Managing Computational Resources with Machine Learning Policies

Alexandros Patras

Computer Systems Lab (CSL)
Department of Electrical and Computer Engineering
University of Thessaly, Greece

Funded by the European Union

This project has received funding from the European Community's Horizon Europe Programme under Grant Agreement #101092912.

MLSysOps

Introduction

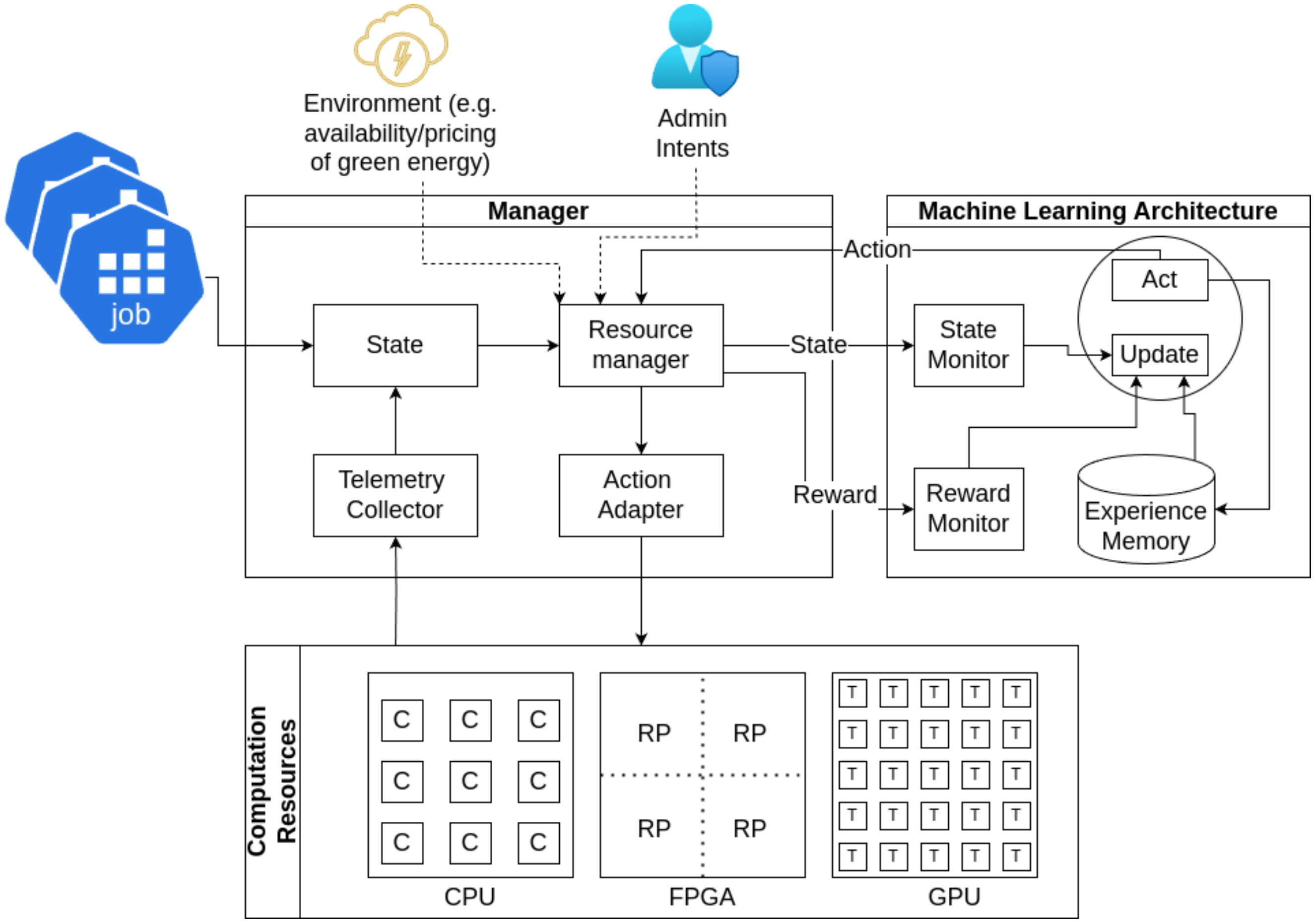
- **Challenge**
  - Efficient Resource Allocation in Edge - Cloud Environments
- **Problem**
  - How to dynamically balance performance requirements, energy efficiency, and system constraints
  - Heterogeneous computing devices (CPUs, GPUs, FPGAs), workload diversity, multiple configuration options
  - Our Focus is on ML workloads
- **Approach**
  - Adaptive Reinforcement Learning (RL)-based congiruation engine in multicore CPUs in cloud nodes.

System Architecture

Key components

- A **system manager** that constantly monitors the system and accepts new job requests.
- An **RL architecture** to guide resource allocation and configuration decisions based on the State of the system
- **Telemetry** collectors to capture HW metrics and **Action** adapters to apply any needed configuration.
- **Heterogeneous** node architecture comprising CPUs, GPUs, FPGA

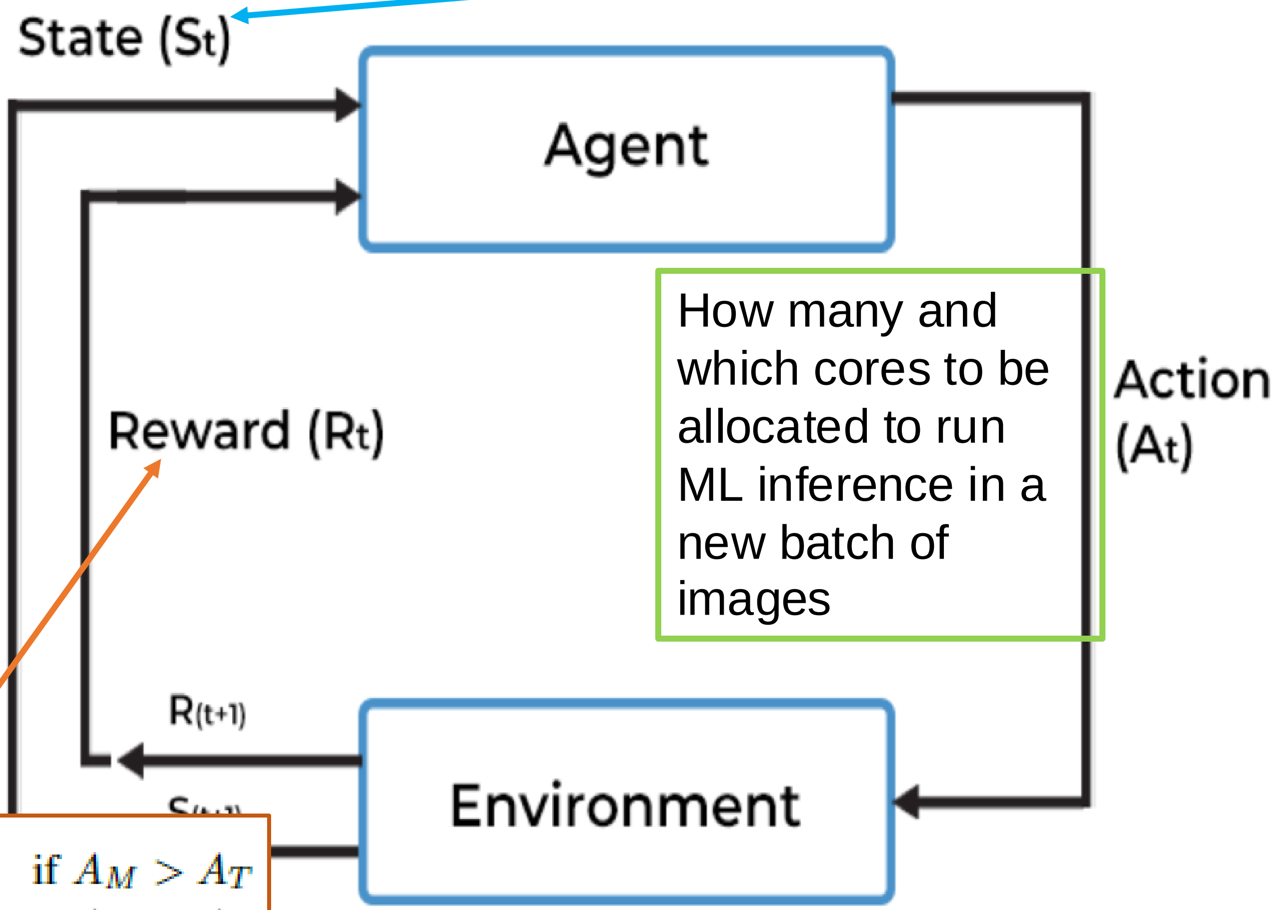
  Note: Only multicore CPUs in this work



Reinforcement Learning Agent

- RL is a type of ML algorithm where an **Agent** learns how to take **Actions** by interacting with the **Environment** to maximize a cumulative **Reward**.
- Our **objective** is to reduce power dissipation without exceeding a user-defined execution latency.
- **Reward** function penalizes Actions that result in high latency (1st criterion) and high power (2nd criterion)

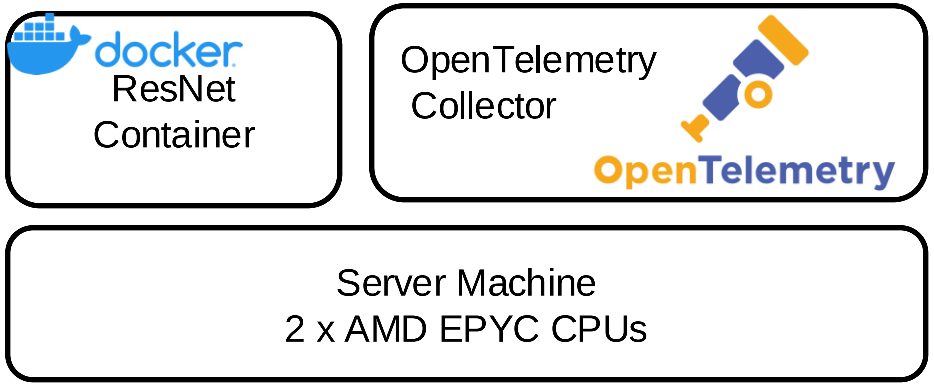
1. Average CPU utilization per core
2. Power Dissipation per Core
3. ML inference latency



$$reward = \begin{cases} -100 & \text{if } A_M > A_T \\ -\sqrt{Power_C} & \text{if } A_M \leq A_T \end{cases}$$

Experimental Set up

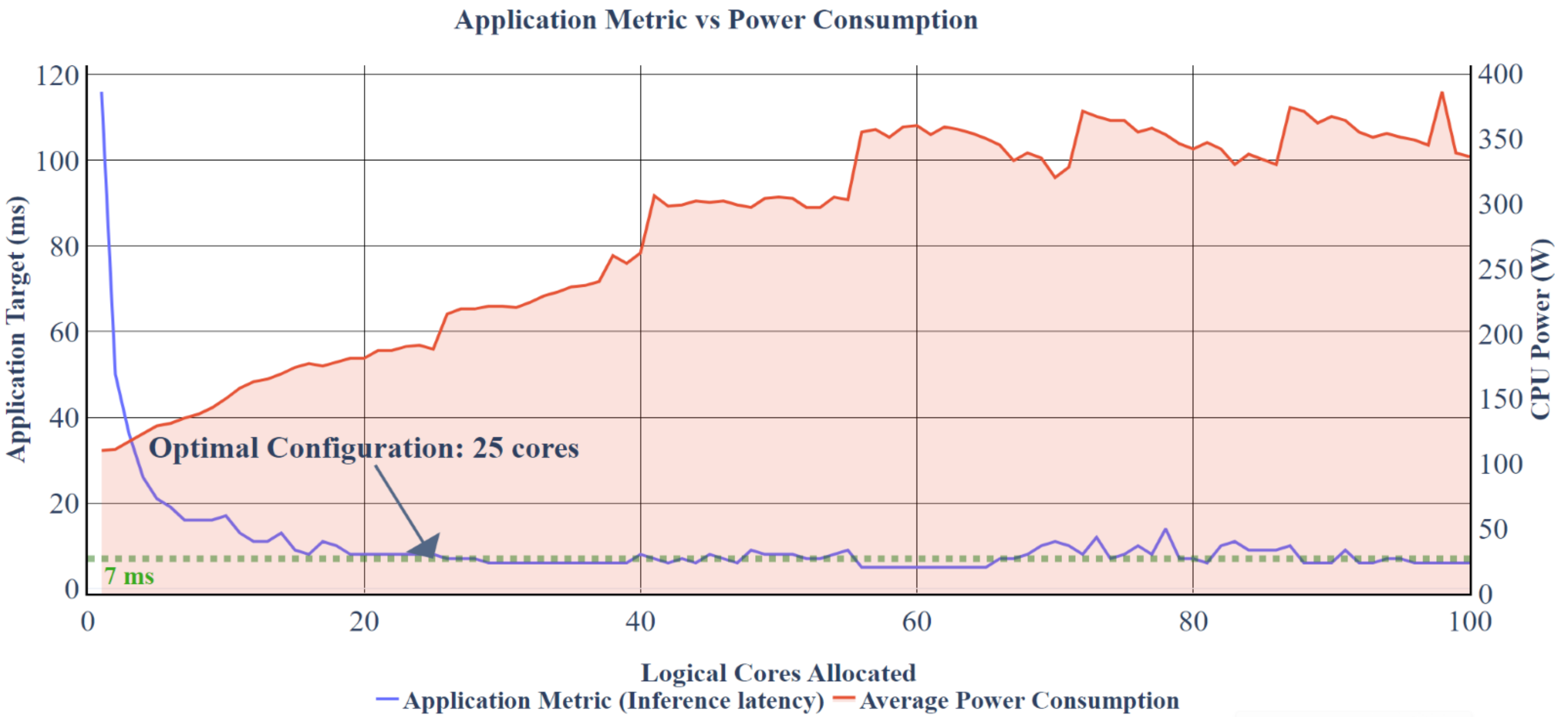
- Datacenter-grade node consisting of a dual-socket AMD EPYC, 64 physical (128 logical) cores
- 1-100 Logical cores actually used
- Metrics recorded using Open Telemetry tools
- Containerized ResNet used as workload
- Trained the RL agents using 200,000 timesteps & offline.



Experimental Evaluation (I)

Set Application Latency Target A_T for batch inference to 7ms

- optimal configuration when 25 logical cores are allocated to run ResNet in parallel
- beyond that, performance stagnates while power (obviously) increases



Experimental Evaluation (II)

Experiments for various Application Latency Targets A_T

RL agent achieves allocations which are very close to the optimal allocations

Application Latency Target (ms)	Predicted Configuration (Cores to allocate)	Handpicked configuration (Cores to allocate)
5	57	61
7	25	25
10	35	34
15	10	11

Conclusion and Future Work

- RL models achieve near-optimal core allocation for performance-constrained ML inference having power dissipation as optimization criterion
- The node-level manager is part of a larger mechanism that implements the MAPE model (Monitor, Analyze, Plan, Execute) at the Cloud-Edge continuum
- Future Work (node-level)
  - Use on Hardware accelerators (GPUs, FPGAs)
  - Thread affinity, voltage/frequency scaling
  - Different versions of the ML workload spanning performance vs accuracy
  - Carbon intensity as additional optimization criterion
  - Data collection

Thank you!