Towards a Workload Trace Archive for Metaverse Systems

Radu Apșan*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
R.Apsan@student.vu.nl

Damla Ural*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
D.Ural@student.vu.nl

Paul Daniëlse*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
P.E.G.Danielse@student.vu.nl

Vlad-Andrei Cursaru*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
V.Cursaru@student.vu.nl

Eames Trinh*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
E.V.T.Trinh@student.vu.nl

Jesse Donkervliet*
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
J.J.R.Donkervliet@vu.nl

Alexandru Iosup
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
A.Iosup@vu.nl

ABSTRACT

Recent years have seen a resurgence of societal interest in Metaverses and virtual reality (VR), with large companies such as Meta and Apple investing multi-billion dollars into its future. With the recent developments in VR hardware and software, understanding how to operate these systems efficiently and with good performance becomes increasingly important. However, studying Metaverse and VR systems is challenging because publicly available data detailing the performance of these systems is rare. Moreover, collecting this data is labor-intensive because VR devices are end-user devices that are driven by human input. In this work, we address this challenge and work towards a *workload trace archive* for Metaverse systems. To this end, we design, implement, and validate librnr, a system to record and replay human input on VR devices, automating large parts of the process of collecting VR traces. We use librnr to collect 106 traces with a combined runtime of 7 hours from state-of-the-art VR hardware under a variety of representative scenarios. Through analysis of our initial results, we find that power use of VR devices can increase by up to 29% depending on the location of the VR device relative to the user-defined play area, and show that noticeable performance degradation can occur when network bandwidth drops below 100 Mbps. Encouraging community adoption of both librnr and the emerging trace archive, we publish both according to FAIR data principles at https://github.com/atlarge-research/librnr.

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; • **Networks** → **Network measurement**.

*These authors contributed equally to this work.

ICPE '24 Companion, May 7–11, 2024, London, United Kingdom

ACM ISBN 979-8-4007-0445-1/24/05
https://doi.org/10.1145/3629527.3651421

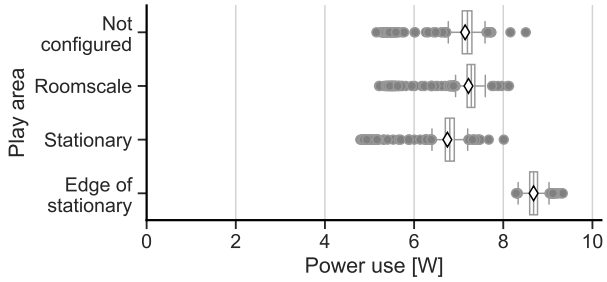

Figure 1: Meta Quest Pro power use when using different play area configurations.

KEYWORDS

metaverse, virtual reality, online gaming, workload traces, performance analysis, real-world experiments, FAIR data, open science

ACM Reference Format:
Radu Apșan, Damla Ural, Paul Daniëlse, Vlad-Andrei Cursaru, Eames Trinh, Jesse Donkervliet, and Alexandru Iosup. 2024. Towards a Workload Trace Archive for Metaverse Systems. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3629527.3651421

1 INTRODUCTION

Over the past decade, there has been an increasing and steady resurgence in interest in the topics of virtual reality (VR) and the *Metaverse*, with prominent examples including Meta's $36 billion investment towards building a metaverse [13], and Apple's entry into the VR market by releasing a $3,500 consumer VR device in February 2024 [4]. The concept of a Metaverse was introduced more than three decades ago [22], and, although community consensus on a definition is still lacking, presents a vision for a novel and fundamentally different way for people to interact with computers and each other. In this vision, many operations that today require a mouse and keyboard are performed using VR devices, which commonly consist of a head-mounted display (HMD) and a pair of hand-held controllers. *If successful, a metaverse holds great potential to benefit society at large across a wide range of application*

domains. For example, a metaverse can benefit medicine [6, 30], mental health [11], construction [2], and law enforcement [20] by improving professional collaborative environments.

For a metaverse to realize its potential and gain large-scale societal adoption, it must provide good performance and energy efficiency under a wide range of use cases. For example, the VR devices that users use to interact with a metaverse have stringent requirements on latency and energy efficiency. Specifically, devices must display a new frame to the user roughly every 14 ms to meet the performance requirement of 72 frames per second, and reduce the probability of motion sickness [25], while providing a battery life in the order of hours.

However, although there has been increasing interest in the topic, understanding the performance and overall behavior of VR systems remains challenging for two important reasons. First, there is a lack of *publicly available traces* that allow researchers to study the behavior of these systems. The collection and analysis of traces is a common and important approach in computer systems to improve understanding of system behavior, and has been successfully applied in a wide range of application domains including cloud computing [7], workflow scheduling [23], and online gaming [12]. Second, collecting traces for VR systems is labor-intensive because VR systems are end-user devices. Similar to smartphones, their realistic workloads consist of applications that respond to human inputs, which are challenging to automate realistically.

Although there exists work that focuses on the performance of metaverse-related technologies, these studies typically focus on the performance of individual subsystems, such as the motion-to-photon latency [27], or propose new techniques to improve performance, for example through computational offloading [8, 29].

In this work, we make an important step towards a *workload trace archive for metaverse systems*, allowing both researchers and developers to better understand the behavior of these state-of-the-art systems. For example, Figure 1 shows a simplified result based on traces collected as part of this work, and shows that the power use of VR device is affected by the play area and VR positioning. To this end, we design librnr, a novel open-source tool that simplifies and partially automates the collection of both user-input and performance-measurement traces. We use librnr to obtain traces from a variety of VR devices and deployment scenarios, and present surprising preliminary results on the behavior of state-of-the-art VR hardware.

Our key contributions are:

- **C1** We design and implement librnr, a novel system to record and replay user-input traces on state-of-the-art VR hardware (Section 3). librnr is designed to be compatible with most state-of-the-art VR applications out of the box, without modifying the application. librnr captures traces in an open format, allowing researchers to analyze user and system behavior.
- **C2** We validate, through real-world experiments, the accuracy and overhead of librnr, and show that librnr can replay human inputs with high accuracy without introducing significant performance overhead (Section 4).
- **C3** We use librnr to bootstrap a workload trace archive for metaverse systems (Section 5). We obtain VR system traces from two popular VR devices under a range of network conditions, while exposed to the same (human input) workload. We analyze our traces and present novel insights into the behavior of VR systems.
- **C4** We publish librnr, and the collected traces, as open-source artifacts according to FAIR research principles, on Github: https://github.com/atlarge-research/librnr/tree/trace_files_and_report/traces.

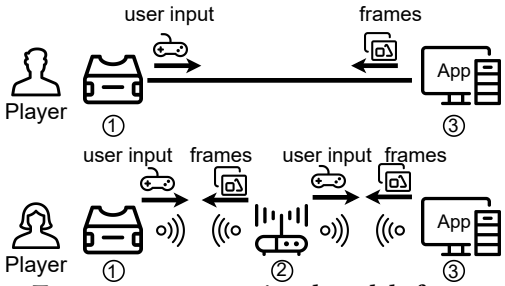

Figure 2: Two common operational models for metaverse applications, linking a user's VR device (left) to a more powerful device (right) for computational offloading. In the two modes, control operations and rendered frames are sent via wire (top) or wirelessly (bottom), respectively.

2 BACKGROUND

In this section, we present a client-centric model for modern metaverse systems and discuss important properties relevant to this work. We present a visual overview of our model in Figure 2.

Our model starts with a user (or player) of the metaverse system, who interacts with the system through a VR device (labeled ① in Figure 2). The VR device typically consists of a HMD and hand-held controllers. The HMD contains displays that show the user a three-dimensional virtual world. The HMD also tracks movements of itself and the hand-held controllers. Together with occasional user button-presses, these control commands (or *controls*) are sent from the VR device to the host.

The host (③) is a machine either close to the user or deployed in a cloud environment and is responsible for simulating and rendering the metaverse application (or *app*), and takes the received controls as inputs for its simulation. The app performs a simulation iteration and renders a new frame at a fixed rate. For VR applications, the frame rate is typically 72 or 90 Hz. The application must consistently produce frames at this rate to maintain good performance. Lower frame rates are noticeable to users and induce motion sickness [25]. Each rendered frame is sent back to the HMD to be displayed to the user, giving the user the illusion of being in a different space.

Depending on how the user sets up their system, controls and frames are sent via a wire or a WiFi router (②). Both deployment methods have advantages and are common. Because the app must respond to user controls in real-time, and frames must be delivered to the HMD at a high rate, data must be exchanged with both low latency (i.e., in the order of milliseconds) and high bandwidth (i.e., in the order of 100 Mbps). When using a wired setup, these requirements are relatively simple to guarantee. When using a WiFi router, meeting these requirements is more challenging, and depends on the quality of the WiFi router of the user. However, a wireless setup

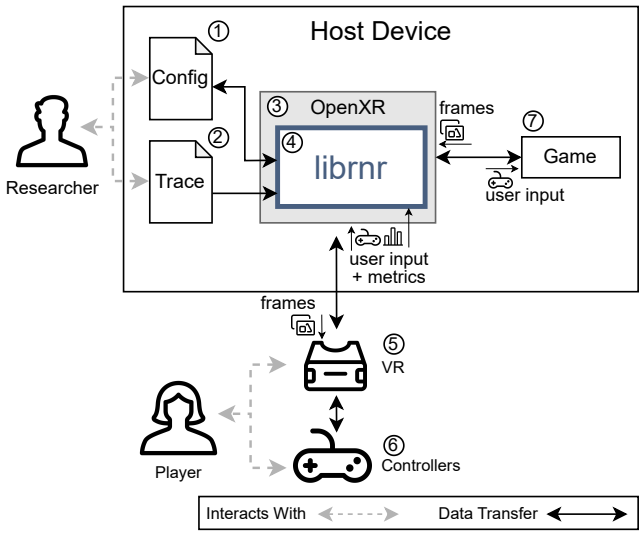

Figure 3: Design overview of librnr.

allows the user to move while using the device without worrying about the length of the cable.

3 DESIGN AND IMPLEMENTATION

In this section, we present the design of librnr. We discuss system requirements and give an overview of its design and implementation.

3.1 System Requirements

We identify here important system requirements for librnr:

- **R1** librnr's system traces must include important system-level metrics for VR applications, allowing users to analyze system behavior under different scenarios.
- **R2** librnr must support a wide range of state-of-the-art VR devices. To construct a workload trace archive that provides a large amount of diverse traces, librnr needs to support a wide variety of VR devices and software applications.
- **R3** While replaying user-input traces, the timing accuracy of events must be in the order of 10s of milliseconds, or up to approximately 10 frames, to ensure the resulting application and system behavior remain the same across replays.
- **R4** librnr should not introduce significant performance overhead. The traces recorded and replayed with librnr should be representative of real-world usage.

3.2 Design and Implementation Overview

Figure 3 presents our design for librnr. We design librnr (④) for two types of users: players and researchers. Players can use librnr to record VR input traces. Doing so only requires them to set librnr to *recording mode* in a configuration file (①). Afterwards, they can interact with the VR device as usual. Applications (⑦) typically obtain user-input information by polling the VR device for the location and orientation of the headset (⑤) and controllers (⑥), and button-press events. When the player starts an application, librnr starts intercepting these calls and writes the values to a trace file.

Researchers can use librnr to replay user-input traces (②) and collect system traces while controlling the environment (e.g., by limiting available network bandwidth) or the system under test (e.g., by using different VR headsets). This allows the researcher to obtain a large amount of system traces through user emulation, automating the most labor-intensive part of obtaining traces for VR systems. When playing back a trace, the researcher sets librnr to *replay mode* through its configuration file and starts the corresponding application. Once the application starts, it will start polling the VR device for user input. However, in replay mode, librnr will overwrite the values obtained from the VR device with values read from the user-input trace, sending previously recorded user inputs to the application.

In both recording and replaying mode, librnr captures important system-level metrics by sampling system performance counters on both the host and VR device at a frequency of 1 Hz (partially addresses Requirement R1). For both devices, these metrics include the information commonly available under the Linux `/proc` filesystem, while for the VR device librnr additionally collects metrics such as the number of frames per second, the amount of time spent on rendering frames, and battery usage data.

We implement librnr as an API layer in OpenXR. We choose OpenXR because it (③) is a modern, open-source standard and API, backed by the majority of major VR device manufacturers, which provides an abstraction layer between applications and the VR device (addresses Requirement R2). Such an abstraction is useful for two important reasons. First, it allows application developers to support a wide variety of VR devices without creating separate implementations of their applications. Second, it allows VR manufacturers to release new VR devices that are backwards-compatible with existing applications without additional engineering effort. OpenXR supports so-called API layers, which are side-loaded libraries that can intercept calls made through the OpenXR API.

4 VALIDATION OF LIBRNR

In this section, we present a preliminary validation of Requirements R3 and R4 from librnr's design (Section 3.1) through real-world experiments. We validate librnr using two setups, PC-A running the Meta Quest Pro (MQP) and PC-B running the Meta Quest 2 (MQ2). For the full experimental setup, see Section 5.1.

Overall, we find:

- **V1** librnr can replay traces with a median delay of 14 ms, or 1 frame, and the delay remains stable over time (Section 4.1). High timing accuracy addresses Requirement R3, and is important for creating reproducible input behavior.
- **V2** librnr does not introduce significant performance overhead (Section 4.2). Low performance-overhead addresses Requirement R4, and is important for collecting representative traces from metaverse systems.

4.1 Replay Timing Accuracy (Requirement R3)

We design librnr to enable researchers to better understand the behavior of different metaverse systems and environments. To this end, it is important that replaying the same input trace results in the same system workload. In this section, we validate this behavior through a real-world experiment in which we compare the timestamps of the events in the recorded trace with those obtained during replay. Overall, we find that librnr's timing is highly accurate, and meets Requirement R3. We visualize this result in Figure 4.

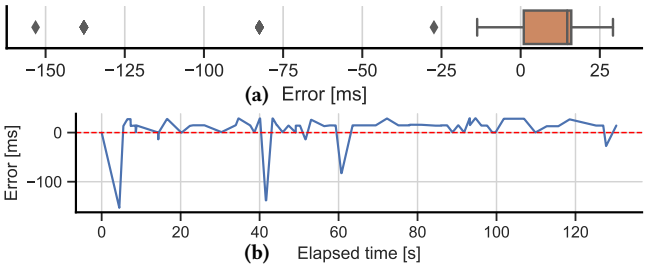

Figure 4: Timing accuracy (i.e., error) of librnr as a statistical summary (top plot) and over time (bottom plot).

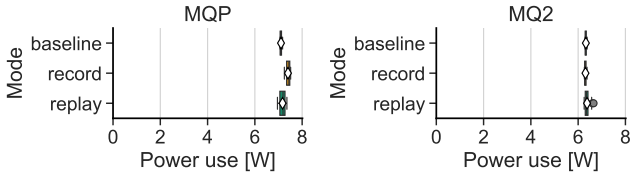

Figure 5: librnr energy overhead on Meta Quest Pro (MQP) and Meta Quest 2 (MQ2).

Figure 4a shows the inaccuracy of timed events while replaying a user-input trace, and reveals that the time difference (i.e., error) is mostly stable between 1 ms and 16 ms, or approximately 0 and 1 frames, based on a frame rate of 72 Hz. The horizontal axis shows the error between when an event was originally recorded, and when it is replayed by librnr. When the error is positive, it indicates that the replay is behind the original recording, and vice versa. Closer to zero is better. The box shows the 25th, 50th, and 75th percentiles, and the whiskers extend up to 1.5 times the inter-quartile range (IQR). The diamonds show all remaining outliers. The plot shows that half of the observed error values (inside the box) have an error between 1 ms and 16 ms, and that the median error value is 15 ms, or 1 frame.

However, the figure shows significant outliers of up to -153 ms, or -11 frames. To investigate these outliers, we visualize the system's behavior over time in Figure 4b. The horizontal axis shows the progression of time during the experiment, the vertical axis shows the error, and the blue curve shows the error over time. The figure shows that the spikes are both rare and isolated events and are caused by the slight error we see throughout the replays.

4.2 System Overhead (Requirement R4)

In this section, we present initial results towards validating the system overhead of librnr on both the VR and host devices using real-world experiments. Although we analyze overhead on power use, GPU utilization, and CPU utilization, we focus in this section on the former two. The results for the CPU utilization are similar to those observed for the GPU utilization.

Figure 5 shows the overhead of power consumption when using librnr with two highly-popular VR devices: the MQP and the MQ2. The figure shows that librnr's power-usage overhead is generally insignificant. The left and right plots in the figure show the power use for the MQP and MQ2 respectively. The horizontal axes show the power use of the VR device, and the vertical axes show the alternative experiment setups. *Baseline* indicates measurements performed without using librnr, whereas *record* and *replay* indicate measurements performed while librnr is recording or replaying a user-input trace, respectively. The boxes summarize the behavior

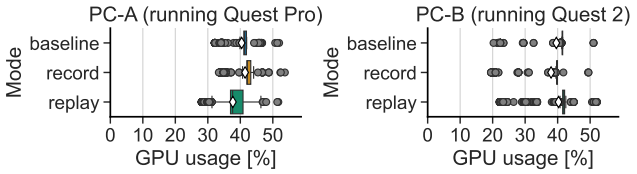

Figure 6: librnr GPU-usage overhead on the two PCs. PC-A is the setup running Windows 11, PC-B is the setup running Windows 10. See §5.1 for exact specifications.

Table 1: Overview of experimental hardware specifications.

Hardware	**PC-A**	**PC-B**
OS	Windows **11**	Windows **10**
CPU	AMD Ryzen 5 7600X	AMD Ryzen 5 7600X
GPU	GeForce RTX **3080**	GeForce RTX **4070**
WiFi	802.11ax	802.11ax
Headsets	**MQ2**	**MQP**
Released	2019	2022
CPU	Snapdragon XR2	Snapdragon XR2+
Battery	3640 mAh	5348 mAh
WiFi	WiFi 6	WiFi 6E

across 6, 6, and 60 traces of approximately 4 minutes per trace for the baseline, record, and replay setups, respectively.

The figure shows that, in most configurations tested, the power use of the device is highly similar. Using librnr on the MQ2, the maximum difference between mean power usage across the baseline, record, and replay setups is 0.07 W. The setups also show similar distributions, with the maximum difference between the minimum and maximum power usage across setups on the MQ2 being 0.17 W.

Similar trends can be observed for the MQP. The maximum mean power usage difference across the three modes is 0.29 W, and the difference between the minimum and maximum power usage across setups on the MQP is 0.36 W.

Figure 6 shows librnr overhead on GPU utilization, and shows that librnr does not incur significant overhead in any of the setups tested. The figures show the GPU utilization when using librnr on the MQP (left) and the MQ2 (right). The horizontal axes show the GPU usage, and the vertical axes show the experiment setup. The boxes show the 25th, 50th, and 75th percentiles, and the box whiskers extend up to 1.5 times the IQR. Grey circles with a black border indicate outliers. From the figure, we observe that the distributions of GPU usage are highly similar across all setups. Specifically, the largest difference between the depicted percentiles is 3.84 percentage points, between 75th and 25th percentiles on the PC-A running MQP, with librnr in replay mode. We also observe a high number of outliers across all setups, ranging from 19.47% to 53.71%.

5 RESULTS

Using the user-input traces collected with librnr, we conduct preliminary real-world experiments on two highly popular VR devices across a variety of environments.

We summarize our experiments in Table 2, and derive the following **M**ain **F**indings:

MF1 VR power use is significantly affected by the location of the VR and the user-defined play area (Section 5.2). Changing

Table 2: Experiment overview.

Section	Experiment	Traces		Bandwidth limit	Devices
		Recordings	Replays	[Mbps]	
§5.2	Effect of play area settings on energy use	1	25	-	PC-B + MQP, PC-B + MQ2
§5.3	Limiting bandwidth effects on VR	2	24	30, 50, 80, 100	PC-A + MQP, PC-B + MQ2
§5.4	VR performance comparison when offloading	4	40	-	PC-A + MQP, PC-A + MQ2

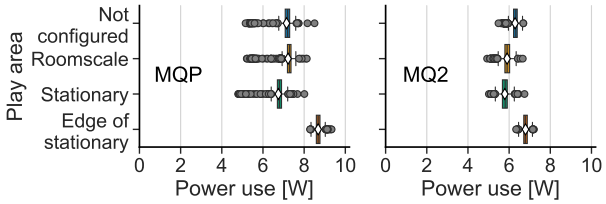

Figure 7: Effect of play area settings on power consumption for the Quest Pro (MQP) and Quest 2 (MQ2) VR devices.

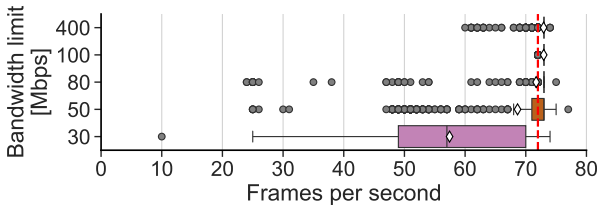

Figure 8: Effect of bandwidth limits on Meta Quest 2 frames per second (FPS). Larger FPS is better, the red dotted line marks the minimum recommended frame rate (i.e. 72 FPS).

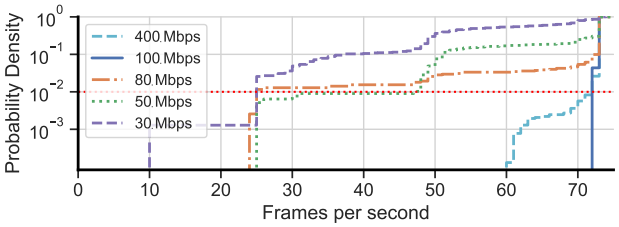

Figure 9: Cumulative probability density function for the frames per second (FPS) on Meta Quest 2. Below the red dotted line is the 99th percentile (i.e., 10^{-2}).

the play area settings and VR location can increase mean power use by up to 29% and 17%, for the MQP and MQ2, respectively.

MF2 Frame rate is significantly affected by available network bandwidth (Section 5.3). For example, user experience significantly deteriorates when using the MQ2 if network bandwidth is limited to 80 Mbps.

MF3 There is no significant difference in performance between the MQP and MQ2 (Section 5.4). Although the MQP is a flagship device and the MQ2 a budget-friendly device, their performance is similar when offloading to a remote machine.

5.1 Experiment Setup

We conduct our experiments using two representative host devices and two popular VR devices, the Meta Quest Pro (MQP), a state-of-the-art VR device designed for professional applications [17], and Meta Quest 2 (MQ2), the best-selling VR device worldwide [5], whose hardware specifications are listed in Table 1. The VR devices are connected to a host device to offload application simulation and rendering. The video frames and user inputs are sent to and from the VR devices through a 5 GHz WiFi router, which provides a bandwidth of up to 1200 Mbps.

We use the application *Beat Saber* as our workload throughout our experiments. Games represent a significant part of the VR market [1], and Beat Saber is one of the best-selling VR games worldwide, with over 4 million copies sold.

5.2 VR Play Area Settings Affect VR Power Use

The play area settings and VR positioning significantly affect the power use of VR devices, increasing the mean power use by up to 29% when the VR is on the edge of the play area. The Oculus software allows two play area modes: stationary mode with a preset circular play area and roomscale mode which allows the user to draw the play area using the controller. Finally, there is the option to disable the play area setting [16] under developer options.

Figure 7 shows the energy consumption for two VR devices, the MQP and MQ2, under the different play area configurations. The horizontal axes show the power use of the device, and the vertical axes show the play area configurations used during the experiment. The boxes show the 25th, 50th, and 75th percentiles, and the white diamonds mark the mean. The lowest mean power use of the MQP is 6.7 W, when using a stationary play area with the VR device placed in the middle. Although changing the play area type to roomscale or turning it off does not significantly affect its power use, placing the VR on the edge of the stationary play area increases its power use by up to 29% to 8.6 W. We see a similar trend on the MQ2, consuming the least average power of 5.8 W for the stationary play area, with the average power increasing by up to 17% when moving the device to the stationary play area edge.

We find that this difference occurs due to the VR device blending the game frames with the live feed from the onboard cameras. This is a safety feature of the device that prevents the user from walking outside the user-determined play area. We recommend that for the most energy-efficient experience, users should remain in the center of the defined play area.

5.3 Limiting Bandwidth Leads to Sudden Performance Degradation

Increasingly limiting the network bandwidth available for streaming video from the host to the VR device leads to sudden performance degradation.

Figure 8 shows this result. The horizontal axis shows the number of frames per second, and the vertical axis shows the bandwidth limits used in the experiment. The boxes show the 25th, 50th, and 75th percentiles, and the white diamond marks the mean. The vertical red-dashed line indicates 72 frames per second, which is the target

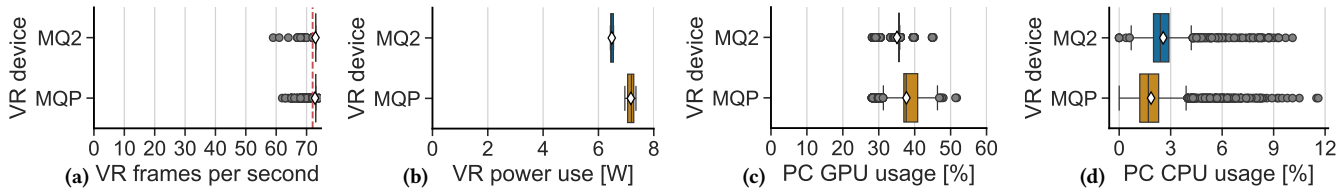

Figure 10: Comparison in workload offloading between Meta Quest 2 (MQ2) and Meta Quest Pro (MQP) using the PC-A setup. See §5.1 for device specifications.

frame rate. The plot shows that for bandwidth limits of 400 Mbps and 100 Mbps, overall performance meets the 72 frames per second (FPS) target, with outliers down to 60 FPS. However, bandwidth limits of 80 Mbps and below introduce significantly lower outliers, down to 10 FPS (!) for a bandwidth limit of 30 Mbps.

Figure 9 presents a reverse cumulative distribution function (CDF) of the same data, allowing us to analyze the tail of the distribution in more detail. The horizontal axis shows the frames per second (FPS), and the horizontal axis shows the fraction of FPS samples. The horizontal red-dashed line indicates the 99th percentile. The 99th percentile is relevant because the system must maintain stable performance over time. The figure shows that for both 400 Mbps and 100 Mbps bandwidth limits, the 99th percentile is 70 FPS and 72 FPS respectively (the corresponding curves intersecting the red dashed line). However, for bandwidth limits of 80 Mbps and below, the performance is significantly worse, with the 99th percentile frame rate being 25 FPS, 25.9 FPS, and 25 FPS for bandwidth limits of 30 Mbps, 50 Mbps, and 80 Mbps respectively. This sudden drop in performance indicates that existing video streaming software for VR systems imposes a clear lower limit on available network bandwidth.

5.4 VR Performance Similar across Devices When Offloading

In this experiment, we compare the performance of the MQP and MQ2. We find that, when offloading the application to a remote machine, the performance of both devices is highly similar. This result is shown in Figure 10.

Figure 10a shows that the performance of both devices is highly similar. The horizontal axis shows the number of frames per second (FPS) displayed to the user, and the vertical axis shows the two VR devices. The vertical red-dashed line shows the performance target of 72 FPS. The plots show that for both the MQ2 and the MQP, the performance is stable at 72 FPS, with outliers down to roughly 60 FPS. Because the frame rates are highly similar, and both devices have roughly the same screen resolution, this indicates that the visual quality on both devices is similar.

Figure 10b shows that the power use of both devices is comparable under this setup. The horizontal axis shows the power use of the VR device, and the vertical axis shows the two VR devices. For both devices, the power usage is stable, with the maximum inter-quartile range (IQR) being 0.2 W, and no outliers outside 1.5 times the IQR. The mean power use is 7.2 W and 6.5 W for the MQP and the MQ2 respectively. Although the MQ2 performs 10% better, we conclude that the energy efficiency of both devices is similar.

Figures 10c and 10d show the GPU and CPU usage of the offloading target (i.e., PC-A). The figures show that the offloading target requires a similar amount of resources to simulate and render the offloaded application. On the horizontal axes, the figures show the GPU and CPU usage respectively. On the vertical axes, the figures show the two VR devices. For both resources, the usage is similar across devices. The GPU usage is 39% and 35% for the MQP and MQ2 respectively, and the CPU usage is 2% and 2.6% for the MQP and MQ2 respectively. The reported CPU usage is the mean across all 12 virtual cores. Based on this result, we conclude that both VR devices have similar requirements for offloading target devices.

6 RELATED WORK

Much related work in the VR research community analyzes device performance under different conditions. Relatively to the body of prior work, we focus on the complex interplay between VR offloading, network conditions, performance, and energy use. Additionally, we publish the collected traces and a novel system to partially automate VR trace collection.

Offloading. Q-VR is a novel system that reduces frame latency by partially offloading frame rendering [26]. Similarly, Danhier *et al.* design and implement a benchmark for video rendering for VR devices using a custom rendering pipeline [9]. Several other studies exist that investigate the impact of computational offloading across the cloud continuum (e.g., differences between edge, local, and cloud) [21, 29]. However, in contrast with this work, these studies do not investigate the effects of network conditions on metrics such as energy use when offloading to a local device.

Network Conditions. There are multiple studies that focus on analyzing the network traffic and usage patterns associated with various VR applications, such as video games [10, 15, 28] and social VR platforms [3]. These studies typically focus on performance [3, 10], user experience [15], and modeling network phenomena [10, 28]. We extend existing work by collecting a broad range of important metrics to characterize the effect that network conditions have on performance, energy use, and overall system behavior.

Energy Efficiency. In the field of energy-efficiency research, studies have investigated the energy use of gaming PCs [18, 19], VR devices [14, 18], cloud gaming [18], and the metaverse [24]. Our work is novel in its evaluation of the interplay between performance, energy use, the system under test, and the deployment environment (e.g., network conditions), and the publication of traces collected in these different scenarios.

7 CONCLUSION AND ONGOING WORK

A metaverse, if successfully implemented, promises to fundamentally change the way people interact with computers. To this end, the industry is investing tens of billions of dollars to develop new VR hardware and software, which are predicted to be the main way

for users to interact with the metaverse. However, understanding the performance of VR devices and their surrounding ecosystem in practice remains challenging due to the lack of publicly available system traces. In this work, we address this challenge by making an important step towards a publicly-accessible workload trace archive for metaverse systems. To this end, we design and implement librnr, a novel tool that partially automates the collection of system traces. Through real-world experiments, we validate librnr and study the behavior of two VR devices: the Meta Quest 2, the best-selling VR device worldwide, and the Meta Quest Pro, a state-of-the-art VR device designed for professional applications. Through our experiments, we find that play area placement can increase VR power usage by up to 1.9 W, and that streaming video to our VR devices with good performance requires at least 80 Mbps of bandwidth. Throughout our experiments, we collect a total of 11 input traces with a total duration of 40 minutes, and 112 system traces with a total duration of 7 hours.

This article is part of ongoing work on a workload trace archive for metaverse systems. To this end, we are working to add support for reproducibility packages to librnr, and are collecting traces for additional (types of) applications, VR devices, and deployments.

ACKNOWLEDGMENTS

This work is funded by a National Growth Fund through the Dutch 6G flagship project "Future Network Services," the projects NWO OffSense (OCENW.KLEIN.209), EU Graph-Massivizer, and CloudStars, and by structural funds from VU Amsterdam. We thank Joshua Offermans for his exploration of synthetically generated user-input traces.

REFERENCES

[1] 2024. Virtual Reality in Gaming Market Size | Global Analysis [2028]. https://www.fortunebusinessinsights.com/industry-reports/virtual-reality-gaming-market-100271 Accessed 2024-02-06.

[2] Pooya Adami, Patrick B. Rodrigues, Peter J. Woods, Burcin Becerik-Gerber, Lucio Soibelman, Yasemin Copur-Gencturk, and Gale M. Lucas. 2021. Effectiveness of VR-based training on improving construction workers' knowledge, skills, and safety behavior in robotic teleoperation. *Adv. Eng. Informatics* 50 (2021), 101431.

[3] Ahmad Alhilal, Kirill A. Shatilov, Gareth Tyson, Tristan Braud, and Pan Hui. 2023. Network Traffic in the Metaverse: The Case of Social VR. In *43rd IEEE International Conference on Distributed Computing Systems, ICDCS 2023 - Workshops, Hong Kong, July 18-21, 2023*. 109–114.

[4] Apple. 2023. Introducing Apple Vision Pro: Apple's first spatial computer. https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro-apples-first-spatial-computer Accessed 2024-01-04.

[5] Martin Armstrong. 2023. Chart: Meta Leads the Way in VR Headsets. https://www.statista.com/chart/29398/vr-headset-kpis/ Accessed 2024-03-06.

[6] Patrick Carnahan, John Moore, Daniel Bainbridge, Gavin Wheeler, Shujie Deng, Kuberan Pushparajah, Elvis C. S. Chen, John M. Simpson, and Terry M. Peters. 2020. Applications of VR medical image visualization to chordal length measurements for cardiac procedures. In *Medical Imaging 2020: Image-Guided Procedures, Robotic Interventions, and Modeling, Houston, TX, USA, February 15-20, 2020*, Vol. 11315. 1131528.

[7] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*. 153–167.

[8] Vittorio Cozzolino, Leonardo Tonetto, Nitinder Mohan, Aaron Yi Ding, and Jörg Ott. 2023. Nimbus: Towards Latency-Energy Efficient Task Offloading for AR Services. *IEEE Trans. Cloud Comput.* 11, 2 (2023), 1530–1545.

[9] Martin Danhier, Karim El Khoury, and Benoît Macq. 2023. An Open-Source Fine-Grained Benchmarking Platform for Wireless Virtual Reality. In *Virtual Reality and Mixed Reality - 20th EuroXR International Conference, EuroXR 2023, Rotterdam, The Netherlands, November 29 - December 1, 2023, Proceedings*, Vol. 14410. 115–121.

[10] Jesse Donkervliet, Matthijs Jansen, Animesh Trivedi, and Alexandru Iosup. 2023. Can My WiFi Handle the Metaverse? A Performance Evaluation Of Meta's Flagship Virtual Reality Hardware. In *Proceedings of the International Conference on Performance Engineering, Coimbra, Portugal, April, 2023*.

[11] Paul M.G. Emmelkamp and Katharina Meyerbröker. 2021. Virtual Reality Therapy in Mental Health. *Annual Review of Clinical Psychology* 17, 1 (2021), 495–519.

[12] Yong Guo and Alexandru Iosup. 2012. The Game Trace Archive. In *11th Annual Workshop on Network and Systems Support for Games, NetGames 2012, Venice, Italy, November 22-23, 2012*. 1–6.

[13] Business Insider. 2022. Charts: Meta's Metaverse Spending Losses, Reality Labs, VR, Mark Zuckerberg. https://www.businessinsider.com/charts-meta-metaverse-spending-losses-reality-labs-vr-mark-zuckerberg-2022-10 Accessed 2024-01-04.

[14] Yue Leng, Jian Huang, Chi-Chun Chen, Qiuyue Sun, and Yuhao Zhu. 2020. Energy-Efficient Video Processing for Virtual Reality. *IEEE Micro* 40, 3 (2020), 30–36.

[15] Yen-Chun Li, Chia-Hsin Hsu, Yu-Chun Lin, and Cheng-Hsin Hsu. 2020. Performance Measurements on a Cloud VR Gaming Platform. In *QoEVMA'20: Proceedings of the 1st Workshop on Quality of Experience (QoE) in Visual Multimedia Applications, Seattle, WA, USA, October 16, 2020*. 37–45.

[16] Meta. 2023. Set up your boundary for Meta Quest. https://www.meta.com/en-gb/help/quest/articles/in-vr-experiences/oculus-features/boundary/ Accessed 2024-03-13.

[17] Meta. 2024. Meta Quest Pro: Premium mixed reality. https://www.meta.com/quest/quest-pro/ Accessed 2024-03-13.

[18] Evan Mills, Norman Bourassa, Leo Rainer, Jimmy Mai, Arman Shehabi, and Nathaniel Mills. 2019. Toward Greener Gaming: Estimating National Energy Use and Energy Efficiency Potential. *Comput. Games J.* 8, 3-4 (2019), 157–178.

[19] Nathaniel Mills and Evan Mills. 2016. Taming the energy use of gaming computers. *Energy Efficiency* 9, 2 (2016), 321–338.

[20] Markus Murtinger, Jakob Carl Uhl, Helmut Schrom-Feiertag, Quynh Nguyen, Birgit Harthum, and Manfred Tscheligi. 2022. Assist the VR Trainer - Real-Time Dashboard and After-Action Review for Police VR Training. In *IEEE International Conference on Metrology for Extended Reality, Artificial Intelligence and Neural Engineering, MetroXRAINE 2022, Rome, Italy, October 26-28, 2022*. 69–74.

[21] Baraka William Nyamtiga, Airlangga Adi Hermawan, Yakub Fahim Luckyarno, Tae-Wook Kim, Deok-Young Jung, Jin Sam Kwak, and Ji-Hoon Yun. 2022. Edge-Computing-Assisted Virtual Reality Computation Offloading: An Empirical Study. *IEEE Access* 10 (2022), 95892–95907.

[22] Neal Stephenson. 1992. *Snow Crash*. Bantam Spectra Books.

[23] Laurens Versluis, Roland Mathá, Sacheendra Talluri, Tim Hegeman, Radu Prodan, Ewa Deelman, and Alexandru Iosup. 2020. The Workflow Trace Archive: Open-Access Data From Public and Private Computing Infrastructures. *IEEE Trans. Parallel Distributed Syst.* 31, 9 (2020), 2170–2184.

[24] Ștefan Vlăduțescu and Georgiana Camelia Stănescu. 2023. Environmental Sustainability of Metaverse: Perspectives from Romanian Developers. *Sustainability* 15, 15 (2023).

[25] Jialin Wang, Rongkai Shi, Wenxuan Zheng, Weijie Xie, Dominic Kao, and Hai-Ning Liang. 2023. Effect of Frame Rate on User Experience, Performance, and Simulator Sickness in Virtual Reality. *IEEE Trans. Vis. Comput. Graph.* 29, 5 (2023), 2478–2488.

[26] Chenhao Xie, Xie Li, Yang Hu, Huwan Peng, Michael B. Taylor, and Shuaiwen Leon Song. 2021. Q-VR: system-level design for future mobile collaborative virtual reality. In *ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021*. 587–599.

[27] Jingbo Zhao, Robert S. Allison, Margarita Vinnikov, and Sion Jennings. 2017. Estimating the motion-to-photon latency in head mounted displays. In *2017 IEEE Virtual Reality, VR 2017, Los Angeles, CA, USA, March 18-22, 2017*. 313–314.

[28] Sihao Zhao, Hatem Abou-zeid, Ramy Atawia, Yoga Suhas Kuruba Manjunath, Akram Bin Sediq, and Xiao-Ping Zhang. 2021. Virtual Reality Gaming on the Cloud: A Reality Check. In *IEEE Global Communications Conference, GLOBECOM 2021, Madrid, Spain, December 7-11, 2021*. 1–6.

[29] Ziehen Zhu, Xianglong Feng, Zhongze Tang, Nan Jiang, Tian Guo, Lisong Xu, and Sheng Wei. 2022. Power-efficient live virtual reality streaming using edge offloading. In *Proceedings of the 32nd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV 2022, Athlone, Ireland, 17 June 2022*. 57–63.

[30] Paul Zikas, Antonis Protopsaltis, Nick Lydatakis, Mike Kentros, Stratos Geronikolakis, Steve Kateros, Manos Kamarianakis, Giannis Evangelou, Achilleas Filippidis, Eleni Grigoriou, Dimitris Angelis, Michail Tamiolakis, Michael Dodis, George Kokiadis, John Petropoulos, Maria Pateraki, and George Papagiannakis. 2023. MAGES 4.0: Accelerating the World's Transition to VR Training and Democratizing the Authoring of the Medical Metaverse. *IEEE Computer Graphics and Applications* 43, 2 (2023), 43–56.