Meterstick: Benchmarking Performance Variability in Cloud and Self-hosted Minecraft-like Games

Ir. Jesse Donkervliet
Prof. dr. ir. Alexandru Iosup

Jerrit Eickhoff
M.Sc. @ TU Delft, AtLarge Research
jerrit.eickhoff@gmail.com
https://atlarge-research.com/opencraft/

Source and data available!
Meterstick: https://github.com/atlarge-research/Meterstick
Data: https://zenodo.org/record/7657838

ICPE 2023
Coimbra, Portugal
Why Minecraft-like Games?

Massively popular:
- Video games are the largest entertainment industry
- Minecraft is the best-selling video game of all time
- More than 173 million people play Minecraft per month
- Thriving industry of third-party content creation

Societally beneficial:
- (not just) Entertainment
- Education
- Activism
- Social Interaction

Global revenue of entertainment industry sectors 2021, in Billions USD

Total sales of highest selling video games, in Millions of copies sold

Minecraft: Connecting More Players Than Ever Before

by Helen Chiang, Studio Head, Mojang Studios  •  May 18, 2020 @ 6:00am

Sources: Newzoo, Statista, Motion Picture Association, IFPI, The Verge, Dot Esports, CurseForge, Xbox, Uncensored Library, Mojang
Minecraft-like Games

Not just Minecraft! Whole genre, characterized by:

• Realtime interaction
• Dynamic, modifiable environments
• Server-client architecture

Multiplayer services typically not operated by the developer, but instead community-hosted.
The Problem

Massively popular == incredibly scalable, right?

Minecraft music festival Block By Blockwest postponed after servers crash
Over 100,000 people logged on to catch virtual performances by Massive Attack and more
By Patrick Clarke | 26th April 2020

Isolated instances which do not scale beyond a few hundred players.¹

Does not account for performance impact of cloud-hosting or environment-based workloads.

A single player can overload or crash Minecraft-like games!

Sources:
A single player crashing the game!? How can this be?

Server-Client Architecture

Minecraft-like Game Workload Model

High Volume
- Backups
- Terrain
- Players

Infrequent Updates
- Meta-State

Low Volume
- Entities

Frequent Updates
- Terrain Simulation
  - Terrain Generation
  - Physics Simulation
  - Lighting
  - Plant Growth
  - Simulated Constructs

Player Behavior
- Movement
- Interactions
- Terrain Modification

Assumed Game State
Volume and Update Frequency

Environment-Based Workload

Inefficient in MLGs

Unique to MLGs → Causes Creates

1: State concerning functional operation of the game server rather than game features, such as administrative logs or user authentication tokens.
Player Workload

Player Avatars Sparse

Players Avatars Dense

Eve Online 13,700 player battle causes performance disruptions

Sources: Polygon
Environment-Based Workloads

Environment comprised of modifiable elements, each with unique properties and individual state. Dynamic, modifiable properties make environment workload in Minecraft-like Games 

**Inefficient:**

- Require state of neighboring elements
- Recalculation on updated state

![Diagram showing physics simulation and lighting effects](image)
Environment-Based Workloads

Entities

- Exists in environment, but is **not player or terrain**

- **Hostile Mob**
- **Passive Mob**
- **Item Entities**

**Cannot be precomputed!**
Environment-Based Workloads

Simulated Constructs

- **Player-constructed** structures consisting of dynamic elements
- “Programmed” to **automatically perform** some in-game task

Automatic resource processing

Logic Gates

Operational 16-bit, 1Hz computer

Sources: Reddit, Reddit
Meterstick Benchmark: Design

- Supports **environment-based workloads**
- Uses **player-emulation** for player contribution to workload
- Deploys Minecraft-like Games experiments on **commercial clouds**
- Collects relevant **application and system metrics**

**Steps:**

1. Deployment
2. Experiments
3. Data retrieval
Meterstick Benchmark: Design

- Workloads, Player Emulation, and Metric Externalization tied, directly or indirectly, to application protocol
- Currently supports Minecraft-like games utilizing the Minecraft protocol

1: Ranges from 150 thousand to 2 million downloads, with some individual mods reaching 223 million downloads. See TechicPack and CurseForge.

Popular mod packs

*Same server technology for different games!*
Instability Ratio (ISR)

- **Stability > lowest latency** for online gaming [1-3]
- **Normalized** measure of instability given a trace of tick durations, based on cycle-to-cycle jitter.
- **Order dependent**

\[
ISR = \frac{\sum_{i=1}^{N_a} | \max(b, t_i) - \max(b, t_{i-1}) |}{N_e \times 2b}
\]

\(b\) = minimum delay between ticks
\(t_i\) = duration of \(i^{th}\) tick
\(N_a\) = actual number of ticks
\(N_e\) = expected number of ticks

**Sources:**
2: Player Perception of Delays and Jitter in Character Responsiveness, Normoyle et al. SAP2014
Instability Ratio (ISR)

- **ISR = 0** if all ticks below $b$!
- **ISR = 0** if all ticks are the same!
- Not meant to be used as standalone performance metric!

\[
ISR = \frac{\sum_{i=1}^{Na} \left| \max(b, t_i) - \max(b, t_{i-1}) \right|}{Ne \times 2b}
\]

- $b$ = minimum delay between ticks
- $t_i$ = duration of $i^{th}$ tick
- $Na$ = actual number of ticks
- $Ne$ = expected number of ticks

Tick duration over time

Trace resulting in ISR = 0

Overloaded! Tick duration = $b$
Experiment - Setup
Minecraft-like Games

Environments
- Minecraft
  Default
- Forge
  Mods
- PaperMC
  Performance

Workloads:
- Control*: Freshly generated world
- TNT*: Fast entity actions, terrain updates
- Farm*: Many simulated constructs
- Lag*: Simulated construct stress test
- Players: 25 moving players in small area
  *Only one player, stationary

Hardware Guidelines:
- 2vCPU: AWS: T3.Large, Azure: Standard_D2_v3

<table>
<thead>
<tr>
<th>Service</th>
<th>vCPU[#]</th>
<th>CPU Speed [GHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server.pro</td>
<td>2</td>
<td>2.4</td>
</tr>
<tr>
<td>Skynode</td>
<td>2</td>
<td>3.6</td>
</tr>
<tr>
<td>Hostinger</td>
<td>3</td>
<td>NP</td>
</tr>
<tr>
<td>Ferox Hosting</td>
<td>Not reported</td>
<td>Not reported</td>
</tr>
<tr>
<td>MelonCube</td>
<td>Not reported</td>
<td>3.4</td>
</tr>
<tr>
<td>Azure</td>
<td>2</td>
<td>Variable</td>
</tr>
<tr>
<td>AWS</td>
<td>1</td>
<td>Variable</td>
</tr>
</tbody>
</table>

Full list of cloud service Minecraft-like game hosting recommendations, and community simulated constructs, available in technical report: https://arxiv.org/abs/2112.06963
Environment-based workloads cause significant performance instability

Sources for Noticeable, Unplayable thresholds:
1: Analysis of factors affecting players' performance and perception in multiplayer games, Dick et al. Netgames 2005
2: Are 100 ms Fast Enough? Characterizing Latency Perception Thresholds in Mouse-Based Interaction, Forch et al. EPCE 2017

Whiskers to 5th, 95th percentiles
Cloud environments cause significant performance variability

Variation of **Instability Ratio** and **Tick time** over 50 iterations of Players workload

Whiskers to 1.5 x IQR
Actionable Insights

Researchers: • Performance analysis of online games should include stability analysis! Common statistical measures can hide performance problems

*Same mean, median, deviation, quantiles, etc., but vastly different instability!*
Actionable Insights

Researchers: • Performance analysis of online games should include stability analysis! Common statistical measures can hide performance problems

Game Developers: • Environment-based workloads severely impact scalability

Assumed Game State Volume and Update Frequency

Revised Game State Volume and Update Frequency
Actionable Insights

**Researchers:** • Performance analysis of online games should include stability analysis! Common statistical measures can hide performance problems

**Game Developers:** • Environment based workloads even more of a scalability concern in Minecraft-like games than previously thought

• Situation improvable by performance engineering, much to be done

Case Study: **PaperMC**
Simulation quality vs. performance tradeoff
Asynchronous threading + heuristics
Actionable Insights

**Researchers:** • Performance analysis of online games should include stability analysis! Common statistical measures can hide performance problems

**Game Developers:** • Environment based workloads even more of a scalability concern in Minecraft-like games than previously thought

• Situation improvable by performance engineering, much to be done

**Cloud Providers:** • Revise hardware recommendations for Minecraft-like games
Actionable Insights

Researchers: • Performance analysis of online games should include stability analysis! Common statistical measures can hide performance problems

Game Developers: • Environment based workloads even more of a scalability concern in Minecraft-like games than previously thought
  • Situation improvable by performance engineering, much to be done

Cloud Providers: • Revise hardware recommendations for Minecraft-like games

Server Hosts: • Compare cloud providers for your Minecraft-like game, consider self hosting
Future Work

• User studies to directly link our **Instability Ratio (ISR)** values to player-perceived quality of experience

• Public **leaderboard of Meterstick scores**, allow players, game designers, and cloud platforms to compare results!

Source and Data Available!

Meterstick:  
[https://github.com/atlarge-research/Meterstick](https://github.com/atlarge-research/Meterstick)

Data:  
[https://zenodo.org/record/7657838](https://zenodo.org/record/7657838)
Selected Opencraft Articles

**Serverless gaming**
Servo: A Use-Case for Serverless Computing in Online Gaming
*Jesse Donkervliet, Javier Ron, Junyan Li, Tiberiu Iancu, Cristina L. Abad and Alexandru Iosup. ICDCS 2023*

**Dynamic consistency**
Dyconits: Scaling Minecraft-like Services through Dynamically Managed Inconsistency
*Jesse Donkervliet, Jim Cuijpers, and Alexandru Iosup. ICDCS 2021*

**Benchmarking online games**
Meterstick: Benchmarking Performance Variability in Cloud and Self-hosted Minecraft-like Games
*Jerrit Eickhoff, Jesse Donkervliet, and Alexandru Iosup. ICPE 2023*

Yardstick: A Benchmark for Minecraft-like Services
*Jerom van der Sar, Jesse Donkervliet, and Alexandru Iosup. ICPE 2019*