EVOLVE: Towards Converging Big-Data, High-Performance and Cloud-Computing Worlds

Achilleas Tzenetopoulos*, Dimosthenis Masouros*, Konstantina Koliogeorgi*, Sotirios Xydis*†, Dimitrios Soudris*, Antony Chazapis‡, Christos Kozanitis‡, Angelos Bilas‡, Christian Pinto[l], Huy-Nam Nguyen§, Stelios Louloudakis††, Georgios Gardikis‡‡, George Vamvakas‡‡, Michelle Aubrun¶¶, Christy Symeonidou§§, Vassilis Spitadakis§§, Konstantinos Xylogiannopoulos‖, Bernhard Peischl‖, Tahir Emre Kalayci**, Alexander Stocker**, Jean-Thomas Acquaviva¶,

*Institute of Communication and Computer Systems (ICCS), Athens, Greece
†Department of Informatics and Telematics (DIT), Harokopio University of Athens (HUA), Greece
‡Institute of Computer Science, FORTH, Heraklion, Greece, [l]IBM Research Europe, Dublin, Ireland
§ATOS/Bull, Paris, France, ††Sunlight.io, Heraklion, Greece, ‡‡Space Hellas S.A., Athens, Greece
¶¶Thales Alenia Space, Toulouse, France, §§NEUROCOM Luxembourg, Luxembourg, ‖AVL List GmbH, Graz, Austria
**Virtual Vehicle Research GmbH, Graz, Austria, ¶DataDirect Networks, Paris, France

***Abstract*—EVOLVE is a pan European Innovation Action that aims to fully-integrate High-Performance-Computing (HPC) hardware with state-of-the-art software technologies under a unique testbed, that enables the convergence of HPC, Cloud and Big-Data worlds and increases our ability to extract value from massive and demanding datasets. EVOLVE's advanced compute platform combines HPC-enabled capabilities, with transparent deployment in high abstraction level, and a versatile Big-Data processing stack for end-to-end workflows. Hence, domain experts have the potential to improve substantially the efficiency of existing services or introduce new models in the respective domains, e.g., automotive services, bus transportation, maritime surveillance and others. In this paper, we describe EVOLVE's testbed, and evaluate the performance of the integrated pilots from different domains.**

***Index Terms*—HPC, Cloud-computing, Big-Data, computing platform, accelerators, interference, resource orchestration**

I. Introduction

As data becomes the center of innovation in modern economy and society, organizations face new challenges and limitations. Although tremendous progress has happened over the last years on increasing productivity for data processing over commodity systems and providing new services with Big Data 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?*

Today, creating new data-intensive services in terms of dataset size and data processing is an onerous and costly process that requires deep expertise, i.e., high performance hardware, complex software stacks, and dedicated per-application testbeds, to achieve the desired performance levels. However, most organizations, and especially small- and medium-sized enterprises, lack those resources and the required skills. EVOLVE is a pan European Innovation Action that aims to build a large-scale testbed by integrating technology from the HPC, Big Data and Cloud worlds [1]. EVOLVE builds and demonstrates the proposed platform with real-life, massive datasets from demanding application areas, e.g., automotive services, maritime surveillance and others.

This work is funded by the EU Horizon 2020 research and innovation programme, under project EVOLVE, grant agreement No 825061.

More specifically, EVOLVE integrates an advanced, HPC-enabled computing platform capable of processing massive and demanding datasets, enriching Big-Data processing with HPC features including acceleration, large memories, fast storage architectures and fast interconnects. It also enables applications to express their logic and datasets in the form of workflows that can be automated, shared, refined and maintained across groups of domain experts without significant IT expertise. Moreover, EVOLVE provides a rich and versatile Big-Data processing software stack that can execute automated workflows by using popular components from the Big-Data world as workflow stages. EVOLVE, using a Cloud-based approach, provides support for sharing a unique testbed across applications, facilitating deployment, access and use. Finally, it offers a capable testbed, integrating the project technologies, for use by ecosystem stakeholders and interested parties.

In this paper, we walk through the final version of EVOLVE's converged testbed. We first discuss and evaluate its hardware stack based on widely-used benchmark suites. Then, we present the core software technologies and their improved versions adopted, for transparent deployment, optimization and management. Various integrated pilots are then evaluated, showing its ability to provide increased performance efficiency, achieving average performance improvements of 26.3× and up to 114×.

The rest of the paper is organized as follows. In Sec. II, we present our platform, by describing and evaluating aspects of the hardware stack that enable the convergence of HPC, Cloud and Big-Data. In Sec. III, we describe the system- and software-level technologies that enable the efficient, yet abstract exploitation of the aforementioned hardware resources, and finally in Sec. IV, we demonstrate the integration of several

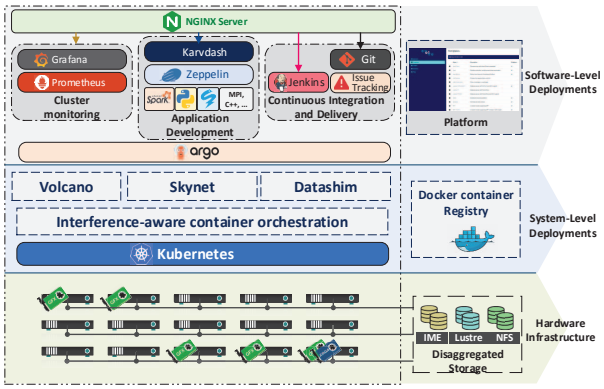


Fig. 1: Overview of EVOLVE platform

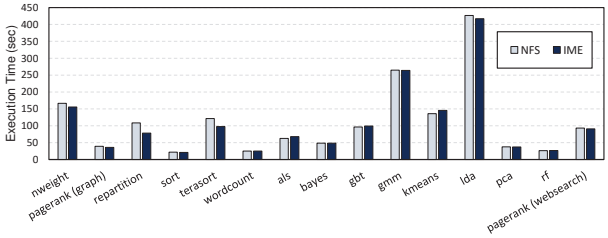


Fig. 2: Storage Capabilities of EVOLVE's HW infrastructure over HiBench benchmark suite

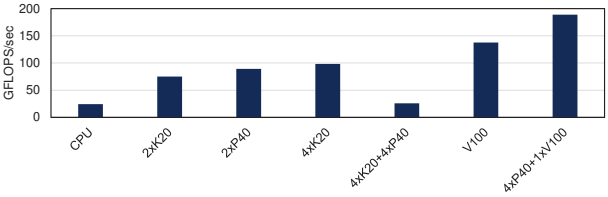


Fig. 3: Computing Capacity of EVOLVE's HW infrastructure over HPCG benchmarks (CPU+GPU)

pilots from different domains and evaluate their performance when deployed over EVOLVE.

II. Evolve's Hardware Stack

EVOLVE provides a high-performance hardware infrastructure, complemented by the tightly integrated system and software stacks. Figure 1 summarizes the integrated technologies of EVOLVE's testbed, ranging from the hardware infrastructure, to system-level and software-level deployments.

Compute Subsystem: Overall, the HPC cluster used in EVOLVE, combines 16 heterogeneous Intel x86 compute nodes (Broadwell, Haswell and Skylake families), with various accelerators and storage technologies installed, as described in Table I. Accelerators, i.e., GPUs and FPGAs, are installed in six of those nodes, while all nodes are interconnected via NVIDIA Mellanox InfiniBand FDR links (56 Gb/s) and run Linux.

Storage Subsystem: The platform uses different types of storage: Network File System serves the home directories for convenience, while Lustre and IME [2] are integrated for high performance I/O operations. IME offers high performance, allowing I/O intensive operations with a focus on bandwidth and large number of I/O data requests. The high degree of parallelism coupled with fast flash devices is delivering the highest level of performance. IME acts as a burst buffer to the Lustre storage pool, which offers the actual capacity.

Computing capability: To evaluate the performance capabilities of EVOLVE's HW infrastructure, we utilize different open-source and in-house developed benchmarks. Specifically, for Big-Data workloads, we use the HiBench benchmarking suite [3], which provides a set of Spark, in-memory applications. Regarding HPC workloads, we use the High-Performance Conjugate Gradients (HPCG) [4] benchmark, which supports MPI, OpenMP and CUDA and is designed to exercise computational and data access patterns which find broad usage in modern Machine Learning and Deep Learning workloads. Last, to test the acceleration capabilities of the platform, we have developed two in-house versions of VGG16 and ResNet50 inference models for our target devices [5].

Figures 2 and 3 and Tab. II show the respective performance results. In Fig. 2 we demonstrate the execution time of different data-intensive benchmarks, when they utilize IME and NFS storage subsystems. As shown, for the majority of them, IME storage provides an average of 5% performance improvement, which is expected to be furtherly improved for concurrent and co-located deployment scenarios. Regarding the computing capacity of the EVOLVE stack, Fig. 3 evaluates the GFLOPS per second achieved, for varied infrastructure compositions. We can clearly see that, GPU accelerators greatly improve the computing capacity of cluster, delivering up to 4x higher throughput (GFLOPS/sec) compared to the CPU-only version. Finally, Tab. II shows the execution time of two CNNs for different implementations and target devices installed in our HW infrastructure. As we can see, the use of GPUs and FPGAs yield speedups that range from 2× up to 11×.

TABLE I: Advanced Computing Platform

CPUs	Intel Xeon {Platinum 8153, E5-2690/2670/2470}
GPUs	Nvidia {Tesla V100, P40, K20Xm}
FPGAs	Altera Arria 10, Stratix 10
Storage	IME, Lustre, NFS

TABLE II: Evaluation of HW accelerators on CNN models.

Platform	**Execution time (ms)** VGG16	ResNet50
TF Intel Xeon Platinum 8153	1067.26	1162.79
TF Nvidia K20Xm	1095.19	410.25
TF Nvidia P40	608.35	187.71
TF Nvidia V100	318.24	143.83
OpenCL Stratix10	103.1	101.21

III. Software Stack of EVOLVE

EVOLVE provides two levels of software stack deployments, i.e., system-level deployments, which aim to provide transparent optimized execution and storage capabilities across the platform and software-level deployments, which expose user-friendly endpoints to end-users.

System-level Deployments: Our platform uses the Kubernetes orchestrator, for seamless, containerized application deployment. Our SW stack extends the resource management scheme of Kubernetes through the integration of Volcano [6] for batch scheduling, and the development of two custom Kubernetes schedulers. First, our interference-aware scheduler [7], places incoming applications on the pool of available resources. To do so, it prioritizes nodes based on the resource contention they experience [8], [9]. Second, Skynet [10] provides real-time resource tuning, that dynamically adjusts application runtime profiles and resource allocation, based on metrics gathered during execution and user-specified target performance metrics. Moreover, to cope with conflicting storage abstractions provided by the available hardware and tackle the heterogeneity of APIs and programming libraries interacting with data, we designed and implemented the Unified Storage Layer (USL) [11]. Datashim [12], the core of USL, mounts the actual datasets to containers, thus unifying access to a diverse set of actual storage protocols and technologies. Finalt, H3 [13] is a high-performance object store that can be used by applications directly (embedded as a library), or mounted as a USL dataset through a FUSE-based compatibility layer.

Software Level Deployments: 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, while some implement supporting services. Microservices include: Kafka, Spark with auto-tuning capabilities [14], TensorFlow, MPI, Dask, Promethes and Grafana for monitoring, as well as CI/CD services, such as Jenkins and Git repository. Users interact with the platform through Karvdash [15] — 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 allows to orchestrate service execution in containers from pre-defined templates, interact with collections of data by utilizing USL, and securely provisions multiple services under one externally-accessible endpoint.

Especially for HPC, EVOLVE enables workflows to seamlessly incorporate MPI-based executables as processing stages, taking a concrete step towards integration of HPC and Big-Data processing. We deploy on-demand virtual clusters — container-based HPC environments that allow MPI codes to run in Kubernetes, using the Slurm job manager and a wide range of libraries and compilers, while maintaining direct access to the InfiniBand network present at physical nodes, as well as the available GPUs and FPGAs. Each virtual cluster hosts a custom Slurm controller that communicates with Skynet to coordinate the actual placement of HPC jobs.

TABLE III: Technologies used per use-case

Technologies/Pilots	❶	❷	❸	❹	❺
IME Fast Storage	✓	✓	✓	✓	✓
GPU/FPGA Acceleration	✓	✓	✓	✓	✓
Resource- & Interference-aware Orchestration		✓		✓	
Big Data Processing		✓	✓		
Scale-out parallel frameworks	✓	✓	✓	✓	✓
One-Pass stream engines			✓	✓	
Prometheus/Grafana Monitoring	✓			✓	

IV. Pilots' Workflows deployment & Evaluation

EVOLVE testbed is evaluated through a diverse set of use cases. In this paper, we focus on five application domains, detailed through ❶ - ❺ in Section IV-A. Table III shows the mapping between the examined use-cases and the corresponding EVOLVE technologies utilized, as reported in Section III.

A. Pilot Description and Evaluation

❶ Object Detection using Knowledge Graph-based Data Integration for Automated Driving: This use-case implements computer vision methods to detect the states of two driver assistance systems, lane assist and adaptive cruise control, using vehicle dashboard videos, in order to assess the acceptance of automated driving. Using a knowledge graph-based data integration framework [16], we deploy an Object detection proof-of-concept. A knowledge graph is used to combine data stored in separate data sources into a single, unified view [17]. We apply the framework to the widely known `Motional` and `Lyft` datasets to evaluate the benefits of using the high-performance integrated computing platform provided by EVOLVE. The services were developed and deployed by exploiting the integrated USL supported by Karvdash and Datashim. Argo workflows are also used, in order to achieve seamless parallel execution in the context of Kubernetes.

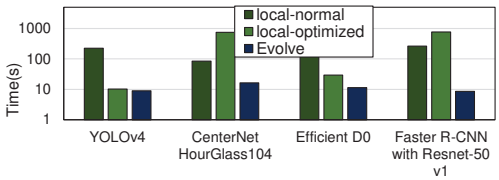


Fig. 4: Object detection models execution time comparison

Evaluation: Through the EVOLVE platform, we are able to create 6,538,057 nodes in 2101.52 seconds for Motional, and 1,047,118 nodes in 445.6 seconds for Lyft, respectively. In comparison to 82,454 nodes in 22.7 seconds and 9,442 nodes in 4.14 seconds delivered by the pre-EVOLVE infrastructures, EVOLVE allowed the exploitation of much larger knowledge graphs leading to more efficient object detection models. Moreover, Fig. 4 compares the sequential execution of different object detection models with the parallelized execution on EVOLVE platform for 1000 images. More specifically, we

plot the normal and optimized local, sequential versions, next to EVOLVE's parallel one. EVOLVE's infrastructure provides up to 30x speedup for the Faster R-CNN with Resnet-50v1 and 5x in the case of CenterNet HourGlass104 respectively.

❷ **Improvement of bus transportation service using observation and historic operational data:** This use-case aims to evaluate the quality of service of buses on a continuous basis based on post-analysis activity (Non Real Time context) and current traffic congestion on public transport network (Real Time context). To do so, planned data are being correlated with actual data, including data that correspond to framing conditions (traffic, weather, demand). Concerning the *Non-Real-Time (NRT)* context, the analysis needs to be applied to longer time periods, visualised in many different ways within acceptable response times. On the other hand, the case of *Real-time (RT)* context concerns the identification of cases where delay is noticed and the corresponding visualisation. In both cases large datasets need to be ingested into the system, transformed and visualised efficiently.

The end-to-end pipeline from data ingestion, to Spark processing and the visualisation has been integrated to the EVOLVE platform. The need was to broaden the scope of the batch data analysis that the authority and operator employs within its decision and planning procedures, while at the same time, become able to get real-time analysis of bus event data streamed from the network. Hence, two different workflows are considered with different setups:

- The Offline activity workflow that aimed to improve the planned public transport (PT) services both from the Operator and the Authority point of view based on certificated and validated data collected from the ITS systems directly managed by the PT companies and/or authorities.
- The Real time activities (RT) that aimed to give information about the traffic congestion on the transport network at the time it happens. These data are useful both for the authority and operator in order to perform real time monitoring activity and to provide the end-users with information about the best travel to plan.

The two data workflows are exploiting IME, Spark in conjunction with Spark streaming engine, and Apache Zeppelin over EVOLVE to improve decision support and active monitoring in public transport operations.

Evaluation: Through EVOLVE, we managed to process and visualize simultaneously 21k of arcs in under 20 seconds. This enabled to i) increase the time period covered by the analysis, from one-month data (220 bus shifts) to a period longer than one year. ii) Speedup the listing of trips for two days of service, iii) decrease the latency for daily queries that summarize transits time, iv) complete the visualization procedure of the real-time status of two to three selected lines/routes in less than 6 seconds. Finally, it is now possible to visualize the traffic congestion on the map of a PT road network, with the possibility to highlight the congested road sections.

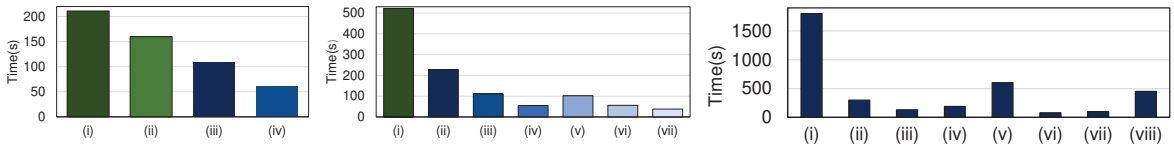
❸ **Maritime surveillance using observation data, historic metadata, and classification models**: Maritime Surveillance has been identified as of top priority at EU level. This challenge is usually addressed by correlating vessel information from SAR (Synthetic Aperture Radar) images with AIS/VHF data. The compute-intensive processing required, imposes today long response time for this approach, which degrades its operational value.

Plain parallelization and distributed computing only partially solves the problem, due to the inherent limitations associated with partitioning the satellite image across multiple compute nodes; HPC and hardware acceleration needs to kick in, towards yielding better results. The aim of the maritime surveillance workflow is to streamline and automate the processes needed to integrate and augment the maritime situational pictures. The workflow includes the following stages: AIS (Automated Identification System) data acquisition, AIS anomaly detection, SAR image acquisition, SAR filtering/preprocessing and vessel detection, AIS/SAR fusion correlation and, finally, visualization. All stages of the workflow are deployed in the EVOLVE platform and are being managed through the Zeppelin front-end. For automating the detection of vessels and any anomalies, that may exist, Argo Workflows are utilized. Two SQL queries are created for filtering the satellite images and AIS data based on location and time, for the two parallel workflows to run.

Select time window and execute Vessel Detection Algorithm: The first query returns the satellite images (SAR) that match the user-defined criteria. Note, that there are two cron-jobs that run periodically and download SAR images (to a persistent volume on EVOLVE Kubernetes cluster) while also storing each image's metadata, e.g., time, geolocation, to a Relational Database. Having the query been executed successfully, for every image found, a new pod is triggered on EVOLVE platform and each one in parallel searches for vessels in the image, saves the result and converts it to GeoJson format for visualization.

Anomaly detection in vessel trajectories based on AIS data: In parallel, another workflow is running to detect anomalies in vessel trajectories based on real AIS data as transmitted by each vessel. The extracted AIS data, are saved to a user-specified location across the USL, and then they are pre-processed, and used for inference by the anomaly detection model. The records that are classified as anomalies are visualized and warnings for unexpected vessels' positions are produced.

Evaluation: Firstly, we evaluated the time required for processing, i.e., vessel detection and classification, in one SAR scene. This was calculated as the average processing time for ten SAR scenes to be processed consecutively. We evaluated four alternative incremental deployment scenarios: i) legacy system with serial processing; ii) Argo workflow with serial processing, on-premises; iii) Argo workflow with serial processing, iv) EVOLVE cluster; v) Argo workflow with parallel processing, EVOLVE cluster. Fig. 5a depicts the improvement in processing time across the four scenarios. Overall, compared to the legacy configuration, the EVOLVE technologies and infrastructure have so far resulted in a considerable (71%) reduction in the processing time, thanks to coarse-grained parallelization.



(a) SAR scene processing (b) Anomaly detection based on AIS data (c) Change detection

Fig. 5: Maritime surveillance (a), (b) and Change detection on satelite images (c). Indexes (i)-(viii) refer to the mentioned differing configuration scenarios.

We also measure the training time for the AIS anomaly detection algorithm, which is used to detect suspicious behaviours in vessel trajectories. The anomaly detection algorithm used has the structure of an Autoencoder, consisting of 6 Convolutional Layers (3 CNNs for the Encoder and 3 CNNs for the Decoder). This algorithm has been trained for 500 epochs, with batch size equal to 4096. The training set has around 3.7 million records, 10% of which used as validation set. After some preprocessing 22 columns have been used for training. Figure 5b below shows the improvement in training time across all different deployments. The evaluated scenarios are i) legacy system, ii) EVOLVE CPU, iii) EVOLVE Spark, iv) EVOLVE Spark-GPU, v) GPU-K20Xm, vi) GPU P40 and vii) GPU Tesla V100. Overall, compared to the first scenario (single CPU in a single machine), the EVOLVE cluster infrastructure has so far resulted in a considerable reduction in the training time equal to 66.3% in worst case, when only CPU is used in EVOLVE platform, achieving up to 92.7% in the best case scenario using a very powerful GPU (Tesla V100).

Finally, we evaluate the accuracy of the AIS anomaly detection. For this evaluation, the architecture of the Autoencoder has been changed. Leveraging the huge memory capacity of EVOLVE's integrated cluster, we increased the batch from 4,096 to 204,800 AIS records and the five-layer Conv2D architecture from 32x16x8x16x32 to 512x256x128x256x512. While the former model resulted to 84% accuracy and 0.1 loss, with the updated configuration deployed in EVOLVE platform, we increased the accuracy to 89% with the same loss.

❹ **Radiometric correction and change detection on satellite images:** With the advent of the Copernicus program and its wealth of open data, the Earth Observation domain is increasingly adopting Big-Data technologies. The advent of efficient data storage, and processing infrastructures, enabled the development of applications that allow the automatic or semi-automatic analysis of large volume of Earth Observation data, like pattern identification from high geographical scale or long-term trends extraction from multi-temporal datasets. An important challenge in multi-temporal change detection is fast access and storage of a big amount of data and the computationally-intensive processing which is necessary. In this context, the use of the EVOLVE platform, which contains HPC features, has been extremely useful. The objective of this pilot consists in detecting changes over the entire Europe during one year with a revisit period of ten days. Thus, this *Change Detection* application requires to access and process about 35,000 tiles (20TB of storage) of *Sentinel-2 data*.

Evaluation: We evaluated 8 different versions of this application: i) CPU-only, ii) CPU-only using TensorFlow(TF), iii) GPU and TF, iv) DASK, v) DASK and GPU, vi)KAFKA and DASK in a small dataset, vii) KAFKA and GPU in small dataset, viii) KAFKA and DASK in a big dataset. DASK and Kafka technologies have allowed to automate the pipeline and have parallel computation between different modules. Argo contributed to the orchestration of the Kubernetes cluster, while the monitoring services and the resources visualization, enabled by Prometheus and Grafana respectively, provided useful insights regarding resource usage. Figure 5c shows that 1) DASK scheduler reduces the computation time by a factor of ten, 2) GPU accelerators reduce also the computation time significantly, 3) the combination of DASK with GPU induced non-negligible overhead in the latency, and 4) the combination of KAFKA stream engines with DASK also reduces the mean computing time per change detection map.

❺ **Data-assisted automotive service development:** Time series pattern detection is illustrated using data for the condition monitoring of large engines. In this case, the cylinder pressure curve is an important source providing time series data. It contains information about all components and its condition. In combination with other engine operation data and by using dedicated algorithms operating on time-series data, it is possible to detect numerous fault symptoms and respective root cause information. Pattern detection and matching is among the most critical functions taking place within the necessary workflow of time-series measurements. Patterns are correlated to numerous patterns that characterize normal behaviours, before a specific pattern can be classified as normal or as problematic. This is a highly consuming process in terms of computational resources, while the response is required to be provided within a second.

There are two use-cases examined in this pilot. The first one is a computational intensive application that executes time series analysis for anomaly detection. Time series from many different channels are accumulated, pre-processed, transformed and finally a pattern detection algorithm compares each one with a reference curve that defines the normal behavior for each channel. The other one is a data volume intensive application, that runs over large batches of automotive measurements data and executes mainly descriptive statistics.

Evaluation: By utilizing Argo workflows for seamless parallelization and deployment of the application over Kubernetes, we managed to improve performance significantly as depicted in Fig. 6 in the first use-case, and by a factor of 40-50% in the second. However, the utilization of GPU did not provide

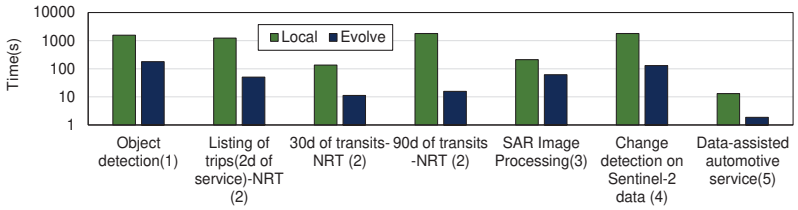


Fig. 6: Legacy and EVOLVE platform comparison on various use-cases

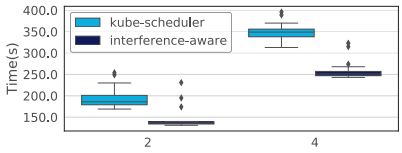


Fig. 7: Interference-aware orchestrator's impact on latency distribution

any improvement on the execution time. The GPU implementation was on small parts of code, which perform matrices multiplication, and that part had insignificant contribution to the overall execution time. Indeed, utilizing the Prometheus profiling services, we show that the specific function contributes already less than 1% of the total execution time.

B. Overall impact of the integrated testbed

EVOLVE platform has either enabled optimizations in several use-cases' development or significantly improved their performance. EVOLVE technologies achieved an average latency improvement of 26.3× related to the legacy deployments.

More specifically, for the latency of 100 images in the object detection pilot, as illustrated in Fig. 6, we achieved a 8.73× speedup w.r.t the legacy system deployment. Afterwards, in Fig. 6 are presented the execution times of three different stages of the Public Transport workflow ❷, i.e., listing of two days of service (24.7× speedup), 30- and 90-day transits (12× and 114× speedup respectively). The SAR image processing for the maritime surveillance pilot was improved by 3.45×, while the change detection on sentinel-2 data presented decreased latency by 13.84×. Finally, regarding the automotive service pilot (❺), while the per cylinder latency is higher due to containerization-induced delays, we achieved a 7× speedup in total.

Finally, EVOLVE retains its efficiency under real-life heavy-loaded scenarios. Fig. 7 shows the impact of interference-aware scheduler on pilots ❷ and ❹, when various levels of resource interference are applied on EVOLVE nodes. As Fig. 7 shows, EVOLVE's orchestrator, identifies and prevents co-locating jobs on contended nodes; thus achieving higher performance (26% on average) compared to the Kubernetes' default scheduler, which also results to proportionally reduced energy consumption.

V. Conclusion

EVOLVE is an multi-partner collaborative initiative that brings us closer to the convergence of HPC, Cloud computing and Big-Data worlds. The advanced compute platform in conjunction with the system- and software-level technologies implemented in the project, contribute to the closely-integrated EVOLVE's testbed. Thus HPC-capabilities, transparency, portability and large-scale data analytics have enabled pilots from various domains to build, port, deploy and optimize their use-cases, on the converged platform. Through adopting EVOLVE technologies, we achieve improved performance figures of 26.3× on average and up to 114× across the whole set of the examined use cases.

References

[1] A. Chazapis, J.-T. Acquaviva, A. Bilas, G. Gardikis, C. Kozanitis, S. Louloudakis, H.-N. Nguyen, C. Pinto, A. Scharl, and D. Soudris, "Evolve: Hpc and cloud enhanced testbed for extracting value from large-scale diverse data," in *Proceedings of the 18th ACM International Conference on Computing Frontiers*, 2021, pp. 200–205.

[2] "Ddn. infinite memory engine," https://www.ddn.com/products/ime-flash-native-data-cache/.

[3] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The hibench benchmark suite: Characterization of the mapreduce-based data analysis," in *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*. IEEE, 2010, pp. 41–51.

[4] J. Dongarra, M. A. Heroux, and P. Luszczek, "Hpcg benchmark: a new metric for ranking high performance computing systems," *Knoxville, Tennessee*, p. 42, 2015.

[5] K. Koliogeorgi, F. E. Keddous, D. Masouros, A. Chazapis, M. Aubrun, S. Xydis, A. Bilas, R. Hugues, J.-T. Acquaviva, H. N. Nguyen *et al.*, "Fpga acceleration in evolve's converged cloud-hpc infrastructure," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2021, pp. 376–377.

[6] "A kubernetes native batch system," https://github.com/volcano-sh/volcano.

[7] A. Tzenetopoulos, D. Masouros, S. Xydis, and D. Soudris, "Interference-aware orchestration in kubernetes," in *International Conference on High Performance Computing*. Springer, 2020, pp. 321–330.

[8] ——, "Interference-aware workload placement for improving latency distribution of converged hpc/big data cloud infrastructures."

[9] D. Masouros, S. Xydis, and D. Soudris, "Rusty: Runtime interference-aware predictive monitoring for modern multi-tenant systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 184–198, 2020.

[10] Y. Sfakianakis, C. Kozanitis, C. Kozyrakis, and A. Bilas, "Quman: Profile-based improvement of cluster utilization," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 3, pp. 1–25, 2018.

[11] A. Chazapis, C. Pinto, Y. Gkoufas, C. Kozanitis, and A. Bilas, "A unified storage layer for supporting distributed workflows in kubernetes," in *Proceedings of the Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems*, 2021, pp. 1–9.

[12] "Datashim: A kubernetes based framework for hassle free handling of datasets," https://github.com/datashim-io/datashim.

[13] A. Chazapis, E. Politis, G. Kalaentzis, C. Kozanitis, and A. Bilas, "H3: An application-level, low-overhead object store," in *High Performance Computing*, H. Jagode, H. Anzt, H. Ltaief, and P. Luszczek, Eds. Cham: Springer International Publishing, 2021, pp. 174–188.

[14] D. Nikitopoulou, D. Masouros, S. Xydis, and D. Soudris, "Performance analysis and auto-tuning for spark in-memory analytics," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 76–81.

[15] "Karvdash: A dashboard service for facilitating data science on kubernetes," https://github.com/CARV-ICS-FORTH/karvdash.

[16] T. E. Kalaycı, B. Bricelj, M. Lah, F. Pichler, M. K. Scharrer, and J. Rubeša-Zrim, "A knowledge graph-based data integration framework applied to battery data management," *Sustainability*, vol. 13, no. 3, p. 1583, 2021.

[17] T. E. Kalayci, E. G. Kalayci, G. Lechner, N. Neuhuber, M. Spitzer, E. Westermeier, and A. Stocker, "Triangulated investigation of trust in automated driving: Challenges and solution approaches for data integration," *Journal of Industrial Information Integration*, vol. 21, p. 100186, 2021.