Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed
Computing

Distributed Setup for Trackmania Reinforcement Learning

Distributed Systems Laboratory

Pius Kriemler

kpius@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Benjamin Estermann, Luca Lanzendörfer
Prof. Dr. Roger Wattenhofer

July 28, 2023

Abstract

This report focuses on the exploration, description, and evaluation of approaches aimed at achieving scalability for rollout workers in the RL framework tmrl, which hosts the Trackmania Roborace League. The competition offers participants the opportunity to benchmark their vision-based autonomous car racing strategies on Trackmania 2020 through solving Real-Time Gym environments. However, the limitation of rollout workers being only executable on Windows machines hinders scalability and policy improvement.

To address this limitation, we present a solution that enables the deployment of rollout workers on Linux-based machines. By utilizing Lutris, Trackmania is run on Linux, and an extension to the tmrl library allows seamless interaction with the game environment from a Linux system. The successful deployment of rollout workers on Linux opens up new possibilities for scaling and enhancing policy learning in the tmrl framework.

The project is nearly complete, with the establishment of most components required for a fully automated pipeline to run workers on a Linux cluster. The final phase involves integrating the necessary pieces by creating a Docker container that initiates the Xvfb-server, launches Lutris, and sets up Trackmania. Fortunately, the individual scripts for these actions are functional and ready for orchestration within the Docker container.

Furthermore, the tmrl-extension, which enables Linux support, has been submitted as a Pull Request to the main repository. Pending review and integration into the main codebase, this extension will enhance the official project, benefiting other users and potentially increasing participation in the Trackmania Roborace League.

Contents

Abstract **i**

1 Introduction **1**

2 The `tmrl`-Framework **3**

2.1 Architecture 3

2.1.1 `tlspyo` 3

2.1.2 `rtgym` 4

3 Running a Windows OS in a Singularity container **5**

3.1 Attempt 1: Running a Windows OS with QEMU, without KVM 5

3.2 Attempt 2: Running a Windows OS with QEMU, with KVM . . 6

4 Running Lutris in a Docker container **7**

4.1 Setting up Trackmania 2020 with Lutris on a local machine . . . 7

4.1.1 Installing Trackmania 8

4.1.2 Wine and Lutris 8

4.1.3 Installing OpenPlanet 8

4.1.4 Wine runners 9

4.2 Setting up Trackmania 2020 with Lutris on a headless VM 9

4.2.1 Xvfb 9

4.2.2 VNC 10

4.3 Adding a Linux Interface to the `tmrl` library 10

4.3.1 Sending keys to Trackmania 11

4.3.2 Capturing screenshots 11

4.4 Automating the deployment with Docker 11

4.4.1 Starting and setting up Trackmania 12

5 Conclusion and outlook **13**

Bibliography **14**

Chapter 1

Introduction

Trackmania [1] is a popular racing video game series developed by Nadeo and published by Ubisoft. It is known for its unique blend of fast-paced racing, creativity, and community-driven content. Its combination of creative track design and multiplayer competitions has made it a favorite among racing game enthusiasts and casual gamers alike.

The RL framework `tmrl` [2] is the host of the Trackmania Roborace League. The competition offers a way for participants to benchmark their vision-based autonomous car racing strategies on Trackmania 2020. It involves solving Real-Time Gym environments allowing competitors to test their self-racing policies. `tmrl` hereby provides a custom version of the gym [3] framework tailored for real-time applications, enabling seamless interaction with Trackmania for vision-based autonomous car racing strategies. Unfortunately, only very few teams competed so far.

One significant limitation of `tmrl` is that rollout workers (`workers`), responsible for generating experiences for the reinforcement algorithm, can only be executed on full-fledged Windows machines. This restriction arises from Trackmania's native compatibility with Windows operating systems. Consequently, the experience creation process cannot be easily scaled, leading to slower policy improvement. Additionally, deploying `tmrl` on high-performance clusters becomes complicated due to the lack of native Windows support on most clusters. Training a policy to successfully complete a track typically requires approximately two days of continuous learning on a single Windows machine [2].

This report offers valuable insights into the `tmrl` project, focusing on the exploration, description, and evaluation of various approaches aimed at achieving scalability for `workers`.

The final result of this project is a setup that allows the deployment of `workers` on Linux-based machines. Trackmania is run on Linux by utilising Lutris [4]. An extension of the `tmrl` library has been implemented to enable seamless interaction with the game environment from a Linux system. This accomplishment overcomes the previous limitation of running `workers` only on Windows machines and opens up new possibilities for scaling and improving policy learning in the `tmrl` framework. The initial attempts to achieve fully automatic deployment have proven successful through the use of Docker.

Chapter 2

The `tmrl`-Framework

`tmrl` is a python framework designed to facilitate the training of policies using deep Reinforcement Learning (RL) for real-time applications such as Trackmania 2020. The framework offers a scalable setup for running the `tmrl` library, allowing users to train policies by mapping observations to appropriate actions. It supports distributed training with the help of `rtgym` [5] and `tlspyo` [6], allowing users to collect samples locally on one or several computers and train the policies remotely on High-Performance Computing clusters.

In this instance, the observation comprises the four most recent frames, car speed, gear, and rpm. The actions consist of gas, brake and steering angle.

2.1 Architecture

The fundamental architecture comprises three main components:

1. `worker`: This component, also called rollout worker, is responsible for executing and interacting with Trackmania 2020 to collect experiences.
2. `trainer`: The trainer component trains the policy networks with the entries in the replay buffer, and calculates updated weights for the `workers`.
3. `server`: The Server component facilitates communication by sending the `worker'` experiences to the `trainer` and distributing the freshly calculated weights from the `trainer` to the `workers`.

2.1.1 `tlspyo`

The communication is done with tls-python-object (tlspyo) library, that is designed to facilitate the easy and secure transfer of Python objects over a network. It provides a simple API for transferring objects between machines or processes known as Endpoints. Endpoints can be part of one or several groups and connect

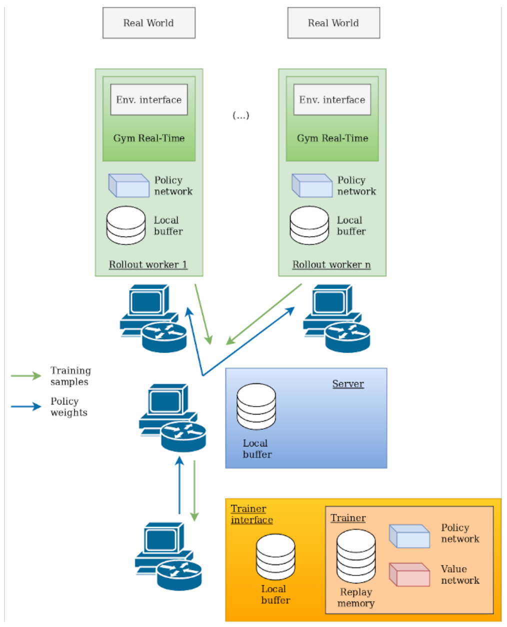

Figure 2.1: Remote training architecture [6]

to a central Relay, the `server` in this case, for communication. It relies on Transport Layer Security (TLS) for secure object transfers over the network, ensuring robust and safe communication

2.1.2 `rtgym`

All the RL related processing is handled by the Real-Time Gym (`rtgym`) library, which is a python framework built on top of Gymnasium [3], designed for real-time implementations of Delayed Markov Decision Processes. It allows users to elastically constrain the times at which actions are sent and observations are retrieved transparently. Users can interact with this environment following the usual Gymnasium pattern, making it compatible with various Reinforcement Learning (RL) frameworks. The core mechanism of Real-Time Gym environments involves elastically constraining time-steps to their nominal duration.

Chapter 3

Running a Windows OS in a Singularity container

The initial solution to address scalability limitations for the `workers` was to utilize a container running the original Windows-based setup proposed from the `tmrl` framework. This approach would re-create the setup used in `tmrl` and has the benefit of being able to use the rest of the `tmrl` pipeline as it is proposed in the repository. It involved running Trackmania on a Windows OS, which was virtualized by QEMU [7] within a Singularity [8] container on a Linux-based cluster.

Singularity was chosen as the containerization method due to its compatibility with the available cluster, instead of using Docker.

The emulator of choice was QEMU [7], as it was already used in the lab and proved to work on the cluster in a Singularity container. Another driver to use QEMU was that the Kernel-based Virtual Machine (KVM) support could be easily enabled and disabled. Enabling KVM brings massive performance benefits by directly leveraging the host operating system's kernel, enabling efficient virtualization and resource utilization for virtual machines. But, kernel access on the cluster is prohibited, as it poses a security risk.

3.1 Attempt 1: Running a Windows OS with QEMU, without KVM

Emulating Windows OS using QEMU without KVM support is technically possible but impractical due to severely degraded performance, making the setup virtually unusable. Enabling KVM was essential to achieve workable performance, reducing the installation process from potentially days to a reasonable timeframe.

3.2 Attempt 2: Running a Windows OS with QEMU, with KVM

Despite being unable to run the setup on the cluster, I proceeded with KVM support to explore a Singularity containerized setup. The VM's performance was excellent, and Trackmania 2020 was successfully installed. However, attempts to run the game resulted in the Windows OS terminating the process with Error Code 0xc0000005, indicating an access violation error [9]. Despite following the official guide for resolution, the issue persisted, and it is suspected that the error may be related to virtualization.

Chapter 4

Running Lutris in a Docker container

Having faced difficulties with emulating the complete Windows OS for running the `tmrl` library's setup, I decided to explore other approaches to run Trackmania on a Linux system. Moreover, I adopted a new strategy, focusing on getting functional components of the solution working swiftly and then incrementally building upon them. This iterative approach allowed for quicker progress and enabled me to tackle challenges more effectively during the development process.

One widely used solution for gaming on Linux is Lutris. Lutris is an open-source gaming platform and launcher specifically designed for Linux users. Its primary goal is to streamline the installation, configuration, and management of video games. With Lutris, gamers can access and play a diverse range of games, including both native Linux games and Windows games through compatibility layers such as Wine and Proton.

4.1 Setting up Trackmania 2020 with Lutris on a local machine

The initial step towards a functional solution on Linux involves ensuring Trackmania 2020 runs seamlessly on a Linux machine. However, running a GUI application on a headless server presents additional challenges, which will be addressed later. To proceed, the focus is on setting up and installing Lutris and Trackmania, a straightforward process. For instance, on Ubuntu systems, Lutris can be directly installed using the package manager apt:

```
apt update && apt install Lutris
```

4.1.1 Installing Trackmania

Once Lutris is set up, Trackmania can be installed through the Ubisoft Connect [10] application. In Lutris, simply install Ubisoft Connect and follow the provided instructions. After installing Ubisoft Connect, launch the application, and it will prompt you to log in to your Ubisoft Connect account. Please note that a free account from Ubisoft Connect is required at this stage. After logging in, you can proceed to install and start Trackmania directly from within the Ubisoft Connect application.

4.1.2 Wine and Lutris

In this subsection, I will provide a brief discussion on how Wine and Lutris works. Wine serves as a compatibility layer facilitating the execution of Windows applications on Unix-like systems, including Linux and macOS. By translating Windows API calls into corresponding Unix-like system calls, Wine enables these non-Windows systems to run Windows executables (EXE files) and utilize Windows Dynamic Link Libraries (DLLs). Additionally, Wine offers emulation for specific Windows behaviors that are not directly compatible with Unix-like systems, emulating Windows registry settings, environment variables, and other components to create a Windows-like environment for seamless application execution. When you install a Windows application through Wine, it creates a new Wine prefix for that application. This prefix contains a virtual Windows file system, registry settings, and other necessary components to make the application work within the Wine environment.

On the other hand, Lutris takes charge of handling the setup and configurations of Wine to facilitate the smooth running of a particular game. Wine, being a somewhat improvised method to make Windows games work on non-Windows systems, often requires a setup that works for each specific game. Once such a setup is found, users can upload and share it on Lutris, making it more accessible and user-friendly. Lutris simplifies the management of Wine and streamlines the process of running Windows games on Unix-like systems.

In this scenario, both Trackmania and Ubisoft Connect coexist within the same Wine prefix, meaning they share the same Windows environment.

4.1.3 Installing OpenPlanet

To obtain crucial information such as the car's speed, current gear, and engine rotations, necessary for creating observations in the reinforcement learning environment, an additional component called OpenPlanet [11] is required. OpenPlanet is a plugin and modding platform for Trackmania that offers additional features and customization options, allowing players and developers to create custom scripts and access game internals. To set up OpenPlanet, you need to manually download the installer EXE and add it to Lutris. Afterward, you can install OpenPlanet using the provided installation wizard, which is thoroughly explained on their website.

4.1.4 Wine runners

While Wine provides a functional compatibility layer, it is not flawless. In addition to the standard Wine version, the community has developed numerous modded Wine-runners to address specific game requirements. Certain games may require a custom-modded runner to function correctly. It is essential to note that experimenting with different runners can often resolve various compatibility issues. Throughout this project, it became evident that trying multiple runners is a valuable troubleshooting approach when encountering problems.

4.2 Setting up Trackmania 2020 with Lutris on a headless VM

On a headless server, graphical user interface (GUI) applications typically require an X server to function properly. The X server is responsible for receiving display requests from the graphical applications and translating them into instructions for the graphics hardware or display driver. It handles tasks such as creating and managing windows, rendering graphical elements, and updating the display as needed. However, on a headless server, there is no physical display or graphics hardware to interact with. To address this, I used a virtual X server called Xvfb (X Virtual Framebuffer).

In a previous attempt, I experimented with X forwarding as a substitute for the missing X server. X forwarding, also known as X11 forwarding, is a lightweight feature in Unix-like systems that enables remote display of GUI applications on a local machine. While it proves efficient for lightweight applications, it fell short in replacing a full X server and lacked the necessary functionality to make Trackmania work effectively.

4.2.1 Xvfb

Xvfb provides a virtual framebuffer in memory, allowing GUI applications to run on the server without the need for a physical display. This setup enables the seamless execution of graphical applications on headless systems.

4.2.2 VNC

Having a virtual X server running enables connecting to it with x11vnc, which is an open-source VNC server for Unix-like systems. It is a technology that allows remote access and control of a computer or server over a network. This enabled me to view the graphical desktop environment and interact with the computer as if I was sitting in front of it. VNC also played a crucial role in setting up and installing Trackmania on the headless server since the GUI interface of Lutris and Ubisoft Connect required interaction.

4.3 Adding a Linux Interface to the `tmrl` library

`tmrl` is built on top of `rtgym`, and to integrate with `rtgym`, you only need to implement interfaces specific to your application, making it compatible with the boilerplate pipeline provided by `rtgym`. In this context, I created a new interface called `TM2020InterfaceLinux`, which is defined in the file `tmrl/custom/custom_gym_interfaces.py`. The required Class and functions are outlined in the pseudo code below:

```
class TM2020InterfaceLinux(RealTimeGymInterface):

    // applies the action given by the policy
    def send_control(self, control):

    // resets the environment and the agent to the initial position
    def reset(self):

    // no-op action for the agent
    def wait(self):

    // returns the observation, the reward, and a terminated signal
    def get_obs_rew_terminated_info(self):

    // returns observation space
    def get_observation_space(self):

    // returns the action space
    def get_action_space(self):

    // initial action at episode start
    def get_default_action(self):
```

In order to appropriately execute the controls given in `send_control` and provide observations when `get_obs_rew_terminated_info` is invoked, supplementary interfaces for sending keys to Trackmania and capturing screenshots were implemented. The code can be found in the files `tmrl/custom/utils/control_keyboard.py` and `tmrl/custom/utils/window.py`, respectively.

4.3.1 Sending keys to Trackmania

To control Trackmania, I used the popular scripting library `xdotool` [12]. `xdotool` is a command-line tool for simulating keyboard input and mouse activity on X11-based systems (Unix-like operating systems with X Window System support). It allows users to automate and script tasks that involve interacting with graphical user interfaces, such as clicking on buttons, typing text, moving the mouse, and more.

It should be emphasized that this approach only permits binary control input. In the Windows version of `tmrl`, the author resolved this by implementing a virtual gamepad, enabling continuous steering inputs in Trackmania. However, in this version, the control is limited to either steering right or not, which doesn't significantly affect the performance of human players and can be expected to be learned by a policy.

4.3.2 Capturing screenshots

The key requirement for taking screenshots is that it is fast, as otherwise the timesteps start to time-out. The python module `fastgrab` [13] is an open-source screen capture package that offers high frame rates and satisfied the speed requirements on a local machine.

4.4 Automating the deployment with Docker

The end-goal was to get an easy scalable `worker`. As previously stated, Singularity was the favored choice for HPC clusters. However, Docker boasted a larger community and a distinct approach to handling permissions inside a container, offering a potentially faster route to containerize Trackmania. Additionally, Singularity possessed a feature that allowed the transformation of any Docker container into a Singularity container. Hence, I made the decision to initially create an automated deployment within a Docker container.

For the *worker* to be operational, Trackmania must be launched and prepared for gameplay, meaning the track is selected and game is ready to play. Once this condition is met, the *worker* can begin gathering experiences by interacting with the game. To achieve this on a freshly started machine, three steps are required: initiating the Xvfb-server, launching Lutris, and starting and setting up Trackmania through Lutris. The first two steps are straightforward when working with a command line interface, while the last step required some scripting.

Trackmania itself runs in the Docker container with the same setup as described above.

4.4.1 Starting and setting up Trackmania

The general process of reaching the desired setup was as follows, assuming that Lutris is opened and everything is installed as described before:

1. Open Ubisoft Connect
2. Within Ubisoft Connect, launch Trackmania
3. Once Trackmania is open, start playing a track

Launch Ubisoft Connect

The window of Lutris exhibits deterministic behavior, allowing it to be opened and repositioned using `xdotool`. This property allowed launching Ubisoft Connect by simulating mouse clicks on specific screen coordinates.

Launch Trackmania

The behavior of the Ubisoft Connect window lacks determinism, leading it to open at different positions and making it resistant to manipulation with `xdotool`, for reasons that are not fully understood. To achieve a consistent window position for the purpose of opening Trackmania with mouse clicks, I utilized computer vision. The script captures a screenshot once Ubisoft Connect is launched and then identifies the location of the top bar of the window by looking for it's unique color. It proceeds to perform a double-click action on that location, causing the window to resize to fullscreen mode. This, in turn, allows for the use of simulated mouse clicks on specific screen coordinates to successfully open Trackmania.

Start playing a track

The Trackmania window still exhibits non-deterministic behavior, but the layout of the menu remains consistent, and it can be controlled using the keyboard. To address the window's non-determinism, a workaround has been found: randomly changing the screen size and then minimizing it consistently leads to the desired minimal window size. This minimal window size is essential as the screenshots taken while playing need to be cropped to the correct dimensions. Once this non-determinism is effectively resolved, starting a track becomes a straightforward process of navigating the menu using the keyboard.

Chapter 5

Conclusion and outlook

The project is currently nearing completion, with most components in place to establish a fully automated pipeline for running `workers` on a Linux cluster. The remaining task involves integrating the final pieces by creating a Docker container that performs the following actions upon startup:

1. Initiates the Xvfb-server
2. Launches Lutris
3. Sets up Trackmania

Fortunately, the individual pieces required for these steps already exist and are functional in the form of separate scripts. The last phase involves combining and orchestrating these scripts within the Docker container to achieve the desired automated pipeline.

The `tmrl`-extension has been submitted as a Pull Request to the main repository. Now, the maintainers of the repository will review the changes and, if everything looks good, they may merge the Pull Request into the main codebase. This will make the `tmrl`-extension a part of the official project, allowing others to benefit from the added functionality and improvements, and hopefully boost the participations in the Trackmania Roborace League.

Bibliography

[1] "Trackmania," accessed: 21.07.23. [Online]. Available: https://www.ubisoft.com/de-de/game/trackmania/trackmania

[2] Y. Bouteiller, "tmrl," https://github.com/trackmania-rl/tmrl, 2020, commit: b142b8e.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[4] "Lutris," accessed: 21.07.23. [Online]. Available: https://lutris.net/

[5] Y. Bouteiller, "rtgym," https://github.com/yannbouteiller/rtgym, 2020, commit: 213df48.

[6] Y. Bouteiller, "tlspyo," https://github.com/MISTLab/tls-python-object, 2022, commit: 743b8f4.

[7] F. Bellard, "Qemu, a fast and portable dynamic translator," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '05. USA: USENIX Association, 2005, p. 41.

[8] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PLOS ONE*, vol. 12, no. 5, pp. 1–20, 05 2017.

[9] "Application error 0xc0000005," accessed: 21.07.23. [Online]. Available: https://answers.microsoft.com/en-us/windows/forum/all/application-error-0xc0000005/6224ae45-a251-4f21-b076-74524618d00a

[10] Ubisoft, "Ubisoft connect," accessed: 21.07.23. [Online]. Available: https://ubisoftconnect.com/de-DE/

[11] "Openplanet," accessed: 21.07.23. [Online]. Available: https://openplanet.dev/

[12] "xdotool," accessed: 21.07.23. [Online]. Available: https://wiki.ubuntuusers.de/xdotool/

[13] M. Herkazandjian, "fastgrab," https://github.com/mherkazandjian/fastgrab, commit: 499ed87.