HORIZON EUROPE FRAMEWORK PROGRAMME

CLOUDSTARS

(grant agreement No 101086248)

Cloud Open Source Research Mobility Network

D2.1 Next-gen Cloud Infrastructure Intermediate report

Due date of deliverable: 31-12-2024
Actual submission date: 24-12-2024

Start date of project: 01-01-2023 Duration: 48 months

Summary of the document

Document Type	Report
Dissemination level	Public
State	v1.0
Number of pages	12
WP/Task related to this document	WP2 / T2.1, T2.3, T2.4
WP/Task responsible	Vrije Universiteit Amsterdam (VUA)
Leader	Alexandru Iosup (VUA)
Technical Manager	Sacheendra Talluri (VUA)
Quality Manager	Maciej Malawski (AGH), Ruben Mayber (UBT)
Author(s)	Sacheendra Talluri (VUA), Matthijs Jansen (VUA), Zebin Ren (VUA), Jesse Donkervliet (VUA)
Partner(s) Contributing	VUA
Document ID	CLOUDSTARS_D2.1_Public.pdf
Abstract	Interim report with details on Cloud Infrastructure design and summaries for outcomes of each task.
Keywords	scheduling, offloading, LLM, memory, online games.

History of changes

Version	Date	Author	Summary of changes
0.1	15-11-2024	Sacheendra Talluri (VUA)	Structure and TOC
0.2	30-11-2024	Sacheendra Talluri (VUA), Matthijs Jansen (VUA), Zebin Ren (VUA), Jesse Donkervliet (VUA)	First Draft
0.3	17-12-2024	Sacheendra Talluri (VUA), Alexandru Iosup (VUA)	Revised Draft
1.0	24-12-2024	Sacheendra Talluri (VUA)	Final version

Table of Contents

1 Executive summary

Cloud services are critical to societal processes such as governance, healthcare, finance, science, socialization, and entertainment. In this project, we research and develop the software infrastructure for the next generation of cloud services based on AI and modifiable virtual environments. This software infrastructure enables applications to utilize existing hardware efficiently and take advantage of hardware advances such as heterogeneous GPU clusters, disaggregated memory, and edge computing.

To build this software infrastructure, we have completed 3 secondments, representing 9.6 secondment months (out of 72). All 3 secondments were by researchers from VU Amsterdam. Two secondments were to IBM Ireland in Dublin and one was to Nearby Computing in Barcelona. We summarize the contribution of these secondments below.

Scheduling LLM Inference and Fine-tuning Workloads on Heterogeneous GPU Clusters (T2.1): This secondment from VUA to IBM Ireland focuses on scheduling in large GPU compute clusters. With the rapid pace of improvement of GPUs and the frequent procurement of new GPUs to expand clusters for new AI applications, organizations are faced with the challenge of dealing with clusters filled with heterogeneous GPU configurations. For example, some nodes can have 8 V100 GPUs, others can have 4 H100 GPUs, and so on. Scheduling AI applications in these clusters is challenging as different AI applications have different scaling/speedup behavior depending on the GPU model and configuration. This secondment aims to build a database of the scaling behavior of AI applications and use it to schedule workloads in compute clusters.

LLM Inference from CXL Memory (T2.3): This secondment from VUA to IBM Ireland focuses on LLM inference and fine-tuning from disaggregated memory. Large Language Models (LLMs) have taken the world by storm and made AI advances accessible via services like ChatGPT and Claude. A major bottleneck in LLM inference and fine-tuning is available GPU memory. Current LLMs can only be run interactively on the highest-end GPUs, such as the Nvidia H100, which have expensive on-chip memory. Compute eXpress Link (CXL) is a new technology that aims to make DRAM accessible over the commodity PCIe interconnect. Enabling LLM inference over CXL would pave the way for using LLMs with commodity hardware and make LLMs more accessible. This secondment characterizes the performance of CXL memory and LLM-based applications using CXL memory.

Deploying Online Games and Modifiable Virtual Environments Across the Compute Continuum (T2.4): This secondment from VUA to Nearby Computing focuses on deploying next-gen interactive applications across the compute continuum. Online games is the largest entertainment industry, larger than movies, books, and music combined. Just like the last generation of hardware improvements enabled Instagram and TikTok, the next generation will enable modifiable virtual worlds where users can interact with each other. This is already on its way with Minecraft, a modifiable virtual world for 10 people at a time, being the best-selling online game, and the Meta Quest VR headsets having sold over 100 million units. However, running online worlds on end-user devices such as the Quest VR headsets comes with performance and battery-life limitations. This work looks into offloading parts of the applications from the end-user device to the edge and the cloud.

We also have two more secondments planned for February/March 2025: one from VUA to Nearby Computing on building a digital twin of the compute continuum (T2.4) and another from VUA to IBM Zurich on disaggregated storage (T2.2). We have faced a bit of a setback in WP2 as our tasks did not include AI when the project was designed, but AI-based projects are popular with industrial partners. This is reflected in WP2's low fraction of secondment months used (9.6/72), but we are actively working on improving this.

2 Secondments Description and Progress

2.1 Scheduling LLM Inference and Fine-tuning Workloads on Heterogeneous GPU Clusters

Visitor:	Matthijs Jansen	Involved tasks:	T2.1
Partner:	VUA	Secondment host:	IBM Ireland

Introduction

Cloud providers have been offering an increasing number of services to simplify the use of increasingly complex and performant hardware and software stacks. These offerings range from Infrastructure-as-a-Services (e.g., AWS EC2, Azure Virtual Networking) to Platform-as-a-Service (e.g., AWS SageMaker, IBM Cloud Kubernetes Service) and Software-as-a-Service (e.g., Google Calendar, Microsoft Outlook). Users submit high-level configurations to these services, such as what application to deploy, and service providers determine the remaining configuration parameters, such as where to deploy the application. This setup not only shifts the burden of complexity management away from the user but also grants service providers significant flexibility in managing systems. This flexibility can be used to achieve higher efficiency and improved user experience [1, 2].

To capitalize on configuration flexibility, service providers require tools and knowledge to reason about which configurations optimize service performance. In this internship at IBM, we research managed machine learning services to assess whether a scheduling policy augmented with knowledge of job configurations improves service performance. We identify 4 levels of knowledge existing scheduling policies possess. First, uninformed policies that rely on the user to supply a complete job configuration. These policies are the default in systems such as SLURM and execute supplied configurations as is. Second, online-informed policies that use runtime information to guide configuration tuning. Examples include the default Kubernetes scheduling policy, which uses resource utilization metrics to guide job placements. Runtime information can also be used continuously to, for example, scale job deployments in response to workload demand. Third, offline-informed policies that use prior knowledge or pre-trained models to guide configuration tuning. Fourth, hybrid policies that use knowledge from offline and online sources to tune configurations. These policies benefit from extensive offline knowledge and adapt to developments at runtime. While schedulers with different levels of knowledge have been extensively compared in the past, those studies focus on the execution of the policies, not on the effect of knowledge on the execution.

Contribution

1. We survey configuration knowledge use in state-of-the-art and state-of-the-practice scheduling policies, analyze their impact on scheduling performance, and design a scheduler knowledge classification.

2. We design a scheduling policy benchmark suite that implements scheduling policies with various levels of knowledge.

3. We evaluate the impact of configuration knowledge on scheduling performance in production systems. Using the evaluation results on real-world systems, we design, implement, and calibrate a simulator to expand our exploration capabilities.

Results

Based on the classification of configuration knowledge in uninformed, online-informed, offline-informed, and hybrid policies, we design and implement a suite of 6 scheduling policies based on the FIFO and Sia schedulers [3]. For FIFO, we implement an uninformed policy (FIFO), three offline-informed policies that augment FIFO (db-fifo), use bin-packing to compare configurations between jobs (db-flat) and optimize for total throughput instead of efficiency (db-flat-scale). For the Sia-based policies, we implement an online-informed version (sia-unaware) and a hybrid-informed version, augmented with knowledge on viable configurations only (sia-aware).

We simulate the performance of these policies using workload from the ACME LLM fine-tuning workload trace [4]. The trace captures the execution of a wide variety of machine learning workloads,

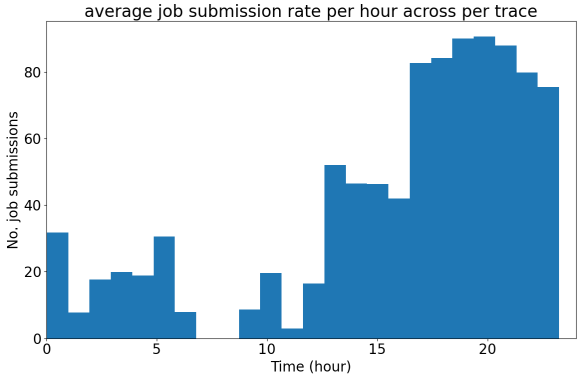


Figure 1: Job arrival distribution.

Figure 2: Job completion time distribution.

including training, fine-tuning, and inference, across multiple months and clusters. We pick the fine-tuning workloads from the trace and sample 10 new traces from the busiest 24-hours. Each trace contains 40 jobs per hour, or 960 jobs in total, and follows a job arrival distribution shown in Figure 1.

We show our preliminary results as a CDF of job completion time across all 10 traces in Figure 2. We observe that the only uninformed policy, FIFO, achieves the worst performance with a much higher job completion time for most executed jobs. The performance gap between informed and uninformed policies shows the effectiveness of configuration knowledge on scheduling performance.

The offline-informed db-flat-scale policy achieves the lowest job completion time for most jobs. It is only outperformed in tail latency by the tail-latency-focused db-fifo policy. Compared to the online and partial-hybrid-informed Sia policies, we show that configuration performance knowledge is vital for optimizing job throughput and job completion time, and less information results in sub-optimal results.

Further visits/Collaborations

We have developed a configuration knowledge exploration scheduler and are wrapping up the design and implementation of the policy suite. We are working towards implementing the framework for physical deployment inside IBM. This is the final development stage and will be followed by writing an article about our findings.

2.2 LLM Inference from CXL Memory

Visitor:	Zebin Ren	Involved tasks:	T2.3
Partner:	VUA	Secondment host:	IBM Ireland

Introduction

Modern applications have increased the demand for memory capacity, hitting the memory wall [5, 6]. Various technologies have been proposed to address the memory wall, such as persistent memory [7], memory disaggregation [8], and compute express link (CXL) [9]. Compute Expressed Link is a potential solution to break this memory wall and has drawn much attention in both industry [10, 11, 12] and academia [13, 14, 15, 16]. CXL memory provides a cache-coherent memory interface built on PCIe, which can be accessed by the CPU with load and store instructions. Existing CXL memory expansion cards can support up to 4 TiB memory with a single PCIe*16 slot and disaggregated CXL memory box supporting up to 16 TiB of memory [17].

Large Language Model (LLM) inference is one kind of application that needs a large amount of memory. For example, the Llama 3.1 [18] 405B model inference needs 810 GiB memory to store the model with FP16 precision for inference. This is larger than a DGX H100 node (8*H100) which contains 640 GiB GPU memory [19]. To reduce the GPU memory usage, many solutions have been proposed to offload GPU memory usage to host memory, such as model offloading [20, 21], KV-cache offloading [22], or caching models in multi-tenant environments for fast loading [23, 24]. These offloading solutions are based on host memory, leading to higher host memory usage, making CXL

memory a good fit for LLM inference workloads. However, CXL memory can not serve as a drop-in solution since it has higher latency and lower bandwidth than the host memory [16, 15], inducing higher overhead and creating multiple challenges in deploying LLM inference workloads with CXL memory. These challenges come from three aspects that we cover in this study: (1) higher latency and lower bandwidth of CXL memory than host memory; (2) lack of understanding of how this lower performance affects the LLM inference workloads specifically; and (3) challenges in mitigating this negative effect on LLM inference workloads.

Firstly, existing performance benchmarks have not studied the performance difference between host and CXL memory with the unique properties of ML inference workloads. Existing studies focus on the latency and bandwidth between CXL memory and CPU. However, the primary workload for LLM inference with GPU memory offloading is moving data between GPU and host memory. As a result, it leads to an incomplete view of the performance characteristics of CXL memory. Secondly, there is a lack of understanding of how this performance difference between host and CXL memory affects the performance of existing LLM inference with GPU memory offloading. The existence of GPU and the extra data path between GPU and memory make the performance of LLM inference workload on CXL memory harder to predict. Thirdly, existing solutions for mitigating the higher latency and lower bandwidth of CXL focus on page placement strategies that move 'cold' memory pages to CXL memory [13, 15]. However, this data movement between the host and CXL memory interferes with the offloading workload that moves data between memory and GPU.

These solutions neither take the predictability of ML inference workloads into consideration, leading to sub-optimal performance. In summary, the large memory capacity but lower performance motivates us to analyze the performance of ML inference with CXL and mitigate the negative effect of this lower CXL memory performance for the LLM inference workloads.

Contribution

1. We characterize the performance of CXL memory to find their performance difference with main memory.
2. We explore the performance of LLM offloading workloads on main memory and CXL memory in different configurations with different models.
3. We provide guidelines on how to efficiently utilize CXL memory expanders with LLM workloads to mitigate their high memory cost.

Results

Our benchmarking environment is a two-socket server equipped with two AMD EPYC 9654 CPUs, each CPU has 96 cores. We disable hyperthreading and boost for stable performance. Each socket has two 64 GiB DDR5 4800 MT/s memory. There are two 64 GiB CXL memory expanders, and a NVIDIA A10 installed on socket 0.

We have finished the microbenchmarks on the performance of CXL memory and part of the macrobenchmarks of characterizing the performance of CXL memory with LLM workloads with CPU memory offloading.

Figure 3 shows the latency and bandwidth of main memory and CXL memory. We evaluate the performance of both memories when they are in the same system and when they are accessed over the network (remote). The left figure shows that there is a significant difference in latency between different memory configurations, while local memory has the lowest latency, remote memory access adds 73.5% latency. CXL memory has 154.3% higher latency than local memory and also has a higher latency if the CXL memory is connected to a remote socket (306.2 ns vs. 390.8 ns).

The right figure shows the bandwidth of different memory configurations. The bandwidth of main memory is higher than CXL memory (70 GiB/s vs. 51 GiB/s). However, the NUMA effect does not significantly affect the bandwidth.

As the main workload of LLM CPU memory offloading is moving data between CPU and GPU memory, we measure the cudaMemCpy bandwidth between GPU memory and different CPU memory configurations. There are two kinds of memories in cudaMemCpy: pageable memory and pinned

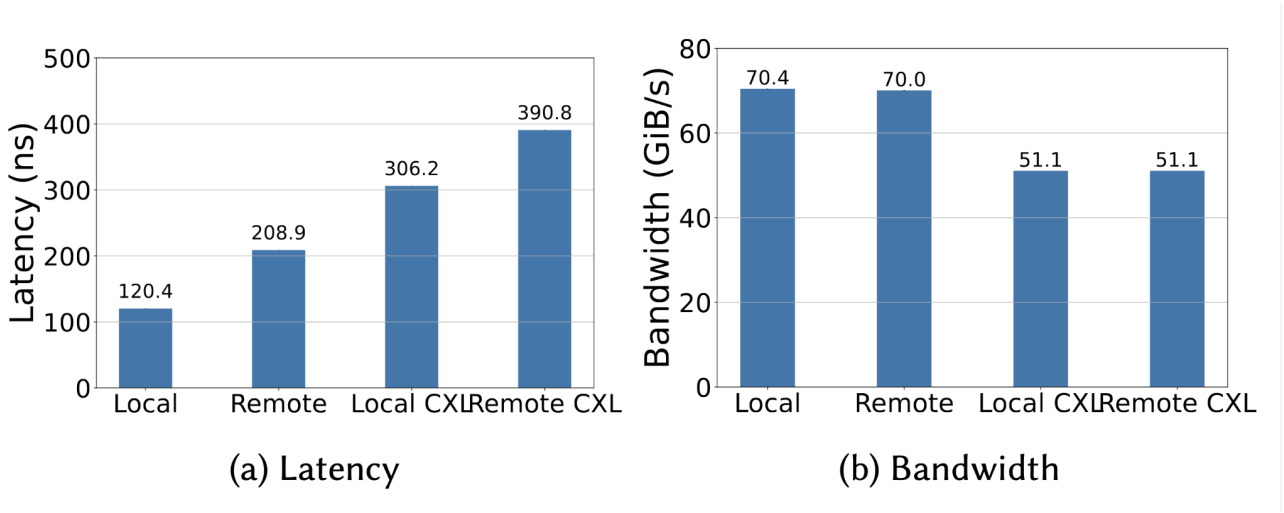


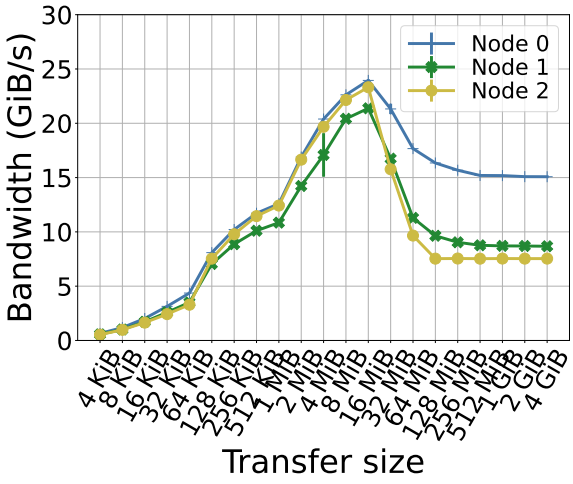
Figure 3: Main memory and CXL memory characteristics.

memory. Pageable memory can be swapped to disk when there is not enough CPU memory, while pinned memory can not be swapped out. However, the data transfer between CPU and GPU memory can only be done between pinned memory and GPU memory. Thus, when using pageable memory, cudaMemCpy utilizes an internal pinned buffer. The data in the pageable memory is first copied to the pinned buffer and then transferred to the GPU memory.

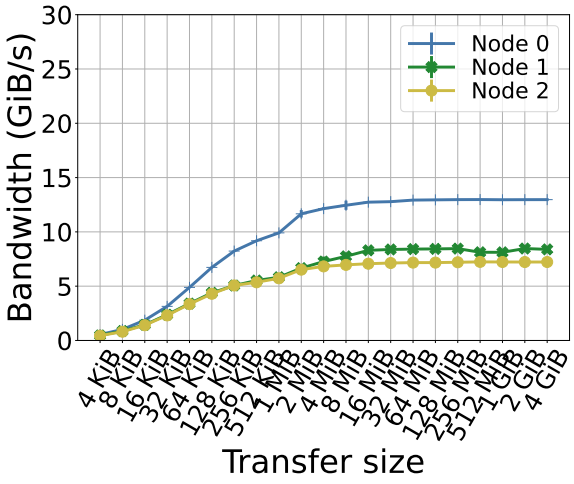
Figure 4 depicts the cudaMemCpy bandwidth on the y-axis with an increasing buffer size on the axis between different CPU memory and GPU memory. We use numactl to pin the CPU memory in a specific NUMA node. H2D means moving data from host to device, and D2H is the reverse direction. Figure 4a shows the host-to-device bandwidth for pageable CPU memory when the processes are pinned on NUMA node 0. As the buffer size increases, the cudaMemCpy bandwidth first increases and then decreases; we identify this is caused by the cache effect. When the buffer can be cached in the L3 cache, the CPU does not need to fetch the data from the memory. As we increase the buffer size, the bandwidth becomes stable. At the stable state, the local memory has the highest bandwidth, and CXL memory has the lowest bandwidth. When moving data from GPU memory to CPU memory Figure 4b, the GPU memory can not be cached in the CPU L3 cache, thus increasing the buffer size leads to an increase of bandwidth until it reaches the maximum bandwidth, we have similar observations when the bandwidth becomes stable. O-9: For pageable memory, local memory has the highest bandwidth, and CXL memory has the lowest bandwidth O-10: When the memory is pinned on CPU (Figure 4c and Figure 4d), both the local CPU memory, remote CPU memory, and CXL memory have comparable performance.

Further visits/Collaborations

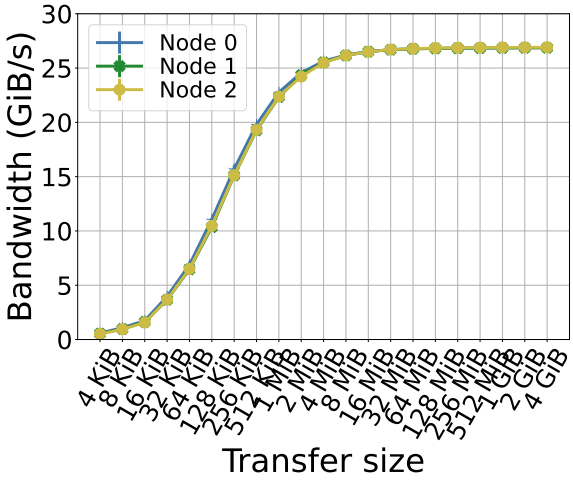
Currently, we have measured the performance of CLX memory with microbenchmarks and simple LLM inference workloads. The next steps have two targets (1) measure the performance of CXL memory expanders with more complex LLM inference workloads and (2) make a simulator to investigate how the performance will change with different hardware configurations, for example, with different numbers of channels of main memory or CXL memory expanders.



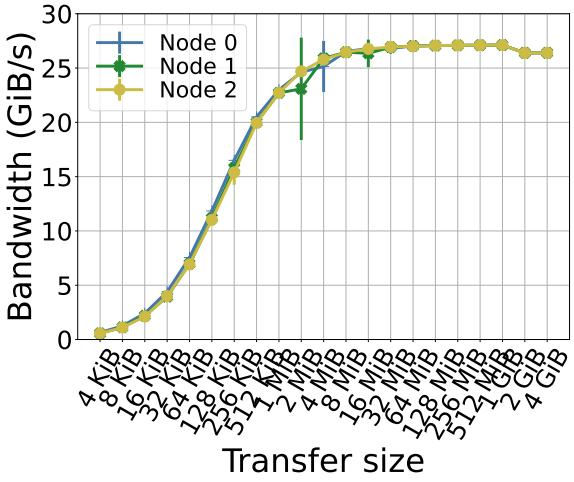
(a) Pageable, H2D



(b) Pageable, D2H



(c) Pinned, H2D



(d) Pinned, D2H

Figure 4: Bandwidth characteristics of different memory types.

2.3 Deploying Online Games and Modifiable Virtual Environments Across the Compute Continuum

Visitor:	Jesse Donkervliet	Involved tasks:	T2.4
Partner:	VUA	Secondment host:	Nearby Computing

Introduction

Video games, which already form the world's largest entertainment industry, are increasingly being used for societally and economically important tasks. There is resurging interest in the creation of metaverses, which promise to integrate a wide range of (social) activities into virtual worlds built on game-like technologies that allow users to modify the virtual world. However, for games and metaverses to become ubiquitous, we must overcome the challenge of delivering good and stable performance under highly varying workloads [25], on resource-constrained devices and networks that sometimes offer poor connectivity. Failing to do so leads users to lose their immersion into the game or metaverse world, which causes loss of enjoyment and decreased results, toxic behavior, and even quitting.

Early approaches address this complex challenge by deploying only the most latency-sensitive services closer to end-users or by specializing deployment of entire game servers to one class of devices in the digital continuum, that is, using only one of cloud, edge, or local (endpoint) devices [26]. Complementing this seminal body of prior work, in this work we address the challenge through differentiated deployment, a first-of-its-kind technique for dynamic, fine-grained, application-level service deployment for virtual worlds, metaverses, and, more generally, modifiable virtual environments.

Gaming technologies are the largest form of entertainment worldwide and likely to further integrate with societal activities. The industry generated $184 billion in revenue in 2023, more than the music and movie industries combined, and includes 3.38 billion users. In part due to the emergence of more complex and expressive virtual worlds over the past decade, e.g., MVEs such as Minecraft, games are increasingly used for purposes other than entertainment. For example, Minecraft, the best-selling game of all time allows players to modify the environment with fine granularity, enabling its use for socialization, education, and activism. These activities, converging with video-streaming and virtual reality devices techniques for collaborative work, are expected to form the basis of metaverses, further integrating societal everyday activities into virtual worlds [27].

An important factor in the large-scale adoption of virtual game worlds for everyday tasks is their ability to perform well on a wide range of user devices and networks. In virtual worlds with hundreds of thousands of concurrent users, where tens to thousands of users closely engage in the same world instance, at least some users will have resource-constrained devices, and at least some of them will experience network issues such as low bandwidth and jittery latency. A metaverse should allow people to access it while traveling, which commonly introduces intermittent connectivity issues up to unavailability, and performance variability due to more limited devices acting as bottlenecks. Mobile users can also change their resource availability quickly, for example, available resources increase when docking a Nintendo Switch or Steam Deck, and portable devices have different performance profiles if they are battery-powered vs. when connected to a charger. MVE experience highly varying load, both shortly because of user activity and (more recently) also non-player workloads, and over longer periods of time as the popularity of games can vary highly, e.g., double or halve, over a matter of days or weeks. These requirements do not match the traditional design of commercial virtual worlds, which are designed as monolithic applications to be deployed on specialized hardware such as cloud or locally placed graphics processing units (GPUs), or purpose-built game consoles, with adaptive techniques based on largely static configurations. Although state-of-the-art research has focused mostly on exploring trade-offs when deploying virtual worlds across the cloud-edge continuum, reducing bandwidth requirements, or designing virtual worlds that can be used as research platforms, related work does not consider performance variability from both workload and resources for complex games, metaverses, and more generally MVEs; this is the focus of our work.

Contribution

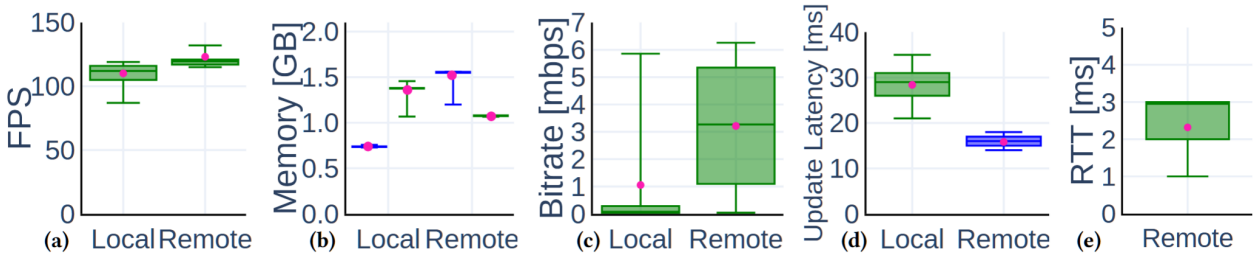


Figure 5: Game offload performance characteristics.

1. We design the first system to address performance variability in online games by dynamically redeploying online game components, and can switch dynamically between a traditional client-server deployment (rendering happens on the user device) to a stream-gaming deployment (rendering happens in the cloud or on the edge). This includes the design of a policy-based scheduler than operates using a flexible approach to improve virtual-world service quality by dynamically adapting to resource availability, and a distributed performance monitoring technique to facilitate performance analysis and automated decision making.

2. We implement a prototype of our design, which includes a modifiable virtual environment, and an experiment framework that automates cloud deployments and metric collection.

3. We perform real-world, large-scale experiments using our prototype. From these experiments, we derive several insights into the efficacy of differentiated deployment.

4. Following open-science and reproducibility principles, we publish Findable, Accessible, Interoperable, and Reusable (FAIR) data, and FOSS artifacts, available on GitHub.

Results

Preliminary results depicted in Figure 5 show that the deployment method significantly affects the behavior and performance of the application, and indicate that our approach is promising. Specifically, The local node (green) experiences significant performance variability before switching to remote rendering with a range of 32 FPS. After the migration, average FPS is increased by 13 FPS and variability is reduced to a range of 16 FPS. Median memory usage drops from 1.36GB to 1.07GB. However, average bandwidth use increases from 1.05mbps to 3.21mbps, in line with previous work demonstrating the high bandwidth of cloud gaming. Surprisingly, the median game RTT of cloud node (blue) post-migration is 12.56ms lower than the local node pre-migration. The ability to dynamically switch to a cloud or remote gaming deployment with minimal interruption is promising. Mobile devices could switch to conserve battery life, or virtual reality (VR) devices could switch between local and streamed rendering based on game environment complexity.

Further visits/Collaborations

Compared to previous work, we designed an automated and repeatable experiment framework, used our preliminary results to improve our design and prototype, and have implemented a significant part of these improvements. We are now finishing the prototype implementation and are designing an extensive suite of experiments to be conducted using our framework, which we will conduct on representative cloud infrastructures. We will combine the outcome of these efforts into an article to be published in the first half of 2025.

3 Conclusions

This report summarizes the secondments performed in Work Package 2 on Next-gen Cloud Infrastructure. The secondments have enabled emerging applications such as AI and modifiable virtual environments to take advantage of hardware advances in GPUs, memory, and edge computing.

Task 2.1 focused on scheduling LLM workloads in heterogeneous GPU clusters. Task 2.3 focused on enabling memory-constrained LLMs to run on disaggregated CXL memory. Task 2.4 focused on offloading modifiable virtual worlds from end-user devices to the edge. These tasks were performed at IBM Ireland and Nearby Computing.

We expect more secondments in the future to IBM Zurich, IBM Ireland, and Nearby Computing. We are working with partners in the project to align secondments with the current interest in infrastructure for AI, in addition to the planned work on cloud infrastructure.

References

[1] M. Copik, G. Kwasniewski, M. Besta, M. Podstawski, and T. Hoefler, "Sebs: a serverless benchmark suite for function-as-a-service computing," in Middleware '21: 22nd International Middleware Conference, Québec City, Canada, December 6 - 10, 2021 (K. Zhang, A. Gherbi, N. Venkatasubramanian, and L. Veiga, eds.), pp. 64–78, ACM, 2021.

[2] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann, "What are you paying for? performance benchmarking for infrastructure-as-a-service offerings," in IEEE International Conference on Cloud Computing, CLOUD 2011, Washington, DC, USA, 4-9 July, 2011 (L. Liu and M. Parashar, eds.), pp. 484–491, IEEE Computer Society, 2011.

[3] S. J. Subramanya, D. Arfeen, S. Lin, A. Qiao, Z. Jia, and G. R. Ganger, "Sia: Heterogeneity-aware, goodput-optimized ml-cluster scheduling," in Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023 (J. Flinn, M. I. Seltzer, P. Druschel, A. Kaufmann, and J. Mace, eds.), pp. 642–657, ACM, 2023.

[4] Q. Hu, Z. Ye, Z. Wang, G. Wang, M. Zhang, Q. Chen, P. Sun, D. Lin, X. Wang, Y. Luo, Y. Wen, and T. Zhang, "Characterization of large language model development in the datacenter," in 21st USENIX Symposium on Networked Systems Design and Implementation, NSDI 2024, Santa Clara, CA, April 15-17, 2024 (L. Vanbever and I. Zhang, eds.), pp. 709–729, USENIX Association, 2024.

[5] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, "Ai and memory wall," IEEE Micro, vol. 44, no. 3, pp. 33–39, 2024.

[6] M. Rhu, "Memory-centric computing systems: What's old is new again." https://www.sigarch.org/memory-centric-computing-systems-whats-old-is-new-again/, Accessed: 2024-09-03.

[7] "Discover advanced memory with intel® optane™ pmem." https://www.intel.com/content/www/us/en/products/details/memory-storage/optane-dc-persistent-memory.html, Accessed: 2024-09-03.

[8] K. T. Lim, J. Chang, T. N. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated memory for expansion and sharing in blade servers," in 36th International Symposium on Computer Architecture (ISCA 2009), June 20-24, 2009, Austin, TX, USA (S. W. Keckler and L. A. Barroso, eds.), pp. 267–278, ACM, 2009.

[9] "Compute express link." https://computeexpresslink.org/about-cxl/, Accessed: 2024-09-03.

[10] "Compute express link (cxl) fpga ip." https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/cxl-ip.html, Accessed: 2024-09-03.

[11] "Panmnesia." https://panmnesia.com/.

[12] "Cxl memory module - box (cmm-b)." https://semiconductor.samsung.com/news-events/tech-blog/cxl-memory-module-box-cmm-b/, Accessed: 2024-09-03.

[13] H. A. Maruf, H. Wang, A. Dhanotia, J. Weiner, N. Agarwal, P. Bhattacharya, C. Petersen, M. Chowdhury, S. O. Kanaujia, and P. Chauhan, "TPP: transparent page placement for cxl-enabled tiered-memory," in Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS 2023, Vancouver, BC, Canada, March 25-29, 2023 (T. M. Aamodt, N. D. E. Jerger, and M. M. Swift, eds.), pp. 742–755, ACM, 2023.

[14] H. Li, D. S. Berger, L. Hsu, D. Ernst, P. Zardoshti, S. Novakovic, M. Shah, S. Rajadnya, S. Lee, I. Agarwal, M. D. Hill, M. Fontoura, and R. Bianchini, "Pond: Cxl-based memory pooling systems for cloud platforms," in Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2023, Vancouver, BC, Canada, March 25-29, 2023 (T. M. Aamodt, N. D. E. Jerger, and M. M. Swift, eds.), pp. 574–587, ACM, 2023.

[15] Y. Sun, Y. Yuan, Z. Yu, R. Kuper, C. Song, J. Huang, H. Ji, S. Agarwal, J. Lou, I. Jeong, R. Wang, J. H. Ahn, T. Xu, and N. S. Kim, "Demystifying CXL memory with genuine cxl-ready systems and devices," in Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2023, Toronto, ON, Canada, 28 October 2023 - 1 November 2023, pp. 105–121, ACM, 2023.

[16] Y. Tang, P. Zhou, W. Zhang, H. Hu, Q. Yang, H. Xiang, T. Liu, J. Shan, R. Huang, C. Zhao, C. Chen, H. Zhang, F. Liu, S. Zhang, X. Ding, and J. Chen, "Exploring performance and cost optimization with asic-based CXL memory," in Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024, Athens, Greece, April 22-25, 2024, pp. 818–833, ACM, 2024.

[17] "Samsung unveils cxl memory module box: Up to 16 tb at 60 gb/s." https://www.anandtech.com/show/21333/samsung-unveils-cxl-memory-module-box-up-to-16-tb-at-60-gbs, Accessed: 2024-09-03.

[18] "Llama 3.1 - 405b, 70b & 8b with multilinguality and long context." https://huggingface.co/blog/llama31, Accessed: 2024-09-03.

[19] "Nvidia dgx h100." https://www.nvidia.com/en-gb/data-center/dgx-h100/, Accessed: 2024-09-03.

[20] R. Y. Aminabadi, S. Rajbhandari, A. A. Awan, C. Li, D. Li, E. Zheng, O. Ruwase, S. Smith, M. Zhang, J. Rasley, and Y. He, "Deepspeed-inference: Enabling efficient inference of transformer models at unprecedented scale," in SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX, USA, November 13-18, 2022 (F. Wolf, S. Shende, C. Culhane, S. R. Alam, and H. Jagode, eds.), pp. 46:1–46:15, IEEE, 2022.

[21] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, B. Chen, P. Liang, C. Ré, I. Stoica, and C. Zhang, "Flexgen: High-throughput generative inference of large language models with a single GPU," in International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds.), vol. 202 of Proceedings of Machine Learning Research, pp. 31094–31116, PMLR, 2023.

[22] W. Lee, J. Lee, J. Seo, and J. Sim, "Infinigen: Efficient generative inference of large language models with dynamic KV cache management," in 18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024 (A. Gavrilovska and D. B. Terry, eds.), pp. 155–172, USENIX Association, 2024.

[23] J. Jeong, S. Baek, and J. Ahn, "Fast and efficient model serving using multi-gpus with direct-host-access," in Proceedings of the Eighteenth European Conference on Computer Systems, EuroSys 2023, Rome, Italy, May 8-12, 2023 (G. A. D. Luna, L. Querzoni, A. Fedorova, and D. Narayanan, eds.), pp. 249–265, ACM, 2023.

[24] Y. Fu, L. Xue, Y. Huang, A. Brabete, D. Ustiugov, Y. Patel, and L. Mai, "Serverlessllm: Low-latency serverless inference for large language models," in 18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024 (A. Gavrilovska and D. B. Terry, eds.), pp. 135–153, USENIX Association, 2024.

[25] J. Eickhoff, J. Donkervliet, and A. Iosup, "Meterstick: Benchmarking performance variability in cloud and self-hosted minecraft-like games," in Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering, ICPE 2023, Coimbra, Portugal, April 15-19, 2023 (M. Vieira, V. Cardellini, A. D. Marco, and P. Tuma, eds.), pp. 173–185, ACM, 2023.

[26] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, "Towards efficient edge cloud augmentation for virtual reality mmogs," in Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose / Silicon Valley, SEC 2017, CA, USA, October 12-14, 2017 (J. Zhang, M. Chiang, and B. M. Maggs, eds.), pp. 8:1–8:14, ACM, 2017.

[27] M. Ball, The Metaverse: Fully Revised and Updated Edition: Building the Spatial Internet. Liveright, 2024.