HOCHSCHULE HEILBRONN

BACHELOR THESIS

Entwicklung und Evaluation einer Plattform für zugängliches Cloud-Gaming zur Bereitstellung von High-End-Spieledemos auf Low-End-Geräten

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Software Engineering

Hochschule Heilbronn
University of Applied Sciences
Fakultät IT

Author:	Frank Mayer
Matrikelnummer:	215965
Betreuung:	Prof. Dr. Thomas Fankhauser
	Prof. Dr. rer. nat. Nicole Ondrusch
Abgabedatum:	22.08.2025

***Abstract* - this is my abstract**

I. Glossar

A. Computerspiel/Videospiel

Die Begriffe Computerspiel und Videospiel werden in dieser Arbeit synonym verwendet. Sie bezeichnen interaktive, elektronische Spiele, die auf verschiedenen Endgeräten wie PCs, Spielkonsolen oder mobilen Geräten ausgeführt werden.

B. Entwickler

Als Entwickler wird das Unternehmen bezeichnet, das die hauptsächliche Verantwortung für die technische und kreative Umsetzung eines Videospiels trägt.

C. Publisher

Der Publisher, auch Verleger genannt, ist das Unternehmen, das die Finanzierung der Entwicklung, das Marketing sowie den Vertrieb eines Videospiels verantwortet.

D. AAA

Ausgesprochen /trɪ̆p′əl eɪ/

Der Begriff AAA, oft auch Triple-A geschrieben, bezeichnet eine Klassifikation für Videospiele, die sich durch besonders hohe Entwicklungs- und Marketingbudgets auszeichnen. [1]

Aktuelle Beispiele verdeutlichen die finanzielle Dimension von AAA-Produktionen:

- *Spider-Man 2* (2023) von Insomniac Games: 315 Millionen US-Dollar [2]
- *Star Wars Jedi: Survivor* (2023) von Respawn Entertainment: 300 Millionen US-Dollar [3]

Durch das hohe Budget, welches unter anderem in die Grafik der Spiele investiert wird, ist der größte Teil der AAA-Spiele als High-End-Spiele zu bezeichnen. [4], [5], [6]

E. Low-End

Der Begriff Low-End bezieht sich auf Hardwarekomponenten oder Endgeräte, die eine vergleichsweise geringe Rechenleistung aufweisen. Dies kann sowohl neuere, kostengünstige Geräte wie einen Raspberry Pi als auch ältere Systeme umfassen, deren Leistung nicht mehr den aktuellen technischen Standards entspricht.

Als Low-End-Spiele werden Titel bezeichnet, die so konzipiert sind, dass sie auch auf leistungsschwacher Hardware flüssig ausgeführt werden können.

F. High-End

Im Gegensatz dazu beschreibt High-End Hardware oder Geräte mit maximal verfügbarer Rechenleistung, die typischerweise in aktuellen, hochpreisigen Desktop-Computern zu finden ist.

High-End-Spiele setzen eine solche leistungsstarke Hardware, insbesondere im Bereich der Grafikprozessoren (GPU), voraus. Auf leistungsschwächeren Systemen wie Spielkonsolen sind sie oft nur mit reduzierten Grafikeinstellungen spielbar, während sie auf Low-End-Geräten in der Regel nicht lauffähig sind.

G. Demo

Eine Demo ist eine kostenlos verfügbare, eingeschränkte Version eines Videospiels, die zu Marketingzwecken veröffentlicht wird. Sie dient dazu, potenziellen Käufern einen Einblick in das Spielerlebnis zu gewähren und sie zum Kauf der Vollversion zu animieren.

Die Einschränkungen können vielfältig sein und umfassen beispielsweise eine zeitliche Begrenzung der Spielzeit, die Beschränkung auf wenige Level oder Spielmodi sowie das Deaktivieren der Speicherfunktion für den Spielfortschritt.

H. AWS

Amazon Web Services ist der größte Cloud-Provider mit einem Marktanteil von über 30%. [7]

a) *EC2:*

AWS EC2 ist ein Web-Service, der in der AWS-Cloud skalierbare virtuelle Server (Instances) auf Abruf bereitstellt.

b) *AMI:*

Da Amazon Machine Image ist ein proprietäres Format für eine AWS EC2-Instanz.

c) *Lambda:*

AWS Lambda ist ein Compute-Service, der Code automatisch und skalierend in Reaktion auf Events ausführt, ohne dass Server verwaltet werden müssen.

II. Forschungsfragen

Die vorliegende Arbeit verfolgt das Ziel, die Konzeption und Realisierbarkeit einer spezialisierten Cloud-Gaming-Plattform für Spieledemos zu untersuchen. Aus dieser Zielsetzung leitet sich die folgende Hauptforschungsfrage (HFF) ab:

HFF: Wie muss eine Cloud-Gaming-Plattform architektonisch und technologisch konzipiert sein, um High-End-Spieledemos performant auf Low-End-Geräten bereitzustellen, und welche Implikationen ergeben sich daraus für Nutzer sowie Betreiber?

Zur systematischen Beantwortung dieser übergeordneten Frage werden die folgenden untergeordneten Forschungsfragen (UFF) untersucht:

- **UFF 1: Welche Architekturmuster und Systemkomponenten eignen sich für die Realisierung einer spezialisierten Cloud-Gaming-Plattform, die auf die Bereitstellung von Spieledemos für leistungsschwache Endgeräte optimiert ist?**
- **UFF 2: Durch welche Kombination von Streaming-Protokollen, Codecs und Infrastrukturtechnologien kann eine geringe Latenz und eine hohe visuelle Qualität bei der Übertragung von Spieledemos sichergestellt werden?**
- **UFF 3: Welchen Mehrwert bietet eine solche Plattform für Endnutzer im Vergleich zu traditionellen Download-Demos, und welche technischen sowie nutzerzentrierten Hürden müssen für eine hohe Akzeptanz überwunden werden?**
- **UFF 4: Welche betriebswirtschaftlichen Geschäftsmodelle sind für den Betrieb einer Cloud-Gaming-Plattform für Spieledemos tragfähig und welche strategischen Vorteile ergeben sich daraus für Spieleentwickler und Publisher?**

A. Hypothesen

Vor jeglicher Recherche, stellte der Autor folgende Hypothesen auf:

- Ein großer Server sollte reichen, um mehrere Spiele gleichzeitig zu betreiben. Monolithische Architektur ist übersichtlich und einfach zu deployen.
- WebRTC ist bekannt und weit verbreitet. Es ist eine gute Wahl für Low-Latency-Videostreaming.
- Nutzer müssten mit Cloud-Gaming-Demos nicht auf Downloads warten und müssten die Hardwareanforderungen der Spiele nicht beachten. Nutzer sind davon abgeschreckt, zusätzliche Software installieren zu müssen oder sich einen Account anzulegen.
- So eine Plattform muss kostenlos angeboten werden, um genutzt zu werden, da sie auf dem als kostenlos verbreiteten Prinzip von Demos basiert. Das wird als zusätzliche, teure Werbemaßnahme bedeuten, dass die Plattform nicht wirtschaftlich ist (abhängig von der Konvertierungsrate).
- Spieler werden diese Cloud-Gaming-Demos nicht mehr nutzen als die aktuell verbreiteten, lokal installierten Demos.

III. Methodik und Vorgehensweise

Um eine fundierte und praxistaugliche Lösung zu entwickeln, wurde eine iterative und explorative Vorgehensweise gewählt. Der Prozess gliedert sich in drei wesentliche Phasen: eine initiale Recherchephase, die Konzeption und Verwerfung eines ersten Architekturansatzes sowie die Entwicklung eines finalen Lösungsdesigns.

A. Phase 1: Fundamentale Recherche und Anforderungsanalyse

Zu Beginn der Arbeit wurde eine grundlegende Literatur- und Marktrecherche durchgeführt, um die technologischen Kernkomponenten und deren Zusammenspiel zu verstehen. Der Fokus lag auf der Schnittstelle zwischen Hardware und Software im Kontext des Renderings von Videospielen. Untersucht wurden insbesondere:

- Die Grafikschnittstelle: Analyse der Funktionsweise und Kommunikation zwischen Grafikprozessor (GPU), zugehörigen Treibern und der Spiel-Engine.
- Hardware-Anforderungen: Evaluierung der am Markt verfügbaren GPU-Herstellern, um die Mindestanforderungen für das flüssige Rendern moderner Spieletitel zu definieren.

Diese Phase schuf die notwendige Wissensbasis, um erste Architekturkonzepte zu entwerfen.

B. Phase 2: Evaluierung eines monolithischen Ansatzes

Basierend auf den ersten Erkenntnissen wurde ein monolithischer Architekturansatz formuliert. Die zentrale Idee bestand darin, mehrere Spielinstanzen ressourcensparend auf einem einzigen, leistungsstarken Host-System zu bündeln. Zur technischen Umsetzung wurden Virtualisierungstechnologien wie Container (z. B. Docker) und leichtgewichtige Micro-VMs (z. B. Firecracker) als mögliche Isolationsmechanismen evaluiert.

Die Untersuchung dieses Ansatzes deckte jedoch eine fundamentale technische Limitierung auf: Die Rendering-Leistung einer einzelnen GPU ist in der Praxis auf eine, bei geringeren Anforderungen auf maximal zwei, grafisch anspruchsvolle Spielinstanzen beschränkt. Eine vertikale

Skalierung durch das Hinzufügen weiterer GPUs in einem System (Multi-GPU-Setups) ist für diesen Anwendungsfall ebenfalls nicht zielführend, da moderne Spiele und deren Engines diese Technologie kaum noch unterstützen und nicht auf die parallele Nutzung mehrerer Grafikkarten ausgelegt sind.

Aufgrund dieser nicht überwindbaren Skalierungsproblematik, die eine wirtschaftliche und performante Bereitstellung für mehrere Nutzer gleichzeitig verhindert, wurde dieser monolithische Ansatz verworfen. Die Erkenntnis war, dass eine starre 1-zu-N-Beziehung (ein Host für N Spiele) im High-End-Gaming-Bereich nicht realisierbar ist.

C. Phase 3: Strategische Neuausrichtung zum Serverless-Modell

Als Konsequenz aus den gewonnenen Erkenntnissen und in Abstimmung mit der Betreuung der Arbeit wurde die Strategie neu ausgerichtet. Der Fokus verlagerte sich auf ein Serverless-Computing-Modell.

Das Kernprinzip dieses Ansatzes besteht darin, für jede Benutzersitzung eine dedizierte und kurzlebige Server-Instanz dynamisch bei Bedarf bereitzustellen. Sobald die Spieledemo beendet wird, wird die Instanz wieder vollständig entfernt. Dieses Modell löst die zuvor identifizierte Limitierung des monolithischen Designs auf elegante Weise:

- Performance: Jeder Nutzer erhält die exklusiven Ressourcen einer Instanz inklusive einer dedizierten GPU, was eine maximale Performance sicherstellt.
- Skalierbarkeit: Das System skaliert horizontal, indem für jeden neuen Nutzer eine neue, unabhängige Instanz gestartet wird. Die Skalierungsgrenze wird somit nur durch die Kapazitäten des zugrundeliegenden Cloud-Providers bestimmt.

Dieser Ansatz bildet die Grundlage für das in den folgenden Kapiteln detailliert beschriebene Lösungsdesign.

IV. Zielgruppe

Um eine nutzerzentrierte Lösung zu entwickeln, wurde eine Umfrage unter Videospiel-Enthusiasten durchgeführt. Den Teilnehmenden wurde das Konzept vorgestellt, Demos von Videospielen zukünftig kostenfrei über eine vom Publisher oder Entwickler bereitgestellte Cloud-Gaming-Plattform zu spielen, anstatt sie auf der eigenen Hardware installieren zu müssen.

Die Stichprobe umfasste 35 Personen, davon 32 aus Deutschland und drei aus den USA. Die Teilnehmerschaft setzte sich überwiegend aus Studierenden des Software-Engineerings sowie Personen anderer Berufs- und Altersgruppen zusammen. Die Teilnahme erfolgte freiwillig und ohne Anreize über das Online-Tool Blocksurvey.

Die Ergebnisse zeigen, dass 80% der Befragten vorwiegend einen Desktop-PC als Spielplattform nutzen.

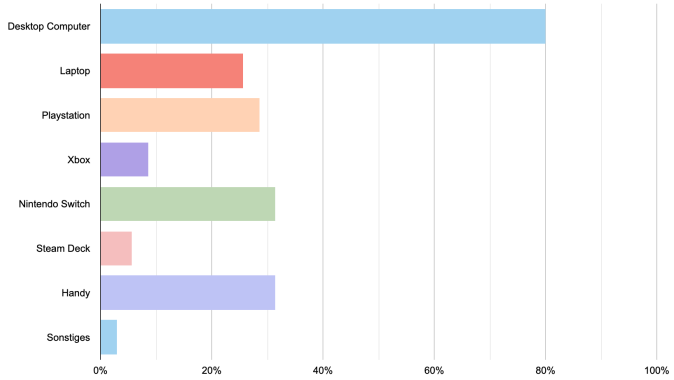

Fig. 1. Umfrage - Auf welchem Gerät spielst du Videospiele?

Hinsichtlich der Installationsgröße gaben 94.3% an, Spiele mit einem Speicherbedarf von 60 GB oder mehr zu nutzen. Bei 37.1% der Befragten überschreitet dieser sogar 100 GB.

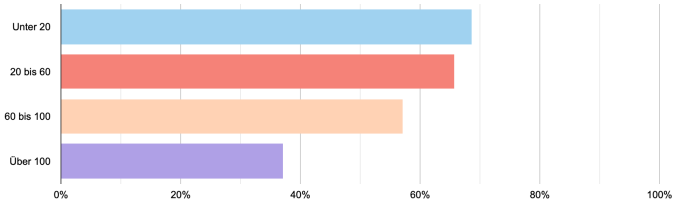

Fig. 2. Umfrage - Wie groß sind die Spiele, die du spielst? (in GB)

Eine deutliche Mehrheit bekundete zudem Interesse an AAA-Spielen. Laut den Angaben auf der Vertriebsplattform Steam stellen diese Titel in der Regel die höchsten Anforderungen an die Hardware (CPU, GPU) und den Festplattenspeicher. [4], [5], [6]

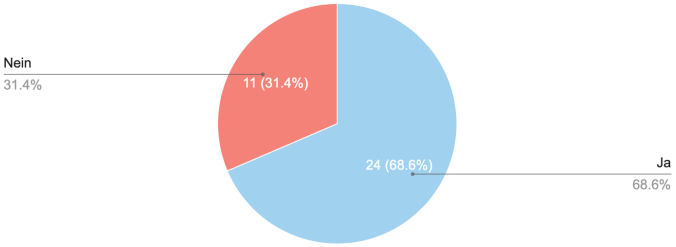

Fig. 3. Umfrage - Interessierst du dich für AAA Games?

Die Umfrage ergab weiterhin, dass die Mehrheit der Befragten bereits Erfahrungen mit Cloud-Gaming-Diensten gesammelt hat. Die Antwortoption “Andere” bezog sich dabei vornehmlich auf den Dienst Google Stadia, welcher zum 18.01.2023 eingestellt wurde. [8]

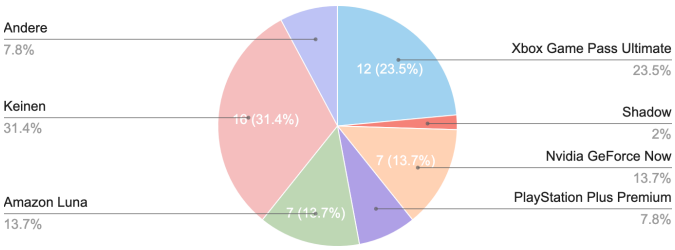

Fig. 4. Umfrage - Welchen Cloud Gaming Anbieter hast du bereits verwendet?

Die von den Teilnehmern angegebene Internetverbindung weist eine durchschnittliche Download-Geschwindigkeit von $325.4\frac{\text{Mb}}{\text{s}}$ (Median: $105\frac{\text{Mb}}{\text{s}}$) und eine durchschnittliche Latenz (Ping) von 28.88 ms (Median: 16 ms) auf.

Die zentrale Fragestellung der Umfrage bezog sich auf die Bereitschaft, Spieledemos via Cloud-Gaming zu nutzen. Eine deutliche Mehrheit von 68.6% befürwortete diesen Ansatz gegenüber dem lokalen Download. Weitere 11.4% zeigten sich unter Vorbehalt zustimmend, wobei die Akzeptanz von Faktoren wie dem Nutzungsaufwand (z.B. Registrierung, Wartezeiten) und der technischen Qualität, insbesondere der Latenz, abhängt.

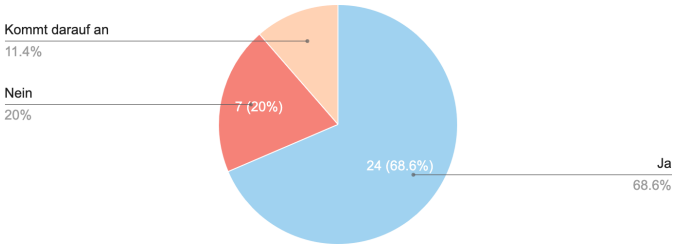

Fig. 5. Umfrage - Würdest du Demos über Cloud Gaming spielen, bevor du ein Spiel kaufst?

Basierend auf den Umfrageergebnissen lässt sich die Zielgruppe, eine Teilmenge der Gesamtspielerschaft, wie folgt charakterisieren:

1) Überwiegende Nutzung von Desktop-PCs als primäre Spielplattform.
2) Breites Interesse an Spielen unterschiedlicher Größe, mit einem explizit hohen Interesse an ressourcenintensiven AAA-Titeln.
3) Vorhandene Erfahrungen mit Cloud-Gaming-Diensten.
4) Hohe Bereitschaft, Spieledemos über Cloud-Gaming anstelle lokaler Installationen zu nutzen.

Das Ziel der vorliegenden Arbeit ist die Konzeption einer Cloud-Gaming-Lösung, die es Publishern sowie Self-Publishing-Entwicklern ermöglicht, Demos ihrer Spiele unkompliziert bereitzustellen. Hierfür wird zunächst der aktuelle Stand der Technik sowie bestehende Lösungsansätze analysiert. Aufbauend auf dieser Analyse wird ein Konzept entwickelt, das spezifische Herausforderungen wie die effiziente Bereitstellung von Demos und den Schutz des geistigen Eigentums adressiert. Dabei wird auf existierende Teillösungen zurückgegriffen, sofern diese für den Anwendungsfall geeignet sind.

V. Kernherausforderungen

Die Implementierung und der Betrieb von Cloud-Gaming-Plattformen sind mit signifikanten Herausforderungen verbunden, welche in diesem Kapitel systematisch analysiert werden.

A. Latenz und Bandbreite

Eine der fundamentalsten Herausforderungen für Cloud-Gaming ist die Gewährleistung einer geringen Ende-zu-Ende-Latenz, die für ein reaktionsfähiges und immersives Spielerlebnis entscheidend ist. In der Fachliteratur wird ein Latenzbudget von $\lesssim 100$ ms als Obergrenze für eine akzeptable Nutzererfahrung angesehen [9]. Das Einhalten dieses Budgets erfordert eine hochoptimierte Übertragungsstrecke vom Rechenzentrum zum Endgerät.

B. Hardware-Performance

Jede serverseitige Spielinstanz eines AAA-Titels erfordert eine leistungsstarke Grafikkarte (GPU), um eine Darstellung mit hohen Grafikeinstellungen und flüssigen Bildwiederholraten zu ermöglichen. Hierfür kommen typischerweise Rechenzentrum-GPUs wie die NVIDIA RTX 6000 oder A10G zum Einsatz.

Als Referenz für die Leistungsanforderungen auf Endnutzerseite dient beispielsweise die für *The Last of Us Part II Remastered* empfohlene NVIDIA GeForce RTX 3060 [6]. Diese erreicht in Kombination mit KI-gestützten Technologien wie Frame Generation [10] Bildraten von circa 100 FPS [11], was ein flüssiges Spielerlebnis sicherstellt.

Im High-End-Segment für AAA-Spiele liegt der technologische Fokus aktuell auf Hardware von NVIDIA. Berichten zufolge konzentrieren sich Wettbewerber wie AMD und Intel strategisch auf andere Marktsegmente, weshalb NVIDIA-Technologien für die hier betrachteten Anwendungsfälle von besonderer Relevanz sind [12], [13].

C. Umfangreiche Binärdateien

Eine logistische Herausforderung stellt das Management der Spieldateien dar. Moderne AAA-Titel erreichen Dateigrößen im Bereich von 100 bis 200 Gigabyte (GB), wie die folgende Auswahl verdeutlicht:

- *STAR WARS Jedi: Survivor* (155 GB) [14]
- *Assassin's Creed Shadows* (115 GB) [15]

- *The Elder Scrolls IV: Oblivion Remastered* (125 GB) [16]
- *Black Myth: Wukong* (130 GB) [17]
- *God of War Ragnarök* (190 GB) [4]
- *FINAL FANTASY VII REBIRTH* (155 GB) [5]
- *The Last of Us Part II Remastered* (150 GB) [6]
- *Dragon Age: The Veilguard* (100 GB) [18]
- *Cyberpunk 2077* (70 GB) [19]

Diese Datenmengen führen zu erheblichen Zeitaufwänden bei der Datenübertragung, beispielsweise bei der initialen Bereitstellung der Spiele auf den Servern. Bei einer serverseitigen Anbindung mit einer Bandbreite von $1\frac{\text{Gb}}{\text{s}}$ resultiert die Übertragung eines 100-GB-Spiels in einer Kopierzeit von: $T(\text{copy}) = \frac{100\ \text{GB}}{1\frac{\text{Gb}}{\text{s}}} = 800s \approx 13.3$ Minuten

Zum Vergleich: Der Download eines solchen Spiels durch einen Endnutzer in Deutschland mit einer durchschnittlichen Geschwindigkeit von $79.1\frac{\text{Mb}}{\text{s}}$ [20] würde circa 2.8 Stunden dauern: $T(\text{copy}) = \frac{100\ \text{GB}}{79.1\frac{\text{Mb}}{\text{s}}} \approx 10113.8s \approx 2.8$ Stunden

Da der Fokus dieser Arbeit auf grafisch anspruchsvollen AAA-Titeln liegt, werden Spiele mit geringerem Speicherbedarf, bei denen diese Problematik weniger ausgeprägt ist, in der weiteren Analyse nicht berücksichtigt.

D. Skalierbarkeit

Die Cloud-Gaming-Plattform muss eine hohe Skalierbarkeit aufweisen, um Lastspitzen bei der Veröffentlichung populärer Spieledemos abfangen zu können. Die Analyse erfolgreicher Demos liefert hierfür wichtige Anhaltspunkte. So erreichte beispielsweise die Demo zu *Stellar Blade* Spitzenwerte von rund 25.000 gleichzeitig aktiven Spielern allein auf der Plattform Steam [21]. Die Infrastruktur muss demnach für eine solche Nutzerzahl ausgelegt sein, um einen stabilen Betrieb zu gewährleisten.

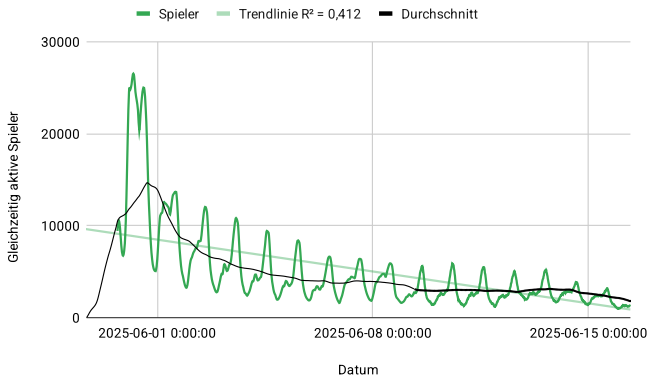

Fig. 6. *Stellar Blade* Demo – Gleichzeitig aktive Spieler auf Steam [21]

E. Kosten pro Nutzer

Ein zentraler Aspekt für die Wirtschaftlichkeit des Betreibermodells sind die Kosten pro Nutzer. Da das anvisierte Geschäftsmodell auf kostenfreien Spieledemos basiert, entfällt eine direkte Monetarisierung des Spielvorgangs. Die Rentabilität muss stattdessen durch nachgelagerte Konversionen, also den Kauf der Vollversion durch von der Demo überzeugte Spieler, erzielt werden. Dies erfordert eine strikte Kontrolle der laufenden Betriebskosten.

Für die nachfolgende Machbarkeitsanalyse wird eine Zielgröße für die Betriebskosten von $1 pro Stunde und Spieler als Referenzwert definiert. Eine detaillierte Wirtschaftlichkeitsberechnung durch den Betreiber müsste Faktoren wie die erwartete Anzahl an Demo-Nutzern, die Konversionsrate zum Kauf der Vollversion sowie eventuelle Lizenzkosten miteinbeziehen.

F. Sicherheit und DRM

Schließlich ist die Gewährleistung der Sicherheit und des Schutzes geistigen Eigentums eine wesentliche Anforderung. Es müssen effektive Mechanismen des Digital Rights Management (DRM) implementiert werden, um zu verhindern, dass Nutzer die Spieldateien von der Plattform extrahieren und unautorisiert weiterverwenden. Dies dient dem Schutz der Rechte der Spieleentwickler und Publisher und ist eine Grundvoraussetzung für deren Kooperation.

VI. Stand der Technik

Google Stadia verwendete H.264 und VP9 über WebRTC. Nvidia GeForce Now sendet H.264 über einen RTP-Stream durch ein UDP Sockert. [22] Shadow lässt den User über UDP und TCP im Client entscheiden. [23]

Neuere Protokolle, die auf QUIC basieren, gewinnen im Echtzeit-Streaming an Bedeutung, sind aber noch in der Minderheit. [24] Zudem können verschiedene Protokolle mit QUIC genutzt werden, hier mangelt es noch an einem etablierten Standard.

Nvidia GeForce Now verwendet kurzlebige VMs. [25]

Auch Google Stadia basiert auf VMs. Google verwendet aber eine eigene Virtualisierungs-Layer. Zudem können Stadia-Inctances direkt (ohne über das Internet gehen zu müssen) miteinander kommunizieren, was einen großen Vorteil bei Multiplayer-Spielen bietet. [26]

Google Stadia verwendet custom GPUs von AMD, die ein Multi-GPU-Setup hinter der Vulkan-API so virtualisieren, dass ein Spiel die Leistung mehrerer GPUs nutzen kann, ohne dass die Engine das unterstützt. Für die Engine sieht es so aus wie eine große GPU. [26]

NVIDIA hat die vGPU-Funktionalität, welche bei GeForce Now eingesetzt wird, auf Consumer-Grafikkarten historisch eingeschränkt, was es für einzelne Benutzer oder kleinere Setups schwierig macht, eine einzelne GPU auf mehrere VMs zu partitionieren, ohne spezialisierte Rechenzentrum-GPUs und die damit verbundenen Lizenzen. [27]

A. Nvidia GPU Technologien

NVIDIA stellt eine Reihe von Schlüsseltechnologien zur Verfügung, die maßgeblich zur Leistungssteigerung bei grafisch anspruchsvollen Spielen beitragen und somit die Hardware-Anforderungen adressieren.

a) *Multi Frame Generation:*

Multi Frame Generation ist ein fortschrittliches Verfahren zur Synthese von Bildern, das darauf abzielt, die wahrgenommene Bildwiederholrate (Framerate) signifikant zu erhöhen. Im Gegensatz zur Generierung eines einzelnen Zwischenbildes, wie es bei früheren Implementierungen der Fall war, ist diese Technik in der Lage, mehrere Bilder zwischen zwei von der Game-Engine gerenderten Frames zu interpolieren. Unter Verwendung von Bewegungsvektoren, optischen Flussfeldern und temporalen Daten aus vorangegangenen Bildern rekonstruiert ein neuronales Netzwerk eine Sequenz von Zwischenbildern. Das primäre Ziel ist die Erzeugung einer als extrem flüssig wahrgenommenen Bewegung, die weit über die native Renderleistung der Hardware hinausgeht, insbesondere in CPU-limitierten Szenarien. [10]

b) *Transformer-basierte Ray Reconstruction:*

Die Transformer-basierte Ray Reconstruction (TRR) ist eine Methode zur Rauschunterdrückung (Denoising) und Detailrekonstruktion von Ray-Tracing-Effekten, die auf einer Transformer-Architektur basiert. Anstelle von konventionellen Faltungsnetzwerken (CNNs) nutzt TRR die Fähigkeit von Transformern, globale Bildkontexte und langreichweitige Abhängigkeiten zwischen Pixeln zu analysieren. Dadurch kann das Modell fehlende oder verrauschte Lichtinformationen (z.B. Reflexionen, Schatten, globale Beleuchtung) mit höherer Genauigkeit und Kohärenz wiederherstellen. Dies resultiert in einer visuell präziseren und artefaktärmeren Darstellung von Ray-Tracing-Effekten im Vergleich zu früheren Denoising-Verfahren. [10]

c) *Transformer-basierte Super Resolution:*

Transformer-basiertes Super Resolution (TSR) beschreibt den Prozess der Hochskalierung eines niedrig aufgelösten Bildes auf eine höhere Zielauflösung mittels eines Transformer-Modells. Ähnlich wie bei der Ray Reconstruction nutzt diese Technik die Stärke von Transformern bei der Erfassung globaler Kontexte. Durch die Analyse von niedrig aufgelösten Eingabebildern, Bewegungsvektoren und historischer Bilddaten rekonstruiert das neuronale Netzwerk ein hochaufgelöstes Bild. Der Vorteil gegenüber CNN-basierten Ansätzen liegt in der potenziell überlegenen Rekonstruktion von feinen Texturen und komplexen Mustern sowie einer verbesserten temporalen Stabilität, da das Modell logische Zusammenhänge über größere Bildbereiche hinweg herstellen kann. [10]

d) *Reflex Frame Warp:*

Reflex Frame Warp ist eine Latenzkompensationstechnik, die speziell für den Einsatz mit Frame-Generation-Verfahren entwickelt wurde. Die durch die Interpolation von Bildern entstehende zusätzliche Latenz wird durch dieses Verfahren aktiv reduziert. Unmittelbar bevor ein generiertes Bild an den Monitor gesendet wird, erfasst das System die aktuellsten Eingabedaten des Nutzers (z.B. Mausbewegungen). Basierend auf diesen Daten wird das bereits fertiggestellte Bild minimal verzerrt (“warped”), um die Darstellung an die letzte bekannte Spieleraktion anzugleichen. Dieser Prozess korrigiert die Diskrepanz zwischen der angezeigten Bildinformation und der Intention des Spielers und reduziert somit die wahrgenommene “Input-to-Photon”-Latenz. [10]

VII. Basistechnologien für den verfolgten Ansatz

Für den Betrieb der Spiele-Server kommen EC2 G5-Instanzen von AWS zum Einsatz. Diese Wahl begründet sich durch die vom Hersteller hervorgehobene Eignung für rechenintensive 3D-Anwendungen sowie die leistungsfähige Netzwerkanbindung. [28] Beide Aspekte sind für den in dieser Arbeit beschriebenen Anwendungsfall von entscheidender Bedeutung.

Konkret wird die Instanzvariante g5.2xlarge genutzt, welche mit 32 GiB Arbeitsspeicher, acht virtuellen CPUs (AMD EPYC 7R32 mit 2.8 GHz), einer Nvidia A10G Grafikkarte (24 GiB Speicher), einer 450 GiB NVMe-SSD und einer Netzwerkbandbreite von bis zu $10\frac{\text{Gib}}{\text{s}}$ ausgestattet ist. [29] Diese Spezifikationen erfüllen die zuvor definierten Hardwareanforderungen vollständig.

Obwohl in den folgenden Kapiteln AWS als beispielhafter Cloud-Anbieter verwendet wird, ist die konzipierte Architektur prinzipiell auch mit den Diensten anderer Serverless-Provider realisierbar.

VIII. Architektur

Zur Gewährleistung der geforderten Skalierbarkeit wurden verschiedene Ansätze für den Betrieb der Spieleserver evaluiert.

Der anfängliche Plan, dedizierte, physisch verfügbare Server zu nutzen, wurde verworfen. Dieser Ansatz kann die erforderliche dynamische Skalierbarkeit nicht realistisch gewährleisten, da Hardware-Ressourcen nicht flexibel und zeitnah an den Bedarf angepasst werden können.

Eine höhere Skalierbarkeit ließe sich durch gemietete Server in einem Rechenzentrum realisieren. Dieser Ansatz ist jedoch kostenineffizient, da die Hardware auch bei ausbleibender Nutzung vollständig bezahlt werden muss.

Als optimale Lösung wurde ein Serverless-Ansatz identifiziert, bei dem Rechenkapazitäten von einem Cloud-Provider wie AWS nur für die tatsächliche Nutzungsdauer angemietet werden. Je mehr Spieler aktiv sind, desto mehr Serverinstanzen werden dynamisch provisioniert. Bei Inaktivität entstehen keine Kosten für die ungenutzte Rechenleistung. Die primäre Begrenzung dieses Modells stellen die Betriebskosten bei hoher Auslastung dar. Zusätzlich ist zu beachten, dass AWS regionale Quotas für die maximale Anzahl gleichzeitig laufender EC2-Instanzen pro Account festlegt. Eine Erhöhung dieser Limits ist auf Anfrage möglich, erfordert bei sehr hohem Bedarf jedoch eine direkte Abstimmung mit AWS, da die Kapazitäten der Rechenzentren nicht unbegrenzt sind. [30]

A. Entwicklung

Der Entwurf der Architektur erfolgte nach einem “Backward Design”-Ansatz, ausgehend vom kritischsten Punkt: der direkten Verbindung zwischen Frontend und EC2-Instanz, über welche der Videospiel-Stream übertragen wird. Um eine minimale Latenz zu erreichen, ist eine direkte Verbindung zwischen dem Frontend des Spielers und der EC2-Instanz essenziell, da diese die Anzahl der Intermediäre in der Kommunikationsstrecke minimiert. [31]

Dies führt zur zentralen Frage der Instanziierung der EC2-Instanz. Da der Start durch eine Aktion des Spielers im Frontend ausgelöst werden soll und AWS-Dienste stark auf einer ereignisgesteuerten Architektur (Event-driven Architecture) basieren, bietet sich die Verwendung einer Lambda-Funktion als Vermittler an. Diese Funktion kann die Anfrage des Frontends entgegennehmen und den Start der EC2-Instanz initiieren.

Dieser Ansatz wirft zwei Folgeprobleme auf:

1) **Herstellung der Verbindung:** Wie erfährt das Frontend die IP-Adresse der EC2-Instanz, um eine Verbindung aufzubauen? Die startende Lambda-Funktion ist aufgrund ihrer kurzen, vorgesehenen Lebensdauer zu diesem Zeitpunkt bereits beendet.
2) **Bereitstellung des Abbilds:** Welches AMI soll für die EC2-Instanz verwendet werden und woher wird es bezogen?

Das Verbindungsproblem lässt sich durch einen Polling-Mechanismus lösen. Das Frontend fragt periodisch eine zweite Lambda-Funktion an, um den Status der Instanz zu überprüfen. Sobald die Instanz betriebsbereit ist, gibt diese Funktion die öffentliche IP-Adresse an das Frontend zurück.

Die Lösung für die Bereitstellung des Abbilds ist mehrschichtig. Es muss ein Mechanismus etabliert werden, der für jedes Spiel ein spezifisches, vorkonfiguriertes AMI bereithält. Zur Zuordnung von Spiel zu AMI eignet sich ein Key-Value-Store wie Amazon DynamoDB. In diesem wird der eindeutige Name des Spiels als Schlüssel (Key) und die zugehörige AMI-ID als Wert (Value) gespeichert.

Die Erstellung der AMIs wird durch den EC2 Image Builder automatisiert. Dieser Dienst startet eine temporäre EC2-Instanz und führt darauf ein Konfigurationsskript aus. Das Skript installiert die erforderliche Software: die Streaming-Anwendung und das eigentliche Spiel. Während die Streaming-Software aus diversen Quellen wie Paketmanagern oder S3 bezogen werden kann, stellt die Distribution des Spiels aufgrund seiner potenziell erheblichen Dateigröße eine Herausforderung dar. Amazon S3 ist hierfür die geeignete Lösung, da der Dienst für die Speicherung und den Abruf großer Datenmengen optimiert ist. [32]

Der Prozess beginnt, wenn der Spielentwickler die Spieldateien in einen S3-Bucket hochlädt, was typischerweise im Rahmen einer CI/CD-Pipeline geschieht. Dieser Upload-Vorgang löst via Amazon EventBridge automatisch den Start des Image-Builder-Prozesses aus.

B. C4: Context

Das System interagiert mit zwei primären Akteuren: dem Spielentwickler und dem Spieler. Der Entwickler stellt die Spielinhalte bereit, indem er sie auf die Plattform hochlädt. Der Spieler nutzt die Plattform, um auf diese Inhalte zuzugreifen und das Spiel zu streamen.

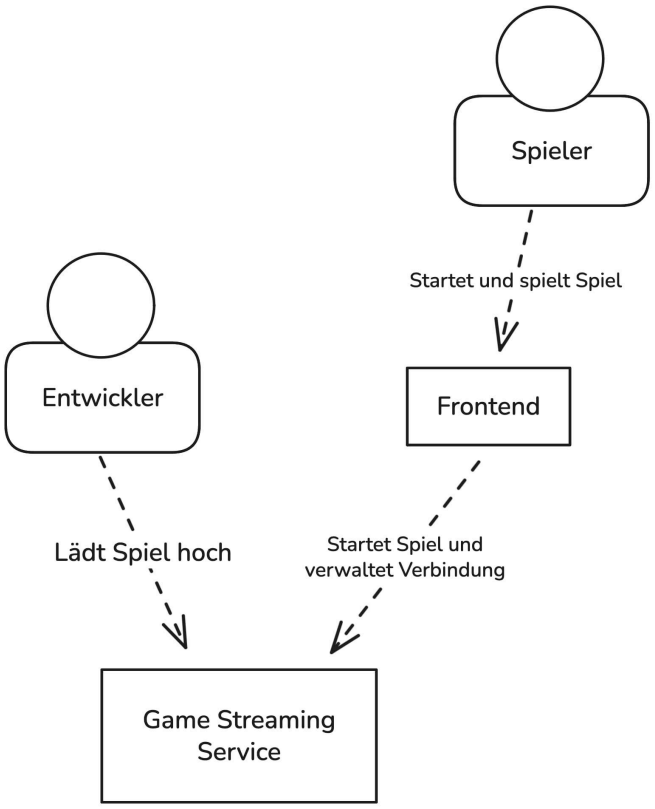

Fig. 7. C4-Modell - Context

C. C4: Container

Die Architektur basiert auf zwei zentralen, asynchronen Prozessen: der automatisierten Erstellung eines AMI und dem spielerinitiierten Start einer EC2-Instanz.

a) *Automatisierte Erstellung des AMI:*

Der Prozess wird durch die Deponierung von Spiel-Binaries in einem designierten S3-Bucket initiiert, beispielsweise durch eine CI/CD-Pipeline oder einen manuellen Upload. Die Erstellung des Objekts im S3-Bucket löst ein Object Created-Event aus, welches von einer Amazon EventBridge-Regel erfasst wird. Simple Storage Service (S3) wurde aufgrund seiner Kosteneffizienz und der Fähigkeit, große Datenmengen zu verarbeiten, als Speicherort für die Binaries gewählt. [32]

EventBridge startet daraufhin eine EC2 Image Builder-Pipeline und übergibt die Metadaten des S3-Objekts, wie Bucket-Name und Objektschlüssel, als Parameter. Die Pipeline instanziiert eine temporäre EC2-Build-Instanz basierend auf einem vordefinierten Rezept. Innerhalb dieser Instanz wird ein Skript ausgeführt, das die übergebenen Parameter nutzt, um die Binaries aus dem S3-Bucket herunterzuladen und die notwendige Software (Spiel und Streaming-Software) zu installieren.

Nach erfolgreicher Konfiguration der Instanz erstellt der Image Builder-Service ein neues AMI. Der Abschluss dieser Operation generiert ein Image Creation Complete-Event. Dieses Event wird ebenfalls von EventBridge verarbeitet, welches eine Lambda-Funktion aufruft und die ID des neu erstellten AMIs übergibt. Die Funktion persistiert diese AMI-ID zusammen mit dem zugehörigen Spielnamen in einer DynamoDB-Tabelle zur späteren Referenzierung.

b) *Spielerinitiierter und asynchroner Instanz-Start:*

Ein Spieler initiiert den Start einer Spielsitzung über eine Frontend-Applikation. Das Frontend sendet eine Anfrage an einen API-Gateway-Endpunkt, welcher die Anfrage an eine Lambda-Funktion weiterleitet. Diese Funktion führt eine Abfrage in der DynamoDB aus, um die korrekte AMI-ID für das angeforderte Spiel zu ermitteln.

Mit dieser AMI-ID instruiert die Lambda-Funktion den Amazon EC2 Service, eine neue Instanz zu starten. Der EC2-Service bestätigt die Initiierung asynchron und gibt eine InstanceId zurück. Diese ID wird von der Lambda-Funktion über das API Gateway an das Frontend propagiert.

Aufgrund der Latenz beim Start einer EC2-Instanz implementiert das Frontend einen Polling-Mechanismus. Es ruft periodisch einen zweiten API-Gateway-Endpunkt auf und übergibt dabei die erhaltene InstanceId. Eine weitere Lambda-Funktion fragt den Status der EC2-Instanz ab. Sobald die Instanz den Zustand “running” erreicht hat, liefert die Funktion die öffentliche IP-Adresse der Instanz zurück. Nach Erhalt der Verbindungsdaten etabliert das Frontend eine direkte Verbindung zur EC2-Instanz, um eine minimale Latenz im Game-Stream zu gewährleisten. [31]

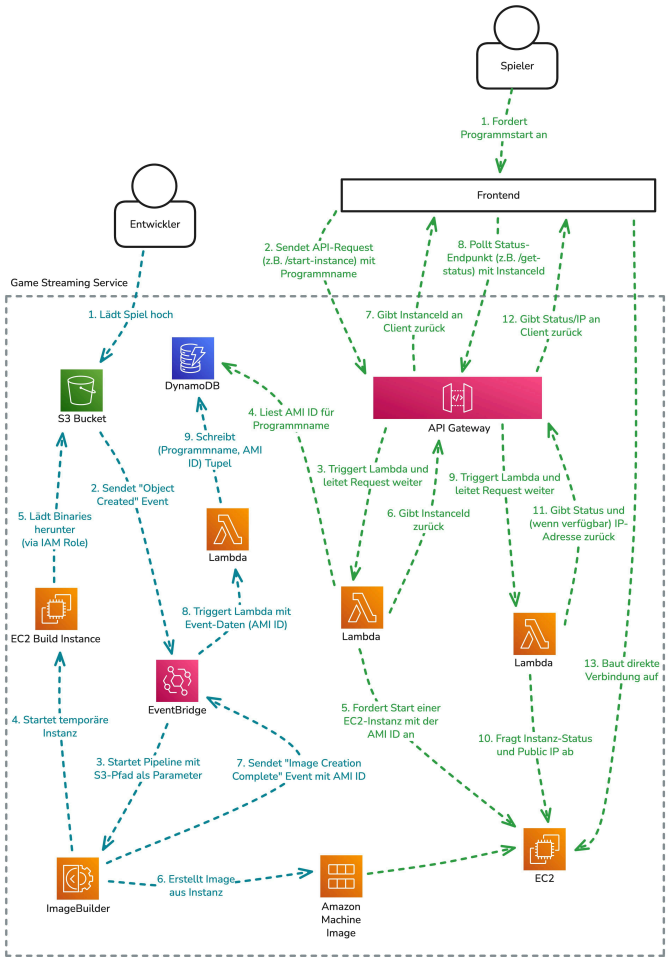

Fig. 8. C4-Modell - Container

IX. Implementierung des Streaming-Dienstes

A. Analyse der Datenströme

Zunächst wird analysiert, welche Datenarten zwischen dem Streaming-Dienst und dem Frontend ausgetauscht werden müssen. Diese lassen sich in ausgehende und eingehende Datenströme kategorisieren:

- **Ausgehende Daten (Server zu Client):**
  - Bilddaten
  - Audiodaten
  - Gamepad-Vibration (optional)
- **Eingehende Daten (Client zu Server):**
  - Steuerungseingaben (Gamepad, Maus, Tastatur, Touchscreen etc.)

B. Notwendigkeit der Datenkomprimierung

Im nächsten Schritt ist zu klären, ob eine Datenkomprimierung für die Übertragung erforderlich ist. Dabei ist zu berücksichtigen, dass der Komprimierungsprozess Latenz verursacht, welche für ein Echtzeitsystem minimiert werden muss. Zur Klärung dieser Frage wird die unkomprimierte Datenmenge des Videostroms berechnet.

Für die Berechnung werden folgende Parameter angenommen: eine Auflösung von Full-HD (1920 x 1080 Pixel), der sRGB-Farbraum (3 Byte pro Pixel) und eine Bildwiederholfrequenz von 60 Bildern pro Sekunde (FPS). Daraus ergibt sich für die reinen Bilddaten folgende Datenmenge:

$$1920 * 1080 * 3 = 6220800 \frac{\text{Byte}}{\text{Frame}} \approx 6 \frac{\text{MiB}}{\text{Frame}}$$

$$6220800 \frac{\text{Byte}}{\text{Frame}} * 60 \frac{\text{Frames}}{\text{s}} = 373248000 \frac{\text{Byte}}{\text{s}} \approx 356 \frac{\text{MiB}}{\text{s}}$$

Ein Vergleich dieses Wertes mit den in der Umfrage ermittelten verfügbaren Bandbreiten der Zielgruppe zeigt, dass bereits die unkomprimierten Bilddaten die Kapazitäten der meisten Nutzer überschreiten. Daraus folgt zwingend die Notwendigkeit einer Datenkomprimierung.

Anschließend muss differenziert werden, welche Daten komprimiert werden sollen. Bild- und Audiodaten eignen sich für eine verlustbehaftete Komprimierung, da geringfügige Informationsverluste von der menschlichen Wahrnehmung kaum bemerkt werden.

Im Gegensatz dazu erfordern die Steuerungseingaben des Spielers sowie die Befehle für die Gamepad-Vibration eine verlustfreie Übertragung, um die Integrität der Aktionen zu gewährleisten. Beide Datenarten weisen jedoch ein sehr geringes Volumen von nur wenigen Bytes pro Übertragung auf. Eine Komprimierung würde hier nur einen marginalen Nutzen erbringen oder wäre aufgrund der geringen Datenmenge pro Paket ineffektiv. Anstelle einer Komprimierung ist hier die Wahl eines effizienten Serialisierungsformats wie Protocol Buffers vorteilhafter.

Folglich konzentriert sich die weitere Betrachtung auf die Übertragung der Bild- und Audiodaten. Alle anderen Daten werden unkomprimiert, jedoch effizient serialisiert, übertragen. Wie das Video kodiert werden kann, hängt vom Client ab. Browser unterstützen üblicherweise H.264 (AVC) [33], AV1 [34], VP8, und VP9 [35].

C. Evaluierung von Übertragungsprotokollen

Herkömmliche HTTP-basierte Protokolle wie HTTP Live Streaming (HLS) und Dynamic Adaptive Streaming over HTTP (DASH) führen aufgrund ihres grundlegenden Designs, bei dem Videodaten in größeren, diskreten Segmenten gesendet werden, häufig zu höheren Latenzzeiten. [36] So sind typische HLS-Segmente mindestens 2 Sekunden lang und können, wenn sie gruppiert werden, zu effektiven Latenzzeiten von 6 Sekunden oder mehr führen. [37]

Nachfolgend werden verschiedene Übertragungsprotokolle analysiert und hinsichtlich ihrer Eignung für die definierten Anforderungen des Cloud-Gaming-Dienstes bewertet.

a) *WebRTC (Web Real-Time Communication):*

WebRTC ist ein standardisiertes Protokoll, das nativ in modernen Webbrowsern implementiert ist und dort keine externen Bibliotheken erfordert. [38] Es nutzt in erster Linie das UDP wegen seiner inhärenten Geschwindigkeit. Ergänzend zu UDP wird das RTP (Real-Time Transport Protocol) verwendet, um wesentliche Funktionen für Medienströme hinzuzufügen, darunter Zeitstempel, Sequenzierung und Zustellungsüberwachung, die für die Synchronisierung entscheidend sind. [39] UDP bietet zwar eine hohe Geschwindigkeit, aber die fehlende Zustellungsgarantie kann in sehr restriktiven NAT-Umgebungen (Network Address Translation) zu Problemen führen. [40] WebRTC stellt Peer-to-Peer-Verbindungen zwischen kommunizierenden Geräten her. Diese Architektur minimiert die Anzahl der in den Kommunikationspfad involvierten Hops, was die Latenzzeit unter idealen Netzwerkbedingungen erheblich reduziert. [38]

Es stellt eine solide Möglichkeit für Game-Streaming dar und wurde auch beim Game-Streaming-Dienst Google Stadia eingesetzt. [22]

b) *SRT (Secure Reliable Transport):*

Latenz von 300 ms bis 500 ms ist zu viel für Game-Streaming. [41]

c) *MoQ (Media over QUIC) und RoQ (RTP over QUIC):*

QUIC basierte Protokolle wie MoQ und RoQ versprechen geringere Latenzen als WebRTC und sind damit perfekt für Game-Streaming geeignet, sie sind jedoch noch sehr experimentell und es gibt keine APIs in Browsern, was die Implementierung deutlich aufwändiger macht. [42], [43]

d) *Ergebnis:*

SRT bietet eine zu hohe Latenz für Cloud-Gaming-Szenarien.

WebRTC ist sehr komplex aber eine gute und weit verbreitete Lösung. Dass der Client bereits als Browser-APIs verfügbar ist, erleichtert die Implementierung und reduziert Fehlerquellen. Dass sich Google für dieses Protokoll entschieden hat, zeigt, dass es eine gute Wahl ist.

QUIC ist vergleichsweise neu und die darauf basierenden Protokolle sind viel effizienter als WebRTC. Die Neuheit ist es auch, was es schwieriger macht, diese Protokolle zu implementieren. Dieser Arbeit fehlen die Ressourcen, um die neuen, QUIC basierten Protokolle umsetzen zu können. Dadurch, dass sowohl Server, als auch Client implementiert werden müssen, ist vor allem zu Beginn der Implementierung nicht klar, ob Probleme am Client oder Server liegen. Ein performanter Client müsste zudem in WebAssembly implementiert werden, um die zwangsläufige Latenz von JavaScript zu umgehen. Die naheliegendsten Programmiersprachen für diesen Client wären Rust, Zig und C (keine Garbage Collection).

Die Wahl fällt aufgrund der angebrachten Punkte auf WebRTC. Sobald mehr Tooling für QUIC verfügbar ist, würde sich das aber ändern.

D. Architektur

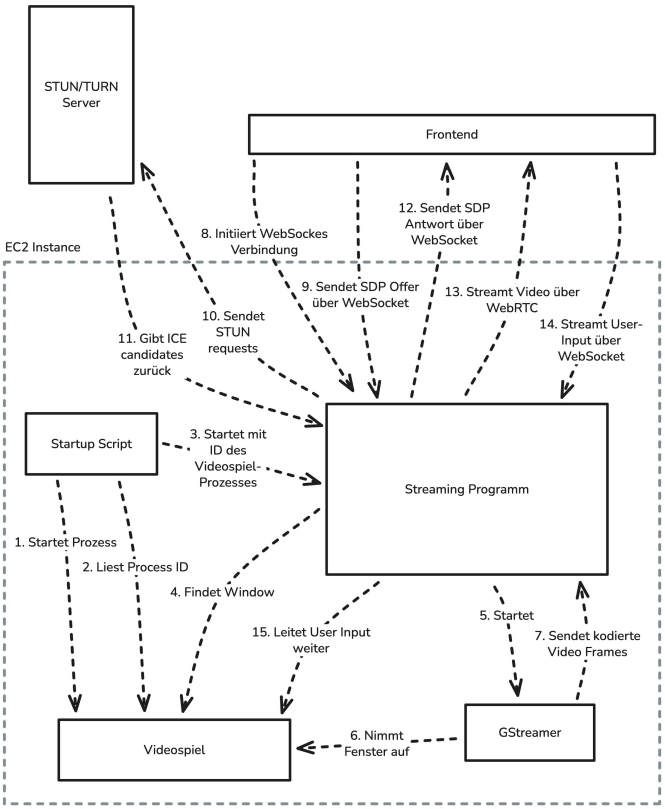

Fig. 9. C4-Modell - Component

X. Proof of Concept

Im folgenden wird ein Proof of Concept des Streaming Services mit WebRTC implementiert.

A. Videoaufzeichnung und Stream-Kodierung

Gängige Tools für diesen Zweck sind: FFmpeg und GStreamer.

Diese Arbeit benötigt ein Tool, dass verlässlich den kompletten Bildschirm oder ein einzelnes Fenster mit Ton aufnehmen kann.

Während FFmpeg auf eine große Menge an unterstützten Formaten ausgelegt ist [44], ist GStreamer auf geringe Latenz und Hardware Acceleration ausgelegt [45].

Mit dem Zeil die geringstmögliche Latenz zu erzielen, wurde GStreamer gewählt. Die vielen Formate, die FFmpeg unterstützt, wurden nicht benötigt.

B. Plattformoptimierung

Um Hardware Acceleration nutzen zu können, müssen für jede Plattform unterschiedliche Optionen für GStreamer mitgegeben werden. Die genaue Bedeutung kann in der Dokumentation von GStreamer nachgelesen werden.

Die Parameter wurden ausgewählt, indem ein Gemini 2.5 Pro RAG (Retrieval-Augmented Generation) mit der Dokumentation von GStreamer erstellt und nach optimalen Einstellungen für den Anwendungsfall Low-Latency-Stream mit modernen Browsern als Zielplattform gefragt wurde.

Das Betriebssystem (Linux, Mac, Windows) wird über conditional-compilation ausgewählt. Auf Linux wird zur Laufzeit bestimmt, ob Wayland oder X11 verwendet wird.

a) *Linux:*

X11:

ximagesrc ist eine X11-Quelle, die ein Bild von dem X11-Display erfasst. Damage wird nicht verwendet, weil Hardware-Encoder wie nvh264enc komplette Frames erwarten.

video/x-raw,framerate=60/1 ist ein Caps-Filter, der unkomprimierte, Rohe Frames mit 60Hz ausgibt.

videoconvert ist eine CPU-basierte Farb- und Format-Konvertierung (z.B. BGRx → I420).

nvideoconvert ist eine NVIDIA-spezifische Konvertierung ins "NVMM"-Speicherformat (für Hardware-Encoder).

video/x-raw(memory:NVMM),width=1920,height=1080 kodiert den im NVIDIA-Video-Memory (NVMM) vorliegenden Videostream zu Full HD (1920x1080) um.

nvh264enc ist ein NVIDIA-spezifischer Encoder, der einen H.264-Stream ausgibt. **preset=low-latency-hq** optimiert für hohe Qualität bei möglichst geringer Latenz. **tune=zerolatency** verhindert B-Frame Reordering/Buffern. **rc-mode=cbr-ld-hq** CBR (Constant Bitrate), Low Delay, High Quality. **bitrate=12000** setzt die Zielbitrate auf $12000\frac{\text{kb}}{\text{s}}$.

h264parse Parsen und ggf. Neuformatieren (Announce von SPS/PPS, NALU-Größen etc.).

rtph264pay RTP-Packetizer für H.264. **config-interval=1** Sendet SPS/PPS (Decoder-Konfigurationsdaten) alle 1 Sekunde neu. **pt=96** Dynamischer RTP-Payload-Type 96.

appsink ist ein Sink-Element, das die Daten an die Applikation (in diesem Fall den Go-Code) übergibt. Der Name **rtpsink** wird verwendet, um das Element im Code zu referenzieren.

```
ximagesrc use-damage=false !
video/x-raw,framerate=60/1 !
videoconvert !
nvideoconvert !
'video/x-raw(memory:NVMM),width=1920,height=1080' !
nvh264enc preset=low-latency-hq tune=zerolatency rc-
mode=cbr-ld-hq bitrate=12000 !
h264parse !
rtph264pay config-interval=1 pt=96 !
appsink name=rtpsink
```

Wayland:

pipewiresrc ist die standard Wayland-Quelle.

video/x-raw,format=BGRx,framerate=60/1 ist ein Caps-Filter, der unkomprimierte, Rohe Frames mit 60Hz ausgibt.

Ab hier ist die Pipeline wie bei X11 beschrieben.

```
pipewiresrc !
video/x-raw,format=BGRx,framerate=60/1 !
videoconvert !
nvideoconvert !
'video/x-raw(memory:NVMM),width=1920,height=1080' !
nvh264enc preset=low-latency-hq tune=zerolatency rc-
mode=cbr-ld-hq bitrate=12000 !
h264parse !
rtph264pay config-interval=1 pt=96 !
appsink name=rtpsink
```

b) *Mac:*

avfvideosrc AVFoundation-Source auf macOS, nimmt den Bildschirm auf.

video/x-raw,framerate=60/1 ist ein Caps-Filter, der unkomprimierte, Rohe Frames mit 60Hz ausgibt.

videoscale CPU-basiertes Skalieren des Bildes.

video/x-raw,width=1920,height=1080 Setzt die Ausgabeauflösung auf 1920×1080 Pixel.

vtenc_h264_hw Apple VideoToolbox H.264 Hardware-Encoder. **realtime=true** Optimiert für minimale Latenz (verringert interne Pufferung). **allow-frame-reordering=false** Deaktiviert B-Frame-Reihenfolge → nur I- und P-Frames → weniger Decoder-Delay. **bitrate=10000** setzt die Zielbitrate auf $10000\frac{\text{kb}}{\text{s}}$. **max-keyframe-**

interval=60 Maximal alle 60 Frames ein Keyframe (bei 60 fps → 1 Keyframe/s).

Ab h264parse ist die Pipeline wie bei Linux X11 beschrieben.

```
avfvideosrc capture-screen=true !
video/x-raw,framerate=60/1 !
videoscale !
video/x-raw,width=1920,height=1080 !
vtenc_h264_hw realtime=true allow-frame-reordering=false
bitrate=10000 max-keyframe-interval=60 !
h264parse !
rtph264pay config-interval=1 pt=96 !
appsink name=rtpsink
```

c) *Windows:*

d3d11screencapturesrc Direct3D-11–basierte Bildschirmquelle unter Windows. Liefert Frames direkt aus der GPU in D3D11-Speicher

video/x-raw(memory:D3D11Memory), framerate=60/1,width=1920,height=1080 ist ein Caps-Filter, der unkomprimierte, Rohe Frames mit 60Hz ausgibt.

d3d11convert Konvertiert D3D11-Oberflächen (Farbformat/Pixel-Layout) in ein Format, das downstream weiterverarbeitet werden kann.

Ab nvh264enc ist die Pipeline wie bei Linux X11 beschrieben.

```
d3d11screencapturesrc !
video/x-raw(memory:D3D11Memory),
framerate=60/1,width=1920,height=1080 !
d3d11convert !
nvh264enc preset=low-latency-hq tune=zerolatency rc-
mode=cbr-ld-hq bitrate=12000 !
h264parse !
rtph264pay config-interval=1 pt=96 !
appsink name=rtpsink
```

d) *Testsystem:*

Getestet wurde auf einem MacBook Pro M4 Max mit 16-core CPU und 64GB unified memory als Server und einem ... als Client.

C. Spielereingaben

Für das Proof of Concept wurde sich auf eine Steuerung mit Maus und Tastatur konzentriert. Später andere Eingabegeräte wie Gamepads einzufügen wäre kein Problem.

WebRTC unterstützen Data-Channels, die beliebige Daten versenden können, die nicht zum eigentlichen Video-Stream gehören. [46] Wenn die Verbindung über den Data-Channel nicht aufgebaut werden kann, wird WebSocket als Fallback verwendet.

Verweise

[1] S. Steinberg, *Videogame Marketing and PR*, 1st ed. New York: iUniverse, 2007.

[2] Bartman013, "List of Insomniac Games Budgets and ROI for Miles, Ratchets And Predictions for all their upcoming titles," r/GamingLeaksAndRumours. Accessed: Jun. 14, 2025. [Online]. Available: https://www.reddit.com/r/GamingLeaksAndRumours/comments/18lums9/list_of_insomniac_games_budgets_and_roi_for_miles/

[3] V. Mirpuri, "At 300 Million, This Star Wars Game Proved Too Much for Xbox to Fill the Void Left by Starfield's Delay in 2022, Reveals Leaked Email", *EssentiallySports*, Sep. 2023, Accessed: Jun. 14, 2025. [Online]. Available: https://www.essentiallysports.com/esports-news-at-three-hundred-million-dollar-this-star-wars-jedi-survivor-proved-too-much-for-xbox-to-fill-the-void-left-by-starfields-delay-in-twenty-twenty-two-reveals-leaked-email/

[4] "God of War Ragnarök." Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/2322010/God_of_War_Ragnark/

[5] "FINAL FANTASY VII REBIRTH." Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/2909400/FINAL_FANTASY_VII_REBIRTH/

[6] "The Last of Us Part II Remastered." Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/2531310/The_Last_of_Us_Part_II_Remastered/

[7] Statista Research Department, "Marktanteile der führenden Unternehmen am Umsatz im Bereich Cloud Computing weltweit im 1. Quartal 2025." Accessed: Jul. 13, 2025. [Online]. Available: https://de.statista.com/statistik/daten/studie/150979/umfrage/marktanteile-der-fuehrenden-unternehmen-im-bereich-cloud-computing/

[8] Google Support, "Stadia Announcement FAQ." Accessed: Jun. 15, 2025. [Online]. Available: https://support.google.com/stadia/answer/12790109

[9] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2012, pp. 1–6.

[10] NVIDIA Corporation, "DLSS 4: Transforming Real-Time Graphics with AI," Technical Report, 2025. Accessed: Jun. 15, 2025. [Online]. Available: https://research.nvidia.com/labs/adlr/DLSS4/

[11] Republic Benchmarks, *The Last of Us Part II Remastered | Gameplay Benchmark | I5-11400 + RTX 3060 | 1080p | 1440p*, (Apr. 03, 2025). Accessed: May 16, 2025. [Online Video]. Available: https://youtu.be/Kz3JjY0p6-4

[12] P. Alcorn, "AMD deprioritizing flagship gaming GPUs: Jack Huynh talks new strategy against Nvidia in gaming market," Tom's Hardware. Accessed: Jun. 15, 2025. [Online]. Available: https://www.tomshardware.com/pc-components/gpus/amd-deprioritizing-flagship-gaming-gpus-jack-hyunh-talks-new-strategy-for-gaming-market

[13] jaykihn0, "Died in Q3 of last year.." Accessed: Jun. 15, 2025. [Online]. Available: https://x.com/jaykihn0/status/1905055296839700601

[14] "STAR WARS Jedi: Survivor." Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/1774580/STAR_WARS_Jedi_Survivor/

[15] “Assassin’s Creed Shadows.” Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/3159330/Assassins_Creed_Shadows/

[16] “The Elder Scrolls IV: Oblivion Remastered.” Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/2623190/The_Elder_Scrolls_IV_Oblivion_Remastered/

[17] “Black Myth: Wukong.” Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/2358720/Black_Myth_Wukong/

[18] “Dragon Age: The Veilguard.” Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/1845910/Dragon_Age_The_Veilguard/

[19] “Cyberpunk 2077.” Accessed: May 16, 2025. [Online]. Available: https://store.steampowered.com/app/1091500/Cyberpunk_2077/

[20] “Steam-Downloadstatistiken.” Accessed: Jul. 07, 2025. [Online]. Available: https://store.steampowered.com/stats/content/?l=german

[21] SteamDB, “Stellar Blade Demo - Steam Charts.” Accessed: Jun. 16, 2025. [Online]. Available: https://steamdb.info/app/3564860/charts/

[22] A. Di Domenico, G. Perna, M. Trevisan, L. Vassio, and D. Giordano, “A network analysis on cloud gaming: Stadia, geforce now and psnow,” *Network*, vol. 1, no. 3, p. 7, 2021.

[23] “Streaming Preferences Explained (TCP & UDP).” Accessed: Jul. 17, 2025. [Online]. Available: https://support.shadow.tech/hc/en-us/articles/32731845984145-Streaming-Preferences-Explained-TCP-UDP

[24] J. Whitehead, “Streaming Protocols for Live Broadcasting: Everything You Need to Know [2025 Update].” Accessed: Jul. 17, 2025. [Online]. Available: https://www.dacast.com/blog/streaming-protocols/

[25] Доктор Веб, “The hunt for vulnerability: executing arbitrary code on NVIDIA GeForce NOW virtual machines,” Habr. Accessed: Jul. 17, 2025. [Online]. Available: https://habr.com/en/companies/drweb/articles/518052/

[26] J. Albano, “A First Look at Google Stadia (Part 1),” DZone. Accessed: Jul. 17, 2025. [Online]. Available: https://dzone.com/articles/a-first-look-at-google-stadia

[27] FredreikSchack, “Isolation of tasks using GPU (not at the same time).” Accessed: Jul. 17, 2025. [Online]. Available: https://forum.proxmox.com/threads/isolation-of-tasks-using-gpu-not-at-the-same-time.131575/

[28] Amazon Web Services, “Amazon EC2 G5 Instances.” Accessed: Jun. 16, 2025. [Online]. Available: https://aws.amazon.com/ec2/instance-types/g5/

[29] Vantage, “g5.2xlarge EC2 Instance - Vantage.” Accessed: Jun. 16, 2025. [Online]. Available: https://instances.vantage.sh/aws/ec2/g5.2xlarge?region=eu-central-1&platform=linux&duration=hourly

[30] “EC2 Instance Quotas.” Accessed: Jul. 10, 2025. [Online]. Available: https://docs.aws.amazon.com/ec2/latest/instancetypes/ec2-instance-quotas.html

[31] A. Fei, G. Pei, R. Liu, and L. Zhang, “Measurements on delay and hop-count of the internet,” in *IEEE GLOBECOM’98-Internet Mini-Conference*, 1998.

[32] O. S. Abiodun and K. S. Olanrewaju, “Design and Implementation of a Scalable Cloud-Based File Sharing System Using Amazon Elastic Compute (EC2) and Amazon Simple Storage (S3),” *International Journal of Advance Research Publication and Reviews*, vol. 2, no. 6, p. 406, Jun. 2025, Accessed: Jul. 10, 2025. [Online]. Available: https://www.ijrpr.com/

[33] “MPEG-4/H.264 video format | Can I use... Support tables for HTML5, CSS3, etc.” Accessed: Jul. 13, 2025. [Online]. Available: https://caniuse.com/mpeg4

[34] “AV1 video format | Can I use... Support tables for HTML5, CSS3, etc.” Accessed: Jul. 13, 2025. [Online]. Available: https://caniuse.com/av1

[35] “WebM video format | Can I use... Support tables for HTML5, CSS3, etc.” Accessed: Jul. 13, 2025. [Online]. Available: https://caniuse.com/webm

[36] M. Hakimi, “Low Latency Streaming.” Accessed: Jul. 18, 2025. [Online]. Available: https://www.ioriver.io/terms/low-latency-streaming

[37] “Video Latency: What Is It & How Does It Matter in Streaming?.” Accessed: Jul. 18, 2025. [Online]. Available: https://castr.com/blog/what-is-video-latency/

[38] B. Sredojev, D. Samardzija, and D. Posarac, “WebRTC technology overview and signaling solution design and implementation,” in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1006–1009. doi: 10.1109/MIPRO.2015.7160422.

[39] Flussonic, “Low-Latency WebRTC Streaming: Real-Time Video at Scale,” Flussonic Blog. Accessed: Jul. 18, 2025. [Online]. Available: https://flussonic.com/blog/article/low-latency-webrtc-streaming

[40] Nabto, “What is NAT Traversal in WebRTC?.” Accessed: Jul. 18, 2025. [Online]. Available: https://www.nabto.com/what-is-nat-traversal-in-webrtc/

[41] OSSRS Community, “SRT.” Accessed: Jul. 18, 2025. [Online]. Available: https://ossrs.net/lts/en-us/docs/v5/doc/srt

[42] Z. Gurel, T. Erkilic Civelek, A. Bodur, S. Bilgin, D. Yeniceri, and A. C. Begen, “Media over QUIC: Initial Testing, Findings and Results,” in *14th ACM Multimedia Systems Conference*, New York, NY, USA: Association for Computing Machinery, 2023, pp. 301–306. doi: 10.1145/3587819.3593937.

[43] D. Mejías, I. Yeregui, Á. Martín, R. Viola, P. Angueira, and J. Montalbán, “Streaming Remote rendering services: Comparison of QUIC-based and WebRTC Protocols.” Accessed: Jul. 13, 2025. [Online]. Available: https://arxiv.org/abs/2505.22132

[44] “FFmpeg Formats.” Accessed: Aug. 03, 2025. [Online]. Available: https://ffmpeg.org/ffmpeg-formats.html

[45] G. K, “Gstreamer vs ffmpeg latency: Choosing the right streaming solution for your business.” Accessed: Aug. 03, 2025. [Online]. Available: https://www.byteplus.com/en/topic/179638?title=gstreamer-vs-ffmpeg-latency-choosing-the-right-streaming-solution-for-your-business

[46] “Data channels.” Accessed: Aug. 06, 2025. [Online]. Available: https://webrtc.org/getting-started/data-channels

Danksagung

Mein aufrichtiger Dank gilt den Teilnehmerinnen und Teilnehmern meiner Umfragen, deren engagierte Mitwirkung eine unverzichtbare Grundlage für die empirischen Ergebnisse dieser Arbeit bildete. Des Weiteren möchte ich Herrn Prof. [Nachname des Betreuers] für die Möglichkeit danken, dieses interessante Thema im Rahmen meiner Bachelor-Arbeit bearbeiten zu dürfen.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe, dass ich sie zuvor an keiner anderen Hochschule und in keinem anderen Studiengang als Prüfungsleistung eingereicht habe und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht.

Bei der sprachlichen Formulierung kam das Large Language Model ‘Gemini 2.5 Pro’ von Google unterstützend zum Einsatz. Es diente ausschließlich der Verbesserung der Ausdrucksweise und Klarheit der Darstellung.

Beim Schreiben von Code wurde SST ‘Open Code’ mit ‘Claude 4.1 Opus’ von Anthropic verwendet. Dabei stammen alle Ideen und Pläne der Implementation von mir. Das LLM hat unmissverständliche Spezifikationen ohne Interpretationsspielraum bekommen, die es implementiert hat.

Der gesamte Inhalt, die wissenschaftlichen Aussagen und die Forschungsergebnisse dieser Arbeit wurden eigenständig von mir entwickelt und verantwortet und sind in keiner Weise durch die Nutzung der Sprachmodelle beeinflusst worden.