End-to-end Autonomic Management of Cloud-Edge Continuum Systems using Reinforcement Learning: Design Approach and Formulation

Theodoros Aslanidis, Dimitris Chatzopoulos

ENHANCE – HiPEAC 2024 - Munich, Germany

Funded by the European Union

European Commission | HORIZON EUROPE

Cloud computing

- **Involves delivering on-demand computing services** including storage, processing power, and applications, over the internet, enabling flexible and scalable access to shared resources.
- **Enables users to access and utilize resources** without the need for physical ownership or maintenance of infrastructure.
- **Provides a centralized platform** for deployment and management of applications, enabling collaboration and efficiency.
- **Limitation:** The escalating number of IoT devices produce massive amounts of data that cloud computing cannot handle.

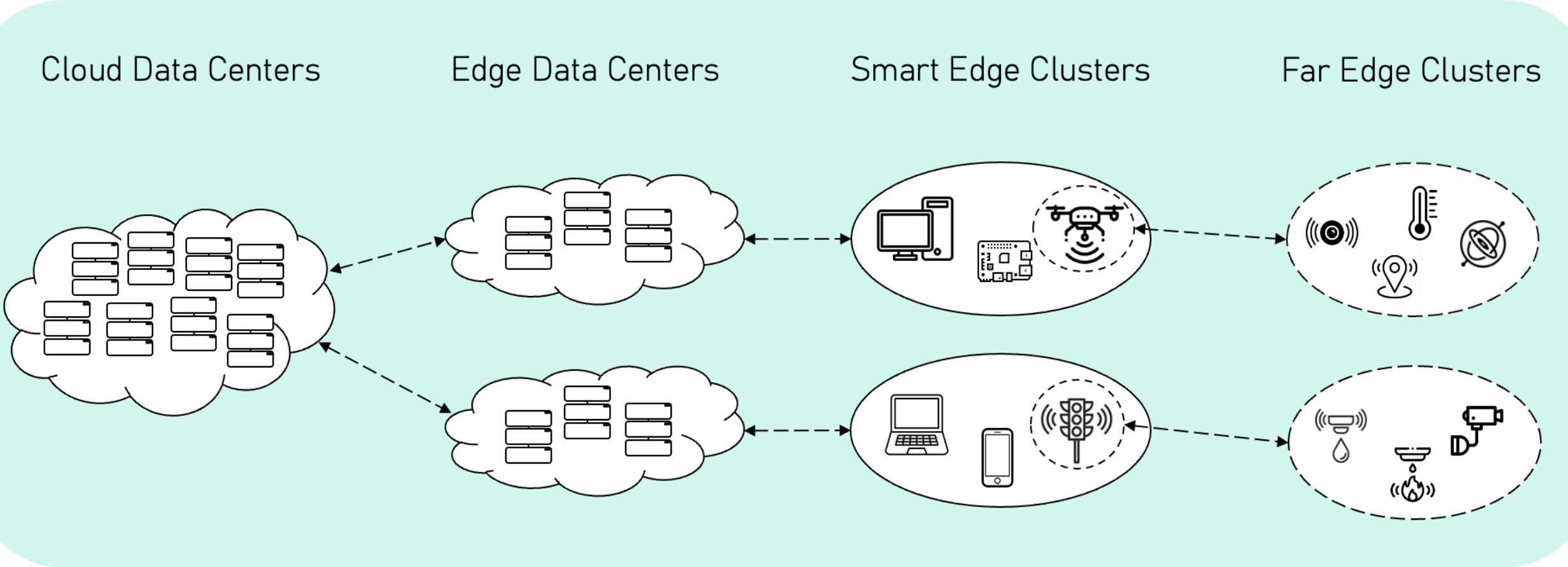
Edge computing

- A distributed computing paradigm that brings processing and storage capabilities **closer to data sources**.
- Significantly **decreases latency** for real-time applications and IoT systems.
- **Enhances reliability and bandwidth efficiency** by minimizing network distance and transmitting only relevant data.
- Employed in **real-time applications** across various sectors, including IoT, smart cities, manufacturing, healthcare, and transportation.
- **Limitation:** not enough computing power

The cloud-edge continuum

- A **hybrid model** that emerges to **overcome the limitations** and challenges opposed by the previous models.
- Combines **robust computing power** and management capabilities of cloud computing and **real-time data processing** of edge computing.
- Includes the spectrum of computing resources from **centralized cloud data centers** to **decentralized edge devices.**
- Ideal for scenarios requiring a **balance** between **centralized control** and **local processing** such as versatile applications in IoT, cyber-physical systems, smart cities, healthcare, and industries demanding real-time responsiveness.

The cloud-edge continuum



ML-assisted cloud-edge continuum

- To **enhance adaptability**, ML techniques can be incorporated into the cloud-edge continuum.
- ML is responsible **managing** and **controlling** the dynamic and heterogeneous nature of the cloud-edge continuum.
- This helps **optimizing decision-making** and **improving overall efficiency** offering **optimal performance** and **responsiveness**.

The outer picture

- The system, viewed externally, operates as an AI-enabled entity
- Makes complex decisions to strike a balance between:
  - overall performance
  - resource utilization
  - energy consumption

Under the hood

- Internally, the system will use a conventional resource management framework with **AI-assisted decision-making**.
- Depending on whether the decision is of high importance, this can be a rule-based or heuristic approach or ask AI to make a more informative decision.
- Different ML models can assist in different types of decisions.
- The AI-assistant can be switched on and off by a system administrator.

Three-layer hierarchical agent structure

- Due to the high **heterogeneity** and the **dynamic** nature of the cloud-edge continuum, we propose a **three-level** agent hierarchy.
  - Provide scalability, adaptability, and efficient resource utilization.
  - Enables specialized agents at each level to handle specific tasks, from fine-grained resource control at the node level to strategic deployment decisions at the continuum level.
- Each agent at every level has access to a diverse set of **ML models**, that are tailored to its specific objectives and requirements.

Responsibilities per level

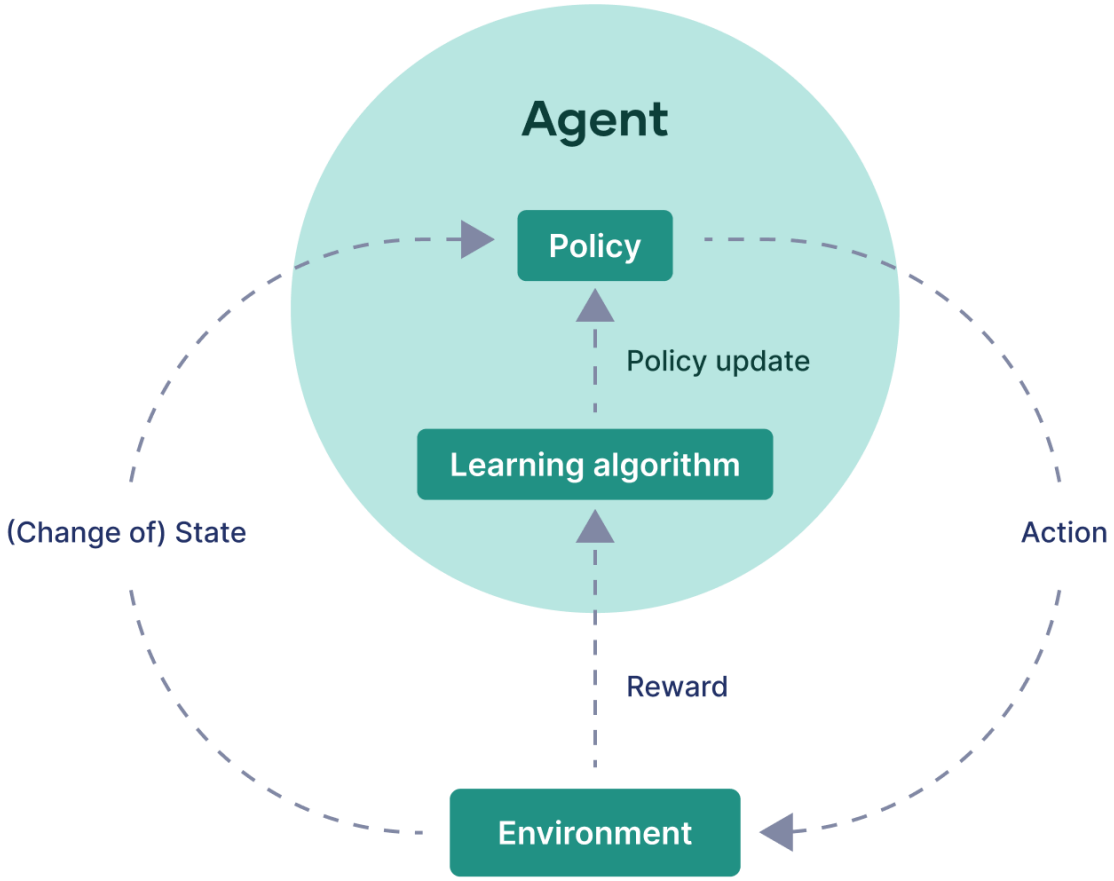
Node	Cluster	Continuum
Device selection (CPU, GPU, FPGA)	Node selection	Cluster selection
CPU, GPU frequency selection	Node connectivity	Application Deployment
Affinity to cores	Resource distribution	Objective Selection
Memory allocation	Resource optimization	Efficiency Optimization
Storage allocation	Load balancing within cluster	Interaction with sys admin
ML Model Selection		
Trigger ML Retraining Process		
Anomaly Detection and Trust Evaluation		
Communicate with other agents		
Logging		

Using RL techniques

- **Why?**
  - Handling non-periodic and complex user patterns.
  - Adapts to dynamic and heterogeneous environments.
  - Providing long-term, strategic decision-making and planning.

- **Exploring RL paradigms:**
  - Hierarchical RL: organize complex tasks into a hierarchy of subgoals
  - Multi-agent RL: multiple agents work together to achieve a collective goal
  - Multi-objective RL: optimize multiple conflicting objectives simultaneously
  - Constrained RL: optimize objective within predefined constraints
  - Combination of the above

General Reinforcement learning framework



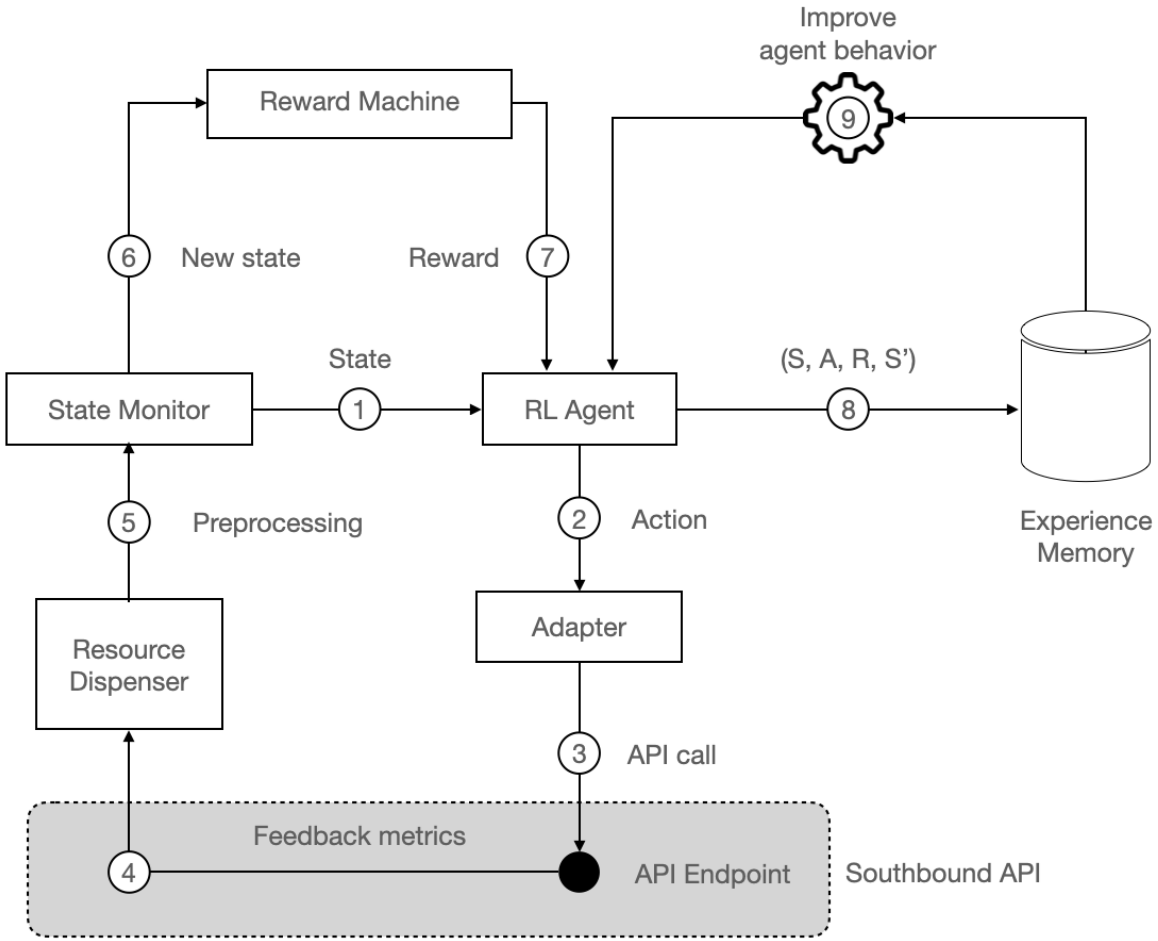
- The *agent*
  - Lives in an *environment*
  - Observes the current *state*
  - Interacts with the environment by making *actions*
  - As a response, receives:
    - The next state
    - A positive or negative *reward*

- A *policy* maps states to actions
  - drives the agent's behavior

RL problem formulation

RL component	Formulation	Example
Agent	Entity that decides a system configuration in the cloud-edge continuum	Entity that decides the application deployment within the cloud-edge continuum
Environment	The cloud-edge continuum	A system with a cloud data center, an edge data center, a workstation and a smart edge device with sensors
State	Telemetry data of the system	CPU utilization of all nodes
Action	Command that triggers a change in system configuration	Migrate a component from one node to another
Reward	Utility function that returns higher values as the performance of the system improves	$R = w_{m_1} \cdot m_1 + \ \ldots + w_{m_k} \cdot m_k - w_{p_1} \cdot p_1 - \ \ldots - w_{p_l} \cdot p_l$ $R: reward\ function$ $m_n:\ system\ metric\ parameter\ to\ maximize$ $w_{m_n}: system\ metric\ parameter\ weight$ $p_n: penalty\ parameter\ to\ minimize$ $w_{p_n}: penalty\ parameter\ weight$

Design approach



Step	Description
1	The State Monitor sends the current state to the RL agent
2	The RL agent chooses the best action
3	The Adapter translates the action to the corresponding API call
4	The system sends the feedback metrics to the Resource Dispenser
5	The Resource Dispenser sends the metrics to the State Monitor
6	The State Monitor creates a new state
7	The Reward Machine calculates the reward and returns it to the agent
8	The (state, action, reward, new state) is saved to the experience memory
9	Improve the agent's behavior

Comparison of different RL categories

RL Category	Benefits	Limitations	Common algorithms
Value-based	Strong convergence properties	Struggle on high-dimensional and continuous action spaces	Q-learning, SARSA, DQN, DDQN
Policy-based	Efficient on large and continuous action spaces	Unstable and slow convergence due to high variance	REINFORCE, PPO, TRPO
Actor-critic	Improved stability	Increased complexity	DDPG, PPO, A2C, A3C, TD3, SAC

Challenges

- **Reward function design:**
  - Crafting a reward function that captures the system performance sufficiently.
- **Telemetry data granularity:**
  - Determining the level of detail in telemetry data to capture system dynamics accurately.
- **Dynamic objective switching:**
  - Implementing the ability to seamlessly switch among diverse system objectives.
- **Global-to-local objective translation:**
  - Translating global objectives into specific, low-level system actions.
- **Learning Feasibility and Robustness:**
  - Ensuring learning is not only feasible but also robust and efficient.

Thank you!

Any questions?

Feel free to contact us at:

theodoros.aslanidis@ucdconnect.ie

Funded by the European Union

European Commission | HORIZON EUROPE