http://www.everest-h2020.eu

dEsign enVironmEnt foR Extreme-Scale big data analyTics on heterogeneous platforms

D6.5 — Use case evaluation report

The EVEREST project has received funding from the European Union's Horizon 2020 Research & Innovation programme under grant agreement No 957269

Project Summary Information

Project Title	dEsign enVironmEnt foR Extreme-Scale big data analyTics on heterogeneous platforms
Project Acronym	EVEREST
Project No.	957269
Start Date	01/10/2020
Project Duration	42 months
Project Website	http://www.everest-h2020.eu

Copyright

Num.	Partner Name	Short Name	Country
1 (Coord.)	IBM RESEARCH GMBH	IBM	CH
2	POLITECNICO DI MILANO	PDM	IT
3	UNIVERSITÀ DELLA SVIZZERA ITALIANA	USI	CH
4	TECHNISCHE UNIVERSITAET DRESDEN	TUD	DE
5	Centro Internazionale in Monitoraggio Ambientale - Fondazione CIMA	CIMA	IT
6	IT4Innovations, VSB – Technical University of Ostrava	IT4I	CZ
7	VIRTUAL OPEN SYSTEMS SAS	VOS	FR
8	DUFERCO ENERGIA SPA	DUF	IT
9	NUMTECH	NUM	FR
10	SYGIC AS	SYG	SK

Project Coordinator: Christoph Hagleitner – IBM Research – Zurich Research Laboratory

Scientific Coordinator: Christian Pilato – Politecnico di Milano

Disclaimer

Deliverable Information

Work-package	WP6
Deliverable No.	D6.5
Deliverable Title	Use case evaluation report
Lead Beneficiary	PDM
Type of Deliverable	Other
Dissemination Level	Public
Due Date	31/03/2024

Document Information

Delivery Date	25/06/2024
No. pages	66
Version | Status	1.1 | Final
Responsible Person	Gianluca Palermo (PDM)
Authors	Gianluca Palermo (PDM), Fabrizio Ferrandi (PDM), Fabien Brocheton (NUM), Jakub Beránek (IT4I), Paulo Silva (IT4I), Jan Křenek (IT4I), Radim Cmar (SYG), Antonio Parodi (CIMA), Riccardo Cevasco (DUF)
Internal Reviewer	Michele Paolino (VOS)

The list of authors reflects the major contributors to the activity described in the document. All EVEREST partners have agreed to the full publication of this document. The list of authors does not imply any claim of ownership on the Intellectual Properties described in this document.

Revision History

Date	Ver.	Author(s)	Summary of main changes
01.03.2024	0.0	G. Palermo (PDM)	Initial draft
01.04.2024	0.1	F. Brocheton (NUM)	Initial NUM Contribution
23.04.2024	0.2	R. Cevasco (DUF)	Initial DUF Contribution
26.04.2024	0.3	R. Cmar (SYG)	SYGIC Contribution
29.04.2024	0.4	P. Silva J. Beranek (IT4I)	IT4I Contribution
04.05.2024	0.5	ALL	Revised contributions
06.05.2024	0.6	K. Slaninova (IT4I)	Revised Contributions in several sections
15.05.2024	0.7	ALL	Document update after EB
16.05.2024	0.8	G. Palermo (PDM)	Content Revision, Project KPIs, Conclusions
20.05.2024	1.0	M. Paolino (VOS) and G. Palermo (PDM)	Internal review and finalization
25.06.2024	1.1	G. Palermo (PDM)	Revisions after Monitors' comments

Quality Control

Approved by Internal Reviewer	May 18, 2024
Approved by WP Leader	May 20, 2024
Approved by Scientific Coordinator	May 22, 2024
Approved by Project Coordinator	June 25, 2024

Table of Contents

1 Executive Summary

This document presents the outcomes of the EVEREST project, with a particular focus on the impact and achievements related to the three target use cases.

Leveraging the EVEREST SDK and the other technologies developed in the project, we built demonstrators for the three target applications, which were chosen to represent a diverse range of challenges. We considered the use case objectives defined by the application partners at the beginning of the project (and refined at the end of the second year).

For each of the three target use cases, several tables were created to show different targets key performance indicators (KPIs) and the actual value achieved within the project timeframe. A description was included for each objective defined in Deliverable D2.1 and revised in Deliverable D2.4, highlighting the path followed to reach the goals and the actual results achieved.

A user story has also been included in the document to illustrate the implications of using the EVEREST design environment for future adopters across different actors.

Finally, project-level KPIs were assessed to measure the success and impact of the EVEREST approach. The results demonstrate the effectiveness of the EVEREST approach in meeting the predefined KPIs. However, not all use cases benefited equally (e.g. WRF-based use cases).

Deliverable highlights

No.	Highlight	Section(s)
1	Assessment of the objectives for the three EVEREST use cases	Section 2
2	User Story for the usage of the EVEREST SDK	Section 3
3	Assessment of project-level KPIs	Section 4

2 Evaluation of EVEREST Use Cases

In this section, we present the evaluation of the EVEREST use cases. This section is driven by the objectives claimed by the use case providers in the context of Deliverable D2.1 and in its revised version Deliverable D2.4, where the main use case goals have been claimed.

2.1 Renewable Energy Prediction

The forecast of the energy generated by a wind farm can be divided into three steps:

1. the weather predicted in the coordinates of the plant by computing a weather prediction model (with WRF);
2. the pre-processing of weather data using a physical model (i.e., turbine power curve) and machine learning for anomaly detection, aimed at maintaining the cleanliness of the dataset for training ML forecasting algorithms
3. the implementation of Machine Learning algorithms for predicting the energy generated.

In terms of computation, the weather forecast is the most node-consuming, with an execution time of hours compared to minutes for the other steps. The main objectives are:

- improving forecast accuracy
- reducing the execution time of the WRF forecast as much as possible.

The individual phases relating to the development of the application from scratch during the EVEREST project are described below.

2.1.1 O1.1 - The implementation of a **DUF** proprietary application for energy production prediction

DUF dispatches a portfolio of wind power production, with the goal of optimizing profit by reducing forecast errors and, therefore, imbalance costs. For this purpose, over the years, **DUF** has selected the best forecast providers on the market. The application was developed from scratch within the EVEREST project in the several stages detailed in Deliverable D6.4, Section 3.7. The entire application workflow can be divided into three main steps: a first step related to WRF weather prediction, a second step for data pre-processing, and a third step for training ML algorithms and energy prediction.

In the first phase, it was necessary to conduct extensive research on the sector's scientific literature to analyze the methods and trends for forecasting the energy of wind farms. A set of publications was identified to analyze various topics such as state-of-the-art Artificial Intelligence techniques (AI) most suitable for short-term wind generation prediction; the best metrics to measure the accuracy of wind power forecasting; the importance of data pre-processing to increase model robustness and to refine the uncertainty of data; the analysis of other WRF test cases; the advantages of hybrid models (physical-statistical); the probabilistic methods (ensemble).

To evaluate the accuracy of the application's forecasts, an industry-leading reference forecast provider was chosen as a benchmark based on a comparative analysis of different providers.

Finally, the development of the workflow has been carried out, dividing it into different steps. The workflow of the entire application, as described in detail in D6.3, involves the use of weather forecasts generated by WRF, improved with data assimilation; data pre-processing using observation data and an anomaly detection module; and finally the use of Machine Learning algorithms to predict the energy generation.

Hereafter, we list the set of sub-objectives identified to reach the final goal of implementing and evaluating the workflow for energy production prediction.

ITEM	DESCRIPTION
Method	Studying the state-of-the-art of scientific literature to understand the most interesting and promising trends
Priority	Must Have
Baseline	N/A
KPI	Availability document with state-of-the-art analysis
Notes	There was no implemented application, thus an assessment of the state of the art and SWAT analysis was needed
How To Measure	The KPI can be considered as reached if there is a document in the EVEREST repository listing the state-of-the-art approaches and trends.
Involved EVEREST Components	ML predictor for energy production forecasting
Target Value	Document available in the EVEREST repository
Reached Value	The document with literature highlights has been made available for internal use in the EVEREST repository.

Description:

In the first phase of the project, to develop the application and evaluate its results, analysis was necessary to investigate the most promising methods for wind energy forecasting.

Intermediate Steps:

1. Identification of state-of-the-art approaches: some articles have been identified in the literature to explore the best state-of-the-art methodologies.
2. Selection of the most interesting approaches: the analysis led to identifying the most suitable ML algorithms for the use case; to implement a hybrid physical-statistical approach to combine its strengths; to identify different pre-processing methods for training ML models; to delve deeper into the advantages of data assimilation; to classify metrics to evaluate the accuracy of forecasting models.
3. Writing of the document: the articles have been summarized in a table indicating key points and feedback
4. Upload of the document in the EVEREST repository for sharing the gathered knowledge

Result:

A complete document about the state of the art of scientific literature in the field has been prepared and released within the EVEREST internal repository. The main feedback is related to choosing a hybrid approach (statistical/physical) with data pre-processing based on the wind turbine power curve. In literature, several articles demonstrate the importance of combining statistical and physical methods and the impact of pre-processing to improve forecast accuracy.

ITEM	DESCRIPTION
Method	Summarizing the state of the practice in the industrial sector to be aligned and possibly outperform the current scenario
Priority	Must Have
Baseline	N/A
KPI	Presence of a comparative analysis for the different methods and different providers
Notes	The commercial services for wind farm energy forecast are "black box" services, with no possibility for the customer to know the methods and approaches implemented
How To Measure	The accuracy of the several forecast providers is measured using mean Absolute Error of the power generated for each hour [MWh], normalized by the installed capacity of the wind farm (NMAE).
Involved EVEREST Components	Not applicable
Target Value	Demonstrate the choice of the reference provider used to evaluate the accuracy of the energy prediction application
Reached Value	Build a comparison of the accuracy results of several forecast providers tested on **DUF** wind farm portfolio, available in the EVEREST repository.

Description:

The assessment aims to verify the state of the art of wind forecast providers for the sale of energy generated on short-term energy markets and to identify a reference provider to use as a benchmark for the application results in terms of forecast accuracy.

Intermediate Steps:

1. Selection of the providers for the comparative analysis, tested on **DUF** wind farm portfolio
2. Collect results related to 2022 **DUF** for wind farm portfolio
3. Collect results related to 2023 **DUF** for wind farm portfolio

Result:

Three providers for the wind power forecast service have been analyzed and one of them has been proven to be the best in terms of accuracy, flexibility, and reliability of the service. This comparative analysis confirmed the choice of the reference provider used up to now to compare the results of the energy forecast application in terms of accuracy and provides a challenging benchmark result. Over the two benchmarked years the selected providers demonstrated an NMAE (Normalized Mean Absolute Error) lower than 10%, while the others had a value larger than 10%.

ITEM	DESCRIPTION
Method	Design of a specific application workflow
Priority	Must Have
Baseline	N/A
KPI	Presence of code deployed within the EVEREST environment
Notes	No baseline was available at the beginning of the project, the entire worflow and prediction application have been developed from scratch
How To Measure	The KPI can be considered as reached if there is the code deployed within the EVEREST environment, producing each day a forecast of the day ahead wind power generation.
Involved EVEREST Components	WRFDA+ML energy predictor
Target Value	Code deployed in the EVEREST repository
Reached Value	Code was developed and tested on **DUF** machines, and in the last phase of the project has been tested and deployed on the EVEREST environment

Description:

The final application workflow has been deeply described in D6.3, Section 5.1. To build the workflow, different development phases have been done which are summarized hereafter.

Intermediate Steps:

1. Data collection: for the training of ML algorithms, one year of WRF data and observation data were needed, mainly wind farm generation data.
   One year of WRF data (**CIMA**) and one of the observation data (**DUF**) were collected to train the ML algorithms (see Deliverable 6.1, Section 5.2.4.1).

2. Variable selection: WRF produces a set of weather variables, it has been necessary to identify which variables bring value to wind energy forecasting.
   The most important variables were selected from the list of weather parameters coming from WRF data. As expected, wind speed proved to be by far the most impactful parameter.

3. Data-preprocessing: The analysis determined that there was a need to introduce filters to eliminate outliers and maintain a good level of data quality for model training.
   The pre-processing methods implemented a filter based on the Power Curve of the Wind Turbine (detailed in D6.3, Section 5.3.) and an anomaly Detection module (D6.3, Section 5.2.2).

4. Training strategy: Study and development of training strategies for the machine learning model. Several tests and analysis to identify the best training strategy for the model has been carried out on the available data.
   Several training strategies were tested, choosing the "increasing period" method, as described in D6.3, Section 5.4. The data relating to the 24 hours of the previous day are added each day.

5. Model Evaluation: Testing and evaluation of Machine Learning algorithms to identify the most suitable algorithm for this specific use case have been carried out.
   Two classes of algorithms were selected, Kernel Methods and Deep Neural Network, each with pros and cons. Considering the features of the datasets, the different levels of difficulty to identify the settings and the first results obtained, the choice fell on Kernel-Ridge with Gaussian Kernel (D6.3, Section 5.4.1).

6. Data post-processing: to reduce further the data noise effect data post-processing methods have been

applied to the produced data.
The post-processing was performed using the Kalman smoothing technique (D6.3, Section 5.3.1).

Result:

After all the previous intermediate steps, the final version of the **DUF** proprietary application for predicting the energy production of a wind farm has been released within the EVEREST repository and tested on the EVEREST environment. Further details on the applications can be found on Deliverable D6.3.

2.1.2 O1.2 - Focus on a daily day ahead forecast (24 hrs day ahead) timeline

The application workflow is focused on forecasting wind energy over the 24 hours of the day following the forecast day. In fact, the objective is to forecast the energy to be sold in the day-ahead market, which is the one with the highest volumes traded on the energy markets for wind sources. The daily workflow is detailed in D6.4, Section 3.7.

ITEM	DESCRIPTION
Method	Understanding pros and cons of different models and approaches
Priority	Must Have
Baseline	ML algorithm Kernel Ridge with Gaussian Kernel without pre-processing and post-processing
KPI	Quality improvement of the Wind Farm Energy prediction
Notes	Power generation forecast accuracy
How To Measure	Mean Absolute Error of the power generated for each hour [MWh], normalized by the installed capacity of the wind farm (NMAE).
Involved EVEREST Components	WRFDA+ML energy predictor
Target Value	Comparison of the baseline approach with the one with pre-processing only, and the one with both pre and post-processing.
Reached Value	Kernel Ridge with pre-processed input dataset and post-processed output, was found as the best method in terms of accuracy. Results summarised here in the table "Results with pre-processing and post-processing" saved in the EVEREST repository.

Description:

As confirmed by the literature, data cleaning is crucial to improve the accuracy of the prediction, especially in cases where large amounts of data are not available to train the models.

Intermediate Steps:

1. Applying pre-processing on the input dataset using the power curve of the wind farm turbines, and testing the baseline model on the filtered data
2. Application of smoothing method on the output provided by the previous method, in order to filter the noise from the signal and obtain a more coherent and realistic wind generation shape

Result:

The original dataset was filtered using the power curve. The output of ML was post-processed using the Kalman smoothing technique. The improvement of the accuracy has been evaluated on a backtesting period

of one year, in comparison with the baseline. The impacts of pre-processing methods on prediction error are explained in D6.3, Section 5.3.3.

ITEM	DESCRIPTION
Method	The selection of the proper methods, metrics and algorithms to be implemented in the workflow for this temporal scale
Priority	Must Have
Baseline	Comparison among different ML algorithms
KPI	Quality improvement of the Wind Farm Energy prediction
Notes	Comparison involved dataset dimension, quality and features; Deep Learning, Recurrent Neural Network, XGBoost algorithms have been tested, besides Kernel-Ridge.
How To Measure	The forecast accuracy using Normalized Mean Absolute Error, and the computational complexity of the solution.
Involved EVEREST Components	ML energy predictor
Target Value	Selection of the method that minimizes the NMAE. Between two models with similar results, the choise fell on the simplest model in terms of computational complexity and interpretability of the solution.
Reached Value	Kernel Ridge with Gaussian Kernel model was selected, as it proved to be the method with the best performance in terms of NMAE in almost all months over the entire backtesting period, and it is also the fastest method. Results are summarised in a table in the EVEREST repository.

Description:

To demonstrate the choice of the Kernel Ridge algorithm, a backtesting comparison with other ML algorithms was carried out.

Intermediate Steps:

- Backtesting with Kernel Ridge algorithm;
- Backtesting with XGBoost algorithm;
- Backtesting with Recurrent Neural Network algorithm.

Result:

Backtests with XGBoost and RNN were performed with the same data pipeline of the Kernel Ridge, over the same test period. The best configuration was found using the Gaussian kernel, with an incremental training window. The optimization parameters were found using the grid search cross-validation technique. Figure 1 shows some results obtained with the different models.

Month	Kernel Ridge [NMAE]	XG Boost [NMAE]	RNN [NMAE]
Jun-21	7.4%	8.1%	8.7%
Jul-21	8.0%	8.2%	8.9%
Aug-21	6.6%	6.8%	7.2%
Sep-21	5.8%	6.0%	6.7%
Oct-21	8.6%	8.1%	8.8%
Nov-21	8.8%	9.2%	9.8%
Dec-21	10.3%	10.3%	11.6%
Jan-22	12.2%	12.8%	14.9%
Feb-22	10.1%	10.7%	12.0%
Mar-22	10.0%	9.9%	9.8%
Average	**8.8%**	**9.0%**	**9.8%**

Figure 1 – Models NMAE comparison

ITEM	DESCRIPTION
Method	The evaluation of probabilistic ensemble approach
Priority	Could Have
Baseline	Baseline prediction without ensemble model
KPI	Quality improvement of the Wind Farm Energy prediction
Notes	The execution of a small ensemble run of WRF simulation requires a quite huge amount of computational power. At this stage of the project, it was not possible to obtain such a number of FPGA-accelerated cores
How To Measure	The forecast accuracy using Normalized Mean Absolute Error, and the computational complexity of the solution
Involved EVEREST Components	WRFDA+ML energy predictor
Target Value	A method to exploit ensemble approach
Reached Value	**CIMA** made available on KAROLINA one month of WRF ensemble forecasts (around 1.1 TB). **DUF** is analyzing the data, with the goal of identifying the best method for the ML algorithms to process the data

Description:

The target of the ensemble analysis is to evaluate the possibility of using a probabilistic method to improve the accuracy of the forecasts. The backtesting analysis was based on one month of WRF data: for each hour an additional 10 forecasts were generated with a small perturbation of the input parameters.

Intermediate Steps:

- Ensamble data availability by CIMA;
- WRF ensamble ML implementation by **DUF**.

Result:

WRF data are available and **DUF** analysis in ongoing, to identify the best method to exploit the ensemble approach and to improve the forecast accuracy. The ensemble runs have been done at the end of the project and are under analysis while writing the document.

2.1.3 O1.3 - Effective run of the service in terms of computational performance efficiency with a specific focus on the WRF workflow

Renewable energy prediction is an intensive application, and the timescale of the forecast has an obvious impact on the processing times. The exploitation of the computational heterogeneous resources shared in the EVEREST project represents the enabling infrastructure to tackle such intensive application. By measuring the time taken by the various components of the workflow, we found that the most intensive part of the use case is the WRF execution on HPC resources. It takes about 85-90% of the computation: the FPGA acceleration of the most intensive WRF components could improve the overall performance.

ITEM	DESCRIPTION
Method	Evaluating the most intensive models and kernels (e.g. physics parameterizations) selected for WRF modelling for cloud-resolving (1-3 km grid spacing) applications
Priority	Must Have
Baseline	WRF CPU version
KPI	% of WRF execution to be accelerated with EVEREST technology
Notes	Not all WRF can be accelerated, profiling of typical executions will be used
How To Measure	Performance profiler to evaluate the execution time of WRF
Involved EVEREST Components	No component is involved. This was done by profiling the code on the standard infrastructure available at the beginning of the project.
Target Value	At least 20%
Reached Value	Most intensive physics parameterization has been identified in terms of the radiation module (RRTMG parameterization, longwave and shortwave components). A profiling of the code has been done on an Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz

Description:

The exploitation of the EVEREST FPGA accelerators to speed-up the execution of the WRF model is not a trivial task and requires multiple technologies development within the EVEREST SDK. The basic idea is to replace the most intensive components of the WRF Fortran codebase with kernels derived through the SDK developed in the EVEREST project. In order to identify the computational bottlenecks of the WRF model, it is necessary to profile the run of the model against a set of representative physical representations of the atmosphere, e.g. a set of significant forecast predictions as cloudy, sunny, and raining weather conditions. This was considered as a benchmark to be analyzed; the profiling of the application indicated a clear hotspot that calls for further investigations: Rapid Radiative Transfer Model for Global Scale (RRTMG), the radiation driver exploited in standard configurations of the WRF model consumes the largest share of the execution time.

Intermediate Steps:

1. Identification of a limited set of use cases to assess the contribution of the different physics packages to the overall WRF computing time ;
2. Extensive profiling of the execution on an Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz;
3. Analysis of code performance, and selection of the most consuming physics package(s).

Results:

A set of case study conditions was selected, and the profiling of the WRF model was performed. As a result, the RRTMG parameterization kernel was selected as the candidate for the acceleration. This work is successfully concluded and can be considered as a preliminary step, mandatory for the following optimization phase. Figure 2 shows the execution time of the WRF model divided in terms of modules and kernels.

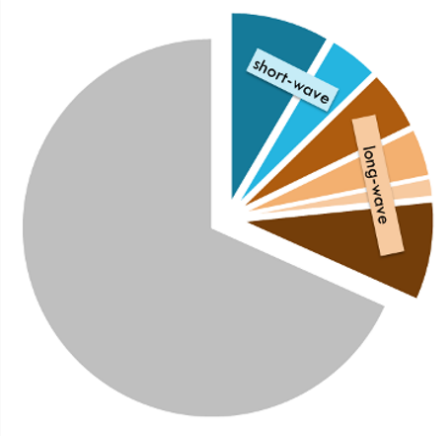


Figure 2 – Execution time of WRF model subdivided in terms of modules and kernels. Only the radiation driver has been highlighted and further split into short-wave and long-wave computations. Profiling on an Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz

ITEM	DESCRIPTION
Method	The optimization of such kernels and models within the EVEREST platform
Priority	Must Have
Baseline	Selected WRF-Kernel CPU version
KPI	Performance and Energy Improvement
Notes	N/A
How To Measure	Comparison between the HPC and accelerated versions of WRF model
Involved EVEREST Components	A plug-in as an extension to the WRF framework to allow substituting standard WRF modules for custom implementations (HW and HW). The EVEREST Kernel Language (EKL), an extension of the CFDlang DSL (see Deliverable D4.5), to express the kernels of RRTMG. Messner, a new compiler front-end and MLIR transformation framework for EKL. Interfaces for HLS tools for hardware generation (i.e.,Bambu and Olympus). Basecamp module for SDK integration.
Target Value	Execution time on FPGA-based target environment, i.e.,IBM resources. Reduction of computing time of around 25%
Reached Value	The achieved speedup resulted to be $4\times$ wrt a high-end CPU (AMD Ryzen 9 7900X3D operating at 3.8GHz) and several orders of magnitude wrt a naive HW design w/o SDK support.

Description:

After identifying the most intensive WRF component, i.e., the Rapid Radiative Transfer Model for Global Scale (RRTMG), the objective has been to adapt this module to the EVEREST technologies. Since the legacy RRTMG scheme is considered outdated and not a worthwhile target for acceleration, it has to be replaced by more modern schemes in the current state-of-the-art weather modeling, the RRTMGP [2] that marks a new milestone in the modernization of the RRTMG radiative transfer model. In particular, as deeply discussed in D6.1, the target has been replacing the gas optics part of the legacy RRTMG scheme, in the form of the new and improved RRTMGP scheme; in this context also the data-flow was pointed out. At this point, the target

for the EVEREST acceleration specifically focused on replacing the gas optics kernel (computation of optical depth “tau”) with its equivalent in the RRTMGP package.

Moreover, as deeply discussed in Deliverable D6.3, minimal modifications were made by implementing a patch for WRF to introduce a plug-in to replace (on demand) the RRTMG radiation modules; this step was also accomplished by identifying a limited set of use cases and specific data retrieval, thus validating the entire system.

Intermediate Steps:

1. Identification of a limited set of use cases for testing phases and specific data retrieval,
2. Identification of a new, more modern RRTM version and link with the WRF code,
3. Acceleration of new RRTM module,
4. Results analysis.

Results:

Both the data retrieval and the plugin have been successfully implemented.

We accelerated the short-wave radiation part of WRF, which corresponds to the CKD kernels (used also in other Global Climate Models). The execution time of the CKD kernels on a server with an AMD Ryzen 9 7900X3D operating at 3.8GHz on 224 points was 160 μs. Running HLS directly on the default implementation produced a HW design with latencies upwards of 1 ms. The hardware design generated from the EKL compiler had a latency of 42 μs. This corresponds to a speedup of $4\times$ compared with a high-end CPU and several orders of magnitude with respect to a naive hardware design w/o SDK support. These results have been obtained without the use of the Bambu HLS engine since the generated code targeted Vivado HLS directly, and there was no time to retarget it to take advantage of the further optimization available in the Bambu engine.

ITEM	DESCRIPTION
Method	The exploitation of the EVEREST platform to support meteorological workflow
Priority	Must Have
Baseline	WRF CPU version
KPI	Availability end-to-end WRF execution with accelerated modules
Notes	N/A
How To Measure	Successful run of the daily workflow
Involved EVEREST Components	Same as in previous table, with the addition of the LEXIS platform.
Target Value	Execution on FPGA-based target environment, i.e.,IBM resources
Reached Value	96% Execution on CPU-based target environment, i.e.,IT4I resources

Description:

The workflow of WRF provides a daily forecast of 48 hours after the assimilation of Italian radar data and IBM Weather Underground meteo station data. The assimilation happens in 3 cycles of 3 hours each. The forecast is guided by GFS (Global Forecast System) dataset provided by NOAA every 6 hours. There is a preprocessing phase performed by using a WRF tool called WPS (WRF Preprocessing SYSTEM) that is needed in order to transform the GFS dataset in a format suitable to be ingested by WRF. This preprocessing phase was implemented inside a Docker container

Intermediate Steps:

1. Execution of the daily workflow on a CPU-based cluster, with delivery on Dewetra platform (a CIMA operated system for real time monitoring, prediction and prevention of flood and wildland fire risks);
2. Execution of the daily workflow on a FPGA-based platform, with delivery on Dewetra platform

Results:

1. In the first 3 months of 2024, after the introduction of Apache AirFlow workflows performed by IT4I, the daily run was delivered with 96% of success. Overall, the daily run was delivered to Dewetra platform for 222 days.
2. The synthesized FPGA module has been available for a couple of months before the end of the project. This implementation was validated using the Vivado simulator. However, despite many efforts, a bug in the Vivado Synthesis tools prevented us from evaluating directly in the hardware preventing us the possibility to finalize the integration within a production environment. Validation runs have been done with the newer RRTMG plugin.

2.1.4 O1.4 - Prediction accuracy comparable to the state of the practice

The objective is to demonstrate the results of the application in terms of forecast accuracy, in the day ahead time horizon, in comparison with the state of the art, represented by the reference provider identified in O1.1.

ITEM	DESCRIPTION
Method	Compare the EVEREST forecasting performance over selected periods and for selected wind farms in terms of deterministic and probabilistic forecasts
Priority	Must Have
Baseline	N/A
KPI	Availability of end-to-end workflow including data assimilation, WRF execution and Wind farm energy prediction
Notes	No complete workflow was available at the beginning of the project
How To Measure	Mean Absolute Error of the power generated for each hour [MWh], normalized by the installed capacity of the wind farm (NMAE).
Involved EVEREST Components	WRF-DA (WRF with data assimilation) + ML app
Target Value	Narrow the gap and possibly outperform the reference provider
Reached Value	Forecast accuracy improvement close to the reference provider (of about 0.8%)

Description:

The full end-to-end workflow has been deployed and it has been possible to compare the results with state of the art provider. The several steps of the workflow are detailed in D6.3, Section 5.1, and are mainly composed by the execution of WRF, collecting the data on the target area and post-process using the ML module developed. The comparison with the reference provider has been done demonstrating the improvement in terms of forecast accuracy for each version, for different periods.

Intermediate Steps:

To reach the final results, several intermediate steps have been done. At the end of the workflow, we first started adopting a pure Kernel Ridge algorithm (baseline), then we included a pre-processing phase, and finally, a Kalman filter method was added.

Month	NMAE BL	NMAE PC	NMAE PC+K	NMAE RP
Jun-21	7,6%	7,4%	7,3%	7,6%
Jul-21	8,8%	8,8%	8,6%	8,2%
Aug-21	7,2%	7,0%	6,8%	7,3%
Sep-21	6,1%	5,8%	5,5%	5,2%
Oct-21	9,3%	8,9%	8,6%	7,0%
Nov-21	9,5%	9,4%	9,0%	8,7%
Dec-21	11,1%	10,8%	10,6%	9,1%
Jan-22	12,2%	12,6%	12,6%	10,9%
Feb-22	10,8%	10,6%	10,4%	9,6%
Mar-22	10,2%	9,8%	9,8%	7,6%
Apr-22	11,9%	11,6%	11,5%	10,1%
May-22	8,7%	8,7%	8,3%	7,7%
Jun-22	8,5%	8,4%	8,0%	7,7%
Average	**9,4%**	**9,2%**	**9,0%**	**8,2%**

Figure 3 – Accuracy improvement in comparison with Reference Provider

Result:

The backtesting analysis is based on the forecast of the energy generated by a wind farm with 34 MW of installed power, using a dataset of WRF data and observation for a period of 13 months. The table below presents the final comparison in terms of forecasting accuracy among the several versions of the ML applications and the **DUF** reference forecast provider (RP):

- NMAE BL: the baseline KernelRidge application
- NMAE PC: the KernelRidge application with the implementation of Power Curve Filter method
- NMAE PC+K: the KernelRidge application with the implementation of Power Curve Filter method and Kalman smoothing techniques, in pre-processing and post-processing components.
- NMAE RP: reference provider

The reached average distance with the reference provider has been of about 0.8%.

ITEM	DESCRIPTION
Method	Implementation of adequate pre-processing components for observational data to be assimilated into the procedure
Priority	Must Have
Baseline	N/A
KPI	Availability of the component
Notes	No data-assimilation component was available at the beginning of the project.
How To Measure	Mean Absolute Error of the power generated for each hour [MWh], normalized by the installed capacity of the wind farm (NMAE)
Involved EVEREST Components	WRFDA+ML app
Target Value	Improve the NMAE respect to the Kernel Ridge method without pre-processing
Reached Value	The NMAE was improved from 9.4% to 9.0% in the selected backtesting period June 2021-June 2022.

Description:

The daily workflow involves the implementation of the hourly generation data of the wind farm related to the day before, as explained in Deliverable D6.4, Section 3.7. These observation data are pre-processed with the WRF data and used for model training.

Intermediate Steps:

To reach the objective we went through the following intermediate steps:

1. Data collection related to the physical power curve of the wind turbines
2. Use the power curve to filter out from the training set the samples that deviate too much from this curve, obtaining a more reliable dataset

Result:

The physical power curve of the wind turbines was collected from the owner of the wind turbines. The training set was filtered using the power curve. The results showed an improvement of the Kernel Ridge performance using this pre-processing technique, as explained in D6.3, Section 5.3.3.

The following histogram (Figure 4) shows the evolution of the forecast error due to the main upgrades of the application:

- the baseline application (BL), with no pre/post-processing
- the baseline application with the Power Curve pre-processing method (PC)
- the baseline application with the Power Curve pre-processing method and the implementation of pre-processing/post-processing Kalman filters (PC+K):

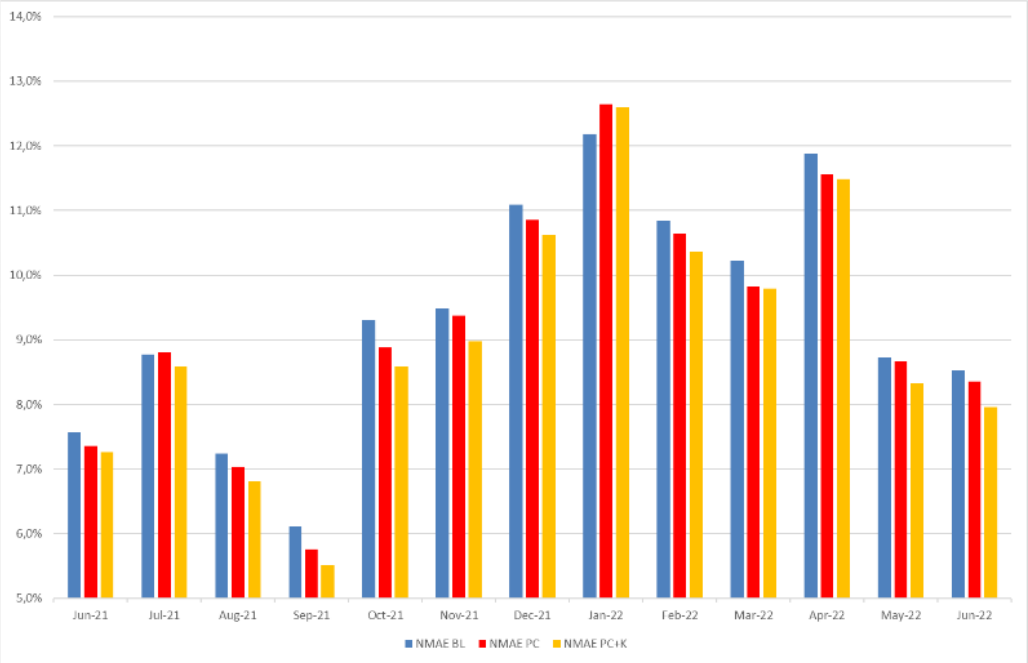


Figure 4 – Impacts of pre-processing on forecasting accuracy

ITEM	DESCRIPTION
Method	Implementation of adequate security policy during observation collection/transfer
Priority	Should Have
Baseline	Workflow without Anomaly Detection module
KPI	% of injected anomalies that are detected and count of anomalies detected in historical data
Notes	The module will be used for detecting possible malicious data ingestion, and for cleaning historical data
How To Measure	Inject anomalies in historical dataset and perform anomaly detection, process historical data and report anomalies
Involved EVEREST Components	AD module
Target Value	21k records (all historical data provided by **DUF**)
Reached Value	21k records

Description:

Referring to the renewable energy use case, anomaly detection is performed on the WRF output data, already elaborated through some pre-processing. It is to take into account the seasonal effects and spatial relations of the dataset that are mandatory to maintain during the anomaly detection; the details of the detection algorithm are reported in Deliverable D3.2. To test the effectiveness of applied methods, it was necessary to actually inject anomalies into the data and validate the detection.

Intermediate Steps:

1. share historical data
2. inject anomalies
3. detect injected anomalies

Results: Anomaly injection methodology has been replicated from earlier anomaly detection results, as described in Deliverable D3.1. Anomaly detection performance has been validated using this final step. Performance is reported in the following: All historical data has been processed. Firstly it has been analyzed for anomalies, which resulted in no significant anomalies being found. This has been described in more detail in Deliverable D6.3. Furthermore, anomalies have been injected to validate the performance of ADLib. Anomalies injected were detected at a good rate. This was validated using a Receiver Operating Characteristic curve, which gives us an idea of the trade-off between true positive rate and false positive rate using different detection thresholds. For example, a threshold which achieves a true positive rate of 0.8 has a false positive rate of roughly 0.25 using a model found through ADLib. In general, an Area Under the Curve of the Receiver Operating Characteristic curve of 0.82 was achieved on anomalies injected using an alpha of 3, which is in-line with previous results. With more extreme anomalies the performance of the model improved as expected, e.g. an alpha of 5 resulting in an AUC of 0.91.

ITEM	DESCRIPTION
Method	Exploiting of the EVEREST computing platform in support of the overall modelling workflow
Priority	Must Have
Baseline	N/A
KPI	Availability of the end-to-end execution of the energy prediction workflow using EVEREST hw/sw platform
Notes	FPGA exploitation at least of one component, complex orchestrator of the different components, data transfer of the daily in-situ data
How To Measure	Rate of successful run of the daily workflow
Involved EVEREST Components	WRF, Pre-processing, Anomaly Detection Module, Machine Learning predictor, post-processing
Target Value	Energy prediction workflow up and running on a daily basis
Reached Value	The end-to-end workflow application is deployed at IT4I (resources of the Karolina supercomputer). The workflow includes: pre-processing, Anomaly Detection Module, ML and post-processing). It's possible to run the workflow from the LEXIS portal too.

Description:

The objective is to operate the full chain of the renewable energy production prediction fully exploiting EVEREST resources and evaluating related performance. Most of the intermediate steps have been already described, from WRF acceleration up to ML prediction while considering anomaly detection. A high-level schematic view of the entire workflow is presented below:

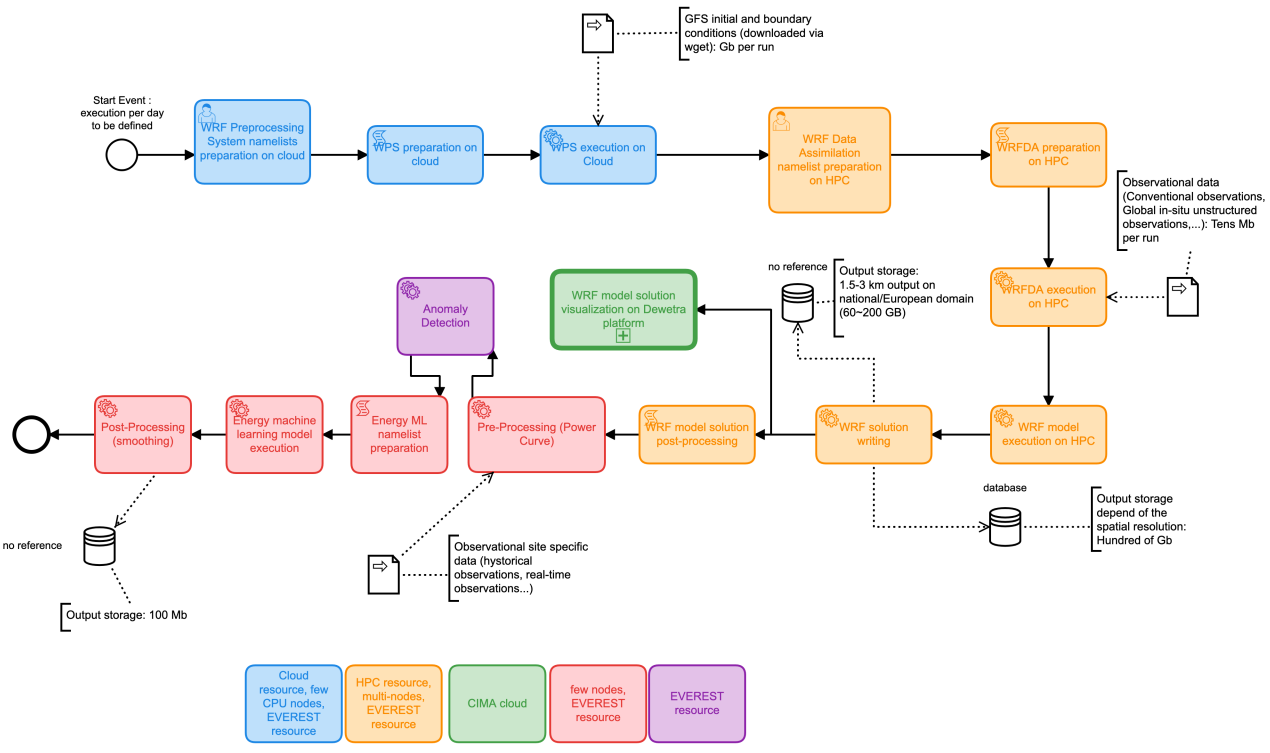


Figure 5 – Full renewable energy prediction workflow

Intermediate Steps:

1. WRFDA (WRF data assimilation) workflow up and running
2. energy prediction component up and running
3. accelerated WRFDA workflow up and running
4. end-to-end energy prediction workflow up and running

Results:

As already reported, after introducing Apache AirFlow workflows performed by **IT4I**, the daily run was delivered with 96% success - reference period first 3 months of 2024.

1. The WRFDA workflow (EVEREST WRF Italy workflow in LEXIS Platform) is executed daily at 3:55 AM (can also be executed manually). In case of failure, it is possible to re-execute each part of the workflow.
2. Energy Prediction part (Pre-Processing, Anomaly Detection, Energy ML) utilize containerization that allows easy execution on Kubernetes or HPC infrastructure. The HPC version of the energy prediction workflow can be executed through the LEXIS Platform as a demonstrator (ML at IT4I workflow).
3. The synthesized FPGA module for the RRTMG plugin arrived only in the last days of the project and there was no possibility to finalize the integration within a production environment.
4. The full renewable energy prediction workflow is running without the accelerated modules (see previous item). It can be executed using the LEXIS portal.

2.2 Air Quality Prediction

The air-quality forecast due to the emission of an industrial site relies on two parts: (i) weather forecast and (ii) atmospheric dispersion forecast.

When the different inputs of the atmospheric dispersion forecast (e.g.: the site's characteristics, the emission) are well known, the quality of this second step depends strongly of the quality of the weather forecast. Moreover, in terms of computation, it is the weather forecast which is the most nodes-consuming with an execution time in hours compared to minutes/seconds for air-quality. In consequence, the EVEREST project focuses on the weather part with two main objectives:

1. Improve the performance of the baseline which is a deterministic forecast performed by **NUM** with the WRF model.
2. Reduce as much as possible the execution time of the WRF forecast.

The following described objectives are pieces to reach these two main targets.

2.2.1 O2.1 - Improved spatial resolution, initialization and forcing of weather forecast

As indicated in the introduction, one main EVEREST objective is to evaluate the impact of a possible FPGA acceleration of WRF. In practice, this work could be done with the pre-existing WRF configuration operated by **NUM** over France, configuration to operate then on EVEREST infrastructure. But it was decided to exploit the EVEREST project to configure a specific WRF forecast simulation over France which could be better than the **NUM** WRF forecast baseline. The main differences of the EVEREST configuration are then:

1. Use as global input, the IFS forecast (Integrated Forecasting System) operated by the European Center ECMWF (European Centre for Medium-Range Weather Forecasts), instead of the GFS forecast operated by the US NOAA center. Over Europe, it is considered that ECMWF forecast is better than the NOAA forecast, but the access is not the same since it is free cost for the GFS and paid cost for the IFS. Anyway, test the IFS use (freely accessible in the framework of a research project like EVEREST) for the **NUM** application is important to evaluate the interest in future. It is even more important that recently ECMWF announces in 2024 plans to have some forecasts in open data.
2. Use an assimilation procedure based on 3DVAR (three-dimensional variational data assimilation) procedure instead of single nudging at borders of the domain. For this assimilation, the observed data use are also different. It was initially expected to Wunderground surface stations and the MeteoFrance precipitation radar measurement.

To achieve this objective, the starting point was the existing WRF workflow developed by **CIMA** and **IT4I** during the LEXIS project[1], and operated by **IT4I** on its infrastructure. But, it was requested first by **IT4I** to rework this workflow and its management (see objective O.2.4). The adaptation of the new workflow to the France use case has been done in a second step, explaining the achievement date presented below.

[1] LEXIS project: https://lexis-project.eu/web/

ITEM	DESCRIPTION
Method	Pushing to the limits of meteorological predictions with a fine resolution (at least 1-3 km) to be more representative of local predictions, add assimilation procedure, add IFS forcing
Priority	Must Have
Baseline	**NUM** forecast at 3km without data assimilation and GFS forcing
KPI	Improvement of accuracy of weather (wind speed and direction mainly) forecast
Notes	Need to simulate new WRF simulations for some dates to compare to baseline which is the current **NUM** forecast
How To Measure	Comparison of RMSE (Root Mean Square Error) on wind direction, wind speed and air temperature between baseline and EVEREST execution
Involved EVEREST Components	New WRF-IFS workflow for France deployed on EVEREST servers
Target Value	Improvement on RMSE between 10 to 20 %
Reached Value	The WRF workflow has been adapted for France domain in order to integrate modification done on assimilation and IFS download parts. Daily automatic forecasts are executed on EVEREST infrastructure to generate WRF-IFS forecast over France. About RMSE, the improvement is in average: 1. For temperature, around 2% for the first 24 hours of forecast, and 6% for the second day of forecast. 2. For wind direction, around -2% for the first 24 hours of forecast, and 1% for the second day of forecast. 3. For wind speed, around 10% for the first 24 hours of forecast, and 13% for the second day of forecast.

Intermediate steps:

1. Data assimilation procedure: For the assimilation, the first task was to validate the observed data use in the procedure. At the start, the objective was to use the MeteoFrance radar observation. But it appears that it was not possible to get an easy research access during the EVEREST project, and the commercial cost was outside the planned budget. At the end, it was decided to not use this data during the assimilation, which can be understandable since **NUM** will never be in position to buy such data compared to the air-quality price market accepted by final customers. The consequence was for **CIMA** to change the configuration of the pre-existing assimilation procedure for France domain. This was performed in July 2023 with the production of new WRFDA execution script developed by **CIMA**.
2. IFS data: Initially it was expected no specific work on this part compared to the pre-existing LEXIS workflow. But, in 2022, ECMWF informs us that the specific WCDA api developed by ECMWF during the LEXIS project will not be maintained. It was requested to use the official ECWMF API (`https://www.ecmwf.int/en/forecasts/accessing-forecasts`) to access the forecast (which remains free for research activity). In august 2023, the new download script (including extraction of parameters requested for the France execution) was released by **CIMA** for integration in WRF workflow by **IT4I**.
3. Adaptation of the WRF-IFS workflow for France: Based on the new WRF workflow using Airflow [1], **IT4I** adapts it to integrate the new assimilation procedure and the new IFS download procedure. The workflow

execution was also configured according to the need of the air-quality use case: from the beginning of 2024, a daily execution of the new WRF-IFS workflow for France is activated on EVEREST infrastructure.

Result:

The baseline is the **NUM** forecast at 3km horizontal grid (2.5km for EVEREST WRF-IFS) with observational and analysis nudging of surface data for assimilation and use of GFS for global forcing. The target KPI is an improvement of accuracy of weather forecast (wind speed and direction mainly) of the new EVEREST WRF-IFS configuration for France, with an expected improvement on RMSE between 10 to 20%. Since operational execution starts at the beginning of 2024, the performance is calculated only for around 3 months, during winter season. The comparison is performed on 56 MeteoFrance weather stations.
The following table presents the global results for 3 parameters: wind speed, wind direction and air temperature. For each parameter, it is indicated:

- the variation of RMSE (-10% means a decrease – so an improvement – of the RMSE),
- for the first 24 hours of forecast (Day J) and for the next 24 hours of forecast (Day J+1),
- on average (for all stations), the best individual result and the worst individual result.

Day of forecast	Result	Wind Speed	Wind direction	Temperature
Day J	Average	-10.3%	+2.1%	-2.1%
Day J	Best result	-51.0%	-14.5%	-27.7%
Day J	Worst result	+62.5%	+41.1%	+28.2%
Day J+1	Average	-13.5%	-1.0%	-5.8%
Day J+1	Best result	-48.1%	-20.6%	-23.2%
Day J+1	Worst result	+55.3%	+44.0%	+19.3%

Table 1 – Variation of RMSE for first and second days of forecast for 56 weather stations

For wind speed, the average improvement is around 10% which is in the limit of the expected KPI. For wind direction and air temperature, the average performance is close to zero (+2% and -2% respectively) for the first 24 hours of forecast. For the forecast from 24 to 48 hours, we observe a light improvement on RMSE of -1% for wind direction and -6% for temperature in average. When we observe the extreme results, we clearly see a large dispersion of the score with a RMSE which can be improved by 51% for one station on wind speed, but also degraded by 62.5% for another station (on wind speed too). This can be observed also on the next figure, presenting for Day J+1, the distribution of the RMSE's variation for wind speed, wind direction and air temperature obtained for each individual station.

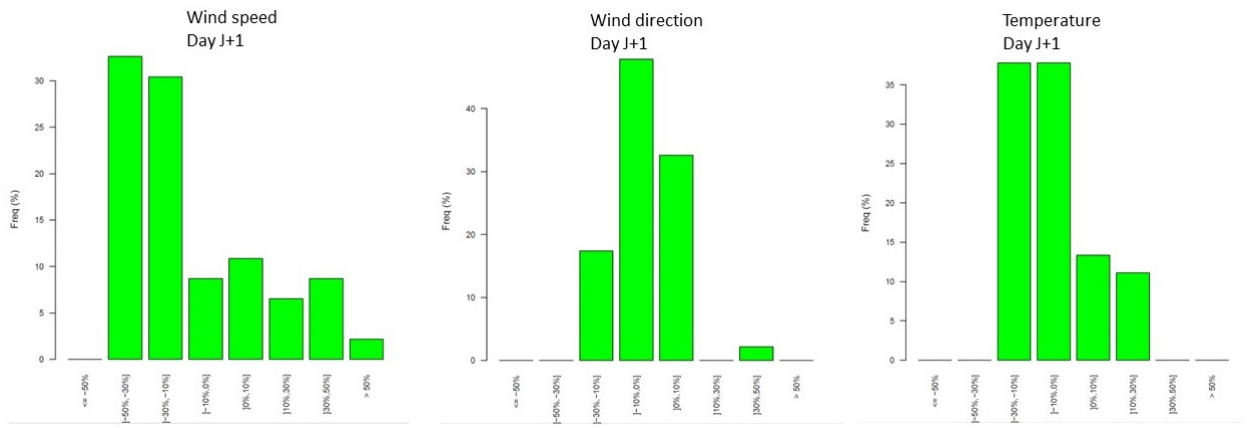


Figure 6 – Distribution of RMSE's variation for wind speed, wind direction and wind temperature for forecast Day J+1 (24 to 48 hours) for 56 weather stations

Clearly, we see for the wind speed, a distribution much more concentrated on an improvement of the RMSE (explaining the 10% of improvement in average), but for some stations it is a degradation. For wind direction and air temperature, it is more a mix of improvement and degradation according to the stations, explaining the average performance close to zero. We can note also that, when we consider the spatial distribution of the score, we don't observe a clear geographical impact. For example, all stations with an increase of RMSE are not located in the same type of topography such as mountains areas and close to Mediterranean Sea which are known as regions more difficult to simulate. We have stations with bad results even at the North of France or close to Atlantic ocean coast.
It is difficult to explain these results at this stage, especially due to the limited period of forecast:

- Do for any reason the GFS forecast (used in **NUM** forecast) better than IFS forecast (used in EVEREST forecast) for this period?
- Do the data assimilated in **NUM** forecast better than those in EVEREST forecast for this period?

Compared to the expected initial KPI, only for wind speed we can consider it is reached. Nevertheless, this KPI is not the main one for the expected improvement on weather forecast. The objective O2.3 based on AI approach is most important. We can conclude that the quality of the EVEREST WRF-IFS forecast for France is sufficient to be exploited for the others objectives.

2.2.2 O2.2 - Effective run of the service in terms of computational performance efficiency with a specific focus on the WRF workflow

The content of this subsection is the same as for objective O1.3 since it is related to WRF, which is a shared component across the two Use Cases.

2.2.3 O2.3 - Moving towards ensemble prediction and AI machine learning approach

As described in Deliverable D6.3, the objective is to apply a machine learning approach to produce locally a better weather forecast than a standalone WRF deterministic simulation. The developed AI approach consists of aggregating an ensemble of weather forecasts using local weather measurement as a forcing parameter of the aggregated prediction. This novel approach for **NUM** was developed during the EVEREST project. The different members used for the aggregation are those accessible by **NUM**: local forecast such as the **NUM** WRF forecast and global forecast such as the GFS forecast. In Deliverable D6.3, results obtained during the development are presented. In this section are presented the final results adding a last member which is the EVEREST WRF-IFS forecast for France.

ITEM	DESCRIPTION
Method	Design IA method to exploit fine and coarse meteorological prediction forced by local observation and evaluate its performance
Priority	Must have
Baseline	No ensemble model available in the baseline implementation
KPI	Improvement of the quality of weather prediction due to ensemble approach
Notes	Final ensemble model is built considering **NUM** forecast, EVEREST WRF forecast, MeteoFrance forecast and GFS forecast
How To Measure	Comparison of RMSE on wind direction, wind speed and air temperature between baseline execution and the aggregation output
Involved EVEREST Components	New ML air-quality model
Target Value	Improvement of 50% on the RMSE.
Reached Value	The improvement on RMSE is: 1. Around 48% in average for temperature on the first 24 hours of forecast, and 57% for the best hour. 2. Around 45% in average for wind speed on the first 24 hours of forecast, and 57% for the best hour. 3. Around 15% in average for wind direction on the first 24 hours of forecast, and 29% for the best hour.

Intermediate steps:

1. Development of the machine learning model: different mathematical approaches were tested, especially according to the parameters. Indeed, if for temperature the temporal evolution is generally continuous, the transition can be more disturbed for the wind, and even more for the precipitation with long periods without events and some instantaneous peaks. We finally decided to focus on temperature and wind. The best found configuration uses a Ridge regression with discounted loss, with objective to minimize the RMSE (root mean square error) between the aggregated prediction and the observation over a period. During the development, we also tested and defined how much previous cycles of a same model execution are interesting to exploit. For example, for a forecast of day D starting at H00, do we use only the **NUM** WRF forecast of day D H00, or also the cycle day D-1 H12, the cycle day D-1 H00, etc. Lastly, the machine learning approach uses an incremental training window, that is to say that at each execution the training is performed using last historical data. A first exploitable model was available in august 2022.
2. Collection of historical data: Even if we use an incremental training window, the best is to start with a longest period as possible. This requires to collect and store the numerical weather forecast for each targeted local weather measurement site. At the start of the project, it was envisioned to evaluate the performance only on the two or three locations concerned by the final air-quality forecast. At the end, since it is here only an evaluation on weather parameters, we decided to use the same MeteoFrance weather stations for this evaluation. If the collection and storage of the **NUM** WRF forecast (baseline) and the GFS forecast was already performed by **NUM**, it was necessary to collect and store the MeteoFrance measurement and the MeteoFrance forecast on **NUM** storage servers. This was totally operational at the end of 2022. During 2023, **NUM** explored the way to increase performance for wind direction and tried to apply the method on precipitation.
3. Perform first evaluation without EVEREST WRF-IFS forecast: Deliverable D6.3 presents the results of the

performance obtained at the beginning of 2023, for the application of this approach without the EVEREST WRF-IFS forecast on one year. The results are an improvement of the RMSE between 40% and 50% for the temperature, from 44% (for 1st hour of forecast) to 26% (for forecast at 24 hours) for the wind speed, and from 31% (for 1st hour of forecast) to 8% (for forecast at 24 hours) for the wind direction.

4. Produce EVEREST WRF-IFS forecast: the operational production of forecast for France the EVEREST WRF-IFS workflow started at the beginning of 2024.

Result:

The next table presents the impact on RMSE of the ML air-quality model considering all members, except the EVEREST WRF-IFS forecast, compared to the initial **NUM** WRF forecast, for the first three months of 2024.

Day of forecast	Result	Wind Speed	Wind direction	Temperature
Day J	Average	-44.4%	-14.5%	-44.5%
Day J	Best Hour	-54.8%	-26.5%	-53.4%
Day J	Worst Hour	-35.4%	-8.0%	-18.8%
Day J+1	Average	-24.2%	+0.6%	-4.3%
Day J+1	Best Hour	-29.1%	-8.5%	-13.4%
Day J+1	Worst Hour	-15.9%	+7.0%	+5.0%

Table 2 – Variation of RMSE for first and second days of forecast for 56 weather stations when WRF-IFS forecast is not included

We can observe:

- for temperature: we have a clear improvement for the first 24 hours of forecast, with in average an improvement of 44.5% of the RMSE. The scores are similar to the ones obtained on the full year (2022). For the second day of forecast, as for the full year, the improvement is not so much visible, in average of 4%.
- for wind direction: the performance are lower on the first three months of 2024 compared to 2023. For example for the first day of forecast, the improvement in average is close to 14% compared to around 19%.
- for wind speed: the impact is better for the three months of 2024 with an improvement in average of 44% compared to 35%. For the second day of forecast, we observe also a clear improvement to apply the ML approach, with an average improvement on RMSE of 24%.

We must remember that the developed approach considers a daily training of the ML model integrating the last forecast day, so the score of one year is necessary biased compared to three months of use. Moreover the meteorological comportment is probably also different.
Now, when we integrate the EVEREST WRF-IFS forecast in the ML approach, the impacts on RMSE are the following ones for the three first months of 2024.

Day of forecast	Result	Wind Speed	Wind direction	Temperature
Day J	Average	-0.6%	-1.1%	-3.5%
Day J	Best Hour	-2.1%	-3.1%	-3.4%
Day J	Worst Hour	-0.6%	+0.7%	+1.1%
Day J+1	Average	-1.9%	-1.4%	-3.7%
Day J+1	Best Hour	-2.2%	-0.7%	-4.5%
Day J+1	Worst Hour	-1.4%	+0.7%	-4.3%

Table 3 – Impact on the variation of RMSE for first and second days of forecast for 56 weather stations when WRF-IFS forecast is included

The addition of the EVEREST WRF-IFS forecast does not improve strongly the RMSE, but again we are here only on three months. Also, the performance of EVEREST WRF-IFS (see objective O2.1) is not so much different from the baseline (**NUM** WRF forecast), and adding a new member can improve the ensemble result only if this member adds better information at some instants. Anyway, in average, we have an improvement around 3 to 4% for temperature. For wind direction, it is around 1 to 2%, whereas again for wind direction, it is not so marked, excepted during the first 24 hours of forecast.

Compared to the initial KPI with an expected 50% improvement on RMSE, and if we focus on the first 24 hours of forecast, we can consider that:

- for temperature: we reach it, with an average improvement of 48% and of 57% for the best hour.
- for wind direction: we don't reach it on this complex parameter, with an average improvement of 15% and of 29% for the best hour.
- for wind speed: we reach it, with an average improvement of 45% and of 57% for the best hour.

The next table presents a secondary objective associated to the machine learning work during EVEREST. The initial idea was to exploit a WRF ensemble produced by EVEREST, in case of demonstrate strong acceleration of WRF execution using FPGA.

ITEM	DESCRIPTION
Method	Exploiting of the EVEREST computing platform in support of the ensemble prediction
Priority	Could have
Baseline	No ensemble model available in the baseline implementation
KPI	Capability of the EVEREST platform to generate WRF ensemble models
Notes	Running also multiple WRF forecast for the ensemble requires more resources than what are available from the prototype platform of the project
How To Measure	RMSE Comparison
Involved EVEREST Components	WRF FPGA acceleration
Target Value	Extra reduction of RMSE by 25% to 50%
Reached Value	none

Intermediate steps:

1. Requires objective O.2.2 to be reached: FPGA acceleration on full WRF execution for France was not available.
2. Requires objective O.2.3 to be reached: as described above, the first operational execution of the EVEREST WRF-IFS forecast for France started at the beginning of 2024.

Results:

First, the machine learning approach used for the air-quality use case, compared to the energy use case, is by construction an ensemble approach that considers external WRF forecasts (the **NUM** WRF forecast, and we can consider the MeteoFrance local forecast as a WRF forecast). So the demonstration to exploit an ensemble of local WRF simulations is already done with the previously achieved result. Second, for different reasons, the EVEREST WRF-IFS forecast workflow was deployed and executed in a second step after the EVEREST

WRF-GFS forecast for Italy. Indeed, it was decided to concentrate the effort of **CIMA** and **IT4I** to configure and operate a WRF ensemble using the EVEREST workflow and infrastructure for the Italy domain and to test it in the energy use case as it is presented in deliverables D6.3 and D6.4. Lastly, it was not possible to test an end-to-end FPGA acceleration for the full WRF-IFS workflow.

2.2.4 O2.4 - Predictive capability better than the current service

This global objective covers three sub-goals:

1. The development and management of an EVEREST WRF workflow using FPGA acceleration, and its evaluation.
2. The development and management of the full chain of execution concerning weather forecast, and its evaluation.
3. The exploitation of weather forecast for the final air-quality forecast, and its evaluation.

Each of them is described and discussed independently below.

2.2.4.1 EVEREST WRF workflow with FPGA acceleration

Inside the full end-to-end workflow for air-quality forecast, WRF forecast takes around 95% of the global computation. FPGA acceleration of the most intensive WRF components could improve the overall execution time performance.

ITEM	DESCRIPTION
Method	Exploiting of the EVEREST computing platform in support of the WRF modelling workflow
Priority	Must Have
Baseline	**NUM** WRF forecast operates on **NUM** HPC server with CPU only
KPI	Speedup of the forecasting with possible use of FPGA acceleration
Notes	N/A
How To Measure	Full CPU time execution of the same WRF simulation without/with FPGA acceleration
Involved EVERST Components	A plug-in as an extension to the WRF framework to allow substituting standard WRF modules for custom implementations (HW and HW). The EVEREST Kernel Language (EKL), an extension of the CFDlang DSL (see Deliverable D4.5), to express the kernels of RRTMG. Messner, a new compiler front-end and MLIR transformation framework for EKL. Interfaces for HLS tools for hardware generation (i.e. Bambu and Olympus). Basecamp module for SDK integration
Target Value	Speed up of factor 2
Reached Value	None. Only performance for acceleration of the selected WRF's kernel was achieved (see O1.3). Given that we focused on the CKD kernel within the radiation module, it is estimated that the maximum achievable speedup is 10% (see dark blue slice in Figure 2).

Intermediate steps:

1. Rework the LEXIS WRF workflow: In practice, **IT4I** has decided to change the orchestration part of LEXIS Platform (responsible for running workflows) from the ATOS Yorc to Apache Airflow solution. This new workflow and its deployment on the EVEREST infrastructure was available at the beginning of the second semester of 2023, using the energy use case as baseline. In a second step, its adaptation to the air-quality use case for France domain was performed as described in O.2.1.
2. Management of the FPGA execution inside the EVEREST WRF-IFS workflow with offload exchange between HPC infrastructure at **IT4I** and FPGA infrastructure at IBM. The demonstration of the management execution of FPGA between **IT4I** and IBM servers was done during the second semester of 2023.

Result:

The synthesized FPGA module has been available for a couple of months before the end of the project. This implementation was validated using the Vivado simulator. However, despite many efforts, a bug in the Vivado Synthesis tools prevented us from evaluating directly in the hardware, and thus from integrating the design within a production environment. Validation runs have been done with the newer RRTMG plugin.From the results described in O1.3 for the kernel's acceleration and considering the execution time of this selected kernel in the full WRF execution it is estimated a speedup of 10% for the full WRF execution.

2.2.4.2 Full chain to produce weather forecast

The objective here is to operate the full chain of weather production as presented in the following figure (this figure includes the final part of the use case which is the air-quality simulation) and evaluate its performance.

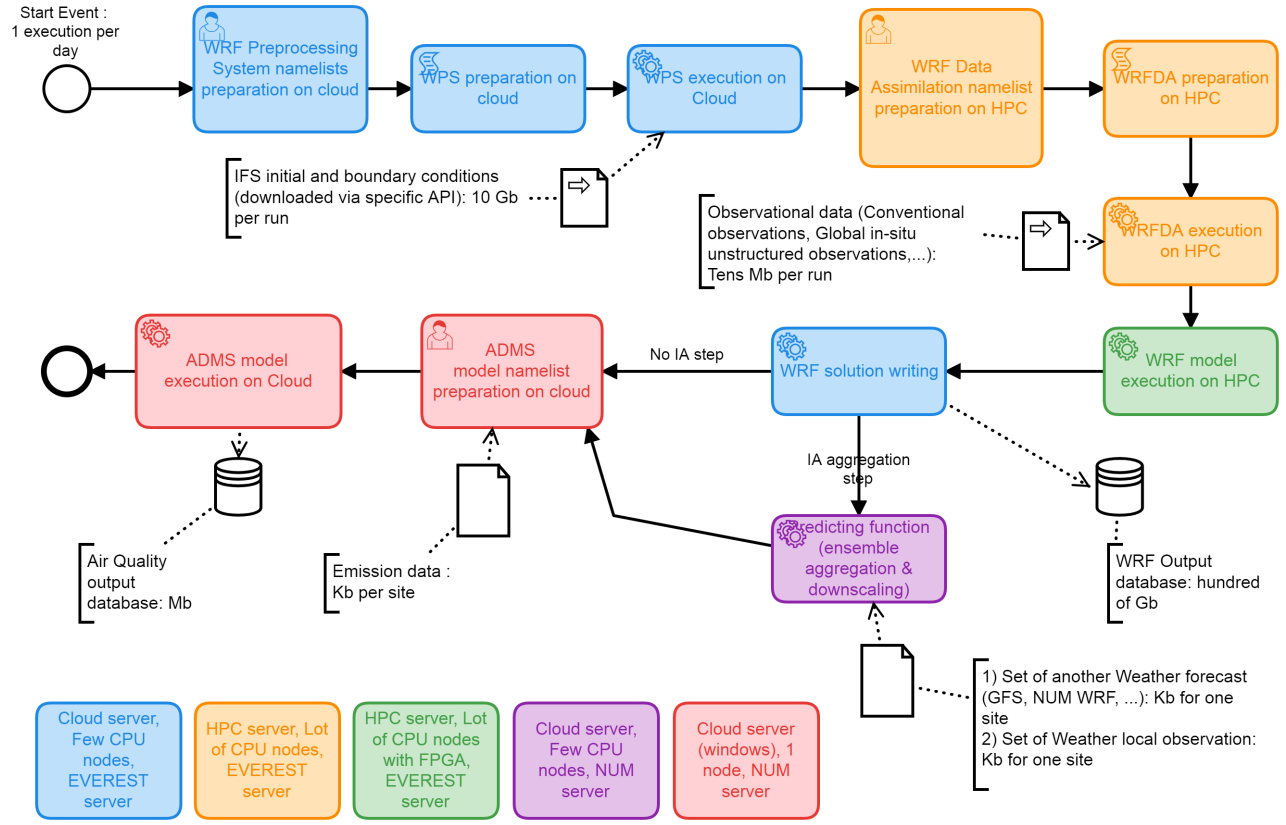


Figure 7 – Full air-quality forecast workflow

The full chain covers the WRF simulation on EVEREST infrastructure, the transfer of the outputs on **NUM** server, and the execution of the ML air-quality model to produce local weather forecast at specific sites.

ITEM	DESCRIPTION
Method	Exploiting of the EVEREST computing platform in support of the overall modelling workflow
Priority	Must have
Baseline	Original Forecasting on **NUM** resources without ensemble approach and no anomaly detection
KPI	Availability of the end-to-and weather prediction workflow that exploits the EVEREST WRF-IFS simulations including the IA approach developed in O.2.2
Notes	
How To Measure	Comparison of execution time and Comparison of RMSE on wind direction, wind speed and air temperature between the baseline and the full EVEREST executions.
Involved EVEREST Components	new EVEREST WRF-IFS workflow, WRF FPGA module, Anomaly detection module, new ML air-quality model
Target Value	Keep performance of individual component, that is to say improvement of RMSE of 50% coupled to a speedup of a factor 2 on the full chain.
Reached Value	Demonstration of FPGA acceleration is not reached (see first item of O.2.4). Concerning RMSE (see O.2.3), we can consider that the target is obtained with an improvement in average around 50% for temperature and wind speed. Demonstration of the possible interest to use EVEREST WRF workflow on **IT4I** infrastructure (CPU only) compared to **NUM** infrastructure.

Intermediate steps:

1. Operation of EVEREST WRF-IFS workflow on EVEREST infrastructure: see details in objective O.2.1 for execution without FPGA and objective O.2.2/O.2.4.1 for execution with FPGA. The daily operation is active since the beginning of 2024.
2. Transfer of EVEREST outputs to **NUM** servers: as described in deliverable in D6.4, (i) Py4Lexis library is implemented for the data downloading from LEXIS DDI storage (storage area of the EVEREST infrastructure for WRF outputs) to **NUM** server; After this transfer, (ii) the EVEREST anomaly detection ADlib is applied to check the validity of this data before to apply the ML air-quality model. These steps are active at the end of 2023.
3. Production / collection of additional weather forecast: see details in objective O.2.3 about the collection of additional weather forecast used by the ML air-quality model. It is operational at the end of 2022.
4. Operation of AI ensemble air-quality model on **NUM** server: see details in objective O.2.3. It is operational at then end of 2022 without EVEREST WRF-IFS outputs and from the beginning of 2024 with the EVEREST WRF-IFS outputs.

Results:

1. Performance of weather forecast:
   Logically the performances observed for this objective are those observed in the individual objective O.2.3 with exploitation of the new WRF-IFS workflow and the air-quality ML module. For wind speed and temperature, the objective of an improvement of RMSE of 50% is reached. For wind direction, which is more complex to simulate at local scale at ground surface, the average improvement of RMSE is only between 15 and 29%.

2. Execution time:
   First, we can mention that the use of anomaly detection library conducts to increase the full execution time of 4 minutes which is reasonable compared to the WRF execution (see below). Secondly, the use of ML air-quality model for ensemble aggregation is negligible with an execution time less than one minute (see deliverable D6.4).
   In consequence, the initial objective in terms of execution time relies on speed up of WRF due to FPGA acceleration. As presented in objectives O.2.2 and O.2.4.1, if acceleration of individual kernel of WRF is shown, the evaluation in an end-to-end full WRF execution was not possible. It is estimated a speedup of only 10% of full WRF execution time. This is far away the initial target of factor 2.
   We can mention here an intermediate result for **NUM** which concerns the evaluation of the execution performance of the new EVEREST WRF-IFS workflow operated on EVEREST infrastructure compared to the baseline (**NUM** WRF workflow on **NUM** server). Indeed, beyond the question of the weather performance, it is interesting for **NUM** from operational point of view to see if the EVEREST management and execution chain can be used in future commercial exploitation. Considering the infrastructure used for the **NUM** baseline and for the EVEREST one with CPU only (see deliverable D6.4), the execution time (for 48 hours forecast) is around 3h20 for the baseline compared to 2h / 1h40 for the EVEREST configuration. Of course, the comparison is biased due to the fact that the configuration of the two WRF executions is not the same, and the number of nodes used is different also. Nevertheless, this result shows that the new EVEREST management of the workflow on distributed environment does not lead to important additional execution time, which was not the case in the previous LEXIS project (https://lexis-project.eu/web/). Moreover, the new WRF-IFS workflow allows to manage each part independently, such as for example manage in advance the download of input data and manage WPS to prepare WRF input for any WRF runs which can use it, or a better management in queue for operational execution every day. Future work for **NUM** with **IT4I** is to consolidate additional points (cost use of CPU, way to change easily WRF configuration, etc.) in order to explore the final interest to use **IT4I**'s service for WRF exploitation.

2.2.4.3 Final air-quality forecast

This is the final objective of the EVEREST project for the air-quality use case. Based on expected improvement on weather forecast, the goal is to improve the performance of air-quality forecast of emission dispersion from industrial site.

ITEM	DESCRIPTION
Method	Statistical assessment of the added value of EVEREST forecasting and ensemble forecast on simulated air quality
Priority	Should have
Baseline	Original Deterministic **NUM** Forecasting
KPI	Improve the quality of the forecasts of real pollution events
Notes	Large set of historical data and runs should be used within a selected period for a robust statistical assessment
How To Measure	Reduction of number of forecasts of false pollution events and increase of number of forecasts of real pollution events
Involved EVEREST Components	All weather forecast components, end-to-end workflow with air-quality execution
Target Value	25% improvement of quality of the air-quality forecast
Reached Value	No available statistics. However, the execution is operational (see comments on results).

Intermediate steps:

1. Identify industrial sites for demonstration: For demonstration, it is important to have industrial sites for which (i) ML air-quality model can be applied, that is to say that local weather measurement is operational and available, (ii) emission forecast input has limited uncertainties in order to limit performance air-quality forecast to only the weather forecast, and (iii) availability of parameters to evaluate statistically the performance of the air-quality forecast such as pollutant air measurement or survey of pollution event. At the start of the project, Unfortunately, TOTAL energies company due to its recent deployment in energy market decided to not let **NUM** uses one of their sites for the EVEREST project due to presence of DUF in the project. In 2021, a list of possible industrial sites was established, and operational collection of data for the demonstration was activated.
2. Prepare and adapt **NUM** air-quality forecast application: As described in deliverable D6.4, the main core of air-quality forecast has been isolated in order to simplify execution. Configuration of envisioned sites in ADMS model has been done and scripts has been prepared to execute air-quality forecast based on deterministic **NUM** weather forecast, deterministic EVEREST WRF-IFS forecast, or on the ML air-quality model. Daily operational execution was possible at the end of 2022.
3. Availability of operational EVEREST WRF-IFS forecast for France: see objective O.2.1 for details, the daily production of the EVEREST WRF-IFS forecast for France is available at the beginning of 2024.

Results:

Unfortunately, it is not possible to produce statistics on the air-quality forecast in link to the late availability of EVEREST WRF-IFS forecast:

- At the beginning of 2024, on the 4 identified industrial sites: (i) one site is no more followed by **NUM**'s system due to the evolution of the site with investment done in 2022 and 2023 on emission treatment system which renders the **NUM** application obsolete, (ii) one another site is in long period of maintenance, and (iii) one site has malfunction concerning its weather measuring station so exploit the ML aggregation model is not possible.
- It was decided at the start by **NUM** to focus the EVEREST demonstration on daily operational execution and not to perform past simulations for the evaluation, which will necessitate configuring workflow and running historical EVEREST WRF-IFS simulation for simulating past events with sufficient outputs for training before these events. Normally, the objective to have an EVEREST WRF-IFS forecast in operation during the year of 2023 will be sufficient to cover a large period for demonstration. Beyond the fact of having 3 months of daily execution, for the remaining identified site, the winter season is not the best season in terms of pollution events, and so to evaluate air-quality performance.

Today, no statistic is available to evaluate the final performance on air-quality forecast. Nevertheless, the main objective for **NUM** of the EVEREST project is first to have an improvement on the weather forecast, which is reached and shown after in application of ML air-quality model (see objective O.2.3). Secondly, since air-quality forecast strongly depends on the weather forecast, it is reasonable to say that the observed results on weather forecast will necessarily lead to an improvement of air-quality simulation. In practice, due to this good performance of the ML approach, **NUM** has started to apply it for some specific clients outside the EVEREST project, and as indicated in deliverable D7.7, its exploitation is clearly planned by **NUM**.

2.3 Traffic Modelling

For the traffic modelling use case **SYG** and **IT4I** ambitions were to optimize and accelerate several parts of the traffic modelling eco-system. The main goals were to a) accelerate machine learning calculation of traffic prediction model operating on a big data as well as the application of prediction inference function used intensively by other subsystems; b) accelerate data pre-processing part, which deals with ingesting big data while performing costly AI map matching operations; c) accelerate critical parts of the traffic simulator, which is used for data boosting in the process of learning the traffic prediction model; d) accelerate (alternative) route calculation. As such, the EVEREST project focused on the traffic modelling part with five specific objectives:

1. Improve the overall performance of the traffic simulation model.
2. Improve the overall performance of neural network traffic prediction model training.
3. Improve data management and computational services and reduce programming effort.
4. Improve the overall performance of map matching kernel.
5. Improve the overall performance of the probabilistic time-dependent routing Kernel.

The following described objectives are pieces to reach the use case targets.

2.3.1 O3.1 - Improve the overall performance of the traffic simulation model

In order to fulfill Objective 3.1, specific methods have been defined and evaluated. A total of 4 methods have been drafted:

1. Optimized stream of queried data from big data sets into processing elements.
2. Exploitation of an external traffic prediction service for each road.
3. Evaluation of the performance for traffic simulation test case.
4. Improve overall system precision by incorporating weather data from WRF model.

ITEM	DESCRIPTION
Method	Optimized stream of queried data from big data sets into processing elements
Priority	Must Have
Baseline	Traffic simulator without the service for data preprocessing and aggregation of FCD (Floating Car Data) data.
KPI	Availability of the module
Notes	Module to processes FCD data collected from a real environment to calculate speeds for each road segment.
How To Measure	Module accessible/callable from Traffic simulator
Involved Everest Components	Accelerated Map-match component
Target Value	Service for FCD data preprocessing and aggregation is developed
Reached Value	Service for FCD data preprocessing and aggregation is available

Description:

This item relates to the development of a module that processes FCD data collected from a real environment and calculates speeds for each road segment. The speed information is then made available to Traffic simulator through a service. In the future, traffic simulator may optionally use it for augmenting the origin-destination matrix based speed inference calculation further to improve functional precision by calibrating it with another in-field sensory information.

Intermediate steps:

- Prepare script to execute Map-Matching processing which converts FCD data into road segment speeds, and exports it in a CSV file format into a date/time organized folder structure.
- Make CSV files available for traffic simulator for download and use.

Results:

- Road segment speeds data is an output of Map-match service preprocessing execution (see Objective O3.4)
- The output is available in Results folder on the path as per configuration of the VM environment executing Traffic modeling Eco-system
- Output files are organized in the form of: one CSV file per day, while each file contains speeds on potentially all roads segments in all time instances across 24 hour time frame
- Traffic simulator granted direct access to the folder structure and can retrieve the data from it in one request as needed

ITEM	DESCRIPTION
Method	Exploitation of an external traffic prediction service for each road
Priority	Should Have
Baseline	Traffic simulator without the neural network Traffic Prediction
KPI	Availability of the integration with a neural network Traffic Prediction
Notes	Before the project, the non-deterministic version of the traffic simulator used only PTDR algorithm. The new deterministic version of the simulator has to integrate both PDTR and Traffic Prediction.
How To Measure	Module Accessible/callable from Traffic Simulator
Involved Everest Components	Accelerated Traffic Prediction Module
Target Value	API for traffic prediction engine is available
Reached Value	API for traffic prediction engine is available

Description:

Traffic prediction is implemented as a service, which in one request provides 1-hour prediction for all major roads in city. The service is addressed within the Objective O3.2 for accelerated optimization. The optimized version of Traffic prediction has two options a) being deployed as a cloud service b) used as a library function. While the service has a general use, Traffic simulator is the one which can use the service intensively as an option further to increase its functional quality.

Intermediate steps:

- Implement and provide API for querying Traffic Prediction engine

- Traffic Prediction service is deployed and can be tested through LEXIS Platform

Results:

- Traffic simulator is ready to utilize accelerated Traffic prediction service of both variants: cloud-based and bus-attached FPGA
- Traffic prediction service as cloud service option is being deployed on **IBM** cloud-FPGA platform using zero-MQ socket implementation
- Traffic prediction as a library function executable on FPGA bus-attached architectures can be utilized when Traffic simulator is being run on **IBM** platform.
- Cloud-based service can be run through LEXIS Platform as a demonstrator.

ITEM	DESCRIPTION
Method	Evaluation of the performance for traffic simulation test case
Priority	Must Have
Baseline	50K vehicle on a single instance of the original traffic simulator
KPI	Number of vehicles possible to simulate thanks to EVEREST improvements
Notes	The baseline refers to the maximum amount of vehicles that was possible to simulate before the improvements carried out within the project
How To Measure	Simulator capability to handle defined number of vehicles
Involved Everest Components	Traffic simulator, `evkit`
Target Value	Execute traffic simulation with 100,000 (100K) vehicles with `evkit` distribution
Reached Value	Executed traffic simulation with 250,000 (250K) vehicles with `evkit` distribution

Description:

The performance of the traffic simulator is bottlenecked by two kernels - the computation of alternative routes and the Monte Carlo simulation kernel (also called PTDR kernel). EVEREST optimizations have improved the performance of the simulator by introducing the ability to offload the computation of these two kernels to multiple CPU cores and also to multiple nodes in a cluster, using the `evkit` tool, which was developed within the EVEREST project[2]. Below we will compare the performance of the simulator in two separate cases: with a parallelized PTDR kernel, and with a parallelized alternative routing kernel.

Intermediate steps:

- Integrate `evkit` into the traffic simulator.
- Enable distributed computation of traffic simulation.
- Parallelize the alternative routing kernel in the traffic simulator.
- Parallelize the PTDR kernel in the traffic simulator.
- Execute traffic simulation with at least 100K vehicles and assess performance improvement against baseline without a parallelized alternative routing kernel.

Results:

[2]The PTDR kernel was also offloaded to an FPGA, this will be discussed in a later section.

- Parallelized PTDR kernel.
- Parallelized alternative routing kernel.
- Successful simulation with 250K vehicles (5× the baseline) with positive indicators of the ability to seamlessly support an even larger number of vehicles.
- Over 100× speedup of the required time to compute each simulation step

PTDR parallelization

To evaluate the performance of the PTDR Monte Carlo kernel, we have gathered a set of over 11,000 vehicle routes, from an execution trace of the traffic simulator. These routes were used for benchmarking the scaling of the PTDR Monte Carlo simulation kernel across multiple cores (each worker corresponds to a single CPU core), using `evkit`.

Figure 8 shows the results of the benchmark. All 11,000 routes were executed using `evkit`, which computed the PTDR kernel across multiple `evkit` workers, each using a single CPU core. The horizontal axis shows the amount of `evkit` workers used and the vertical axis displays the duration to calculate the Monte Carlo simulation for all input routes. Three configurations were benchmarked, with an increasing amount of Monte Carlo samples (more samples generate more accurate simulation, at the cost of longer computation). The results show that `evkit` can efficiently scale the computation of embarrassingly parallel kernels. The benchmark was executed on the Karolina cluster[3], on a single compute node with 128 cores and 256 GB RAM. Note that even though all workers were on the same node, `evkit` could also easily execute them on separate nodes.

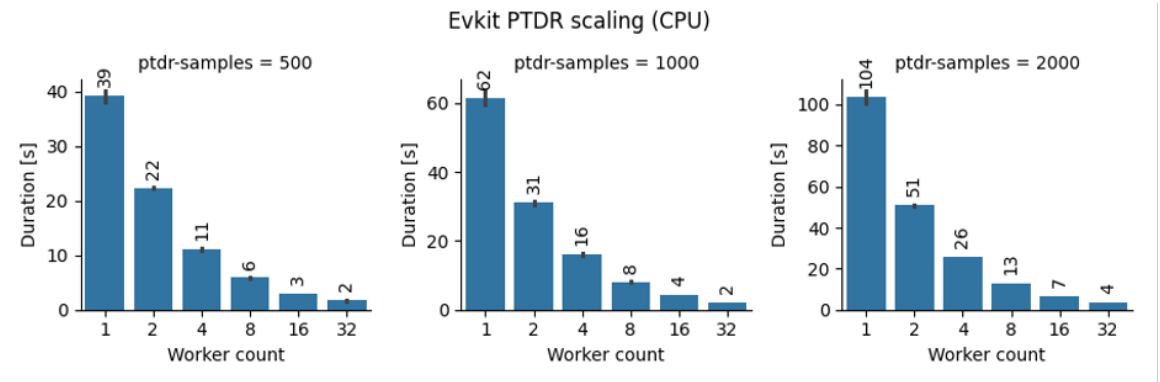


Figure 8 – Scaling of the PTDR kernel across multiple workers

Alternative routing parallelization

Several rounds of experiments were executed in order to assess the performance of the traffic simulator while parallelizing the alternative routing with `evkit` versus a baseline without `evkit`. There have been, among others, experiments with 2K, 5K, 10K, 50K, 60K 100K, and 250K vehicles. For easier readability, we present an overview of the main results with a 60K dataset as well as a 250K dataset in this deliverable.

Experiment: 60K vehicles

One of the designed experiments considered a dataset with 60K vehicles, where in the first 5 minutes the number of active cars increases to 20K and ramps up to 60K active vehicles within 30 min of simulation.

The following job configurations were launched on the CPU partition of the Barbora[4] supercomputer, located at **IT4I**'s premises:

[3] Karolina HPC cluster - documentation: https://docs.it4i.cz/karolina/introduction/
[4] Barbora HPC cluster - documentation: https://docs.it4i.cz/barbora/compute-nodes/

Attribute	Value	Attribute	Value
round-frequency-s	5	los-vehicles-tolerance-s	5
k-alternatives	3	travel-time-limit-perc	0.1
map-update-freq-s	15	saving-interval-s	100

Table 4 – Experiment: 60K vehicles - simulation setting

1. Single node, single worker, Dijkstra (no `evkit`), 3 alternatives.
2. Single node, single worker, Plateau, 3 alternatives.
3. Single node, 114 workers, Plateau, 3 alternatives.
4. 2 nodes, 114 workers, Plateau, 3 alternatives.
5. 4 nodes, 114 workers, Plateau, 3 alternatives.
6. 8 nodes, 114 workers, Plateau, 3 alternatives.
7. 16 nodes, 114 workers, Plateau, 3 alternatives.

Note that we have used 95% of CPUs (114) of each node, to leave 5% available for the system. The third entry of each configuration is the used routing method. The Dijkstra method uses the `networkx` library for routing in Python, and does not leverage `evkit`. The Plateau method uses a C++ routing implementation, and is parallelized using `evkit`.

Each of the jobs considered the simulation setting described in Table 4 and explained hereafter:

- *Round-frequency-s* is the interval (in seconds, of simulation time) for vehicle selection to be moved in one simulation step.
- *k-alternatives* is the number of alternative routes calculated for each vehicle.
- *map-update-freq-s* is the frequency of updating the map with current speeds (in seconds, in simulation time).
- *los-vehicles-tolerance-s* is the time tolerance (in seconds, of simulation time) to count which vehicles (i.e., their timestamps) are considered for the calculation of Level of Service (LoS) in a segment.
- *travel-time-limit-perc* represents the time (%) differential (in seconds, of simulation time) of the time between the current route and the fastest alternative route - meaning that in example vehicles only change to another path if the alternative path is 10% faster.
- *saving-interval-s* represents the time interval in which the state of simulation is periodically saved.

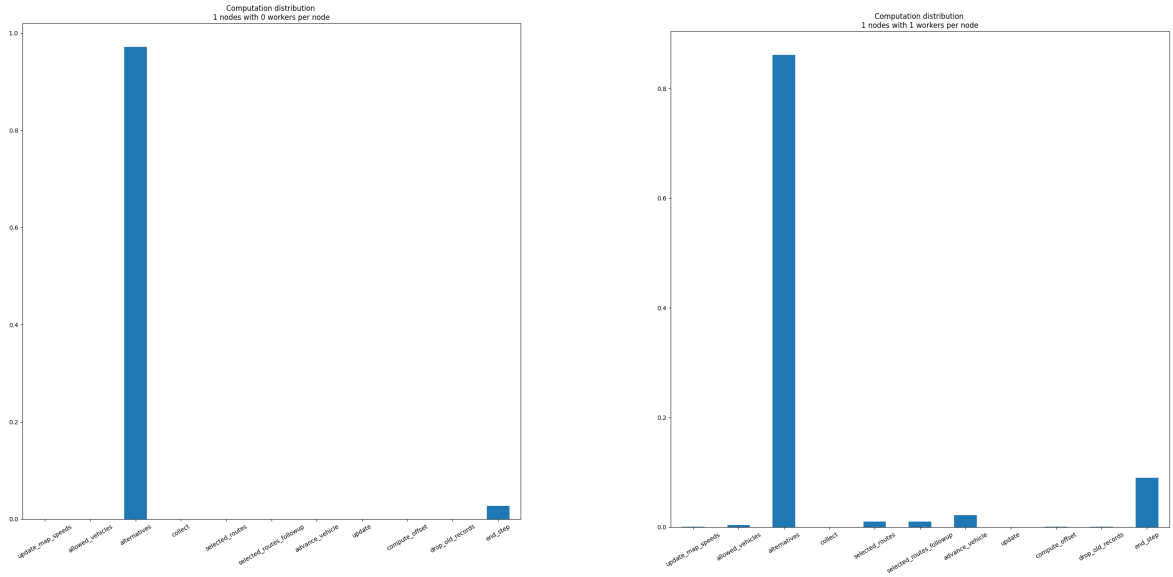
Table 5 presents the results of the execution of simulation under the aforementioned conditions with a time limit of 60 minutes or 512 simulation steps.

Table 5 shows that in all the cases where `evkit` is applied, there is an improvement either in simulation time or number of simulation steps executed. Comparing the first and second row of the table we observe that despite taking the same execution time, the number of simulation steps increases from 16 to 230, meaning that more vehicles have moved along their routes within the simulation. The required time per simulation step (last column) is one of the most relevant metrics. As we can observe, we were able to achieve a 105 times speedup when compared to the baseline version.

When comparing the distribution plot of the scenario without `evkit`, depicted in Figure 9, with a scenario with `evkit` such as the one with 2 nodes (as shown in Figure 10), we can observe a significant improvement in the most expensive computational task of the baseline version: computation of alternative routes, represented by the *alternatives* function. On the left (a), we observe that the *alternatives* function takes more than 95% of computational time, while *end_step* (to save the map) takes approximately 5%, meaning that all other functions

Number of Nodes	Workers per node	Computation Time (seconds)	Steps Completed	Computation Time Speed up	Time per Step (seconds)	Time per Step Speed up
1	1 (no `evkit`)	3600	16	Baseline	225	Baseline
1	1	3,600	230	1.00x	15.65	14×
1	114	1,736	512	2.07x	3.39	66×
2	114	1,272	512	2.83x	2.48	90×
4	114	1,170	512	3.08x	2.28	98×
8	114	1,118	512	3.22x	2.18	103×
16	114	1,097	512	3.28x	2.14	105×

Table 5 – Experiment: 60K vehicles - multiple scenarios of distribution vs baseline without `evkit` (time limit: 1 hour, step limit: 512 steps) - Barbora supercomputer (CPU partition)



(a) Simulation without `evkit` (baseline): 1 node, 1 worker (Python version of alternative routes: Networkx Dijkstra)

(b) Simulation with `evkit`: 1 node, 1 worker (Plateau (C++) version of alternative routes)

Figure 9 – Baseline version vs `evkit` version of traffic simulation

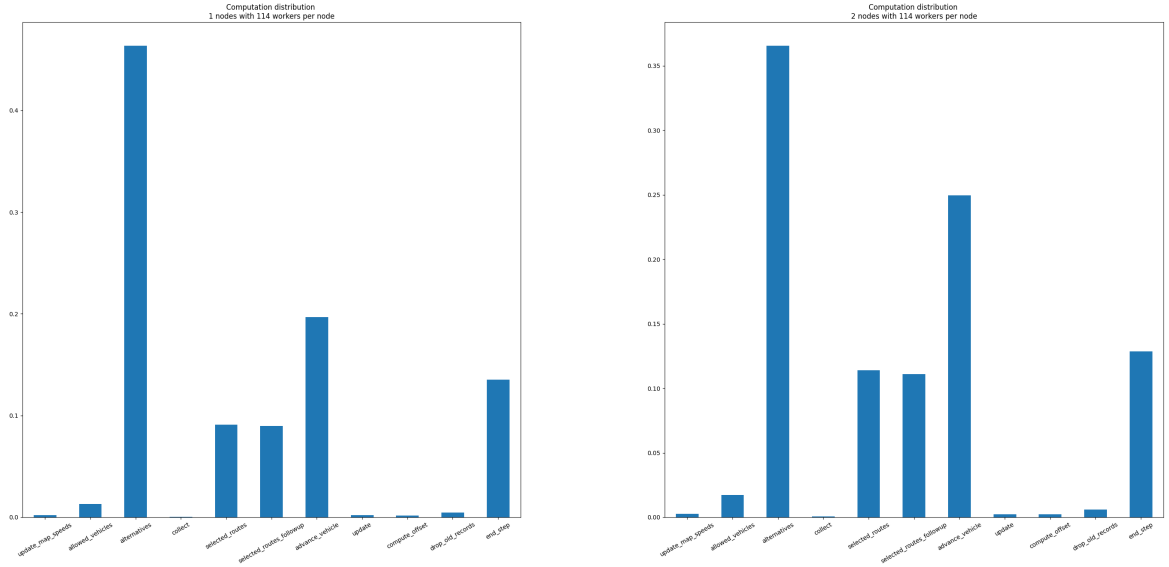
take a residual amount of time and the computation of the alternatives is the function requiring the most optimization.

On the other hand, when comparing two scenarios using `evkit` (1 node vs 2 nodes with 114 workers each), we can see in Figure 10 that while the computational effort decreases for the computation of the alternative, other functions of the simulator become more evident. In particular, the functions that stand out the most are the *advance_vehicle* and *end_step*. This means that `evkit` has not only performed its intended outcome but has also supported the identification of other computational tasks that will become a subject of further optimization work on the traffic simulator (beyond the EVEREST project scope).

Experiment: 250K vehicles

The other set of results considers a dataset with 250K vehicles (5 times more than the number of vehicles used for the baseline). This simulation considers a scenario in which vehicles are navigated through Prague, in different directions, from multiple origins to multiple destinations. The scenario simulates a time period between 5am until 9pm, where 250K vehicles are injected into the simulation over a period of 16 hours.

The job considered the simulation setting presented in Table 6; the difference to the previous one is highlighted in bold. The main change is in the saving interval, which is longer and not required to be so frequent under traditional simulation vs benchmark. Similarly, the second change is in the map update frequency which



(a) Simulation with `evkit`: 1 node, 114 workers

(b) Simulation with `evkit`: 2 nodes, 114 workers each

Figure 10 – `evkit` enhanced version: 1 node vs 2 nodes, 114 workers each

was changed from 15 seconds to 60 seconds - an appropriate value under normal situations and not for benchmarking when we want to stress the system.

Attribute	Value	Attribute	Value
round-frequency-s	5	los-vehicles-tolerance-s	5
k-alternatives	3	travel-time-limit-perc	0.1
map-update-freq-s	**60**	saving-interval-s	**1,000**

Table 6 – Experiment: 250K vehicles - simulation setting

The following graphic presents the results of the execution of the simulation under the aforementioned conditions.

Number of Nodes	Workers per node	Computation Time	Steps Completed	Time per Step (seconds)
4	114	11h:02m:04s	58,112	0.68

Table 7 – Results for complete 250K vehicles simulation with `evkit`

Table 7 shows that the simulation was successfully completed within 11 hours with approximately 58,000 steps. This translates into being able to compute each step in only 0.68 seconds in having requiring only 11h of computation time for a 16 hours of real-life simulation.

Figure 11 depicts different perspectives of the simulation with 250K vehicles. For instance, on the top left, it is possible to observe that traffic-intensive was higher in two periods (morning period and afternoon period) - as the simulation was execution from 5am until 9pm. There were more than 10,000 vehicles active in each computational step during the morning period, with up to 6,000 of them requiring the computation of alternative routes. At the same time, it is possible to see on the bottom left part that the computation of the alternatives is less expensive than the *advance_vehicle* function (which is aligned with the previously presented results). The upper right and lower right graphs showcase an indication of the duration of the simulation steps, which (as shown in Table 7) take less than 1 second to compute.

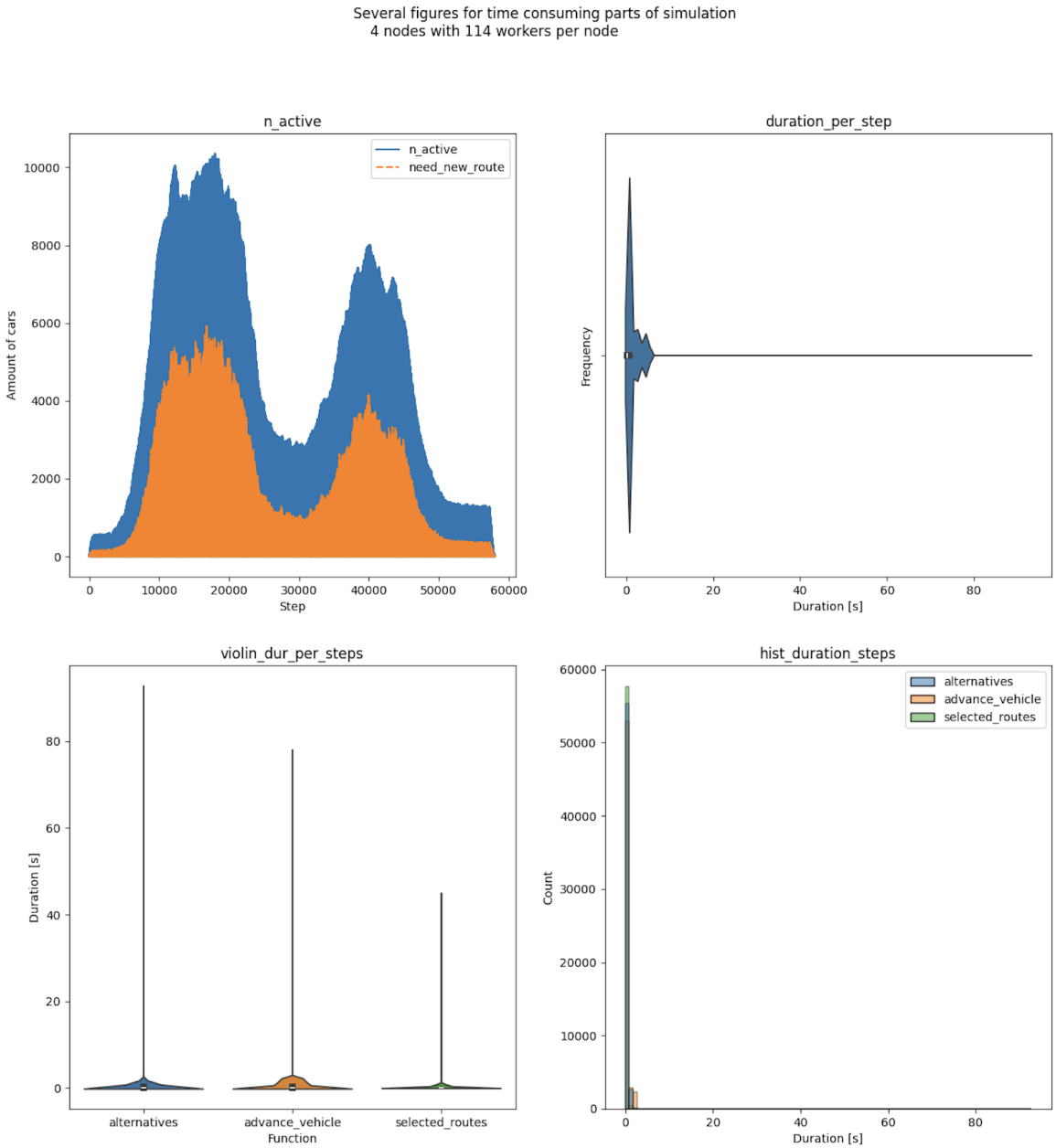


Figure 11 – From top left and right to bottom left and right: a) Number of active cars throughout the simulation; b) Frequency/duration of each simulation step; c) Comparison of 3 main functions of the simulator: alternatives computation, advance vehicle, and select route; d) Histogram with duration of the steps.

Figure 12 depicts a snapshot of the video that shows the visualization of the simulation with 250K vehicles. Visualization details have already been described in D6.3. Nevertheless, it is relevant to showcase a snapshot in this document as the figure highlights, for instance, the total number of vehicles, the number of active vehicles at the specific time of the simulation, the simulation scenario details (e.g., nodes, workers, alternative routes, map update frequency or travel time limit) as well as other metrics like the total driving time or the total amount of kilometers driven by the vehicles.

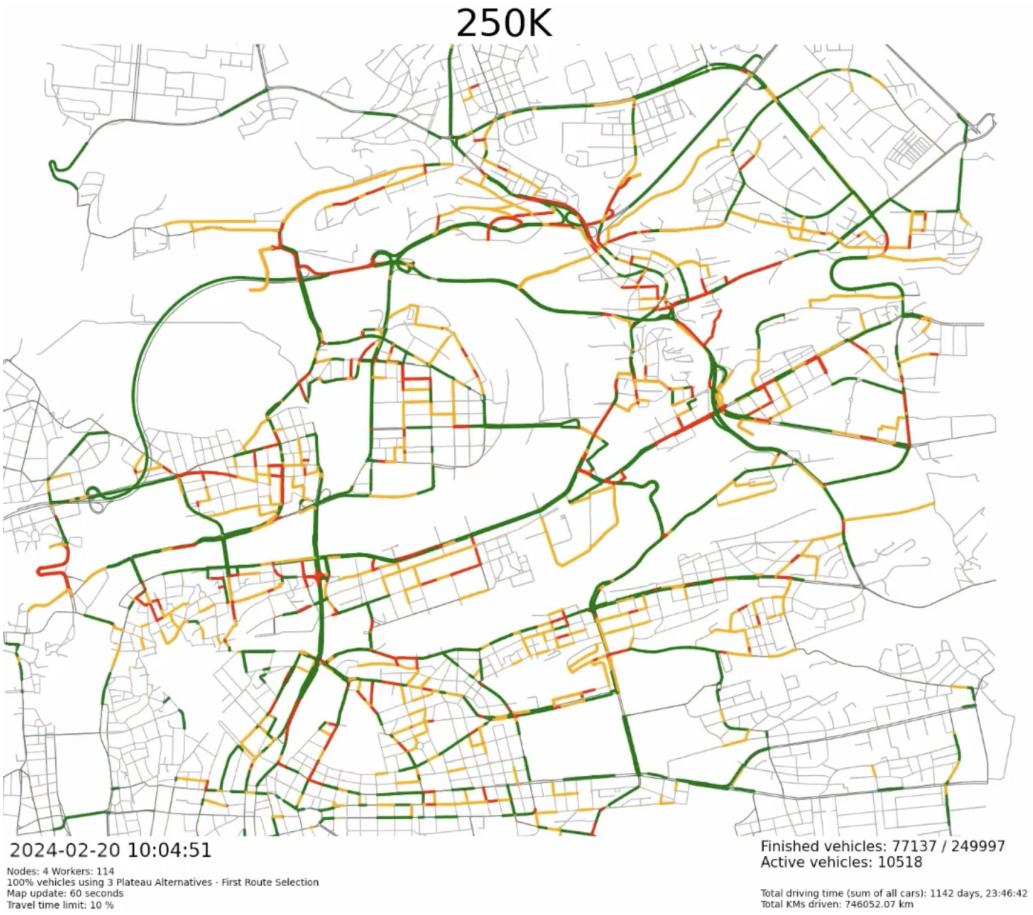


Figure 12 – Snapshot of the 250K vehicles simulation, around 10am, Prague, Czech Republic.

ITEM	DESCRIPTION
Method	Improve overall system precision by incorporating weather data from WRF model
Priority	Could Have
Baseline	Traffic simulator without weather forecast model connected
KPI	Availability of a weather forecast module connected to the simulator
Notes	This objective is not part of the EVEREST project, however it will be evaluated the effort and impact of having it.
How To Measure	Module Accessible/callable from Traffic Simulator
Involved Everest Components	Weather service component
Target Value	To define the approach on how to implement the processing of weather forecast information into the traffic simulator.
Reached Value	The implementation details have been analysed and defined.

Description:

Even though this particular item was not part of the EVEREST project, it has been proposed as a "Could Have" and the involved partners have assessed the necessary aspects to consider its implementation.

Intermediate steps:

- Define the type of events and associated information.
- Specify the impact of events in traffic simulations.
- Define how the traffic simulator considers weather forecast events.
- Assess the impact and effort of incorporating weather forecast events into the traffic simulator.

Results:

- Event types:
  Events such as wind speed (e.g., visibility distance or lane obstruction), precipitation (e.g., visibility distance, pavement friction), fog (e.g., visibility distance), pavement condition (e.g., road damage) or water level (e.g., lane submersion).
- Impact of the events:
  The aforementioned events have the following impacts on the traffic simulation: Traffic speed change, speed variance, travel time delay or road capacity changes.
- Usage within the traffic simulator:
  The traffic simulator may include an event handler that periodically receives weather forecast events and maps them into specific geographical regions. These regions are then converted into simulation segments and the simulation conditions (mentioned above) can be adjusted accordingly on the internal representation of the map. In this way the modifications to the current version of the simulator are really limited.
- Lessons learnt:
  Given the nature of this method (i.e., could have), the implementation has not been prioritized. Nevertheless, the foundation for the implementation work to proceed has been laid out and the added feature may be added beyond the timeline of the project.

2.3.2 O3.2 - Improve the overall performance of neural network traffic prediction model training

AI kernel of the focus is the neural network (NN). The specificity and the challenge of the kernel component is that for speed predictions for the entire city we need to employ tens of thousands of prediction models simultaneously, while parallel architectures of course have limits in a number of kernels to be exploited. The optimization goal is to execute this whole prediction batch in a sub-second cycle as this function is a part of the real-time online services, and of course with cost efficiency. Need for parallelization and cost-efficiency is well addressed with FPGA acceleration.

In order to fulfill the objective, two methods have been defined and evaluated:

1. Employment of accelerated AI computation kernels as enabler for a fast loop prediction calculation utilizing heterogeneous architectures with efficient data management
2. Evaluation of computational performance, precision and energy cost with respect to existing implementation

ITEM	DESCRIPTION
Method	Employment of accelerated AI computation kernels as enabler for a fast loop calculation. The utilization of heterogeneous architectures with efficient data management
Priority	Must Have
Baseline	CPU execution (Python)
KPI	Availability of prediction kernel on FPGA and its seamless integration toolkit
Notes	Focus was the cloud-based FPGA implementation option. Bus-attached FPGA implementation option has been performed only experimentally with manual support
How To Measure	Availability of a command line compilation cookbook
Involved Everest Components	Traffic prediction module, `dosa`, `olympus`
Target Value	prediction kernel is FPGA translated supporting parallelism by allowing multiple instances
Reached Value	prediction kernel is FPGA translated supporting paralellism by allowing multiple instances

Description

Traffic prediction component is about to execute distributed prediction for 10,000 (or more) different configurations of neural network kernel. This brings us the possibility to engage wide range of architectures starting from fast sequential execution to highly parallelized solutions. EVEREST technology further offers us to enlarge this space with the possibility to use FPGA acceleration, and further more on two platforms: cloud-based FPGA and bus-attached FPGA. EVEREST compilation flow brings the automation for FPGA acceleration options, which includes 1) conversion from Python function to the FPGA bitstream and 2) generating necessary interfacing code for a client integration. With the cloud-based option a socket connection is generated, while with the bus-attached option the host code interface is generated.

Intermediate steps

- Generic neural network inference python implementation servicing 10,000 prediction models
- Set up user interface for annotation of fixed-point, performance and power consumption
- `dosa` tool for converting python code to intermediate representation ONNX/MLIR
- `dosa` optimization and export for FPGA synthesis
- `olympus` integration of memory and HDL codes into deployable bitstream
- Deployment of the component on Alveo for and demonstrate an external use

Results

- Python TensorFlow and PyTorch implementations provided to serve both as a) benchmark and b) source to EVEREST optimization. See GitLab references to source code in Deliverable D6.3, Section 3.5.1.
- Interface for optimization directives (fixed-point and performance, etc.) is provided by EVEREST Basecamp tool through json-constraints file. See details in Deliverable D6.4, Section 3.5.
- `dosa` end-to-end compilation flow is demonstrated with Jupyter notebook environment (see GitLab reference in Deliverable D6.4, Section 3.5). It starts from using python-to-ONNX public tool and then applying EVEREST optimization methodology on ONNX towards a) bistream directly, b) MLIR intermediate output.

- For cloud-based FPGA service, `dosa` generates deploy script applicable to **IBM** cloud infrastructure. Clients can thus access service through zetoMQ enable scoket client. See details in Deliverable D6.4, Section 3.5.
- `olympus` is used for the bus-attached FPGA architecture target, which starts from MLIR representation output of `dosa`. `olympus` generates library components to be used for an integration into a client application.

Traffic prediction service integration

The following table details the different configurations that have been produced with the purpose of the evaluation of the EVEREST toolchain and the performance optimization. Table 8 indicates which EVEREST tools used to produce the current results, but the combination of dosa and olympus could also be applied to future cloudFPGA deployments.

Acceleration case	Tool	Availability
CPU 1 kernel 4 cores baseline	Python	yes
cloudFPGA single kernel	`dosa`	yes
cloudFPGA multiple kernels	`dosa`	yes
bus-attached FPGA single kernel	`dosa` & `olympus`	yes (experimental)
bus attached FPGA multiple kernels	`dosa` & `olympus`	yes (experimental)

Table 8 – Architecture options of the Traffic prediction acceleration

With the cloud-FPGA implementation option, we use `dosa` to synthesize into an FPGA architecture. Table 9 presents post place-and-route results for a single Traffic prediction neural network kernel. Considering the capacity of the standard cloud-FPGA kernel, the theoretical limit for parallel instantiation is 29.

Kernel	Cycles	LUTs	FFs	BRAMs	DSPs	Frequency	Theoretical Parallel Limit
network-based single kernel	70	11,579	10,777	0	0	156 MHz	29

Table 9 – Synthesis results of the NN-kernel

At the end of the day, `dosa` and `olympus` tools are wrapped in the EVEREST Basecamp facade tool, which exposes the necessary user-level parameterization, such as fixed-point definition, and architecture and performance targets.

ITEM	DESCRIPTION
Method	Evaluation of computational performance, precision and energy cost with respect to existing implementation
Priority	Must Have
Baseline	CPU-based sequential execution of 10K prediction
KPI	Speedup w.r.t. purely CPU based, % energy saving using FPGA acceleration
Notes	Using measurements we evaluated only the cloud-FPGA implementation option
How To Measure	Execution time and power measurement for 10,000 predictions FPGA vs CPU
Involved Everest Components	Traffic prediction module, `dosa`, `olympus`
Target Value	5× speedup w.r.t. CPU based, 50 % energy saving using FPGA acceleration
Reached Value	24× speedup and 79% energy saving with the use of 8 kernel instances. The CPU profiling measures have been taken on an AMD EPYC 7643 CPU.

Description

EVEREST tool options provide user with possibility to configure and evaluate various architecture targets. There are two options in this case, either useing the cloudFPGA devices or bus-attached FPGA acceleration. Both targets allow a play with parallelizing options, where the limiting factor is the size of FPGA chips being used.

Intermediate Steps

- Measure performance on a selected performing CPU-based system for 10,000 predictions (time duration and power consumption). The CPU profiling measures have been taken on an AMD EPYC 7643 CPU.
- Measure performance for 10,000 predictions using FPGA acceleration (time duration and power consumption)

Results

- Baseline non-FPGA computation on Intel-based 4-core architecture takes approximately 6 seconds for 10K models prediction (for details see Deliverable D6.3 Section 3.5.2). Note that the target of 10K model prediction is suitable for medium-size city like Prague, while larger cities might even require multiples of 10k models.
- The EVEREST performance target was to reach a 5× speedup. When deploying a single kernel on **IBM**s cloudFPGA platform, we only reach a 3× speedup. However, the FPGA used for **IBM**s cloudFPGA card fits 8 NN-kernels, which increases the speed up factor up to 24× and exceeds our target. Note that the measurements also includes the network latency.
- Further speedup and energy-efficiency gains could be achieved by using the `olympus` tool to handle the management of the NN coefficients across a shared pool of kernels.
- Finally, it would be beneficial to use the larger logic capacity of the bus-attached Alveo FPGA cards, where more than 40 Traffic Prediction kernels could fit into a single FPGA. Post-synthesis estimations provided a latency of less than 100 cycles at 220MHz for each prediction module.
- In conclusion, the FPGA implementation proves to be greatly more performing and energy efficient than its CPU counterpart.

Traffic Prediction performance

Table 10 shows the execution of 10,000 predictions benchmark for the defined architecture configurations.

Acceleration case	Kernels	Performance 10k predictions	Energy (J) 10k predictions	Speedup
4-core CPU single kernel (baseline)	-	5.5s	617	baseline
cloudFPGA single kernel	1	1.9s	16.3	3×
cloudFPGA multiple kernels	8	0.23s	2.7	24×

Table 10 – Performance Evaluation of the Architecture options of the Traffic prediction acceleration. The CPU profiling measures have been taken on an AMD EPYC 7643 CPU.

2.3.3 O3.3 - Improve data management and computational services and reduce programming effort

The goal of this objective was to make it possible to execute the EVEREST use-cases in a distributed manner, while allowing the offload of selected kernels to an FPGA accelerator device. This should ideally be possible in a way that is transparent to the application developer and that does not require them to spend too much effort to parallelize their application.

In order to fulfill the objective, three methods have been defined and evaluated:

1. Exploiting of an effective EVEREST computing platform to support distributed computation and optimized data flows
2. Availability of EVEREST module that manages data exchange from and to accelerated kernels
3. Heterogeneous resources to be exploited in a simple way, e.g. with minimum number of lines of code

ITEM	DESCRIPTION
Method	Exploiting of an effective EVEREST computing platform to support distributed computation and optimized data flows
Priority	Must Have
Baseline	N/A
KPI	Availability of end-to-end WFs for the Traffic use case
Notes	Before the project, the modules were available as separate instances
How To Measure	Verify whether or not distributed computation is enabled on the traffic simulator
Involved Everest Components	Traffic simulator, `evkit`
Target Value	Traffic simulator running with `evkit` on a distributed cluster
Reached Value	Traffic simulator running with `evkit` on a distributed cluster, with PTDR and alternative routing kernels integrated, and with LEXIS integration.

Description

The `evkit` distributed runtime was created, and integrated with the traffic simulator. This allows the simulator to execute its two most resource-intensive kernels (PTDR MonteCarlo simulation and alternative routing)

on remote nodes of a cluster. The whole end-to-end traffic simulation workflow that uses `evkit` has also been integrated with the LEXIS system, which allows users of the simulator to execute the workflow on a target cluster in a very simple way.

Intermediate Steps

1. Design and implement interfaces for swapping kernel implementations in the traffic simulator
2. Integrate `evkit` into the traffic simulator
3. Distribute PTDR computation in the traffic simulator
4. Distribute alternative routing computation in the traffic simulator
5. Enable execution of the traffic simulator through LEXIS

Results

- The traffic simulator has gained the ability to use a different implementation of the PTDR (the Monte Carlo simulation) and alternative routing kernels.
- The `evkit` distributed runtime has been created.
- `evkit` has been integrated with the traffic simulator, where it allows parallelizing of both the PTDR and alternative routing kernels.
- Parallelization of the kernels can happen both on multiple cores and also on remote nodes of a cluster.
- It is possible to execute the whole traffic simulator workflow end-to-end through LEXIS.

ITEM	DESCRIPTION
Method	Availability of EVEREST module that manages data exchange from and to accelerated kernels
Priority	Must Have
Baseline	N/A
KPI	Availability of almost transparent accelerator integration in the application code
Notes	None
How To Measure	Test if an accelerated kernel can be executed on an FPGA using `evkit` integrated with `olympus`
Involved Everest Components	`evkit`, `olympus`
Target Value	`evkit` integrated with `olympus`
Reached Value	`evkit` integrated with `olympus` for PTDR and Map Matching.

Description:

EVEREST data exchange is provided on two levels. First, `evkit` transfers inputs from the traffic simulator through the network to an `evkit` worker, where the kernel is computed, and then it also transfers the outputs back to the simulator. Second, `evkit` integrates the `olympus` tool, using which it transfers data from the host node to a bus-attached FPGA devices, and then again reads the computed kernel results back from the accelerator. The traffic simulator code does not know anything about the specifics of the FPGA acceleration, which is thus fully transparent. It also uses a very simple Python interface for exchanging data with the kernels using `evkit`.

Intermediate Steps:

1. Implement network data exchange between `evkit` and the traffic simulator.
2. Integrate `evkit` with `olympus` to enable FPGA kernel execution with data exchange.
3. Implement data (de)serialization needed for executing PTDR kernel on an FPGA.
4. Implement data (de)serialization needed for executing Map matching kernel on an FPGA.

Results:

- `evkit` implements support for offloading the PTDR kernel to a bus-attached FPGA device.
- `evkit` has been integrated with the `olympus` tool, and can transfer data both to and from a bus-attached FPGA device.
- The traffic simulator has been integrated with `evkit`, and can use it to easily invoke kernels and exchange data with them over the network (provide them inputs and read their outputs). The traffic simulator code passes data to `evkit` in the form of Python objects, and doesn't need to know about the specifics of FPGA data transfer.

ITEM	DESCRIPTION
Method	Heterogeneous resources to be exploited in a simple way, e.g. with minimum number of lines of code
Priority	Must Have
Baseline	No FPGA accelerators in the code
KPI	Number of kernels mapped to FPGA using the Everest Framework
Notes	Evaluated kernels are Traffic Prediction, MapMatching, and PTDR. We used survey evaluation technique detailed in the Description section
How To Measure	number of components mapped and evaluated
Involved Everest Components	`bambu`, `olympus`, `evkit`
Target Value	Retargeting of Map-matching, Traffic Prediction, PTDR evaluated as low-effort
Reached Value	Retargeting of Map-matching, Traffic Prediction, PTDR evaluated as minimum/small effort based on the survey

Description

There are three traffic modeling use-cases (Map-matching, Traffic Prediction, PTDR) that are subject to FPGA offloading and thus should be addressed by EVEREST compilation flow. These use-cases contain computational kernels written in both C++ and Python, both of these languages are supported by EVEREST.

Ideally, it should be possible to offload these kernels to FPGAs with a low effort spent by the application developers. This has been achieved up to various levels, depending on the specific use-case.

It is obvious objective that the ideal tooling would do perfect automation by not requiring the developer to do any modification on the original code. This would really be ambitious. We identified the following categories, which must be addressed and which need developer engagement to some extent:

- code adaptation for compatibility to automated FPGA transformation
- code adaptation for kernel identification
- support for fixed-point refinement

With the survey of limited audience of developers we finally evaluated the above categories for a developer's effort as:

- none (1)
- minimal (2)
- small (3)
- medium (4)
- difficult (5)

Our goal is on average to have the score - less than 3.

Results

- The Map matching kernel is written in C++ using STL (Standard Template Library). EVEREST well supports compilation flow towards FPGA offloading, which is the job of `bambu` tool. All must-have source code changes are the translation of STL functions (e.g. vector) into their ETL (Embedded Template Library) counterparts. ETL syntax is very similar in nature to STL, it only adds the boundary limits to structures. E.g. infinitely sized STL-vector needs to be declared as ETL-vector with a hard limit to e.g. 20 items. This type of source code adaptation has been evaluated as minimal and acceptable to users.
  Convenient support of EVEREST is also in the definition of kernels, which are required to be offloaded to FPGA. The tool requirement is to express the code/kernel as a function, which defines all inputs and outputs in the function argument list. Finally #pragma annotation needs to appear in front of the function definition, which explicitly defines arguments as input or output and requests a specific interface protocol. This type of source code adaptation may require restructuring of the code (creating functions), however, with good designs this is typically already in place initially, or otherwise it leads to better modularity. By developers this adaptation is evaluated as minimal and driving good design practice.
  Optional modification of the source code can occur when redefining built-in C++ floating-types types into fixed-point definitions. This is supported through the `bambu` libbambu C++ library. Application developers can resort to this translation when they want to increase performance at the expense of some precision loss. This conversion is syntactically acceptable as it at its minimum requires changes on variable declaration only. However, going deeper into complex mathematical formulas, it is sometimes necessary to split a formula atomically to create intermediate variables so that they can be explicitly defined in fixed-point. There is no automation for the inference of optimal fixed-point definitions, so it is the responsibility of application developers. To conclude the level of this modification can be tedious, yet acceptable, as it only needs to be applied locally where required.
- The Traffic prediction kernel is written in Python using alternatively Tensorflow or PyTorch library. EVEREST tool represented with `dosa` optimizer component starts with ONNX representation, which can be obtained automatically with the export functions of Pytorch or Tensorflow (i.e. "tf2onnx). Thus there is no need for source code modification due to python feature support to represent dataflow function in ONNX format. ONNX is then taken over by `dosa` to perform all necessary optimization operation leading towards final FPGA bitstream output.

  The kernel identification to be directed to acceleration is also addressed with the ONNX export. In case of Tensorflow, there are some limitation for the function, which is subject to tf2conn function support, i.e. the function needs to be expressed as Python function and annotated with the "@tf.function" decorator. Wrapping kernel into a function is obviously a typical design practice and adding annotation to it is a minimal modification.

  `dosa` also supports an advanced feature of automated assistance on a fixed-point refinement of the kernel dataflow. As a developer, you only need to present the required precision and testing data to `dosa` tool, and it will automatically infer fixed-point characterization of the kernel without further developer interaction (this feature is only supported for Pytorch models).

- The PTDR Monte-Carlo simulation kernel was originally implemented in Rust, however it proved to be too complex to offload it to an FPGA using the `olympus` and `bambu` tools. Therefore, it has been re-implemented in C++ during the course of the EVEREST project. This C++ version originally used STL, but it was modified to use ETL instead, in a similar fashion as the Map matching kernel, to make it amenable for FPGA acceleration.

  An FPGA bitstream can be generated from the C++ kernel using the `bambu` and `olympus` tools through a series of scripts. The scripts should be tuned by performance engineers to obtain the best results.

As previously defined, we present the evaluation of developers' effort per category for all the three use-cases, see Table 11:

Usecase	Language	Kernel identification effort	Kernel transformation compatibility effort	Fixed-point refinement effort
Map matching	C++	small	minimal	small/medium
Traffic prediction	Python	minimal	none	none
PTDR	C++	minimal	medium	n/a

Table 11 – Evaluation of developers effort per use case

According to our methodology, the average score of the programming effort to adapt to EVEREST technology is 2.2, which concludes the minimal to small effort.

2.3.4 O3.4 - Improve the overall performance of map matching kernel

The goal of the objective is to accelerate the Map-matching component (MMA), which is the type of batch calculation component that processes millions of vectors each day converting them to road speeds, and which takes several hours to execute on CPU. Our idea is to bring the duration of calculation into near-real-time terms. MMA comprises several sub-components, which we call sub-kernels, i.e. GPS-projection, Trellis, Viterbi, Interpolator. EVEREST allows us a play on accelerating the overall component in a truly heterogeneous way, where some sub-kernels can be offloaded to FPGA, while some might stay in CPU space. EVEREST compilation flexibility allows us to investigate and evaluate multiple options to find the optimal configuration with respect to proper heterogeneity and parallelism.

To fulfill the objective, two methods have been defined:

- Acceleration of Map-match kernel through optimization of its sub-components
- Evaluation of computational performance and energy cost

ITEM	DESCRIPTION
Method	Acceleration of Map-match kernel comprised of GPS-projection, Dijkstra and Viterbi components forming Hidden Markov model.
Priority	Must Have
Baseline	Map Matching on CPU
KPI	Availability of a heterogeneously accelerated kernel and its well-defined integration flow
Notes	Map-match component has finally been split into 4 kernels: GPS-projection, Trellis, Viterbi, Interpolator
How To Measure	Availability of intelligible cookbook on a compilation toolflow chain
Involved Everest Components	Map-match module, `bambu`, `olympus`, dfg-mlir
Target Value	Relevant kernel(s) of MMA component are FPGA accelerated
Reached Value	All kernels of MMA are FPGA-translated

Description

Map-matching algorithm component (MMA) is about processing short GPS vehicle trajectories (vectors) obtained from vehicle sensors (or Traffic simulator) and converting them into speeds. For a medium-sized city (like Prague) this is about 1 million vectors, which takes several hours on standard CPU architectures. EVEREST technology allows us to evaluate various architecture setups through annotation, defining which of the sub-kernels we want to offload to FPGA space and with what parallelizing factor. Another inherent parallelization option emerges from treating the sub-kernels as dataflow black boxes, which can be executed in parallel wherever data flow dependency allows it. For measurement and performance evaluation we have finally used benchmarking with 1,000 vectors.

Intermediate Steps

- Prepare FCD data input from Sygic or Traffic simulator
- Kernel(s) written in C++ ETL and functionally equivalent in CPU execution
- Kernel(s) synthesizable by `bambu`, and functionally equivalent in HDL simulation
- Ohua DSL pseudo-code for all kernels data interactions
- Ohua code lowering to `olympus`
- Synthesis of the whole MMA code into object files by Ohua/`olympus`/`bambu`
- Execution of the MMA with testbench data input and output

Results

- Benchmark input of 1,000 input vectors have been prepared and made public through ZENODO platform (for details see Deliverable D1.4).
- C++ kernels were rewritten into C++ ETL, which has been verified to be functionally equivalent to the original code through CPU execution. The source code versions are referred in Deliverable D6.3 Section 3.5.1. C++ ETL syntax is a variant of C++ STL library, which requires minor manual modifications (mainly defining array boundaries of structures) while providing benefits of synthesizability to FPGA through `bambu`.
- All kernels are synthesizable by `bambu`, and were proven, functionally equivalent in HDL simulation. Synthesis figures are listed below in Table 13

- Fixed-point optimization has been applied on GPS-projection kernel (using libbambu fixed-point library), which was found as the most critical in performance as well as in FPGA area consumption as also indicated in Table 13.
- EVEREST Basecamp end-to-end compilation (using the tools `dlg-mlir` and `olympus`) has been done for several configuration variants. The architecture target chosen for this workflow was the FPGA bus-attached compilation option. The architecture variants are indicated in Table 12.
- the MMA heterogeneous version was tested for functional equivalence being deployed in **PDM** premises using the same Alveo FPGA cards as in the **IBM** cluster.

MMA component under acceleration consists of 4 sub-kernels, whereas each can be offloaded to FPGA independently. Everest flow makes sure interfacing is generated automatically so that it makes seamless integration of CPU parts and those FPGA offloaded. The following table details the different configurations that have been produced with the purpose of the evaluation of the EVEREST toolchain. The table indicates whether the kernel is executed in the host (CPU) or offloaded (FPGA), whereas the mark '||' denotes the parallelizing option.

Acceleration case	Projection	Trellis	Viterbi	Interpolate
Baseline	CPU	CPU	CPU	CPU
Parallel projection	CPU ||	CPU	CPU	CPU
Offloaded projection	FPGA ||	CPU	CPU	CPU
Offloaded Viterbi	CPU	CPU	FPGA	CPU
Offloaded all	FPGA	FPGA	FPGA	FPGA

Table 12 – Compilation configuration options of the MMA kernel ('||' denotes the parallelizing option)

We use the `bambu` tool to translate C++ descriptions to Verilog descriptions, which are then synthesized into an FPGA bitstream. Table 13 presents post place-and-route metrics related to the MMA kernels in isolation and fused together, which can be used to estimate an upper bound to the number of kernels that can be instantiated in parallel on the FPGA (without considering the area occupied by memory controllers and integration logic). The top-level component area does not precisely correspond to the sum of the areas of each sub-component, since, in the former case, `bambu` can apply optimizations across sub-components. The execution time relates to the processing of one input vector. The target board used is a Xilinx Alveo U55C. Each external memory operation is assumed to take 1 clock cycle.

Kernel	Cycles	LUTs	Registers	BRAMs	DSPs	Frequency	Theoretical Parallel Limit
Gps-Projection flp	31737467	79498	71185	68	336	280 MHz	16
Gps-Projection fxp	27412500	19280	18831	8	96	251 MHz	68
Viterbi	765	6285	5470	8	0	339 MHz	210
Trellis	2153816	19739	12341	278	33	196 MHz	7
Interpolate	71802	20301	16791	8	20	278 MHz	65
Top-level	3600064	77836	44432	334	217	147 MHz	6

Table 13 – Synthesis results of sub-kernels including the theoretical limit for parallelization

ITEM	DESCRIPTION
Method	Evaluation of computational performance and energy cost
Priority	Must Have
Baseline	CPU based sequential execution of 20M FCD points
KPI	Speedup w.r.t. purely CPU based, % energy saving using FPGA acceleration
Notes	Performance evaluation compared to 16-core CPU processor baseline
How To Measure	Software timers, HDL clock frequency and number of cycles, FPGA power monitor
Involved Everest Components	Map-match module, `bambu`, `olympus`, dfg-mlir
Target Value	5× performance improvement and/or -50% power consumption
Reached Value	performance > 6×, energy -80%. The measurements have been taken on an AMD EPYC 7282 CPU, 64 GB RAM.

Description

EVEREST compilation flow provides user with the possibility to configure and generate various architecture targets. The different targets under exploration have been presented in the previous method, and in the following we will provide their performance figures. For components offloaded to FPGA we played with the parallelization factor, potentially up to the limits of the Alveo u280 Xilinx FPGA card. For the more fair evaluation of our optimization we measured the baseline performance on the state-of-the-art CPU architecture: 16-core AMD EPYC 7282 CPU, 64 GB RAM, CentOS 7.9.

Intermediate Steps

- Measure performance on a selected performant CPU-based system for processing of 1000 vectors (time duration and power consumption)
- Measure performance for the same testbench using FPGA acceleration (time duration and power consumption)

Results

Referring to Table 14, the following conclusions can be drawn:

- Baseline non-FPGA computation on Intel-based 4-core architecture (for details see Deliverable D6.3 Section 3.4.2) takes approximately 12 hours for 1 million vectors, which is a typical usecase for middle-size city. Scaling it to 1,000 vectors the duration is approximately 44 seconds.
- Over the course of the project we transformed the initial baseline into memory optimized alternative, where memory is split into 5×5 grid to allow for faster memory operations. This transformation is also the enabler for better employment of parallelisation techniques with EVEREST. The transformation improves the performance from 44s to 19s (the factor 2.3×).
- For the more fair comparison of our optimization we finally used state-of-the-art CPU for our baseline: 16-core AMD EPYC 7282 CPU, 64 GB RAM. This resulted in yet another increase of the performance of our baseline from 19s to 9.6s (the factor 2×).
- The initial performance target of speedup 5× has been achieved with several architecture configurations all revolving around offloading GPS-projection to FPGA with a parallelization option. Already 4 parallel instances provided us with speedup of 5×, while higher degree of parallelization increased speedup negligibly. Yet we finally achieved the speedup of 6.4×.
- Offloading other kernels like Viterbi showed no gains so they have been retained in the CPU space.

- Energy-wise the FPGA implementation proves to be greatly more efficient than its CPU counterpart. With offloading just the GPS-projection kernel we achieved the power reduction -80%.
- Note that the performance speedup has been referred to the optimized baseline (i.e. factor >6×). If we were to compare against the initial baseline, the speedup factor would be 30×. Nevertheless, with all the techniques combined we reduced the map-matching computation for 1 million vectors from 12 hours to 25 minutes, which has been **SYG** product goal to be able to process all data overnight.

Table 14 – Performance evaluation of MMA kernel acceleration variants. The CPU measurements have been taken on an AMD EPYC 7282 CPU.

Acceleration case	Parallelization	Performance 1k vectors	Energy (average power)	speedup
Baseline 4core	-	44s	-	baseline
Baseline 4core -memopt	-	19s	-	baseline
Baseline 16core -memopt	-	9.6s	1150 J	baseline
Dfg-mlir optim	-	6.1s	734 J	1.6×
Parallel Gps-projection	4x CPU	3.7s	444 J	2.6×
Offloaded Gps-projection	1x FPGA	2.8s	396 J	3.4×
Offloaded Gps-projection	4x FPGA	1.8s	262 J	5.3×
Offloaded Gps-projection	8x FPGA	1.75s	254 J	5.5×
Offloaded Gps-projection	2xCPU + 8x FPGA	1.5s	219 J	6.4×
Offloaded Viterbi	1x FPGA	9.9s	1408 J	1.0×

2.3.5 O3.5 - Improve the overall performance of the probabilistic time-dependent routing kernel

The goal of this objective was to accelerate the PTDR Monte Carlo simulation kernel, with the goal of alleviating the computational bottleneck that it has been previously causing in the traffic simulator.

This objective has been divided into two steps, port the kernel to an FPGA device, and then optimize it and measure if the accelerated kernel can provide speed-up.

ITEM	DESCRIPTION
Method	Acceleration of PTDR kernel Including Montecarlo Sampling for time to destination estimation
Priority	Must Have
Baseline	PTDR Montecarlo on CPU (C++)
KPI	Availability of FPGA accelerated kernel
Notes	N/A
How To Measure	Evaluate that the PTDR kernel can run on an FPGA within an end-to-end traffic simulator workflow
Involved Everest Components	`bambu`, `olympus`, mARGOt, Virtualization extension
Target Value	PTDR FPGA kernel is available
Reached Value	PTDR FPGA kernel is available and integrated with the EVEREST run-time

Description

The Monte Carlo kernel was written using a simplified variant of C++, leveraging the ETL library. It is possible to generate an FPGA bitstream from it using the `olympus` and `bambu` tools. The kernel can be found in the `ptdr-monte-carlo-kernel`[5] repository. The kernel has also been integrated with the EVEREST runtime environment, including `evkit`, mARGOt, the virtualization extension, and it can be used in an end-to-end execution of the traffic simulator.

Intermediate Steps

1. Prepare simplified PTDR kernel written in C++, leveraging ETL.
2. Synthesize kernel with `bambu`.
3. Integrate onto Alveo u55c on **IBM** cluster using `olympus` and generate host driver API.
4. Integrate the PTDR kernel into `evkit`.

Results

- The Monte Carlo kernel has been ported to an FPGA bitstream with the `bambu` and `olympus` tools.
- The kernel has also been integrated with `evkit`, and can be used in an end-to-end execution of the traffic simulator.

ITEM	DESCRIPTION
Method	Evaluation of computational performance and energy cost
Priority	Must Have
Baseline	CPU-based sequential PTDR execution of several thousand requests
KPI	Speedup w.r.t. CPU based, % energy saving using FPGA acceleration
Notes	N/A
How To Measure	Software timers, FPGA power monitors.
Involved Everest Components	`bambu`, `olympus`
Target Value	3× performance improvement and -30% energy consumption
Reached Value	Performance on FPGA is orders of magnitude slower than on CPU, FPGA energy consumption has not been measured, therefore there is no comparison with the CPU energy consumption. The CPU profiling measures have been taken on an AMD EPYC 7643 CPU.

Description

To evaluate the performance of the PTDR Monte Carlo kernel on FPGA devices, we have used the same route dataset that was used in the "Evaluation of the performance for traffic simulation test case" task described in Table 2.3.1, however this time we have used only 500 input routes (a ∼20× smaller input), otherwise the benchmark would take an unnecessarily long time.

In this case, the experiment was performed on the **IBM** cluster, which contains two bus-attached FPGA devices on each node. We have compared the performance of the PTDR kernel running on an FPGA with the kernel running on a single CPU core. The results of this experiment can be seen in Figure 13. The horizontal axis shows the amount of used workers (in this case, a single worker represents a single FPGA device or a single CPU core).

[5] https://code.it4i.cz/everest/ptdr-monte-carlo-kernel/-/blob/main/include/ptdr/profiler/profiler.hpp

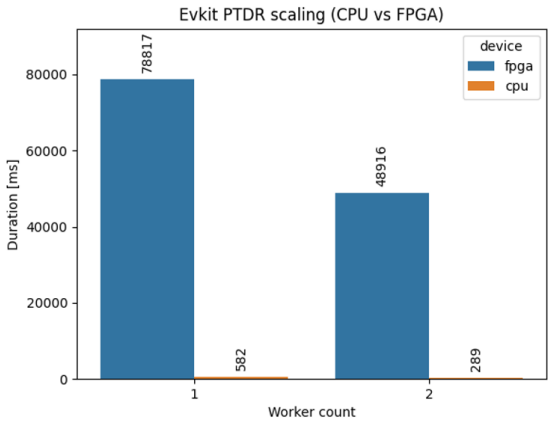


Figure 13 – PTDR kernel: `evkit` workers on FPGA vs CPU

We have also evaluated the energy (Watt) consumption of the traffic simulator and the PTDR kernel when the kernel runs on the CPU. We have performed this measurement on the **IBM** cluster, which has two sockets, each with an AMD EPYC 7643 CPU. The energy consumption was measured through the RAPL interface provided by the `amd_energy` Linux kernel module.

The following measurement methodology was used to gather the results. First, we have measured the steady-state energy consumption of the system, without any computationally demanding tasks being executed in the background. This value was determined to be approximately 157 Watts at the time of the measurement (note that this is a combined value that sums the consumption of both sockets on a single node of the cluster). Then we have executed the traffic simulator with the PTDR kernel running on a single CPU core for one minute, and measured the difference in energy consumption.

Intermediate Steps

1. Execute sequential CPU version of the PTDR kernel on the **IBM** cluster
2. Measure the time of the PTDR kernel running on the CPU
3. Optimize `bambu` synthesis of the kernel
4. Apply system-level optimizations via `olympus`
5. Measure the time of the PTDR kernel running on the FPGA

Results

- `evkit` is able to scale computation across multiple FPGAs, as can be seen in Figure 13.
- The absolute performance of the kernel running on an FPGA device is orders of magnitude slower than the CPU version. The kernel can be replicated multiple times on the FPGA to increase the degree of parallelism, but the number of available memory channels towards HBM is always going to be a bottleneck. Thus, despite a huge parallelization opportunity, the HBM channels limit the number of accelerators that can be instantiated on the FPGA.
- We have measured the energy consumption of the traffic simulator with the PTDR kernel on a single CPU core to be approximately 6 Watts. As we do not have a baseline for the energy consumption of the bus-attached FPGA device, we cannot compare this result with the energy consumption of an FPGA accelerator.

3 User Story: Traffic Prediction using the EVEREST SDK

One of the key goals of the EVEREST SDK is to enable application developers to deploy their applications on heterogeneous infrastructure in general and in particular on FPGA-accelerated hardware. Although the benefits of such a deployment (e.g., improved latency performance or throughput, improved energy efficiency) are widely accepted, very few developers have the ability to optimize applications for deployment on FPGA-accelerated infrastructure and they also lack knowledge about the additional skills required to assist them. The user story illustrates how the EVEREST SDK addresses this issue by first defining the tasks and skills of the personas involved in the development and deployment process, then describing their task/problem and how they address it using the EVEREST SDK, and finally showing the benefits for each persona.

It describes a desired feature or functionality and answers three key questions:

- Who: Who is the user trying to achieve something? (e.g., a network administrator, a mobile app user)
- What: What specific task or problem does the user want to solve through this optimization? (e.g., reduce app loading times, and improve traffic prediction accuracy)
- Why: What benefit will the user gain from this optimization? (e.g., enhance user experience, optimize resource allocation)

The user story serves to illustrate that:

- the EVEREST SDK enables a new class of users to optimize their applications on heterogeneous, FPGA-accelerated infrastructure and substantially reduces development time
- based on the traffic prediction example below, potential users can map their own applications to personas and tasks and determine the potential benefits of using the EVEREST SDK
- the gaps identified in new users' stories will guide the evolution of the EVEREST SDK

For the example user story, we use the optimization and acceleration of the traffic prediction application described in Deliverable D6.3. The full user story can be replayed from a Jupyter notebook, e.g. the traffic prediction example notebook in the EVEREST SDK repository https://github.com/everest-h2020/everest-sdk/blob/main/examples/applications/traffic_prediction/01_traffic_prediction_full_flow.ipynb, which is sketched in Figure 14. However, EVEREST technology is adaptable and can support various types of applications, not just those related to traffic prediction, and the user story can be contextualized within a different application environment.

The user story involves the following personas:

- ML-Developer (little knowledge of FPGA-acceleration)
- Workflow Specialist (app deployment and integration)
- Performance Engineer (fixing problems and adding features to the SDK, remote)
- Infrastructure Specialist (manages heterogeneous, FPGA-accelerated platform)

ML-developer. *Skills: python / pytorch / tensor-flow; app development; HW-awareness.*

(S)he gets the job to develop an app that uses an AI-model to predict future congestion on each road segment of a city. (S)he trains a small model for each individual road segment. For a mid-size European city, this leads to few 10000 models which all have the same architecture but different weights. (S)he integrates all models into an app, which predicts congestion in the near future (up to 1hr ahead) for all road segments of the respective city. After delivering the app to the workflow specialist, (s)he receives a note telling her/him that her/his app is several orders of magnitude too slow (by the time the app returns a result, the prediction is no longer valid).

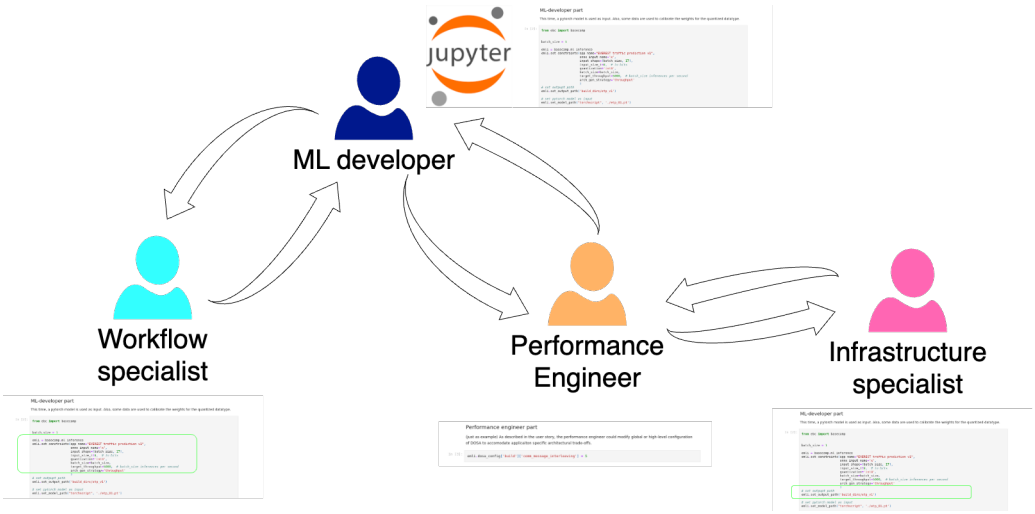


Figure 14 – User Story of the EVEREST SDK

(S)he finds that re-loading the model 10,000s of times destroys performance and also learns from the workflow specialist that their infrastructure offers FPGA devices for acceleration. (S)he changes the application by exploiting the fact that the model architecture remains unchanged and annotates some performance-critical components in the code. The next step is to use the FPGA-aware compilation flow to produce an accelerated version of her/his app. (S)he downloads the EVEREST SDK and uses it to compile her/his annotated code for FPGA-accelerated heterogeneous infrastructure. The basecamp facade compiler (https://github.com/everest-h2020/everest-basecamp) removes the barriers usually preventing the use of FPGA-accelerated infrastructure by providing a simple CLI to get started:

```
ebc-cli ml_inference onnx --json-constraints <path-to-json-constraints> <path-to-model.onnx> <
    path-to-output-directory>
```

Listing 1 – Example to compile and generate HLS/HDL files for a ML model represented in ONNX.

`ebc-cli` stands for EVEREST Base Camp command line interface, wich is the common command line interface designed for integrating the tools of the EVEREST SDK. More details are explained with: `ebc-cli -help`. (S)he runs into some issues and involves a performance engineer with FPGA expertise to address them. Communication with the performance engineer is simple because they both work on a common Jupyter notebook. The result is an optimised version of the app.

The ML-developer now includes the build-flow in her/his standard CI-CD pipelines and adjusts a few parameters of the app to meet the updated specs (s)he receives from the workflow specialist. The automation allows her/him to make those adjustments without involving the performance specialist. After verification, (s)he hands off the new version to the workflow specialists who can now meet performance targets.

Workflow specialist (app deployment and integration). *Skills: understands end-user-application performance targets; masters the workflow tools incl. deployment and profiling; infrastructure aware.*

(S)he gets the job to deploy an application that identifies the shortest possible path through a large network of roads for 10'000s of requests. Sets up the workflow including an app that runs inference on an AI-model, which he gets from the ML developer. Sets up the end-to-end workflow and realizes that the inference task is orders of magnitude too slow for her/his requirements.

(S)he meets with the ML-developer and finds that the FPGAs available in her/his infrastructure portfolio might be able to provide substantial speed-up for the slow ML-inference app in his/her workflow. After some time, (s)he receives a FPGA-accelerated version of the app from the ML-developer, downloads the EVEREST SDK and uses the runtime components to re-deploy the FPGA-accelerated app on the target infrastructure. While the app runs out-of-the-box on the FPGA-accelerated infrastructure, some additional optimizations might be needed to optimize the performance of the data-exchange between the various components. This requires some adjustments in collaboration with the infrastructure specialist, (s)he finally meets performance targets.

Performance engineer *Skills: detailed knowledge of EVEREST SDK components involved in compiling to*

FPGA accelerated heterogeneous infrastructure; detailed knowledge of heterogeneous FPGA-CPU platforms

(S)he understands the details of the EVEREST SDK components required to compile ML applications to FPGA-accelerated heterogeneous infrastructure. Analyses the requirements of the ML developer. Based on this analysis, adjusts and optimizes the necessary configuration for the compilation tool-chain. For example, as shown in Listing 2, the performance engineer decides that a higher pipeline depth inside the communication layer of the FPGA would increase overall throughput and hence increases the `comm_message_interleaving` and `maximum_pipeline_store_per_node` parameters. The first parameter increases also the parallelism between computation and communication inside of the FPGA at the expense of latency. Another parameter that could increase the performance is the flag to allow multiple CPU clients as receiver of the results from the FPGA(s), to avoid a bottleneck in the network-stack of the CPU. Finally, the performance engineer can also tune the utilization goals of each FPGA to trade-off the difficulty of the FPGA place-and-route vs. the communication overhead between many FPGAs. Therefore, the performance engineer could allow DOSA to violate the utilization limit slightly, if this could reduce the required FPGA nodes, as shown in the lower half of Listing 2.

```
  "build" : {
    "comm_message_interleaving": 8,
    "maximum_pipeline_store_per_node": 16,
    "insert_debug_cores": false,
    "allow_multiple_cpu_clients": true
  },
  "utilization": {
    "max_utilization_fpgas": 0.90,
    "utilization_exception": 0.05
  },
```

Listing 2 – Updated DOSA config file to meet performance goals

Infrastructure specialist. *Skills: understands heterogeneous (cloud) infrastructure in general and the EVEREST target platform in particular; knows the runtime components of the EVEREST SDK needed to deploy in an efficient way applications on FPGA-accelerated heterogeneous infrastructure.*

(S)he gets a request from the workflow specialist and supports her/him in the debugging and optimization of an ML-inference app on the target infrastructure of the EVEREST SDK.

```
ebc-cli airflow create <workflow-name>
```

Listing 3 – Creation of a workflow using airflow with basecamp

This command creates a new airflow-based workflow, which is needed to execute the code through the target LEXIS platform with an off-load to IBMs FPGA-accelerated deployment. A number of optimizations are applied by modifying the initial workflow for maximum throughput and minimal latency of the data-transfers involved in the execution. In addition to that, the infrastructure specialist can exploit the EVEREST run-time to maximize the resource.

4 Highlights of Project Results and KPIs (on Use Cases)

This section aims to provide a comprehensive overview of the project's key highlights and achievements, in line with the project-level Key Performance Indicators (KPIs) defined in the project proposal. Each highlight encapsulates the project's significant outcomes, advancements, and overall progress. For a more thorough understanding of the methodologies employed and the results obtained, we have provided direct and convenient links to other project documents (deliverables) instead of duplicating the analysis. In addition to that, we revised a bit the order of the sub-challenges to make the story within the section easier to follow.

Challenge 1 - To foster the adoption of heterogeneous architectures within the community, by improving the quality of the results, while reducing the time-to-market, development cost, and programming effort.

- *CH1.1 - Reduce by 50% the development costs of building demonstrators by using the EVEREST design environment and alternative variants of the target system architecture.*

  The EVEREST project contributed in several ways to reduce the development cost of application demonstrators. As an example for the case of map-matching, the EVEREST SDK, and in particular the `dfg-mlir` dialect, `bambu`, and `olympus` provide developers with the necessary means to rapidly prototype and develop heterogeneous applications by allowing them to move kernels between hardware and software with a handful of simple annotations. Manual approaches require lots of glue code to be written which is tedious and error-prone. The utilization of high-level automatic transformations, high-level synthesis, and an MLIR-based toolchain for heterogeneous systems permits the straightforward evaluation of alternative designs, which would otherwise be cumbersome or impossible to assess. Similarly, this applies to other analyzed cases.

  Quantitative numbers are hard to extract. However, during the execution of the project, experienced hardware designers implemented kernels by hand that were later generated automatically with the help of DSLs and the EVEREST MLIR-based compilation toolchain (see Deliverable D4.5). In the case of WRF, manual designs for the radiation module took about 3 weeks, with 1 extra week of integration for testing. With the automated flow, these kernels can be lowered in a matter of hours. In addition to compilation/synthesis support, also the run-time part of EVERST (including Virtualization, application tuning, and distributed multi-node run-time management) reduced the cost for building working demonstrators and also a more efficient deployment: 1) it is possible to overcommit FPGA resources, thus enabling parallel prototyping of different applications, 2) virtual machine deployment is faster and easier to maintain and reproduce and 3) application developers don't need to buy the hardware anymore, they could rent part of an FPGA-virtualization enabled infrastructure from a cloud provider.

- *CH1.3 - Reduce programming efforts by one order of magnitude.*

  Domain-specific languages (DSLs) have emerged as a powerful tool in software development, offering a targeted approach to solving complex problems within specific domains. One of the key advantages of using DSLs is the significant reduction in programming effort they afford. By designing languages tailored to specific domains, developers can express concepts and tasks in a more natural and intuitive manner, abstracting away low-level implementation details. This results in increased productivity, as developers are able to focus more on domain-specific challenges rather than dealing with the intricacies of a general-purpose programming language. Furthermore, domain-specific languages (DSLs) facilitate code reusability and maintainability, as domain experts can easily understand and modify DSL-based code, thus accelerating development cycles and enhancing overall software quality. In particular, within EVEREST, DSLs are known to be much more concise than implementations using mainstream programming languages like C/C++, Fortran or Rust. At the same time, DSLs can be leveraged to more easily explore different variants (at compile-time or at runtime), as done within EVEREST where the DSLs are used as the front-end of a more complicated toolchain composed of different optimization modules. This was demonstrated earlier in the project in Deliverable D4.2 where from a simple specification (Figure 2 in Deliverable D4.2) multiple variants were created with almost zero effort.

  To provide a more precise example on the EVEREST Use Cases, when dealing with WRF, our numpy-like EKL DSL (see Deliverable D4.5) uses 100 lines of code that replace the implementation of the kernels in the RRTMG module which uses more than 2,000 lines of legacy Fortran. This example should also

consider that the use of the automatic generation of offloading and linking removes lots of boilerplate code. Moreover, what is still hidden is that the EVEREST SDK automatically generates lot of synthesis and integration scripts which otherwise require human intervention. For example, Bambu automatically generates hundreds of Tcl code lines to control the simulation and synthesis backend scripts. Moreover, the automatic generation is done in a flexible way since the synthesis scripts depend on the type of FPGA technology used, so they require a deep understanding of the RTL synthesis backend used. Similar happens to `dosa` which enhances the developer productivity significantly by allowing to compile a ONNX model to (multiple) FPGAs with just one command, without requiring the user to have detailed FPGA knowledge (see Deliverable D4.5). Hence, using `dosa`, what was previously the work of multiple days, if not weeks, is now possible within hours with almost no interactions;

- *CH1.2 - Reduce by 50% the time to identify anomalies.*

  The EVEREST SDK allows developers to deploy anomaly detection modules at any point throughout their workflow with minimal effort. Anomaly detection can serve as *input sanitization* to protect the models or to detect other security events. Within the project a library called ADLib has been developed (see Deliverable 3.2). Furthermore, it has been integrated within the use cases. Within the traffic use case, integration had to be done at a lower level, meaning ADLib was not used as a stand-alone library but as a function within their existing codebase. This use case was chosen as a target to accelerate the internal anomaly detection model. Concretely, the software implementation of the function which represented the largest part of execution time ($>$ 99%) has been accelerated onto an FPGA using `dosa`. This resulted in a drastic improvement in execution time from a range of 1.2k - 45k microseconds in software depending on the input data, to a worst-case execution time of 1.318 microseconds on an FPGA. In addition to that, as shown in the related table of Section 2 of this document, all the injected anomalies within a dataset from **DUF** have been identified, demonstrating a great improved capability of the developed methods. The target value of 50% can be considered as met.

Challenge 2 - To enable the acceleration of computation and data accesses on heterogeneous distributed architectures.

- *CH2.2 - Improve by 2x the execution performance.*

  As an activity within the project, but also part of our assessment procedure of the project KPIs within the project, we have developed the use case demonstrators tailored to different target fields utilizing the EVEREST design environment. These demonstrators served as driver for requirements but also as practical implementations showcasing the capabilities and functionalities of our project.

  During the evaluation phase, we identified a heterogeneous behavior in terms of the advantages (and disadvantages) of using the EVEREST SDK (and an FPGA acceleration). These differences originate from different sources, the complexity of the analyzed kernels, the maturity of the tools involved, and also when the final kernel has been made available for optimization within the project timeframe.

  Regarding the WRF-based use cases, we already explained in other deliverables (i.e. Deliverable D6.3 and Deliverable D4.5) that we focused on the most computationally intensive part, namely, the radiation module. As explained in Deliverable D4.5. The computational cost of the `taumol_sw` function was reduced by 3$\times$ by adopting the EVEREST SDK. Similar but even better, has been the adoption of the EVEREST SDK to support the the Map-Matching application within the traffic modelling use case. This was also due to the application type (more dataflow oriented) that offered multiple kernels to offload on the FPGA (see Deliverable D6.3 for more details). Offloading it to hardware and replicating it has yielded speedups of more than 6$\times$ against an optimized version (while more than 30$\times$, with respect to the version originally available at the beginning of the project). Details on the Map Matching optimizations and results can be found within Section 2.3.4 of this document (Objective 3.4 - Improve the overall performance of map matching kernel). In the case of Traffic prediction, the ML inference deployed on the **IBM** cloudFPGA platform results in a user-side end-to-end speed-up of 24$\times$ for the prediction of a road segment using 8x kernel parallelisation option, which is compared to a CPU-only solution. The results are reported within Section 2.3.2 of this document (Objective 3.2 - Improve the overall performance of neural network traffic prediction).

- *CH2.1 - Reduce by 30% the energy cost.*

  In comparison to traditional CPUs, FPGAs have consistently been identified as devices offering significant advantages in terms of energy efficiency. This is due to the fact that FPGAs, and in particular the logic designed on them, are tailored to perform specific tasks with respect to the more general-purpose CPUs. Consequently, they represent an attractive choice for applications where energy efficiency is important. However, this is not always true since the data transfer cost and the use of larger devices can in same case erode this gain. In EVEREST, we had the opportunity to measure the energy consumption for two modules of the traffic use case on different platforms (cloudFPGA and Alveo-based).

  For the Traffic prediction, the ML inference deployed on the **IBM** cloud-FPGA platform with the maximum parallel configuration option, achieving 24x speedup compared to the CPU only solution, uses approximately 5x less energy (-79%) than the CPU counterpart [6]. For more details see Objective 3.2 within Section 2.3.2 of this document.

  In the case of Map-matching for the sweat spot architectural configuration using the bus-attached FPGA option (Alveo-based), which yielded 6× speedup compared to 16-core CPU architecture [7], we similarly obtained 5× less energy consumption (-80%) with respect to the CPU execution. Details can be found in Section 2.3.4 of this document.

Challenge 3 - To significantly improve the results of the target UCs renewable energies production prediction, air-quality monitoring, traffic modeling, and prediction.

- *CH3.1 – Improve by 10x the performance of simulations for renewable energies prediction.*

  The primary objective of the energy prediction use case was to facilitate the internal deployment of a **DUF** application, thereby eliminating the need to rely on external providers. This goal has been fully achieved since the entire end-to-end pipeline, including data collection, WRF execution, data post-processing (with anomaly detection) and ML power prediction, has been deployed and was operational during the final period of the project (and tested on a large set of historical data). In particular, the ML engine developed within EVEREST and the pre- and post-processing methods demonstrated a performance that was very close to that of the best provider on the market (see Section 2.1 of this document). The scenario of one WRF forecast per day for the 24 hours of the day ahead proved to be sufficient to make the next day's forecasts suitable for sale in the day-ahead market. Therefore, it was not necessary to accelerate WRF. However, more frequent forecasts were expected to reach the intra-day market and/or to adopt predictions based on ensemble data. Within the entire energy prediction workflow the part most computationally intensive part is without any doubt the data generation using WRF. The lack of acceleration within a production environment for the end-to-end WRF workflow reduced the expected huge impact of this activity.

- *CH3.2 - Improve by 2x the response time of the air-quality predictions.*

  Despite the final demonstration of EVEREST development (new WRF-IFS workflow and new air-quality ML approach) was not performed on final outputs (air pollution events) due to the limited time, an actual improvement in the forecast of main weather parameters (wind and temperature) are observed. Globally, an improvement of around 50% have been obtained in terms of the major quantities to be monitored for detecting pollution events, as shown in Section 2.2.3 of this document, mainly due to the new ML approach. Unfortunately, within the project time frame, the project was not able to finalize the end-to-end accelerated Workflow related to WRF. The estimations done on the WRF kernels provide hints on the possibility to further accelerate the execution by a global end-to-end speedup of 10%. However, due to this good performance, **NUM** started to test the ML approach for some specific clients outside the EVEREST project, and as indicated in Deliverable D7.7, its exploitation is clearly planned by **NUM**.

- *CH3.3 - Improve by 3x the overall performance of traffic model framework.*

  The traffic simulator model framework has greatly benefited from various improvements across its modules (e.g. ML Traffic Prediction, Map Matching). On the side of the traffic simulator itself, the integration

[6] The CPU profiling measures have been taken on an AMD EPYC 7643 CPU.
[7] The CPU profiling measures have been taken on an AMD EPYC 7282 CPU

of evkit enabled not only the parallelisation of parts of the code as also supported the distribution of the computation across the multiple nodes of HPC clusters. These improvements enabled for instance a faster and more distributed computation of alternative routes, the implementation of an approach for accessing map updates in a distributed manner or the offloading of parts of the computation to FPGA. These improvements allowed the traffic simulator to support simulations with at least 250,000 vehicles, compared to the previous 50,000, and the project results indicate that the simulator supports simulations with even more vehicles. At the same time, the results have indicated a speed-up of more than 100 times compared to the baseline version. Additionally, the usability of the simulator has also been improved as it now supports deployment within the LEXIS platform, providing an additional deployment option for its users.

5 Conclusion

In the final phase of the project specific efforts were dedicated for planning, execution, and evaluation of the impact of our efforts across the three distinct use cases. In summary, this document provides a comprehensive overview of the project's significant achievements in relation to the requirement of the use cases and in terms of project's Key Performance Indicators (KPIs). Particular attention has been considered to the main project claims which were on the reduction of development costs and programming efforts, improved performance, and energy efficiency through the use FPGAs and by exploiting the EVEREST design environment. The use case evaluations were conducted considering both functional and extra-functional results. Additionally, a user story detailing the seamless usage of our EVEREST System Development Kit (SDK) for the traffic prediction module has been included, focusing the attention on the involved entities.

Within the document, we tried to underline the key advancements and challenges that could also be found within the requirements of the application providers: (i) Enhanced adoption of heterogeneous architectures by reducing development costs, programming efforts, and anomaly identification time; (ii) Accelerated computation and data access on heterogeneous distributed architectures, resulting in improved execution performance and reduced energy costs; (iii) Significant improvements in renewable energy prediction simulations, air-quality predictions, and traffic model framework performance. With this perspective, we can mention that several lessons were learned while making the wrap-up of the document. The results reached by the project were heterogeneous, with several successes, but also some failures. The latter have been determined by the computation patterns adopted by the target kernels, such as PTDR, or by the large complexity of the target modules, like WRF-RRTMG, that did not limit the required effort to the project claimed contribution. The achievements are nicely reported in the previous section linked to the target project KPIs.

In addition to that, before closing the document, we can describe our conclusions on the use of the EVEREST SDK to the target UCs. Within the project, it became evident that increasing design complexity necessitates the adoption of high-level synthesis (HLS) to enhance productivity. Utilizing HLS for generating both accelerator kernels and the associated system architecture allows for targeting various platforms more effectively. The use of various types of DSL could even increase the abstraction. However, it was also clear that acceleration could only be applied to selected portions of applications, necessitating complex and time-consuming code rewriting. For what we experienced, a solution to the old problem of HW/SW partition requiring to have a good understanding of the code functionality is still necessary, since it impacts a lot the resulting performances. Furthermore, hardware generation was sometimes limited by FPGA resources, particularly memory blocks, and custom data formats required careful accuracy analysis at the application level. The project also highlighted the significant challenges involved in porting large and legacy codebases (WRF) to FPGA-compatible formats. However, this large effort that impacted also for a long interval of time within the project, limited then the optimization possibilities. On a positive note, the modularity and interoperability of the SDK were significant added values, enabling the integration of different dialects, tools, and platforms, which greatly enhanced the flexibility and utility of the EVEREST SDK.

To conclude, we think that, in addition to the results and achievements we had within the project, we gained lot of knowledge and insights that could provide a foundation for future improvements and applications in FPGA support tools.

References

[1] Bas Harenslak and Julian de Ruiter. Data Pipelines with Apache Airflow. Manning Publications, 2021. ISBN: 9781617296901.

[2] Robert Pincus, Eli J. Mlawer, and Jennifer S. Delamere. Balancing accuracy, efficiency, and flexibility in radiation calculations for dynamical models. Journal of Advances in Modeling Earth Systems, 11(10):3074–3089, 2019.