Funded by
the European Union

HORIZON EUROPE FRAMEWORK PROGRAMME

CLOUDSTARS

(grant agreement No 101086248)

Cloud Open Source Research Mobility Network

D4.1 AI-based methods for Cloud and Edge environments

Due date of deliverable: 31-12-2024
Actual submission date: 18-12-2024

Start date of project: 01-01-2023 Duration: 48 months

Summary of the document

Document Type	Report
Dissemination level	Public
State	v1.0
Number of pages	23
WP/Task related to this document	WP4 / T4.1, T4.2, T4.3, T4.4, T4.5
WP/Task responsible	Barcelona Supercomputing Center (BSC)
Leader	Josep Lluís Berral (BSC)
Technical Manager	Josep Lluís Berral (BSC)
Quality Manager	Aurora González (UMU), Luís Veiga (INESC-ID)
Author(s)	Josep Lluís Berral, Pol García, Ferran Agulló (BSC); Jana Vatter, Herbert Woisetschläger, Alexander Isenko (TUM); Aurora González, Carlos Hernández, Antonio Martínez, Pablo Fernández (UM); Josef Spillner (ZHAW); Nathanael Nussbaumer, Shashikant Ilager (TUW)
Partner(s) Contributing	BSC, ZHAW, TUM, UM, TUW
Document ID	CLOUDSTARS_D4.1_Public.pdf
Abstract	Report on the researched and developed methods on AI for Cloud/Edge management and configuration, characterization and modeling of infrastructures and applications.
Keywords	Artificial Intelligence, Machine Learning, Cloud Optimization, Containerization, Virtualization, Distributed AI, Federated Learning, Security Layer for AI, LLMs, SLMs, GPU enablers

History of changes

Version	Date	Author	Summary of changes
0.1	01-11-2024	Josep Lluís Berral (BSC)	Structure and TOC.
0.2	29-11-2024	Josep Lluís Berral, Pol García, Ferran Agulló (BSC); Jana Vatter, Herbert Woisetschläger, Alexander Isenko (TUM); Aurora González, Carlos Hernández, Antonio Martínez, Pablo Fernández (UM); Josef Spillner (ZHAW); Nathanael Nussbaumer, Shashikant Ilager (TUW)	First draft.
0.3	16-12-2024	Josep Lluís Berral (BSC); Aurora González (UMU); Luís Veiga (INESC-ID)	Revised draft.
1.0	18-12-2024	Josep Lluís Berral (BSC)	Final version.

Table of Contents

1 Executive summary **1**

2 Secondements Description and Progress **3**
2.1 Barcelona Supercomputing Center - IBM New York 3
2.1.1 Towards the throughput limit in LLM serving 3
2.1.2 Addressing Throughput vs Fairness trade-off in the Serving of LLM Adapters 3
2.2 Zurich University of Applied Sciences - Nearby Computing BCN 5
2.2.1 Observability tools and interfaces for AI-driven orchestration 5
2.2.2 ML-driven orchestration Observability tools and interfaces for AI-driven orchestration 6
2.3 Technical University of Munich - IBM New York 6
2.3.1 Choosing a Classical Planner with Graph Neural Networks 6
2.3.2 Enabling large-scale interactive exploration in data lakes with fast vector-based similarity search 8
2.3.3 Why Utilizing The Maximum Amount Of Memory (Almost) Never Leads To A Better Training Throughput 8
2.4 University of Murcia - IBM New York & IBM Ireland 9
2.4.1 On Serving Image Classification Models 9
2.4.2 Optimizing Cybersecurity Models for Edge Devices 12
2.4.3 On Confidential Computing Scheduling for LLMs 12
2.4.4 LLM-based security policy processing automation using Retrieval Augmented Generation 14
2.5 Technical University of Wien - IBM New York 16
2.5.1 Feasibility of Probabilistic Programming for LLMs in Ansible 16
2.5.2 Energy Efficient LLM Inference through Dynamic Prunning 17

3 Secondments in Progress **18**

4 Summary of Contributions **19**
4.1 Task 4.1 - Optimizing LLM and Model Serving through GPU Accelerators 19
4.2 Task 4.2 - Resource Planning for GNN processes 19
4.3 Task 4.3 - Resources Trade-off for LLMs 19
4.4 Task 4.4 - Security for Models and LLMs in Edge-IoT 19
4.5 Task 4.5 - AI-Driven Orchestration 19
4.6 Tasks and Secondees Relation 20

5 Conclusions **21**

1 Executive summary

In this deliverable, we provide the scientific summary of the different tasks and advances in Cloud-Stars, Work Package 4, related to Cloud systems for AI and Data Analytics. This includes those tasks focusing on the researched and developed methods of AI for Cloud/Edge management and configuration, characterization, and modeling of infrastructures and applications. Specifically, the general objectives for the current tasks are:

- AI and Machine Learning (ML) for optimizing Cloud systems: advance on ML techniques, like statistical and deep learning, for managing containerized Cloud systems.
- Optimization and security on Big Data systems: advance in techniques and frameworks for providing a security layer on platforms for data processing and serverless systems.
- Distribution on AI and Data Analytics models: advance in federation of AI techniques (e.g. deep learning and data streams) on data processing platforms on the Edge.

Current secondees are Barcelona Supercomputing Center (BSC), Universidad de Murcia (UMU), Zurich University of Applied Sciences (ZHAW), Technical University of Munich (TUM) and Technical University of Viena (TUW); visiting as hosts IBM T.J.Watson Laboratories in Yorktown, New York (YKT), IBM Research in Zurich (ZRL), IBM Ireland (IRL) in substitution of IBM Israel, and Nearby Computing Barcelona (NBC). During the first period (M1 - M24), 13 visits were made, representing 46.5 secondment months (out of 118). Such secondments have produced collaborations and research on topics such as 1) Scheduling and orchestration of AI workloads for GPU utilization, also AI-based management systems; 2) Service of models for image inference and impact of input size in GPU-based systems; 3) AI planning towards leveraging Graph Neural Networks and XGBoost for training processes. From the current visits, five tasks have received contributions from secondments at IBM New York, IBM Ireland, and NearbyComputing:

- Barcelona Supercomputing Center (at IBM New York)
  - Performance and energy efficiency of SMLs in resource-constrained environments, i.e. GPUs partitioning. (T4.1, T4.3, T4.4; BSC at IBM New York)
  - Scheduling strategies for LLM adapters in resources, towards fairness/performance trade-off. (T4.1, T4.3, T4.4; BSC at IBM New York)
- Zurich University of Applied Sciences (at NearbyComputing Barcelona)
  - Observability tools and interfaces for AI-driven orchestration. (T4.5; ZHAW at NBC Barcelona)
  - ML-driven orchestration Observability tools and interfaces for AI-driven orchestration. (T4.5; ZHAW at NBC Barcelona)
- University of Murcia (at IBM New York and IBM Ireland)
  - Impact on the performance of input size on single GPU inference model serving. (T4.1, T4.4; UMU at IBM New York)
  - Study of GPU performance for image classification and time series predictive algorithms. (T4.4; UMU at IBM New York)
  - Unsupervised FL optimization methods and aggregation. (T4.4; UMU at IBM New York.)
  - Optimization Techniques Implementation in FL, methodologies for sensor data and RT Anomaly mechanisms. (T4.1, T4.4; UMU at IBM Ireland)
- University of Munich (at IBM New York)
  - AI Planning: Online Planner Selection with GNNs + XGBoost towards efficient training processes. (T4.2; TUM at IBM New York)

  - Deep Learning Models Training Across Clouds and Continents. (T4.3; TUM at IBM New York)
  - Benchmarking Federated Machine Learning Applications in Edge Computing Systems. (T4.4; TUM at IBM New York)

- University of Viena (at IBM New York)
  - Feasibility of Probabilistic Programming for LLMs in Ansible. (T4.1; TUW at IBM New York)
  - Energy Efficient LLM Inference through Dynamic Pruning. (T4.1; TUW at IBM New York)

Finally, there are 3 secondments in progress at this time, regarding UMU at IBM Ireland, and 2 more secondments in preparation starting January 2025. In the next Deliverable 4.2, expected for Month 48, the continuation of secondments will continue for BSC, UMU and TUM, covering Tasks between 4.1 and 4.5.

2 Secondements Description and Progress

2.1 Barcelona Supercomputing Center - IBM New York

2.1.1 Towards the throughput limit in LLM serving

Visitor:	Pol Garcia Recasens	Involved tasks:	T4.1, T4.3, T4.4
Partner:	BSC	Secondment host:	IBM New York. NY, USA

Introduction

Recent advancements in Large Language Models (LLMs) have revolutionized natural language processing (NLP), setting new performance benchmarks across various tasks. However, serving these models remains resource-intensive, often requiring distributed systems to manage their memory and computational demands. The emergence of Small Language Models (SLMs) provides a pathway for resource-constrained scenarios, enabling high-performance NLP tasks with fewer resources. Despite their smaller size, SLMs encounter limitations in serving throughput due to the sequential nature of autoregressive decoding and low arithmetic intensity. Our work investigates how large batching can achieve Pareto-optimal throughput on single accelerators, and how model replication can optimize resource utilization for SLM inference.

Contribution

During the stay, we focused on evaluating the performance of Small Language Models (SLMs) in resource-constrained deployment scenarios. While large models like GPT-3 require significant resources, we explored smaller models capable of offering competitive performance with reduced computational costs. To do so, we benchmarked models ranging from 125 million to 13 billion parameters on a cluster of 4 NVIDIA A100 GPUs, studying optimization techniques such as dynamic and continuous batching for combining the inference of multiple requests, and PagedAttention to reduce memory fragmentation on the storage of intermediate results in the GPU. We generate requests from the ShareGPT dataset, a collection of real conversations with ChatGPT. The prompt of each request is composed by 512 input tokens, and we limit the generation to 256 output tokens. Therefore, each request requires storing KV pairs of up to 768 tokens in the KV cache, divided in blocks of 16 tokens allocated on-demand. This allowed us to restrict the experimental setting and study the theoretical bound where batching no longer improves performance. Once we find the optimal batch size, we limit the memory allocation for that instance, and run multiple instances on the same GPU. We show how this improves GPU utilization and the use of computational resources.

Results

Our analysis showed that for small models a single high-end accelerator has enough memory to reach a Pareto-optimal throughput frontier given a large batch of requests. Beyond that point, allocating more memory results in minimal or no improvements. In light of our results, we pave the way for new optimizations in model serving, presenting an initial set of findings that show how model replication on a single device improves overall inference performance. Further analysis should consider a more realistic serving scenario with heterogeneous requests and devices, and explore model replication with more suitable techniques.

Future visits/Collaborations

Currently, the collaboration is extended (in remote), expanding the experiments towards a novel publication during 2025, and the possibility of next visits even after CloudStars.

2.1.2 Addressing Throughput vs Fairness trade-off in the Serving of LLM Adapters

Visitor:	Ferran Agulló López	Involved tasks:	T4.1, T4.3, T4.4
Partner:	BSC	Secondment host:	IBM New York. NY, USA

Introduction

In line with the growth and widespread of Large Language Models (LLMs), we can observe a spread in the demand for LLM adapters. While LLM are large-scale models trained to have a high proficiency in a wide range of Natural Language Processing (NLP) tasks, adapters are small-scale additions that adjust the general knowledge of LLMs to a more concrete and specific task. The use of adapters is faster than training from scratch or fine-tuning a new model, which is a tedious and long process that requires meticulous data curation and high computational capabilities, and yields higher performance than methods such as In-Context Learning (ICL) [1] or prompt tuning [2, 3, 4, 5] when the task at hand is different enough from the data used to train the LLM.

In this secondment, we have focused on scaling the serving of LLM adapters taking into account both throughput and fairness. Despite the fact that adapters are smaller than their base model, their size is not negligible, and all systems face a hardware limit that caps the total number of adapters that can be served simultaneously. In this way, more adapters can be requested simultaneously than the ones that can be served. The system scheduler needs to decide which adapters to serve first, in what order, and when to leave space for other adapters. If it is not managed properly, it can provoke a collapse in the throughput of the system or some serious starvation issues in the final users who will receive an unfair service by the system. If adapters are constantly switched over, little batching will be possible, and loading times from storage may slow down the system, causing a decrease in the final throughput. On the other hand, if adapters swap very infrequently, users that send requests to waiting adapters will experience delayed responses, receiving an unfair service compared to other users.

Contribution

This balance between fairness and throughput was the principal focus of the work. The problem to be solved, could be summarized in the following question: "In which priority, order and sequence the requests for LLM adapters should be served to achieve a determined and acceptable trade-off between throughput and fairness when more adapters are requested than the ones that can be provided simultaneously?" Such "determined" refers to the ability to change the ratio of the two factors using an input parameter at the start of the execution. We define "acceptable" as the capacity to increase system fairness without significantly reducing throughput, or vice versa.

In order to tackle this problem, we proposed a novel scheduling algorithm that is able to balance performance and fairness when scaling the number of served LLM adapters over the number that can be provided simultaneously. This algorithm provides fine-grained control over the above trade-off through an input parameter, that will determine how the algorithm provides the priority, order and sequence in which the requests of the system are meant to be served. The algorithm is inspired on classical scheduling theory, and more especially, on the *polling system model* [6, 7, 8, 9, 10, 11]. We, as well, leverage the work from Sheng et al. (VTC) in order to make our algorithm take decisions based on fairness.

Results

At the current moment, we are finishing the experiments and preparing the research publication for the beginning of the coming year. We are comparing our findings to the default versions of vLLM [12] and S-LoRA [13]. They are two frameworks developed as a result of two research publications about minimizing memory fragmentation, and they have become two notorious frameworks in the overall serving research community.

The principal result is that the proposed scheduling algorithm is able to achieve the same throughput of these two frameworks when serving LLM adapters with a clear improvement in the fairness among the users of the system. In addition, if fairness among users needs to be prioritized over throughput, the scheduling algorithm can be easily configured to increase the fairness to a desired level at the cost of a corresponding decrease in the throughput of the system.

Future visits/Collaborations

During the experimentation phase of this work, we bumped into the Pareto-optimal throughput plateau studied by Pol Garcia Recasens in a previous secondment. As a result, we are currently working doing an expansion of his previous work towards a novel publication next year.

2.2 Zurich University of Applied Sciences - Nearby Computing BCN

2.2.1 Observability tools and interfaces for AI-driven orchestration

Visitor:	Ranjan Ojha	Involved tasks:	T4.5
Participant:	ZHAW	Secondment host:	NearbyComputing. BCN, ES

Introduction

Kubernetes as an omnipresent infrastructure to host cloud workloads has good techniques to adjust the resource provisioning to the workload needs through static scaling and autoscaling. However, autoscaling is known to be an emergent area with an unsatisfactory degree of automation. In this secondment, building upon previous research work conducted at ZHAW and the application use case presented by the NearbyOne platform, the goal was to explore more intelligent scaling within Kubernetes environments. Kubernetes offers a variety of autoscaling options, categorized as pre-emptive (predictive) or reactive (responsive). Additionally, these algorithms can be classified as cluster autoscalers (adding/removing nodes) or workload autoscalers (adjusting pod replicas). This diversity creates challenges for deployment personnel, especially in Infrastructure as a Service (IaaS) contexts. IaaS providers lack visibility into deployed applications yet must adhere to service level agreements (SLAs), often leading to over-provisioning. Furthermore, the growing range of Kubernetes cluster types, including geographically distributed edge clusters, adds complexity. Considering these factors and the potential for multi-cluster algorithms, a vast number of permutations require testing. Our hypothesis has been that a new tool could address this gap by facilitating highly reproducible experimentation that maintains realistic application behavior. This enables the evaluation of various hypotheses to identify optimal autoscaling configurations.

Contribution

This secondment contributed to a realistic workload simulation based on user activity simulation. Such simulations can be used to validate and refine predictive autoscaling strategies. The initial investigation into web server load simulation revealed two primary estimation approaches. The first approach focuses on user behavior modeling. This estimation considers a multitude of factors to predict realistic traffic patterns, simulating scenarios like peak user activity. Factors influencing this model could include user demographics, access times, and interaction patterns. The second approach emphasizes application behavior prediction. Here, the estimation aims to capture how the application itself responds to user input. This approach considers factors like resource usage patterns based on different functionalities within the application. By combining these two estimation segments, we created a comprehensive model of the overall cluster load experienced by the web server. The model was realised with two existing open source tools: stress-ng, with a wrapper to incorporate Kubernetes pods, and k6s, as well as helper tools like Grafana to validate the results visually. The resulting system is reusable in other Kubernetes-based cloud autoscaling contexts.

Results

To create realistic user behavior patterns, we obtained anonymized container usage data from Google Cloud for a month. While the data exhibited variations in sampling frequency, we downsampled it by daily averaging to create a per-day usage rate for the entire month. Subsequently, we normalized the data using min-max normalization. Our focus wasn't on replicating specific application usage but rather on user behavior patterns, based on the assumption that increased user requests would translate to higher resource utilization. The results show a root mean square error (RMSE) of not more than 0.22 across ten pods, confirming a highly accurate prediction of workload given a known user request pattern. Hence, our experiments demonstrate the effectiveness of our tool in approximating real-world application behavior. This capability makes it suitable for training various algorithms by providing a reproducible black-box testing environment that simulates potential application scenarios. The successful replication of actual application behavior from a Google dataset using our methodology strengthens this claim. Our results dataset has been published under the name 'Simulated application load on a Kubernetes system based on Human traffic pattern' at Zenodo: https://zenodo.org/records/11210559

Future visits/Collaborations

This has been the first out of several planned secondments of research staff from ZHAW to NearbyComputing. It defined the topic stream of the future secondments, the second of which also took place later in 2024 and is documented below.

2.2.2 ML-driven orchestration Observability tools and interfaces for AI-driven orchestration

Visitor:	Sepideh Shamsizadeh	Involved tasks:	T4.5
Participant:	ZHAW	Secondment host:	NearbyComputing. BCN, ES

Introduction

Autoscaling and orchestration of cloud workloads across sites/clusters remains a challenge. While the previous secondment has contributed to workload estimation and scaling decisions, it is also important to keep the user (engineer deploying the application) in the loop and give hints on scaling and other configuration settings based on the user intent. The secondment therefore focuses on integrating Large Language Models (LLMs) into the NearbyOne orchestration platform to enhance decision-making for cloud-edge platform orchestration. This involves designing an LLM-powered decision engine to optimize energy efficiency and reduce latency in resource allocation and workload distribution. The project addresses challenges such as collecting platform metrics and translating high-level user intents into intelligent orchestration actions. At the time of reporting, two out of three secondment months have passed. Hence, the following description describes work in progress.

Contribution

Significant progress has already been made during this secondment. Scripts have been developed to simulate Prometheus and Thanos for monitoring and metrics collection, alongside a tailored script to retrieve marketplace-related data from the NearbyOne platform. These efforts support the design of an LLM-based decision-making engine that translates high-level user intents into actionable commands for the platform and Kubernetes, such as deploying, deleting, migrating, or auto-scaling. Additionally, a preliminary demo has been conceptualized to showcase the integration of LLMs within the orchestration platform. Here, the open research question is the selection of language models ranging from domain-specific vocabulary to general-purpose trained models which are known to be suboptimal from an energy efficiency point of view due to high resource consumption during training.

Results

Initial experiments using simulated Prometheus and Thanos environments demonstrate that the LLM-powered decision engine effectively interprets and analyzes system metrics. Closing the loop from out-of-bounds metrics to missing or incorrect configuration directives in the NearbyOne Kubernetes configuration appears to be possible. Results highlight the feasibility of incorporating also marketplace data into the decision-making process, enabling smarter orchestration of preconfigured applications. Early tests suggest optimal orchestration decisions, with ongoing work focused on validating scalability and further improving relevant metrics under real-world conditions.

Future visits/Collaborations

As the second secondment from ZHAW to NearbyComputing, it consolidated the previous work and led to a more results-oriented exchange. Moreover, it helped define the subsequent (third) secondment, which will take place in the second project phase in early 2025, and therefore intensifies the exchange around intelligent cloud-edge infrastructures and how to make this intelligence useful for distributed applications deployment.

2.3 Technical University of Munich - IBM New York

2.3.1 Choosing a Classical Planner with Graph Neural Networks

Visitor:	Jana Vatter	Involved tasks:	T4.2
Partner:	TUM	Secondment host:	IBM New York. NY, USA

Introduction

Automated planning is a major area of AI. The main objective is to realize sequences of actions which are then typically executed by autonomous systems, self-driving vehicles, or intelligent agents. For each planning problem, a planner can be chosen. As numerous planners have been developed which adress different aspects, planning is a complex task. Due to the fact that no single planner is optimal for all domains, portfolio-based approaches have emerged. Here, multiple planners are aggregated into a portfolio. Out of this portfolio, the optimal planner for each individual task is selected. The choice of optimal planner is a difficult task and can be tackled by using machine learning approaches. One of these approaches includes Graph Neural Networks (GNNs) where the graphs represent individual planning problems. Like this, we can learn the optimal planner for a given planning problem. The goal is to enhance a large-scale planning dataset to improve the performance of the planner selection task. In addition, different ways to pick a planner are explored and we use Extreme Gradient Boosting (XGBoost) in combination with GNNs for a resource-efficient approach.

Contribution

Our work explores the planning portfolio, the corresponding planning graphs, four GNN architectures, different node features, and various ways to choose a planner. In more detail, we investigate the strengths and weaknesses of the GNN architectures and analyze how they impact the predictions. As features, we focus on the node degree and neighbor type. The first way how we choose a planner is by predicting the probability that a planner solves the task, the second way is to predict the time it takes each planner to solve the task. To select a planner in a resource-efficient, but effective way, we use graph representations obtained by GNNs as input to other ML methods such as XGBoost.

Dataset and features

The International Planning Competition (IPC) dataset we are using contains tasks from various domains and the performance of the 17 planners in the portfolio. There are 2,439 planning graphs in the dataset, 2,294 graphs for training and validation and 145 graphs for testing. Splits for cross-validation are also provided, a random split and a domain-preserving split. The domain-preserving split groups tasks of the same domain and ensures they are not split up. As node feature, the node type is used indicating the node represents a constant, an action, or an effect. The labels are tensors with 17 entries, each entry representing one planner of the portfolio. The value of the entries is the time needed for the planner to solve the task. In case the time is above the threshold of 1,800 seconds, it is set to 10,000 seconds.

GNN training

As GNN architectures, we use the Graph Convolutional Network (GCN), Gated Graph Neural Network (GGNN), Graph Attention Network (GAT), and Graph Isomorphism Network (GIN). GCN is commonly used in a variety of tasks and is inspired by image convolution. GGNN not only focuses on local information but also on long-range relations within the graph. GAT uses an attention mechanism that supports the detection of local features. GIN is based on the Weisfeiler-Lehman graph isomorphism test which indicates the similarity of two graphs making it especially powerful for graph-level predictions.

Results

Overall, the accuracy when predicting the time is higher than predicting the probability. GIN and GAT prove to be especially powerful with an accuracy of around 0.77 (time-based task). When enhancing the node features with the in- and out-degree, the accuracy of GCN rises up to 0.87 (time-based task). Another line of experiments explores combining GNNs with XGBoost. Here, we can perform a third task: multiclass classification to directly pick one of the 17 planners. Our results show that the classification task now performs comparable to the time-based task and we get accuracy values of slightly under 0.8 for GCN and GAT. It should be noted that the GNN plus XGBoost approach is more resource-efficient than using a GNN only. The GNN experiments need to be trained on a GPU while the combination with XGBoost can be run on a CPU in the same amount of time.

Future visits/Collaborations

We are continuing to collaborate and working on publishing the work at an AI-related conference.

2.3.2 Enabling large-scale interactive exploration in data lakes with fast vector-based similarity search

Visitor:	Herbert Woisetschläger	Involved tasks:	T4.4
Participant:	TUM	Secondment host:	IBM New York. NY, USA

Introduction

As the amount of unstructured data within organizations increases rapidly, we need to find ways of creating meaningful data representations in an unsupervised or semi-supervised fashion. For this, efficient vector operations at scale are a critical success factor. This entails data summarization systems capable of handling several hundred thousand requests with minimal processing latencies, which will be discussed in detail. In this secondment work, we presented a system based on TorchServe, FAISS, and Milvus for generating vector embeddings, similarity search, and clustering over high-dimensional vectors, along with its performance evaluation and challenges.

Contribution

Principal work has focused on scaling data summarization for exploring massive-scale unstructured data lakes as part of WP 4.4. IBM Research has been developing a product line to enable unsupervised exploration of massive amounts of data, including but not limited to images, videos, text, and audio. The main problem with the implementation at the time was its high processing latency for data clustering and its inability to run similarity search over high dimensional data representations (vectors). As such, the task at hand was to create a new pipeline that allows real-time computing over millions of data points in interactive user sessions to identify sample categories that a model has problems predicting.

Results

The solution to the problem at hand was to create a modular pipeline that is capable of running in a hybrid cloud setup. By decoupling the embedding generator (TorchServe) from the database (Milvus) and the analytics suite (FAISS), we enable distributed use cases and maximize throughput. Our core results show that the new design accelerates the data summarization pipeline by up to 7.1x and brings processing latencies below 50ms for a data batch, i.e., enables real-time computing for interactive user sessions.

Future visits/collaborations

Such collaboration will continue through future students and researchers at TUM.

2.3.3 Why Utilizing The Maximum Amount Of Memory (Almost) Never Leads To A Better Training Throughput

Visitor:	Alex Isenko	Involved tasks:	T4.3
Participant:	TUM	Secondment host:	IBM New York. NY, USA

Introduction

We worked on an emerging strategy to train large deep learning models on scalable infrastructures known as Fully Sharded Data Parallel (FSDP). This strategy covers a portion of task T4.3, which is focused on runtime estimation and cluster consumption for deep learning workloads by exploring the stability of common training algorithms used in state-of-the-art data centers. FSDP uses a model-sharding approach where model parameters are split across multiple devices. However, the actual job runtime and VRAM consumption are not always consistent with current theoretical models.

Contribution

During his stay, Alexander profiled state-of-the-art models to help estimate training runtimes and memory consumption. They produced a summary of the insights gathered by profiling FSDP intricately, which resulted in finding non-deterministic memory allocation behavior by PyTorch and providing some solutions to alleviate the problem. Additionally, they created a testbench for profiling FSDP, focusing on VRAM consumption and analyzing the training of state-of-the-art models. Finally, they published a blog post in the CloudStars dissemination blog that provides a full write-up of their work.

Results

The detailed results have been made public as a blog post (Click here). A summary of such post is as follows:

The experiments revealed several critical insights regarding GPU memory management and its influence on training throughput:

- Throughput vs. Minibatch Size: Training throughput increased steadily with minibatch size until reaching a critical threshold, beyond which performance degraded. For the T5-3B model, this threshold was identified at a minibatch size of 64. Beyond this size, at minibatch 69, throughput decreased by 12% due to memory management inefficiencies despite the model staying under the OOM threshold.

- Memory Fragmentation: Memory fragmentation emerged as a significant factor affecting throughput. While PyTorch's CUDA Caching Allocator (CCA) tries to optimize memory reuse, it can lead to increased fragmentation in scenarios with fluctuating memory demands. This was evident in minibatch sizes close to the memory limit, where memory fragmentation resulted in degraded performance.

- CUDA Allocation Retries: The experiments highlighted a high frequency of CUDA allocation retries at larger minibatch sizes, particularly with FSDP. These retries caused significant synchronization overhead, reducing throughput. For example, minibatch size 69 exhibited retries at nearly every training step, while minibatch size 64 showed reduced retries but still highlighted room for optimization.

- Expandable Segments: The introduction of the expandable_segments parameter helped mitigate some issues by stabilizing memory allocation. However, this came with trade-offs, including a small but consistent throughput penalty, especially noticeable at smaller minibatch sizes.

- Model-Specific Behavior: Interestingly, not all models exhibited the same allocation patterns under FSDP. For instance, the GPT2-XL model (1.6B) showed more stable memory behavior, even at high minibatch sizes. This discrepancy suggests that model architecture influences memory allocation dynamics, warranting further investigation.

- Ad Hoc Solutions: While expandable_segments provided a partial fix, other CCA tuning parameters either increased memory usage or introduced errors, highlighting the need for more robust solutions. Alternative memory management approaches, like ColossalAI's Gemini, showed promise in early trials but require integration into mainstream frameworks like PyTorch.

The findings emphasize the importance of memory management in achieving optimal training throughput, particularly when using strategies like FSDP for large-scale models. While current solutions address some issues, the persistent challenge of memory fragmentation and allocation retries underscores the need for further research and development in this domain. Future work could focus on integrating custom memory managers or refining existing tools to enhance the scalability and efficiency of deep learning training pipelines.

Future visits/collaborations

Such collaboration will continue through future students and researchers at TUM.

2.4 University of Murcia - IBM New York & IBM Ireland

2.4.1 On Serving Image Classification Models

Visitor:	Aurora González Vidal	Involved tasks:	T4.1, T4.4
Participant:	UM	Secondment host:	IBM New York. NY, USA

Introduction

Effective model inference has become essential in numerous interactive applications [14, 15, 16]. In deep learning, inference accounts for up to 90% of the infrastructure costs associated with developing and running ML applications, highlighting the need for high-performance, cost-effective inference infrastructure[1].

Examples of image processing applications include E-commerce and retail (Amazon and Pinterest use image inference for product search, discovery, and recommendation), social media (Instagram for suggesting filters and detecting offensive content), autonomous vehicles to detect objects, pedestrians, and lane markings, enabling safe navigation, healthcare [17], precision agriculture[18] etc.

Model serving systems must be scalable, ensuring high throughput and efficient resource utilization across compute units. This work aims to establish foundations for model inference in serverless environments, focusing on dynamic resource allocation based on inference load and deadline requirements. We will examine the fundamental factors for developing a generalizable optimization model to aid in scheduling with deadline guarantees, optimizing available resource usage.

Contribution

Our work investigates inference time for image classification to propose a preliminary mathematical model that optimizes computing resources in soft and relaxed inference. We conducted experiments using a single GPU and the EfficientNet model [19], a benchmark in computer vision known for its efficient scaling of network depth, width, and resolution. EfficientNet's scaling coefficient, ϕ, creates network variants from B0 to B7, with deeper and wider architectures supporting higher resolutions. For our study, we used EfficientNet-B0 which was pre-trained on the ImageNet dataset and we will measure memory usage and perform general system profiling.

Measuring memory usage

The *torch.cuda.memory_allocated()* function is a method provided by the PyTorch library for Python, and it is used to determine the amount of GPU memory currently allocated by your PyTorch tensors and variables. This function specifically reports the amount of memory allocated in bytes on the GPU device you are currently using for your PyTorch operations. In deep learning, the data that is processed by neural networks is often stored in tensors (multi-dimensional arrays) on the GPU. When creating those tensors and performing operations on them, PyTorch allocates GPU memory to store these tensors. This memory allocation is necessary for computation. In order to monitor GPU memory during the inference process, we have computed the prior and posterior allocated memory.

General system profiling

GPU warmup was originally proposed in 2017 [20]. Besides the GPU being in a power-saving state, there can be a number of other reasons why the first launch of a kernel could be slower than further runs because the GPU starts from a cold state. GPUs have startup overhead [21], including memory allocation and driver initialization. Accounting for these delays is fundamental when a deadline needs to be achieved: just-in-time compilation, transfer of kernel to GPU memory, and cache content are some things to be taken into account.

All those parameters can be profiled. For such a purpose we have used the single file implementation of a hardware monitor from one of the authors [22]. Such a script includes 164 features obtained making use of *psutil* that enables tracking of network bandwidth, disk read/write bandwidth, disk read/write counters, context switches, average CPU load (1,5,15 min), memory utilization and process memory (resident, virtual, lib, etc.). And if a GPU is available, over *pynvml* and *torch*: framebuffer memory, bar1 memory, gpu/memory utilization, temperature, power, throttle reasons, statistics retrieved by *torch.cuda.memory_stats()*, average (small, large, all) segment size, average block size, average inactive block size, allocation rounding overhead, memory fragmentation.

Results

The experiments were conducted on a system running the Debian 6.1.0-10-amd64 Linux distribution with kernel version 6.1.38-1. The hardware platform was x86_64 and the system was equipped with an NVIDIA A100 GPU with 40 GB of VRAM, and computations were performed using CUDA version 11.0.

[1] `https://aws.amazon.com/ec2/instance-types/inf1/`

Having studied experimentally the batch input size influence on memory and time and the mini-batch size influence on time and other parameters we were able to formulate two mathematical models for the optimization of image inference.

In the soft real-time inference use-case, our goal is to minimize the number of GPUs that are needed to deliver a response within a certain time considering again the constraints of available GPUs, warm-up times, and linear batch size-latency relationships. The decision variables are as follows:

- t_i: The number of times GPU$_i$ is used (an integer).
- mbs_i: The mini-batch size chosen for GPU GPU$_i$ (an integer).
- N_G: The number of GPUs to be used (an integer)

The constants:

- T: The total available time. This should not be exceeded by any of the GPUs, given that they work in parallel (a decimal number).
- N: The number of images that need to be processed in total in the given time (an integer).
- $NGPU$: The maximum number of GPUs available (an integer)
- M_i: The maximum number of times GPU$_i$ can be used (a constant)
- $Size_i$: The images' input size for GPU$_i$

The functions:

- L_i: Latency per mbs$_i$ for GPU$_i$
- W_i: Warm-up time for GPU$_i$
- MB_i: The maximum mini-batch size for GPU$_i$ (a function of $Size_i$).

Then, the optimization problem can be formulated as follows:

$$\begin{aligned} \min \quad & N_G \\ \text{s.t.} \quad & \text{Maximum}_i(W_i(\text{mbs}_i) + t_i \cdot L_i(\text{mbs}_i)) \leq T \\ & \sum_i (t_i + 1) \cdot \text{mbs}_i \geq N \\ & 1 \leq \text{mbs}_i \leq MB_i \quad \text{for all } i \\ & 0 \leq t_i \leq M_i \quad \text{for all } i \\ & 1 \leq N_G \leq NGPU \end{aligned} \tag{1}$$

In a relaxed inference use case, our goal is to maximize the number of images processed in a given time while considering the constraints of available GPUs, warm-up times, and linear batch size-latency relationships. The definitions with regard to decision variables, constraints, constants and functions are almost the same as previously, except that we do not need to minimize the number of GPUs that are used in this scenario. We are considering that a number of GPUs are reserved for this task and we are interested in the operation of the system in order to maximize the throughput. The optimization problem can be formulated as follows:

$$\begin{aligned} \max \quad & NGPU \times \sum_i (t_i + 1) \cdot \text{mbs}_i \\ \text{s.t.} \quad & \text{Maximum}_i(W_i(\text{mbs}_i) + t_i \cdot L_i(\text{mbs}_i)) \leq T \\ & 1 \leq \text{mbs}_i \leq MB_i \quad \forall i \\ & 0 \leq t_i \leq M_i \quad \forall i \end{aligned} \tag{2}$$

Solving this optimization problem will give us information about how to use the available GPUs optimally, meaning with what mini-batch size and how many times, according to our limits.

For more in-depth results, I refer the reader to the published conference paper [23].

Future visits/collaborations

We have continued this collaboration in time with other secondments, looking for inference optimization of different processes in Large Language Models.

2.4.2 Optimizing Cybersecurity Models for Edge Devices

Visitor:	Carlos Hernández Hidalgo	Involved tasks:	T4.1, T4.4
Participant:	UM	Secondment host:	IBM Ireland. Dublin, IR

Introduction

As edge devices become increasingly integral to cybersecurity systems, optimizing machine learning models for real-time intrusion and anomaly detection in resource-constrained environments has become a pressing challenge. The secondment's primary goal was to develop and enhance machine learning models capable of performing efficiently on devices with limited computational resources. This work specifically focused on deploying advanced optimization techniques to achieve a balance between accuracy, model compactness, and deployment feasibility in edge scenarios.

Contribution

The contributions were the implementation of optimization techniques for machine learning, with a particular emphasis on Quantization-Aware Training (QAT) and TinyML. These methods were essential to enabling the deployment of models on devices with limited computational power while maintaining performance. Using the CICIDS2017 dataset as a benchmark, I developed and evaluated models for real-time anomaly detection.

I began by leveraging AutoGluon for automated model selection and optimization, identifying suitable architectures that were then replicated in PyTorch to incorporate QAT. This process required substantial model re-engineering to address quantization constraints. Strategic adjustments, such as learning rate tuning and early stopping, ensured training stability. Additionally, I implemented the INTELLECT methodology using real sensor data to demonstrate its effectiveness in edge-based anomaly detection scenarios.

Results

The project achieved significant advancements in optimizing machine learning models for edge environments. The quantized models developed during the secondment exhibited competitive performance, achieving substantial size reductions with minimal accuracy loss. This balance between compactness and accuracy highlights the feasibility of deploying such models in real-world edge applications, paving the way for efficient anomaly detection in resource-constrained settings.

Future visits/Collaborations

The secondment fostered a strong collaborative relationship with IBM Ireland, opening avenues for future work. We are planning a follow-up internship next summer (2025) in order to continue our collaboration. Planned efforts include refining QAT implementations, expanding the scope of datasets for anomaly detection, and integrating advanced notification mechanisms into real-world edge devices. These initiatives aim to bridge the gap between theoretical advancements in machine learning and their practical deployment in resource-limited environments.

2.4.3 On Confidential Computing Scheduling for LLMs

Visitor:	Antonio Martínez Ibarra	Involved tasks:	T4.1
Participant:	UM	Secondment host:	IBM New York. NY, USA

Introduction

Confidential Computing addresses the need to protect data in use through computation within hardware-based Trusted Execution Environments (TEEs) [24]. These secure environments ensure data integrity, confidentiality, and code integrity by preventing unauthorized access or modification

[25]. Confidential Virtual Machines (CVMs) operate within attested TEEs, shielding the VM's code and data from the hypervisor, host OS, other VMs, and the TEE's hosting environment [26]. AI extends the need for protection to models, weights, algorithms, and outcomes [27]. Large Language Models (LLMs), designed for complex natural language tasks, depend on sensitive data and are computationally intensive [28], making them ideal for deployment in confidential environments to safeguard data and ensure model integrity. Executing LLM inference in confidential environments poses challenges due to security-related computational overhead, highlighting the importance of efficient workload scheduling to meet performance and Service Level Agreements (SLAs). Optimizing scheduling and provisioning strategies for LLM workloads in CVMs is critical for secure and efficient deployment. This work explores tradeoffs between latency, SLA attainment, throughput, and GPU utilization in confidential and non-confidential settings.

Contribution

Experiments are set up to evaluate inference performance by generating test data for batches, studying model loading and unloading times, profiling batch performance for each model, and designing scheduling techniques. We measure system performance through metrics such as throughput, latency, SLA attainment, and GPU usage under different input patterns, scheduling strategies, and SLA limits, in both confidential and non-confidential settings.

Each experiment runs for 20 minutes, during which requests are sent to models based on predefined distributions and input rates. These requests are handled using specific scheduling strategies and tested against multiple SLA thresholds (40, 60, and 80 seconds). Identical experiments are conducted in both confidential and non-confidential modes. The experimental setup includes a Python script simulating traffic patterns and rates, a Flask API to batch and process inference requests using large language models (LLMs), and a Bash script iterating through combinations of SLAs, traffic means, and scheduling strategies. Outputs include CSV files with detailed request logs, including timestamps, model usage, batch size, latency, throughput, and CPU/GPU utilization, as well as logs capturing model switches, runtime, and other relevant details.

We explore three traffic patterns: Gamma, representing random arrivals with inter-arrival times following a Gamma distribution; Bursty, simulating alternating high-activity and idle periods; and Ramp, which gradually increases traffic to a peak before tapering off. All traffic patterns are tested with different input rates to ensure comparable average request arrival rates.

To meet SLA constraints and optimize throughput, scheduling strategies incorporate various logic components. The "Best Batch" component maximizes throughput by waiting for full batch sizes, while the "Timer" component ensures SLA compliance by processing incomplete batches if necessary to avoid delays. "The Partial Batch" component processes smaller batches before switching models, and "Select Batch" component adjusts batch size dynamically based on traffic rates and SLA limits. These strategies are tested individually and in combination to evaluate their effectiveness under different conditions. The strategies tested as combinations of those components and their specific goals ordered by increasing complexity are (i) "Best Batch" to set a baseline; (ii) "Best Batch + Timer" to meet SLAs while maintaining a reasonable throughput; (iii) "Select Batch + Timer" to meet SLAs better and (iv) "Best Batch + Partial Batch + Timer" to meet SLAs and achieve higher throughput.

The experiments are constrained by the availability of a single CVM with one GPU, capable of loading only one model at a time. We use three models from Huggingface—Granite (granite-7b-base), Gemma (gemma-7b), and Llama (Meta-Llama-3.1-8B)—to study their performance across various configurations. The experiments were conducted on a virtual machine with an NVIDIA H100 80GB HBM3 with driver version 550.54.14, CUDA Version 12.4, PyTorch 2.4.0+cu121 and Python 3.10.12. The system was running Ubuntu 22.04-amd64 Linux distribution.

Results

Latency is measured as the time taken since a request is sent from the user until it is dispatched by the server after performing inference. Non-CC is 20-30% lower than in CC mode, and that conduct is shared across all the three input distributions, being the bursty pattern yields worse results than gamma and ramp distributions. The same applies to higher SLAs.

SLA attainment is measured as the percentage of requests that get a response in less time than the SLA limit. For SLA 40 completion rates are lower because there is not enough time to handle the workload and for SLA 60 and 80 completion rates improve, as expected. Since SLA attainment is inversely proportional to latency, the bursty pattern has the lowest percentage of attainment. The scheduling method that works best for these metrics is the "SelectBatch+Timer" as intended. The reason behind that is that while other strategies aim for higher batch sizes and therefore wait more before processing, this one usually deals with lower batch sizes, thus a lower overall waiting time before processing. As for confidential vs non-confidential settings: with SLA 40 completion rates are 50% vs 70%, with SLA 60 CC vs Non-CC stands at 70% vs 85%, and for SLA 80 both modes achieve above 90% for CC and non-CC, respectively.

Overall throughput is the throughput measured as total requests processed divided by total runtime. The three strategies that contain the "BestBatch" logic achieve a similar throughput, much higher than the one with "SelectBatch". Overall throughput is similar for every pattern, with bursty being slightly lower. Throughput for the non-confidential setting is between 45 and 70% higher than for CC mode, even though the processing rate during inference, that is the number of requests processed divided by the time since the models start to process the batch until it has finished, is similar across all input rate means, patterns and strategies. Consequently, the additional overhead of loading times from swapping models translates into that difference in throughput.

GPU usage only accounts for GPU usage exclusively for inference, since we are interested in utilizing GPU the maximum time possible for that purpose, and it is measured as the percentage of total runtime that the models are performing inference. Results show that the GPU usage is around 50% higher for Non-CC than for CC, and in both cases that utilization percentage is lower than 50%. The same occurs for the other SLAs. Then, the question regarding where the remaining time is may arise. That time is most likely in loading the models into the GPU and to a much lower extent, in unloading models and the scheduling process itself.

The number of model swaps was also recorded and we found that the swap count is similar for both CC and Non-CC, which means that each switch for CC is much more expensive in terms of time than for Non-CC.

To summarize, the main findings for the CC and Non-CC comparison are:

- Latency is higher and SLA attainment is lower in the CC setting, mainly due to model loading times.
- Overall throughput and GPU usage is lower for CC, again due to model loading times (even though model switches are the same).
- Is it possible to estimate CC setting values knowing how the Non-CC setting performs? It would be possible to make a reasonable estimation, but a clear relationship needs more work to tell with certainty.

The repository of this project is publicly available at `https://github.com/Antonio-MI/sincere`

Future visits/collaborations

We are working on publishing the results of the secondment in a AI-related conference and open to explore further into the topic of scheduling for confidential computing.

2.4.4 LLM-based security policy processing automation using Retrieval Augmented Generation

Visitor:	Pablo Fernández Saura	Involved tasks:	T4.4
Participant:	UM	Secondment host:	IBM New York. NY, USA

Introduction

The secondment focused on developing a Retrieval Augmented Generation (RAG)-based framework for processing Windows systems security mitigation policies written in the STIXv2 format. Security mitigation policies often include high-level guidelines that require precise and actionable steps for effective implementation. To bridge this gap, the RAG approach integrates a vector database and a sequence of large language model (LLM) queries to enhance the translation of abstract policies into executable tasks and corresponding Win32 API calls. The RAG methodology is particularly advantageous in this context, as it provides a mechanism to retrieve relevant contextual knowledge dynamically, overcoming limitations of standalone LLMs in domain-specific tasks. The project also sought to evaluate the performance of this framework compared to a traditional, non-RAG-based approach. The emphasis was on reducing irrelevant API calls, improving precision, and enhancing recall to ensure that all essential API calls for the given policies are correctly identified. This work highlights the critical role of RAG in enabling intelligent retrieval systems that augment LLM capabilities and demonstrates its application in a real-world use case with significant implications for Windows system security.

Contribution

This secondment delivered several key contributions, advancing both the theoretical and practical aspects of RAG-based frameworks in security contexts:

- *Pipeline Design:* A novel pipeline was developed to process high-level security policies, transforming them into actionable tasks and API calls. This included detailed workflows for prompt engineering and integrating LLMs with retrieval systems.
- *Comprehensive API Call Repository:* Over 2600 Win32 API call descriptions were scraped, processed, and embedded into a vector database, forming a critical knowledge base for the RAG framework.
- *Evaluation Metrics and Results:* Extensive experiments demonstrated the superiority of the RAG-driven approach in precision, recall, and F1-score, validating the hypothesis that combining retrieval-based context with LLMs can significantly improve task-to-API translation.
- *Open-Source Contributions:* The project led to the creation of reusable tools and scripts for future extensions of this work and potentially to be used for other related projects.

These contributions not only address the immediate use case of Windows security policy translation but also provide a template for adapting similar methodologies to other domains.

Methodology

The methodology employed in this work reflects a structured approach to leveraging RAG for security applications:

1. *Policy Selection and Ground Truth Creation:* A subset of mitigation policies for Windows systems was chosen from a MITRE GitHub repository, focusing on those translatable into API calls. Tasks derived from these policies were generated using a structured LLM prompt, and ground-truth API calls were manually curated for validation purposes.
2. *API Repository Construction:* A Python web scraper, utilizing the *bs4* library, retrieved detailed descriptions of over 2600 Win32 API calls. Each API call was stored as a text document containing its name and a brief description, forming the basis for the vector database.
3. *Vector Database Population:* Documents were processed using LangChain functionalities, including document loaders, text splitters, and embedding models. The all-mpnet-base-v2 model from HuggingFace was employed to generate embeddings, which were stored in a Chroma vector database.
4. *Prompt Engineering for LLMs:* Two distinct prompts were crafted: the first to convert policies into tasks and the second to generate API calls from task descriptions. These prompts included detailed instructions to guide the LLM outputs, ensuring consistency and relevance.
5. *RAG Pipeline Implementation:* The pipeline incorporated a retriever to fetch the most relevant API descriptions for each task. These retrieved descriptions, combined with the task details, were fed into an LLM to produce the final set of API calls.

6. *Experimental Validation:* The RAG pipeline was tested on 14 tasks derived from six policies, with performance compared against a non-RAG baseline using metrics such as precision, recall, and F1-score.

This systematic approach ensured robust testing of the RAG framework while maintaining reproducibility and scalability.

Results

The experimental results underscore the advantages of the RAG-driven approach:

- *Precision:* The RAG framework reduced irrelevant API calls in outputs, achieving a precision score significantly higher than the non-RAG baseline.
- *Recall:* By leveraging a vector database for contextual retrieval, the framework ensured all relevant API calls were included, improving recall values.
- *Efficiency:* The integration of embeddings and vector retrieval systems streamlined the translation process, reducing overall computation time while enhancing output quality.

A detailed analysis revealed that tasks involving highly specialized policies benefited most from the RAG approach, where retrieval of domain-specific knowledge played a critical role in improving outcomes. These findings validate the initial hypothesis and demonstrate the potential of RAG for practical applications in system security.

Future visits/collaborations

We expect and encourage future visitors to continue this work, working on extending the ground truth, enhancing the experimentation process, adapting the framework to other environments (e.g., Linux-based systems security policies), etc.

2.5 Technical University of Wien - IBM New York

2.5.1 Feasibility of Probabilistic Programming for LLMs in Ansible

Visitor:	Nathanael Nussbaumer	Involved tasks:	T4.1
Participant:	TUW	Secondment host:	IBM New York. NY, USA

Introduction

The primary goal of this secondment was to investigate the feasibility of generating synthetic values for options in Ansible programs using probabilistic programming. These synthetic values would be used in templated Ansible programs and be integrated into real world datasets of Ansible Programs. These would serve as valuable resources for training Large Language Models (LLMs) on the Ansible programming language. As of now, our research on this problem is promising but not conclusive and TUW will further collaborate with IBM on this subject.

Contribution

Ansible is a programming language designed for software provisioning and application deployment. As such it is often used in cloud orchestration and managing cloud infrastructure. Due to endless possibilities, Ansible is a complex programming language and creating an Ansible program is not always straightforward. With the recent rise of Large Language Models for coding (Github Copilot, AWS CodeWhisprere, ChatGPT, etc...), writing code has become more accessible than ever before. While these models often work well with popular programming languages (Python, Java, JavaScript), their performance is often lacking with less used programming languages, due to the smaller size of training data. Creating a LLM that works well with Ansible would help improve the language's accessibility and reduce time spent on creating Ansible programs. Unfortunately, as with other less-used programming languages, only relying on publicly available data for training an LLM on Ansible is not sufficient and other approaches need to be found for improving model performance. One such approach is generating synthetic data to increase the size of our training set.

Ansible, as a language, is structured in Modules and every Module has Options. For some popular Modules in Ansible, the distributions of Option values are well known (many examples exist) while others lack examples. To assist in creating a synthetic dataset of Ansible programs and further develop an LLM for Ansible code generation it is key that good values are generated. This is why we narrowed the project scope down to generating synthetic values for options, which can be used in Ansible templates.

Results

- Building an end-to-end model pipeline for generating synthetic values for Ansible Options.
- Evaluating multiple Topic Modeling algorithms on parts of our problem.
- Developing a novel Topic Modeling strategy.

These results allow us to keep an open research collaboration with each other and continue working on this problem together. It is important to us, to put the time we spend in the secondment to good use, which is also why we are not rushing to publish our findings and developed code just yet, but rather take our time to complete the evaluation properly.

Future visits/Collaborations

IBM and TUW will further collaborate on this topic and work together towards an open-source release of our developed code in the next 2-3 months. Furthermore, we will add new algorithms to our model pipeline for comparison and plan on finalizing the evaluation of all algorithms in the same time frame. Finally, we will develop different probabilistic models to make use of the learned topic distributions and compare their results with each other. Depending on the novelty of our findings we are planning on publishing our results afterwards.

2.5.2 Energy Efficient LLM Inference through Dynamic Prunning

Visitor:	Shashikant Ilager	Involved tasks:	T4.1
Participant:	TUW	Secondment host:	IBM New York. NY, USA

Introduction The secondment focused on energy-efficient LLM inference. I collaborated with the Hybrid Cloud Infrastructure team, with Eun Kyung Lee as my manager. I also had frequent meetings with Yue Zhu and Chen Wang. LLM inference platforms are becoming highly resource-intensive due to the high number of user requests and the diverse models available for tasks in the backend infrastructure. This necessitates developing efficient inference mechanisms that manage trade-offs between accuracy, throughput, and resource consumption.

Contribution and Results We conducted a detailed literature review and performed a state-of-the-art analysis. Specifically, we focused on efficient inference for code generation. We addressed the code generation problem using dynamic early exit as a pruning method to manage the trade-off between accuracy, latency, and energy consumption. The problem was modeled as a dynamic learning agent with a Markov Decision Process. We solved this problem using a reinforcement learning agent, implemented with LLama 3B and OPT-2.7B parameters. Our initial experiments suggest that it is possible to significantly reduce the energy cost of code generation tasks by developing an adaptive learning agent for early exit. This work will contribute to developing efficient LLM inference platforms.

3 Secondments in Progress

At the moment of writing this document, there are three secondments to be completed:

- Diego Sánchez García (T4.1; UMU to IBM Ireland), from September to December 2024.
- Jose Manuel Bernabé Murcia (T4.1; UMU to IBM Ireland), from September to December 2024.
- Eduardo Cánovas (T4.1; UMU to IBM Ireland), from September to December 2024.

They are expected to return in mid-December, and their contributions and future collaborations will be reported in the next deliverable D4.2.

Additionally, there are secondments in preparation, to start January 2025, with the process of travel, visa, and work plan in progress:

- Pol García Recasens (T4.5; BSC to IBM Ireland), from January to July 2025.
- Joan Oliveras Torra (T4.1; BSC to IBM New York), from February to June 2025.

Such secondments and continuous collaboration will also be reported in the next deliverable D4.2.

4 Summary of Contributions

4.1 Task 4.1 - Optimizing LLM and Model Serving through GPU Accelerators

In this task, the dedicated work explores advancements in efficient model serving, spanning various domains including large and small language models (LLMs), image classification, and confidential computing. The first contribution investigates throughput **optimization for Small Language Models (SLMs)**, demonstrating that on single accelerators, Pareto-optimal throughput can be achieved via large batching and model replication. Also, in image classification serving, mathematical **optimization models were developed to balance GPU usage** with throughput and deadlines. And lastly, the **energy-efficient LLM inference work** employs dynamic pruning using reinforcement learning for code generation tasks, achieving energy savings without substantial accuracy loss. Collectively, these contributions address critical challenges in model serving, offering practical optimizations to balance resource efficiency, scalability, and security.

4.2 Task 4.2 - Resource Planning for GNN processes

The research done in this task focuses on **improving automated planning** by selecting optimal planners for specific tasks using machine learning, particularly **Graph Neural Networks (GNNs)**. The research utilized a portfolio of 17 planners and a dataset from the International Planning Competition (IPC) to train and validate their models. By leveraging four GNN architectures (GCN, GGNN, GAT, and GIN) and combining GNNs with Extreme Gradient Boosting (XGBoost), the study achieved significant advances in predicting planner performance. This hybrid approach proved to be more resource-efficient while maintaining high accuracy, opening pathways for scalable planner selection solutions.

4.3 Task 4.3 - Resources Trade-off for LLMs

The research done in this task focuses on developing a novel scheduling algorithm to address the **trade-off between throughput and fairness when serving requests for LLM adapters**. LLM adapters, though smaller than their base models, can face hardware limits when requested in high volumes. The scheduling algorithm provides fine-grained control over prioritizing throughput or fairness through a configurable input parameter. Initial results demonstrate the algorithm's ability to match the throughput of established frameworks like vLLM and S-LoRA while improving fairness among system users.

4.4 Task 4.4 - Security for Models and LLMs in Edge-IoT

This task focuses on security and privacy in modeling Cloud systems. The first research effort focuses on enhancing machine learning models for **intrusion and anomaly detection on resource-constrained edge devices**. By leveraging techniques like Quantization-Aware Training (QAT) and TinyML framework, the study achieved compact and efficient models that maintained competitive accuracy. Also, the second research efforts focused on developing a Retrieval-Augmented Generation (RAG) framework to automate the translation of **high-level Windows security policies (in STIXv2 format) into actionable tasks** and API calls. The methodology integrated a vector database and large language models (LLMs), addressing challenges in precision and recall of API call outputs. The framework showcased its superiority over traditional approaches, significantly improving task-to-API translation accuracy while laying a foundation for applications in broader security contexts.

4.5 Task 4.5 - AI-Driven Orchestration

This last task focuses on AI-Driven orchestration. The first research efforts focus on improving Kubernetes autoscaling by **simulating realistic workloads to optimize resource provisioning**. By leveraging user behavior modeling and application behavior prediction, the performed secondment created a comprehensive model for web server load using open-source tools such as stress-ng, k6s, and Grafana. Real-world data from Google Cloud was employed to validate the approach, achieving high accuracy in predicting workload patterns. The outcome demonstrated the tool's potential for black-box testing in Kubernetes-based autoscaling scenarios, forming a basis for future research.

Furthermore, a second research effort focuses on **integrating Large Language Models (LLMs)** into the NearbyOne platform to **enhance orchestration decisions**. This work aimed to translate user intents into efficient scaling and resource allocation actions while optimizing energy use. Using simulated environments with Prometheus and Thanos, the team demonstrated early success in designing an LLM-powered decision engine capable of interpreting metrics and making actionable recommendations. Ongoing work includes refining scalability and validating results under real-world conditions.

4.6 Tasks and Secondees Relation

		Host	Task 4.1	Task 4.2	Task 4.3	Task 4.4	Task 4.5
BSC	Pol Garcia Recasens	YKT	X				
	Ferran Agulló López	YKT	X		X	X	
ZHAW	Sepideh Shamsizadeh	NBC					X
	Ranjan Ojha	NBC					X
TUM	Jana Vatter	YKT		X			
	Herbert Woisetchläger	YKT				X	
	Alex Isenko	YKT			X		
UMU	Aurora González Vidal	YKT	X			X	
	Carlos Hernández Hidalgo	IRL	X			X	
	Antonio Martínez Ibarra	YKT	X				
	Pablo Fernández Saura	YKT				X	
TUW	Nathanael Nussbaumer	YKT	X				
	Shashikant Ilager	YKT	X			X	

Table 1: Relation of Secondments per Task

5 Conclusions

This deliverable reports and summarizes the secondment tasks performed for Work Package 4 on AI-Driven Cloud Management. Such secondment works have made advances on topics related to optimizing LLMs and classic ML model serving, enhancing the computational efficiency of GPU accelerators on LLM and AI workloads, and strengthening security in AI/LLM processing frameworks.

Task 4.1 focused on innovations in LLM and SLM inference optimization, exploring GPU batching, model replication and energy-efficient strategies. Task 4.2 focused on Graph Neural Networks (GNNs) for automated planner selection, employing hybrid models that integrate GNN architectures with machine learning techniques, improving in resource-efficient and accurate planning. Additionally, Task 4.3 focused on novel scheduling algorithm that effectively balances throughput and fairness in LLM adapter request management. Task 4.4 focused on security for Models and LLMs in Edge-IoT environments, enabling efficient anomaly detection models for constrained devices, and creating RAG frameworks to translate complex security policies into actionable API calls with high precision. Finally, Task 4.5 focused on AI-driven orchestration by advancing Kubernetes autoscaling models through realistic workload simulations and enhancing decision-making via LLM integrations in orchestration platforms.

Such efforts have been performed in the laboratories of IBM New York (USA), IBM Dublin (Ireland) and NearbyComputing (Barcelona). Next secondments are focused on continuing such research lines, with new visits to those industrial partners.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in Advances in Neural Information Processing Systems (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[2] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (C. Zong, F. Xia, W. Li, and R. Navigli, eds.), (Online), pp. 4582–4597, Association for Computational Linguistics, Aug. 2021.

[3] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, eds.), (Online and Punta Cana, Dominican Republic), pp. 3045–3059, Association for Computational Linguistics, Nov. 2021.

[4] G. Qin and J. Eisner, "Learning how to ask: Querying LMs with mixtures of soft prompts," in Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, eds.), (Online), pp. 5203–5212, Association for Computational Linguistics, June 2021.

[5] S. An, Y. Li, Z. Lin, Q. Liu, B. Chen, Q. Fu, W. Chen, N. Zheng, and J.-G. Lou, "Input-tuning: Adapting unfamiliar inputs to frozen pretrained models," ArXiv, vol. abs/2203.03131, 2022.

[6] S. Borst and O. Boxma, "Polling: past, present, and perspective," Top, vol. 26, pp. 335–369, 2018.

[7] R. D. van der Mei and S. C. Borst, "Analysis of multiple-server polling systems by means of the power-series algorithm," Stochastic Models, vol. 13, no. 2, pp. 339–369, 1997.

[8] S. C. Borst, "Polling systems with multiple coupled servers," Queueing systems, vol. 20, pp. 369–393, 1995.

[9] H. Takagi, "Queuing analysis of polling models," ACM Computing Surveys (CSUR), vol. 20, no. 1, pp. 5–28, 1988.

[10] O. J. Boxma, H. Levy, and J. A. Weststrate, "Efficient visit orders for polling systems," Performance Evaluation, vol. 18, no. 2, pp. 103–123, 1993.

[11] M. I. Reiman and L. M. Wein, "Dynamic scheduling of a two-class queue with setups," Operations Research, vol. 46, no. 4, pp. 532–547, 1998.

[12] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," 2023.

[13] Y. Sheng, S. Cao, D. Li, C. Hooper, N. Lee, S. Yang, C. Chou, B. Zhu, L. Zheng, K. Keutzer, J. E. Gonzalez, and I. Stoica, "S-lora: Serving thousands of concurrent lora adapters," 2024.

[14] A. Creswell, M. Shanahan, and I. Higgins, "Selection-inference: Exploiting large language models for interpretable logical reasoning," arXiv preprint arXiv:2205.09712, 2022.

[15] A. Gujarati, R. Karimi, S. Alzayat, W. Hao, A. Kaufmann, Y. Vigfusson, and J. Mace, "Serving dnns like clockwork: Performance predictability from the bottom up," in 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), pp. 443–462, 2020.

[16] A. Ali, R. Pinciroli, F. Yan, and E. Smirni, "Optimizing inference serving on serverless platforms," Proceedings of the VLDB Endowment, vol. 15, no. 10, 2022.

[17] A. Gopalakrishnan, N. P. Kulkarni, C. B. Raghavendra, R. Manjappa, P. Honnavalli, and S. Eswaran, "Primed: Private federated training and encrypted inference on medical images in healthcare," Expert Systems, p. e13283, 2022.

[18] N. Razfar, J. True, R. Bassiouny, V. Venkatesh, and R. Kashef, "Weed detection in soybean crops using custom lightweight deep learning models," Journal of Agriculture and Food Research, vol. 8, p. 100308, 2022.

[19] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in International conference on machine learning, pp. 6105–6114, PMLR, 2019.

[20] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," arXiv preprint arXiv:1706.02677, 2017.

[21] P. Kousha, B. Ramesh, K. K. Suresh, C.-H. Chu, A. Jain, N. Sarkauskas, H. Subramoni, and D. K. Panda, "Designing a profiling and visualization tool for scalable and in-depth analysis of high-performance gpu clusters," in 2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC), pp. 93–102, IEEE, 2019.

[22] A. Isenko, "Basic Hardware Monitor," Sept. 2023.

[23] A. González-Vidal, A. Isenko, and K. Jayaram, "On serving image classification models," in Proceedings of the 9th International Workshop on Serverless Computing, pp. 48–52, 2023.

[24] C. C. Consortium, "Confidential computing: Hardware-based trusted execution for applications and data," Nov. 2022.

[25] C. C. Consortium, "A technical analysis of confidential computing," Nov. 2022.

[26] C. C. Consortium, "Common terminology for confidential computing," Dec. 2022.

[27] T. L. Foundation, "The case for confidential computing: Delivering business value through protected, confidential data processing," July 2024.

[28] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," arXiv, Oct. 2024.