Towards efficient HW acceleration in edge-cloud infrastructures: The SERRANO approach ⋆ ⋆⋆

Invited Paper

Aggelos Ferikoglou[1], Ioannis Oroutzoglou[1], Argyris Kokkinis[1], Dimitrios Danopoulos[1], Dimosthenis Masouros[1], Efthymios Chondrogiannis[2], Aitor Fernández Gómez[3], Aristotelis Kretsis[4], Panagiotis Kokkinos[4], Emmanouel Varvarigos[4], and Kostas Siozios[1]

[1] Aristotle University of Thessaloniki, Thessaloniki, Greece
{aferikog, ioroutzo, akokkino, ddanopou, dmasoura, ksiop}@physics.auth.gr
[2] Innovation Acts Ltd., Nicosia, Cyprus
timchros@gmail.com
[3] IDEKO, ICT and AUTOMATION research group, Elgoibar, Basque Country, Spain
afgomez@ideko.es
[4] National Technical University of Athens, Athens, Greece
{akretsis, kokkinop, vmanos}@mail.ntua.gr

Abstract. Nowadays, we witness an ever-increased number of applications deployed over Edge, Cloud and HPC infrastructures. This rapid explosion of computing devices across the computing continuum poses new challenges in terms of providing a power-efficient, secure and automatic way for deployment of different applications in such heterogeneous environments. Moreover, the need for performance efficient deployments within such environments, has introduced the presence of hardware accelerators over the entire computing stack.

In this paper, we present SERRANO's approach for providing efficient HW accelerated deployments over edge-cloud infrastructures. First, we give a brief overview of the SERRANO project, describing its goals and objectives, providing a high-level overview of SERRANO's platform architecture and presenting the use-cases involved. Then, we describe SERRANO's approach for providing efficient HW accelerators by identifying trade-offs between performance, accuracy and power consumption and also demonstrate how SERRANO aims to automate the optimization process through machine learning models in order to construct a generic optimization heuristic to fine-tune programs for both GPU and FPGA accelerators. Through some illustrative examples, we showcase that by applying approximation and optimization techniques, we are able to achieve an average decrease of 28% in power consumption for FPGA

⋆ This work has been supported by the E.C. funded program SERRANO under H2020 Grant Agreement No: 101017168

⋆⋆ Aggelos Ferikoglou, Aitor Fernández Gómez, Argyris Kokkinis, Aristotelis Kretsis, Dimitrios Danopoulos, Dimosthenis Masouros, Efthymios Chondrogiannis, Emmanouel Varvarigos, Ioannis Oroutzoglou, Kostas Siozios, Panagiotis Kokkinos

devices and trade-off between performance and power usage for GPUs, achieving up to ×1.21 speedups and 8% power improvement.

Keywords: Edge Computing · Cloud Continuum · Hardware Accelerators· GPU · FPGA · Heterogeneous .

1 Introduction

In recent years, emerging applications in Machine Learning, security or cloud storage have achieved great success and have been among the most responsible for the high demands in data-center or edge workloads. Different vertical sectors with diverse requirements have gained traction and induced a rise to a number of fundamental challenges that relate to the application deployment, the support of heterogeneous systems and the provided security. According to several analytical agencies, the global market for such cloud services will reach 332 billion USD in 2021 [1]. This is 50 billion USD more than 2020 and 100 billion USD more than 2019. Clouds are used in almost all high-tech industries: in software development, in projects based on IoT, for big data analysis, etc. Many organizations have already moved their workloads to the cloud. According to another research, migrations to the public cloud has grown from 89% to 92% in the last three years and over 80% of firms with more than 1000 workers use multiple cloud platforms, while, by 2024, this percentage is expected to jump up to 90% [2].

Additionally, there is a movement to define an intent-based paradigm of operating federated infrastructures consisting of edge, cloud and HPC resources that will make the process of application deployment automated across the various computing technologies and platforms. The enormous compute needs of such services and applications operating in the cloud requires joining different processing units in a networked-based system such that each task is preferably executed by the unit which is able to efficiently perform it. Heterogeneous computing represents a well established way to achieve further scalability in the computing sector, while, at the same time service assurance mechanisms are required along with a safety-critical and coordinated mechanism to adjust the required tasks.

A solution to overcome these problems and to fully align with the current trends of the cloud computing sector is to introduce various specialized hardware acceleration platforms. Such devices (e.g., GPUs, FPGAs) can achieve higher performance than typical processing systems and also significantly higher performance for the same power envelope [8]. The use of highly specialized units designed for specific workloads can greatly enhance server or edge CPUs and their power budget. Last, a cognitive orchestration of these devices can lead to a single borderless infrastructure which can shrink the existing technology gaps.

In this paper, we present an overview of the SERRANO H2020 project towards Transparent Application Deployment in a Secure, Accelerated and Cognitive Cloud Continuum. SERRANO aims to introduce a novel ecosystem of cloud-based technologies, spanning from specialized hardware resources up to software toolsets, evaluated through three well-defined use cases in cloud storage

services, fintech and manufacturing. We focus on SERRANO's approach for efficient hardware acceleration in the edge-cloud computing continuum, by applying device-specific optimizations and approximation techniques on FPGA and GPU devices. We describe SERRANO's vision to apply such optimizations in an automated manner and leverage them for power-/performance-efficient application deployments within the underlying platform. By employing a set of illustrative examples, we showcase that by applying a set of optimization techniques on accelerated kernels, we are able to trade-off between power and performance efficiency both on GPU and FPGA accelerators, realizing SERRANO's ambition for dynamic, requirement-driven deployments on heterogeneous infrastructures.

2 The SERRANO project

SERRANO[5] aims to deliver a novel ecosystem of hardware- and software-based technologies under the SERRANO platform, contributing to critical cloud related areas, including: *a)* application deployment, *b)* resource interoperability, *c)* privacy and security and *d)* cognitive and autonomous operation. SERRANO's approach is expected to reduce significantly the design and development time, through infrastructure agnostic application development, providing high quality services that utilize efficiently the available resources, thus, boosting the digital productivity. Below, we describe the project's objectives, give an overview of SERRANO's architecture and briefly present the use-cases involved.

2.1 SERRANO Goals & Objectives

In the context of its mission, SERRANO aims to satisfy 6 concrete research objectives, that are going be validated through several individual success indicators (KPIs). Initially, SERRANO **aims to define an intent-driven paradigm of federated infrastructures, with edge, cloud and HPC resources**. An abstraction layer will automate the operation and the full exploitation of the available diverse resources in order to simplify the developer's workload. The intent-driven feature of this layer will enable applications to express their high-level requirements in an infrastructure agnostic manner and translate them to infrastructure-aware configuration parameters.

On a second level, it ambitions to **develop security and privacy mechanisms for accelerated encrypted storage over heterogeneous and federated infrastructures**. SERRANO will ensure GDPR compliant distributed secure storage and data sharing, accessible at low latency with cryptographic primitives and network coding techniques. This essential capability that protects the content itself, complies with privacy implications for personal or confidential information and enables HPC applications to offload their data to edge or cloud, overcoming the local storage restrictions.

SERRANO targets to **provide workload isolation and execution trust on untrusted physical tenders** as well. It desires to deliver a secure, lightweight,

[5] https://ict-serrano.eu/

and efficient framework that embraces interoperable microservices in the cloud, the fog and the edge, providing specific solutions for the low-level software stack.

Moreover, it **focuses on providing acceleration and energy efficiency at both the edge and cloud**. SERRANO aims to introduce advanced "inline" and "online" data sampling in order to decrease the amount of data offloaded from edge/fog to cloud/HPC and therefore will develop a novel "smart sampling" technique based on approximate computing for dynamic data packing of multiple inputs into a single variable.

Furthermore, SERRANO attempts to **develop an hierarchical architecture for end-to-end cognitive orchestration and transparent application over edge/fog and cloud/HPC infrastructures**. To this end, SERRANO will develop intelligent and autonomous orchestration mechanisms that will automatically determine the most appropriate resources to be used. It will exploit multi-objective optimizations, graph theory, AI/ML techniques and heuristics to design an algorithmic toolkit, aiming to provide different trade-offs between optimality and complexity.

Finally, it visions to **demonstrate the capabilities of the secure, disaggregated and accelerated SERRANO platform in supporting highly-demanding, dynamic and safety-critical applications** with 3 use cases (UCs) across different domains with heterogeneous needs. The UCs include secure storage for data protection, secure launching of ultra-large number of fintech processing performing real-time operations, and advanced real-time anomaly detection of manufacturing machines.

2.2 SERRANO's architecture overview

The heterogeneity of distributed edge/cloud environments has revealed the importance of efficient resource orchestration. When multiple applications request different types of resources, meeting the requests in an optimal way becomes a challenging task, particularly when the applications have high resource demands. Heading to more complex infrastructures, the development of intelligent and self-managed orchestration mechanisms is a key factor to optimize resource utilization and meet the application requirements. SERRANO aims to contribute to the aforementioned open issue by proposing a scheme for end-to-end cognitive orchestration together with closed-loop control, based on the principles of observe, decide and act.

SERRANO targets an orchestration system that manages the underlying heterogeneous infrastructure at a more abstract and disaggregated manner compared to the current state-of-the-art solutions. This is achieved through a hierarchical architecture, consisting mainly of three components: a) the central resource orchestrator b) a set of resource-hosted local orchestrators and c) the telemetry framework. Fig. 1 depicts the orchestrator's architecture.

Each application submitted to the SERRANO platform, provides a set of high-level requirements that describe the desired application state. The central resource orchestrator being aware of the underline edge, cloud and HPC resources and the current infrastructure state, decides the optimal placement of

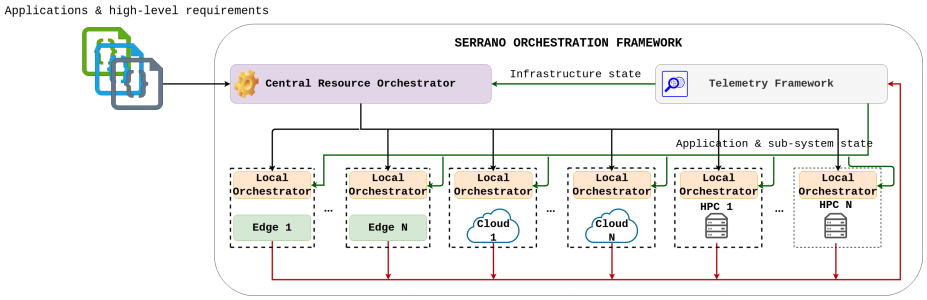


Fig. 1: SERRANO's high-level orchestration architecture

the application. Based on AI/ML algorithms, this mechanism aims to satisfy the user-defined requirements and efficiently use the underline infrastructure. When the optimal placement is decided, the central resource orchestrator assigns the workloads to the selected resources along with the desired performance state and coordinates the required data movement. Then the control is passed to local orchestrators that are responsible for the actual deployment based on the desired performance requirements.

SERRANO resource orchestrator follows a declarative approach, instead of an imperative one, for describing the workload requirements to the local orchestrator. This approach provides several degrees of freedom to the local orchestrator for serving in an optimal manner the request, satisfying both the central orchestrator and the resources' objectives (e.g trade-off accuracy against power for inference tasks that their accuracy constraint is over-satisfied). Finally, it is evident that the service assurance provided by the central and local orchestrators would not be possible without a telemetry framework that captures information concerning the current infrastructure's and applications' state.

2.3 SERRANO's Use-Cases

The performance of the SERRANO platform will be evaluated based on three use-cases from different scientific domains. Each use case consists of challenging case studies that will be deployed and tested on the SERRANO platform, showcasing the platform's capability to address multiple computationally intensive problems, each one with varying requirements.

Use-Case 1: Secure Data Storage The first use-case will be from the field of security and distributed data storage. Its purpose is to demonstrate the platform's capacity for high-performance, yet secure data storage across cloud and edge devices. For this case, SERRANO aims to explore network coding techniques that fragment the encoded data in multiple parts, allowing the encoded pieces to be stored in distributed locations, and thus provide a secure storage solution. For SERRANO, the computing ecosystem consists of data center and edge nodes that will perform high-performance computations for the

encoding and decoding tasks and transfer the data pieces securely across heterogeneous environments without compromising data integrity and privacy. **Use-Case 2: High-Performance Fintech Analysis** The second use-case belongs to the finance sector, and more specifically in the domain of portfolio management and analysis. This use-case will underline the platform's capability for high-performance processing of various AI algorithms for investment management applications in a secure computing environment. Through the SERRANO platform, the automatic management of personalized portfolios and the market prediction mechanisms will be accelerated, scaling-up the overall efficiency by performing precise predictions and increasing the number of managed portfolios. The enhanced security will be fulfilled by the transparent execution of different tasks in multiple devices, allowing processes to switch dynamically among different cloud and edge nodes, and hence decreasing the platform dependency.

Use-Case 3: Machine Anomaly Detection in Manufacturing Environments The third use-case belongs to the machine anomaly detection domain. Specifically, high-frequency sensors generate data that are processed in real-time in order to automatically detect anomalies in machines. However, due to the large volumes of the acquired data, edge devices have limitations for analyzing and detecting the faulty parts. Through this use-case the platform's capacity to analyze large volumes of data and perform high-performance real-time computations will be demonstrated. In the context of SERRANO project cloud devices are planned to be exploited for storing and performing anomaly detection algorithms on the acquired data. Moreover, SERRANO's orchestration mechanisms aim to move data across different platforms in order to achieve real-time anomalies detection through high-performance computations.

3 Efficient acceleration in heterogeneous architectures

SERRANO hardware infrastructure exposes a wide range of acceleration devices for both edge and cloud environments. Moving towards the transparent utilization of heterogeneity, SERRANO is enabled to manage and orchestrate accelerated kernels, to optimally meet their requirements (e.g accuracy, latency). In addition, by providing a source-to-source kernels' transformation mechanism, SERRANO aims to automatically apply device specific acceleration and resource optimizations in order to minimize developers' effort and enable efficient utilization of the underline infrastructure.

3.1 Target hardware infrastructure

Heterogeneous hardware architectures have increased their range of practical applications especially in the cloud and edge domains. SERRANO will introduce a novel deployment model of accelerators both in the cloud and edge sectors which will influence the use of parallel and distributed algorithms. FPGAs and GPUs as hardware platforms will be attached directly in servers or shared over the network in edge workloads. The SERRANO HW infrastructure is expected

to reduce the power and execution times of the assigned tasks by extending the existing development tools and providing novel schemes for scheduling, communication, and synchronisation with the programmable accelerators.

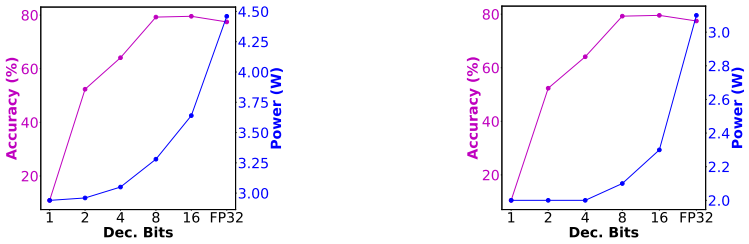
The aim is to virtualize the available hardware accelerators such as Xilinx Alveo FPGAs or Nvidia T4 GPUs in the servers through an efficient device passthrough scheme which will limit the overheads and isolate the devices in each different guest operating system (OS). Overall, a novel VM appliance model for provisioning of data to shared accelerators will be introduced scaling intelligently, automatically and transparently from edge (local cloud) to public cloud.

3.2 Optimization techniques for efficient FPGA acceleration

FPGA devices have been proven to be a promising acceleration alternative when programmed with optimal configuration [7]. Designing hardware for FPGAs can be performed at varying levels of abstraction with the commonly used being the register-transfer level (RTL) and the algorithmic level. These methodologies differ; RTL (i.e. VHDL, Verilog) is used for circuit-level programming while algorithmic level methodologies such as High Level Synthesis (HLS) are used for describing designs in a more abstract and user-friendly way [5].

Device specific HW optimizations: SERRANO will leverage HLS tools in order to provide accelerated kernels for the computationally intensive tasks of the use cases. Employing HLS for FPGA design, enables a faster and more flexible development process compared to RTL. By adding different directives on a C/C++ or OpenCL code, users are able to instruct the HLS compiler to synthesize kernels. In particular, the kernels will be designed by using the Xilinx Vitis framework [3] which provides a unified OpenCL interface for programming edge (e.g MPSoC ZCU104) and cloud (e.g Alveo U200) Xilinx FPGAs. In this manner, the kernel designing process is simplified and thus more effort can be put to the design space exploration (DSE) phase which targets the performance optimization with respect to the architecture and resources of the available FPGAs. It is evident that the kernel acceleration process is device specific, meaning that different HLS pragmas should be applied to the same application when it is targetted to different FPGAs (e.g in loops with multiple iterations, different unrolling factors should be applied for a U200 and a ZCU104 FPGA, due to the different available resources).

Approximate computing techniques for FPGAs: In order to efficiently use the FPGA resources of the SERRANO infrastructure and minimize the power consumption, approximate computing techniques (ACT) will be performed on the accelerated applications. ACTs are used in computationally complex, error-resilient applications, trading-off algorithmic performance with power consumption and resource utilization. Over the years, various ACTs were presented in the literature, targeting different layers of the computing stack [18]. SERRANO mainly focuses on software based ACTs, such as precision scaling, loop perforation and approximate memoization, as they can be easily applied to kernels.



(a) Alveo U200 (b) MPSoC ZCU104

Fig. 2: Accuracy and power consumption vs decimal bits for Gaussian Naive Bayes algorithm in U200 (a) and ZCU104 (b) platforms

Precision scaling [11, 9] is a technique that changes the bit-width of input or intermediate operands to reduce storage and computing requirements. Xilinx [4] provides tools and libraries that support a wide range of fixed-point precision data types, enabling the creation of power and resource efficient designs. Fig. 2a and 2b show the relation between accuracy and power for different bit width approximations, for the Gaussian Naive Bayes algorithm on U200 and ZCU104 platforms. As the figures depict, for 8 bit approximation an average decrease of 28% in power consumption can be achieved without sacrificing accuracy. Loop perforation [12, 16] is another ACT that selectively skips entire loop iterations to reduce computational overhead and hence resource and power consumption. Finally, approximate memoization [19, 17] stores the results of expensive function calls for later use and returns the cached values when similar inputs reoccur. By replacing functions that are complex to implement on hardware with a simple look-up table, memoization leads to significant resource and power savings.

3.3 Optimization techniques for efficient GPU acceleration

In order to provide acceleration and energy efficiency at both the edge and the cloud as visioned, SERRANO makes use of GPU accelerators in order to meet the desired requirements. The project aims not only to accelerate the provided use cases, but to study and apply several parallel kernels' optimizations in order to tune them in terms of both performance and power efficiency. Therefore, to achieve applications' close-to-peak efficiency, several optimization techniques are applied.

Approximate computing techniques for GPUs: SERRANO aims to apply approximation techniques in order to further improve its applications' efficiency. Such as the FPGA case, precision scaling will be applied by executing the kernels on after-Volta architecture featured Tensor Cores [15], instead of CUDA cores, in order to enable mixed-precision computing and provide efficient implementations. Driven primarily by the need for training in deep learning, Volta, Turing and Ampere GPU architectures provide specific programmable

matrix multiply–accumulate units able to deliver a theoretical peak performance of 110 teraFLOP/s in FP16-TC. More specifically, the V100 GPU (used for the SERRANO purposes) features 8 Tensor Cores per streaming processor for a total of 640 Tensor Cores where each Tensor Core can compute D=A×B+C per clock cycle. A and B must be in FP16, but C and D can be in either FP16 or FP32. The multiplication occurs in FP16 and is accumulated in FP32 with other products and therefore is able to further accelerate operations using the powerful tool of low-precision floating-point arithmetic.

Furthermore, SERRANO manages to trade accuracy for latency and power efficiency by perforating thread iterations. Such as in the loop perforating technique in loops of serial programs, launching GPU kernels with fewer threads and processing a part of the output result instead of the whole, promises significant gains in both performance and power consumption by loosing accuracy.

Kernel's block coarsening transformation: Block coarsening is an optimization for parallel applications such as GPU programs. It refers to the kernel transformation that merges together the workload of 2 or more thread blocks and therefore reduces their total number by leaving the number of threads per block the same. Consequently, it merges multiple neighboring blocks in order to deal with the problems associated with extensive fine-grained parallelism.

Kernel's thread coarsening transformation: In contrast with block, thread coarsening transformation fuses together 2 or more neighboring threads within the same block and can be proved beneficial on several parallel programs and architectures. In that case, it reduces the number of threads per block, while leaving the number of launched blocks the same. It is able to increase the amount of work performed by each kernel by replicating the instructions in its body, while it reduces the number of instantiated threads at the runtime.

Insights about thread VS block coarsening optimizations: Similar but with different impacts, both the block and thread coarsening transformations are able to improve efficiency of various parallel applications on different architectures. While they both reduce the total number of each kernel's threads, they distribute them in a different way across GPU's resources and thereupon affect kernel's execution at runtime differently. Listing 1.1 depicts a simple squared kernel written in CUDA, while Listings 1.2 and 1.3 demonstrate how it is transformed after block and thread coarsening transformation accordingly. Integer bc and tc parameters, named as BCF and TCF for the rest of the work, constitute an effective design parameter and refer to the number of blocks and threads merged together accordingly.

From an architectural perspective, GPUs map blocks to SMs (multiprocessors) and threads grouped in warps to CUDA cores. Therefore, reducing blocks per kernel with block coarsening, reduces the number of occupied SMs, while thread coarsening reduces the number of threads per block and therefore the number of warps scheduled by each SM, while the number of SMs occupied by the kernel remains the same. Fig. 3 depicts a simplified architectural view of mapping between software and hardware resources when a kernel is launched after block and thread coarsening transformation.

```
__global__ void square (float *in, float *out){
    int gid=blockIdx.x*blockDim.x+threadIdx.x
  out[gid]=in[gid]*in[gid]; }
```

Listing 1.1: Original squared CUDA kernel

```
__global__ void square_bc (float *in, float *out, int bc){
    int gid=(blockIdx.x*bc)*blockDim.x+(threadIdx.x*bc)
    for (int index=0; index<bc; index++)
      out[gid+index]=in[gid+index]*in[gid+index]; }
```

Listing 1.2: Block coarsened squared kernel

```
__global__ void square_tc (float *in, float *out, int tc){
    int gid=blockIdx.x*(blockDim.x*tc)+(threadIdx.x*tc)
    for (int index=0; index<tc; index++)
      out[gid+index]=in[gid+index]*in[gid+index]; }
```

Listing 1.3: Thread coarsened squared kernels

Even though both transformations attempt to improve parallel applications' efficiency, their impact seems to highly depend on exogenous factors such as device architectures, type of kernels, input size and data types. In order to highlight the difference between these transformations and their impacts' variation through different input sizes, we apply both optimizations on syr2k, a linear algebra CUDA kernel hosted in Polybench GPU Suite [10]. Polybench initially launches syr2k kernel with 2 dimensional thread blocks with total 1024 threads each (`blockDim.x`=32, `blockDim.y`=32) while we apply the transformations for 1, 2, 4, 8 and 16 factors for block and thread coarsening. The final transformed kernel is evaluated for 2 different input sizes (512×512 and 1024×1024 fl. point input matrices) on a 128-core NVIDIA Maxwell GPU featured in Jetson Nano Development Kit and their experimental results are presented in Fig. 4.

It is clear, that from Fig. 4, we observe trade-offs between performance and power consumption that vary through both transformations and input sizes. More specifically, for 512×512 input matrices, all the block coarsening factors seem to improve performance and degrade power efficiency compared with the initial kernel, while thread coarsening presents a more complicated behaviour with 2, 4 and 8 TCFs to provide the most efficient solution from the performance perspective with big losses in power consumption. On the other hand, for 1024×1024 input matrices, 2, 4 and 8 BCFs seem again to boost performance by burdening the power efficiency, while thread coarsening degrades performance and improve the power for big TCFs. Finally, we clearly discover that for small and big input matrices, both of the optimization techniques deliver an improvement in performance with $\times 1.212$, $\times 1.216$, $\times 1.044$ and $\times 1.033$ speedups on figures 4a, 4b, 4c and 4d respectively. For the case of power, only thread coarsening transformation (for TCF=16) delivers an optimal solution, reducing by 110 mW the initial kernel's power usage.

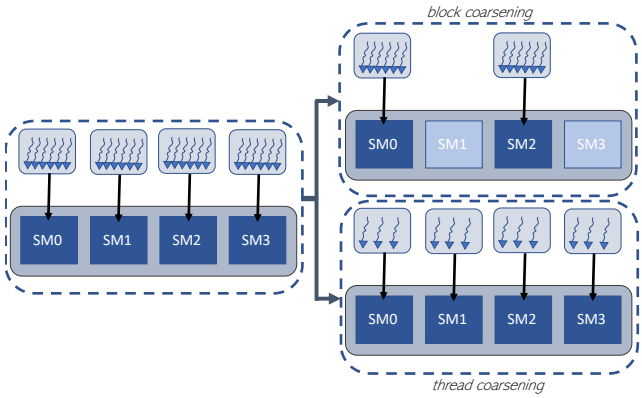
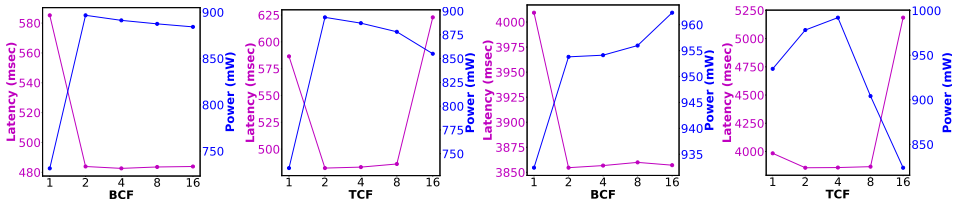


Fig. 3: Block and thread coarsening transformations from hardware perspective



(a) 512x512 syr2k (b) 512x512 syr2k (c) 1Kx1K syr2k (d) 1Kx1K syr2k

Fig. 4: Block (a,c) and thread (b,d) coarsening's latency and power trade-offs for syr2k kernel for various input sizes on 128-core NVIDIA Maxwell GPU

3.4 Automatic optimization heuristics

Employing fine-tune implementations on both GPU and FPGA accelerators enables performance and power efficient executions, meeting the SERRANO's objectives set in subsection 2.1. However, studying optimization problems and building hand-written heuristics constitutes a time consuming and demanding task, that often leads to suboptimal solutions. Selecting and tuning manually appropriate features is a difficult task even for human experts, mainly because of the large design decisions space and its variation through different hardware devices. Hence, accurate automatic optimization heuristics are necessary for dealing with the complexity and diversity of modern hardware and software, in order to avoid time loss and inaccurate code transformations.

SERRANO aims to simplify this optimization process by building automatic machine learning heuristics able to predict optimal decisions based on representative features of unseen programs. There are several works constructing automatic heuristics for optimization of either serial or parallel programs. Wang et al. [20] use machine learning techniques to optimize applications for multi-cores, while Magni et al. [14] were the first to predict the optimal thread coarsening factor for OpenCL kernels, based on code's static features. Finally, based on their

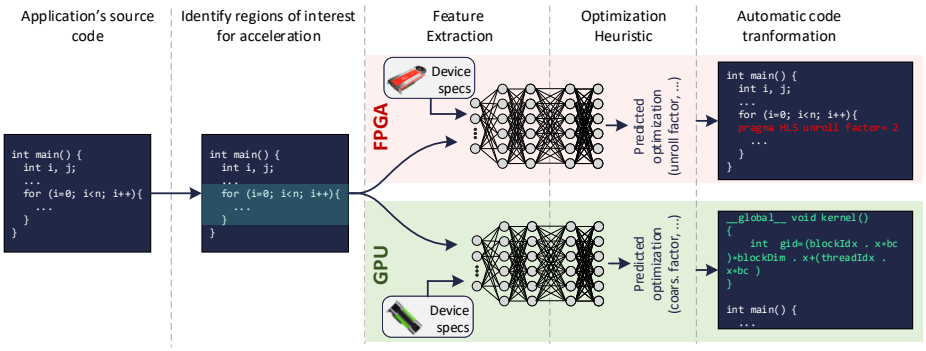


Fig. 5: Generic view of SERRANO's automatic optimization heuristic

works, Cummins et al. [6] were the first to make use of Deep learning models to automatically extract kernel's representative features to be fed to their predictive model for optimal decisions of parallel optimization problems. However, to our knowledge, there is no published work to address automatic optimization transformations as universally as SERRANO aims to do.

In contrast to previous works that build heuristics for specific optimization problems and processors, SERRANO ambitions to construct an end-to-end machine learning based heuristic to automate various applications' optimizations for both GPU and FPGA accelerators. Fig. 5 gives a intuitive view of how SERRANO's model will be constructed and its several processing steps. At first, applications' source codes are planned to be fed into the model as training inputs, where an automatic source-to-source compiler is responsible for converting them to accelerator compiler-compatible representations (CUDA and HLS for our case). Afterwards, through feature engineering, the model will extract the representative code's features in order to feed them to the model for the learning process. Finally, new unseen programs are provided to the trained model in order to automatically make its decisions about accelerators' optimizations problems and provide fine-tuned kernels in terms of both time and power requirements.

3.5 Hardware accelerated deployments within SERRANO

The source-to-source transformation will provide a library of kernels for the computationally intensive tasks of each use case, targeting GPUs and/or FPGAs. Furthermore, different versions of the aforementioned kernels will be produced by applying thread/block coarsening and approximate computing techniques for the GPU and the FPGA case respectively. This library will provide several degrees of freedom to the SERRANO orchestration framework for optimally meeting the application requirements and efficiently using the underline infrastructure. This approach differentiates from former state-of-the-art orchestration schemes [13] that specify the exact resources an application requires.

As it was mentioned in section 2.2, each use case application submitted to the SERRANO platform, will provide a set of high level requirements that describe the desired application state. The central orchestrator, based on the current infrastructure state acquired from telemetry, will decide whether the computationally intensive task of the application will be mapped on a particular GPU or FPGA. Then the application and the specifications are forwarded to the local orchestrator of the selected device. Local orchestrator, based on the current device and application state, is able to change between different versions of this kernel (different block coarsening factors, different bit-width approximations etc.) in order to satisfy the application requirements and optimally use the underline resources.

4 Conclusions & Future Work

In this work, we presented an overview of the SERRANO H2020 project, focusing on the efficient deployment of HW accelerated kernels on the edge-cloud computing continuum. SERRANO aims to create an abstraction layer that will fully exploit the available hardware resources and automate/orchestrate their use. The overall hardware acceleration will be performed in a holistic and automated manner overcoming the existing platform barriers stemming from the heterogeneity of computing units. The efficient utilization of accelerators in both edge and cloud applications will significantly improve the performance and power efficiency of the target workloads while at the same time novel transprecision computing mechanisms will be exploited to examine the accuracy versus resource or speed tradeoffs. Finally, as a future work, SERRANO aims to develop new key concepts and approaches to cloud infrastructures that aim to close existing technology gaps and aspires to have strong innovation potential in a wide number of real-world applications from different markets.

References

1. Gartner Forecasts Worldwide Public Cloud End-User Spending in 2021. https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-percent-in-2021, accessed: 2021-10-6
2. The Future of Cloud Computing: Why businesses opt for cloud. https://innovecs.com/blog/future-of-cloud-computing/, accessed: 2021-10-6
3. Vitis platform, https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html
4. Xilinx, https://www.xilinx.com/
5. Cong, J., Liu, B., Neuendorffer, S., Noguera, J., Vissers, K., Zhang, Z.: High-level synthesis for fpgas: From prototyping to deployment. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **30**(4), 473–491 (2011)
6. Cummins, C., Petoumenos, P., Wang, Z., Leather, H.: End-to-end deep learning of optimization heuristics. In: 2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT). pp. 219–232. IEEE (2017)

7. Danopoulos, D., Kachris, C., Soudris, D.: A Quantitative Comparison for Image Recognition on Accelerated Heterogeneous Cloud Infrastructures, pp. 171–189 (09 2019). https://doi.org/10.1201/9780429399602-8
8. Danopoulos, D., Kachris, C., Soudris, D.: Fpga acceleration of approximate knn indexing on high- dimensional vectors. In: 2019 14th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC). pp. 59–65 (2019). https://doi.org/10.1109/ReCoSoC48741.2019.9034938
9. Finnerty, A., Ratigner, H.: Reduce power and cost by converting from floating point to fixed point. WP491 (v1. 0) (2017)
10. Grauer-Gray, S., Xu, L., Searles, R., Ayalasomayajula, S., Cavazos, J.: Auto-tuning a high-level language targeted to gpu codes. In: 2012 innovative parallel computing (InPar). pp. 1–10. Ieee (2012)
11. Ko, J.H., Fromm, J., Philipose, M., Tashev, I., Zarar, S.: Precision scaling of neural networks for efficient audio processing. arXiv preprint arXiv:1712.01340 (2017)
12. Koliogeorgi, K., Zervakis, G., Anagnostos, D., Zompakis, N., Siozios, K.: Optimizing svm classifier through approximate and high level synthesis techniques. In: 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST). pp. 1–4. IEEE (2019)
13. Kouris, A., Venieris, S.I., Bouganis, C.S.: Towards efficient on-board deployment of dnns on intelligent autonomous systems. In: 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). pp. 568–573. IEEE (2019)
14. Magni, A., Dubach, C., O'Boyle, M.: Automatic optimization of thread-coarsening for graphics processors. In: Proceedings of the 23rd international conference on Parallel architectures and compilation. pp. 455–466 (2014)
15. Markidis, S., Der Chien, S.W., Laure, E., Peng, I.B., Vetter, J.S.: Nvidia tensor core programmability, performance & precision. In: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 522–531. IEEE (2018)
16. Sidiroglou-Douskos, S., Misailovic, S., Hoffmann, H., Rinard, M.: Managing performance vs. accuracy trade-offs with loop perforation. In: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. pp. 124–134 (2011)
17. Sinha, S., Zhang, W.: Low-power fpga design using memoization-based approximate computing. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **24**(8), 2665–2678 (2016)
18. Stanley-Marbell, P., Alaghi, A., Carbin, M., Darulova, E., Dolecek, L., Gerstlauer, A., Gillani, G., Jevdjic, D., Moreau, T., Cacciotti, M., et al.: Exploiting errors for efficiency: A survey from circuits to applications. ACM Computing Surveys (CSUR) **53**(3), 1–39 (2020)
19. Tziantzioulis, G., Hardavellas, N., Campanoni, S.: Temporal approximate function memoization. IEEE Micro **38**(4), 60–70 (2018)
20. Wang, Z., O'Boyle, M.F.: Partitioning streaming parallelism for multi-cores: a machine learning based approach. In: Proceedings of the 19th international conference on Parallel architectures and compilation techniques. pp. 307–318 (2010)