EVEREST: A design environment for extreme-scale big data analytics on heterogeneous platforms

Christian Pilato[†], Stanislav Bohm[‖], Fabien Brocheton[‡‡], Jeronimo Castrillon[§], Riccardo Cevasco[††],
Vojtech Cima[‖], Radim Cmar[x], Dionysios Diamantopoulos[*], Fabrizio Ferrandi[†], Jan Martinovic[‖],
Gianluca Palermo[†], Michele Paolino[**], Antonio Parodi[¶], Lorenzo Pittaluga[††], Daniel Raho[**],
Francesco Regazzoni[‡], Katerina Slaninova[‖], Christoph Hagleitner[*]

[*]*IBM Research Europe, Switzerland*, [†]*Politecnico di Milano, Italy*, [‡]*Università della Svizzera italiana, Switzerland*,
[§]*Technische Universität Dresden, Germany*, [¶]*Centro Internazionale di Monitoraggio Ambientale, Italy*,
[‖]*IT4Innovations, VSB – Technical University of Ostrava, Czech Republic*, [**]*Virtual Open System, France*,
[††]*Duferco Energia, Italy*, [‡‡]*NUMTECH, France*, [x]*Sygic, Slovakia*

***Abstract*—High-Performance Big Data Analytics (HPDA) applications are characterized by huge volumes of distributed and heterogeneous data that require efficient computation for knowledge extraction and decision making. Designers are moving towards a tight integration of computing systems combining HPC, Cloud, and IoT solutions with artificial intelligence (AI). Matching the application and data requirements with the characteristics of the underlying hardware is a key element to improve the predictions thanks to high performance and better use of resources.**

We present EVEREST, a novel H2020 project started on October 1st, 2020 that aims at developing a holistic environment for the co-design of HPDA applications on heterogeneous, distributed, and secure platforms. EVEREST focuses on programmability issues through a *data-driven* design approach, the use of hardware-accelerated AI, and an efficient runtime monitoring with virtualization support. In the different stages, EVEREST combines state-of-the-art programming models, emerging communication standards, and novel domain-specific extensions. We describe the EVEREST approach and the use cases that drive our research.

I. Introduction

Thanks to the pervasive use of technology, we collect volumes of data that are doubling every two years. *Big Data analytics* aims at extracting hidden knowledge from the data and take valuable actions, often with the support of *artificial intelligence* (AI) [1]. For example, precision medicine can create custom medical practices after analyzing patients' data that are continuously collected. Data sources are inherently distributed and heterogeneous, while accurate predictions and decisions often lead to demanding processing requirements. High-Performance Big Data Analytics (HPDA) applications expose a high parallelism and can benefit from hardware acceleration, but the limited bandwidth and the excessive power consumption for data movements put high pressure on communication and storage. HPDA applications thus require (1) efficient hardware acceleration for data processing [2], [3], (2) communication cost reduction, by moving the computation closer to the data sources [4], [5], and (3) data protection from unauthorized accesses during all application phases [6]. To accelerate data processing, HPDA systems will combine heterogeneous nodes [7], with general-purpose processors and FPGA devices, across different technologies (e.g., HPC, cloud computing, and edge devices). Optimizing communication and storage requires to match the characteristics of the target system (e.g., distribution of the nodes and communication infrastructure, size of on-chip and off-chip memories, and number of memory channels) and the applications (e.g., data distribution and access patterns). AI methods and security threats may impose additional application and architectural constraints, especially when the data and the computation are geographically distributed. Since a one-fits-all solution is impossible, future HPDA systems will be *data-driven* with application-specific optimizations to match the application requirements, the nature and locality of the data, and the hardware characteristics [4]. Programming such systems necessitates the use of complex data management techniques and domain-specific annotations, which are not well supported in current design frameworks. This leaves most of the effort to the application developers. Solutions to these programmability issues demand methods to represent functional and non-functional properties, drive the hardware-software compilation, and dynamically manage the underlying distributed hardware in order to obtain fast, scalable, and secure HPDA systems.

The EU project EVEREST (dEsign enVironmEnt foR Extreme-Scale big data analyTics on heterogeneous platforms - http://www.everest-h2020.eu) proposes a design environment for HPDA applications on distributed and heterogeneous systems. The EVEREST target system seamlessly combines nodes with IBM POWER9 CPUs and coherent FPGA accelerators (for cloud computing), and disaggregated FPGA devices [8] (for edge computing). The EVEREST design environment complements state-of-the-art programming models (e.g., OpenCL, SYCL, OpenMP) with domain-specific extensions to (1) provide extra characteristics of the algorithms and data, (2) exploit the available hardware resources with alternative code/hardware variants, (3) promote the use of high-level synthesis (HLS) [9] for generating AI accelerators, and (4) improve the dynamic control of the distributed execution [10], [11].

II. EVEREST Approach

Our **EVEREST System Development Kit (SDK)** is a design environment to ease the description, optimization and execution of Big Data applications with heterogeneous data sources onto FPGA-based architectures, operating at design and run time.

At *design time*, we focus on (1) the application description along with non-functional requirements, (2) the generation of several hardware and software variants, and (3) the customization of the distributed memory architecture. We aim at developing a data-driven hardware/software compilation framework that takes as input an application description using a combination of workflow libraries, AI libraries and frameworks, and domain-specific extensions. The compilation engine explores code variants and uses HLS for generating hardware accelerators. We represent the resulting application with mainstream parallel programming models (like SYCL). Flexible memory managers will enable to co-optimize computation, communication, and storage, to move the computation closer to the data, and to implement hardware-assisted data protection.

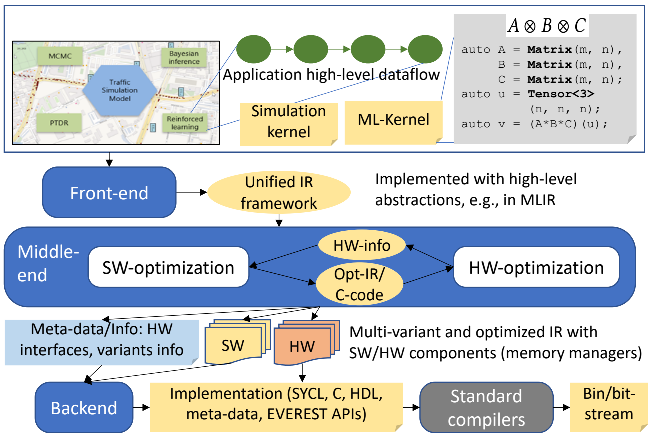
At *runtime*, we build a virtualized environment to dynamically select the code variant to execute for each task, based on the workload and data conditions. The virtualized environment will abstract hardware characteristics of the EVEREST nodes (based on different CPU architectures e.g., x86 on the cloud and ARM/RISC-V on the edge) to present an integrated execution environment for the applications. This combined solution allows designers to match the data requests with the underlying hardware to optimize the data transfers, exploit the spatial parallelism with the hardware accelerators, and react to changes in the workload conditions.

III. Data-driven Compilation Framework

A. *Application specification and definition of requirements*

The EVEREST design framework receives as input the application description (i.e., a workflow pipeline where each node can be specified in C/C++ or with proper AI libraries). Industry-grade applications often encompass end-to-end data processing workflows composed of a large number of interconnected computational tasks of various granularity. EVEREST will feature a scalable platform based on HyperLoom [10] for describing and executing complex workflows in large scale distributed environments with various virtualized heterogeneous resources. The envisioned platform aims to improve resource utilization and reduces the overall workflow processing time.

Application experts are offered **embedded domain-specific languages** (DSLs) to express the semantics and security requirements of computational tasks to enable high-level code optimizations. DSL extensions have been successfully demonstrated in many domains, such as computational fluid dynamics [12], hybrid particle-mesh simulations [13], tensor expression optimizations [14]–[16], and dataflow languages [17]. EVEREST proposes a data-centric approach for security, dealing with confidentiality, authentication and integrity of the data handled by the system with **hardware-assisted data protection** applied to both edge devices and data center nodes. EVEREST will propose a comprehensive library of optimized accelerators for memory and near memory encryption, fitting the area, energy and performance constraints of the platforms. We will include information flow tracking, monitoring, and protection against malicious uses, including side-channel and buffer-overflow attacks [18].


MCMC
Bayesian inference
Traffic Simulation Model
PTDR
Reinforced learning
Application high-level dataflow
Simulation kernel
ML-Kernel
$A \otimes B \otimes C$
Front-end
Unified IR framework
Implemented with high-level abstractions, e.g., in MLIR
Middle-end
SW-optimization
HW-info
Opt-IR/ C-code
HW-optimization
Meta-data/Info: HW interfaces, variants info
SW
HW
Multi-variant and optimized IR with SW/HW components (memory managers)
Backend
Implementation (SYCL, C, HDL, meta-data, EVEREST APIs)
Standard compilers
Bin/bit-stream

```
auto A = Matrix(m, n),
     B = Matrix(m, n),
     C = Matrix(m, n);
auto u = Tensor<3>
           (n, n, n);
auto v = (A*B*C)(u);
```

Fig. 1. Overview of the data-driven compilation flow.

EVEREST aims at developing a unified MLIR representation for the transparent support of several high-level ML frameworks (e.g., TensorFlow or PyTorch) and high-level optimizers (e.g., XLA, Glow, TVM) [19].

B. *Generation of software and hardware variants*

DSLs will be used to concisely express performance-critical functionality and annotate data characteristics and requirements. Tensors and particles are two examples of EVEREST data-centric programming abstractions that will enable optimization of data communications and generation of custom memory subsystems. DSLs for expression languages will enable highly-optimized kernel generation either in software or hardware to enlarge the optimization space [14], while allowing more control for provably safe execution [15].

For a higher-level coordination of the workflow kernels, EVEREST will look at functional abstractions to implicitly express the application dataflow [17], [20] and its integration on HyperLoom [10]. The compiler front-end unifies the orchestration and the kernel specifications into a single MLIR as shown in Fig. 1. We will extend the LLVM compilation framework [21] with dedicated MLIR dialects [22] for domain-specific kernels. The tool chain will support standard exchange formats used in machine learning (e.g., NNEF or ONNX).

The middle-end of the compilation flow will rely on high-level architecture models [23, Chapter 6] [24] and simulators [25], [26] to explore the design space and create **multiple hardware and software variants**. These variants are performance/energy trade-offs that are exposed to the runtime system. For instance, a software-only implementation could explore layouts of particles as array-of-structures or structure-of-arrays, or could tile complex tensor expressions to fit the memory hierarchy while allowing different threading implementations for the runtime. Hardware variants could implement a chain of tensor operations directly on the FPGA logic before writing back to main memory. Hardware/software partitioning will be driven by annotations and the two parts will be co-optimized, including hardware estimations for code-snippets (cf. Fig. 1).

EVEREST will leverage FPGA resources to create **hardware accelerators with high-level synthesis**, especially for dataintensive and AI tasks. In EVEREST, we use Bambu, an open-source HLS tool based on both GCC and LLVM [27]. Bambu will optimize execution and memory bandwidth of accelerators. Data distribution introduces additional challenges in terms of variable read/write latency and energy based on the location of the data. Since the memory behavior of an application ranges from statically predictable patterns [3] to irregular memory accesses [2], we will use a **fully automated and transparent memory management** at both compile time and runtime with a combination of polyhedral-based transformations [28], multi-port memories [29] and dedicated micro-architectures to schedule the memory accesses [5], interleave the memory requests and hide the communication latency with the distributed memories [30]. We will generate and optimize such accelerators based on the information extracted from the DSL annotations. EVEREST will extend high-level synthesis for the automatic integration of security features, like application-specific dynamic information flow tracking [18], [31]. We will also develop and use a library of cryptographic functions, to ensure data integrity, confidentiality, and authentication. Such cryptographic routines will match application requirements and dynamic behaviors. Dedicated hardware monitors will detect anomalies with respect to the expected data behaviors (timing patterns, access patterns, typical sizes and ranges), activating proper dynamic adaptation in the form of "auto-protection".

Given the set of variants, the backend will generate software implementation relying on state-of-the-art programming models (e.g. SYCL) to enable seamless integration in the tooling infrastructure. Meta-information about the variants will be provided to the runtime system to support dynamic selection. Finally, standard toolchains will be used to generate binaries and bitstreams for the target devices.

IV. Virtualization-based Runtime Optimization

EVEREST features a **distributed runtime support** to manage and coordinate the computation across the different system nodes. Tasks are defined in a way that allows runtime migration of both data and computations. FPGA accelerated applications and the runtime framework will be designed with a **virtualized environment** to abstract the hardware resources. This approach improves efficiency and security. Also, we will be able to seamlessly move the computation between edge nodes and also between edge and cloud parts. The runtime layer optimizes the use of heterogeneous and distributed resources by parallel application instances running in different virtual machines (VMs). The EVEREST virtualized runtime environment automatically manage the code to run and configure the hardware based on the workload conditions and the data distribution. virtualization techniques will abstract hardware characteristics of the heterogeneous target nodes to present an integrated execution environment for the applications. As described in Section V, the nodes may feature different CPUs (i.e., x86 in the cloud and ARM/RISC-V in the edge [32]) and accelerators (e.g., GPUs and FPGAs [33]). Fig. 2 shows an overview of the EVEREST virtualized runtime environment. Its implementation includes both hypervisor and guest OS extensions to manage, optimize,

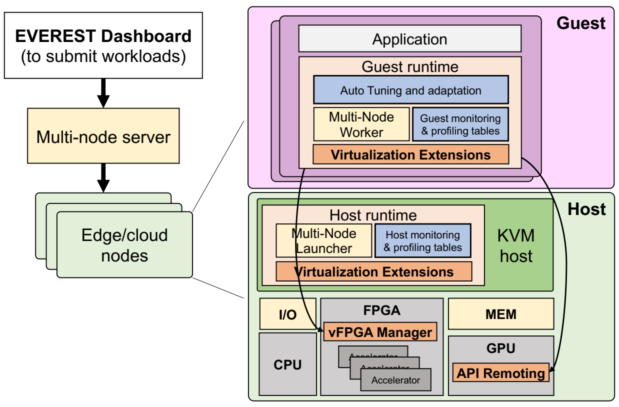


Fig. 2. Virtualized runtime environment overview.

and monitor the access to hardware from guest applications. These extensions provide:

1) **Data protection layer.** The system monitors the execution to identify malicious attacks (see Section III-A and react by using the security mechanisms added by the compiler.
2) **Dynamic hardware-software adaptation strategy.** We propose an intelligent policy to select the code variant or hardware configuration to execute, among the ones pre-generated at compile time, based on the system status.
3) **Virtualization support and hypervisor extensions.** Hardware configurable parameters, including accelerator APIs, are exposed directly to the applications inside the VMs, requiring also guest OS enhancements (e.g., drivers).

The EVEREST virtualization environment will interact with the underlying hardware to select the variants to execute. EVEREST provides **dynamic application auto-tuning capabilities** based on mARGOt [11], a dynamic decision-maker that performs an automatic selection of the variant to execute for each critical kernel identified at compile time. For example, a variant that makes heavy access to unavailable hardware resources can be replaced by a variant that fits better with the system status. This selection is based on (1) the dynamic characteristics of the target system (e.g., available resources) [34], (2) the optimization goal set for execution (e.g., performance or energy consumption) [35], [36], (3) the additional dynamic requirements (e.g., security monitoring, data features [37]), and (4) the available techniques for data management (e.g., data representations and distributed allocation). The selection will generalize the concept of affinity between the code variants and the available system configurations and requirements. Hardware monitors will collect the information to make the selection.

Guest programs will configure the underlying hardware or make specific requests based on workload conditions, environment changes and the availability of specific hardware resources (e.g., communication channels, remote notes). API remoting techniques will improve data exchanges. The distributed runtime also leverages the configuration of pre-defined hardware resources for deep learning, like reconfigurable AI networks.

V. EVEREST Target System

In EVEREST, we envision a hierarchy of processing environments, as shown in Fig. 3. The outermost layer (*End-*

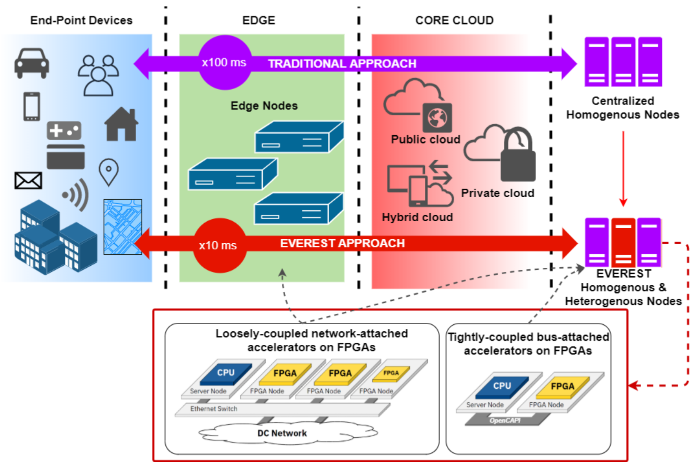


Fig. 3. EVEREST ecosystem overview.

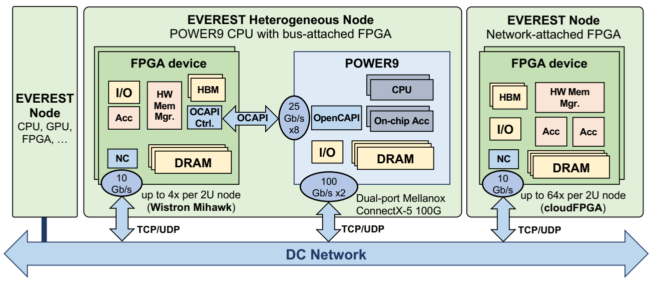


Fig. 4. EVEREST featured system as a combination of heterogeneous nodes with OpenCAPI cache coherent and TCP/UDP protocols.

Point Devices) receives the stream of data and performs initial processing under strict latency constraints and with the limited performance available in end-point devices. These requirements dictate very fast data pre-processing, inference and perhaps only limited training. Depending on the application, these edge nodes can be complemented by an inner-edge environment that does more extensive processing, training and data analysis. The inner-edge environment features more powerful hardware and less stringent requirements for real-time processing. The results of this layer are then forwarded to the core cloud services (public, private or hybrid), where more extensive analysis and model building is performed on heterogeneous hardware.

Today's edge nodes are typically scaled versions of cloud servers, which primarily combine CPUs with tightly-coupled co-processors (e.g. GPUs). However, CPUs and GPUs are optimized towards batch processing of in-memory data and can hardly provide deterministic performance for the processing of streaming data coming from the I/O channels of end-point devices. Future edge servers call for a new heterogeneous computing node tailored to the processing of streaming data at low power consumption and high energy efficiency. To support this vision, the EVEREST project targets distributed architectures composed of industry established computing nodes, with CPUs and GPUs, as well as experimental heterogeneous nodes with FPGAs. Each experimental node may feature one or more FPGA devices for hardware acceleration and one or more physical memories (either local or external to the FPGA), as shown in Fig. 4. Such systems will run Linux as Operating System (OS) and a hypervisor to manage the hardware resources. Note that the EVEREST approach is not limited to these architectures. In fact, specifying the workflow pipelines at a higher level of abstraction, within the specifications of EVEREST SDK and virtualization technology, as discussed previously, will enable the porting of the applications to architectures with heterogeneous GPU-based nodes and end-user embedded devices. We aim at developing a **small multi-node demonstrator** based on the technology and the components available during the project's timeline. To develop the EVEREST SDK for heterogeneous systems, we focus on two state-of-the-art FPGA-based research platforms: a CPU-managed system that rely on tightly-coupled bus-attached FPGAs [38] and an FPGA-disaggregated system that relies on loosely-coupled network-attached FPGAs [8]. The final EVEREST demonstrator will feature both nodes to examine how the different architectural configurations can accommodate big data workloads at the edge and on the cloud.

In EVEREST, the POWER9 node with tightly-coupled bus-attached FPGAs forms the basic platform to research these challenges. In addition, the system is augmented with loosely-coupled network attached FPGAs to increase the parallel processing capability from multiple I/Os streams. The latter platform will be also evaluated as an edge node, where processing closer to the multiple I/O streams can offer great performance due to the low-latency and high-energy efficiency of the FPGAs. Both bus-attached FPGAs and network-attached FPGAs are envisioned as a scale-up opportunity of the POWER9 node, while multiple such nodes will be extended across the data-center, as a scale-out configuration.

The demonstrator based on loosely-coupled network-attached FPGAs will rely on the cloudFPGA platform [8]. CloudFPGA is a research platform that disaggregates the FPGA accelerator from the server, turning it into a stand-alone computing resource. Such network-attached FPGAs can be deployed at large scale and independently of the number of CPU servers in the data-center (DC). The network attachment allows them to seamlessly connect with each other as well as with one or more CPUs. The resulting disaggregated heterogeneous computing infrastructure is capable to dynamically adapt to the scale of any workload. Meanwhile, large-scale applications ranging from business analytics to scientific simulations and AI have started to scale out using distributed frameworks such as Hadoop, Spark, HyperLoom, and Tensorflow. The cloudFPGA platform enables a user to acquire, distribute, configure and operate stand-alone network-attached FPGAs at large scale in DC infrastructures. The use of a shell-role architecture combined with partial reconfiguration provides for isolation of the system management functions from the user logic within the network-attached FPGA. This approach protects the integrity of the DC network by creating a separation between privileged and non-privileged user logic functions.

VI. EVEREST Use Cases

We drive our research with three industrial applications: (1) a weather analysis-based **prediction model for the energy trading market**, (2) an application for **air-quality monitoring** of industrial sites, and (3) a **traffic modeling framework for intelligent transportation** in smart cities. The applications are representatives of future HPDA applications: they have large and heterogeneous data sets (*Volume* and *Variety*), including historical and real-time data (*Velocity*) with important security concerns during communication and storage (*Veracity*). They are also aligned with the United Nations Sustainable Development Goals (no. 7, 9, 11 and 13).

A. *Weather-based predictions for renewable energy production*

In 2017, for the first time, the European Union generated more electricity from wind, solar and biomass than from coal according to new analysis from *Sandbag* and *Agora Energiewende*. The European energy market shows a strong interplay between the different energy sources: for example, a drought period can affect the hydroelectric production, demanding gas generation to counterbalance. Also, the prediction of energy production from renewable sources (in particular wind) is uncertain. Renewable energy production forecasting systems currently rely on an ensemble of meteorological predictions provided by global circulation models with grid spacing between 15 and 25 km and hourly temporal resolution. This ensemble predicts variables such as 2m temperature, near-surface wind speed, incoming solar radiation, and rainfall depth, that become input of a subsequent deep learning model trying to characterize the complex input/output relationship of the given power plant under consideration. Even using ensemble approaches, large uncertainties still exist when operating forecasting systems based on meteorological variables predictions at tenths of km's, especially when dealing with sudden local changes in cloud cover and wind intensity.

In EVEREST, we aim at reducing the cost of imbalance in case of severe meteorological ramp-up/down events. The application will forecast the energy produced by a wind farm in the next day with a 24-hour prediction on a hourly basis. Thanks to transparent hardware acceleration, we will be able to increase the resolution of weather forecast ensembles to better predict high-localized meteorological variations at hourly scale [39], [40]. Thanks to AI tools, we will combine the resulting weather models with historical data.

B. *Air-quality monitoring in industrial sites*

Every year new publications show the impact of air quality on public health. The latest figures from the World Health Organization (WHO) show that air pollution kills an estimated seven million people worldwide every year. WHO data shows also that 9 out of 10 people breathe air containing high levels of pollutants, and that the economic impact of this pollution on health is estimated to 5.7K billions of dollars per year. Industry contributes to this impact and must adapt on the one hand to increasing regulatory constraints and on the other hand to citizen pressure, in particular due to the development of low-cost air-quality sensors providing massive amounts of (low quality) spatial information. To support manufacturers, NUMTECH offers Plum'air, a service that allows an industrial site to collect real-time information about the monitoring and control of the pollution. In forecast mode, it can be used as a decision tool for an industrial site to adapt its activity in order to reduce its impact, especially in the transition phase before the implementation of heavy investments in terms of emission treatment or reduction systems. In this mode, Plum'air aims at forecasting the environmental impacts due to atmospheric releases of an industrial site at local scale (within 10 km from emission sources).

In EVEREST, we forecast the environmental impacts of chemical pollutants combining high-resolution weather ensembles with local data. Together with hardware acceleration, we will be able to obtain accurate information about the environment so that the industrial site can promptly delay production activities that may have an impact (e.g., increase of atmospheric releases) or activate emission reduction treatments.

C. *Traffic modeling for intelligent transportation*

Traffic modelling and prediction is a critical component for Smart Cities to build their intelligent traffic management system (ITS). Our approach for designing such a component is by creating a traffic modelling ecosystem comprised of tightly coupled processing elements such as reading big sensory data real-time and of a long-history records; traffic simulator which boosts the raw sensory data dataset into rich training sequences; traffic prediction model which learns from the training data set; route calculation as a service exploiting traffic prediction model. As the main data input into the system we will use provisioned origin-destination matrix (O/D) and a large historical data set of floating car data (FCD). FCD is represented by geo position and the speed of vehicle sensed approximately each 5 seconds from navigation devices, that is from millions of devices every day over the period of several years. However, our model will operate on selected cities (like Vienna) counting thousands of vehicles daily. Traffic simulator simulates individual clients driving around the smart city by combining both macro and microscopic approaches, optimizing the traffic flow [37], [41], [42]. The simulator calculates traffic model in near-real time while it requires access to historical records and a streaming long data chunks for ML predictions. It updates the traffic model for various conditions as well as it can generate training sequences for traffic predictions.

In EVEREST, we will improve the key processing components of the traffic modeling eco-system. The use of efficient AI methods will allow the edge nodes to collect and process more data, while addressing all privacy and security concerns. Also in this case, the EVEREST SDK will allow non-expert designers to easily express the application requirements for the compilation framework and the runtime system.

D. *Why using the EVEREST SDK?*

The applications will demonstrate how the EVEREST SDK can unleash novel market opportunities for the respective companies. The EVEREST SDK will provide following benefits:

- **Quality of predictions**: the possibility of integrating real-time and historical data by means of AI will allow more accurate predictions. This aspect is crucial for all applications as their commercial value lies on precise and timely knowledge extracted from the data.

- **Performance and energy efficiency**: the efficient use of heterogeneous resources and, in particular, hardware acceleration will reduce the time and the energy spent for obtaining the results with significant competitive advantage. For example, intra-day renewable energy prediction will open new market opportunities. Similarly, industrial sites require fast and efficient systems for air-quality monitoring.
- **Dynamic adaptation**: due to the distributed and heterogeneous nature of the data (e.g., traffic data), the combination of code and hardware variants, dynamic autotuning, and virtualization will enable a transparent use of the hardware resources even in case of changes to the configurations.
- **Design productivity**: non-expert programmers will use domain-specific extensions to express the semantics of the application and the security requirements of the data. The EVEREST SDK will automatically carry out the related optimizations, broadening the customers that can be reached with complex heterogeneous platforms.
- **Programmability support**: the EVEREST SDK will hide the platform details to the application, enabling the porting across target platforms with different characteristics.

VII. Concluding Remarks

EVEREST provides a data-driven design framework for extreme-scale Big Data applications on distributed FPGA-based architectures. The EVEREST SDK combines multiple domain-specific languages, compiler optimizations, and HLS to generate multiple code variants that are dynamically selected by matching characteristics of the application and the available hardware. Our major goals are not only to accelerate the application execution, but also to ease the design of complex AI-enabled applications by non-expert programmers, hiding most of the details of the underlaying hardware system.

Acknowledgements

This project has received funding from the EU Horizon 2020 Programme under grant agreement No 957269.

References

[1] S. I. Venieris, A. Kouris, and C.-S. Bouganis, "Toolflows for mapping convolutional neural networks on FPGAs: A survey and future directions," *ACM Comput. Surv.*, vol. 51, no. 3, Jun. 2018.

[2] V. G. Castellana *et al.*, "High level synthesis of RDF queries for graph analytics," in *Proc. of ICCAD*, 2015, p. 323–330.

[3] R. Zhao *et al.*, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *Proc. of FPGA*, 2017, p. 15–24.

[4] K. Kambatla *et al.*, "Trends in big data analytics," *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2561 – 2573, 2014.

[5] M. Minutoli *et al.*, "Efficient synthesis of graph methods: A dynamically scheduled architecture," in *Proc. of ICCAD*, 2016.

[6] C. Jin, V. Gohil, R. Karri, and J. Rajendran, "Security of cloud fpgas: A survey," *arXiv preprint arXiv: 2005.04867*, 2020.

[7] P. Mantovani *et al.*, "Agile SoC development with Open ESP," in *Proc. of ICCAD*, 2020, pp. 1–9.

[8] F. Abel *et al.*, "An fpga platform for hyperscalers," in *Proc. of HOTI*, 2017, pp. 29–32.

[9] R. Nane *et al.*, "A survey and evaluation of FPGA high-level synthesis tools," *IEEE Trans. CAD Integ. Cir. Sys.*, vol. 35, no. 10, Oct. 2016.

[10] V. Cima *et al.*, "Hyperloom: A platform for defining and executing scientific pipelines in distributed environments," in *Proc. of PARMA-DITAM*, 2018, pp. 1–6.

[11] D. Gadioli, E. Vitali, G. Palermo, and C. Silvano, "margot: A dynamic autotuning framework for self-aware approximate computing," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 713–728, 2019.

[12] N. A. Rink *et al.*, "CFDlang: High-level code generation for high-order methods in fluid dynamics," in *Proc. of RWDSL*, 2018, pp. 1–10.

[13] S. Karol *et al.*, "A domain-specific language and editor for parallel particle methods," *ACM Trans. on Mathematical Software*, vol. 44, no. 3, 2018.

[14] A. Susungi *et al.*, "Meta-programming for cross-domain tensor optimizations," in *Proc. of GPCE*, 2018, pp. 79–92.

[15] N. A. Rink and J. Castrillon, "TeIL: a type-safe imperative Tensor Intermediate Language," in *Proc. of ARRAY*, 2019, pp. 57–68.

[16] T. Chen *et al.*, "TVM: An automated end-to-end optimizing compiler for deep learning," in *Proc. of OSDI*, 2018, pp. 578–594.

[17] S. Ertel, A. Goens, J. Adam, and J. Castrillon, "Compiling for concise code and efficient I/O," in *Proc. of CC*, 2018, pp. 104–115.

[18] C. Pilato, K. Wu, S. Garg, R. Karri, and F. Regazzoni, "TaintHLS: High-level synthesis for dynamic information flow tracking," *IEEE Trans. CAD Integ. Cir. Sys.*, vol. 38, no. 5, pp. 798–808, 2019.

[19] V. Sze *et al.*, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, 2017.

[20] S. Ertel *et al.*, "STCLang: State thread composition as a foundation for monadic dataflow parallelism," in *Proc. of Haskell Symposium*, 2019.

[21] C. Lattner *et al.*, "Making Context-Sensitive Points-to Analysis with Heap Cloning Practical For The Real World," in *Proc. of PLDI*, 2007.

[22] C. Lattner *et al.*, "MLIR: A compiler infrastructure for the end of Moore's law," *arXiv preprint arXiv:2002.11054*, 2020.

[23] J. Castrillon and R. Leupers, *Programming Heterogeneous MPSoCs: Tool Flows to Close the Software Productivity Gap*. Springer, 2014.

[24] C/DA - Design Automation, "IEEE 2804-2019 - IEEE standard for software-hardware interface for multi-many-core," Jan. 2020.

[25] Lowe-Power et al., "The gem5 simulator: Version 20.0+," *arXiv preprint arXiv: 2007.03152*, 2020.

[26] C. Menard *et al.*, "System simulation with gem5 and systemc: The keystone for full interoperability," in *Proc. of SAMOS*, 2017, pp. 62–69.

[27] C. Pilato and F. Ferrandi, "Bambu: A modular framework for the high level synthesis of memory-intensive applications," in *Proc. of FPL*, 2013.

[28] Y. Wang, P. Li, and J. Cong, "Theory and algorithm for generalized memory partitioning in high-level synthesis," in *Proc. of FPGA*, 2014.

[29] C. Pilato *et al.*, "System-level optimization of accelerator local memory for heterogeneous systems-on-chip," *IEEE Trans. CAD Integ. Cir. Sys.*, vol. 36, no. 3, p. 435–448, Mar. 2017.

[30] C. Pilato *et al.*, "A runtime adaptive controller for supporting hardware components with variable latency," in *Proc. of AHS*, 2011, pp. 153–160.

[31] C. Pilato *et al.*, "Securing hardware accelerators: A new challenge for high-level synthesis," *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 77–80, 2018.

[32] T. Sechkova, E. Barberis, and M. Paolino, "Cloud & edge trusted virtualized infrastructure manager (vim)-security and trust in openstack," in *Proc. of WCNCW*, 2019, pp. 1–6.

[33] S. Chiotakis, S. Pinneterre, and M. Paolino, "vfpgamanager: A hardware-software framework for optimal fpga resources exploitation in network function virtualization," in *Proc. of EuCNC*, June 2019, pp. 47–51.

[34] E. Paone *et al.*, "Evaluating orthogonality between application auto-tuning and run-time resource management for adaptive OpenCL applications," in *Proc. of ASAP*, 2014, pp. 161–168.

[35] D. Gadioli *et al.*, "SOCRATES — a seamless online compiler and system runtime autotuning framework for energy-aware applications," in *Proc. of DATE*, 2018, pp. 1143–1146.

[36] R. Khasanov and J. Castrillon, "Energy-efficient runtime resource management for adaptable multi-application mapping," in *Proc. of DATE*, Mar. 2020, pp. 909–914.

[37] E. Vitali *et al.*, "An efficient monte carlo-based probabilistic time-dependent routing calculation targeting a server-side car navigation system," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[38] D. Diamantopoulos and C. Hagleitner, "Helmgemm: Managing gpus and fpgas for transprecision gemm workloads in containerized environments," in *Proc. of ASAP)*, vol. 2160-052X, 2019, pp. 71–74.

[39] M. Lagasio *et al.*, "Predictive capability of a high-resolution hydrometeorological forecasting framework coupling wrf cycling 3dvar and continuum," *Journal of Hydrometeorology*, vol. 20, no. 7, 2019.

[40] M. Lagasio *et al.*, "A synergistic use of a high-resolution numerical weather prediction model and high-resolution earth observation products to improve precipitation forecast," *Remote Sensing*, vol. 11, no. 20, 2019.

[41] M. Golasowski *et al.*, "Alternative paths reordering using probabilistic time-dependent routing," *Advances in Intelligent Systems and Computing*, vol. 1036, pp. 235–246, 2020.

[42] V. Ptošek *et al.*, "Real time traffic simulator for self-adaptive navigation system validation," in *Proc. of EMSS*, 2018, pp. 274–283.