Nexthosting

Study · Game servers

Server stress test: How many protocol bots can Vanilla, Paper, Purpur and Fabric with Lithium handle?

Four server programs for Minecraft and three JVM profiles, measured with protocol bots on a test VM

NOT AN OFFICIAL MINECRAFT PRODUCT. NOT APPROVED BY OR ASSOCIATED WITH MOJANG OR MICROSOFT.

Type of study	Own measurement
Publisher	Nexthosting
Authorship	Erik Friedl (ORCID: https://orcid.org/0009-0008-2784-4314), Nexthosting
Version	1.0.0 of Nov 22, 2026
Period	Nov 21, 2026 to Nov 22, 2026
Sample	18 of 18 Measurement runs evaluated
License	CC BY 4.0
Online version	nexthosting.net/en/studies/server-haertetest-minecraft

Contents

1	Key results	3
2	Summary	4
3	Results	4
4	Method	10
5	Sample and period	12
6	Limits of what the study can show	13
7	Data and verifiability	14
8	Sources	18
9	Disclosure	19
10	How to cite this study	19
11	Changes and corrections	20

1 Key results

80 bots

Paper with Aikar flags

Averaged over the measurement window, Paper stayed, up to 80 bots in all three repetitions, below the limit of 50 ms p95 MSPT fixed in advance, and in one of them up to 85. With the stricter limit of at least 19.5 TPS on average, it was 80 in all three repetitions.

Evidence: laeufe.csv, Field kapazitaet · laeufe.csv, Field kapazitaet_tps · Source [1]

75 bots

Purpur, not separable from Paper at capacity

Purpur reached 75 to 80 bots. The range overlaps with that of Paper (80 to 85), so this measurement shows no difference between the two.

Evidence: laeufe.csv, Field kapazitaet

65 bots

Fabric with Lithium

Fabric with Lithium reached 65 to 70 bots. That is above Vanilla and below Paper and Purpur; the ranges do not overlap in either direction.

Evidence: laeufe.csv, Field kapazitaet

55 bots

Vanilla

Averaged over the measurement window, Vanilla stayed, in all three repetitions and up to 55 bots, below 50 ms p95 MSPT. With an average of at least 19.5 TPS over the measurement window, it was only 50.

Evidence: laeufe.csv, Field kapazitaet · laeufe.csv, Field kapazitaet_tps

16.31 TPS

The MSPT limit alone is not enough

At 55 bots, Vanilla was below 50 ms p95 MSPT on average over the measurement window, but ran at only 16.31 to 17.16 instead of 20 ticks per second. This is why this study reports the capacity at at least 19.5 TPS alongside the capacity fixed in advance.

Evidence: daten.csv, Field tps · Source [2]

80 bots

Default G1 and Aikar flags: no demonstrated capacity difference

Paper with default G1 reached 80 to 85 bots, and with the Aikar flags likewise 80 to 85. There is no demonstrated difference in capacity. Observed in addition: for the p99 value, the Aikar flags were separately lower in 7 of 13 shared levels, default G1 in none. On average, the Aikar runs came earlier in the run order, so part of the gap may stem from drift over the measurement night (see limitations).

Evidence: laeufe.csv, Field kapazitaet · p99-aikar-g1.csv, Field spieler · Source [1]

70 bots

Generational ZGC came in lower

Paper with Generational ZGC reached 70 bots in all three repetitions, and thus fewer than with default G1 or the Aikar flags (80 to 85 each). This applies to this load profile, this heap and this Java version.

Evidence: laeufe.csv, Field kapazitaet · Source [3]

18 of 18 runs

All runs valid

All 18 runs met the rules fixed in advance. Averaged over a level, CPU steal reached at most 0.061%, and the load generator's cores at most 5.72%.

Evidence: laeufe.csv, Field lauf · daten.csv, Field steal_pct · daten.csv, Field cpu_bots_pct

2 Summary

> **Research question:** How many concurrent protocol bots can Vanilla, Paper, Purpur and Fabric with Lithium handle on the same hardware, and how strongly do JVM settings affect the result?

We put Vanilla, Paper, Purpur and Fabric with Lithium under load in Minecraft 26.3 on the same virtual machine. Protocol bots joined in stages until the server was overloaded. All 18 runs were valid under the rules fixed in advance.

Capacity is the highest number of bots up to which the p95 tick time stayed below 50 ms on average over the measurement window, in all three repetitions. Paper reached 80, Purpur 75, Fabric with Lithium 65 and Vanilla 55 bots. Paper and Purpur cannot be separated at the capacity defined in advance; for all other pairs of programs, the ranges of the repetitions do not overlap.

For the JVM settings, no demonstrated capacity difference appeared on Paper between default G1 and the Aikar flags. Generational ZGC came in below them at 70 bots.

Tick time alone does not reveal every overload: at 55 bots, Vanilla stayed below 50 ms p95 on average, but reached only 16.31 to 17.16 instead of 20 ticks per second. This is why each capacity is accompanied by the value at an average of at least 19.5 TPS. The figures apply to this VM and this bot profile and cannot be transferred directly to real players.

3 Results

How to read the results

The capacity fixed in advance is based on the p95 tick time. The second capacity instead requires that the server manages at least 19.5 of 20 ticks per second on average over the measurement window and that all bots stay connected. Both report the smallest value from three repetitions. A difference counts only if the ranges of the repetitions do not overlap.

Capacity of the programs

Highest bot count with below 50 ms p95 MSPT on average over the measurement window in all three repetitions, all with Aikar flags.



Capacity of the programs at at least 19.5 TPS

Added afterwards: highest bot count with at least 19.5 TPS on average in all three repetitions.



Capacity of Paper per JVM profile

Highest bot count with below 50 ms p95 MSPT on average over the measurement window in all three repetitions.



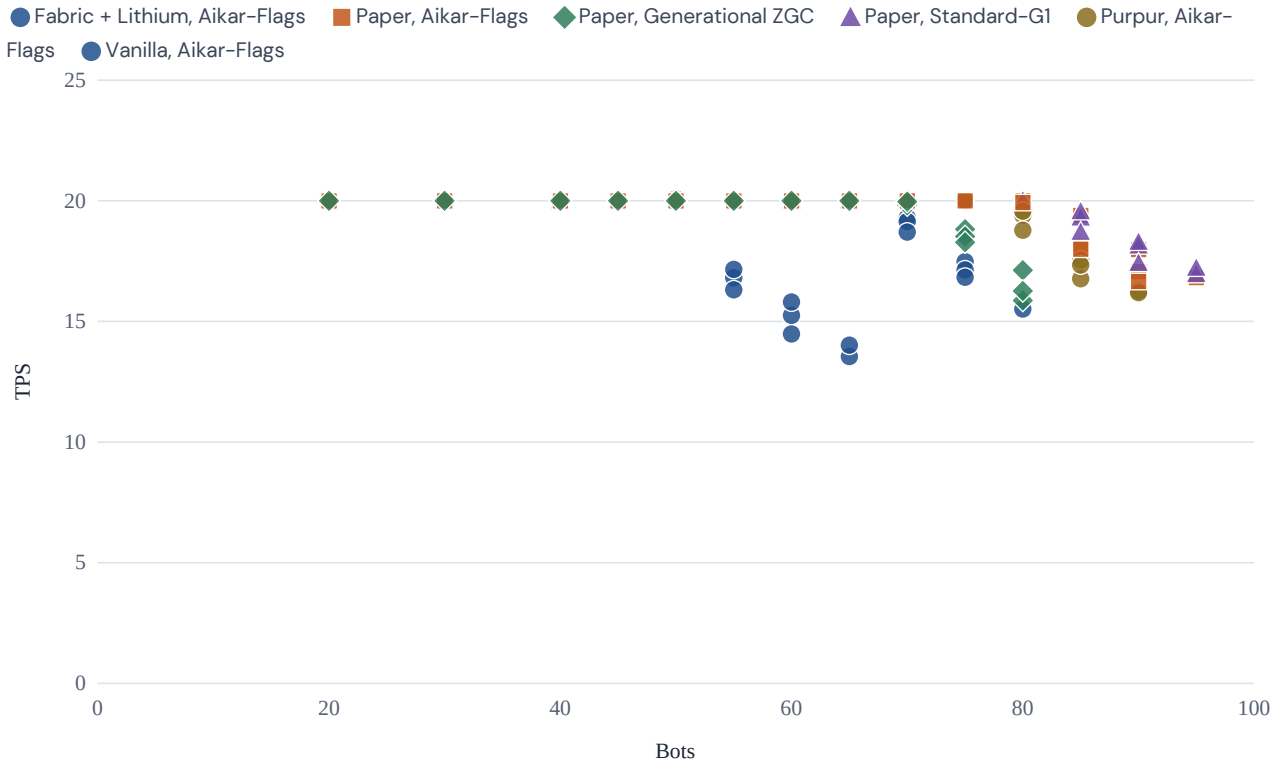
p95 tick time per level

Each point is one level of a run, shown as the mean of the measurement window. The limit of 50 ms applies to this mean.



Actual tick rate per level

Each point is one level of a run. The target is 20 ticks per second.



All combinations

Combination	Capacity	Highest run	At 19.5 TPS	Highest run (TPS)	p95 at 50, min	p95 at 50, max	p99 at 50, min	p99 at 50, max
Paper, Aikar-Flags	80	85	80	80	27.8 ms	29.8 ms	31.2 ms	33.4 ms
Paper, Standard-G1	80	85	80	85	28.3 ms	29.5 ms	34.4 ms	36.1 ms
Purpur, Aikar-Flags	75	80	75	80	29.4 ms	31.0 ms	33.2 ms	35.0 ms
Paper, Generational ZGC	70	70	70	70	34.1 ms	35.4 ms	39.9 ms	40.3 ms
Fabric + Lithium, Aikar-Flags	65	70	65	65	36.1 ms	37.3 ms	43.4 ms	45.0 ms
Vanilla, Aikar-Flags	55	55	50	50	42.7 ms	43.5 ms	50.2 ms	55.1 ms

All 18 runs

No.	Combination	Rep.	Capacity	TPS there	At 19.5 TPS	Valid
1	Paper, Aikar-Flags	2	85	19.40	80	Ja
2	Paper, Aikar-Flags	1	80	19.74	80	Ja
3	Paper, Generational ZGC	1	70	20.00	70	Ja
4	Purpur, Aikar-Flags	1	80	19.40	75	Ja
5	Fabric + Lithium, Aikar-Flags	2	70	19.30	65	Ja
6	Purpur, Aikar-Flags	2	75	20.00	75	Ja
7	Paper, Standard-G1	2	85	19.30	80	Ja
8	Vanilla, Aikar-Flags	3	55	16.81	50	Ja
9	Vanilla, Aikar-Flags	1	55	16.31	50	Ja
10	Fabric + Lithium, Aikar-Flags	3	65	20.00	65	Ja
11	Paper, Standard-G1	3	85	19.56	85	Ja
12	Vanilla, Aikar-Flags	2	55	17.16	50	Ja
13	Fabric + Lithium, Aikar-Flags	1	65	20.01	65	Ja
14	Paper, Standard-G1	1	80	20.01	80	Ja
15	Paper, Generational ZGC	3	70	19.82	70	Ja
16	Purpur, Aikar-Flags	3	80	19.57	80	Ja
17	Paper, Aikar-Flags	3	80	19.93	80	Ja
18	Paper, Generational ZGC	2	70	19.97	70	Ja

Pairwise comparisons

A difference counts as demonstrated if the ranges of the three repetitions do not overlap.

Metric	A	A min	A max	B	B min	B max	Demonstrated
kapazitaet_p95	Vanilla, Aikar-Flags	55.0	55.0	Paper, Aikar-Flags	80.0	85.0	Ja
kapazitaet_tps	Vanilla, Aikar-Flags	50.0	50.0	Paper, Aikar-Flags	80.0	80.0	Ja
p95_bei_50	Vanilla, Aikar-Flags	42.7	43.5	Paper, Aikar-Flags	27.8	29.8	Ja
p99_bei_50	Vanilla, Aikar-Flags	50.2	55.1	Paper, Aikar-Flags	31.2	33.4	Ja
kapazitaet_p95	Vanilla, Aikar-Flags	55.0	55.0	Purpur, Aikar-Flags	75.0	80.0	Ja
kapazitaet_tps	Vanilla, Aikar-Flags	50.0	50.0	Purpur, Aikar-Flags	75.0	80.0	Ja
p95_bei_50	Vanilla, Aikar-Flags	42.7	43.5	Purpur, Aikar-Flags	29.4	31.0	Ja
p99_bei_50	Vanilla, Aikar-Flags	50.2	55.1	Purpur, Aikar-Flags	33.2	35.0	Ja
kapazitaet_p95	Vanilla, Aikar-Flags	55.0	55.0	Fabric + Lithium, Aikar-Flags	65.0	70.0	Ja
kapazitaet_tps	Vanilla, Aikar-Flags	50.0	50.0	Fabric + Lithium, Aikar-Flags	65.0	65.0	Ja
p95_bei_50	Vanilla, Aikar-Flags	42.7	43.5	Fabric + Lithium, Aikar-Flags	36.1	37.3	Ja
p99_bei_50	Vanilla, Aikar-Flags	50.2	55.1	Fabric + Lithium, Aikar-Flags	43.4	45.0	Ja
kapazitaet_p95	Paper, Aikar-Flags	80.0	85.0	Purpur, Aikar-Flags	75.0	80.0	Nein
kapazitaet_tps	Paper, Aikar-Flags	80.0	80.0	Purpur, Aikar-Flags	75.0	80.0	Nein
p95_bei_50	Paper, Aikar-Flags	27.8	29.8	Purpur, Aikar-Flags	29.4	31.0	Nein
p99_bei_50	Paper, Aikar-Flags	31.2	33.4	Purpur, Aikar-Flags	33.2	35.0	Nein
kapazitaet_p95	Paper, Aikar-Flags	80.0	85.0	Fabric + Lithium, Aikar-Flags	65.0	70.0	Ja
kapazitaet_tps	Paper, Aikar-Flags	80.0	80.0	Fabric + Lithium, Aikar-Flags	65.0	65.0	Ja
p95_bei_50	Paper, Aikar-Flags	27.8	29.8	Fabric + Lithium, Aikar-Flags	36.1	37.3	Ja
p99_bei_50	Paper, Aikar-Flags	31.2	33.4	Fabric + Lithium, Aikar-Flags	43.4	45.0	Ja
kapazitaet_p95	Purpur, Aikar-Flags	75.0	80.0	Fabric + Lithium, Aikar-Flags	65.0	70.0	Ja
kapazitaet_tps	Purpur, Aikar-Flags	75.0	80.0	Fabric + Lithium, Aikar-Flags	65.0	65.0	Ja
p95_bei_50	Purpur, Aikar-Flags	29.4	31.0	Fabric + Lithium, Aikar-Flags	36.1	37.3	Ja
p99_bei_50	Purpur, Aikar-Flags	33.2	35.0	Fabric + Lithium, Aikar-Flags	43.4	45.0	Ja
kapazitaet_p95	Paper, Aikar-Flags	80.0	85.0	Paper, Standard-G1	80.0	85.0	Nein
kapazitaet_tps	Paper, Aikar-Flags	80.0	80.0	Paper, Standard-G1	80.0	85.0	Nein
p95_bei_50	Paper, Aikar-Flags	27.8	29.8	Paper, Standard-G1	28.3	29.5	Nein
p99_bei_50	Paper, Aikar-Flags	31.2	33.4	Paper, Standard-G1	34.4	36.1	Ja
kapazitaet_p95	Paper, Aikar-Flags	80.0	85.0	Paper, Generational ZGC	70.0	70.0	Ja

Metric	A	A min	A max	B	B min	B max	Demonstrated
kapazitaet_tps	Paper, Aikar-Flags	80.0	80.0	Paper, Generational ZGC	70.0	70.0	Ja
p95_bei_50	Paper, Aikar-Flags	27.8	29.8	Paper, Generational ZGC	34.1	35.4	Ja
p99_bei_50	Paper, Aikar-Flags	31.2	33.4	Paper, Generational ZGC	39.9	40.3	Ja
kapazitaet_p95	Paper, Standard-G1	80.0	85.0	Paper, Generational ZGC	70.0	70.0	Ja
kapazitaet_tps	Paper, Standard-G1	80.0	85.0	Paper, Generational ZGC	70.0	70.0	Ja
p95_bei_50	Paper, Standard-G1	28.3	29.5	Paper, Generational ZGC	34.1	35.4	Ja
p99_bei_50	Paper, Standard-G1	34.4	36.1	Paper, Generational ZGC	39.9	40.3	Ja

4 Method

1 **Fixing the programs**

All programs ran on Minecraft 26.3. Java, server programs and Lithium were verified against the published checksum. No published checksum exists for the Fabric starter, so its own SHA-256 value is recorded. The server.properties were identical for all programs; the remaining settings stayed at each program's defaults. The only other change concerns Paper and Purpur: the per-IP-address connection throttle was switched off, because all bots come from the same loopback address. Lithium describes itself as an optimization mod that is not meant to change game mechanics (source 4), and Purpur as a drop-in replacement for Paper (source 5).

2 **One shared world**

The unmodified Vanilla server generated a world once with seed 20261121, containing 31,329 fully generated chunks. Each run started from a fresh copy of this world.

3 **Separate cores**

The server was pinned to cores 0 to 3 and the load generator to cores 4 to 6. The measurement script was not pinned to any core. During the measurement, the raw data was packed every 15 minutes at lowest priority on core 7. The entry messung: 7 in plan.json was not implemented as a core pinning. Server and RCON were reachable only on the loopback address, and online-mode=false was set only in this isolated setup. Erik Friedl approved the Minecraft EULA for these test servers.

4 **Protocol bots**

A custom lightweight client speaks the real network protocol of version 26.3: login, configuration, play phase, keep-alive and teleport confirmations. The client sends chunk acknowledgements when it detects a chunk-batch message. This was the case for Vanilla and Fabric; for Paper and Purpur, it counted no such message in any run. The protocol database used by mineflayer did not yet include version 26.3 at the time of measurement. Each bot has a fixed seed, sends at a target rate of 20 position packets per second and moves at walking speed (0.2158 blocks per tick) between random waypoints. Every six bots form a group, and the groups are 256 blocks apart. The bots fly in creative mode at height 120.

5 **Levels and measurement windows**

The levels were 20, 30 and 40 bots, then in steps of five. After joining came a 60-second settling period and a 120-second measurement window with one sample every 5 seconds. A run ended after two levels above the limit or below 15 TPS.

6 **Metrics**

For all programs, MSPT p50, p95 and p99 came from the /tick query command via RCON, which evaluates the last 100 ticks. The level value is the mean of these blocks over the measurement window. TPS was calculated from the progress of game time. In addition came RAM of the Java process, CPU per core group, CPU steal and the number of bots confirmed via the player list.

7 **Preliminary runs before the plan**

Before the plan was fixed, two pilot runs (Paper, Vanilla) and short trial runs of all combinations were carried out. After that, the level grid and measurement times were fixed; the 50 ms limit had been set beforehand. The preliminary runs are not included in the evaluation.

8 **Validity**

Fixed in advance: CPU steal below 2%, load generator below 70% CPU and below 20 ms timer lag (p99), complete samples, all bots connected as long as the server is not overloaded, and two sample bots demonstrably moving within the measurement window. Invalid runs would have been repeated; there were none.

9 **Capacity and differences**

Fixed in advance: the capacity of a run is the highest level up to which all levels stay below 50 ms p95 MSPT on average over the measurement window and all bots are connected. The capacity of a combination is the smallest value of its three runs. A difference counts only if the ranges of the repetitions do not overlap.

10 Supplement added afterwards

After the measurement it became apparent that the tick rate in some cases already fell below 20 TPS while the p95 tick time was still below 50 ms on average. For this reason, the capacity at at least 19.5 TPS and the metrics at 50 bots were additionally evaluated. Level 50 is the highest at which all combinations still reached at least 19.5 TPS on average over the measurement window. These evaluations were not fixed in advance and are labeled accordingly in the data.

Measurement setup

Game version	Minecraft 26.3 (protocol 777)
Vanilla	26.3
Paper	26.3, build 257
Purpur	26.3, build 2660
Fabric	Loader 0.20.1, installer 1.1.2
Lithium	0.26.4 for 26.3
Java	Temurin 25.0.5.1+2
Heap	10 GB (Xms equal to Xmx)
JVM profile default G1	-XX:+UseG1GC
JVM profile Aikar flags	Flags as per the PaperMC documentation
JVM profile Generational ZGC	-XX:+UseZGC
Virtual machine	8 vCPU under KVM, 31.3 GiB RAM, 100 GiB virtual disk, Debian 12
Reported CPU	QEMU Virtual CPU, AuthenticAMD; the physical model cannot be determined from within the VM
Cores	Server 0 to 3, bots 4 to 6, backup 7, measurement script unpinned
View distance	10 chunks, simulation distance 10 chunks
Game mode	Creative, difficulty normal
Measurement period	21.11.2026, 20:12 UTC to 22.11.2026, 06:59 UTC

5 Sample and period

Period	Nov 21, 2026 to Nov 22, 2026
Population of the study	18 Measurement runs
Evaluated	18 Measurement runs

Selection	Six combinations of program and JVM profile with three repetitions each, in a randomized order fixed in advance.

6 Limits of what the study can show

Bots are not players. The bots fly, do not build, do not fight, do not chat, use no inventory and trigger no redstone circuits. Real players generate a different load. The absolute bot numbers can therefore not be transferred to a real server. The world was generated in advance and the bots did not leave this area. The load from generating new chunks is therefore not included.

One VM. Measurements were taken on a single virtual machine with four cores for the server. On other hardware the values will differ, and the gaps between the programs may change as well.

One test profile. There was one world, one seed, one movement profile and, apart from Lithium, no mods or plugins. The programs were not individually tuned. With an adjusted configuration or a different load, the order may look different. The day-night cycle ran along. Because all runs follow the same schedule, certain levels always fall into the night. The influence was not recorded separately in this measurement.

Different default configuration. The default configuration of Paper and Purpur contains the setting entity-activation-range (animals and monsters 32 blocks), which is missing from the Vanilla and Fabric templates. Whether all programs performed the same simulation work was not checked, and entity counts were not recorded.

MSPT and TPS diverge. For Vanilla, the p95 tick time at 55 bots was below 50 ms on average over the measurement window, but the tick rate was 16.31 to 17.16 TPS. The cause was not investigated. For Vanilla, the capacity fixed in advance is therefore 5 bots above the value at at least 19.5 TPS. For the five other combinations, both minimum values agree.

Means instead of individual values. The limit of 50 ms applies to the mean of the 100-tick blocks over the measurement window. In 13 of 18 runs, at least one individual block at the capacity level was at 50 ms or above, at most 54.2 ms. The limit of 19.5 TPS also applies to the mean of the measurement window. In the levels of the capacity at at least 19.5 TPS, individual 5-second samples were below it, the lowest at 17.62 TPS (Vanilla, 50 bots).

Drift over the measurement night. At the same bot count, the p95 tick time rose with position in the run order, at 50 bots by roughly 0.11 ms per run (evaluated afterwards). The cause was not investigated in this measurement. The order was random but not balanced in blocks, so the spread of the repetitions also contains this effect.

Metrics added afterwards. The capacity at at least 19.5 TPS, the comparison level with 50 bots and the p99 comparison between Aikar flags and default G1 were chosen only after seeing the data. They are supplementary observations and not tests fixed in advance.

Coarse grid, three repetitions. In the relevant range the levels are five bots apart, and smaller differences cannot be measured. Three repetitions per combination show the spread only roughly. No statistical test was calculated.

RAM values not comparable. The Aikar flags include AlwaysPreTouch, which reserves memory already at startup (source 1). The measured memory therefore depends on the JVM setting and does not show the program's requirements. This study makes no statement about which program needs less RAM.

JVM comparison on Paper only. The three JVM profiles were measured only with Paper, a 10 GB heap and Java 25. The program comparison ran only with the Aikar flags. Neither result can readily be transferred to other combinations.

Snapshot. The results apply to the stated versions and builds. Later versions may behave differently.

7 Data and verifiability

To repeat the measurement you need a Linux machine with eight cores and about 32 GB RAM, as used here, plus Node.js and Python 3. The scripts expect the working directory /root/studien/haertetest, download Java and the same program versions as long as those downloads remain available, verify the checksums, generate the world and run the measurement plan. Seed, order, levels and rules are in plan.json. Different hardware, different builds or a different load on the host system lead to different absolute values.

File	Size	SHA-256
daten.csv Measurements per level (CSV)	35.6 KB	a2c2bfdfcc8ddfce2f6eac4044c15aeb ed57f4d0fab95b8a48af72c47e2f9e9e
daten.json Measurements per level (JSON)	94.6 KB	f798d400aba35617cf30f8c0cd2ad07c f6aace6e19e02300a0a1b822c6ac980e
laeufe.csv Result per run	2.7 KB	3a7f0af91987e982f8e95f97c60ef833 0ea9ba8a9f4688e76c64f8514b3486d1
zusammenfassung.csv Result per combination	1.1 KB	8ba8269c39e20d767c2a8c87dfb805fd 0d937bb59bd448680f23401f0e66bd88
vergleiche.csv Pairwise comparisons	4.7 KB	99575f03f6a3f2075bcd20426a18e86d 6d14ad5410618ebb101050c3fabcda9d
p99-aikar-g1.csv p99 Aikar flags and default G1	0.5 KB	4d0b6c017068a81642cddc339b1623f2 71d5d543abaf0ea9e4722152907e64ac
proben.csv Individual samples	517.9 KB	7fa08cf87e94c1e58c680ae9cec668ba 6d56da228bda6be08849d8232db27c33

File	Size	SHA-256
plan.json Measurement plan	5.9 KB	cc335b804f52374f6c77afda5c55dd19 dbf5fa03d595c8f3a9bb72cedd569192
messprotokoll.txt Measurement log	8.1 KB	f234fa1d0a16b312777dd0b93fb12f0e b13161fbdbb13125501e034837705a64
herkunft.txt Origin of the programs	1.4 KB	5f9e921b1c4fe697d2e676b7b5777eee ae784a4c4418b211230e7ac333bb676d
liesmich.txt Readme	1.4 KB	8a444fae54933207d6f896459ac23511 c541a656d0fc2da2a26b2f69091f0935
01-einrichten.sh Setup	3.4 KB	27627241f4e7964be26e9b3ac2b3c1de 74482e323348569f6e7ba6e9290b7ea9
02-vorlagen.py World and templates	4.1 KB	fc70d06d68b18d70425e0987794f06da 257659424a168433c2048577258d4cd5
03-plan.py Measurement plan	3.5 KB	773d40d0d4dd1323c748a8c383ecf1ff b0fb91299cbea545f21144dfa083a034
gemeinsam.py Shared components	5.3 KB	da16b51f5b78fd55c070c1eaa9b43485 cd7fd1654ce8c47e71df87b26bcb7e20
messlauf.py Measurement run	10.9 KB	395ec326a334855d03a333879da16f90 0730d0307f06065f078f79af8de21637
hauptlauf.py Main measurement	4.6 KB	7b1602088aba30734e706e35ab894d4c 918a91eec8b0846a68a2ee392c6cec2f
bots.mjs Load generator	14.3 KB	6fa44c58b3c56a31fb77fb19602cc06a ab93f08e8e8984a40bb95f2ad1064a10
welt-pruefen.py World check	1.5 KB	e0bb7dcc1e2327e9b916355b3e3dba55 2751a2f6c13989ffd07f10c234c0940e
auswertung.mjs Evaluation	10.0 KB	42dc3452a2061bad7ede2d84610cdf8c 8ef3adb5e7c6c7b8593c502aa2b85a33
baue-studie.mjs Study build	39.2 KB	c0e76feab583afa11baf3bbce105203f 4a3ebcf57e66adf44380bb7854a8b90b

Fields of the data files

Field	Unit	Meaning
lauf		Identifier of the measurement run with position in the random order
kombination		Identifier made up of program and JVM profile
anzeige		Readable name of the combination
programm		Server program
jvm		JVM profile

Field	Unit	Meaning
wiederholung		Number of the repetition, 1 to 3
versuch		Attempt within the repetition, 1 is the first
reihenfolge		Position in the fixed random order
gueltig		Run meets all validity rules fixed in advance
lauf_gueltig		The associated run is valid
spieler	Bots	Target number of bots in this level
spieler_min	Bots	Smallest bot count confirmed via the player list in the measurement window
spieler_verbunden	Bots	Bot count confirmed via the player list in this sample
unter_grenze		p95 MSPT below 50 ms on average over the measurement window and all bots connected
p50_ms	ms	Median of the tick time; in levels, mean of the 100-tick blocks
p95_ms	ms	95th percentile of the tick time; in levels, mean of the 100-tick blocks
p95_max_ms	ms	Highest p95 value of a 100-tick block in the measurement window
p99_ms	ms	99th percentile of the tick time; in levels, mean of the 100-tick blocks
mspt_mittel_ms	ms	Mean tick time according to /tick query
tps	Ticks/s	Actual tick rate from the progress of game time; in levels, mean of the measurement window
tps_min	Ticks/s	Lowest tick rate of a 5-second sample in the measurement window
ram_gb	GiB	Memory used by the Java process (RSS)
cpu_server_pct	%	Mean utilization of the four server cores
cpu_bots_pct	%	Mean utilization of the three cores of the load generator
steal_pct	%	CPU steal across all eight cores
bots_verzug_p99_ms	ms	Highest p99 lag of the timers in the load generator
getrennt	Bots	Bots unexpectedly disconnected up to this level
proben		Number of 5-second samples in the measurement window

Field	Unit	Meaning
zeit_unix	s	Time of the sample as Unix time
erste_ueberlast	Bots	First level whose mean p95 MSPT is above the limit of 50 ms
kapazitaet	Bots	Capacity fixed in advance; in the summary, smallest value of the three runs
tps_bei_kapazitaet	Ticks/s	Mean tick rate at the capacity level
kapazitaet_tps	Bots	Highest level up to which all levels held at least 19.5 TPS on average over the measurement window; in the summary, smallest value
kapazitaet_tps_190	Bots	Like kapazitaet_tps, with 19.0 TPS as the limit
kapazitaet_tps_198	Bots	Like kapazitaet_tps, with 19.8 TPS as the limit
p95_bei_50_ms	ms	p95 MSPT at the level with 50 bots
p99_bei_50_ms	ms	p99 MSPT at the level with 50 bots
tps_bei_50	Ticks/s	Tick rate at the level with 50 bots
stufen_gemessen		Number of measured levels of the run
ungueltig_grund		Reason for invalid runs, otherwise empty
im_programmvergleich		Combination belongs to the program comparison
im_jvm_vergleich		Combination belongs to the comparison of the JVM profiles
laeufe_gueltig		Valid runs of the combination
laeufe_ungueltig		Invalid runs of the combination
kapazitaet_median	Bots	Median of the capacity across the runs
kapazitaet_max	Bots	Largest capacity value of the runs
kapazitaet_tps_median	Bots	Median of kapazitaet_tps
kapazitaet_tps_max	Bots	Largest value of kapazitaet_tps
tps_bei_kapazitaet_min	Ticks/s	Lowest tick rate at the capacity across the runs
p95_bei_50_min_ms	ms	Smallest p95 value at 50 bots
p95_bei_50_max_ms	ms	Largest p95 value at 50 bots
p99_bei_50_min_ms	ms	Smallest p99 value at 50 bots
p99_bei_50_max_ms	ms	Largest p99 value at 50 bots
ram_bei_50_gb	GiB	Mean RSS at 50 bots
cpu_bei_50_pct	%	Mean utilization of the server cores at 50 bots

Field	Unit	Meaning
vergleich		Program comparison or JVM comparison
kennzahl		Compared metric
regel		Whether the metric was fixed in advance or added afterwards
a		First combination
a_min		Smallest value of the first combination
a_max		Largest value of the first combination
b		Second combination
b_min		Smallest value of the second combination
b_max		Largest value of the second combination
unterschied_belegt		The ranges of the repetitions do not overlap
besser		Combination with the more favorable value, only for a demonstrated difference
aikar_min	ms	Smallest p99 value with Aikar flags at the level
aikar_max	ms	Largest p99 value with Aikar flags at the level
g1_min	ms	Smallest p99 value with default G1 at the level
g1_max	ms	Largest p99 value with default G1 at the level
aikar_niedriger		All three Aikar values are below all three G1 values
g1_niedriger		All three G1 values are below all three Aikar values

8 Sources

[1] PaperMC (2026): Aikar's flags.
https://docs.papermc.io/paper/aikars-flags/
Retrieved on Nov 21, 2026

[2] Minecraft Wiki (2026): Commands/tick.
https://minecraft.wiki/w/Commands/tick
Retrieved on Nov 21, 2026

[3] OpenJDK: JEP 490: ZGC: Remove the Non-Generational Mode.
https://openjdk.org/jeps/490
Retrieved on Nov 21, 2026

[4] Lithium-Projekt auf Modrinth (2026): Lithium.
https://modrinth.com/mod/lithium
Retrieved on Nov 21, 2026

[5] PurpurMC: Purpur Documentation.
https://purpurmc.org/docs/purpur/
Retrieved on Nov 21, 2026

9 Disclosure

Own interest

Nexthosting sells game servers itself, including servers for Minecraft, and therefore has a commercial interest in this topic. None of the tested programs comes from Nexthosting. The measurement ran on a test VM ordered by Nexthosting. The study does not rate any hosting providers' offers. Nexthosting is responsible for this study, contact: support@nexthosting.net.

Authorship

Erik Friedl, Nexthosting. The publisher is Nexthosting.

License

CC BY 4.0: https://creativecommons.org/licenses/by/4.0/deed.en

10 How to cite this study

APA

Friedl, E. & Nexthosting. (2026). Server stress test: How many protocol bots can Vanilla, Paper, Purpur and Fabric with Lithium handle? (Version 1.0.0) [Report]. Nexthosting. https://nexthosting.net/en/studies/server-haertetest-minecraft

BibTeX

```
@techreport{nexthosting2026serverhaertetestminecraft,
  author     = {Erik Friedl and {Nexthosting}},
  title      = {Server stress test: How many protocol bots can Vanilla, Paper, Purpur and Fabric with Lithium handle?},
  institution= {Nexthosting},
  type       = {Study},
  year       = {2026},
  version    = {1.0.0},
  url        = {https://nexthosting.net/en/studies/server-haertetest-minecraft},
  urldate    = {2026-11-22},
  language   = {english}
}
```

11 Changes and corrections

Version	Date	Type	Change
1.0.0	Nov 22, 2026	Version	First version.