SAMI 2023 • IEEE 21st World Symposium on Applied Machine Intelligence and Informatics • January 19-21, 2023 • Herl'any, Slovakia

Evaluation and technological solutions for a dynamic, unified cloud programming development environment

Ease of use and applicable system for uniformized practices and assessments

|  |  |  |
| --- | --- | --- |
|  |  |  |
| György Molnár  *Óbuda University*  *Kandó Kálmán Faculty of Electrical*  *Engineering*  Budapest, Hungary molnar.gyorgy@uni-obuda.hu  *Széchenyi University,*  *János Csere Apáczai Faculty*  Győr, Hungary molnar.gyorgy@sze.hu | Cserkó József  *John Von Neumann University*  *GAMF Faculty of Engineering and*  *Computer Science*  Kecskemét, Hungary  *Széchenyi University,*  *Doctoral School of Multidisciplinary*  *Engineering Sciences (MMTDI)*  Győr, Hungary  cserko.jozsef@gamf.uni-neumann.hu | Karl Éva  *Várkerti Primary School*  Várpalota, Hungary  *Széchenyi University,*  *Doctoral School of Multidisciplinary*  *Engineering Sciences (MMTDI)*  Győr, Hungary karl.eva@varkerti.hu |

***Abstract*—With the acceleration of the Internet in Web 2.0, Cloud computing is a new paradigm to ofer dynamic, reliable and elastic computing services. Efcient scheduling of resources or optimal allocation of requests is one of the prominent issues in emerging Cloud computing. Considering the growing complexity of Cloud computing, future Cloud systems will require more efective resource management methods. In some complex scenarios with difculties in directly evaluating the performance of scheduling solutions, classic algorithms (such as heuristics and meta-heuristics) will fail to obtain an efective scheme. Deep reinforcement learning (DRL) is a novel method to solve scheduling problems. Due to the combination of deep learning and reinforcement learning (RL), DRL has achieved considerable performance in current studies. To focus on this direction and analyze the application prospect of DRL in Cloud scheduling, we provide a comprehensive review for DRL-based methods in resource scheduling of Cloud computing. Through the theoretical formulation of scheduling and analysis of RL frameworks, we discuss the advantages of DRL-based methods in Cloud scheduling. We also highlight diferent challenges and discuss the future directions existing in the DRL-based Cloud scheduling.**

2023

IEEE 21st World Symposium on Applied Machine Intelligence and Informatics (SAMI) | 979-8-3503-1986-6/23/$31.00 ©2023 IEEE | DOI

: 10.1109/SAMI58000.2023.10044519

**The isolated environment offered by the virtual space guarantees private work sessions, and multi-project activities can also be handled more easily and uniformly. The analysis of the teaching program presented below provides insight into the practical and appropriate solutions offered by the containerization process. In addition to these, the topic will also cover the widely used and popular Docker application that supports the implementation of the process, as well as the NodeJS environment closely related to the project.**

***Keywords—teach program, virtual environment, containerization, Docker, NodeJS, cloud-based services***

INTRODUCTION

In recent years, there has been steady growth in the development of cloud-based Integrated Development Environments (IDEs) to provide fast and reliable development services to software developers. The trend has been accelerated by the emergence of Docker Container-based online IDE solutions, which allow developers to access their development environments from anywhere with just an internet connection. This paper examines the effect of Docker Container-based online IDE [2] solutions on software developers from an economical and practical perspective. Furthermore, this paper will explore how Docker Containerbased online IDE solutions can revolutionize software development.

The development of modern computing systems and their applications have brought about various programming environments, each with advantages and disadvantages. For example, specific programming languages are more suited for web development, while other programming languages provide better support for machine learning algorithms. Additionally, specialized programming environments exist to solve distinct programming tasks, such as scripting and software automation, but they come with decreased readability compared to more general-purpose programming languages. Furthermore, programming environments can also be divided into language-specific and platform-specific ones. Unfortunately, platform-specific programming environments often require significant effort in their setup, which is often incompatible across different computers. Additionally, relying on platform-specific libraries and frameworks can introduce version compatibility and maintenance issues. These problems can be daunting to newcomers and experienced programmers and should be considered when selecting a programming environment.

Uniformized programming environments offer significant advantages to students. These environments provide a preconfigured and consistent work environment easily shared between users and machines. This uniformity provides a common platform for course material, enabling efficient collaboration and providing a consistent experience across classrooms and institutions. Furthermore, these environments allow efficient tracking of once identified issues and bugs and enable efficient sharing of software libraries and code among users. Finally, these environments can also offer connectivity to external tools, such as cloud resources, allowing students to access additional software, data, and resources [14]. Overall, uniformized programming environments provide students with consistent, accessible, and robust platforms to facilitate their learning of programming.

In our work, we create a unique, virtual-server-based system for programming students. This solution is helpful for both practice and examination purposes.

ALTERNATIV SOLUTIONS

There are many paid or free tools on the Internet. Unfortunately, most of them are too expensive for education purposes.

1. Cloud9: Cloud9 is a development platform that lets anyone write, run, and debug code in various

979-8-3503-1986-6/23/$31.00 ©2023 IEEE 000237 Gy. Molnár et al. • Evaluation and Technological Solutions for a Dynamic, Unified Cloud Programming Development Environment

languages, in any browser, or on any platform. It comes pre-installed with popular development tools and integrated libraries for many languages.

1. AWS Cloud9 IDE: Amazon Web Services Cloud9 IDE offers an integrated development environment based on the Eclipse platform. The IDE contains unique features that allow developers to create, debug, and deploy applications on AWS's servers.
2. Codeanywhere: Codeanywhere is a collaborative cloud-based IDE that allows teams of developers to work side-by-side on projects. It includes support for a variety of programming languages, as well as integration with popular version control systems.
3. Code Envy: Code Envy is a cloud-based development environment that supports popular programming languages, such as Java and

JavaScript. It allows developers to edit, debug, and deploy their code in the cloud without any external installation.

1. Koding: Koding is a cloud-based IDE that provides a graphical interface for quickly creating and testing web applications. It supports various programming languages, including Python, Ruby, HTML, CSS, and more.

These are good solutions for companies or individuals, but some abilities are missing for education purposes.

Missing important features: full access to the virtualized operating system, teachers can create templates, and preinstalled software.

TECHNOLOGY FOUNDATIONS

*VPS*

A Virtual Private Server (VPS) is a virtualized server, a software simulation of physical server hardware. It is a secure and cost-effective solution to host websites and applications, enabling multiple users to run multiple operating systems and software on a single server. A VPS typically also provides data storage and remote access via the Internet, creating a mix of physical and virtual infrastructure. VPS architecture allows us to create cost-effective applications.

We use an Ubuntu VPS. This OS is the host of Docker containers. The VPS runs a special software for proxying HTTP requests named HAProxy.

HAProxy is a high-performance load balancer and a reverse proxy that enables an optimized delivery of applications. It helps improve an application's availability and performance by distributing the workload across multiple servers. It can also be used for other types of network traffic, such as HTTP, MSBB, DNS, TCP, and UDP. [9] In addition, HAProxy can detect server failures in real-time and can be used as a proxy for HTTP, HTTPS, SMTP, POP3, and IMAP servers. It also supports WebSockets, content switching, and SSL/TLS offloading, among other features.

Furthermore, HAProxy can route requests to multiple web servers while ensuring that the traffic is distributed optimally. It also helps to ensure that IP whitelisting and rate-limiting rules are applied. HAProxy provides SSL certificate management, automatic SPOF prevention, and health checks to ensure that only healthy servers receive traffic.

HAProxy has config files that contain domain mappings. The host machine runs Docker containers on specified ports and HAProxy redirects requests to these.

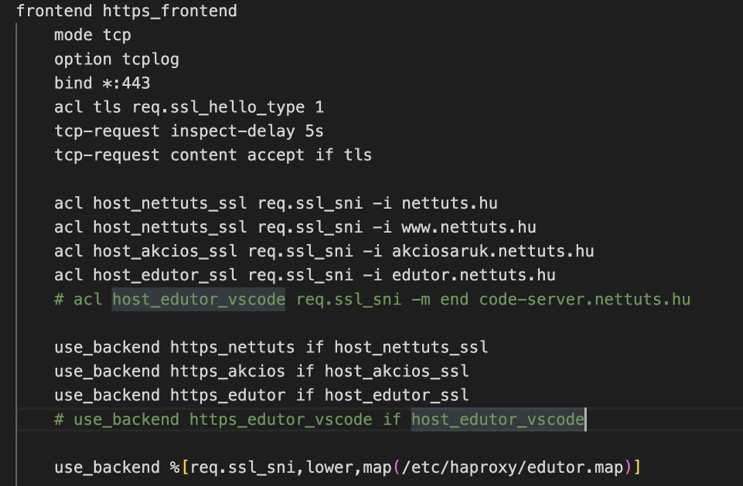


Fig. 1. Example HAProxy config

*Docker*

Docker [1] is an open-source software platform that enables developers to easily package, deploy, and manage applications in a virtual environment known as a container. Docker Compose is an open-source tool for defining, running and managing multi-container applications. It provides a way to define and configure all the services (containers) that make up an application in a YAML file. It also allows users to manage and scale their applications effectively.

To manage containers, we use dockerfile and dockercompose files. The dockerfile contains the configuration of the container instance, such as VSCode server installation steps, exposed ports, and secured directories.

The docker-compose file contains environment variables and a list of real ports.

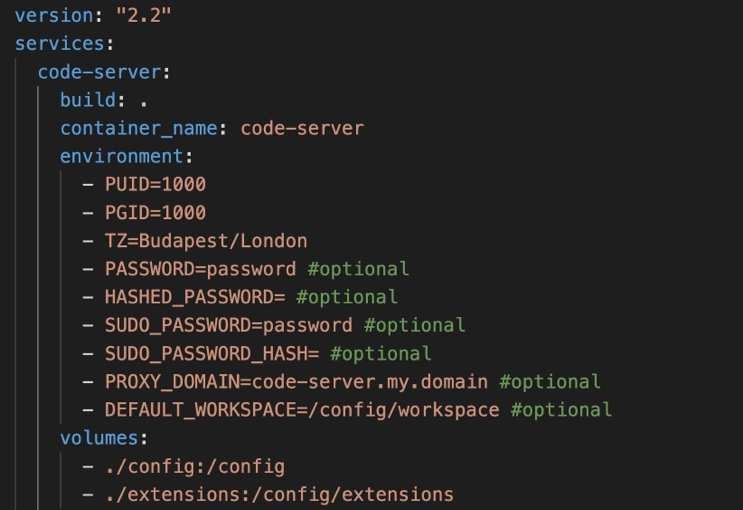


Fig. 2. Example docker-compose.yml

*Web Server*

NodeJS is a JavaScript-based run-time environment for building web applications. NodeJS allows us to write serverside JavaScript code for our web applications and run it directly in the server environment. [8] In addition, NodeJS is an open-source software, so anyone can use it to create great web applications quickly and easily.

We needed a secure and easy-to-use interface to manage containers. We chose NodeJS and a web interface. Each

000238 SAMI 2023 • IEEE 21st World Symposium on Applied Machine Intelligence and Informatics • January 19-21, 2023 • Herl'any, Slovakia

Fig. 3. The login page

DIFFICULTIES

So, therefore this project was pretty complex, and there were many difficulties.

When students start in a new environment, the system often duplicates them. So we created a routine that checks the environment names and opens the existing ones if available.

The second was with the resources. [4] We ran Angular applications in the Docker container. Angular needs a lot of memory and CPU time for building applications because it is built with TypeScript. Browsers do not know TypeScript code; therefore, before running an application in the browser, it is necessary to create JAVASCRIPT codes from it. Converting TypeScript to JAVASCRIPT is the building task.

When the container ran with 1GB of memory, the VSCode server hung an out-of-memory error. So we had to find the proper memory amount for this task. It will be 1.3GB of memory. After all, we created a new and more memoryefficient builder for Angular written by GoLang instead of the original NodeJS implementation.

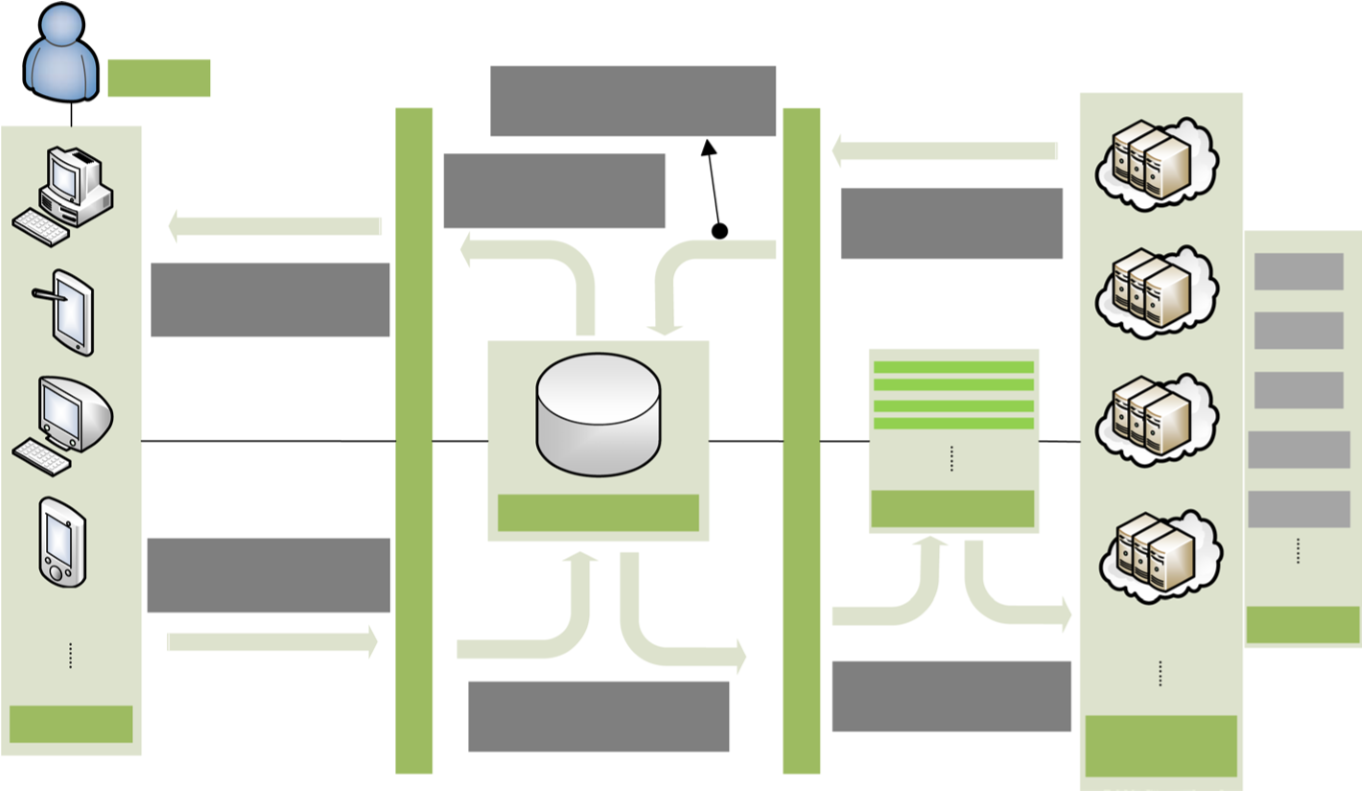
We found that each setup requires thorough pre-testing for good functionality. It requires testing memory, CPU, and file system.

SUMMARY

Cloud-based development environments provide quick, easily accessible opportunities for development, including collaboration [5] [14]. They also support education well by significantly reducing the time required to prepare and manage the technical implementation of educational processes. In the current educational system, the role of teachers must also constantly evolve and adapt to the direction of digitalization [6] [11] [15]. With central regulation, it is possible to quickly and efficiently provide the necessary technological background for teaching, and a failure of a particular machine does not affect the entire system [10] [12]. There are many cloud servers that offer their services to the market, but we must consider that their use can be financially burdensome for education and teachers may have limited options. Taking into account modern technological opportunities, this article presents a solution for creating a unique, virtual server-based system designed for the unified and efficient teaching of programming to university students.

Using VPS Docker, we provide a container-based environment for students to work on a unified interface, yet independently and without interference. The containers run simultaneously on the server, and Docker provides a platform for managing the containers that allow for the setting of different environmental variables, installation, development, and testing of applications in an isolated environment [7] [13]. We provide an Ubuntu LTS server system for VPS. This Debian-based Linux operating system provides a high level of security and scalability for such systems, and it is completely free. The biggest challenge in our work was the proper use of resources, but an effective solution to this could be the further development or migration of the application to Kubernetes. The current development environment is only suitable for serving small groups, but with Kubernetes, it can easily be integrated into the system of large providers, and the management of resources is also optimized.

The benefits of using virtual environments are undeniable, and the educational programs running in them also bring the same advantageous characteristics, such as mobility, flexibility, and consideration of personal attitudes, as their counterparts used on other platforms. With the system we introduced, we wanted to present an alternative option for using technology for educational purposes, which we plan to develop further in order to publish many additional possibilities in the future.



Cloud computing provisions computing resources based on CPU, RAM, GPU, Disk Capacity, and Network Bandwidth. Additionally, "time" and "space" are crucial resources. Time refers to the service life cycle of the Cloud platform, while space indicates the physical location for devices. Electrical components in Cloud systems rely on energy and operate within these time and space constraints. Therefore, Cloud computing fundamentally utilizes electric energy conversion per unit of space and time. The limited resource capacity of Cloud computing increases costs and energy consumption. Moreover, issues like long response times and delays can reduce the quality of service (QoS). Efficient, energy-saving resource scheduling is critical for the future of Cloud computing.

Challenges arise due to the vast scale of devices, the complexity of scenarios, unpredictability of user requests, randomness of components, and variable operating temperatures. Common strategies to resolve resource management include multi-phase approaches, virtual machine migration, queuing models, service migration, workload and application migration, task migration, and scheduling algorithms. The scheduler, designed based on these algorithms, is central to efficient resource management. Users submit tasks to a Cloud center, which uses scheduling algorithms to allocate tasks to server nodes, providing services to users.

The scheduling problem in distributed systems is typically NP-complete or NP-hard. Solutions involve various methods, including Dynamic Programming, Probability algorithms, Heuristic methods, Meta-Heuristic algorithms, Hybrid algorithms, and Machine Learning (ML). While traditional methods struggle with the complexities of Cloud computing scheduling, the application of ML, especially Deep Reinforcement Learning (DRL), has shown promise.

There are many surveys that provide detailed, comprehensive reviews of various fields in Cloud computing. Examples related to Cloud resource optimization management include:

**5**

A fundamental framework of RL

* Reviews of workflow scheduling in Cloud and analyses of its techniques, classifying them based on objectives and execution mode.
* Focus on performance interference of virtual machines, revisiting interference-aware strategies and co-optimization-based approaches for scheduling optimization.
* Literature surveys on task scheduling strategies, including meta-heuristic algorithms and discussions on scheduling methodologies and limitations.
* Reviews of load balancing algorithms for virtual machine placement, covering heuristic, meta-heuristic, and hybrid algorithms.
* Comprehensive surveys of evolutionary approaches in Cloud resource scheduling, including genetic algorithms, ant colony optimization, and particle swarm optimization.
* Taxonomy of meta-heuristic scheduling techniques in Cloud and fog computing, from physics-based to biology-based algorithms.

Some surveys discuss the application of machine learning (ML) in Cloud scheduling. For example, ML methods for resource provisioning in edge-Cloud applications include deep neural networks (DNN), support vector machines (SVM), decision trees, Bayesian networks, and others. In mobile edge computing, ML models like fuzzy control, tree-based naive Bayes, and SVM have been applied for computation and communication control. Other reviews focus on prediction and classification approaches like SVM, k-nearest neighbors (KNN), and deep learning (DL) in Cloud resource management.

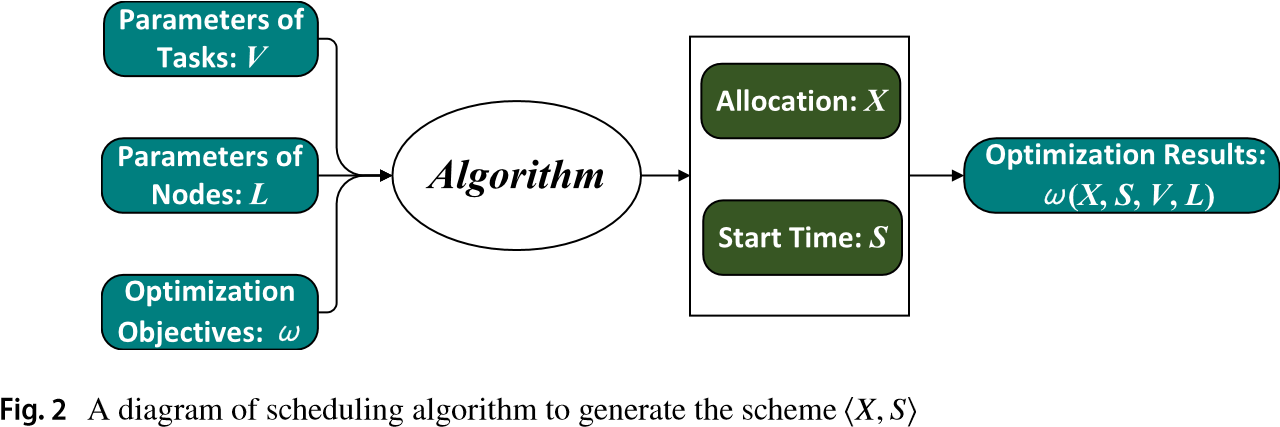
While there are many surveys covering ML applications in Cloud scheduling, there is no specific survey on deep reinforcement learning (DRL) in this area, which is a novel and evolving field. Researchers are actively exploring the use of reinforcement learning (RL), particularly DRL, in Cloud scheduling.

Recognizing the potential of DRL in Cloud scheduling, a comprehensive survey is needed for existing research using DRL-based methods. The key contributions of this paper include:

1. A comprehensive review of existing scheduling algorithms in Cloud computing.
2. An analysis of the frameworks of RL and DRL from a model structure perspective.
3. A structured review of existing research using DRL in Cloud scheduling.
4. Identified challenges and future directions for DRL-based methods in Cloud scheduling.

The rest of the paper is structured as follows:

* Section 2 reviews existing scheduling algorithms in Cloud computing.
* Section 3 presents an analysis of RL and DRL structures in Cloud scheduling.
* Section 4 discusses existing research using DRL methods in Cloud scheduling.
* Section 5 lists challenges and future directions for applying DRL in Cloud scheduling.
* Section 6 concludes the paper.



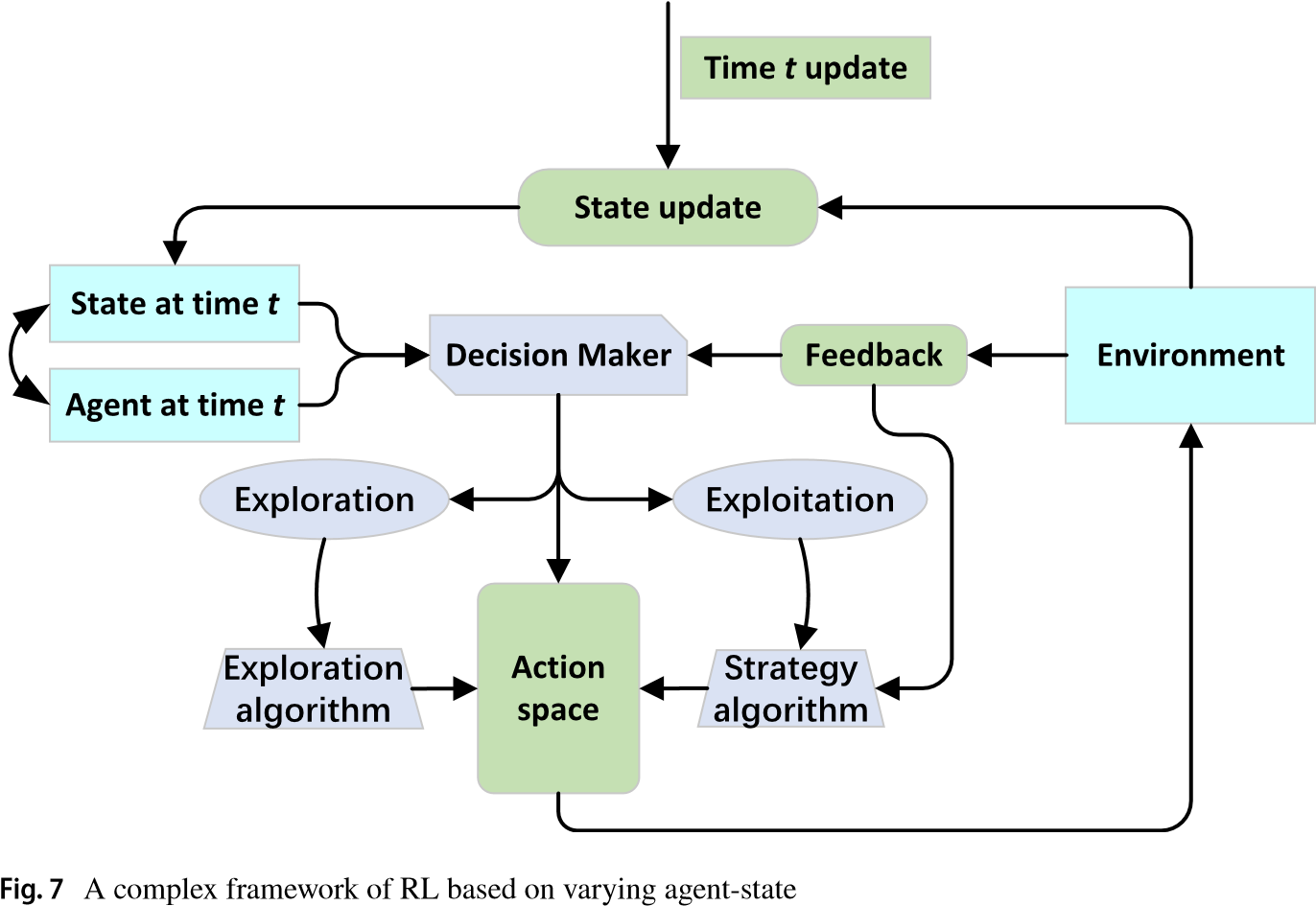
In reinforcement learning (RL), the agent learns a strategy through trial and error, requiring a balance between exploration and exploitation. Exploration involves trying new actions to discover their effects, while exploitation focuses on making the best decision based on known information. Methods like the greedy approach, random selection, and metaheuristic techniques simulate the decision-making process between exploration and exploitation. The Markov Decision Process (MDP) is commonly used to model the action choice process, while the Bellman Equation, a dynamic programming equation, is frequently used to update the action strategy. This RL framework, consisting of action selection and strategy updates, is illustrated in Fig. 6.

Although the RL framework depicted in Fig. 6 can solve some optimization problems, it lacks sufficient consideration for temporal changes in both system and agent states. As a result, it is inadequate for addressing time-related scheduling problems.

In more complex scenarios, the state of both the system and agent changes over time. Decisions must be made in real-time based on these evolving states, and feedback from the environment directly influences the strategy. Therefore, an agent-state-based RL framework, as shown in Fig. 7, is more suitable for such cases.

In most real-world situations, systems are not completely independent and often change due to external stimuli. The environment in Fig. 7 represents the internal system environment, which cannot account for interference from other external systems. As a result, the RL system in Fig. 7 should be considered an autonomous system because the agent and environment evolve according to fixed rules. Autonomous systems include examples like computer games, Go, and large-scale natural language processing tasks, where the system's regulation remains stable without external modification.

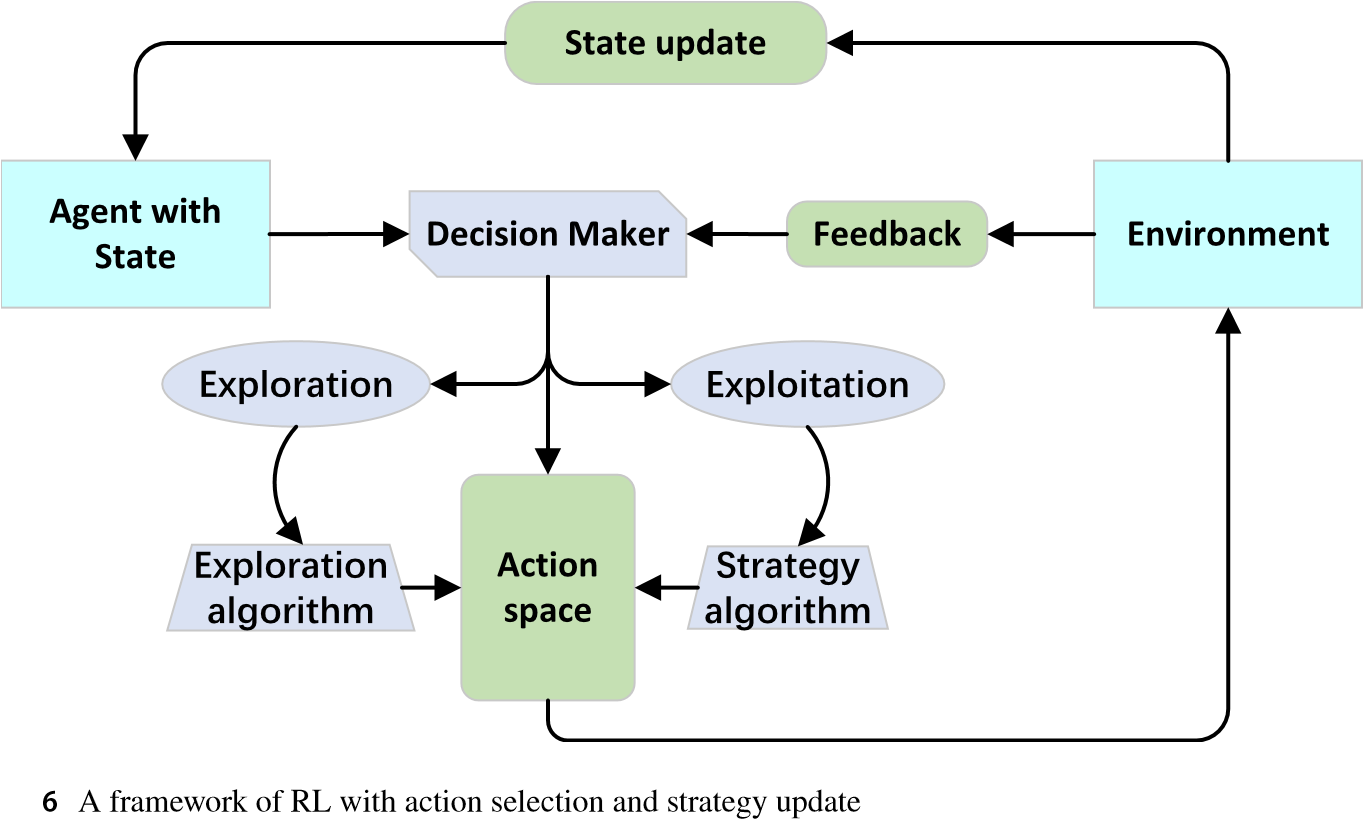
However, certain systems, such as vehicle movement or competitive sports, are affected by external factors. Similarly, a Cloud computing system is not autonomous. It is influenced by time-varying user demands, optimization objectives, and requests. The update function of the internal environment, as well as the agent-state and decision-making function, also varies over time. For instance, the revenue ratio of Cloud computing can fluctuate throughout different periods of the day. Thus, the decision-making process in Cloud computing must adapt to these dynamic changes.



ADDITIONAL OPPORTUNITIES

We need to create an admin panel to change the program's settings. The required subpage is environment manager, user manager, and resource manager.

We plan to use Kubernetes for effective container management. [3]

Finally, there is a requirement for a template system for docker-compose files. In this place, teachers can create and reuse their templates.

REFERENCES

1. David, Jaramillo., Duy, V., Nguyen., Robert, S., Smart. (2016).

Leveraging microservices architecture by using Docker technology. 15.

1. Gail, C., Murphy. (2019). Beyond integrated development environments: adding context to software development. 73-76. doi:

10.1109/ICSE-NIER.2019.00027

000239 Gy. Molnár et al. • Evaluation and Technological Solutions for a Dynamic, Unified Cloud Programming Development Environment

1. Eric, Brewer. (2015). Kubernetes and the path to cloud native. 167167. doi: 10.1145/2806777.2809955
2. Chen, Jianhai., Hou, Wenlong., He, Qinming., Zhang, Miao., Huang, Butian. (2018). Remote memory volume management method and system of Docker container.
3. Pierluigi, Riti. (2018). Pro DevOps with Google Cloud Platform: With Docker, Jenkins, and Kubernetes. doi: 10.1007/978-1-4842-3897-4
4. Éva, Karl., Molnár, György. (2022). IKT-vel támogatott STEM készségek fejlesztésének lehetőségei a tanulók körében. 144-155
5. Kinary, Jangla. (2018). Accelerating Development Velocity Using

Docker: Docker Across Microservices. doi: 10.1007/978-1-4842-

3936-0

1. Mithun, Satheesh., Bruno, Joseph, D’mello., Jason, Krol. (2015). Web Development with MongoDB and NodeJS.
2. Willy, Tarreau. (2015). HAProxy Configuration Manual v1.5.14
3. Stephen, R. Smoot., Nam, K. Tan. (2012). Private Cloud Computing: Consolidation, Virtualization and Service-Oriented Infrastructure.
4. Beáta, Orosz (2021). Learner Experiences Related to Digital Education Schedules in Light of Empirical Data ACTA POLYTECHNICA HUNGARICA 18 : 1 pp. 141-157. , 17 p.
5. Elod Gogh, Reka Racsko, Attila Kovari (2021). Experience of SelfEfficacy Learning among Vocational Secondary School Students, ACTA POLYTECHNICA HUNGARICA 18 : 1 pp. 101-119.
6. József , Katona (2021). Clean and Dirty Code Comprehension by Eyetracking Based Evaluation using GP3 Eye Tracker, ACTA POLYTECHNICA HUNGARICA 18 : 1 pp. 79-99
7. Zoltán Balogh, Kristián Fodor, Martin Magdin, Jan Francisti, Štefan Koprda, Attila Kővári (2022). Development of Available IoT Data Collection Devices to Evaluate Basic Emotions, ACTA POLYTECHNICA HUNGARICA 19 : 11 pp. 165-184
8. Mónika Rajcsányi Molnár. Anetta Bacsa Bán. (2021). Towards Digitalisation Student Experiences in Online Education at a Higher Education Institution, JOURNAL OF APPLIED TECHNICAL AND

EDUCATIONAL SCIENCES, Vol. 11, No. 1, pp. 88-110

000240