information

Review

A Systematic Mapping Study of MMOG Backend Architectures

Nicos Kasenides and Nearchos Paspallis *

School of Sciences, University of Central Lancashire—Cyprus Campus (UCLan Cyprus), 7080 Larnaka, Cyprus
* Correspondence: npaspallis@uclan.ac.uk; Tel.: +357-2469-4091

Received: 26 June 2019; Accepted: 19 August 2019; Published: 21 August 2019

Abstract: The advent of utility computing has revolutionized almost every sector of traditional software development. Especially commercial cloud computing services, pioneered by the likes of Amazon, Google and Microsoft, have provided an unprecedented opportunity for the fast and sustainable development of complex distributed systems. Nevertheless, existing models and tools aim primarily for systems where resource usage—by humans and bots alike—is logically and physically quite disperse resulting in a low likelihood of conflicting resource access. However, a number of resource-intensive applications, such as Massively Multiplayer Online Games (MMOGs) and large-scale simulations introduce a requirement for a very large common state with many actors accessing it simultaneously and thus a high likelihood of conflicting resource access. This paper presents a systematic mapping study of the state-of-the-art in software technology aiming explicitly to support the development of MMOGs, a class of large-scale, resource-intensive software systems. By examining the main focus of a diverse set of related publications, we identify a list of criteria that are important for MMOG development. Then, we categorize the selected studies based on the inferred criteria in order to compare their approach, unveil the challenges faced in each of them and reveal research trends that might be present. Finally we attempt to identify research directions which appear promising for enabling the use of standardized technology for this class of systems.

Keywords: cloud computing; software architecture; Platform-as-a-Service; massively multiplayer online games

1. Introduction

Cloud computing has revolutionized almost all aspects of software production. Especially when it comes to widely and highly available services, cloud computing offers many competitive solutions, with numerous advantages [1]. While the term *cloud computing* was coined many years ago [2], it is still widely used to describe a range of technologies. These are commonly classified in layers such as *Infrastructure-as-a-Service* (IaaS), *Platform-as-a-Service* (PaaS), *Software-as-a-Service* (SaaS), and so forth. Naturally, the Web is the *killer app* of cloud computing, which explains to a large extent the design of some PaaS-based frameworks—for example, Google's AppEngine [3] is largely based on the *servlet* which was itself designed to facilitate large-scale Web-based systems [4].

At the same time, resource-intensive applications—such as *Massively Multiplayer Online Game* (MMOG) backends—have certain peculiarities and distinct requirements when compared to other types of software—such as web applications—which have long been deployed on the cloud. This study focuses on MMOGs as this class of applications highlights a unique set of properties: a very large number of users and a very large system state—normally spanning across several computing nodes—and a high likelihood of conflicting resource usage. While we focus on MMOGs, most findings also apply to other large scale systems such as scientific, educational, training or military simulations.

The characteristics of this type of systems present challenges to developers who aim to build and deploy these on cloud-based infrastructure. Despite these challenges, recent advancements enable more and more commercial MMOGs to be hosted on the cloud.

This paper presents a systematic mapping study of the state-of-the-art in MMOG development and deployment. First, we investigate the scope and approach of several related publications and identify aspects that have been deemed important by researchers of MMOG development. We use these aspects as criteria and perform a categorization of the selected studies with respect to each criterion, which allows us to compare these approaches. The objective of this comparison is to pick out the advantages, disadvantages and challenges faced in each approach—which may allow developers to choose the one that best fits their needs. More importantly, we aim to reveal the research trends in terms of the approaches used to enable MMOG backends by spotting emerging patterns, especially in recent years. These trends can indicate potential directions for future research towards bringing MMOGs closer to the cloud.

The rest of this paper is organized as follows: Section 2 provides the motivation for this paper and Section 3 describes the methodology used in this study, including its research objectives. Then, Section 4 presents the list of criteria selected for evaluating and comparing the selected papers. Section 5 provides a detailed discussion of the state-of-the-art for developing and deploying MMOG backends. These approaches are then analyzed and discussed in Section 6, along with identified opportunities and future trends. Finally, the paper closes with conclusions in Section 7.

2. Motivation

Multiplayer games, especially MMOGs have attracted significant attention over the last decade. As Ducheneaut et al. [5] put it, "*[these games] took the world of online gaming by storm*" [5]. Hugely successful games were facing first-of-a-kind challenges in terms of accommodating the high influx of players as well as the increasing game complexity. For instance, World of Warcraft, one of the most popular MMOGs, had "*to break down the game's large subscriber base into more manageable units [...]. Each server [could] host a community of about 20,000 players*" ([5], p. 284). Naturally, the potential of using cloud computing is huge and to some extent an inescapable choice, especially for smaller game studios or individuals opting for an MMOG offering.

Other types of applications, such as enterprise and web applications have been deployed on the public cloud for years and at different layers such as IaaS, PaaS and SaaS. However, despite MMOGs sharing many common challenges, most developers traditionally opt for their own custom solutions which are deployed mostly on-premise and in few cases, cloud IaaS environments. This study is motivated by the existence of a gap, in terms of the technology used to enable MMOGs. We aim to show that by utilizing higher-level solutions, in the layers of PaaS or BaaS, developers can adopt utility computing's benefits to MMOG backends.

To do so we identify the challenges that hinder the development of MMOGs using more standardized software technologies. Our objective is to eventually address this gap by providing the models, methods and tools to allow the development of MMOGs on commercial cloud platforms. Utilizing such environments would provide advantages to game providers, software engineers and players which are not present in the existing methods currently used. Existing surveys related to MMOG backends have been focused on specific infrastructures or architectures and did not look at the greater landscape of possible solutions [6–11]. On the other hand, relevant cloud computing and software engineering surveys do not explicitly discuss MMOGs, but instead, cover a broader range of applications. Hence, to the best of our knowledge, there is no existing study that enables us to review multiple approaches in MMOG development. From a development perspective, we believe that this study will be of interest to game providers and software engineers working on MMOGs. Similarly, researchers working on resource-intensive applications, MMOGs or otherwise, may find the findings of this study useful. Through this study, all interested parties can identify the challenges of MMOG backend development. Furthermore, the report aims to provide insight into the existing approaches and their advantages/disadvantages with regard to these challenges. This study also provides a categorization and comparison of these approaches which could assist developers in choosing which one to use based on their context. Finally, the study aims to identify new areas that are less explored and could be fertile for further research and development.

3. Methodology

This paper is a systematic mapping study focusing on software technology used to realize the backend for MMOGs. This includes the types of infrastructure and architecture that can be used to enable such complex systems, as well as how their performance, security and data persistence can be improved. Our methodology follows the protocol described by Keele et al. [12], which allows the "*identification of evidence clusters and evidence deserts to direct the focus of future systematic reviews and to identify areas for more primary studies to be conducted*" [12].

3.1. Research Questions

Our main objective is to explore the state-of-the-art in the area of software architectures used to enable MMOG backends, to address these research questions:

1. What are the main challenges in developing MMOG backends?
2. Which criteria are the most relevant to categorize studies into groups?
3. What are the research trends over time, in terms of the approaches used for MMOG backends?
4. Are there indications of alternative, promising research directions for realizing MMOG backends?

3.2. Search Strategy

Our search for relevant literature was conducted online, using standard services and digital libraries. The search was partitioned in three rounds. First, we searched on Google Scholar for keywords or combinations of keywords relevant to the topic, such as *MMOG, Massively, Multiplayer, Online, Games, Gaming, Client-Server, Peer-to-peer—or P2P, Architecture, Infrastructure, Distributed, Cloud, On-premise, Development, Software* and so forth. In the second round, we searched electronic libraries such as *ACM, Springer, IEEE, ScienceDirect* and *Elsevier*. A search for the same keywords as in the first round was conducted, which revealed some additional resources. Finally, our third round targeted relevant publications *referenced* from the original list of papers.

3.3. Selection Criteria

To prioritize and select among the identified papers, we used a combination of *inclusion* and *exclusion* criteria. The inclusion criteria helped ensure that the selection process was objective and the exclusion ones helped limit irrelevant papers. These criteria are summarized in Table 1.

Table 1. Inclusion and Exclusion Criteria used in the study.

Inclusion Criteria	**Exclusion Criteria**
The paper relates to MMOGs or similar large distributed systems.	The paper does not relate to software engineering or software architecture or software technology.
The paper relates to cloud technology which could be applied to developing MMOG backends.	The paper does not provide details for any of the areas of interest in this study.
The paper touches at least one of the identified research questions, either directly or indirectly.	The paper is not relevant to any of the targeted research questions.
	The paper was not published in a peer-reviewed journal or conference proceedings.
	The full paper is not available for download.

Review Process

A total of 161 resources were considered during our study. Each publication was reviewed by reading at least its abstract and given a level of significance according to our research. The level of significance appointed to each paper was determined from the inclusion criteria described above. For example, a paper meeting all of the inclusion criteria is regarded as extremely useful to our research objectives. Our first review excluded 70 out of 161 papers. We then conducted a second round of reviews, during which we read the full text of each paper and checked its significance to our research questions, as well as its compatibility with the inclusion/exclusion criteria. A final set of 38 papers resulted from this review round.

3.4. Data Collection

The research questions listed above were used to guide the data extraction process from the selected set of papers. The following list shows how the research questions were *expressed* so they could be used to extract meaningful data from each paper. While the first three research questions could be answered by studying each paper individually, the 4th one required a review approach. Therefore, answering questions 1-3 first is crucial to identify any patterns that emerge from all of the papers. In addition, we grouped similar publications released by the same authors into groups so that the data provided by them came into context.

1. Do the authors identify any challenges in their methodologies? Are these relevant to the study?
2. Is the paper focused on a certain area of MMOG development/deployment? How are the described methods evaluated?
3. What is the approach utilized by the authors to implement the backend of their MMOG? Do they use any specific tools? In what context was their study conducted? When was it published?
4. When data from all studies are collected and analyzed, is there any emerging correlation between time and the methods used? Are there any *gaps* that have not been explored so far?

4. Selection of Criteria

As one of the main goals of this paper is to identify and compare the *state-of-the-art* approaches in developing MMOG backends, we start by identifying criteria to guide the comparison. These are naturally extracted from the studied papers. Table 2 summarizes the identified areas, along with the number of papers that are related to them, sorted by frequency in descending order. Another study conducted by Chu [13] is consistent with these results.

Table 2. Frequency of the criteria in descending order. Note that some papers focus on multiple areas and in those cases, the papers were added to all applicable areas.

Area of Study	Frequency of Papers
Infrastructure	11
Architecture	10
Performance	7
Scalability	6
Persistence	4
Security	3

To allow a meaningful comparison, we break each criterion into sub-categories and then classify each referenced paper in one or more of them, as discussed in the following subsections and summarized in Table 3.

4.1. Infrastructure

This criterion is used to identify the type of infrastructure used in each study. This can be broadly defined as a set of technology components (mainly physical components such as servers and other networking hardware) which lie at the foundation of an MMOG's service. This criterion is further broken into four sub-categories:

- *Dedicated (D)*: Use of network facilities that are specifically purposed to enable one type of application and require direct management at the hardware level.
- *Private clouds (PrC)*: Use of proprietary clouds that are built and maintained privately and can offer higher availability, scalability etc.
- *Public clouds (PuC)*: Use of public clouds which are owned by a third party. Services are leased to the game provider for a given price model. These are further categorized as IaaS and PaaS.
- *Hybrid clouds (HC)*: In the context of this paper we define a hybrid cloud as a combination of private and public clouds to enable MMOGs.
- *Unknown (U)*: Unknown—that is, no information could be identified.

4.2. Architecture

The architecture criterion identifies which communication architecture is being used in each paper. Architectures define how the components of a system are configured to interact with each other to partition and execute the given workload.

- *Client Server (CS)*: This approach offloads most of the workload on a central server that performs processing and provides data to requesters called clients.
- *Peer-to-peer (P2P)*: P2P splits the workload among equipotent peers in a network and uses algorithms to synchronize processing. Each peer preforms a part of the workload and can send data to other peers.
- *Hybrid (H)*: A hybrid approach utilizes both the client-server and peer-to-peer architectures at different levels in the architecture.
- *Unknown (U)*: Unknown—that is, no information could be identified.

4.3. Performance

The performance criterion looks at how each study conducted an evaluation of the performance of their proposed solution. It primarily identifies the methods that were used for the evaluation as well as the type of tools that have been used to carry it out. Thus, we provide categories first based on the *types of performance evaluation*:

- *Simulation (S)*: The authors have used computer simulations to conduct their experiments and evaluate the performance of their solutions.
- *Modelling (M)*: The authors provide mathematical or computational models and utilize them to predict the performance of their solution.
- *Unknown (U)*: Unknown—that is, no information could be identified.

We further identify categories based on the *tools used in performance evaluation*:

- *Pre-existing MMOGs (P)*: Studies that have been identified to use pre-existing MMOGs to carry out their evaluation.
- *Other (O)*: Use of other types of applications or case-studies to carry out an evaluation.

4.4. Scalability

The scalability criterion measures how efficiently a system is able to serve a changing number of concurrent users. We use this criterion to measure not only how versatile a system is in terms of the number of users it can support but also how quickly and easily it can be configured to do so.

- *Not scalable (NS)*: This category describes a system that cannot support a varying number of users (and thus workload).
- *Manually Scalable (MS)*: These systems can respond to changing workloads but depend on supervision from a system administrator—and usually the manual procurement and installment of hardware.
- *Elastic—or automatically scalable—(E))*: Such solutions respond to changing workloads by automatically allocating resources without any supervision.
- *Unknown (U)*: Unknown—that is, no information could be identified.

4.5. Persistence

This criterion categorizes the types of persistence that the related papers have utilized in their implementations. For the papers that have mentioned the use of a database system we have identified a diverse set of approaches, all of which fall in one of the two categories below:

- *Relational Databases (R)*: Relational Database Management Systems (RDBMSs) store data as rows and columns in a table and use the Structured Query Language (SQL) to describe relationships between them.
- *NoSQL (N)*: Non-relational database systems which have no schema and rely on collections of data instead of tables.
- *Unknown (U)*: Unknown—that is, no information could be identified.

Table 3. Criteria and categories used for classifying approaches for developing Massively Multiplayer Online Game (MMOG) backends.

Criterion	**Categories**

Infrastructure	D = Dedicated
	PrC = Private Cloud
	HC = Hybrid Cloud
	PuC = Public Cloud
	U = Unknown

Architecture	CS = Client-Server
	P2P = Peer-to-peer
	H = Hybrid architecture
	U = Unknown

Performance	Types of simulation:
	S = Simulation
	M = Modelling
	U = Unknown

	Testing software: P = Pre-existing MMOGs O = Other types of applications

Scalability	NS = Not scalable
	MS = Manually scalable
	E = Elastic (automatically scalable)
	U = Unknown

Persistence	R = Relational Databases
	N = NoSQL
	U = Unknown

Security	L = Loose security (not controlled by architecture)
	T = Tight security (controlled by architecture)
	U = Unknown

4.6. Security

The security criterion differentiates the practices the authors have used to secure their solutions against security threats.

- *Loose security (L)*: Defines practices to secure a solution that are not controlled at the architecture level.
- *Tight security (T)*: Defines security practices that are controlled at the architecture level.
- *Unknown (U)*: Unknown—that is, no information could be identified.

4.7. Other Criteria

Besides the criteria already identified in Section 4, some more are also deemed relevant and discussed in this section. For instance, a report by Google [14] proposes several auxiliary features which are typical in game platforms: *Leaderboards*, *Matchmaking*, *Online lobby and chat*, *Authorization*, *Party/Group/Team formation*, *User and group profiles*, *Social feeds*, *Analytics* and *Presence* [14].

In addition to these, Shaikh et al. [15] mention the use of a developer portal, through which "*[developers] can interact with the game platform*" [15]. This portal can be used to deploy a game with specific policies for server requirements, make queries about various information generated by games and manage their settings. Similarly, game administrators can use such a portal to "*get information about platform conditions*" such as network utilization, server availability and game statistics. Furthermore, the authors talk about an auxiliary *game deployment service*, which uses a web-based publisher which accepts a simple policy information and an upload of the game software and installation script. The deployment server uses this data to automatically deploy a game by creating a new *Data Center Model* (DCM) that "*represents logical and physical assets under its management*". The DCM uses a cluster with a specific software stack. Lastly, Shaikh et al. [15] also describe game content distribution identifying it as a problem that requires some mediation. Due to the high frequency and size of patches released for each game and to the *flash-crowd* nature of these downloads, game content distribution incurs extra bandwidth cost to the servers. The authors utilize a P2P architecture "*in order to deliver content to users quickly while preserving bandwidth at the publisher's content servers*", specifically for integrating Quake II with BitTorrent clients.

5. Approaches for Developing and Deploying MMOG Backends

Game developers have used a wide range of approaches to enable online multiplayer games since they became popular in the 1990s. In this section, we survey the state of the art in the deployment of MMOGs and identify the unique attributes of these applications. Based on these, we categorize each approach according to the specified criteria so they can be easily compared. For this paper, we focus on the realization of the backend aspects of MMOGs.

5.1. Infrastructure

5.1.1. Dedicated Infrastructure

The conventional approach to enabling MMOGs is for game providers to install, configure and maintain a dedicated or private infrastructure for each game [15]. Such large, static infrastructure is used by games like *World of Warcraft* (WoW), a *Massively Multiplayer Online Role Playing Game* (MMORPG) as described in Reference [16]. MMORPGs like WoW require hundreds or sometimes thousands of servers to run. For game developers, creating, operating and maintaining such a large infrastructure is in itself a challenge. This challenge highlights the need to find the best possible hosting solution, take risks by making the necessary investments and then tackling infrastructure issues manually [17].

Chu [13] describes a dedicated infrastructure built to enable MMOGs as one consisting of four main components: *database servers*, *Game servers (public and secure)*, *Game clients* and a *Web application* *server* that integrates game servers with clients [13]. In this system, different types of tasks such as database transactions, game logic and so forth, are distributed to different machines on the network with each machine carrying out a specific set of tasks. Secure servers are used to handle incoming HTTP requests but are only accessible through other game servers whereas public servers interface the secure servers to provide access only to the appropriate functions for the clients.

Another study proposes to improve the dedicated infrastructure approach by utilizing a dedicated cluster of servers which enables their MMORPG services to overcome scalability issues [18]. The authors identify scalability and availability limitations as the main problems of the dedicated infrastructure approach.

5.1.2. Private/Proprietary Clouds

In recent years, a fairly decent amount of studies have carried out research in the use of private/proprietary clouds for hosting MMOGs. We distinguish a private/proprietary cloud from dedicated infrastructure when the infrastructure attains the following attributes [19]:

- High availability: Cloud systems offer high availability, usually 99% or higher.
- Elasticity: In cloud-based systems, there are enough resources and provisioning policies for the system to scale up or down, following the on-demand model.
- Virtualization: Access to the infrastructure is not done physically/directly but through a virtualization medium.
- Automation: A cloud system automates a number of administrative processes such as storage configuration, security policies and more.

Nae et al. [20] propose a dynamic resource provisioning solution based on private cloud infrastructure, which aims to address the over-provisioning of resources and "*low-cost market joining problems*" encountered in the dedicated infrastructure approach [20]. They utilize geographically distributed data centers which can serve game operators simultaneously. Game operators can submit requests to these data centers for resources (such as CPU, memory, etc.) Depending on each data center's service model, requests are either served immediately or queued. When a request is scheduled, resources are allocated (or provisioned) to the game operators. In their study, the authors also discuss the use of virtualization for such cases. They argue that "*compute clouds provide generic functionality for on-demand hosting and provisioning of resources*" and that "*cloud computing [has] the potential to eliminate the scalability barriers in MMOG hosting through scaling by credit card*". This suggests that limitations are imposed on a cloud-based system not because of limits in resources but because of the costs associated with them. Adding to that, the authors discuss the potential benefits of virtualization in the context of hosting MMOGs on a private IaaS environment. Even though virtualization adds "*a certain degree of overheads*", creating virtual machine (VM) images with the required software is relatively time-efficient because these can be "*deployed on all supported platforms*". Arguing in favor of this approach, the study attests that "*the advantages of virtualization are rather important when using heterogeneous computing resources*".

Another study [17] describes a platform for online games that also follows the on-demand paradigm. The platform is based on open standards, off-the-shelf software and utilizes virtualization of resources from a cluster of servers configured to host MMOGs. The platform uses a layered architecture which ranges from low-level infrastructure to high level layers that act as application-level services. At the lowest level, the infrastructure consists of shared clusters of game servers, database servers, content servers and so forth. This can be expanded as the number of players grows so that games can be deployed on multiple hosts. The system layer implements non-game-specific functions such as network monitoring and server provisioning. These functions can be customized and can contain code from different software stacks. A novelty presented in this article is the *Provisioning Manager* (PM), a service that manages and automatically provisions game servers by collecting performance and availability metrics. The PM responds to changes in demand by implementing data and performance models that allow it to make decisions on when and how to allocate or de-allocate resources. The use of the PM, load prediction models and standardization of installation and configuration work-flows classify this solution as cloud-based.

Dhib et al. [21] utilize a private cloud infrastructure to conduct tests about *Quality of Experience* (QoE) in MMOGs [21]. The physical servers are distributed over multiple data centers with each physical server hosting several VMs. Consequently, each VM runs a game server that is responsible for "managing a limited area" of an MMOG's world. The authors use a pricing model based on Amazon's EC2 to calculate potential costs on the allocated resources. Their experiments concluded that by using their method "*the cost per player decreases smoothly when more players get connected*" while the QoE remains at a "*minimal threshold*".

5.1.3. Public/Commodity Clouds

In a public cloud, the infrastructure is owned by and exists on the premises of a cloud provider. The cloud provider may choose to make cloud services available to the public for a given price model. Popular organizations owning public clouds are Amazon, Google and Microsoft each having a large market share and providing a large variety of services and features.

An article by Mishra et al. [22] describes public clouds as platforms that "aim to realize economies of scale and increased utilization by sharing resources or services as available through technologies such as virtualization and multitenancy" and provides popular examples at the IaaS layer such as *Amazon Elastic Compute Cloud* (EC2), *Google Compute Engine*, *Windows Azure Cloud Services*, *Rackspace* and so forth [22].

Mishra et al. [22] argue that in the context of MMOGs, utilizing public clouds requires techniques for offloading computational and data-intensive tasks from client devices to cloud servers. An important issue when working with public clouds is that even though they provide resources at scale, the cloud provider's data centers are rarely located close to users—"resulting in large communication latency in the network infrastructure". Several solutions such as Cloudlets [23] alleviate this problem by providing specific resources closer to the player.

Najaran and Krasic [24] discuss the ability of public clouds to provide resources at a much lower cost compared to private clouds or dedicated infrastructure: "renting resources for a game with hundreds of players will cost just a few cents per player per hour" [24]. For MMOGs specifically, the author argues that Amazon's EC2 infrastructure coupled with the peer-to-peer architecture enables games to be "distributed amongst multiple nodes" allowing fast-paced types of games like *First Person Shooter* (FPS) to be hosted on a public cloud. To evaluate this approach, the authors created a simple FPS game with computer-controlled players randomly walking in a virtual world. By evaluating both the performance of client machines and the Amazon EC2 cloud as well as the system's ability to scale, the author concludes that this approach scales "*an order of magnitude more players than state of the art FPS game servers currently support*". Regarding latency, the worst-case scenario report was in the order of "*tens of milliseconds*".

Another study explores the trade-off between resource allocation cost and delay. Dhib et al. [25] propose an architecture for MMOGs to address the problem of dynamic resource provisioning. They utilize a three-layer MMOG architecture on top of public cloud infrastructure composed of physical and virtual machines and game services that run games. Their approach uses a dynamic resource allocation approach that only allocates resources when necessary to minimize the allocation cost and keep the delay under a specific threshold. Experiments show that their approach can maintain this trade-off when compared to other strategies such as over-provisioning and under-provisioning.

Dhib et al. [26] further enhance their previous research by developing a new model that captures the intrinsic trade-off between response delays and their corresponding costs. The authors propose a virtual machine placement algorithm which approximates the trade-off point of delay and allocation cost. Using a Matlab tool, they evaluate their placement algorithm and compare it against Random, Greedy and Minimum allocation cost algorithms. The results from this more recent research show that their solution is effective in maintaining the balance between delay and allocation cost. Research conducted in References [25,26] has contributed significantly in making the use of cloud infrastructure more appropriate for this type of applications.

At the PaaS layer, Zahariev [27] describes Google's App Engine (GAE) as a system which allows the development of server-side applications on Google's scalable infrastructure [27]. The PaaS layer allows developers to upload their applications on public provider's infrastructure and deploy it immediately. A critical advantage of the PaaS layer is that "there are no servers to maintain and no administrators needed". Even more importantly, deploying applications on the PaaS layer allows automatic scaling and load balancing. Specifically for App Engine, other features include persistent storage options, user authentication and e-mail APIs, fully integrated development environments simulating GAE and access to a free tier which allows applications to be hosted for free up to certain quotas—enabling inexpensive experimentation. They also note that Amazon Web Services (AWS) provides similar features by utilizing EC2 (VMs), key-value stores (SimpleDB) and web-service-based filesystems (S3) at the IaaS layer. Comparing the two, the author says that IaaS systems may "take more time to build" but the "*resulting system will be more extensible*". On the other hand, PaaS based systems are more constrained but offer very efficient, streamlined MMOG development.

Shabani et al. [3] discuss using Google products such as GAE and Google's Datastore to create distributed systems. According to the authors, the primary role of GAE is "to serve applications which are simultaneously accessed by many users" and "have the same power as Google's applications". The environment of GAE is designed for "real-time dynamic applications" running on Google infrastructure that provides on-demand resources and automated load balancing. Adding to the survey of Reference [27], the authors mention programming languages that are supported by GAE: Java and Python. An update on the programming languages and tools supported by GAE reveals that there is a larger range of tools to choose from today: Java, Python, Go, PHP, .NET, Ruby and Node.js, making it possible for developers to create applications based on every popular technology stack available. Ultimately, GAE can be used to create various types of applications, ranging from simple web applications to "powerful commercial products" [3].

5.1.4. Hybrid Clouds

A hybrid cloud is a composition of two or more types of clouds. Hybrid clouds are usually employed to harness the advantages of the combined types of clouds, while addressing their issues.

To investigate the potential of virtualization technology to dynamically provision resources outside the operator's data center, Reference [28] proposes a hybrid resource provisioning model which uses smaller, less expensive and self-owned data centers complemented by public cloud resources. The privately owned data centers operate pre-installed game servers while an additional set of public cloud servers offer access to virtualized resources. As described, game hosts pool resources from both to serve several games simultaneously. This approach adds an additional virtualization overhead to the standard commodity cloud approach. This overhead is necessary so that resources from both private and public clouds can be dynamically provisioned using an integrated set of policies. To evaluate their method, the authors consider various metrics related to the virtualization overhead, such as the time taken to instantiate VMs and a resource under-allocation metric. Results have shown that virtualization policies employed by private and public clouds alike are important for good performance.

Negrão et al. [29] describe a hybrid cloud solution which breaks down high-level tasks into sub-tasks that can be offloaded to public cloud resources [29]. More specifically, the system identifies an overload event and breaks down high-level tasks that are executed by an overloaded server into smaller sub-tasks. These tasks are partitioned into two groups. Firstly, important sub-tasks that have strong timing constraints or require the game data to be kept at the original server are categorized as "*core tasks*" and are kept on the private cloud. On the other hand, sub-tasks with more lenient timing requirements are categorized as "*background tasks*" and can be offloaded to temporarily acquired public cloud resources to relieve the private cloud servers. These background tasks are designed to be game-independent, requiring no game state to be computed but only abstract data such as geometric positions. The characteristics of this approach make it ideal for "*temporary overload situations*" and for execution on unreliable environments. Results from empirical evaluation show that the solution achieves modest frame rates (i.e., 10 frames per second) for up to 1500 clients. Finally, it "*gives application programmers more freedom*" while still allowing applications to benefit from the advantages of task partitioning.

5.2. Architecture

In a similar fashion with infrastructure, developers and researchers use multiple types of architecture to support their MMOGs. These architectures vary in terms of organization and communication patterns but can be categorized into three main groups.

5.2.1. Client-Server Architecture

The client-server architecture is a distributed structure that partitions the workload among different resource providers (called servers) and resource requesters (called clients). Machines in a distributed system use messages in a pre-defined language to communicate. In a typical client-server system, the client can request resources from a server through these messages and the server responds by providing the resource. Client devices are not aware of one another and cannot message each other—unless the messages are explicitly forwarded by the servers.

A study by Assiotis and Tzanov [30] discusses architectures for MMOGs [30]. The authors propose a centralized distributed architecture (aka client-server) to support a large number of concurrent users without sacrificing efficiency or security. As described, the client-server architecture traditionally uses a single server which handles traffic to and from all clients—for example, *Quake* and *Doom* are games using this architecture. From an architectural standpoint, this presents some challenges: Firstly, the need to support a large number of players means that a lot of data needs to be communicated through internet. Quoting the authors of Reference [30], "*[...] as the information transferred between the players and the game server is large, the bandwidth required to support a huge number of players is enormous.*" Secondly, very large worlds require "*huge computational power*" to simulate, meaning that processing needs to be split to multiple computing nodes.

At the center of this study is the *locality of interest*. Using this concept, game developers separate large worlds into smaller regions that can be hosted on different nodes. This architecture allows both bandwidth and computational power requirements to be spread out over many nodes. However, this architecture creates new challenges:

- Players are not always interested in receiving updates about certain areas of the map—especially if these areas are far away.
- When two players are near the border between two parts, they still need to *see* and *interact* with each other—however this is not trivial when these are hosted on separate nodes.
- Regardless of synchronization scheme, there is a possibility the game state will be invalid for events that occur near borders and affect players on both sides.

To solve the first problem, the authors introduce a concept called *Area of Interest* (AoI). In this concept, each player has their own AoI from which they are able to receive event updates—the area spans outwards from the player's position for a certain distance. Players are naturally not interested in receiving updates about events they cannot see or hear because they are too far away. The size of an AoI can vary depending on the player type. For example, a player carrying a sniper rifle needs to have a larger AoI than a player with a pistol because of the range of his equipment. As a result of this concept, players are only subscribed to a limited area of the game world drastically reducing the bandwidth required to communicate the game's state.

For the second and third problems, the authors have identified four distinct scenarios that need to be handled when players are near border areas:

- A player standing near the border of two regions hosted by different servers needs to be able to receive event updates within their AoI from both servers.
- A player may suddenly move to an area handled by a different server (this is usually known as "teleporting" in games).
- An event originating in one server may end up in a region covered by another server. A typical example of this is shooting a rocket that travels from one area to another before exploding.
- An event that occurs near the border may affect multiple regions, hosted on different servers. An example of this is a bomb exploding at a border, affecting players in adjacent regions.

By describing further solutions to these problems –for example subscribing a player to both servers when they are within a certain distance of their border– the authors provide a novel strategy with which large-scale worlds can be distributed on a client-server architecture. The results of this study show that using these solutions the authors have managed to improve the efficiency of the client-server architecture and thus the performance of MMOGs.

Nae et al. [20] argue that *"today's MMOGs operate as client/server architectures"* [20]. Specifically, the game server is used to simulate a world via computational and data operations by receiving and processing commands from the clients. Based on these commands, the server computes a global state of the game world which represents the positions of and interactions taking place between entities. Finally, the server sends responses containing the new state back to the client devices which render this information to the player. Nae et al. [16] argue that a good game experience is paramount to keep players engaged and has a direct impact on the income of the game operators [16]. For this reason, operating an efficient architecture is of huge importance. To support thousands of simultaneous players, the authors describe three parallelization techniques commonly used with client-server architectures:

- *Zoning*: Partitions the game world into areas that are "*handled independently by separate machines*". This technique is particularly useful in slow-paced games such as MMORPGs.
- *Replication*: Parallelizes game sessions with large numbers of players gathering in certain hot-spots. Each server computes the state of a number of "active entities" that are based on it, while it synchronizes the state of other "shadow entities" that are based on different machines. This technique is primarily used in fast-paced games such as FPS games.
- *Instancing*: "*Distributes the session load by starting multiple parallel instances of highly populated zones*". These zones are independent from each other.

5.2.2. Peer-to-Peer Architecture

Another type of architecture is the peer-to-peer architecture (P2P), which partitions the workload among equipotent, equally privileged peers in a network. This architecture makes each peer a participant in the hosted application by utilizing a portion of its resources (such as processing power, storage etc.) and making it available to other peers. Each peer can thus be both a client (requester) and server (supplier) at the same time.

GauthierDickey et al. [31] discuss the peer-to-peer architecture extensively, as their selected approach to enable a fully distributed MMOG [31]. They argue that P2P:

- Reduces delay for messages and eliminates localized congestion,
- Allows players to launch their own games without a lot of investment,
- Allows games to overcome bottlenecks of the server-only computation,
- Is more resilient and available because it does not have a single point of failure.

In addition, the authors explain how peer-to-peer storage can work in the context of an MMOG. When utilizing peer-to-peer storage, the consistency of data must be guaranteed using mutual exclusion. The two sub-types of P2P architectures described are the *unstructured* and *structured* *networks*. In unstructured P2P, clients can transfer files to each other directly while in structured P2P a distributed hash table is responsible for converting resource names into addresses in the network. This requires the maintenance of routing tables by each peer, using special algorithms. The authors of Reference [31] also explain how P2P computation occurs using completely distributed scheduling. Unlike the client-server architecture, peer-to-peer utilizes the down-time of peers to provide computational power to the peers that need it. The authors, however, admit that cheating is an issue with P2P systems that support MMOGs. The nature of this fully distributed approach makes them vulnerable to state manipulation—something that must be addressed to successfully utilize P2P as an MMOG architecture.

Kavalionak et al. [32] state that the client-server architecture is the most used option when it comes to MMOG architecture [32]. However, they assert that this approach has limitations which cause it to have scalability limits.

On the other hand, the peer-to-peer approach can offer the following advantages:

- Inherent scalability, as the available resources grow with the number of users,
- Robustness as systems using this architecture can self-repair when a peer fails,
- Avoids bottlenecks as network traffic is distributed among the users.

Others, like Mildner et al. [33] focus on the performance aspect of architectures for MMOGs. The authors propose a P2P overlay for a Networked Virtual Environment (NVE) for an MMOFPS game. Their approach tries to minimize the overhead for connection management to create a highly responsive system. Instead of using sender-oriented message distribution, which is used by most existing systems, the authors utilize a *publish-subscribe mechanism* to avoid overlay inconsistencies and map user interest within an NVE more efficiently. Moreover, they propose a Geocast algorithm which sends messages to an arbitrary set of users based on their positions. They implement an NVE system on both a simulation environment and the pre-existing game PlanetΠ4. The results of their experiments have shown that their approach offers a scalable and consistent overlay by limiting the number of connections per user, which is crucial especially in scenarios of crowding/flocking.

5.2.3. Hybrid Architecture

Kavalionak et al. [32] also propose a novel cloud based architecture with hybrid architecture consisting of two components: the *positional action manager* which manages positions of entities and the *state action manager* which enables the storage of entity states without the need to transfer them across nodes [32].

The authors claim that this is the "*first [work] proposing the integration of P2P and cloud computing*". Furthermore, they explain that hybrid MMOG architectures aim to "*exploit and combine*" the advantages of both P2P and client-server architectures. An issue with this type of architecture is the strategy for the partition of the virtual environment. Using the first method—*spatial partitioning*—the game world is divided into regions which are then distributed to the peers with the most resourceful peer entering that area becoming the manager of it. For example, the authors of Reference [18] propose a hybrid system which includes a central server and a *pool* of peers. In this system, the central server hosts the MMOG and distributes the game to other peers once it reaches its full resource capacity. The second method—*functional partitioning*—delegates important functions to the peers.

Jardine and Zappala [34] utilize the functional partitioning approach and categorize types of moves within an MMOG into two groups: *Positional moves* which occur when a player moves in the game environment and *State-changing moves* which change the game state (such as when a player attacks another player) [34].

By distinguishing between these two types of events, functional partitioning can be used to delegate only a subset of the total events using the P2P approach. The authors explain that positional moves are comprised of abstract data (such as the position of a player) and do not contain any entity specific information—thus making them easier to distribute among non-reliable peers. On the other hand, state-changing moves contain entity-specific data. In this hybrid system, a central server appoints peers as regional servers which handle positional moves for a specified region of the world. Because they contain only abstract data, the consistency of the game will remain intact even if a peer (which is considered non-reliable) abruptly leaves the game session. Conversely, state-changing moves—which contain the state-specific data and would compromise the game state consistency if lost—are assigned to the central server which is more reliable than a *regular* peer.

More recent advancements in research, such as those proposed by Matsumoto and Okabe [35] further analyze the features of MMOGs and investigate cheats that may be caused by the P2P type of architectures. After considering the types of cheats possible as well as the detectability of each cheat type, the authors propose a collusion-resilient hybrid P2P framework for MMOGs which utilizes data 'scrambling' and the Chinese Remainder Theorem to guard against data corruption. The authors evaluated the proposed framework and compared it with other approaches, concluding that it was more effective –especially in cases where various peers colluded with each other– even though it did not completely eliminate the possibility of cheating.

Zhang et al. [36] identify the challenges of Virtual Reality MMOGs (VR-MMOGs): stringent latency, high bandwidth and large scale. They propose a hybrid gaming architecture to achieve more efficient distribution of work by placing local view updates on edge clouds for faster responses, higher bandwidth and global state updates on center cloud for higher scalability. To achieve this, they use a service placement algorithm which dynamically places a user's service on edge clouds while they move across different access points. To evaluate their approach, the authors conduct simulation experiments using their approach compared to other gaming approaches. They find that their approach is a "*viable solution for supporting VR-MMOGs*".

Similarly, Plumb et al. [37] explore the benefits of adapting hybrid architectures for use with edge servers. They propose AvatarFog, a solution for forming hybrid P2P clusters of nodes using game design to decide the network topology instead of the physical structure of the client-server architecture. They focus on improving latency between the players and look at the interactions between them in the virtual world instead of the physical connections of clients to servers. Their approach groups players together using gameplay as a common factor rather than their position in the world. By creating a custom simulator, they are able to evaluate the performance of AvatarFog and conclude that their approach "*improves latency and server resources over the traditional server and client model*".

5.3. Scalability

One of the most fundamental characteristics of MMOGs is the need to scale up or down to accommodate a changing number of active players. Scalability goes beyond game mechanics or features. It ultimately determines the game operator's business strategy and the amount of revenue that can be generated from an MMOG. In this section we discuss the importance of scalability and how it can be achieved to create more resource-efficient and therefore, profitable games.

The acronym MMOG itself expresses the need to accommodate scaling: *Massively Multiplayer*. Online games must be inherently scalable and preferably elastic if they are to succeed. Blackman and Waldo [38] argue that MMOGs have requirements "*unlike many other [applications that tend to be] embarrassingly parallel [and] optimized for throughput*" [38]. In contrast to these applications, online games' most important requirement is their ability to scale to potentially "*thousands or hundreds of thousands [of players] interacting across a large number of servers*".

Lu et al. [39] argue that scalability of a system can be measured in "*terms of the number of players supported*" and "*is of great importance to ensure commercial success*" [39]. These authors also argue that scalability can be achieved by designing a system to accommodate additional servers and then distributing the computational load across them. They break scalability into two separate topics: *consistency* and *load balancing*. In particular, the authors describe a system based on server clustering which aims to achieve scalability without sacrificing consistency.

5.3.1. Consistency

Consistency is defined as *"the need to provide players with mutually consistent views of the gaming arena in a timely manner to allow fair game play"*— it is one of the major challenges that arise when a system becomes scalable [39]. As the number of players (and thus transactions) grows so does the number of servers in the system. At this point, it becomes increasingly difficult to avoid inconsistency or poor performance without efficient consistency management. The authors argue that a solution to this problem is utilizing localized gameplay, breaking down areas of the world into smaller parts that can be managed relatively easily. They categorize techniques used to provide *manageable consistency* through localized gameplay into two further groups:

- Geographic: The world is divided into regions at initialization. For example a room inside a building can be considered a separate geographic region. This is similar to *zoning* mentioned in Reference [16].
- Behavioral: The world is divided into further sub-divisions based on the interaction patterns of players. For example, a difference in the size of the AoI—discussed in Reference [16,30]—may alter a player's ability to influence the state of the game.

Geographically influenced regionalization reduces the consistency problem into a more manageable size using three rules: (i) Players cannot interact across different duplicated worlds, (ii) Players cannot interact across different regions and (iii) Players should interact *intricately* with players they specifically target.

This further categorizes the types of interaction into two groups, based on the level of consistency required: (i) a view type, where the players can only see other players and their actions—this requires weaker consistency and (ii) an intricate interaction type in which players directly interact with each other—which requires strong consistency. In the latter type, event ordering and synchronization of processes such as those mentioned in Reference [31] are of paramount importance.

Additionally, Chuang et al. [40] introduce EventWave, an event-driven programming model which "*allows developers to design elastic programs with inelastic semantics*" [40]. This is achieved by (i) allowing logical nodes to execute multiple events in parallel and (ii) allowing a single logical node to be distributed over multiple physical nodes, while guaranteeing atomic event semantics. Events can be ran in parallel in a distributed system provided that they do not access the same state. The authors leverage this by identifying which events are state-independent of other executing events and propose a runtime model that can execute these events in parallel using context mapping, a technique that maps event contexts to specific nodes. The ultimate benefit of EventWave is that programmers can reason about their programs without considering elasticity—they can focus on the program logic and not the scaling.

5.3.2. Load Balancing

Load balancing is defined as the attempt to "*efficiently distribute an application's processing requirements across a number of servers*" [39]. Lu et al. [39] categorize load balancing strategies into two groups: (i) *Player-based* where players are allocated to different servers as they join a game and (ii) *interaction-based* where servers manage resource allocation based on the interaction patterns of players.

Player-based load balancing can be utilized when servers have to be removed or added without affecting the gameplay. Additionally, when players are allocated in duplicate worlds, the load must be balanced to support interactions between them. Messages exchanged between multiple servers to accommodate such interactions may take up all available bandwidth which highlights the importance of the concept AoI. This concept is pivotal in such cases as it can be used to limit inter-server communications. On the other hand, interaction-based load balancing can be utilized in cases where *crowding* [39] or *hotspots* [41]—a high concentration of players in a specific region—slows down or completely disables the system's ability to provide gameplay. The interaction-based technique is used after the game world has been regionally balanced and can alleviate the imbalance by associating player actions with processing requirements during runtime, which is accomplished in load prediction.

A recent study by Meiländer and Gorlatch [42] looks at the load balancing problem from a performance perspective. The authors propose a generic scalability model for Real-Time Online Applications (ROIA) of which MMOGs are a subset. The model 'monitors the application's performance at runtime and predicts the benefit-cost ratio of load-balancing decisions'. This is achieved by weighing the benefits of the load-balancing actions against their overheads in terms of time and resources. By measuring this ratio, the model can recommend whether to distribute workload and how often to do so. The authors use two types of resources, computation in terms of CPU and communication in terms of network to evaluate the quality of their model in a multiplayer shooter game simulation. Meiländer and Gorlatch [42] claim that their evaluation has proven their model to be 'both adequate and accurate', allowing higher efficiency of load balancing actions on clouds.

Farlow and Trahan [43] state the importance of keeping the computational load balanced between the servers that host a game world. However, this balance is "*subject to constraints such as player satisfaction and maximum server computational capacity*". The authors describe the problem of load balancing as NP-Complete and develop heuristics that can monitor load balancing and "*bring an unbalanced system back into balance*". Instead of making load-balancing decisions on each movement, the authors use breakpoints during which load balancing operations take place. They also define the Load Balancing Factor (LBF), which is the difference in load between the highest and lowest loaded server. They use LBF to determine if a load-balancing action needs to be carried out at each breakpoint. If an action needs to take place, the system can *add/shed* zones or rejoin them accordingly. They measure the effectiveness of their system, BreakpointLB, by running experiments using various parameters in a simulator. Their results show that BreakpointLB "*is able to bring the load into balance*" more efficiently when the LBF is higher and is better when compared to algorithms such as ChenLB.

5.3.3. Load Prediction and Provision

One of the problems faced by game providers is the provision of resources for MMOGs. Traditionally, resources are made available by installing and operating large dedicated infrastructures, sometimes with hundreds of servers [20]. However, the dynamic nature of MMOGs coupled with static infrastructure leads to over-provision or under-provision of resources, both of which lead to financial loss and make it difficult to join the competitive market. Nae et al. [20] propose a "*dynamic provisioning method in which the amount of resources if first predicted and then obtained dynamically*" from servers external to the MMOG operator [20]. The authors have used analytical load models for CPU, memory and network resources, taking into account player number and interaction types. As they describe, dividing the virtual world into smaller areas is proven to improve the accuracy of their real-time prediction models. Their measurements have shown that the proposed load prediction algorithms were able to "*reduce MMOG operation costs*" while their neural network predictor was "*the best resource provisioning*" strategy evaluated. Load prediction is also found to have a positive effect on reducing latency and the ability of the system to service multiple MMOGs simultaneously. The authors also claim that their resource prediction algorithm "*performed significantly better [than other similar algorithms]*" and highlight the importance of using dynamic over static resource allocation.

Shaikh et al. [15] utilize dynamic resource provisioning in their prototype implementation of a platform for online games [15]. The *Provisioning Manager* (PM)—a core part of their study—is a software component that automatically provisions resources for MMOGs by first collecting performance and availability metrics from the infrastructure and then responding to changes by adding or removing servers. Similar to [16], the PM also implements a set of data and performance models to achieve this but also collects system statistics such as CPU utilization, memory usage and bandwidth consumption for analysis. Furthermore, the PM is responsible for provisioning servers for auxiliary services, such as content distribution or game deployment when necessary. Lastly, through their study [15], Shaikh et al. [15] were able to identify where utility computing can be leveraged to provide on-demand resources for online games. Other works, including References [15,16,29,44] have also commended the benefits of cloud computing, especially for its inherent scalability.

To address the problem of resource-provisioning in MMOGs, Ghobaei-Arani et al. [45] propose an autonomous resource-provisioning framework based on cloud infrastructure. They use a load prediction service that *'anticipates'* distribution of game entities from trace data using the ANFIS prediction model. Their framework splits an MMOG into tiers: the gateway tier, responsible of functioning as a bridge between a client and the game layer; the cellular tier, responsible for processing commands from the players and the database tier, responsible for storing the data elements of the game. Their fuzzy decision tree algorithm estimates the proper number of resources that should be allocated to each of these tiers. The authors use a real workload using RunEscape and a synthetic workload to evaluate their approach using simulations. Experiments have shown that this approach outperformed others in terms of accuracy and performance, leading to more efficient resource provisioning for MMOGs.

5.4. Persistence

A large variety of storage systems have been utilized in support of MMOGs. In this section we discuss how different studies have used varying types of data persistence systems, depending on their needs, to enable MMOGs on both dedicated and cloud infrastructures.

For instance, Spanner is "*Google's highly available global SQL database*", which manages data replication and transactions at a large scale [46]. Brewer [46] argues that based on the CAP theorem, systems can only have two of the three properties: *Consistency*, *Availability* and *Partition tolerance*. In other words, databases which need to be distributed across many nodes—and thus be scalable—cannot be fully consistent and available at the same time. According to Reference [47], either of the two needs to be sacrificed: "*Relaxing consistency [allows] the system to remain highly available whereas making consistency a priority means that the system will not be [fully] available*". To work around the CAP theorem, Vogels [47] suggests the use of eventual consistency, a form of weak consistency [47]. Eventual consistency guarantees that given no new updates to an object, all accesses will return the last updated value. Depending on system load, latency, and so forth, there is a specific inconsistency window during which consistency failures may occur. The author argues that this inconsistency has to be tolerated because (i) it results in an improvement in performance under highly concurrent conditions and (ii) can handle partitioning of the data which would otherwise render the system unusable.

MMOGs have several differences in requirements when compared to other types of software [38]: (i) Latency is more important than throughput in order to support fast response times for users; (ii) Unlike most other applications, MMOGs require a higher ratio of data writes to reads; (iii) Users are more willing to tolerate the loss of data due to failure as long as the recovered state remains consistent, in contrast to other applications that involve real-world goods or payments.

The authors discuss data storage in Project Darkstar: "*an infrastructure for building online game worlds*". They propose a new approach which uses *write caching* to cache data locally on each node if the data is only being used by that node. If any other nodes require access to the modified data, then the node flushes the modified data to the central server so that it can be accessed. The suggested approach can lead to lower network latency, especially since most of the data changes are utilized on the same node. As a bonus, it allows the addition or removal of nodes and avoids the need for redundancy and backups because nodes do not store "globally important data". This allows this storage option to be scalable while maintaining consistency.

In their survey of big data and cloud computing, Agrawal et al. [48] outline the features which a cloud system must possess to effectively utilize *cloud economics*: (i) scalability, (ii) elasticity, (iii) fault tolerance, (iv) self-manageability and (v) ability to run on commodity hardware [48]. The authors argue that traditional RDBMSs are not optimized for use on the cloud as they were created for use on enterprise infrastructure and say that "*the hefty cost*" associated with them is less attractive for deployment of large scale applications on the cloud. Instead, they focus on a newer generation of distributed data stores which utilize key-value pairs. This type of data storage—known as NoSQL—has been very successful and widely adopted, mainly because of its ability to scale on cloud systems. Nevertheless, the authors argue that the key-value type of data stores lack in functionality when compared to the traditional RDBMSs, limiting the set of applications that can be created with them.

Researchers at Google have designed appropriate storage systems such as *BigTable* [49] and *Megastore* [50] to meet the demands of their online services. They identify several conflicting requirements of modern applications—such as MMOGs—that are based on the Internet:

- The applications must be highly scalable to accommodate a potentially large audience of users,
- Rapid development of features and fast time-to-market is essential for competitiveness,
- Services must be responsive, therefore a system must have low latency,
- The system should provide a consistent view of data—therefore updates need to be made visible immediately,
- Services must be highly available and resilient to multiple types of failure.

While relational databases provide a rich set of features, the authors agree that they are difficult to scale. On the other hand, NoSQL datastores such as Bigtable [49] and Cassandra [51] are highly scalable but have limited APIs, loose consistency and fewer features—which complicates application development. Megastore [50] falls in the middle of these two types of data storage, "*[blending] the scalability of NoSQL datastores with the convenience and functionality of a traditional RDBMS*". It provides fully serializable ACID semantics over distant replicas of data which leads to low latency. Additionally, by using synchronous replication, Megastore is able to achieve both high availability and strong consistency at the same time. The Megastore relies on both RDBMSs' schema to define its data model, while featuring the row-column model structure of NoSQL. Data is entered as entities which contain a set of properties, which are essentially key-value pairs of strings, numbers and so forth.

Google's Datastore [3] is one of the latest incarnations of Megastore. The Datastore is a similar, highly available, highly scalable distributed data storage that can be utilized by developers using the Google Cloud Platform. Applications developed to run on Google's App Engine utilize the Datastore to create web applications while encompassing the advantages of both RDBMS and NoSQL systems. In addition, the Google Query Language (GQL) can also be used as an efficient way to run queries on the data, similar to RDBMS. GQL is very similar in syntax to SQL, even though it has some limitations. One of these limitations is the lack of complex queries such as join queries.

More recent research by Diao et al. [51] shows that strong consistency can be provided for scenarios where systems need to be both highly available and scalable, by implementing a lightweight mechanism which detects failures and reacts accordingly when needed. Firstly, they classify MMOG data into four sets: (i) account data, (ii) game data, (iii) state data and (iv) log data. Secondly, they describe an approach that processes modifications of state data by using an in-memory database in real-time. These changes can be synchronously propagated to other players in an acceptable amount of time. Then the data is backed up to the disk database periodically to allow recovery to a previous state. Furthermore, the authors say that popular MMORPGs such as World of Warcraft and Second Life utilize Relational Database Management Systems (RDBMS)—predominantly MySQL and Microsoft SQL Server. They argue that RDBMSes do not fully satisfy the requirements of MMOGs and propose the use of Cassandra, a cloud data management system. Cassandra is able to support bulk writes and rare read operations which is the predominant scenario for MMOGs. Because of Cassandra's weak support for strong consistency, the authors also propose a solution based on eventual consistency to achieve strongly consistent data storage. Furthering their research in [52], the authors investigate how to benefit from the advantages of cloud data management solutions while addressing their shortcomings with regard to MMOGs. After their analysis of typical architecture and data management requirements for MMOGs and their categorization of MMOG into four groups, the authors propose using multiple data management systems in a single MMOG to manage diverse data sets accordingly. Data which requires strong consistency and security (e.g., account data) is managed by an RDBMS, while data requiring scalability and performance (e.g., logs and state data) are stored in a cloud storage system. The authors implemented a simulation environment where many clients can interact with many servers and a game prototype based on an open-source MMOG to evaluate their approach. They found that the guarantee of high-level consistency in Cassandra is not efficient, thus proposing a timestamp-based model that solves this problem.

5.5. Performance

The performance of MMOGs and other resource-intensive applications is considered a critical factor for their success [16,21]. This section surveys techniques used by researchers to *measure the performance* of their solutions as well as approaches they used to *improve the performance* of MMOGs.

Often, the performance of fast-paced games is measured in *frames-per-second* as well as in terms of *latency*. Unsurprisingly, the latency expectations vary based on the game type.

For instance, these are some common game genres with their expected/desired latency, as it was found in the related work:

- *First Person Shooter* (FPS): 100 ms –250 ms [20,31,53].
- *Real-Time Strategy* (RTS): 500 ms–1000 ms [31,53].
- *Role-Playing Games* (RPG): 1 s–2 s [20,53].

Dhib et al. [21] argue that "*ensuring an acceptable Quality of Experience (QoE) for all players is a fundamental requirement [for cloud-based games]*" [21]. The authors propose a mathematical model for measuring the QoE in MMOGs. They identify the *global response delay* as the most notable metric, which is dependent on several other parameters such as the CPU and memory capacity. Furthermore, they support that the network distance between the user and the server plays a significant role in the response delay. They evaluate: (i) the performance of a cloud-based MMOG in terms of response delay, using simulations and (ii) the degradation of the QoE as a function of the number of allocated VMs and number of players, using an empirical approach. Furthermore, they propose a mathematical model which expresses the QoE as a function of the network and processing delays. Using this model, the authors propose a dynamic VM allocation strategy which attempts to minimize the cost per customer, while ensuring that a "minimal threshold" of QoE is maintained. Data gathered shows that the approach resulted in "high player satisfaction" while maintaining 99% of QoE.

Lin and Shen [54] propose a lightweight system called CloudFog, which aims to improve the performance of cloud-based games by incorporating "supernodes"—powerful nodes that are located between the end-users and the cloud [54]. CloudFog works by computing the game state on the cloud but then uses the supernodes to carry out several intensive tasks such as video rendering and streaming to client nodes. The authors identify challenges that hinder their games' success, such as: *latency* (response delay), *network connection*, *user coverage* and *bandwidth cost*. They utilize simulations to evaluate the performance of *CloudFog* and assess it against other systems. They measure (i) latency, (ii) playback continuity and (iii) user coverage. With these, the authors examined the effectiveness of the supernodes approach and concluded that CloudFog reduces latency, bandwidth consumption and bandwidth cost, while it has a positive impact on user coverage.

Successful deployment of any MMOG requires an "*ultralow-delay cost-efficient design*" [18]. Thus, Barri et al. [18] investigate how the costs can be minimized while satisfying the QoE requirement for players. They present a resource allocation framework for cloud infrastructure that benefits from virtualized resources. They consider all resources that can impact delay after presenting an 'accurate delay model' to control the QoE. They further propose the use of a new delay-aware cost-minimization resource allocation scheme and perform simulations to evaluate its performance. Their MATLAB simulations are conducted on cloud servers that are randomly distributed across a large geographic area and the workload model of the simulations is obtained from real online game data. Results obtained from these simulations show that the new scheme outperforms others in terms of cost efficiency while still satisfying the delay requirement which is paramount to a satisfactory QoE.

Gascon-Samson et al. [55] state that in order to maintain the quality of experience, the game state update message "*must be delivered within specific time bounds*", depending on the type of MMOG. They describe flocking—the gathering of players in hotspots—as a challenge, because of the high bandwidth requirements it places on a single server. To provide more efficient state updates the authors present DynFilter, a game-oriented message processing middleware which filters out state update messages from entities located far away in order to reduce bandwidth needs. DynFilter is based on the publisher-subscriber pattern instead of sender-oriented message models. By running experiments on Amazon's EC2 platform, they prove that DynFilter is able to maintain bandwidth use within quotas while still maintaining the QoE through the delivery of state update messages.

Similarly, Yusen et al. [56] propose a fairness-aware state update scheduling algorithm that minimizes inconsistencies while guaranteeing fairness in Multi-Server Distributed Virtual Environments (MSDVEs). They argue that MSDVEs suffer from saturation which leads to high bandwidth use and huge resource demand. To achieve timely state dissemination and guarantee fairness, they first devise a new metric which uses time-space inconsistency to measure unfairness in an MSDVE. Secondly, they proposed a fairness-aware update scheme that ensures that updates are issued to different clients at the same time. Lastly, their algorithm called FairLMH minimizes inconsistency in MSDVEs. By conducting simulations, the authors were able to prove that FairLMH providers better fairness compared to other similar algorithms in multiple scenarios.

Assiotis and Tzanov [30] discuss fault tolerance in a client-server architecture for MMOGs [30]. Looking at performance from another angle, they state that a "*system should recover the entire state of the world it represents as it was prior to the crash very quickly and as transparently as possible*". This suggests that another measure of a system's performance could be (i) the frequency (or rather infrequency) of errors, (ii) its ability to transparently recover to a valid state and (iii) the time taken for the recovery to take place. While the authors say that their client-server architecture is not inherently fault-tolerant, they propose mirroring and replication to prevent any players from being "locked out" of the game due to server crashes.

Jardine and Zappala [34] propose and evaluate a hybrid architecture using a simple MMOG and automated players (bots) which are programmed to move toward game objectives as quickly as possible [34]. The authors created the game in a way that it can be played on both client-server architectures and hybrid architectures. Assuming stable Internet connections for all players, they run a series of experiments each consisting of fifty players with a player joining that game at an average of one second. They conduct their experiment on client-server architectures and subsequently on hybrid architectures and measure incoming and outgoing bandwidth and latency. These experiments have revealed that the hybrid architecture can "*save considerable bandwidth for the central server*" and that "*latency can be kept low*" as long as there are enough peers capable of acting as regional servers.

Nae et al. [16] evaluate their proposed MMOG ecosystem by utilizing trace-based simulation [16]. The authors propose an analytical model that can be used to express the overheads of virtualization from cloud resources. They evaluate the effect of utilizing virtualized resources on the quality of gameplay using RuneScape, a popular MMOG and state that virtualized resources can "*negatively affect the MMOG session at high load volumes*" (higher than 90%), which is what they initially expected. These authors demonstrate the usefulness of their ecosystem by showing that resource under-allocation grows only linearly with the VM size and start time while the bandwidth and virtualization penalty have little impact on performance. In a relevant study [28], the same authors also assessed the impact of data center policies on the quality of resource provisioning. They found that "*[dynamic resource provisioning] can be much more efficient than its static alternative even when the data centers are busy*". To support this, they present experimental results from trace-based simulations which highlight the real-time parallelization and load balancing of a game prototype using external data center resources—ultimately illustrating the advantage of dynamic resource provisioning.

Furthermore, El Rhalibi and Al-Jumeily [57] propose a dynamic AoI management that aims to minimize delay and network traffic for MMOGs based on a hybrid architecture [57]. They use simulations with 125, 500 and 1000 peers respectively to carry out their evaluation, with scenarios comprising of both client-server and hybrid architectures. The results of their study show that their AoI management approach produces lower delay and network traffic when used on their hybrid architecture compared to the client-server architecture without AoI management.

Negrão et al. [29] evaluate their hybrid cloud solution using a setup comprising of nine machines [29]. They compare the performance of their system when all servers are state partitioned against a version with mixed task servers and state partitioned servers. They obtain results from a varying number of clients simulated by bots, showing the difference in performance between the two approaches—which underscore the improvement in performance in terms of higher frame rates and lower bandwidth.

EventWave [40], on the other hand, is evaluated using "*synthetic microbenchmarks [which allow the authors to] vary the number and size of independent [event] contexts*", effectively allowing for a greater range of experimental setups to be tested. The authors use two case studies to test their approach, a key-value store and a game server. Their experiments utilize a generator which produces a series of events to measure the system's performance in terms of throughput. Using microbenchmarks, the authors discovered that "*as the number of physical nodes increases, throughput does not drop; in fact it increases*". Drawing conclusions from this, the authors claim that (i) the maximum throughput of EventWave applications does not drop as logical nodes spread over physical nodes and (ii) EventWave is able to harness the computational resources of multiple physical resources to maintain performance under high levels of load. To test EventWave as a game server, the authors deployed 128 clients over 16 Amazon EC2 instances. They generated "artificial load" in the game world to simulate game behavior. To evaluate this case study, the authors measure the average latency of clients as they move around the game world. When their proposed elasticity mechanism is not activated, the server is unable to process client requests fast enough, resulting in a dramatic increase in latency. However, when EventWave is activated, the server scales up to take all available physical nodes. In the opposite direction, the system scales down when the number of players is reduced. This dynamic migration (i) allows the provision of resources to serve the game more rapidly, (ii) reduces the average latency experienced by the clients and (iii) allows the system to scale up or down according to demand.

Baker et al. [50] argue that the development of Megastore was "*aided by strong emphasis on testability*" [50]. To detect bugs in their system, the authors used a pseudo-random framework—"*the network simulator*". This simulator is capable of "*exploring the space of all possible orderings and delays of communications between simulated nodes or threads*". Given the same seed, the simulator is also capable of reproducing the same behavior. Using this tool, the authors were able to detect bugs when a problematic sequence of events triggered an assertion failure in their code. They claim that even though an exhaustive search of all the possible states is impossible, the simulator explores "*more than is practical by other means*". Using real-world deployments, the authors also observed that Megastore provides a latency tail "*significantly shorter than that of the underlying layers*". Furthermore, applications based on Megastore can withstand planned or unplanned outages with "*little or no manual intervention*".

5.6. Security

Security is an issue that is sometimes overlooked when it comes to developing games. Admittedly, it may not be a critical aspect of games or is at least not a priority when it comes to their development. However, modern MMOGs feature much more than a game world and simulated entities. Today's games feature micro-transactions and sometimes store sensitive personal data, both of which require developers to take a serious stance on security to avoid financial loss and/or legal problems in the future. In this section, we explore the security concerns that have been expressed by relevant sources to provide insight into the relevance of security for MMOGs.

Shaikh et al. [15] discuss security in the context of their proposed on-demand platform for MMOGs [15]. They argue that by provisioning resources on a server-by-server basis, their system simplifies many issues arising from the *"fine-grained resource sharing"* approach they use. The authors argue that while it may be desirable to host games with low resource requirements on the same server, this would require *"sufficient protection"* so that no game is allowed to corrupt another game's data. Furthermore, the authors discuss issues related to the peer-to-peer distribution model. They argue that while this model is appealing because of lower bandwidth costs, it adds a liability for game providers who use it to distribute content: *"Players must allow [...] untrusted machines to connect to their own machines"*, which exposes them to malicious actions.

The authors of References [31,32,34] also support that the peer-to-peer architecture may lead to security problems. Kavalionak et al. [32] state that the lack of a central authority *"hinders security and anti-cheating enforcement"* and argue that when clients have *"heterogeneous constraints on computational, storage and communication capabilities"*, they become vulnerable to exploitation. Furthermore, GauthierDickey et al. [31] argue that cheating is a problem that *"plagues modern games"*. The author agrees that the main problem of the P2P approach is data manipulation and identifies the types of cheats that can be employed by malicious users:

- *Fixed-delay cheat*: where a fixed amount of delay is added into each packet.
- *Timestamp cheat*: where timestamps are changed to alter when events occur.
- *Suppressed update cheat*: where updates are purposedly not sent to other players.
- *Inconsistency cheat*: where different updates are sent to different players.

GauthierDickey et al. [31] provide several solutions to the cheating problem. One of these solutions is "Lockstep", which secures against cheats by dividing game time into rounds during which players send a cryptographic hash of their move to other players. While this approach secures a game against cheating, it presents a drawback of unacceptably high "playout latency"—beating the purpose of the means. Other methods, such as Asynchronous Synchronization [58] and the Sliding pipeline protocol [59] used to resolve this problem are also subject to problems such as high latency and incomplete protection against all possible types of cheats.

Jardine and Zappala [34] propose a P2P/client-server hybrid architecture in which the critical processing events occur on a central server, while non-critical positional updates occur using the P2P approach [34]. These authors claim that the ability to cheat *"is significantly limited"* in their hybrid architecture because the central server *"controls all access to game state"*. They discuss that one possible attack is when a regional server (P2P based) drops or delays some of the state updates (Suppressed update cheat). Their solution copes with this issue by having clients monitor regional server updates for latency and loss and then reporting these issues to the central server which may replace them if enough players complain. In addition, the central server requires each regional server to send a positional update periodically. If three consecutive updates are missed, then the regional server is considered to have failed. This mechanism provides *"additional protection against poor performance or failure"*. The authors also identify another possible attack when *"a player acting as a regional server [joins] its own region"*. If this is allowed, the player may be able to see how other players move before the move is made. The authors remove this possibility by making the central server replace a regional server that moves into its own region. Furthermore, the regional server *"may attempt to collude with other players in the region"*. To tackle this problem the authors implement an auditing mechanism that checks if each state-changing move made was legitimate by using logs to verify that a player had enough time to move into the area where the action occurred. Finally, players may also *"receive an unfair advantage by joining many regions at the same time"*. To eliminate this problem, the central server controls regional server assignments. Whenever a player moves between two regions, they have to contact the central server, get the identity of the new regional server and allow the central server to update the membership list for the affected regions.

5.7. Other Trends in Cloud-Based Games

Besides the previous points which were discussed based on the identified criteria, a few more approaches exist which are better described as a *separate trend*. These approaches are based primarily on server-side processing and light client-side rendering. While the main difference between these games concerns their architecture, there are further significant differentiations such as their monetary model.

For instance, a study by Lin and Shen [60] argues that building, deploying and maintaining large data centers is cost-prohibitive [60]. These authors propose an alternative, lightweight system called *CloudFog*. The *fog* concept uses powerful super-nodes that act as intermediaries between cloud servers and client machines. Using this approach, the intensive computation of the game state occurs in the cloud. This is also known as *Cloud Gaming* [53]. Updates are sent by the cloud servers to super-nodes, which update the virtual world, render the game graphics and stream it as video to the players. An advantage of this solution is that users can play resource-intensive games without expensive hardware. In addition, latency is reduced because the nearby super-nodes are used to render and transmit video which would otherwise have to be transmitted by far-away cloud servers. Thirdly, it helps with the reduction of bandwidth costs as the download of video is done from nodes on the edge rather than at the core of the cloud. On the downside, this approach requires high-speed, stable Internet connections which is not guaranteed in all contexts such as in mobile games.

The use of edge computing is not unique to *cloud gaming*. A study by Burger et al. [61] argues that the performance of MMOGs in terms of latency depends on the geographic distribution of players. In order to minimize latency, the game servers should move closer to the players and towards the edge of the network. To prove this, the authors analyze match histories and statistics from *Steam*, a popular gaming platform. Using this data, they develop a model which can predict player location and match duration. They use their models to evaluate the migration of MMOG Dota 2 matches toward the edge using an event-based simulation framework. Their emphasis lies on how the server placement impacts the QoE of games. The results of the study show that deploying edge servers in many cases reduces the distance of a player to a server by half, thus reducing the load on the dedicated server. Ultimately, a higher number of edge servers with smaller capacities appears to be more beneficial compared to a more powerful dedicated server despite the added operational overload.

More recently, a study by Plumb and Stutsman [62] argues that "*Google's Edge Network changes everything we have concluded about peer-to-peer networks over the past decade*". As the authors describe, Google's Edge Network allows the inclusion of trusted peers in untrusted node clusters, allowing developers to explore P2P algorithms while maintaining the security of their data. The authors investigate the possible advantages of this approach for game developers and MMOGs. They gather ping data and map the population counts of several areas in the United States, which helps run out a simulation to compare existing solutions such as a 'traditional topology' and an 'edge topology' with their own 'Optimized edge network'. Their results indicate that this optimized solution 'presents potential' both in terms of performance (latency) and maintaining security in a P2P network.

An extreme approach for thin-client computing, named *Stadia*, is developed by Google [63]. This is a cloud gaming service, built to stream games at high resolutions and frame rates. It requires an Internet connection but no gaming hardware (such as high-end GPUs and RAM) on the client-side. Stadia renders game graphics on the cloud-based hardware and uses YouTube-like functionality for streaming media to the user. As a cloud-based solution, Stadia offers "*tremendous scale*"and provides a limited variety of hardware and software stacks for developers to use. While it does support multi-player games, it is not designed to realize MMOGs, even though its architecture appears to be fertile for that kind of games too. On the other hand, Arcade by Apple [64] is an online store which offers games that operate offline—which excludes MMOGs [64]. The games execute on the player's device but use the cloud to sync their state so the players can seamlessly switch devices.

6. Analysis and Future Trends

The development and deployment of MMOGs is a challenging, albeit promising area. While the advent of cloud computing has helped bring down operational costs for large distributed systems, the core functionality of MMOG backends is still heavily dependent on custom-tailored backends. We summarize our findings in Table 4, which lists how the current state-of-the-art handles the criteria identified in Section 4. For analytical purposes, studies by the same authors or studies conducted within the same domain have been grouped in a single entry.

Table 4. Comparing the studied approaches using the identified criteria (Infrastructure, Architecture, Scalability, Persistence, Performance and Security).

Approach	Infrastructure	Architecture	Scalability	Persistence	Performance	Security
GauthierDickey et al. [31]	D	P2P	NS	U	U	L
Assiotis and Tzanov [30]	D	CS	NS	U	M & P	L
Shaikh et al. [15], Shaikh et al. [17]	PrC	U	E	R	M, S & P	U
Lu et al. [39]	D	CS	MS	U	S & M	U
Jardine and Zappala [34]	U	H	MS	U	S & P	T
Chu [13]	D	CS	MS	R	U	T
Blackman and Waldo [38]	D	U	MS	R	U	U
Nae et al. [16], Nae et al. [20]	PrC	CS	E	U	S, M & P	U
Chang et al. [49], Baker et al. [50]	PrC	U	E	N	S & O	U
Weng and Wang [44]	PrC	CS	E	U	M & P	U
Chuang et al. [40]	PrC	U	E	N	S & P, O	U
Carter et al. [65]	U	H	MS	U	S	U
Shabani et al. [3]	PuC-PaaS	U	E	N	U	U
Kavalionak et al. [32]	PrC	P2P	E	U	U	U
Lin and Shen [54], Lin and Shen [60]	PrC	P2P	MS	U	S, M & O	U
Diao et al. [51]	U	U	E	N	S	U
Gascon-Samson et al. [55]	PrC	CS	MS	U	S,P	U
Dhib et al. [21]	PrC	CS	E	U	S	U
Negrão et al. [29]	HC	CS	MS	U	S	U
Burger et al. [61]	PrC	U	MS	U	S,P & O	U
Dhib et al. [25]	PuC-IaaS	CS	MS	U	P	U
Yusen et al. [56]	U	CS	MS	U	S	U
Basiri and Rasoolzadegan [66]	PrC	U	U	U	S	U
Matsumoto and Okabe [35]	U	P2P	MS	U	M	L
Diao [52]	PrC	P2P	MS	N	S,P	U
Dhib et al. [26]	PuC-IaaS	U	MS	U	S	U
Mildner et al. [33]	D	P2P	MS	U	S,P	U
Zhang et al. [36]	PrC	H	U	U	S	U
Google [14]	PuC-IaaS PuC-PaaS	U	E	U	U	U
Plumb and Stutsman [62]	PrC	P2P	MS	U	S	L
Meiländer and Gorlatch [42]	PrC	U	E	U	S & M,P	U
Plumb et al. [37]	U	H	MS	U	S	U
Farlow and Trahan [43]	U	CS	MS	U	S	U
Ghobaei-Arani et al. [45]	PuC-IaaS	CS	E	U	M & P	U

6.1. Infrastructure

We can observe that most of the examined papers (55%) discuss the use of private cloud infrastructure to deploy MMOG backends. The dedicated approach is still popular (21%), followed by public cloud IaaS layer infrastructure at 14%. It is worth pointing out that the combination of all types of cloud infrastructures yields a percentage of 79%, which underscores the shift towards cloud infrastructures to power MMOG backends in recent years.

This is also confirmed by Figure 1 which shows the use of infrastructure type over time. While the dedicated approach was the main choice prior to 2010, when cloud computing was not yet widely used, innovations since 2010 appear to have contributed to a shift towards various types of cloud environments, especially private clouds. A smaller subset of these approaches have also utilized public cloud solutions. In these cases, IaaS is the most popular at 66% while PaaS is used in the remaining 33%. While there are valid reasons for this—such as the operators needing to have full control of the performance—it can also be argued that it deprives said solutions from benefiting of the advantages of public clouds—elasticity, low cost, high availability, global coverage, and so forth.

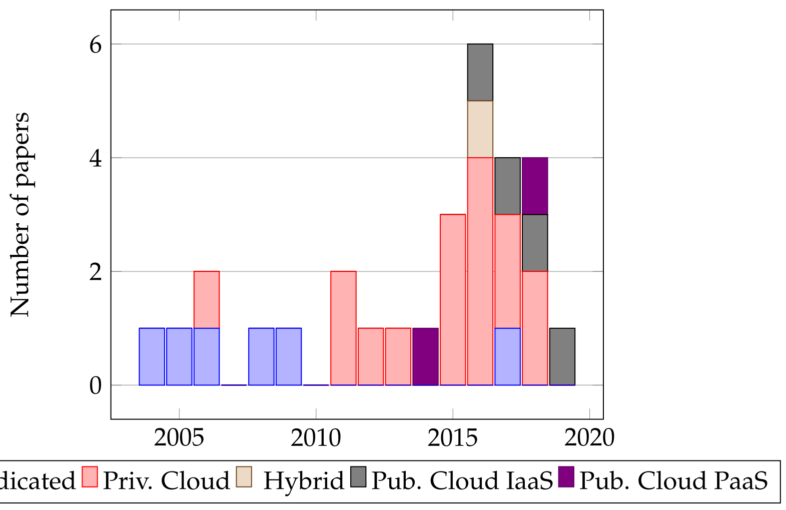

Figure 1. Choice of infrastructure over time—as derived from the studied works.

6.2. Architecture

Regarding architectures, a majority of the papers opted for the well-known *client-server* model (52%), while *peer-to-peer* architectures were less utilized at 31%. There also appears to be a trend towards hybrid architectures—at least among researchers. Figure 2 shows that the studied papers have attempted to build on all types of architecture for MMOG backends over the years. While the client-server model is slightly more popular, not one architecture appears to be the evident dominant, nor one is trending to become so.

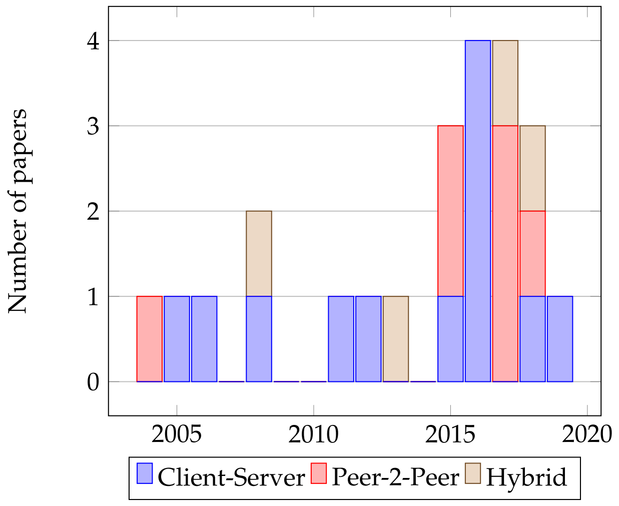

Figure 2. Choice of software architecture over time—as derived from the studied works.

6.3. Performance

In most—but not all—games, the *performance* is a critical feature of the game as it directly affects the quality perceived by the players. The majority of the papers we have studied used simulations to evaluate the performance of their approaches, either applying them to pre-existing MMOGs or other types of case studies and simulations. The majority (73%) of studies assessed their proposals with simulations. Of these, 76% utilized some form of MMOG, either existing or implemented by the authors, while only the remaining 24% utilized another type of application. The use of simulations instead of mathematical modeling highlights the usefulness of this method for performance evaluation of MMOGs. It is also worth noting that most simulations examined simple metrics such as latency, bandwidth and general resource consumption to measure performance, while only a small subset of them used their own composite metrics.

6.4. Scalability

A critical requirement for MMOG backends is *scalability*. Especially in the early phase of deploying a new game, we can expect significant fluctuations in the user demand, so providing an elastic approach is extremely important. Figure 3 shows the scalability of the approaches described in our selected studies over time. Overall, the majority of these approaches (94%) has some form of scalability, either manually scalable or elastic. This confirms the significance of scalability in MMOG backends. Additionally, we observe that non-scalable approaches were only used in the early years of MMOGs and have since been phased out and replaced by manually scalable and elastic approaches. Furthermore—and as anticipated—there appears to be a correlation between *scalability* and *infrastructure*. The increase of cloud infrastructure use in recent years–as it was shown in Figure 1—appears to coincide with more advanced scalability—as shown in Figure 3.

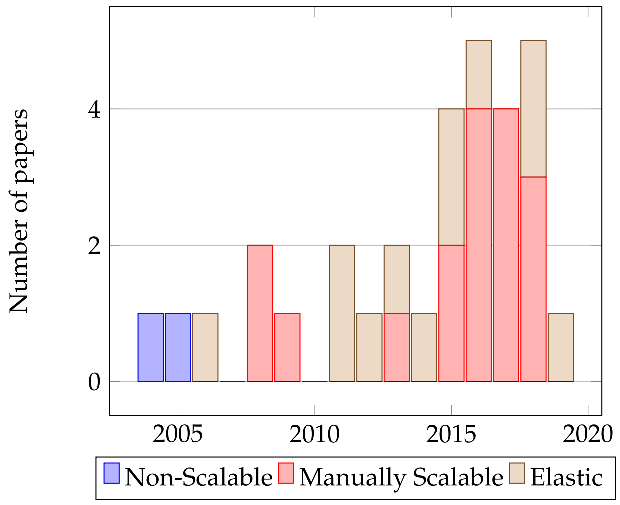

Figure 3. Scalability over time—as determined in the studied works.

6.5. Persistence

The choice of strategy to achieve data *persistence* affects a game's performance and efficiency when scaling. Most games (especially those with persistent worlds) require data persistence to store data like the game state and user information. Database systems are used to enable persistent storage but RDBMSs and NoSQL systems are fundamentally different. Table 4 shows that 5 out of 8 studies (63%) have used NoSQL systems while the remaining (37%) used RDBMSs. More importantly, we can observe that all of the studies prior to 2010 used RDBMSs, while all studies conducted after 2010 have used NoSQL. While the totality of these results cannot be substantiated nor generalized to all existing studies, it shows a clear trend towards using NoSQL databases in MMOGs.

6.6. Security

Last, *security* appears to be mostly overlooked when developing MMOGs. This is reflected in our table, as only 6 out of 34 entries even discuss this criterion. However, the importance of security should not be underestimated. An ever-increasing number of modern games feature micro-transactions involving real money and store personal user data on their servers, making security a necessary aspect of any MMOG. Regardless, these security policies are often discussed in conjunction with P2P architectures, where security needs to be handled more carefully. In such cases, most studies (67%) prefer to use loose security—security based on algorithms and encryption rather than architectural design. The use of tight security—which is based on architectural design– relies on hybrid architectures which make use of the client-server at the system's core to safeguard critical information.

It should be noted that our analysis is limited by the fact that the approaches we have examined do not always align fully with the selected criteria. This is unavoidable, as many of the surveyed works are research-focused and as such, they do not extensively cover all aspects of the underlying system—for example, few have explicitly discussed *security* implications of their proposals.

6.7. Directions for Cloud-Based MMOG Backends

Our survey reveals some useful trends in the industry and has reinforced our hypothesis that commoditizing the development of MMOG backends remains primarily an open problem. Additionally, it showcases that there is a high interest in the industry both for the development of MMOG backends and for utilizing the benefits of cloud computing.

The current state-of-the-art shows that the developers resort primarily to proprietary and public IaaS solutions when it comes to MMOG backends. Although additional methods of hosting do exist, they can be regarded as variants or specializations of these more fundamental categories. While IaaS carries many of the benefits of cloud computing, it does not offer the simplicity and infrastructure-agnostic approach provided in PaaS. Based on the trends revealed by this study and knowledge of PaaS systems, we argue that an opportunity exists to enable MMOG backends deployed directly on PaaS, thus raising the level of abstraction from the developer's perspective.

7. Conclusions

The distributed nature of MMOGs has allowed many of them to become very popular very quickly. At the same time, MMOG developers and researchers have proposed novel solutions to overcome the challenges inherent in the development and deployment of such games and particularly their backends.

In this paper, we conducted a systematic mapping study of the state-of-the-art in software technology used to develop, deploy and operate MMOG backends. We have defined criteria and categories for classifying the different approaches based on a focus analysis of several selected papers. Based on the identified criteria, we studied and analyzed relevant works, placed them into respective categories and then analyzed and summarized our findings. Our results have revealed a trend towards the utilization of cloud-based solutions for realizing MMOG backends, a fact which we argue provides an opportunity for further research and assessment. For instance, the trend towards cloud platforms can possibly continue toward the more abstract layers of PaaS and public clouds. We believe this is a promising direction for research. By developing new models, methods and tools this layer can be utilized and ultimately offer high-level cloud-based solutions for developing and operating MMOG backends more efficiently.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; et al. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58. [CrossRef]
2. Regalado, A. Who Coined 'Cloud Computing'? MIT Technology Review. 2011. Available online: https://www.technologyreview.com/s/425970/who-coined-cloud-computing (accessed on 23 April 2019).
3. Shabani, I.; Kovaçi, A.; Dika, A. Possibilities offered by Google App Engine for developing distributed applications using datastore. In Proceedings of the 2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), Tetova, Macedonia, 27–29 May 2014; pp. 113–118.
4. Bergsten, H. *JavaServer Pages*; O'Reilly: Tokyo, Japan, 2003; pp. 34–37.
5. Ducheneaut, N.; Yee, N.; Nickell, E.; Moore, R.J. Building an MMO With Mass Appeal: A Look at Gameplay in World of Warcraft. *Games Cult.* **2006**, *1*, 281–317. [CrossRef]
6. Krause, S. A case for mutual notification: a survey of P2P protocols for massively multiplayer online games. In Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games, Worcester, MA, USA, 21–22 October 2008; pp. 28–33.
7. Carter, C.; El Rhalibi, A.; Merabti, M. A survey of aoim, distribution and communication in peer-to-peer online games. In Proceedings of the 2012 21st International Conference on Computer Communications and Networks (ICCCN), Munich, Germany, 30 July–2 August 2012; pp. 1–5.
8. Webb, S.; Soh, S. A survey on network game cheats and P2P solutions. *Aust. J. Intell. Inf. Process. Syst.* **2008**, *9*, 34–43.
9. Gilmore, J.S.; Engelbrecht, H.A. A survey of state persistency in peer-to-peer massively multiplayer online games. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *23*, 818–834. [CrossRef]
10. Abdulazeez, S.A.; El Rhalibi, A.; Merabti, M.; Al-Jumeily, D. Survey of solutions for Peer-to-Peer MMOGs. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Anaheim, CA, USA, 16–19 February 2015; pp. 1106–1110.
11. Cai, W.; Shea, R.; Huang, C.Y.; Chen, K.T.; Liu, J.; Leung, V.C.; Hsu, C.H. A survey on cloud gaming: Future of computer games. *IEEE Access* **2016**, *4*, 7605–7620. [CrossRef]

12. Kitchenham, B.; Charters, S.; Budgen, D.; Brereton, P.; Turner, M.; Linkman, S.; Jorgensen, M.; Mendes, E.; Visaggio, G. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; EBSE Technical Report Version 2.3; EBSE: Goyang, Korea, 2007.
13. Chu, H.S. Building a Simple Yet Powerful MMO Game Architecture. Verkkoarkkitehtuuri Part. 2008. Available online: https://www.ibm.com/developerworks/library/ar-powerup1/ (accessed on 23 April 2019).
14. Google. Overview of Cloud Game Infrastructure. 2018. Available online: https://cloud.google.com/solutions/gaming/cloud-game-infrastructure (accessed on 5 March 2019).
15. Shaikh, A.; Sahu, S.; Rosu, M.C.; Shea, M.; Saha, D. On demand platform for online games. *IBM Syst. J.* **2006**, *45*, 7–19. [CrossRef]
16. Nae, V.; Prodan, R.; Iosup, A. Massively multiplayer online game hosting on cloud resources. *Cloud Comput. Princ. Paradig.* **2011**, *19*, 491–509.
17. Shaikh, A.; Sahu, S.; Rosu, M.; Shea, M.; Saha, D. Implementation of a service platform for online games. In Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games, Portland, OR, USA, 30 August–3 September 2004; pp. 106–110.
18. Barri, I.; Roig, C.; Giné, F. Distributing game instances in a hybrid client-server/P2P system to support MMORPG playability. *Multimed. Tools Appl.* **2016**, *75*, 2005–2029. [CrossRef]
19. LeadingEdgeTech.co.uk. How Is Cloud Computing Different from Traditional IT Infrastructure? 2019. Available online: https://www.leadingedgetech.co.uk/it-services/it-consultancy-services/cloud-computing/how-is-cloud-computing-different-from-traditional-it-infrastructure/ (accessed on 17 March 2019).
20. Nae, V.; Iosup, A.; Prodan, R. Dynamic resource provisioning in massively multiplayer online games. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 380–395. [CrossRef]
21. Dhib, E.; Boussetta, K.; Zangar, N.; Tabbane, N. Modeling Cloud gaming experience for Massively Multiplayer Online Games. In Proceedings of the 13th IEEE Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2016; pp. 381–386.
22. Mishra, D.; El Zarki, M.; Erbad, A.; Hsu, C.H.; Venkatasubramanian, N. Clouds+ games: A multifaceted approach. *IEEE Internet Comput.* **2014**, *18*, 20–27. [CrossRef]
23. Satyanarayanan, M.; Bahl, V.; Caceres, R.; Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]
24. Najaran, M.T.; Krasic, C. Scaling online games with adaptive interest management in the cloud. In Proceedings of the 2010 9th Annual Workshop on Network and Systems Support for Games, Taipei, Taiwan, 16–17 November 2010; pp. 1–6.
25. Dhib, E.; Zangar, N.; Tabbane, N.; Boussetta, K. Resources allocation trade-off between cost and delay over a distributed Cloud infrastructure. In Proceedings of the 2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), Hammamet, Tunisia, 18–20 December 2016; pp. 486–490.
26. Dhib, E.; Boussetta, K.; Zangar, N.; Tabbane, N. Cost-aware virtual machines placement problem under constraints over a distributed cloud infrastructure. In Proceedings of the 2017 Sixth International Conference on Communications and Networking (ComNet), Hammamet, Tunisia, 29 March–1 April 2017; pp. 1–5.
27. Zahariev, A. *Google App Engine*; Helsinki University of Technologyx: Espoo, Finland, 2009; pp. 1–5.
28. Nae, V.; Prodan, R.; Fahringer, T.; Iosup, A. The impact of virtualization on the performance of massively multiplayer online games. In Proceedings of the 8th Annual Workshop on Network and Systems Support for Games, Paris, France, 23–24 November 2009; p. 9.
29. Negrão, A.P.; Veiga, L.; Ferreira, P. Task based load balancing for cloud aware massively Multiplayer Online Games. In Proceedings of the 2016 IEEE 15th International Symposium onNetwork Computing and Applications (NCA), Cambridge, MA, USA, 24–27 October 2016; pp. 48–51.
30. Assiotis, M.; Tzanov, V. A distributed architecture for massive multiplayer online role-playing games. In Proceedings of the 5th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames' 06), Hawthorne, NY, USA, 10–11 October 2005.
31. GauthierDickey, C.; Zappala, D.; Lo, V. Distributed Architectures for massively multiplayer online games. In Proceedings of the ACM NetGames Workshop, Portland, OR, USA, 30 August 2004.
32. Kavalionak, H.; Carlini, E.; Ricci, L.; Montresor, A.; Coppola, M. Integrating peer-to-peer and cloud computing for massively multiuser online games. *Peer-to-Peer Netw. Appl.* **2015**, *8*, 301–319. [CrossRef]

33. Mildner, P.; Triebel, T.; Kopf, S.; Effelsberg, W. Scaling online games with NetConnectors: A peer-to-peer overlay for fast-paced massively multiplayer online games. *Comput. Entertain. (CIE)* **2017**, *15*, 3. [CrossRef]
34. Jardine, J.; Zappala, D. A hybrid architecture for massively multiplayer online games. In Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games, Worcester, MA, USA, 21–22 October 2008; pp. 60–65.
35. Matsumoto, K.; Okabe, Y. A Collusion-Resilient Hybrid P2P Framework for Massively Multiplayer Online Games. In Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy, 4–8 July 2017; Volume 2, pp. 342–347.
36. Zhang, W.; Chen, J.; Zhang, Y.; Raychaudhuri, D. Towards efficient edge cloud augmentation for virtual reality mmogs. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose, CA, USA, 12–14 October 2017; p. 8.
37. Plumb, J.N.; Kasera, S.K.; Stutsman, R. Hybrid network clusters using common gameplay for massively multiplayer online games. In Proceedings of the 13th International Conference on the Foundations of Digital Games, Malmo, Sweden, 7–10 August 2018; p. 2.
38. Blackman, T.; Waldo, J. *Scalable Data Storage in Project Darkstar*; Technical Report; Sun Microsystems: Menlo Park, CA, USA, 2009.
39. Lu, F.; Parkin, S.; Morgan, G. Load balancing for massively multiplayer online games. In Proceedings of 5th ACM SIGCOMM workshop on Network and System Support for Games, Singapore, 30–31 October 2006; p. 1.
40. Chuang, W.C.; Sang, B.; Yoo, S.; Gu, R.; Kulkarni, M.; Killian, C. Eventwave: Programming model and runtime support for tightly-coupled elastic cloud applications. In Proceedings of the 4th Annual Symposium on Cloud Computing, Santa Clara, CA, USA, 1–3 October 2013; p. 21.
41. Nae, V.; Prodan, R.; Fahringer, T. Cost-efficient hosting and load balancing of massively multiplayer online games. In Proceedings of the 2010 11th IEEE/ACM International Conference on Grid Computing, Brussels, Belgium, 25–28 October 2010; pp. 9–16.
42. Meiländer, D.; Gorlatch, S. Modeling the scalability of real-time online interactive applications on clouds. *Future Gener. Comput. Syst.* **2018**, *86*, 1019–1031. [CrossRef]
43. Farlow, S.; Trahan, J.L. Periodic load balancing heuristics in massively multiplayer online games. In Proceedings of the 13th International Conference on the Foundations of Digital Games, Malmo, Sweden, 7–10 August 2018; p. 29.
44. Weng, C.F.; Wang, K. Dynamic resource allocation for MMOGs in cloud computing environments. In Proceedings of the 2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, Cyprus, 27–31 August 2012; pp. 142–146.
45. Ghobaei-Arani, M.; Khorsand, R.; Ramezanpour, M. An autonomous resource provisioning framework for massively multiplayer online games in cloud environment. *J. Netw. Comput. Appl.* **2019**, *142*, 76–97. [CrossRef]
46. Brewer, E. Spanner, Truetime and the Cap Theorem. 2017. Available online: https://ai.google/research/pubs/pub45855 (accessed on 20 August 2019).
47. Vogels, W. Eventually consistent. *Commun. ACM* **2009**, *52*, 40–44. [CrossRef]
48. Agrawal, D.; Das, S.; El Abbadi, A. Big data and cloud computing: current state and future opportunities. In Proceedings of the 14th International Conference on Extending Database Technology, Uppsala, Sweden, 21–24 March 2011; pp. 530–533.
49. Chang, F.; Dean, J.; Ghemawat, S.; Hsieh, W.C.; Wallach, D.A.; Burrows, M.; Chandra, T.; Fikes, A.; Gruber, R.E. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst. (TOCS)* **2008**, *26*, 4. [CrossRef]
50. Baker, J.; Bond, C.; Corbett, J.C.; Furman, J.; Khorlin, A.; Larson, J.; Leon, J.M.; Li, Y.; Lloyd, A.; Yushprakh, V. Megastore: Providing Scalable, Highly Available Storage for Interactive Services. In Proceedings of the Conference on Innovative Data System Research (CIDR), Asilomar, CA, USA, 9–12 January 2011; pp. 223–234.
51. Diao, Z.; Zhao, P.; Schallehn, E.; Mohammad, S. Achieving Consistent Storage for Scalable MMORPG Environments. In Proceedings of the 19th International Database Engineering & Applications Symposium, Yokohama, Japan, 13–15 July 2015; pp. 33–40.
52. Diao, Z. Cloud-based Support for Massively Multiplayer Online Role-Playing Games. Ph.D. Thesis, University of Magdeburg, Magdeburg, Germany, 2017.

53. Shea, R.; Liu, J.; Ngai, E.C.H.; Cui, Y. Cloud gaming: architecture and performance. *IEEE Netw.* **2013**, *27*, 16–21. [CrossRef]
54. Lin, Y.; Shen, H. Cloud fog: Towards high quality of experience in cloud gaming. In Proceedings of the 2015 44th International Conference on Parallel Processing, Beijing, China, 1–4 September 2015; pp. 500–509.
55. Gascon-Samson, J.; Kienzle, J.; Kemme, B. Dynfilter: Limiting bandwidth of online games using adaptive pub/sub message filtering. In Proceedings of the 2015 International Workshop on Network and Systems Support for Games, Zagreb, Croatia, 3–4 December 2015; p. 2.
56. Yusen, L.; Deng, Y.; Cai, W.; Tang, X. Fairness-aware Update Schedules for Improving Consistency in Multi-server Distributed Virtual Environments. In Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques, Prague, Czech Republic, 22–23 August 2016; pp. 1–8.
57. El Rhalibi, A.; Al-Jumeily, D. Dynamic Area of Interest Management for Massively Multiplayer Online Games Using OPNET. In Proceedings of the 2017 10th International Conference on Developments in eSystems Engineering (DeSE), Paris, France, 14–16 June 2017; pp. 50–55.
58. Baughman, N.E.; Levine, B.N. Cheat-proof playout for centralized and distributed online games. In Proceedings of the IEEE INFOCOM 2001, Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), Anchorage, AK, USA, 22–26 April 2001; Volume 1, pp. 104–113.
59. Jamin, S.; Cronin, E.; Filstrup, B. Cheat-proofing dead reckoned multiplayer games. In Proceedings of the 2nd International Conference on Application and Development of Computer Games, Hong Kong, China, 6–7 January 2003; Volume 67.
60. Lin, Y.; Shen, H. Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of experience. In Proceedings of the IEEE 35th International Conference on Distributed Computing Systems (ICDCS), Columbus, OH, USA, 29 June–2 July 2015; pp. 734–735.
61. Burger, V.; Pajo, J.F.; Sanchez, O.R.; Seufert, M.; Schwartz, C.; Wamser, F.; Davoli, F.; Tran-Gia, P. Load dynamics of a multiplayer online battle arena and simulative assessment of edge server placements. In Proceedings of the 7th International Conference on Multimedia Systems, Klagenfurt, Austria, 10–13 May 2016; p. 17.
62. Plumb, J.N.; Stutsman, R. Exploiting Google's Edge Network for Massively Multiplayer Online Games. In Proceedings of the 2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC), Washington, DC, USA, 3 May 2018; pp. 1–8.
63. Google. Stadia: Take Game Development Further Than You Thought Possible. 2019. Available online: https://stadia.dev/about (accessed on 23 April 2019).
64. Apple. Arcade: Games That Redefine Games. 2019. Available online: https://www.apple.com/lae/apple-arcade (accessed on 23 April 2019)
65. Carter, C.J.; El Rhalibi, A.; Merabti, M. A novel scalable hybrid architecture for MMOG. In Proceedings of the 2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), San Jose, CA, USA, 15–19 July 2013; pp. 1–6.
66. Basiri, M.; Rasoolzadegan, A. Delay-aware resource provisioning for cost-efficient cloud gaming. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *28*, 972–983. [CrossRef]