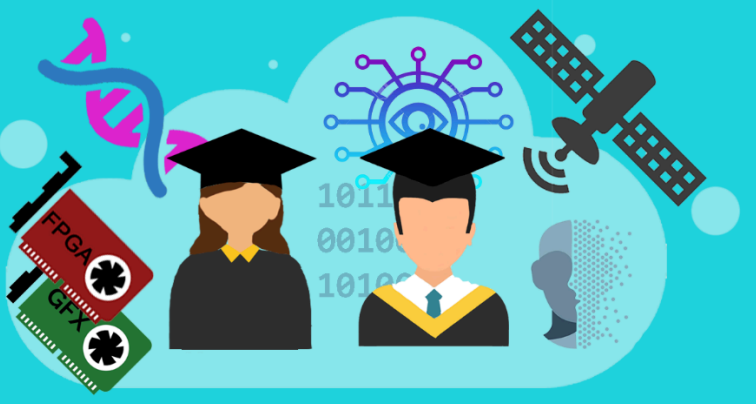
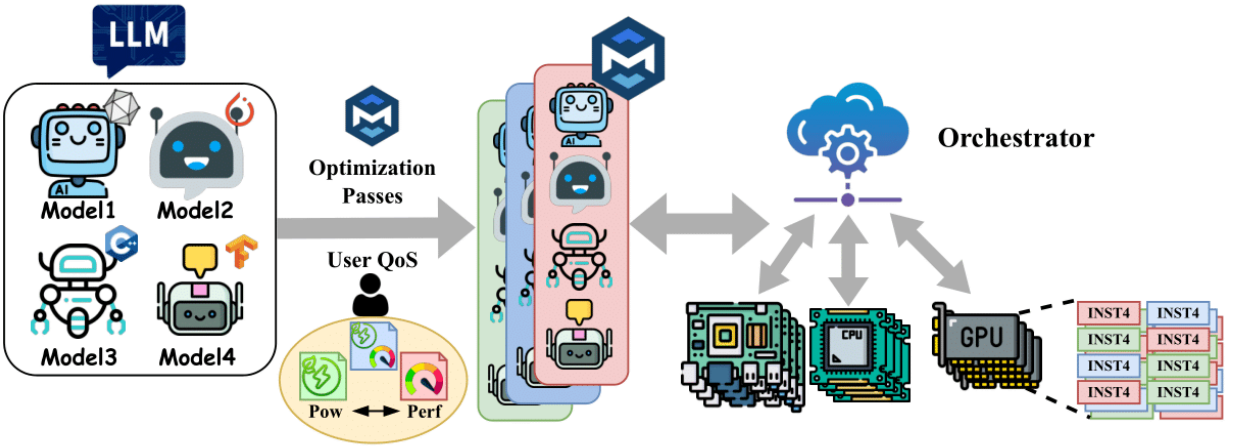
Diploma Thesis

Microproccessors and Digital Systems Laboratory



MLIR-GPU constraint-driven compilation and deployment of Large Language Models(LLMs)

Modern LLM deployments face a widening gap between performance demands and energy limits: Users have diverse Service Level Objective (SLO) requirements, while infrastructure providers must operate under power and resource constraints, a challenge compounded by the heterogeneity of devices. Today's pipelines mostly focus either on exploring the power-performance trade-offs on a single device or for targeting many devices by sacrificing optimization capabilities[1]. Utilizing a SLO-aware solution that starts from workloads described in MLIR dialects and reasons jointly about multiple objectives, e.g., latency and energy, across multiple targets is therefore essential. In this thesis, we will design a runtime framework for deploying LLM workloads under explicit latency-energy SLO constraints across heterogeneous compute targets.



Profiling and MLIR Passes:

Depending on the application, a user may want to aim in different positions of the power-performance space and so, profiling the target LLMs to extract their execution patterns becomes significant. When it comes to enabling multiple targets, MLIR is a compilation stack that supports multilevel optimization and flexibility in choosing seamlessly the target hardware for deployment. Several works have been presented targeting different accelerators (CPUs, GPUs, TPUs etc.) by incorporating different MLIR descriptions (e.g. an LLM model written in PyTorch[2]) of the same applications.

Orchestrator, Scheduling and QoS constraints:

Users will provide the implemented framework with a high-level description of the models, e.g., Pytorch, TensorFlow, ONNX, along with some target SLO, i.e., latency. The framework that will be developed in this thesis, will leverage the generated variants from the previous steps, to make orchestration decisions. Specifically, at runtime, given the underlying set of HW nodes, their resource availability, and the specified constraints, i.e., latency.

Having a high-level description of the models a user wants to run, written in any known high-level description language (PyTorch, TensorFlow, ONNX, etc.), an Orchestrator will make decisions about resource allocation of the underlying HW nodes (node = different device or combination of devices) by deploying different versions of those models, driven by certain power and performance needs. Describing those needs as SLO constraints from the users' side, the Orchestrator may dynamically adapt resource allocations, change the deployed model versions or reconfigure device partitions,e.g., MIG[1] for GPUs. Evaluation will benchmark popular LLM tasks across devices, comparing the current static placement methodologies with the proposed adaptive orchestrator in terms of SLO compliance, and resource efficiency, e.g., energy/power.

RELATED MATERIAL:

[1]Fraser Mince, Dzung Dinh, Jonas Kgomo, Neil Thompson, and Sara Hooker." The grand illusion: the myth of software portability and implications for ML progress". NeurIPS '23

[2] Torch-MLIR: The torch-mlir project aims to provide first class support from the pytorch ecosystem to the mlir ecosystem

[3]Hsin-I Cindy Liu et al., "TinyIREE: An ML Execution Environment for Embedded Systems From Compilation to Deployment". IEEE Micro, https://doi.org/10.1109/MM.2022.3178068

PREREQUISITES:

- C, Bash, AI model Python Libraries (PyTorch, Tensorflow)

CONTACT INFORMATION:

- Georgios Alexandris, Ph.D Student, NTUA: (galexandris@microlab.ntua.gr)
- Panagiotis Chaidos, Ph.D Student, NTUA: (pchaidos@microlab.ntua.gr)
- Achilleas Tzenetopoulos, Ph.D Candidate, NTUA: (atzenetopoulos@microlab.ntua.gr)
- Professor Dimitrios Soudris, NTUA: (dsoudris@microlab.ntua.gr)
- Assistant Professor Sotirios Xydis, NTUA: (sxydis@microlab.ntua.gr)

[1] https://www.nvidia.com/en-eu/technologies/multi-instance-gpu/