Towards Supporting Millions of Users in Modifiable Virtual Environments by Redesigning Minecraft-Like Games as Serverless Systems

Jesse Donkervliet
VU Amsterdam

Animesh Trivedi
VU Amsterdam

Alexandru Iosup
VU Amsterdam

Abstract

How can Minecraft-like games become scalable cloud services? Hundreds of Minecraft-like games, that is, games acting as modifiable virtual environments (MVEs), are currently played by over 100 million players, but surprisingly they do not scale and are frequently not published as cloud services. We envision a new architecture for large-scale MVEs, supporting much larger numbers of concurrent users by scaling up and out using serverless technology. In our vision, developers focus on the game (business) logic, while cloud providers manage resource management and scheduling (RMS) and guarantee non-functional properties. We provide a definition for MVEs, model their services and deployments, present a vision for large-scale MVEs architected as serverless systems, and suggest concrete steps towards realizing this vision.

1 Introduction

Games provide entertainment to hundreds of million of players worldwide. In 2019, this entertainment industry generated a revenue of $152.1 billion [52], which is more than four times the global box office revenue [17] and ten times the music industry's revenue [58]. We focus in this work on an emerging type of game, exemplified by Minecraft, which offers players a *modifiable virtual environment (MVE)*. Minecraft is one of the most popular games of all time: it has sold more than 176 million copies [55] and has over 100 million active monthly players [25] (which, in comparison, is more than the global number of MacOS users [54]). Surprisingly, Minecraft uses replicated, non-communicating game instances, which do not scale beyond a few hundred concurrent players [68]. We envision MVEs with millions of concurrent players together in a seamless virtual world. In this paper we make a case to redesign these MVE environments as serverless systems.

The unique features of MVEs are generally beneficial. By allowing users to construct and deconstruct the world in complex ways, MVEs enable creative user-behavior that is currently not possible in other games. MVEs are also useful for other important societal tasks. Minecraft: Education Edition contains lessons on a diverse set of topics, for example, computer science lessons in which students construct their own digital computers and history lessons in which they explore UNESCO world-heritage sites [1]. Minecraft has also been used for social activism, for example, in protecting Europe's last primeval forest from illegal logging [51].

> **Definition**: A *modifiable virtual environment* is a real-time, online, multi-user environment which allows its users (i.e., players) to modify the virtual world's objects (e.g., player apparel) and parts (e.g., terrain), create new content by connecting components, and interact with the world through programs.

Through their unique features, MVEs also inspire large communities of developers and players. Modifications (mods) are programmatic game changes, ranging from user-interface tweaks to entire sub-games within the MVE, designed and developed by communities and individuals outside the control of the original game developer or publisher. Modding is popular for MVEs, e.g., there are currently over 55,000 mods available for Minecraft alone [2].

We conclude there exists a strong case for supporting *MVEs through a new class of cloud services, with the lucrative potential of hundreds of millions of players*. However, we argue that enabling cloud services for MVEs requires a careful investigation, motivated by scalability issues.

Although Minecraft has more than 100 million of active players, there is a looming *scalability challenge*. Minecraft's current scalability is only achieved by replicating game instances that do not exchange state. Thus, players join a single instance, and do not interact in-game with users on other instances. Moreover, both recent academic findings and industry practice indicate the *scalability* of a *single game's instance* (analyzed in Section 2.1) is limited. Minecraft instances, and several community-created games using the same protocol, scale only to 300-400 players under favorable condi-

tions [68], and Minecraft Realms, a cloud-operated Minecraft service, does not allow more than 10 simultaneous players in each world-instance [48]. Many games similar to Minecraft likely[1] suffer from similar scalability limitations.

State-of-the-art approaches for scaling traditional online-games, such as world-partitioning [26] and interest management techniques [10,43], are designed to address scalability issues caused by player-avatars, and assume largely immutable, low-complexity virtual worlds. In contrast, MVE players have fine-grained control over the environment by modifying its terrain, components, and programs.

> **Our vision**: Minecraft-like games and, more generally, MVEs, will become cloud-based services scalable to millions of concurrent players (users).

Toward our vision, and aligned with the goals and methods of the research program on massivizing computer systems [31], we propose a novel architecture where MVEs are redesigned as serverless systems, making large-scale MVEs a "killer application" for modern cloud infrastructures. In this article, we set a course towards this vision:

1. We model current MVEs as an set of interconnecting services (in Section 2). We model the services operating a current game instance, and describe how these game instances are currently deployed as cloud services.
2. We envision for large-scale MVEs (in Section 3). In our vision, MVEs are serverless; running as a collection of services. We describe for these services their operation, state synchronization, and deployment.
3. To move towards our vision, we propose three challenging and timely areas of research (in Section 4). The areas are: using serverless functions to schedule MVE services, dynamic consistency units for MVEs, and scheduling MVE services at cloudlets.

2 MVE Model

In this section, we model the services operating a current game instance, and describe how the scalability of these game instances is limited, despite their cloud deployment.

2.1 Inside MVEs: the Minecraft-like Game

This section discusses the common distributed architecture used for MVEs, depicted in the top part of Figure 1.

MVEs typically use *client-server* architectures, where each player runs locally a client that connects to a remote server. The *client* caches in memory the part of the world which is

[1] Anecdotally, the authors of this vision paper have personally experienced the lack of scalability of several Minecraft alternatives. No systematic study of this situation currently exists.

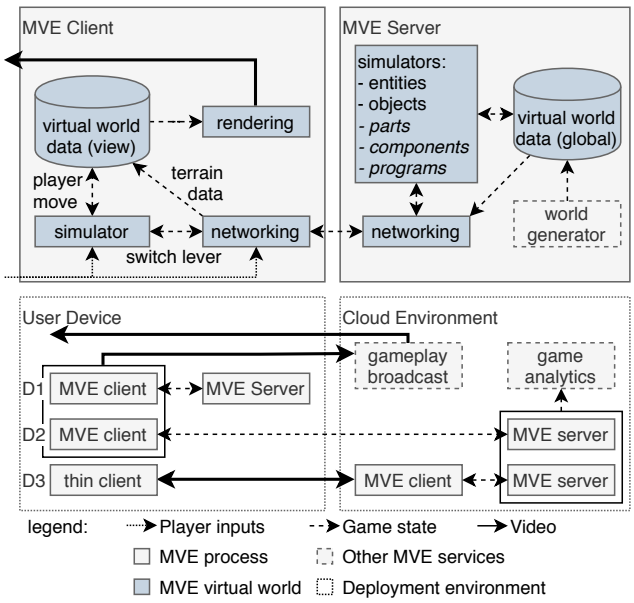

Figure 1: MVE processes and services (top row), and current deployment models (bottom row).

of interest to the player, typically, around the player's representation (avatar). The client aims to render this part of the world approximately 30 to 60 times per second, and to show the result to the player. The player can interact with the world by performing various MVE-specific actions. A networking component sends these actions to the server, where their effects on the world are simulated. Concurrently, the networking component continuously receives state updates from the server, which are then shown to the user. To give the illusion of continuous updates yet provide some game-state *consistency*, clients may speculatively simulate actions locally while waiting for state updates.

The *server* is commonly architected as a multi-threaded monolithic application with three important responsibilities. Firstly, it maintains a persistent copy of the game state, which includes the terrain and the status of all players and non-playable characters (NPCs). Secondly, it simulates the virtual world, which includes both traditional services (e.g., player actions and NPCs) and MVE-specific services (e.g., components and player-written programs), typically 20 times per second, creating an illusion of continuous time to the player. Thirdly, MVE-specific, the server generates procedurally an endless world for players to explore.

Adding the MVE-specific services to the server simplifies the design, but limits how services can scale independently.

2.2 Current Deployment Models for MVEs

In this section, we discuss three common ways of deploying MVEs. Each of these is shown in the lower half of Figure 1.

The *self-hosted* deployment model (D1) is for operators, or players if the game server is sufficiently easy to manage, to host their own MVE server. This method requires buying and

maintaining own infrastructure. For example, many players deploy Minecraft under this model, but other Minecraft-like games can require considerable tuning and maintenance.

Following pioneering work arguing during the late-2000s for operating online games in clouds as opposed to self-hosting [40, 49], MVEs are starting to be offered as cloud-based services. There are two key models of cloud deployment. Firstly, in the *Game as a Service* model (D2), players pay a fee to the cloud operator, who takes responsibility for the infrastructure and resource management and scheduling (RMS) required to run the MVE server. Players run only their client software, and connect to the cloud-based MVE server. Minecraft Realms exemplifies these services.

Secondly, in the *streamed games* model[2] (D3), both the server and the client are moved to the cloud. Players run only a *thin client*, which sends their inputs to the cloud, and receives a fully-rendered video-stream of the player's local view. This moves the responsibility of rendering to the cloud, giving access to the virtual world even to resource-constraint devices. However, in contrast to the previous methods, it requires high-bandwidth, low-latency, and low-jitter network connections from player to cloud datacenter [3, 13].

The three deployment models move the client and server between the user-owned and cloud-operated infrastructure, but limit individual service-scaling.

3 Vision: Serverless MVEs

Single MVE instances do not scale beyond a few hundred players [68], despite increasingly being deployed in clouds. Towards scaling to millions of single-world players, we envision serverless MVEs. In our vision, depicted in Figure 2:

1. MVEs become serverless: they are split into services, operated professionally by the cloud operator, and provisioned on-demand by the game developer or operator. This improves elasticity and multi-tenancy.
2. MVEs require specialized consistency models, which take into account the trade-off between various quality of service (QoS) (e.g., latency) and quality of experience (QoE) guarantees. This offers ways to control the bandwidth budget yet offer good gameplay experience.
3. MVE services run on the user's device, the cloud, or on *cloudlets*, depending on available resources and requirements. This enables QoS and QoE improvements.

3.1 Serverless Operation

While state-of-the-art scalability techniques assume servers to primarily simulate players, MVEs can additionally contain arbitrarily complex connections of components and programs that can change the entire environment with fine-granularity.

[2]This modeled was mislabeled as "cloud gaming" in the early 2010s, by OnLive and similar services, as a marketing stunt.

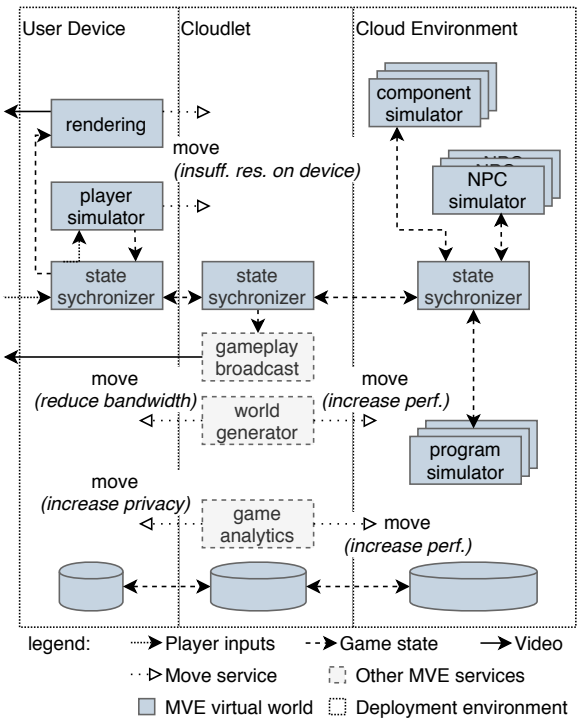

Figure 2: Our vision: a serverless large-scale MVE.

In our vision, MVEs can scale by using *serverless computing*. Serverless computing is an emerging cloud architecture in which systems run as a collection of small services, or *functions* [22, 34]. A function contains executable code which is executed in isolation by the arrival of events [4, 53, 71].

Figure 2 depicts a high-level view of the possible services; in practice, each could be comprised of several functions. We identify the *virtual world* and three other classes of MVE services. The virtual world services (blue, solid outline) provide an immersive experience to players. The *rendering service* translates game state into video through which the player experiences the virtual world. The *player simulator* translates player actions (e.g., mouse clicks) into game actions (e.g., terrain modification). Because these services process directly the player's input, their response latency needs to be of 10s of milliseconds to maintain QoE, and thus should run close to the player [12, 16, 76]. *NPCs*, *components*, and *programs* are simulated using independent service instances. The latter two services are unique to MVEs. To allow interaction between the services, they exchange volatile state through a system-wide state synchronization service, and persistent game state through a distributed database. Because these services can interact with all players, they run in a central cloud.

The other three important classes of MVE services (gray, dashed outline) are (i) *game analytics*, which enables game operators to extract information from the large amounts of data produced by the game; (ii) *procedural content generation (PCG)*, which uses world generators to create an endless virtual world; and (iii) *meta-gaming network* services, which

allow players to interact outside the virtual world. *Gameplay broadcasting* is a popular meta-gaming service, which allows players to live-stream gameplay to thousands of viewers [32]. Although these services are not part of the interactive virtual world, they provide an important supporting role and fit well in our vision of serverless MVEs.

In our vision of serverless MVEs, its services are independently scheduled by the cloud operator, leading to high scalability and elasticity. How players interact with each other and with the environment has a significant effect on the workload, even for traditional games [50]. The unique features of MVEs can magnify this phenomenon [68]. Scheduling many small, independent services is conveniently parallel, which leads to high scalability. Moreover, adding and removing services is fine-grained, leading to good elasticity properties [27]. (Services that need to exchange state, so where consistency is an issue, are considered in Section 3.2.)

We conjecture serverless operation could lead to performance isolation for MVEs, even under multi-tenancy. For example, a serverless rendering service (as shown in Figure 2) can be specialized to execute a single task (i.e., rendering), thus, ensuring low performance variability in gaming [35]. This simplifies performance isolation and makes it easier for a cloud operator to consolidate multiple such services together on the same hardware, even in the presence of accelerated hardware, for example, GPUs [41].

Building MVEs as a collection of services increases the modularity of these systems, making it easier for players to create content (e.g., mods), a driving force behind the popularity of Minecraft and commercially available game engines.

3.2 Specialized Consistency Models for An Immersive Virtual World

Essential for the MVE is to provide an *immersive virtual world* where players must not experience significant inconsistency with other players, when performing one of the following four classes of in-game actions: (i) *modifying regular objects*, which entails adding, changing, or removing them from the world; (ii) *modifying the game-world*, e.g., allowing players to construct and remove world-parts such as mountains, valleys, oceans, and other natural phenomena; (iii) *connecting in-game components*, to create new complex constructions that may not be envisioned by the game's developers; and (iv) *supporting player-written, arbitrary programs*, as a way to interact with the world with precise control. The last three classes distinguish MVEs from other (popular) game-genres.

Maintaining a real-time completely consistent MVEs game state is challenging due to large communication overheads. We envision a system-wide service that dynamically determines the importance of each player action to all other nearby players, and synchronizes these using a distributed state synchronization service. This service uses consistency models that quantify the amount of inconsistency in the system, and policies that selectively and temporarily allow inconsistencies in places where players are unlikely to notice. Such consistency models could be applicable to other real-time distributed systems where users need multiple consistency models [66].

3.3 Differentiated Deployment for Different MVE Services

Current deployment models used by MVEs, as shown in Figure 1, do not fit well the requirements for immersive virtual worlds and other MVE services. In our vision, MVE services are deployed across user devices, clouds, and cloudlets, depending on available resources and QoS constraints.

Due to stringent QoS requirements, rendering is typically performed on the user device. However, this can be challenging for devices with limited resources (e.g., smartphones). Scheduling these services on *cloudlets* provides a compromise. It provides lower latency than the cloud, and has more available resources than the user device.

In state-of-the-art gameplay broadcasting, players receive game-state updates from cloud-operated servers, render and record video, and send this back into the cloud for distribution to viewers. In a serverless MVE, this approach can be simplified. A serverless gameplay broadcasting service, scheduled in the cloud, or on cloudlets close to viewers, can obtain the necessary game state through the state-synchronization service, and use it to render and distribute video independently, without further input from the player.

Privacy and data protection regulation (e.g., GDPR) may introduce novel challenges for cloud-based game analytics when the players and the game operator reside in different jurisdictions. Scheduling game analytics closer to the player may provide a solution to these challenges, analyzing data locally and anonymizing it before storing it a central cloud.

4 Towards Scalable, Cloud-Based MVEs

Matching the vision described in Section 3, we describe here three timely research challenges towards supporting millions of players in serverless MVEs. We aim to design novel solutions to these challenges in future work, and apply them to an open-source MVE. To evaluate their scalability improvements, we aim to observe application-level metrics (e.g., maximum number of players, simulation rate) and system-level metrics (e.g., instructions per cycle, bandwidth usage) while benchmarking the system using realistic and heavy workloads (e.g., many players that build and connect components).

> **C1**: Serverless functions for independent scheduling of MVE services.

While MVE services such as world generation may be easily expressed as serverless functions, important research challenges remain. Providing MVE services using serverless functions requires guaranteeing non-functional proper-

ties, which is an open research problem [22]. More advanced features of serverless computing, such as keeping state and communicating with other functions using networking [65] or shared storage [36, 37, 56] are actively areas of research, but may be required to support more complex MVE services. Combining these features may enable a serverless version of distributed actor models [5] (or Actors as a Service [63]), which matches well our proposed architecture.

Using serverless architectures is already being explored for MVE-related services. For example, PyWren [33] and Graphless [67] are serverless systems for data- and graph-processing, respectively, and may enable game analytics.

> **C2**: Dynamic consistency units to control and limit MVE inconsistency.

Continuous consistency [74] is a consistency model that quantifies the inconsistency in a system, and allows the system to operate under arbitrary consistency bounds.

Unfortunately, the real-time nature of MVEs does not tolerate unavailability—some value must always be returned to (rendered for) the user. To address this problem, we propose *dynamic* consistency units as a timely research topic. In such a model, MVE operators specify policies that dynamically determine inconsistency bounds for each player, and even for each type of data—world objects, terrain, etc [66]. A dedicated consistency service then communicates updates to MVE services based on their bounds. We aim to evaluate experimentally the potential for scalability of these models.

> **C3**: Schedule services at cloudlets to resources to improve QoS and cost.

Cloudlets are resource-rich compute clusters connected to the Internet that are specifically available for nearby devices [62], which play an important role in *edge computing*, where such resources are deployed near the Internet's edge, close to end users [61]. Examples include servers available at internet service provider (ISP) end-stations and cell-towers.

Initial research on using cloudlets for gaming is promising. Computational offloading to cloudlets [20] enables high QoS for players with resource constrained devices. Streaming games from cloudlets can deliver low-latency video to users, while reducing bandwidth load from remote clouds [42, 76]. Game analytics may run on processing engines specifically designed for cloudlets at the Internet's edge [24]. Further research could investigate if rendering can take place at multiple cloudlet locations concurrently, to efficiently provide gameplay broadcasting for live audiences. Evaluating the benefits of running other computational MVE services on cloudlets, such as world generation, remains an open research challenge.

The community is exploring how to move serverless systems to the edge [57], and how to use efficiently cloudlets for game workloads [15]. For example, resource sharing between small units of computation on cloudlets [59] is particularly relevant to serverless MVEs. Data storage [72] or caching [75] could make data from all four game functions available to users with low latency and reduced cost. Further research should explore using physical hot-spots as gaming caches.

5 Related Work

We propose here a novel, integrated vision of scalable MVEs, which we see achieved through a combination of diverse systems-level techniques and cloud deployment models.

This article is not the first that argues for increasing the scalability of MVEs. Manycraft [19] increases the scalability of single Minecraft instances to support up to 1,000 players in a static world. Koekepan [21] modifies Minecraft to be used as a research platform and adds scalability through world partitioning. The Yardstick [68] benchmark was used to show Minecraft and other MVEs do not scale well.

The MVE services identified in this article can be built using distributed actors. Implementations of this model are available (e.g., Akka and Orleans [7, 11]), but do not include specialized consistency models for latency-sensitive MVE state changes, and require RMS from the game operator.

The approach we propose leverages existing techniques for scaling horizontally and vertically across servers. Common approaches include partitioning geographical areas of the virtual world [18, 30, 39, 45, 70], and partitioning based on entities [14, 28, 29, 46]. Our vision is a generalization of these approaches, allowing independent scheduling.

Last, dynamic consistency is different from scalability techniques used in traditional large-scale online games [73], such as consistency models for areas of interest [10, 43, 69] and reduced synchronization [8, 9, 47, 60, 64], and consistency models for key-value stores, e.g., Cassandra [38], Pileus [66].

6 Conclusion

In this work, we focus on an emerging type of game, exemplified by Minecraft, which offers players a *modifiable virtual environment* (MVE). Although Minecraft is one of the most popular games of all time, it uses replicated, non-communicating game instances, which do not scale beyond a few hundreds of concurrent players.

We propose large-scale MVEs, run as a collection cloud-native services, as a "killer application" for modern cloud infrastructures. In this article, we set a course towards this vision with a three-fold contribution. Firstly, we model current MVEs as a system of interconnecting services, and describe how these game instances are currently running as cloud services. Secondly, we envision serverless MVEs, which run as independently scheduled services that synchronize using specialized consistency models. Thirdly, we propose serverless computing, dynamic consistency units, and scheduling MVE services on cloudlets as challenging and timely areas of research to move towards this vision.

7 Discussion Topics

Discussion Point and Feedback

1. How to deal with technical limitations of serverless platforms, such as bounded execution time and limited communication possibilities?
2. How to leverage existing consistency models and performance techniques from the area of datacenter storage? We understand that there is a large body of work in the area [38,66] which are optimized for key-value stores. Can we adapt these models for gaming workloads (many writes) and requirements (real-time)?
3. On the use of specialized consistency models and protocols in clouds. We would like a discussion around the topic, not limited to our challenge C2. Are there also practical issues to be considered? Is there experience with running such services in clouds, especially related to performance variability in networks?
4. On the use of cloud-edge resources. How does the community see the practical aspects of such approaches? Any use-cases related to ours?

Controversial Topics

1. Are gaming and MVEs important topics to the HotCloud community? The current paper discusses challenges specifically in the context of gaming, which by itself affects many people, but also represents a broader class of emerging AR/VR applications that interact with a shared world.[3] All these applications need predictable low-latency decision making and data processing, access to a shared state, and flexible scalability. We believe starting with gaming may synthesize reusable abstractions, mechanisms, and ideas applicable to other domains.
2. Are serverless architectures a good fit for MVEs? Players' ability to add and remove (often independent) simulated components to the virtual world fits well with the elasticity provided by serverless architectures. However, MVE simulations may share state with each other, and may need to run for longer periods of time. We are familiar with the many platforms used in serverless operations in the cloud,[4] related analysis studies [6,22,34,44], and use-cases [23,33], but know that one size does not fit all, and we would appreciate feedback and a discussion on the experience of the community in running serverless workloads, especially large-scale and/or with interactive elements.

Acknowledgments

We thank Aurojit Panda and the anonymous reviewers. Work partially funded by the Dutch NWO project MagnaData.

[3] Example of such applications are remote surgery, digital 3D modeling tools in architecture, self driving cars, and online collaboration and teaching.

[4] E.g., Amazon Lambda, Microsoft Azure Functions (family), Google Cloud Functions, IBM Cloud Functions, and the K8s family (Knative, etc.);

References

[1] History Blocks | Minecraft: Education Edition. https://education.minecraft.net/lessons/history-blocks. Accessed: 2020-05-07.

[2] Mods - Minecraft - CurseForge. https://www.curseforge.com/minecraft/mc-mods. Accessed: 2020-05-07.

[3] Maha Abdallah, Carsten Griwodz, Kuan-Ta Chen, Gwendal Simon, Pin-Chun Wang, and Cheng-Hsin Hsu. Delay-sensitive video computing in the cloud: A survey. *TOMM*, 14(3s):54:1–54:29, 2018.

[4] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. Firecracker: Lightweight Virtualization for Serverless Applications. In *NSDI*, pages 419–434, 2020.

[5] Gul Agha. Concurrent Object-Oriented Programming. *CACM*, 33(9):125–141, 1990.

[6] Mohammed Al-Ameen and Josef Spillner. Systematic and open exploration of FaaS and Serverless Computing research. In *Proceedings of the European Symposium on Serverless Computing and Applications, ESSCA@UCC 2018, Zurich, Switzerland, December 21, 2018*, volume 2330, pages 30–35, 2018.

[7] Phil Bernstein, Sergey Bykov, Alan Geller, Gabriel Kliot, and Jorgen Thelin. Orleans: Distributed Virtual Actors for Programmability and Scalability. Technical Report MSR-TR-2014-41, 2014.

[8] Ashwin R Bharambe, John R Douceur, Jacob R Lorch, Thomas Moscibroda, Jeffrey Pang, Srinivasan Seshan, and Xinyu Zhuang. Donnybrook: enabling large-scale, high-speed, peer-to-peer games. In *SIGCOMM*, pages 389–400, 2008.

[9] Ashwin R Bharambe, Jeffrey Pang, and Srinivasan Seshan. Colyseus: A Distributed Architecture for Online Multiplayer Games. In *NSDI*, 2006.

[10] Jean-Sébastien Boulanger, Jörg Kienzle, and Clark Verbrugge. Comparing interest management algorithms for massively multiplayer games. In *NetGames*, page 6, 2006.

[11] Sergey Bykov, Alan Geller, Gabriel Kliot, James R Larus, Ravi Pandya, and Jorgen Thelin. Orleans: cloud computing for everyone. In *SOSP*, page 16, 2011.

[12] John Carmack. Latency mitigation strategies. https://danluu.com/latency-mitigation/ (mirror), 2020. Accessed: 2020-02-15.

[13] Kuan-Ta Chen, Yu-Chun Chang, Hwai-Jung Hsu, De-Yu Chen, Chun-Ying Huang, and Cheng-Hsin Hsu. On the quality of service of cloud gaming systems. *IEEE Trans. Multimedia*, 16(2):480–495, 2014.

[14] Roman Chertov and Sonia Fahmy. Optimistic load balancing in a distributed virtual environment. In *NOSSDAV*, page 13, 2006.

[15] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia Systems*, 20(5):503–519, 2014.

[16] Mark Claypool and Kajal T Claypool. Latency and player actions in online games. *CACM*, 49(11):40–45, 2006.

[17] Comscore. Comscore Reports Highest Ever Worldwide Box Office. https://bit.ly/BoxOffice2020, Feb 2020. Accessed: 2020-02-08.

[18] Yunhua Deng and Rynson W H Lau. Dynamic load balancing in distributed virtual environments using heat diffusion. *TOMCCAP*, 10(2):16:1—-16:19, 2014.

[19] Raluca Diaconu, Joaquin Joaquín Keller, and Mathieu Valero. Manycraft: Scaling Minecraft to Millions. In *NetGames*, pages 1:1—-1:6, 2013.

[20] Luobing Dong, Meghana N Satpute, Junyuan Shan, Baoqi Liu, Yang Yu, and Tihua Yan. Computation Offloading for Mobile-Edge Computing with Multi-user. In *ICDCS*, pages 841–850, 2019.

[21] Herman A. Engelbrecht and Gregor Schiele. Transforming Minecraft into a research platform. In *CCNC*, pages 257–262, 2014.

[22] Erwin Van Eyk, Alexandru Iosup, Simon Seif, and Markus Thömmes. The SPEC cloud group's research vision on faas and serverless architectures. In *Proceedings of the 2nd International Workshop on Serverless Computing, WOSC@Middleware 2017, Las Vegas, NV, USA, December 12, 2017*, pages 1–4, 2017.

[23] Sadjad Fouladi, Riad S Wahby, Brennan Shacklett, Karthikeyan Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads. In *NSDI*, pages 363–376, 2017.

[24] Xinwei Fu, Talha Ghaffar, James C Davis, and Dongyoon Lee. EdgeWise: A Better Stream Processing Engine for the Edge. In *ATC*, pages 929–946, 2019.

[25] Ben Gilbert. How many people are playing 'Minecraft'? Over 112 million every month. https://bit.ly/MCMonthlyPlayers, Sep 2019. Accessed: 2020-02-08.

[26] John S Gilmore and Herman A Engelbrecht. A Survey of State Persistency in Peer-to-Peer Massively Multiplayer Online Games. *TPDS*, 23(5):818–834, 2012.

[27] Nikolas Herbst, André Bauer, Samuel Kounev, Giorgos Oikonomou, Erwin Van Eyk, George Kousiouris, Athanasia Evangelinou, Rouven Krebs, Tim Brecht, Cristina L. Abad, and Alexandru Iosup. Quantifying cloud performance and dependability: Taxonomy, metric design, and emerging challenges. *TOMPECS*, 3(4):19:1–19:36, 2018.

[28] Haiyang Hu, Yizhi Ren, Xu Xu, Liguo Huang, and Hua Hu. Reducing view inconsistency by predicting avatars' motion in multi-server distributed virtual environments. *J. Netw. Comput. Appl.*, 40:21–30, 2014.

[29] Shun-Yun Hu, Jui-Fa Chen, and Tsu-Han Chen. VON: a scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, 2006.

[30] Takuji Iimura, Hiroaki Hazeyama, and Youki Kadobayashi. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In *NetGames*, pages 116–120, 2004.

[31] Alexandru Iosup, Alexandru Uta, Laurens Versluis, Georgios Andreadis, Erwin Van Eyk, Tim Hegeman, Sacheendra Talluri, Vincent van Beek, and Lucian Toader. Massivizing Computer Systems: A Vision to Understand, Design, and Engineer Computer Ecosystems Through and Beyond Modern Distributed Systems. In *ICDCS*, pages 1224–1237, 2018.

[32] Adele Lu Jia, Siqi Shen, Dick H. J. Epema, and Alexandru Iosup. When game becomes life: The creators and spectators of online game replays and live streaming. *TOMCCAP*, 12(4):47:1–47:24, 2016.

[33] Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. Occupy the cloud: distributed computing for the 99%. In *SoCC*, pages 445–451, 2017.

[34] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Jayant Yadwadkar, Joseph E Gonzalez, Raluca Ada Popa, Ion Stoica, and David A Patterson. Cloud Programming Simplified: A Berkeley View on Serverless Computing. *CoRR*, abs/1902.0, 2019.

[35] Ram Srivatsa Kannan, Lavanya Subramanian, Ashwin Raju, Jeongseob Ahn, Jason Mars, and Lingjia Tang. Grandslam: Guaranteeing slas for jobs in microservices execution frameworks. In *EuroSys*, 2019.

[36] Ana Klimovic, Yawen Wang, Christos Kozyrakis, Patrick Stuedi, Jonas Pfefferle, and Animesh Trivedi. Understanding Ephemeral Storage for Serverless Analytics. In *ATC*, pages 789–794, 2018.

[37] Ana Klimovic, Yawen Wang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, and Christos Kozyrakis. Pocket: Elastic Ephemeral Storage for Serverless Analytics. In *OSDI*, pages 427–444, 2018.

[38] Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. *Operating Systems Review*, 44(2):35–40, 2010.

[39] Kyungmin Lee and Dongman Lee. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution. In *VRST*, pages 160–168, 2003.

[40] Yeng-Ting Lee and Kuan-Ta Chen. Is server consolidation beneficial to mmorpg? A case study of world of warcraft. In *CLOUD*, pages 435–442, 2010.

[41] Yusen Li, Chuxu Shan, Ruobing Chen, Xueyan Tang, Wentong Cai, Shanjiang Tang, Xiaoguang Liu, Gang Wang, Xiaoli Gong, and Ying Zhang. GAugur: Quantifying Performance Interference of Colocated Games for Improving Resource Utilization in Cloud Gaming. In *HPDC*, pages 231–242, 2019.

[42] Yuhua Lin and Haiying Shen. CloudFog: Leveraging Fog to Extend Cloud Gaming for Thin-Client MMOG with High Quality of Service. *TPDS*, 28(2):431–445, 2017.

[43] Elvis S Liu and Georgios K Theodoropoulos. Interest management for distributed virtual environments: A survey. *ACM Comput. Surv.*, 46(4):51:1—-51:42, 2014.

[44] Pedro García López, Marc Sánchez Artigas, Gerard París, Daniel Barcelona Pons, Álvaro Ruiz Ollobarren, and David Arroyo Pinto. Comparison of Production Serverless Function Orchestration Systems. *CoRR*, abs/1807.1, 2018.

[45] Fengyun Lu, Simon E Parkin, and Graham Morgan. Load balancing for massively multiplayer online games. In *NetGames*, page 1, 2006.

[46] John C S Lui and M F Chan. An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems. *TPDS*, 13(3):193–211, 2002.

[47] Tom Magrino, Jed Liu, Nate Foster, Johannes Gehrke, and Andrew C Myers. Efficient, Consistent Distributed Computation with Predictive Treaties. In *EuroSys*, pages 36:1—-36:16, 2019.

[48] Mojang. Minecraft Realms. https://www.minecraft.net/en-us/realms-plus. Accessed: 2020-02-08.

[49] Vlad Nae, Alexandru Iosup, Stefan Podlipnig, Radu Prodan, Dick H. J. Epema, and Thomas Fahringer. Efficient management of data center resources for massively multiplayer online games. In *SC*, page 10, 2008.

[50] Vlad Nae, Alexandru Iosup, and Radu Prodan. Dynamic Resource Provisioning in Massively Multiplayer Online Games. *TPDS*, 22(3):380–395, 2011.

[51] Angela Natividad. How Greenpeace Used Minecraft to Stop Illegal Logging in Europe's Last Lowland Primeval Forest. https://bit.ly/MinecraftGreenpeace, Jan 2018. Accessed: 2020-02-08.

[52] Newzoo. Newzoo Global Games Market Report. https://bit.ly/GamesMarketReport2019, 2019. Accessed: 2020-02-08.

[53] Edward Oakes, Leon Yang, Dennis Zhou, Kevin Houck, Tyler Harter, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. SOCK: Rapid Task Provisioning with Serverless-Optimized Containers. In *ATC*, pages 57–70, 2018.

[54] Matthew Panzarino. Apple pushes the reset button on the Mac Pro. https://bit.ly/MacOSUsers, Apr 2017. Accessed: 2020-02-08.

[55] Saxs Persson. Celebrating 10 Years of Minecraft. https://bit.ly/MC10year, May 2019. Accessed: 2020-02-08.

[56] Qifan Pu, Shivaram Venkataraman, and Ion Stoica. Shuffling, fast and slow: Scalable analytics on serverless infrastructure. In *NSDI*, pages 193–206, 2019.

[57] Thomas Rausch, Waldemar Hummer, Vinod Muthusamy, Alexander Rashed, and Schahram Dustdar. Towards a Serverless Platform for Edge AI. In *HotEdge*, 2019.

[58] RIAA. Year-end 2019 RIAA music revenues report. https://bit.ly/MusicIndustry2020, Feb 2020. Accessed: 2020-03-16.

[59] Amit Samanta, Lei Jiao, Max Mühlhäuser, and Lin Wang. Incentivizing Microservices for Online Resource Sharing in Edge Clouds. In *ICDCS*, pages 420–430, 2019.

[60] Nuno Santos, Luís Veiga, and Paulo Ferreira. Vector-Field Consistency for Ad-Hoc Gaming. In *Middleware*, volume 4834, pages 80–100, 2007.

[61] Mahadev Satyanarayanan. The Emergence of Edge Computing. *IEEE Computer*, 50(1):30–39, 2017.

[62] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Cáceres, and Nigel Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[63] Johann Schleier-Smith. Serverless Foundations for Elastic Database Systems. In *CIDR*, 2019.

[64] Siqi Shen, Shun-Yun Hu, Alexandru Iosup, and Dick H J Epema. Area of Simulation: Mechanism and Architecture for Multi-Avatar Virtual Environments. *TOMM*, 12(1):8:1—-8:24, 2015.

[65] Arjun Singhvi, Sujata Banerjee, Yotam Harchol, Aditya Akella, Mark Peek, and Pontus Rydin. Granular Computing and Network Intensive Applications: Friends or Foes? In *HotNets*, pages 157–163, 2017.

[66] Douglas B Terry, Vijayan Prabhakaran, Ramakrishna Kotla, Mahesh Balakrishnan, Marcos K Aguilera, and Hussam Abu-Libdeh. Consistency-based service level agreements for cloud storage. In *SOSP*, pages 309–324, 2013.

[67] Lucian Toader, Alexandru Uta, Ahmed Musaafir, and Alexandru Iosup. Graphless: Toward Serverless Graph Processing. In *ISPDC*, pages 66–73, 2019.

[68] Jerom van der Sar, Jesse Donkervliet, and Alexandru Iosup. Yardstick: A Benchmark for Minecraft-like Services. In *ICPE*, pages 243–253, 2019.

[69] Paolo Viotti and Marko Vukolic. Consistency in Non-Transactional Distributed Storage Systems. *ACM Comput. Surv.*, 49(1):19:1—-19:34, 2016.

[70] Bart De Vleeschauwer, Bruno Van Den Bossche, Tom Verdickt, Filip De Turck, Bart Dhoedt, and Piet Demeester. Dynamic microcell assignment for massively multiplayer online gaming. In *NetGames*, pages 1–7, 2005.

[71] Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, and Michael M Swift. Peeking Behind the Curtains of Serverless Platforms. In *ATC*, pages 133–146, 2018.

[72] Junjie Xie, Chen Qian, Deke Guo, Xin Li, Shouqian Shi, and Honghui Chen. Efficient Data Placement and Retrieval Services in Edge Computing. In *ICDCS*, pages 1029–1039, 2019.

[73] Amir Yahyavi and Bettina Kemme. Peer-to-peer architectures for massively multiplayer online games: A Survey. *ACM Comput. Surv.*, 46(1):9, 2013.

[74] Haifeng Yu and Amin Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Trans. Comput. Syst.*, 20(3):239–282, 2002.

[75] Yiming Zeng, Yaodong Huang, Zhenhua Liu, and Yuanyuan Yang. Joint Online Edge Caching and Load Balancing for Mobile Data Offloading in 5G Networks. In *ICDCS*, pages 923–933, 2019.

[76] Wuyang Zhang, Jiachen Chen, Yanyong Zhang, and Dipankar Raychaudhuri. Towards efficient edge cloud augmentation for virtual reality MMOGs. In *SEC*, pages 8:1—-8:14, 2017.