EVOLVE Python library. https://bitbucket.org/sunlightio/evolve_python_library/

2.4.2 Visualization services.

EVOLVE develops high-performance visualization services that help analysts navigate big data repositories across multiple metadata dimensions. These services are intended to complement but not compete with process-oriented monitoring and visualization frameworks such as Grafana [24] and Kibana [25]. The goal is to render results of computational processes together with relevant knowledge automatically extracted from the public debate, in the form of digital content streams from news sites, social media platforms, etc. To provide the desired contextualization, the services not only render the results, but also perform complex aggregation, filtering, indexing, natural language processing, metadata enrichment, translation and knowledge graph alignment processes in the background. When implementing this ontextualization process, special emphasis has been placed on temporal and geographic dimensions, both in terms of rendering statistical data ingested through a REST API as well as the ability to put results in the context of stakeholder communication (by means of aligning metadata attributes, e.g. the geographic location).

Building upon earlier work of webLyzard in regard to visual analytics dashboards for big data applications [26], [27], [28], EVOLVE focuses on improving the visualization engine in line with the pilot requirements, including the automated extraction of pilot-specific metadata. There are two main front end components, a resultsfocused Visual Analytics Dashboard (VAD) (Fig. 3) and a processfocused Execution Dashboard (ED). On the back end, contextualizing data with external sources requires a tightly-knit infrastructure of services that bring together the required functionalities of data acquisition, data alignment and data visualization. Additionally, the on-demand nature of live documents for data computation as inherent to the EVOLVE notebooks requires repository management for fully automated creation and cleanup of data repositories. Ultimately, the goal is to allow external users to unlock the storytelling potential of the microservices with minimal effort, as a single RESTful API endpoint. To the notebook user, requesting a contextualized visualization is encapsulated in a single API request for ease of use.

[24] Grafana: The open observability platform. https://www.grafana.com/

[25] Kibana: Explore, Visualize, Discover Data. https://www.elastic.co/kibana/

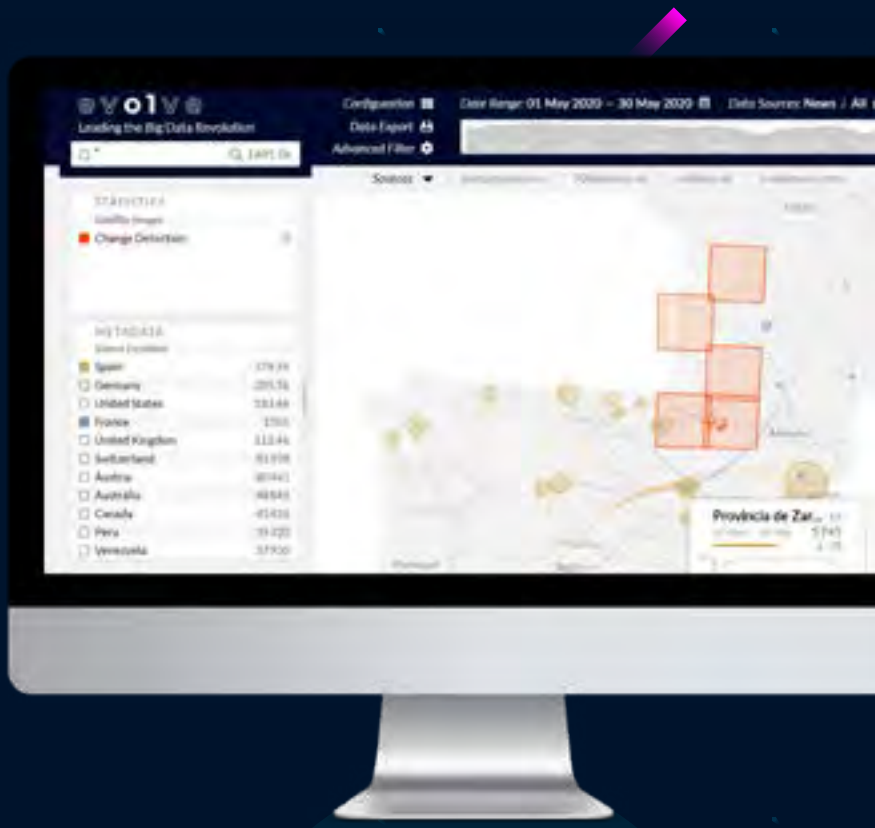
[26] A. Scharl, A. Weichselbraun, M. Göbel, W. Rafelsberger, and R. Kamolov. 2016. Scalable Knowledge Extraction and Visualization for Web Intelligence. In 2016 49th Hawaii International Conference on System Sciences (HICSS). 3749–3757. https://doi.org/10.1109/HICSS.2016.467

[27] A. Scharl, D. Herring, W. Rafelsberger, A. Hubmann-Haidvogel, R. Kamolov, D. Fischl, M. Föls, and A. Weichselbraun. 2017. Semantic Systems and Visual Tools to Support Environmental Communication. IEEE Systems Journal 11, 2 (June 2017), 762–771. https://doi.org/10.1109/JSYST.2015.2466439

[28] A. Brasoveanu, M. Sabou, A. Scharl, A. Hubmann-Haidvogel, and D. Fischl. 2016. Visualizing statistical linked knowledge for decision support. Semantic Web 8 (11 2016), 113–137. https://doi.org/10.3233/SW-160225

(Fig.3)

The “Files” tab in Karvdash allows users to directly interact with their “home” datasets via their browser.



2.5 Monitoring

Detailed observation is instrumental in any experimental process, and a critical part in any product design cycle. Therefore, as an innovation testbed, the EVOLVE platform includes a vertical Monitoring layer to both stir the design, and to help end-users exploit the system in an efficient way. We focus monitoring on three axes:

- Processors, including node-level CPUs and accelerators;
- Software stacks, with individual probes on different software middlewares used;
- Storage, as it is critical to the performance of application level workflows.

We use standard open-source solutions for end-to-end monitoring of both the hardware and software stack and make the results available via a graphical interface, or via API calls for other purposes (i.e., placement and scheduling decisions). Also, monitoring is not limited to depicting the current behavior; the EVOLVE platform also supports the observation of failures and failure recovery.

Conclusion

HPC, Cloud and Big Data share a common point: the need for performance at scale. However, they differentiate by many aspects, most importantly the programming workflow and software stack. Their convergence has appeared as an elusive if not impossible promise; they are commonly thought of as distinct technologies remaining distant, frozen and immobile in their respective markets. The EVOLVE project challenges — in a very pragmatic way — the idea that a single computing infrastructure can integrate the best of all worlds; and is actually already pushing things in that direction, **by building an heterogeneous computing platform in conjunction with the software components, that enables applications to scale out in a flexible, transparent, and portable manner.**

evolve

Leading the Big Data Revolution

@evolve_h2020

www.evolve-h2020.eu

European Union's Horizon 2020 research and innovation programme under grant agreement No 825061