What Methods Exist for Creating Real-time Massive Multiplayer Experiences, and is there Room for Increasing the Player Limit Beyond ~1k?

Author: Rian Rutherford

Research Introduction

This research essay is looking into methods for supporting experiences with 1,000 players or more in massively multiplayer-online games, and investigating methods that exist for enabling real-time massive multiplayer gameplay. It will also be looking into past, current, and future multiplayer games to grasp the present trajectory of the industry.

MMO means Massive Multiplayer Online. There are many games out there with large player counts that aren't MMOs. Because of this, games with 100+ player lobbies will be considered MMOs or be put in the same category as MMOs even if they do not market themselves as such.

The term 'lobby' will be used to describe experiences with X number of players that perceive themselves to be in a single persistent world with no loading screens or zones/shards separating them and other players into regions. Planetside 2 is an example of a game with a lobby limit of 2,000 players (Yoon. 2013). This word was chosen because game instance, session, and others have specific meanings and using them could cause confusion.

Most successful games that achieve a lobby of 1k or greater are at least a decade old. The industry hasn't seen any new successful titles that come close to a lobby with 1,000 players for about the last decade (Appendix: A).

Basic History of Games with Large Player Counts

'MMO' is sometimes used as a marketing buzzword in industry now. Some games like Palia, released in 2023 (Singularity 6 Corporation. 2023), try to market themselves as an MMO to entice new players, despite not even being MMO in scale. Palia supports 25 players per lobby maximum and has the MMO tag on Epic Games Store and says "cozy sim MMO game" in their description (Epic Games. 2024).

Traditionally MMO used to refer to experiences with over 100 players. This is evident by the fact older games called MMOs used to support at least 1,000 players.

True MMO experiences are games like these on the graph below:

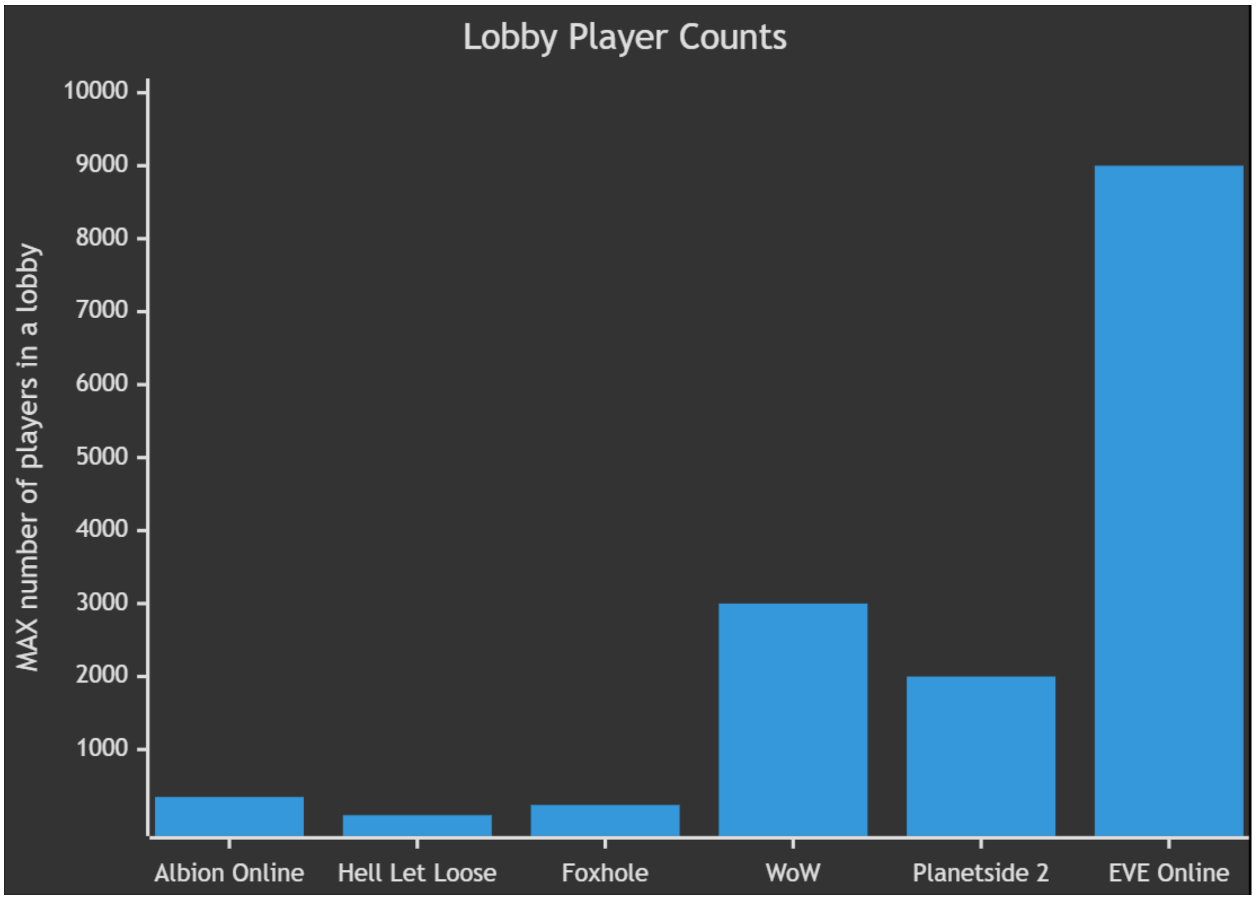


Fig. 1: Bar graph showing the difference between games in terms of lobby player count. Appendix: A, has the data used to make the graph. See Appendix: B, for the tool used to make the bar graph.

A trend with these games is that all the new games (see left three games in bar graph) have lower lobby player counts than the old games (see right three games in bar graph). MMO games and games that are MMO like in scale have decreased the player count per lobby over the years. This is the opposite of computer hardware, which has been improving over the last two decades. With the improvements in technology, it can be assumed people will push the boundaries of what is possible, but current trends show otherwise.

There are many factors that affect the lobby limit of games: game design, software architecture, hardware architecture, professional knowledge & experience, and economic support from investors & publishers. Richard Bartle (2013) believes the decline in the MMO genre is caused by forgetting what makes them special, development costs, and lack of innovation.

Two of the multiple fixes Richard Bartle suggests are:

1. Enable and allow designers to experiment (Bartle, 2013. Page 8: Let designers design).
2. Modular technology to support easier iteration and prototyping (Bartle, 2013. Page 6: Development).

He believes if this were done new MMOs can be prototyped and created without the huge upfront development costs that currently prevent companies from making new MMOs. It also means that we'll see innovation in the MMO genre because modular technology would allow ideas to be tried with lower overhead in terms of time and money spent.

Networking Topologies Used for Multiplayer Games

The networking topologies for multiplayer games are:

1. Single Server
2. Server Meshing
3. Peer-to-Peer
4. Hybrid Peer-to-Peer & Sever

Single Servers are the most common way of making multiplayer games. This is how multiplayer games were originally made, and how most game developers choose to make their games today. They are chosen because they're the easiest option for developing multiplayer games, have the lowest network latency possible reducing gameplay lag, and prevent cheaters because the server controls everything.

The drawback is multiplayer games that use first-person, or third-person, player characters can't exceed lobbies more than approximately 200 players. Some genres can do more, but on average most are limited to 100 – 200 player lobbies.

For example, Foxhole was unable to get more than 150 players working using a single server for their top-down 3D World War 2 like game (Foxhole. 2019).

Server Meshing is far less popular than Single Server because while it can support 1,000+ players it is complicated to create. There aren't any existing game engines or frameworks that implement server meshing in a generic manner that can be used to create new game titles. Usually, Server Meshing is custom made as proprietary software for the specific game that's being developed.

There are multiple ways of implementing Server Meshing. Most companies that go this route will need to develop their own proprietary technology over a few years for the specific game they are making.

The most basic form of Server Meshing is the master/node or database/node architecture. The concept is simple, you have a single master server or database which is responsible for storing all game data and is responsible for tracking which nodes are currently using which parts of the data. The nodes are individual game servers that connect to the master server or database and handle the individual players and simulate parts of the game world. The relationship is 1:many:many, where 1 is the master server/database, the first many are the gameplay server nodes, and the last many are the clients/players that are connected to those nodes.

An example of server meshing is MultiPaper (MultiPaper. 2024). The default vanilla server version of Minecraft is limited to 250 players (MultiPaper. 2024), to overcome this limit some people developed MultiPaper which is a vanilla implementation of the Minecraft server software. What makes MultiPaper different is that it uses Server Meshing instead of Single Server allowing it to scale to over 1,000 player lobby. One of the current users of MultiPaper is a YouTuber called Marcusk who uses it to run events with 1,000 players (Appendix: C).

Another version of Server Meshing mentioned in "Matrix: Adaptive Middleware for Distributed Multiplayer Games" by Balan, Ebling, Castro, and Misra (2005) is where you have a server, and as portions of the world become more densely populated the server spawns server(s) to handle different sub regions, and the sub-server(s) can do the same thing recursively. This creates a tree-like structure where the server scales to multiple servers and back as demand fluctuates.

The benefit of the method proposed by Balan, Ebling, Castro, and Misra (2005) is it can scale servers dynamically to match demand without restarting, fragmenting, or lots of complicated algorithms. Dynamically increasing and decreasing the number of servers used creates cost saving since IDLE servers can be turned OFF automatically.

Balan, Ebling, Castro, and Misra (2005. Page 9: Summary of Other Results) speculate this method of dynamic Server Meshing could support 1 million players. This is possibly true because Novaquark was able to get 30,000 player bots working in the same lobby (Dual Universe. 2019), and they used a similar dynamic Server Meshing method when they implemented Server Meshing in their game Dual Universe (2016). Novaquark never reached these numbers when the game was released, so it is unclear how scalable their implementation is.

This type of server meshing by Dual Universe (2016); and Balan, Ebling, Castro, and Misra (2005) will be called *Recursive Server Meshing* for the rest of the paper.

Peer-to-Peer had a lot of research around 2005 - 2011 when MMO games (MMOG) were extremely popular (Baquero. 2023). You can still find research on the topic as late as 2016 (Abdulazeez, El Rhalibi, and Al-Jumeily. 2016).

Game developers are motivated to make games that use peer-to-peer (P2P) for multiplayer instead of Single Server or Sever Meshing because in P2P there are no servers and therefore no server cost. While it is technically true a server is required to create the initial P2P connection, the cost is negligible. It's so minor that Steam and Epic Games both provide these P2P connection setup servers for free (Steamworks. 2024. *Features*) (Steamworks. 2024. *Multiplayer*) (Epic Games. 2024).

P2P also scales naturally with demand since the game is hosted by the playing players themselves. Because of these two properties, no servers and natural scaling, P2P has been researched over the years (Baquero. 2023) to see if it could be a viable solution for MMOGs. In theory it provides a solution that could support an unlimited number of players in a single lobby and cost almost nothing to run.

Many of the researched peer-to-peer solutions for MMOGs share the same concepts but have subtle differences. Network groups, or similar, is something they all have in common. Network groups are separate P2P groups where players join or leave P2P groups based on their location in the game world (Knutsson, Lu, Xu, and Hopkins. 2004). This creates one seamless persistent lobby without a player limit or loading screens for all players playing the game. To implement this grouping mechanism, many solutions use the Voronoi Diagram (Aurenhammer. 1991).

The problem with P2P games is that they're more complicated to develop than Single Server, have higher network latency when compared to Single Server or Server Meshing which leads to a poor player experience, and it's hard to prevent cheating in P2P games making cheating very common.

Many researchers ignore the cheating aspect when looking into how to use P2P for MMOGs and instead focus on how to manage the network groups and maintain a constant world without weird bugs. This is an issue because without a viable solution to prevent cheating MMOGs will never be able to use P2P technology without ruining the player experience because of hackers plaguing the game in every aspect.

Two proposed solutions for preventing hackers are having a trusted player that is responsible for preventing others from cheating, and a voting system that all players participate in which is used to kick players believed to be hacking (Liu, and Lo. 2008).

Both methods for handling cheaters are easy to bypass. In the first solution for example, the trusted player could also be the hacker which means they can do whatever they want with no one to stop them. In the second solution for example, most of the players in the region are hackers. You can now kick all other players from the game, cheat and make items without collecting resources, then move to another network group after hacking and play as normal. There would be no way to detect the items that were made without resources.

Hybrid Peer-to-Peer & Sever uses servers for player dense regions or specific player activities like 40 player dungeons with an end boss; then other small activities use P2P grouping (Barri, Roig, and Giné. 2016).

Unfortunately, this method often has the same drawbacks as P2P because the same exploits that can be done in P2P still exist in *Hybrid Peer-to-Peer & Server* depending on implementation. In some cases, certain types of exploits or hacks might not be possible because of the hybrid nature. Regardless there will still be hacking on any of the P2P network groups which will negatively affect the player experience.

Identifying the Optimal Technologies to Support 1k+ Player Lobbies

Future MMO games should use:

1. Server Meshing
2. IPv6
3. Low-level Languages (C, C++, or Rust)
4. Data-Driven Programming

Server Meshing is the best solution for creating a game that scales to support over 1,000 players. This is because *Single Servers* do not scale well above an estimated ~200 players; and *Peer-to-Peer* solutions do not have any decent methods for preventing hacking while also being complicated to implement.

Of the multiple ways Server Meshing can be implemented, Recursive Server Meshing should be the preferred implementation to use because it naturally scales with the number of players online. This leads to saving money because you only use hardware when it’s needed instead of preserving and using hardware that currently doesn’t have any player traffic.

IPv6 is the latest version of the internet protocol and will eventually replace IPv4 (Siddiqui. 2017). While not true for everyone on the public internet, where IPv6 is properly implemented it’s more efficient than IPv4 (Zakari, Musa, Bekaroo, Bala, Hashem, and Hakak. 2019). IPv6 is more efficient because the protocol enables packets to take shorter routes between origin and destination over the internet. This means you can increase the amount of data you send to or receive from players with better connections on IPv6.

This is important because for the player to see all the other players in their field of view or render distance you need to have enough bandwidth to send everything. If latency is reduced, then players have less input lag which results in an improved player experience.

Lower latency means player inputs can be sent directly to the server and processed, and the server sends a response/update instead of the current prediction/rollback methods used. This would reduce complexity when making multiplayer games and reduce potential exploits because networking would be simpler.

Low-level Languages (C, C++, or Rust) compile to code that runs more efficiently resulting in more data processed or less power consumption if compared to workloads done in higher level languages like C# or Java (Kostya, Slesarev. 2023).

Using C, C++, or Rust sometimes comes at the cost of increasing development times compared to C# or Java because when working on large codebases it takes longer for changes to compile, therefore testing and iteration takes longer.

If you have enough time, resources, and a team with sufficient experience to invest in making your codebase in C, C++, or Rust then it should be considered. Development will likely be slightly slower and more expensive than with Java or C#, but it pays off later. Your application will use less memory and execute faster than C# or Java resulting in more work done on the same hardware with the same amount of power consumed.

By using a programming language that performs operations more efficiently, you will reduce recurring monthly server running costs which means in the long term the game will be cheaper to operate. Improved efficiency also enables more players per server node in a server mesh.

Data-Driven Programming is where you treat things as collections of data, and you have systems that operate on collections of data. This is different to Object Orientated Programming (OOP) where data and systems are combined to make unique objects.

The benefits of *data-driven programming:*

1. Improved CPU cache utilization because it uses cache prefetching to load memory into the CPU cache before it is used which prevents unused CPU cycles.
2. Supports multithreading by executing multiple systems at the same time on the data. It's also easy to find which parts can be multithreaded because you can look at systems to find out what they do with the data to know if they can run in parallel.
3. There is reduced random memory access preventing cache prefetch misses that waste CPU cycles because data is organized into continuous collections of data in memory instead of being pointers to many different parts of memory.

One implementation of *data*-driven programming is Entity Component Systems (ECS). Two known ECS runtimes are Flecs (Mertens. 2024) and Bevy (Anderson. 2024): Flecs is a framework implemented in C, and Bevy is a game engine implemented in Rust. ECS runtimes are liked because they provide the benefits of data-driven programming while also being modular allowing them to be easily extended. Code written for one application can be easily ported to other applications that use the same ECS framework/runtime/library.

All these factors make ECS efficient at scale when handling lots of data, which is what an MMO server is doing. Using an existing ECS runtime like Bevy or Flecs should be considered by any modern MMO because it will improve hardware utilization leading to reduced server expenses.

The Likely Future of 1k+ MMO Experiences

There are fewer MMOs created that can support 1,000+ player lobbies than there used to be. Games that call themselves MMOs with small lobby counts are increasingly becoming more common. These new games should be called Live Service games instead of MMOs because they don't meet the scale of what is considered an MMO like Planetside 2 or EVE Online do. We can see these games are becoming more common because of a recent release called Palia (limited to 25 player lobbies), and another upcoming MMO in development codenamed Ghost by Fantastic Pixel Castle (Fantastic Pixel Castle. 2023) which is unlikely to support more that 100 players per shard/lobby because their MMO is based on sharding. Fantastic Pixel Castle hasn't stated the lobby limit, but since they want to use shards, like how Palia does it, it can be assumed that shards likely won't be able to hold more than 100 players at most.

MMOs that focus on scale are dying, it's highly probable that in 5 years time we'll have a whole generation of gamers who have never seen or played an MMO at the scale of 1,000+ players.

From a technology standpoint the best way to fix this problem is to utilize IPv6, data-oriented programming, Sever Meshing, and low-level programming languages to make the technology powering MMOGs. Developers who want to make MMOs that support 1,000+ players will also need to research & test game designs for MMOs and make their systems modular to facilitate research & testing.

If these things are done the technology for making 1,000+ player lobby MMOs will exist, helping to reduce the decline of 1,000+ player MMOs. There will be more variety of MMOs because there will be opportunities to explore and test new ideas.

Bibliography

Sandbox Interactive. 2017. *Albion Online*. [video game]. Platforms: PC (Windows, Linux, mac OS).

Albion Online Wiki. 2022. *SMART CLUSTER QUEUE*. [website page]. Available at: https://wiki.albiononline.com/wiki/Smart_Cluster_Queue [accessed 08/02/2024]

Black Matter. 2019. *Hell Let Loose*. Platforms: PC (Windows), PlayStation 5, Xbox Series X/S.

Team17. 2024. *Hell Let Loose*. [website page]. Available at; https://www.team17.com/games/hell-let-loose/ [accessed 08/02/2024].

Siege Camp. 2022. *Foxhole*. [video game]. Platforms: PC (Windows).

Steam. 2022. *Less Than 2K Players?* [website forum]. Available at: https://steamcommunity.com/app/505460/discussions/0/3419936083036029102/ [accessed 12/02/2024].

Blizzard Entertainment. 2004. *World of Warcraft*. [video game]. Platforms: PC (Windows & Mac OS X).

Quora. 2024. *How many players can play one game of World of warcraft at the same time?* [website forum]. Available at: https://www.quora.com/How-many-players-can-play-one-game-of-World-of-warcraft-at-the-same-time [accessed 12/02/2024].

Reddit. 2024. *Does anyone know the actual server player limit? How many players can be on a server before a queue starts?* [website forum]. Available at: https://www.reddit.com/r/classicwow/comments/d7hhq8/does_anyone_know_the_actual_server_player_limit/ [accessed 12/02/2024].

Rogue Planet Games. 2012. *Planetside 2*. [video game]. Platforms: PC (Windows).

Yoon, A. 2013. *Sony Online Entertainment to use PlanetSide 2 engine for all its future MMOs*. [website article]. Available at: https://www.shacknews.com/article/78613/sony-online-entertainment-to-use-planetside-2-engine-for-all [accessed 06/02/2024].

CPP Games. 2003. *EVE Online*. [video game]. Platforms: PC (Windows, Linux, Mac OS X).

EVE Online. 2024. *Two Guinness World Records Titles Broken!* [website article]. Available at: https://www.eveonline.com/news/view/two-guinness-world-records-titles-broken [accessed 12/02/2024].

CPP Games. 2024. *EVE Online Breaks Two GUINNESS WORLD RECORDS™ TITLES in One Day with Biggest Battle in Video Game History*. [website article]. Available at: https://www.ccpgames.com/news/2020/eve-online-breaks-two-guinness-world-records-tm-titles-in-one-day-with [accessed 12/02/2024].

Singularity 6 Corporation. 2023. *Palia*. [video game]. Platforms: PC(Windows), Nintendo.

Epic Games. 2024. *Palia*. [game website store page]. Available at: https://store.epicgames.com/en-US/p/palia-0d428e [accessed 15/02/2024].

BARTLE, A. Richard. 2013. *The Decline of MMOs*. United Kingdom: University of Essex. Available at: https://mud.co.uk/richard/The%20Decline%20of%20MMOs.pdf [accessed 15/02/2024].

Foxhole. 2019. *Devblog 61*. [website blog]. Available at: https://www.foxholegame.com/post/devblog-61 [accessed 14/03/2024].

MultiPaper. 2024. *How MultiPaper works*. [website page]. Available at: https://multipaper.io/howitworks.html [accessed 14/03/2024].

Balan, R.K., Ebling, M., Castro, P. and Misra, A. 2005. *Matrix: Adaptive middleware for distributed multiplayer games*. In Middleware 2005: ACM/IFIP/USENIX 6th International Middleware Conference, Grenoble, France, November 28-December 2, 2005. Proceedings 6 (pp. 390-400). Springer Berlin Heidelberg. Available at: https://ink.library.smu.edu.sg/sis_research/1207 [accessed 14/03/2024].

Dual Universe. 2016. *Dual Universe DevDiary - Massively Multiplayer Server Technology Pre-Alpha Video*. [youtube video]. Available at: https://www.youtube.com/watch?v=QeZtqoydXpc [accessed 14/03/2024].

Dual Universe. 2019. *Dual Universe | 30,000 simulated players in a continuous single shard*. [youtube video]. Available at: https://www.youtube.com/watch?v=pLh2OoTj8vc [accessed 14/03/2024].

Baquero, C. 2023. *What Ever Happened to Peer-to-Peer Systems?* [ACM article]. Available at: https://cacm.acm.org/blogcacm/what-ever-happened-to-peer-to-peer-systems/ [accessed 14/03/2024].

Abdulazeez, S.A., El Rhalibi, A. and Al-Jumeily, D. 2016. *Evaluation of scalability and communication in MMOGs*. In 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 393-398). IEEE. Available at: https://researchonline.ljmu.ac.uk/id/eprint/2462/1/PID3979995.pdf [accessed 14/03/2024].

Steamworks. 2024. *Features*. [service documentation]. Available at: https://partner.steamgames.com/doc/features [accessed 21/03/2024].

Steamworks. 2024. *Multiplayer*. [service documentation]. Available at: https://partner.steamgames.com/doc/features/multiplayer [accessed 21/03/2024].

Epic Games. 2024. *CONNECT YOUR COMMUNITY ACROSS ALL PLATFORMS*. [website page]. Available at: https://dev.epicgames.com/en-US/services [accessed 21/03/2024].

Knutsson, B., Lu, H., Xu, W. and Hopkins, B., 2004. *Peer-to-peer support for massively multiplayer games*. In IEEE INFOCOM 2004 (Vol. 1). IEEE. Available at: https://www.cs.ubc.ca/~krasic/cpsc538a/papers/p2p-games-infocom04.pdf [accessed 17/03/2024].

Aurenhammer, F. 1991. *Voronoi diagrams—a survey of a fundamental geometric data structure*. ACM Computing Surveys (CSUR), 23(3), pp.345-405. Available at: https://dl.acm.org/doi/pdf/10.1145/116873.116880 [accessed 17/03/2024].

Liu, H.I. and Lo, Y.T. 2008. *DaCAP-a distributed Anti-Cheating peer to peer architecture for massive multiplayer on-line role playing game*. In 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID) (pp. 584-589). IEEE. Available at: https://ieeexplore.ieee.org/abstract/document/4534267 [accessed 17/03/2024].

Barri, I., Roig, C. and Giné, F. 2016. *Distributing game instances in a hybrid client-server/P2P system to support MMORPG playability*. Multimedia Tools and Applications, 75, pp.2005-2029. Available at: https://link.springer.com/article/10.1007/s11042-014-2389-0 [accessed 17/03/2024].

Siddiqui, A. 2017. *RFC 8200 – IPv6 has been standardized*. [internet society article]. Available at: https://www.internetsociety.org/blog/2017/07/rfc-8200-ipv6-has-been-standardized/ [accessed 17/03/2024].

Zakari, A., Musa, M., Bekaroo, G., Bala, S.A., Hashem, I.A.T. and Hakak, S. 2019. *IPv4 and IPv6 protocols: A Comparative performance study*. In 2019 IEEE 10th Control and System Graduate Research Colloquium (ICSGRC) (pp. 1-4). IEEE. Available at: https://ieeexplore.ieee.org/abstract/document/8837050 [accessed 17/03/2024].

Kostya., Slesarev, A. 2023. *benchmarks*. [github repository]. Commit a107833fcf21311bba9d2eb9fecc6a29198aa553. Available at: https://github.com/kostya/benchmarks [accessed 19/03/2024].

Mertens S. 2024. *Flecs*. [C programming framework]. Available at: https://www.flecs.dev/flecs/ [accessed 19/03/2024].

Anderson C. 2024. *Bevy*. [Rust game engine]. Available at: https://bevyengine.org/ [accessed 19/03/2024].

Fantastic Pixel Castle. 2023. *About Fantastic Pixel Castle*. [website page]. Available at: https://fantasticpixelcastle.com/ [accessed 01/04/2024].

Works Consulted

Donkervliet, J., Trivedi, A. and Iosup, A. 2020. *Towards supporting millions of users in modifiable virtual environments by redesigning Minecraft-Like games as serverless systems*. In 12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20). Available at: https://www.usenix.org/system/files/hotcloud20_paper_donkervliet.pdf [accessed 21/03/2024].

Macedomia, M.R., Zyda, M.J., Pratt, D.R., Brutzman, D.P. and Barham, P.T. 1995. *Exploiting reality with multicast groups: A network architecture for large-scale virtual environments*. In Proceedings Virtual Reality Annual International Symposium'95 (pp. 2-10). IEEE. Available at: https://ieeexplore.ieee.org/abstract/document/512473 [accessed 21/03/2024].

Hu, Y., Bhuyan, L.N. and Feng, M., 2012. *P2P consistency support for large-scale interactive applications*. Computer Networks, 56(6), pp.1731-1744. Available at: https://www.cs.ucr.edu/~bhuyan/papers/COMNET-2012.pdf [accessed 21/03/2024].

Diaconu, R., Keller, J. and Valero, M. 2013. *Manycraft: Scaling minecraft to millions*. In 2013 12th annual workshop on network and systems support for games (NetGames) (pp. 1-6). IEEE. Available at: https://inria.hal.science/hal-01079509/document [accessed 21/03/2024].

Hu, S.Y., Chang, S.C. and Jiang, J.R. 2008. *Voronoi state management for peer-to-peer massively multiplayer online games*. In 2008 5th IEEE Consumer Communications and Networking Conference (pp. 1134-1138). IEEE. Available at: https://www.researchgate.net/publication/224305544_Voronoi_State_Management_for_Peer-to-Peer_Massively_Multiplayer_Online_Games [accessed 21/03/2024].

Seweso. 2014. *Are there any peer-to-peer mmo's? And if not, wouldn't a blockchain like bitcoin be ideal for that?* [forum post]. Available at: https://www.reddit.com/r/gamedev/comments/1ra3jm/are_there_any_peertopeer_mmos_and_if_not_wouldnt/ [accessed 21/03/2024].

Appendix

A) Old & New MMOs

Recent Games on Graph:

- Albion Online, released 2017 (Sandbox Interactive. 2017). Supports 350 player lobbies (Albion Online Wiki. 2022) - This wiki is an official source.
- Hell Let Loose, released 2019 (Black Matter. 2019). Supports 100 player lobbies (Team17. 2024).
- Foxhole, released 2022 (Siege Camp. 2022). Supports 250 player lobbies (Steam. 2022).

Old Games on Graph:

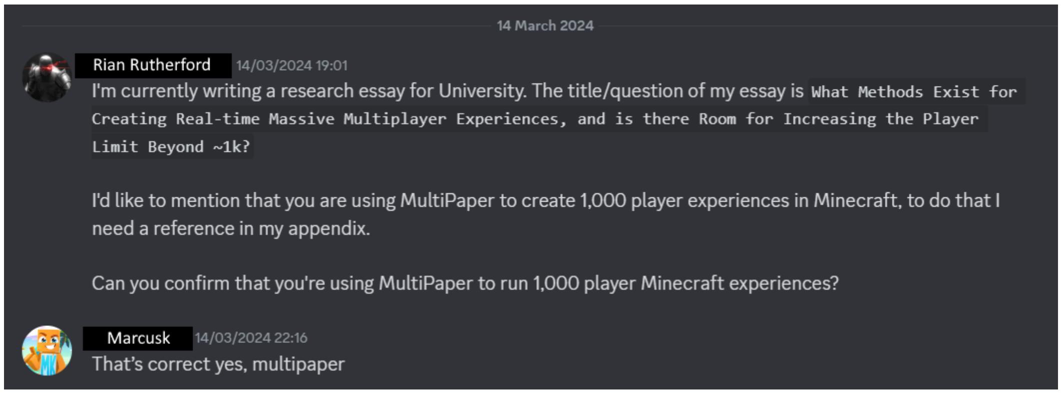
- World of Warcraft, released 2004 (Blizzard Entertainment. 2004). No one knows the lobby limit, but it's speculated to be over 3,000 players (Quora. 2024) (Reddit. 2024).
- Planetside 2, released 2012 (Rogue Planet Games. 2012). Supports 2,000 player lobbies (Yoon. 2013).

- EVE Online, released 2003 (CPP Games. 2003). The game does not have a lobby limit, but instead slows down to allow more players in the lobby. The most players ever in an EVE Online lobby was about 9,000 players (EVE Online. 2024) (CPP Games. 2024).

B) Mermaid

Used mermaid.js website editor to make the graph:
https://mermaid.live/edit#pako:eNo9UMFqwzAM_RWhswNZux2Ww6CsHTu0bDBYx-oelEZdzRyrOA4klP77lIROh2fx0JOe3wUPUjEWaANodf3hRDFlJSeamOSSZ7C4lrLs4d1TzxGepQ2psTdRRp1rYGdx4UsnAd6Cd4EtGtW9svew5gRrkWbkXqQ7iWcDW9kOE7ozcGpcxTCbNKvP1f-O_XSjn25Y3Cy-ILR1qS7kCOfRTwMuAIEfLFqEuxyy7EkfrUldUoTd_CE3A2lgdq8wz8d2xEfFPRqsOdbkKo3jMggtphPXaqLQtqL4O_z4qnPUJvnowwGLFFs22J4rSrx09BOpxuJlvlGWK5ckbqZ8x5gNnil8i9xmrn-t43Yv

C) Multipaper 1k


14 March 2024

Rian Rutherford 14/03/2024 19:01
I'm currently writing a research essay for University. The title/question of my essay is `What Methods Exist for Creating Real-time Massive Multiplayer Experiences, and is there Room for Increasing the Player Limit Beyond ~1k?`

I'd like to mention that you are using MultiPaper to create 1,000 player experiences in Minecraft, to do that I need a reference in my appendix.

Can you confirm that you're using MultiPaper to run 1,000 player Minecraft experiences?

Marcusk 14/03/2024 22:16
That's correct yes, multipaper