evolve

Leading the Big Data Revolution

The EVOLVE Pilots

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

Vassilis Spitadakis
NEUROCOM LUXEMBOURG S.A.

Jean-Thomas Acquaviva
DataDirect Networks

Konstantinos Chasapis
DataDirect Networks

Antony Chazapis
Institute of Computer Science
FORTH

Angelos Bilas
Institute of Computer Science
FORTH

Michelle Aubrun
Thales Alenia Space

Teddy Debroutelle
CybeleTech

Claudio Disperati
MemEx Srl

Stefan Feit
BMW AG

Georgios Gardikis
Space Hellas S.A.

Alen Kopic
KOOLA

Bernhard Peischl
AVL List GmbH

Konstantinos Xylogiannopoulos
AVL List GmbH

Abstract

Seven pilots are challenging EVOLVE's testbed with varying needs and goals.

The applications are supporting actual business cases from diverse domain areas such as agriculture, preventive machine maintenance, public transport, maritime surveillance, mobility services, image change detection, and automotive data-driven systems. They apply multiple algorithms on different datasets, in terms of complexity and sizes; as such, they require different technologies, facing several concerns. Nevertheless, the EVOLVE platform, with its unique hardware setup and software stack, is able to accomodate them all.

Overview

The current status of the EVOLVE infrastructure encompasses both hardware and software innovations. The EVOLVE platform incorporates infrastructure with HPC hardware and associated middleware, plus a customized surrounding software environment focused on efficient job management. Use case developers have invested into porting their applications on EVOLVE and now enjoy the expected improvements: speed-up, scalability, as well as ease of deployment. The workflow engine allows deploying complex pipelines with tightly synchronized stages, the notebook approach simplifies the development of new features, and the performance dashboard helps developers to navigate through the complexity of performing optimizations.

The seven pilot applications included in EVOLVE are diverse, with different technical requirements that stress the infrastructure in multiple ways. Different types of data and computation patterns demand different types of technologies, combining Big Data, HPC and the Cloud. Individually, the technologies are available to support pilot demands. And while pilots do not share the same common technology stack, they coexist on the same cluster, sharing cloud resources and data infrastructures.

However, performance is not the only goal to reach.

Business owners need to be able to experiment with different technological options; they require quicker reactions in the product development process, while they also need safety when they decide to pick up a certain technology. The EVOLVE teams demonstrate how the innovative platform can provide the means for them to minimize their efforts and investment, in order not only to improve their application and fulfill their scaling needs, but also to be able to perform experiments and adapt their data pipelines in fluid conditions. Reduced time-to-market is of utmost importance for EVOLVE to demonstrate through these pilots. Business-wise this is the most significant achievement reached by EVOLVE; to become appealing by not necessarily requiring a highly-detailed level of technology knowledge.

Each pilot application validates the overall vision of EVOLVE and bears significant scientific results in its field. The results can have a direct impact on the European citizens with improved security, better mobility, or optimized agriculture. This paper presents the pilots in more detail, along with the obtained results and the challenges faced during development.

The EVOLVE Pilots

Maritime Surveillance

The aim of the maritime surveillance pilot in EVOLVE is to assess the value which the EVOLVE technologies can bring to the sector. Using a maritime surveillance platform developed in-house by Space Hellas as a starting point, we have been adapting and re-engineering its main components in order to benefit from the EVOLVE technology propositions.

The maritime surveillance workflow includes both the analysis of AIS data (Automatic Identification System – identity and course information broadcasted by all ships) as well as detection of vessels in SAR (Synthetic Aperture Radar) images, acquired either by satellites or patrolling aircrafts. AIS and SAR targets are correlated to identify potential non-cooperating vessels within the Area of Interest (AoI). In parallel, anomaly detection algorithms based on Deep Learning are applied on AIS time-series data, to identify suspicious behaviours of vessels, such as abnormal routes, intermittent transmissions, manipulated data etc. The maritime integrated situational picture (ISP) is visualized in a GIS-based GUI, where the vessels and all the associated metadata, along with the satellite imaging scenes, are displayed on a map.

Within EVOLVE, work focused on transforming the existing pipeline to adapt to the EVOLVE logic. This involved the architectural re-engineering of the maritime surveillance platform into a workflow-based containerized structure, with the aim of taking advantage of the EVOLVE accelerations, such as parallelization, fast storage and hardware-accelerated distributed processing. More specifically, the platform was decomposed into its basic elements, which were then deployed as Docker containers, following the general approach of EVOLVE. Containers were in turn organized into a workflow, first using Argo directly and then using the bespoke Zeppelin interpreter. Dedicated data ingestion modules were developed, ingesting both AIS data feeds as well as SAR scenes corresponding to the defined AoI. The SAR scene processing stage was upgraded using the open-source Sumo algorithm and visualization was achieved natively within Zeppelin.

The deployment of the SAR processing tasks as Argo workflow stages greatly facilitate their parallelization, merely by configuring the workflow accordingly. Parallelization, combined with the significant computing resources of the EVOLVE platform, resulted in decreasing the SAR processing time by at least 70% compared to the original, pre-EVOLVE deployment – thus greatly improving the response time of the system. At the same time, the AIS anomaly detection algorithm was also ported to Docker and mapped to an Argo workflow stage. This Deep Learning task is easily configured from Zeppelin to take advantage of two accelerating features of EVOLVE: GPU resources available in the cluster and an Apache Spark microservice for Big Data processing.

These features have resulted in decreasing the algorithm's training time by approximately 75% compared to CPU-only processing in the legacy system.

Satellite Image Change Detection

The image change detection pilot is another example that highlights the value of EVOLVE technologies in the spatial domain. Using a change detection building block developed in another project as a starting point, Thales Alenia Space has modified the pipeline architecture and adapted the modules in order to benefit from the EVOLVE technology propositions.

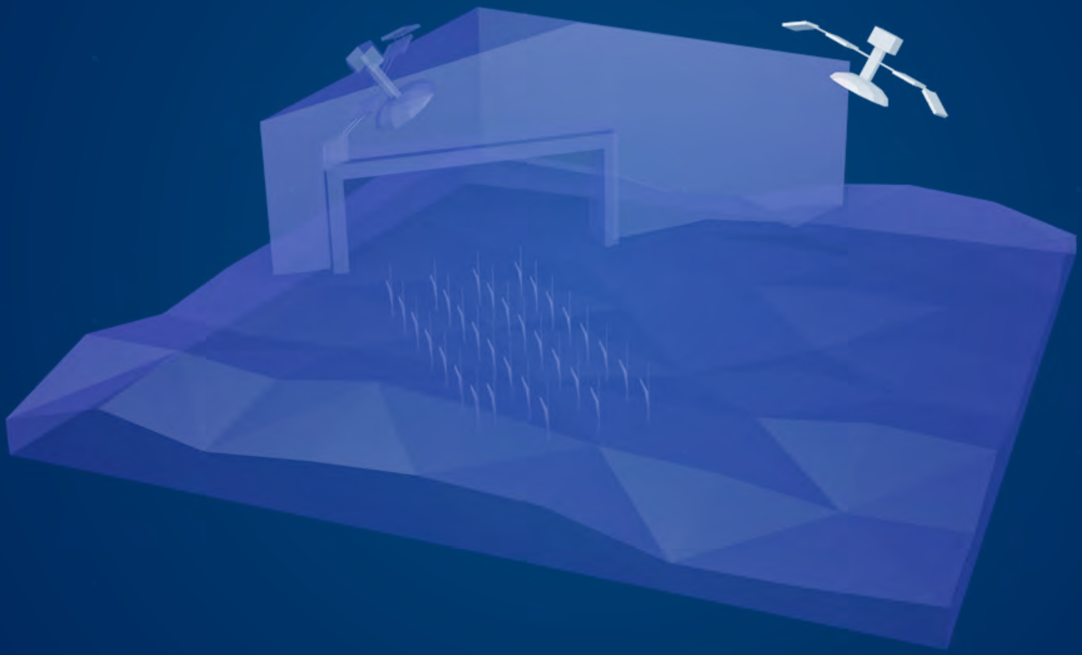
The change detection workflow includes three major steps. The first one downloads Sentinel-2 data products from one of the Copernicus access platforms. The second step processes each Sentinel-2 data product in a single Sentinel-2 image with the bands of interest. After the two time-consecutive Sentinel-2 images of the same field-of-view have been generated, the last step computes the change detection map between them. Before comparison, the Sentinel-2 images are projected via a neural network model into a more robust feature space, which is invariant to the lighting and atmospheric conditions that do not represent changes of interest.

In the context of the EVOLVE project, Thales Alenia Space has used technology components, such as parallelization frameworks, stream analytic engines, fast storage, and hardware accelerators, in order to automate, optimize, and boost the performance of the change detection tool. More specifically, the different modules have been redesigned into micro-services that achieve one basic function and do not make many iterations on the data products. These modules have been encapsulated in containers, which communicate with each other using Kafka topics, which allow coarse-grain parallelism.

Argo Events has been used to create Kubernetes pods from Kafka messages and release the resources used once the calculation is complete. A Dask scheduler has been deployed for parallel computing inside the modules, and the module with the Artificial Intelligence algorithm has been (1) rewritten in pure TensorFlow to take full advantage of the GPU accelerators and (2) ported to FPGAs via OpenCL to compare the performance.

The deployment of Kafka, Argo Events and Dask technologies has allowed not only to automate the execution of the change detection pipeline, but also to considerably improve the processing time (by a factor greater than 10), without increasing the total number of resources used. Furthermore, the different configurations of the AI module tested have led up to a speed up of about 30% for the GPU accelerators and about 90% for the FPGA accelerators thanks to minimizing data transfers.

Optimizing agri production yield using numerical models and massive historic data



Optimizing agri production yield using numerical models and massive historic data

A main problem that is challenging today is anticipation of yield potentials to make better agri-food decisions. In this scope, CybeleTech developed automated tools to identify crop lands in a region from satellite images and then to predict crop growth for accurate in-season forecasts of the total agricultural production in one region, e.g. Europe or the Corn Belt. Currently, we are faced with the problem of massive data from the Sentinel-1, Sentinel-2 and ERA5 satellites. We do not have the computational capacity to execute our codes. These two phases of agricultural production forecast offer significant challenges for data processing, as follows:

Phase 1:
Crop identification via processing of satellite images

This phase identifies the type of crop on soil based on remote sensing data. It is divided into two different stages, namely the learning stage and the forecast stage. During the learning phase, the model, based on CNNs, is trained using a series of satellite images. We have worked extensively on supervised learning by comparing outputs with real datasets but have also experimented with unsupervised learning. During the forecast stage, the trained model is used on every pixel of a large region (covering multiple tiles of satellite images). This stage is (schematically) represented by the sequence of operations: read pixel data from storage, compute model from data, write result to storage. In the forecast stage, computations are spatially independent, so this step can run in parallel for every pixel, imposing a large load on the storage system.

Phase 2:
Data assimilation in plant growth

In the second phase, plant growth models are used to make numerical simulations of plant development in the field during the growing season. A state of the soil-plant-atmosphere system is simulated day by day, taking into account soil composition and climate data. The state of the system is described by plant variables, such as biomass of different organs or leaf area coverage, soil status with e.g. water content in different sublayers and so on. Although this is an exciting approach that opens up new possibilities, plant growth models have margin errors because:

- **The entire crop functioning is not described in the model processes;** for instance, the model may not simulate pest disasters.
- **The mechanistic processes included in the model are never fully accurately described** because we are trying to model a complex living system (there is no equivalent of Navier-Stokes equations for plant growth...).

Optimizing agri production yield using numerical models and massive historic data

To correct these errors, we use data assimilation methods, an approach similar to Kalman filtering for linear systems, thanks to data for plant development coming from remote field sensors, acquired during the growing season.

Model simulations are corrected online during the season using such data. Since we are not dealing with a simple linear model and we do not particularly expect errors to be Gaussian, we use methods that are numerically more complex than Kalman filters. The method is based on a Bayesian framework where a lot of trial simulations are produced for a different set of model parameters to identify a better representation of reality. With this method, large numbers of particles have to be generated to get a correct representation of the actual probability distribution. We have never been able to test this approach at large scale before because of the required number of particles. This is an HPC/BD problem that can benefit from fast storage to process states of the particles as subsets. At each step, the amount of data that would be read/written/processed are at least a few TBytes to achieve good results. In addition, several instances of this (data assimilation) phase may run concurrently because data are spatialized (satellite image), with one instance corresponding to one pixel.

The first stages of pilot development, up to M12, focused on the first part of the workflow, crop recognition. This part is deployed as a container, following the general approach of the EVOLVE platform. Then, the simulation stage was deployed as a container. The two containers communicate with each other. The first step uses TensorFlow and the CUDA library. The second step uses MPI. Then we improved the way of retrieving and storing images from Sentinel-1 and Sentinel-2. The metadata are better managed in order to accelerate the response time of the different requests.

The first step (crop identification), cultural recognition uses TensorFlow and CUDA, thus GPU resources available on the HPC platform. Data are provided by the Copernicus program. The algorithms developed make it possible to identify the crops present on the input images (wheat, maize, rapeseed, barley). The output is a map in false colors representing different crops. Once the cultural variety has been determined at a geographical point, we proceed to the second step (simulation), where we select the appropriate growth model to estimate the yield of the field. This step uses MPI.

A big challenge has been to be able to train the CNNs over a large geographic region. Currently, the geographic area for learning is 20x20 km. The learning process requires 40 GB of RAM.

We are working on increasing the learning area. Another challenge has been to find the best method to optimize the calculation time of the different parameters in the simulation step.

To correct these errors, we use data assimilation methods, an approach similar to Kalman filtering for linear systems, thanks to data for plant development coming from remote field sensors, acquired during the growing season. Model simulations are corrected online during the season using such data. Since we are not dealing with a simple linear model and we do not particularly expect errors to be Gaussian, we use methods that are numerically more complex than Kalman filters. The method is based on a Bayesian framework where a lot of trial simulations are produced for a different set of model parameters to identify a better representation of reality. With this method, large numbers of particles have to be generated to get a correct representation of the actual probability distribution. We have never been able to test this approach at large scale before because of the required number of particles. This is an HPC/BD problem that can benefit from fast storage to process states of the particles as subsets. At each step, the amount of data that would be read/written/processed are at least a few TBytes to achieve good results. In addition, several instances of this (data assimilation) phase may run concurrently because data are spatialized (satellite image), with one instance corresponding to one pixel.

The first stages of pilot development, up to M12, focused on the first part of the workflow, crop recognition. This part is deployed as a container, following the general approach of the EVOLVE platform. Then, the simulation stage was deployed as a container. The two containers communicate with each other. The first step uses TensorFlow and the CUDA library. The second step uses MPI. Then we improved the way of retrieving and storing images from Sentinel-1 and Sentinel-2. The metadata are better managed in order to accelerate the response time of the different requests.

The first step (crop identification), cultural recognition uses TensorFlow and CUDA, thus GPU resources available on the HPC platform. Data are provided by the Copernicus program. The algorithms developed make it possible to identify the crops present on the input images (wheat, maize, rapeseed, barley). The output is a map in false colors representing different crops. Once the cultural variety has been determined at a geographical point, we proceed to the second step (simulation), where we select the appropriate growth model to estimate the yield of the field. This step uses MPI. A big challenge has been to be able to train the CNNs over a large geographic region. Currently, the geographic area for learning is 20x20 km.

The learning process requires **40 GB of RAM.**

We are working on increasing the learning area. Another challenge has been to find the best method to optimize the calculation time of the different parameters in the simulation step.

Improvement of bus public transport services

The overall scenario concerning Bus Public Transport (hereinafter "PT") service is constantly changing very fast in terms of ITS and digital tools, alternative bus fuels, operator capabilities and so on. In any case, for transport company managers, the possibility to optimize the service (scheduled and operated), to reduce costs and to enrich the service portfolio offered remains still open; as do relevant issues. In any case, there is also the awareness to adopt suitable digital tools in order to monitor and analyze the operated services, so as to promptly manage any emergencies or criticalities. Many data are available today for analysis of transit services, but not many solutions exist to carry out the respective data and service assessments reliably. In fact, powerful applications and tools are needed in order to allow the management of big data sizes related to long service periods (up to a year) or real time data which deal with service events occurring on the network. This latter case envisages the need to analyze this information and to run algorithms very quickly and many times a day.

These aspects were the main motivations for defining I) "Not Real Time [NRT]" and II) "Real Time [RT]" bus transportation use cases. The first use case, NRT, aims to elaborate and store "big data" related to a PT service, in order to analyze them with specific visualization/ business intelligence tools developed by MemEx and Tiemme before the EVOLVE project. This tool was not able to analyze data over a longer period than 1 month of operated PT service. This was a critical constraint, because only with an overall view of data collected over a long period and grouped by "lines" and "paths" it is possible to identify the "technical" and "operational" problems which affect the service and the related data.

Improvement of bus public transport services

Without EVOLVE the assessment for a longer period (than 1 month) could be very difficult requiring long running times and filtering operations. Thanks to the EVOLVE-implemented workflow the data collected through the "AVM – Automated Vehicles Monitoring" system (already implemented by Tiemme, a company which operates the PT services in South Tuscany) are ingested and stored in the EVOLVE platform, in order to make them available for subsequent analysis.

Concerning the RT use case, design and implementation activities of a specific tool for PT service monitoring and management have been carried out within the EVOLVE project. In particular, starting from the existing webLyzard platform, Tiemme and MemEx provided technical and functional specifications to add additional functions to webLyzard solution (visualization tool), in order to visualize and manage information and data about the public road network status in terms of traffic congestion. Also in this case, as for the NRT one, the main data source is the AVM system which provides bus event information allowing to estimate the service status, regularity, and other useful Key Performance Indicators. Based on specific algorithms the traffic congestion on road sections regarding the involved network areas is estimated and presented on the map.

The two use cases involved specific workflows interfacing dedicated Web Services implemented in the Tiemme/MemEx environment. On the EVOLVE platform data are stored and downloaded by visualization or relevant tools thanks to the "Spark Thrift Server (STS)" implemented.

The benefits concerning the Public Transport use case can be evaluated by considering the reduction of the spent resources in terms of human effort and lost time to make service data assessment and real time transport service management. The HPC workflows implemented in the EVOLVE project, help to analyze – in both "on-line" and "off-line" ways – the service data for technical and operational problems identification. Solving these issues, the "level of service" can be increased as can be the optimization of the scheduled service.

In order to evaluate the reproducibility and transferability of the implemented workflows several PoCs have been realized:

1. Improvement of PT service performance/reliability: like the "NRT" testbed but with a different data source;

2. Improvement of service operation and provision of user information: like the "RT" testbed but with a different data source;

3. "AFC (Automated Fare Collection) system – payment transactions analysis": in this case the kind of data integrated using the same workflow is totally different from the testbed dataset already considered.

Based on the positive results of these three PoCs, the high level of reproducibility and transferability of the implemented solution have been verified.

Anomaly detection in internal combustion engines

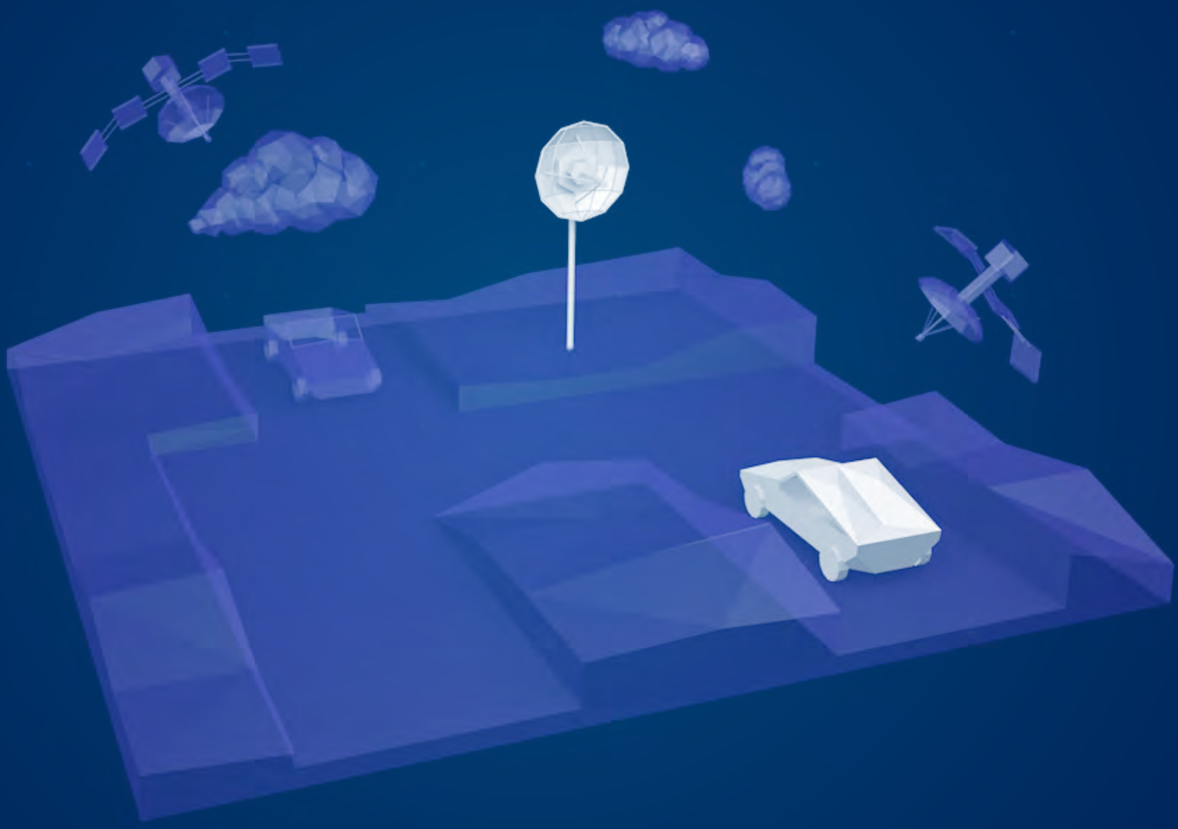
The main focus of the AVL EPOS system pilot is the detection of anomalies or in general of abnormal behaviors of internal combustion engines. The current system has the ability to analyze the Cylinder Pressure signal recorded for every cylinder and classify each cylinder in predefined classes. The system is very mature and stable and the reason of transferring an already existing technology on the EVOLVE platform was to (a) evaluate its performance in comparison to advanced high-performance computing systems, (b) consider the possible expansion to control and analyze multiple input channels in real time, and (c) take the opportunity to test the system on a distributed, Cloud environment. With the help of EVOLVE it has been possible to examine the potential and limitations of the system and provide an accurate evaluation that is critical for future development.

The EPOS system is used for the analysis of time series (mainly cylinder-pressure related) and has the ability to detect with high accuracy and efficacy abnormalities that have occurred and are related to (a) the performance of the engine and/or (b) potential failure of the sensor recording the engine values. Using advanced pattern detection algorithms and related data structures, EPOS can record any possible deviation from a predefined normal behavior, as this has been documented using historical data to create a “reference curve”. Every new time series entering the system is transformed to a discrete sequence using a complex discretization algorithm that translates real values to alphabet-based values. Then the sequence is compared to the reference curve sequence and an advanced algorithm detects every possible repeated pattern among the two sequences.

The system is built on top of AVL's preexisting system Concerto-EPOS, which is a Windows application, and is highly optimized to run on a single laptop computer. The first challenge of the EVOLVE evaluation was to transform the codebase to a portable Python-based implementation and bypass any Windows dependencies. After the transformation phase, the system was containerized and uploaded to the EVOLVE platform. Using Argo, it was possible to execute the system in full parallelization for all cylinders. In order to improve performance, we used the Scalene Python profiler that detects hidden points that can further be improved in the Python code. Additionally, the file format was changed to parquet which allowed us to take advantage of parallelization properties and eliminate serial data loading.

The EPOS system has a single process execution schema. As the original system was already optimized and extremely fast, the execution inside a single container on the EVOLVE platform had a small impact on its performance. With EVOLVE it has been possible to further expand its applicability on multiple inputs and scale out execution to multiple nodes, significantly improving runtimes when monitoring multiple engines simultaneously. To validate this case, a fleet dataset has been used to perform parallel analysis, which resulted in an improving the performance between 50% to 70% compared to sequential execution.

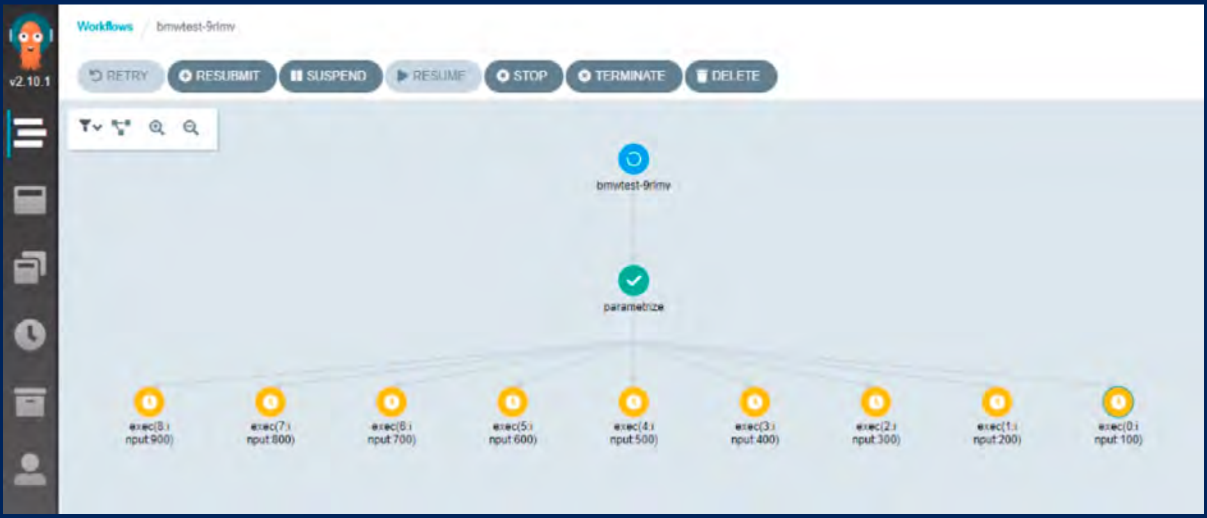
Ride hailing



Ride hailing

The goal of the pilot provided by BMW is to measure the impact of parallelization in simulations of a ride hailing service on EVOLVE. This can be achieved in different manners: First, the parallelization of different simulation scenarios, like varying fleet sizes or other system parameters. Second, as such simulations can run independently for each parameter set, a horizontal scaling of the problem is also possible.

To realize this, BMW uses a Zeppelin notebook to create the different parameter sets and then starts an Argo workflow where all simulations run in parallel. In this case the EVOLVE platform speeds up the calculation time, because a higher number of CPUs is available, than for a single server.



The next milestone is to run parts of each individual simulation horizontally parallelized. The algorithm that assigns new user requests to vehicles of the on-demand mobility fleet is based on the so-called nearest neighbor policy. Hence, for each vehicle within a certain radius around the pickup location of a newly requested ride, the shortest route is calculated, and the closest vehicle is assigned to the request. This procedure is currently conducted sequentially but can be run in parallel, since the routing of each vehicle can be computed independently from every other vehicle.

An acceleration of the assignment process directly translated to quicker response times to user requests, which significantly increased the received service quality. BMW looks forward to quantitatively measuring the speed-up potential and further optimizing the simulations run on EVOLVE.

Predictive vehicle maintenance

Predictive maintenance aims to identify vehicle maintenance issues before they occur by leveraging data from warranty repairs with current vehicle sensor data. A Machine Learning model is implemented and deployed on the EVOLVE platform. There are three important stages during the deployment process: Dataset sync, Docker image, Argo workflow:

Dataset sync: There is an Argo workflow created that syncs the input learning dataset on EVOLVE and new data present on a remote server. The crucial command used in this workflow is rsync. In order to create connections ssh keys are used.

Docker image: Python code and different Python libraries are used for the Machine Learning model. Once the code is ready for deployment, a Docker image is created. New changes in the code means a new image tag. The created image is pushed to the EVOLVE platform in a private container registry.

Argo workflow: When the input dataset and Docker image are on the EVOLVE platform, everything is ready to run the training process. Argo Workflows is used for this. The workflow is fairly simple: Run Docker images in parallel for different ML algorithms.

All interactions with EVOLVE are done via Zeppelin. The problem is computational (millions of records) and requires utilization of EVOLVE CPUs as well as GPUs. It is impossible to run the model on a local machine when the number of records is bigger than 1 million.

Conclusion

Seven different use cases are challenging the benefits and business impact of the EVOLVE platform, which can accommodate different computation and data patterns in terms of volume needs, performance goals and business anticipations. In this exercise, EVOLVE deployments across pilot cases have resulted in several findings, in respect to:

Performance: The use cases are using technology components, such as parallelization frameworks, stream analytic engines, fast storage, and hardware accelerators that automate, optimize, and boost the performance of applications. Training times of neural network models have decreased up to 75% in respect to pre-EVOLVE legacy systems, even with optimizations applied. EVOLVE has enabled execution with multiple inputs and on multiple nodes, exploiting the microservices design, and coarse-grain parallelism on top of GPU and FPGA accelerators.

Business-level achievements: Earth analysis and transport network planning applications have managed to expand the sizes of areas that are being processed, in similar or even less time. This is of utmost importance for applications which need to expand their data geographically. Moreover, car assignment services respond rapidly to user requests, which is the main indicator for the desired Quality of Service. EVOLVE is providing a promising environment for these achievements.

Development efficiency: Human effort and development-time to adapt and deploy workflows are considerably decreased. The use of Zeppelin and the EVOLVE dashboard facilitates quick adaptation of data processing pipelines. Deployment of processing tasks as Argo workflow stages greatly simplify their parallelization, merely by configuring the workflow accordingly. The reproducibility and transferability of implemented workflows is also important; it has been verified through the application of similar pipelines into other use cases of similar workflow stages.

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