MINISTERUL EDUCAŢIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul Ingineria Software și Automatică

Admis la susţinere
Şef departament:
FIODOROV Ion dr., conf.univ.

„___” ________ 2024

Motor de randare a graficii distribuite

Proiect de master

Student:	_____________	**Filipescu Mihail, IS-221M**
Coordonator:	_____________	**Poștaru Andrei, asist. univ.**
Consultant:	_____________	**Catruc Mariana, lect. univ.**

Chişinău, 2024

Rezumat

În ultimii ani, jocurile în cloud au apărut ca un paradigmă în plină expansiune, revoluționând furnizarea de jocuri ca serviciu prin exploatarea puterii resurselor din cloud. Abordarea predominantă în cadrul cadrului actual de jocuri în cloud implică implementarea întregului motor de joc în mașini virtuale (VM-uri). Acest lucru este necesar datorită cuplării strânse a diferitelor subsisteme ale motorului de joc, incluzând grafica, fizica și inteligența artificială.

Eficiența acestei abordări convenționale depinde de capacitatea VM-ului din cloud de a furniza în mod constant niveluri ridicate de performanță, disponibilitate și fiabilitate. Cu toate acestea, această presupunere se confruntă cu provocări formidabile din cauza naturii intricate a degradării calității serviciului atât în interiorul, cât și în afara mediului de cloud. Factori precum defectele de sistem, problemele de conectivitate la rețea și limitele de date impuse de consumatori contribuie la peisajul precar în care serviciul de jocuri operează, ducând adesea la întreruperi disruptive ale serviciului de jocuri.

În răspuns la aceste provocări, obiectivul proiectului nostru inovator este să conceptualizeze și să dezvolte un motor grafic distribuit care adoptă o strategie de cuplare flexibilă, dezangajând eficient redarea grafică de motorul de joc. Această abordare revoluționară facilitează execuția dinamică a motorului de joc pe o rețea de VM-uri din cloud și dispozitive ale clientului.

Spre deosebire de cadrele prevalente, soluția noastră introduce o schimbare de paradigmă, permițând jocurilor să funcționeze fără probleme chiar și în fața degradării performanței și a eșecurilor serviciului de cloud. Această reziliență împuternicește dezvoltatorii de jocuri să capitalizeze pe o gamă de API-uri grafice heterogene, eliberându-i de constrângerile impuse de sistemele de operare și specificațiile hardware.

Prin dezangajarea redării grafice de motorul de joc, motorul nostru nu numai că îmbunătățește adaptabilitatea și robustețea jocurilor în cloud, dar și asigură o experiență de joc mai rezistentă și tolerată la defecte. Acest lucru nu numai că răspunde nevoilor jucătorilor care cer jocuri neîntrerupte, dar și împuternicește dezvoltatorii să exploreze noi frontiere în designul de joc și fidelitatea grafică fără a fi împiedicați de limitările impuse de arhitecturile tradiționale de joc.

Abstract

In recent years, cloud gaming has emerged as a burgeoning paradigm, revolutionizing the delivery of games-as-a-service by harnessing the power of cloud resources. The prevailing approach in existing cloud gaming frameworks involves deploying the entire game engine within Virtual Machines (VMs). This is necessitated by the inherent tight-coupling of various game engine subsystems, encompassing graphics, physics, and artificial intelligence.

The efficacy of this conventional approach hinges on the ability of the cloud VM to consistently deliver high levels of performance, availability, and reliability. However, this assumption faces formidable challenges due to the intricate nature of Quality of Service (QoS) degradation both within and beyond the confines of the cloud environment. Factors such as system failures, network connectivity issues, and consumer-imposed data caps all contribute to the precarious landscape wherein the gaming service operates, often resulting in disruptive game service outages.

In response to these challenges, the scope of our innovative project is to conceptualize and develop distributed graphics engine that embraces a loose-coupling strategy, effectively disentangling the graphical renderer from the game engine. This groundbreaking approach facilitates the dynamic execution of the game engine across a network of cloud VMs and client devices.

Unlike the prevalent frameworks, our solution introduces a paradigm shift by enabling games to operate seamlessly even in the face of performance degradation and cloud service failures. This resilience empowers game developers to capitalize on a spectrum of heterogeneous graphical APIs, liberating them from the constraints imposed by Operating Systems and hardware specifications.

By decoupling the graphical renderer from the game engine, our engine not only enhances the adaptability and robustness of cloud gaming but also ensures a more resilient and fault-tolerant gaming experience. This not only caters to the needs of gamers who demand uninterrupted gameplay but also empowers developers to explore new frontiers in game design and graphical fidelity without being encumbered by the limitations imposed by traditional gaming architectures.

Contents

LIST OF FIGURES **9**

INTRODUCTION **10**

1 DOMAIN ANALYSIS **11**

1.1 Game Engines 11

1.2 Renderer 12

1.3 Cloud gaming 12

1.4 Issues and challenges 13

1.4.1 Interaction Delay Tolerance 13

1.4.2 Video Streaming and Encoding 14

1.4.3 Cloud gaming limitations 15

1.5 Related Work 16

2 SYSTEM SPECIFICATIONS AND DESIGN **18**

2.1 Rationale 18

2.2 Cloud component 19

2.2.1 Buffering 20

2.2.2 Discovery 21

2.2.3 Conversion 21

2.2.4 Drawing 22

2.3 Mitigation 23

2.3.1 Interlacing 23

2.3.2 Hot-Swapping 23

2.3.3 Local and Cloud Rendering 24

2.3.4 Adaptive Cloud Rendering in Unstable Networks 24

2.3.5 Offline Cloud Redundancy 25

2.3.6 Cloud Failure Resilience 27

2.4 Architecture 28

2.4.1 Distributed Graphical Rendering Engine 28

2.4.2 Graphic API Independence 29

2.4.3 Hot Swapping 29

2.4.4 Frame Interlacing 30

2.4.5 Cloud Rendering Setup 31

2.4.6 Adaptive Frame Distribution 32

2.4.7 Command Processing Workflow 32
2.4.8 Rendering and Transmission 33
3 SYSTEM IMPLEMENTATION 35
3.1 OpenGL 35
3.1.1 Basic concepts 35
3.1.2 Rendering process 35
3.1.3 Features and Capabilities 36
3.2 Vulkan 36
3.2.1 Basic Concepts 36
3.2.2 Rendering Process 36
3.2.3 Features and Capabilities 37
3.3 Video Streaming and Encoding 37
3.3.1 Definition 37
3.3.2 How it works 38
3.3.3 Key Components 38
3.3.4 Video Encoding 38
3.3.5 Video Codecs 39
3.3.6 Adaptive Bitrate Streaming (ABR) 39
3.3.7 Transcoding 39
3.4 Results 39
3.4.1 Experimental Setup 40
3.4.2 Performance Analysis 41
3.4.3 Overall Performance Gains 42
CONCLUSIONS 43
REFERENCES 44

LIST OF FIGURES

1.1 Generic Game Loop 11

1.2 Graphics Pipeline 12

1.3 Cloud Gaming Architecture 13

2.1 Stable network Cloud 24

2.2 Unstable network 25

2.3 No internet connection 26

2.4 Cloud instance failure 27

3.1 Framerate to cloud-client workload 41

INTRODUCTION

Video games are now the largest entertainment industry in the world, growing to $143.5b revenue in 2020 [1] with the PC distribution platform; Steam, reaching over 22 million concurrent players alone [2]. Cloud gaming - a paradigm whereby gaming is delivered as a service by leveraging cloud resources - has begun to gain increasing popularity in society, providing advantages over traditional desktop and console gaming including lower installation times, reduced hardware cost, greater device portability, and the ability to leverage cloud resources for higher graphical quality.

A design principle shared across all existing cloud gaming frameworks is that service is provisioned by deploying a game instance (an instantiation of a game program) within a Virtual Machine (VM), whereby game state is manipulated and resulting frames are encoded and streamed to a consumer client device based on user input [3]. This approach is necessary primarily due to the game engine architecture: Monolithic systems with multiple subsystems tightly coupled with each other and the underlying operating system to facilitate aspects of graphics, physics, audio, and AI.

However such a design results in limitations within cloud gaming due to a strong dependence on performance and dependability of the cloud game instance and its connection to client devices. In the cloud, game instances deployed within VMs are exposed to a plethora of detrimental datacenter behaviors, spanning interference [10], resource contention [11], and failures [12]. Such behaviors - in isolation or in tandem - result in Quality of Service (QoS) degradation in terms of lower interactivity and frames per second (FPS), lower graphical quality (lower resolution, bitrate), as well as reduced availability and reliability [13]. While various approaches have been proposed to alleviate such issues [4, 14, 15], they all assume a stable network connection established between the client device and the cloud game instance. Thus, such approaches are unable to tolerate cloud outages, prolonged network disconnection, or consumer data caps, resulting in consumers unable to access games entirely.

One approach to overcome these limitations would be to dynamically distribute all game engine subsystems across the cloud. A distributed game engine would allow for dynamic deployment and reconfiguration of all subsystems across both cloud and client systems in response to the specified QoS requirements. However this is currently not possible in existing game engine architectures due to tight-coupling with their underlying platforms [16]. Taking graphical rendering (a core game subsystem) as an example, game systems are developed to use specific graphics APIs (OpenGL, Vulkan, etc.), and require specialized developer knowledge to take advantage of advanced hardware features (e.g. real-time ray-tracing). Thus it is not possible to transition to other graphics APIs and platforms without significant development effort and time.

Bibliography

1. B. Newzoo, "The global games market report," *Amsterdam: gamesindustry. com* 2017.
2. V. Inc. (2020) Steam: Game and player statistics.[Online]. Available: :https://store.steampowered.com/stats/
3. C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen,"Gaminganywhere: an open cloud gaming system," in *Proceedings of the 4th ACM multimedia systems conference*, 2013, pp. 36–47.
4. K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming." *n Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, 2015, pp. 151–165.
5. D. De Winter, P. Simoens, L. Deboosere, F. De Turck, J. Moreau, B. Dhoedt, and P. Demeester, "A hybrid thinclient protocol for multimedia streaming and interactive gaming applications," in *Proceedings of the international workshop on Network and operating systems support for digital audio and video*, 2006, pp. 1–6.
6. Microsoft. (2020) Microsoft project xcloud. [Online]. Available: https://www.xbox.com/en-US/xbox-game-streaming/project-xcloud
7. Google. (2019) Google stadia. [Online]. Available: https://stadia.google.com/gg/
8. T. Koehler, A. Dieckmann, and P. Russell, "An evaluation of contemporary game engines," in *Proceedings of the 26th eCAADe Conference*, 2008, pp. 743–750.
9. E. F. Anderson, S. Engel, P. Comninos, and L. McLoughlin, "The case for research in game engine architecture," in *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, 2008, pp. 228–231.
10. M. Kambadur, T. Moseley, R. Hank, and M. A. Kim, "Measuring interference between live datacenter applications," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.* IEEE, 2012, pp. 1–12.
11. P. Garraghan, X. Ouyang, R. Yang, D. McKee, and J. Xu, "Straggler root-cause and impact analysis for massivescale virtualized cloud datacenters," *IEEE Transactions on Services Computing*, vol. 12, pp. 91–104, Jan 2019.
12. X. Chen, C. Lu, and K. Pattabiraman, "Failure analysis of jobs in compute clouds: A google cluster case study," in *2014 IEEE 25th International Symposium on Software Reliability Engineering*, Nov 2014, pp. 167–177.
13. A. Avizienis, J. . Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, , vol. 1, no. 1, pp. 11–33, Jan 2004.
14. 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*, 2017, pp. 1–14.
15. J. R. Lange, P. A. Dinda, and S. Rossoff, "Experiences with client-based speculative remote display." in *USENIX Annual Technical Conference*, 2008, pp. 419–432.
16. D. Maggiorini, L. A. Ripamonti, E. Zanon, A. Bujari, and C. E. Palazzi, "Smash: A distributed game engine architecture," in *2016 IEEE Symposium on Computers and Communication (ISCC)*), 2016, pp. 196–201.
17. K. Group. (2017) Khronos opengl 4.6 specification. [Online]. Available: https://registry.khronos.org/OpenGL/specs/gl/glspec46.core.pdf

18. Khronos Group. (2020) Khronos vulkan specification. [Online]. Available: https://registry.khronos.org/vulkan/specs/1.2-extensions/html/vkspec.html
19. H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu, "Placing virtual machines to optimize cloud gaming experience," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 42–53, 2015.
20. Ubitus. (2019) Ubitus gamecloud. [Online]. Available: https://ubitus.net/
21. Y. Xu, Q. Shen, X. Li, and Z. Ma, "A cost-efficient cloud gaming system at scale," *IEEE Network, vol. 32, no. 1*, pp. 42–47, 2018.

    John Hosking, and Yun Yang