Nexthosting

Studie · Gameserver

Server-Härtetest: Wie viele Protokoll-Bots tragen Vanilla, Paper, Purpur und Fabric mit Lithium?

Vier Server-Programme für Minecraft und drei JVM-Profile, gemessen mit Protokoll-Bots auf einer Test-VM

KEINE OFFIZIELLE MINECRAFT-VERÖFFENTLICHUNG. NICHT GENEHMIGT VON ODER VERBUNDEN MIT MOJANG ODER MICROSOFT.

Art der Studie	Eigene Messung
Herausgeber	Nexthosting
Autorenschaft	Erik Friedl (ORCID: https://orcid.org/0009-0008-2784-4314), Nexthosting
Version	1.0.0 vom 22.11.2026
Zeitraum	21.11.2026 bis 22.11.2026
Stichprobe	18 von 18 Messläufe ausgewertet
Lizenz	CC BY 4.0
Fassung online	nexthosting.net/studien/server-haertetest-minecraft

Inhalt

1	Kernergebnisse	3
2	Zusammenfassung	4
3	Ergebnisse	4
4	Methode	10
5	Stichprobe und Zeitraum	12
6	Grenzen der Aussagekraft	13
7	Daten und Nachprüfbarkeit	14
8	Quellen	18
9	Offenlegung	19
10	So zitierst du diese Studie	19
11	Änderungen und Korrekturen	20

1 Kernergebnisse

80 Bots

Paper mit Aikar-Flags

Paper blieb im Mittel des Messfensters in allen drei Wiederholungen bis 80 Bots unter der vorab festgelegten Grenze von 50 ms p95-MSPT, in einer davon bis 85. Mit im Mittel mindestens 19,5 TPS als strengerer Grenze waren es in allen drei Wiederholungen 80.

Beleg: laeufe.csv, Feld kapazitaet · laeufe.csv, Feld kapazitaet_tps · Quelle [1]

75 Bots

Purpur, bei der Kapazität von Paper nicht zu trennen

Purpur erreichte 75 bis 80 Bots. Die Spanne überlappt mit der von Paper (80 bis 85), deshalb weist diese Messung keinen Unterschied zwischen beiden aus.

Beleg: laeufe.csv, Feld kapazitaet

65 Bots

Fabric mit Lithium

Fabric mit Lithium erreichte 65 bis 70 Bots. Das liegt über Vanilla und unter Paper und Purpur, die Spannen überlappen in beiden Richtungen nicht.

Beleg: laeufe.csv, Feld kapazitaet

55 Bots

Vanilla

Vanilla blieb im Mittel des Messfensters in allen drei Wiederholungen bis 55 Bots unter 50 ms p95-MSPT. Mit im Mittel des Messfensters mindestens 19,5 TPS waren es nur 50.

Beleg: laeufe.csv, Feld kapazitaet · laeufe.csv, Feld kapazitaet_tps

16,31 TPS

Die MSPT-Grenze allein reicht nicht

Bei 55 Bots lag Vanilla im Mittel des Messfensters unter 50 ms p95-MSPT, lief aber nur mit 16,31 bis 17,16 statt 20 Ticks je Sekunde. Darum nennt diese Studie neben der vorab festgelegten Kapazität auch die Kapazität bei mindestens 19,5 TPS.

Beleg: daten.csv, Feld tps · Quelle [2]

80 Bots

Standard-G1 und Aikar-Flags: kein belegter Kapazitätsunterschied

Paper mit Standard-G1 erreichte 80 bis 85 Bots, mit Aikar-Flags ebenfalls 80 bis 85. Bei der Kapazität gibt es keinen belegten Unterschied. Ergänzend beobachtet: Beim p99-Wert lagen die Aikar-Flags in 7 von 13 gemeinsamen Stufen getrennt niedriger, Standard-G1 in keiner. Die Aikar-Läufe lagen im Mittel früher in der Reihenfolge, ein Teil des Abstands kann daher aus der Drift über die Messnacht stammen (siehe Grenzen).

Beleg: laeufe.csv, Feld kapazitaet · p99-aikar-g1.csv, Feld spieler · Quelle [1]

70 Bots	**Generational ZGC lag niedriger** Paper mit Generational ZGC erreichte in allen drei Wiederholungen 70 Bots und damit weniger als mit Standard-G1 oder Aikar-Flags (je 80 bis 85). Das gilt für dieses Lastprofil, diesen Heap und diese Java-Version. Beleg: laeufe.csv, Feld kapazitaet · Quelle [3]
18 von 18 Läufen	**Alle Läufe gültig** Alle 18 Läufe erfüllten die vorab festgelegten Regeln. Im Mittel einer Stufe erreichte der CPU-Steal höchstens 0,061 %, die Kerne des Lastgenerators höchstens 5,72 %. Beleg: laeufe.csv, Feld lauf · daten.csv, Feld steal_pct · daten.csv, Feld cpu_bots_pct

2 Zusammenfassung

> **Fragestellung:** Wie viele gleichzeitige Protokoll-Bots schaffen Vanilla, Paper, Purpur und Fabric mit Lithium bei gleicher Hardware, und wie stark beeinflussen JVM-Einstellungen das Ergebnis?

Wir haben Vanilla, Paper, Purpur und Fabric mit Lithium in Minecraft 26.3 auf derselben virtuellen Maschine belastet. Protokoll-Bots traten stufenweise bei, bis der Server überlastet war. Alle 18 Läufe waren nach den vorab festgelegten Regeln gültig.

Als Kapazität zählt die höchste Botzahl, bis zu der die p95-Tickzeit im Mittel des Messfensters unter 50 ms blieb, und zwar in allen drei Wiederholungen. Paper erreichte 80, Purpur 75, Fabric mit Lithium 65 und Vanilla 55 Bots. Paper und Purpur lassen sich bei der vorab festgelegten Kapazität nicht voneinander trennen, bei allen anderen Programmpaaren überlappen sich die Spannen der Wiederholungen nicht.

Bei den JVM-Einstellungen zeigte sich an Paper kein belegter Kapazitätsunterschied zwischen Standard-G1 und den Aikar-Flags. Generational ZGC lag mit 70 Bots darunter.

Die Tickzeit allein zeigt nicht jede Überlast: Vanilla blieb bei 55 Bots im Mittel unter 50 ms p95, erreichte aber nur 16,31 bis 17,16 statt 20 Ticks je Sekunde. Deshalb steht neben jeder Kapazität auch der Wert bei im Mittel mindestens 19,5 TPS. Die Zahlen gelten für diese VM und dieses Bot-Profil und lassen sich nicht direkt auf echte Spieler übertragen.

3 Ergebnisse

So liest man die Ergebnisse

Die vorab festgelegte Kapazität beruht auf der p95-Tickzeit. Die zweite Kapazität verlangt stattdessen, dass der Server im Mittel des Messfensters mindestens 19,5 von 20 Ticks je Sekunde

schafft und alle Bots verbunden bleiben. Beide nennen den kleinsten Wert aus drei Wiederholungen. Ein Unterschied zählt nur, wenn sich die Spannen der Wiederholungen nicht überlappen.

Kapazität der Programme

Höchste Botzahl mit im Mittel des Messfensters unter 50 ms p95-MSPT in allen drei Wiederholungen, alle mit Aikar-Flags.



Kapazität der Programme bei mindestens 19,5 TPS

Nachträglich ergänzt: höchste Botzahl mit im Mittel mindestens 19,5 TPS in allen drei Wiederholungen.



Kapazität von Paper je JVM-Profil

Höchste Botzahl mit im Mittel des Messfensters unter 50 ms p95-MSPT in allen drei Wiederholungen.



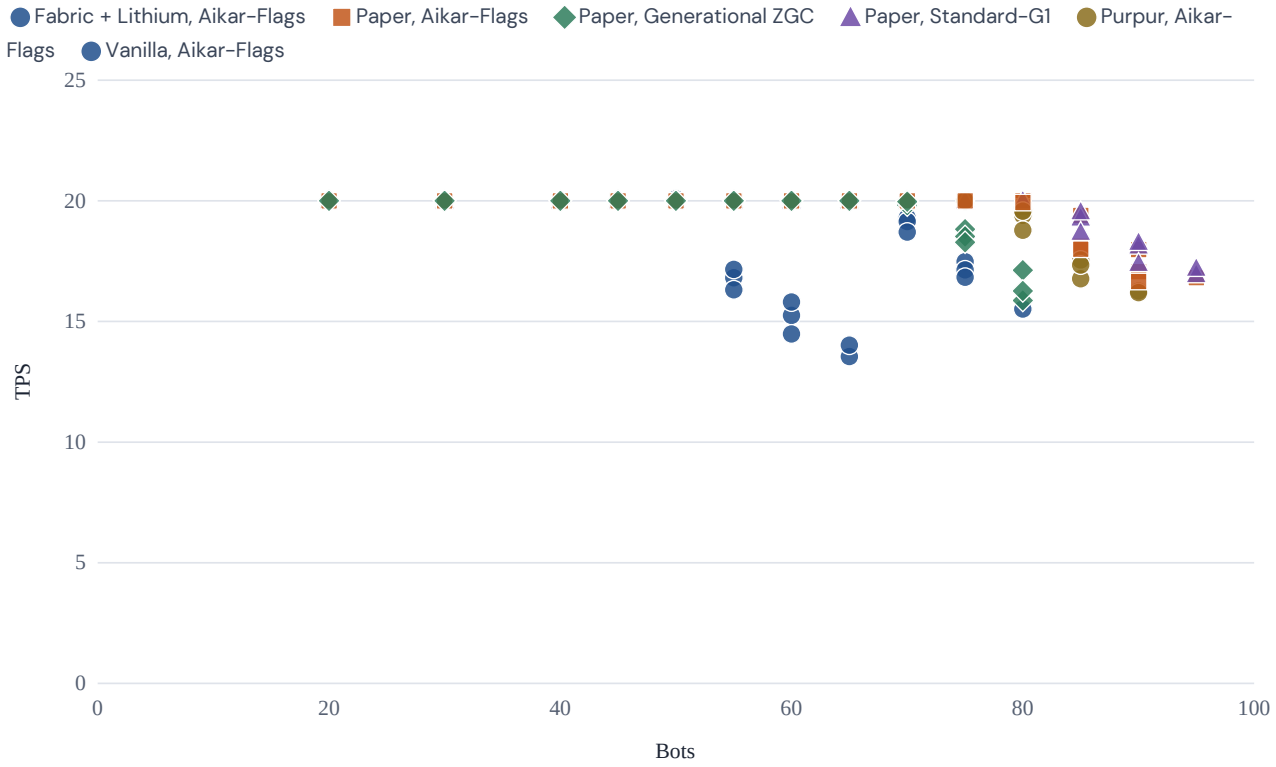
p95-Tickzeit je Stufe

Jeder Punkt ist eine Stufe eines Laufs, dargestellt als Mittelwert des Messfensters. Die Grenze von 50 ms gilt für diesen Mittelwert.



Tatsächliche Taktrate je Stufe

Jeder Punkt ist eine Stufe eines Laufs. Das Ziel sind 20 Ticks je Sekunde.



Alle Kombinationen

Kombination	Kapazität	Höchster Lauf	Bei 19,5 TPS	Höchster Lauf (TPS)	p95 bei 50, min	p95 bei 50, max	p99 bei 50, min	p99 bei 50, ma
Paper, Aikar-Flags	80	85	80	80	27,8 ms	29,8 ms	31,2 ms	33,4 m
Paper, Standard-G1	80	85	80	85	28,3 ms	29,5 ms	34,4 ms	36,1 m
Purpur, Aikar-Flags	75	80	75	80	29,4 ms	31,0 ms	33,2 ms	35,0 m
Paper, Generational ZGC	70	70	70	70	34,1 ms	35,4 ms	39,9 ms	40,3 m
Fabric + Lithium, Aikar-Flags	65	70	65	65	36,1 ms	37,3 ms	43,4 ms	45,0 m
Vanilla, Aikar-Flags	55	55	50	50	42,7 ms	43,5 ms	50,2 ms	55,1 m

Alle 18 Läufe

Nr.	Kombination	Wdh.	Kapazität	TPS dort	Bei 19,5 TPS	Gültig
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

Paarvergleiche

Ein Unterschied gilt als belegt, wenn sich die Spannen der drei Wiederholungen nicht überlappen.

Kennzahl	A	A min	A max	B	B min	B max	Belegt
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

Kennzahl	A	A min	A max	B	B min	B max	Belegt
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 Methode

1 **Programme festhalten**

Alle Programme liefen auf Minecraft 26.3. Java, Server-Programme und Lithium wurden mit der veröffentlichten Prüfsumme geprüft. Für den Fabric-Starter gibt es keine veröffentlichte Prüfsumme, dort ist der eigene SHA-256-Wert festgehalten. Die server.properties waren für alle Programme gleich, die übrigen Einstellungen blieben auf den Standardwerten des jeweiligen Programms. Die einzige weitere Änderung betrifft Paper und Purpur: Die Verbindungsdrossel je IP-Adresse wurde abgeschaltet, weil alle Bots von derselben Loopback-Adresse kommen. Lithium beschreibt sich als Optimierungs-Mod, der die Spielmechanik nicht verändern soll (Quelle 4), Purpur als direkten Ersatz für Paper (Quelle 5).

2 **Eine gemeinsame Welt**

Der unveränderte Vanilla-Server erzeugte mit Seed 20261121 einmal eine Welt mit 31.329 vollständig generierten Chunks. Jeder Lauf startete von einer frischen Kopie dieser Welt.

3 **Getrennte Kerne**

Der Server lief fest auf den Kernen 0 bis 3, der Lastgenerator auf den Kernen 4 bis 6. Das Messskript war nicht an einen Kern gebunden. Während der Messung wurden die Rohdaten alle 15 Minuten mit niedrigster Priorität auf Kern 7 gepackt. Der Eintrag messung: 7 in plan.json wurde nicht als Kernbindung umgesetzt. Server und RCON waren nur auf der Loopback-Adresse erreichbar, online-mode=false war nur in diesem isolierten Aufbau gesetzt. Die Minecraft-EULA hat Erik Friedl für diese Testserver freigegeben.

4 **Protokoll-Bots**

Ein eigener schlanker Client spricht das echte Netzwerkprotokoll von Version 26.3: Anmeldung, Konfiguration, Spielphase, Keep-Alive und Teleport-Bestätigungen. Chunk-Bestätigungen sendet der Client, wenn er eine Chunk-Batch-Meldung erkennt. Das war bei Vanilla und Fabric der Fall, bei Paper und Purpur zählte er in keinem Lauf eine solche Meldung. Die von mineflayer genutzte Protokolldatenbank führte Version 26.3 zum Messzeitpunkt noch nicht. Jeder Bot hat einen festen Seed, sendet mit einem Soll-Takt von 20 Positionspaketen je Sekunde und bewegt sich in Gehgeschwindigkeit (0,2158 Blöcke je Takt) zwischen zufälligen Wegpunkten. Je sechs Bots bilden eine Gruppe, die Gruppen liegen 256 Blöcke auseinander. Die Bots fliegen im Kreativmodus auf Höhe 120.

5 **Stufen und Messfenster**

Die Stufen waren 20, 30 und 40 Bots, danach in Fünferschritten. Nach dem Beitritt folgten 60 Sekunden Einschwingzeit und 120 Sekunden Messfenster mit einer Probe alle 5 Sekunden. Ein Lauf endete nach zwei Stufen über der Grenze oder unter 15 TPS.

6 **Messgrößen**

MSPT p50, p95 und p99 kamen bei allen Programmen aus dem Befehl /tick query über RCON, der die letzten 100 Ticks auswertet. Der Stufenwert ist der Mittelwert dieser Blöcke im Messfenster. Die TPS wurden aus dem Fortschritt der Spielzeit berechnet. Dazu kamen RAM des Java-Prozesses, CPU je Kerngruppe, CPU-Steal und die per Spielerliste bestätigte Botzahl.

7 **Vorläufe vor dem Plan**

Vor der Festlegung des Plans liefen zwei Pilotläufe (Paper, Vanilla) und kurze Probeläufe aller Kombinationen. Danach wurden Stufenraster und Messzeiten festgelegt, die Grenze von 50 ms stand schon vorher. Die Vorläufe fließen nicht in die Auswertung ein.

8 **Gültigkeit**

Vorab festgelegt: CPU-Steal unter 2 %, Lastgenerator unter 70 % CPU und unter 20 ms Zeitgeber-Verzug (p99), vollständige Proben, alle Bots verbunden, solange der Server nicht überlastet ist, und zwei Stichproben-Bots bewegen sich im Messfenster nachweisbar. Ungültige Läufe wären wiederholt worden, es gab keine.

9 **Kapazität und Unterschiede**

Vorab festgelegt: Die Kapazität eines Laufs ist die höchste Stufe, bis zu der alle Stufen im Mittel des Messfensters unter 50 ms p95-MSPT bleiben und alle Bots verbunden sind. Die Kapazität einer Kombination ist der kleinste Wert ihrer drei Läufe. Ein Unterschied gilt nur, wenn sich die Spannen der Wiederholungen nicht überlappen.

10 Nachträgliche Ergänzung

Nach der Messung zeigte sich, dass die Taktrate teils schon unter 20 TPS fiel, während die p95-Tickzeit im Mittel noch unter 50 ms lag. Darum wurden zusätzlich die Kapazität bei mindestens 19,5 TPS und die Kennzahlen bei 50 Bots ausgewertet. Die Stufe 50 ist die höchste, in der alle Kombinationen im Mittel des Messfensters noch mindestens 19,5 TPS erreichten. Diese Auswertungen waren nicht vorab festgelegt und sind in den Daten so gekennzeichnet.

Messaufbau

Spielversion	Minecraft 26.3 (Protokoll 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 für 26.3
Java	Temurin 25.0.5.1+2
Heap	10 GB (Xms gleich Xmx)
JVM-Profil Standard-G1	-XX:+UseG1GC
JVM-Profil Aikar-Flags	Flags laut PaperMC-Dokumentation
JVM-Profil Generational ZGC	-XX:+UseZGC
Virtuelle Maschine	8 vCPU unter KVM, 31,3 GiB RAM, 100 GiB virtuelle Festplatte, Debian 12
Gemeldete CPU	QEMU Virtual CPU, AuthenticAMD; das physische Modell ist aus der VM nicht feststellbar
Kerne	Server 0 bis 3, Bots 4 bis 6, Sicherung 7, Messskript ungebunden
Sichtweite	10 Chunks, Simulationsweite 10 Chunks
Spielmodus	Kreativ, Schwierigkeit normal
Messzeitraum	21.11.2026, 20:12 UTC bis 22.11.2026, 06:59 UTC

5 Stichprobe und Zeitraum

Zeitraum	21.11.2026 bis 22.11.2026
Grundgesamtheit der Studie	18 Messläufe
Ausgewertet	18 Messläufe
Auswahl	Sechs Kombinationen aus Programm und JVM-Profil mit je drei Wiederholungen, in einer vorab festgelegten zufälligen Reihenfolge.

6 Grenzen der Aussagekraft

Bots sind keine Spieler. Die Bots fliegen, bauen nicht, kämpfen nicht, chatten nicht, nutzen kein Inventar und lösen keine Redstone-Schaltungen aus. Echte Spieler erzeugen eine andere Last. Die absoluten Botzahlen lassen sich deshalb nicht auf einen echten Server übertragen. Die Welt war vorab erzeugt und die Bots verließen diesen Bereich nicht. Die Last durch das Erzeugen neuer Chunks ist deshalb nicht enthalten.

Eine VM. Gemessen wurde auf einer einzigen virtuellen Maschine mit vier Kernen für den Server. Auf anderer Hardware fallen die Werte anders aus, auch die Abstände zwischen den Programmen können sich ändern.

Ein Testprofil. Es gab eine Welt, einen Seed, ein Bewegungsprofil und außer Lithium keine Mods oder Plugins. Die Programme wurden nicht einzeln optimiert. Mit angepasster Konfiguration oder anderer Last kann die Reihenfolge anders aussehen. Der Tag-Nacht-Wechsel lief mit. Weil alle Läufe demselben Zeitplan folgen, fallen bestimmte Stufen immer in die Nacht. Der Einfluss wurde in dieser Messung nicht getrennt erfasst.

Unterschiedliche Standardkonfiguration. Die Standardkonfiguration von Paper und Purpur enthält die Einstellung entity-activation-range (Tiere und Monster 32 Blöcke), die in den Vorlagen von Vanilla und Fabric fehlt. Ob alle Programme dieselbe Simulationsarbeit gerechnet haben, wurde nicht geprüft, Entitätenzahlen wurden nicht erfasst.

MSPT und TPS weichen ab. Bei Vanilla lag die p95-Tickzeit bei 55 Bots im Mittel des Messfensters unter 50 ms, die Taktrate aber bei 16,31 bis 17,16 TPS. Die Ursache wurde nicht untersucht. Bei Vanilla liegt die vorab festgelegte Kapazität deshalb 5 Bots über dem Wert bei mindestens 19,5 TPS. Bei den fünf anderen Kombinationen stimmen beide Kleinstwerte überein.

Mittelwerte statt Einzelwerte. Die Grenze von 50 ms gilt für den Mittelwert der 100-Tick-Blöcke im Messfenster. In 13 von 18 Läufen lag in der Kapazitätsstufe mindestens ein einzelner Block bei 50 ms oder darüber, höchstens bei 54,2 ms. Auch die Grenze von 19,5 TPS gilt für den Mittelwert des Messfensters. In den Stufen der Kapazität bei mindestens 19,5 TPS lagen einzelne 5-Sekunden-Proben darunter, am niedrigsten bei 17,62 TPS (Vanilla, 50 Bots).

Drift über die Messnacht. Bei gleicher Botzahl stieg die p95-Tickzeit mit der Position in der Laufreihenfolge, bei 50 Bots um rund 0,11 ms je Lauf (nachträglich ausgewertet). Die Ursache wurde in dieser Messung nicht untersucht. Die Reihenfolge war zufällig, aber nicht in Blöcken ausgeglichen, deshalb enthält die Streuung der Wiederholungen auch diesen Effekt.

Nachträgliche Kennzahlen. Die Kapazität bei mindestens 19,5 TPS, die Vergleichsstufe mit 50 Bots und der p99-Vergleich zwischen Aikar-Flags und Standard-G1 wurden erst nach Sicht auf die Daten gewählt. Sie sind ergänzende Beobachtungen und keine vorab festgelegten Prüfungen.

Grobes Raster, drei Wiederholungen. Die Stufen liegen im relevanten Bereich fünf Bots auseinander, kleinere Unterschiede sind nicht messbar. Drei Wiederholungen je Kombination zeigen die Streuung nur grob. Ein statistischer Test wurde nicht gerechnet.

RAM-Werte nicht vergleichbar. Die Aikar-Flags enthalten AlwaysPreTouch, das den Speicher schon beim Start reserviert (Quelle 1). Der gemessene Arbeitsspeicher hängt deshalb an der JVM-Einstellung und zeigt nicht den Bedarf des Programms. Diese Studie trifft keine Aussage darüber, welches Programm weniger RAM braucht.

JVM-Vergleich nur an Paper. Die drei JVM-Profile wurden nur mit Paper, einem Heap von 10 GB und Java 25 gemessen. Der Programmvergleich lief nur mit Aikar-Flags. Beide Ergebnisse lassen sich nicht ohne Weiteres auf andere Kombinationen übertragen.

Momentaufnahme. Die Ergebnisse gelten für die genannten Versionen und Builds. Spätere Versionen können sich anders verhalten.

7 Daten und Nachprüfbarkeit

Für eine Wiederholung braucht man eine Linux-Maschine mit acht Kernen und rund 32 GB RAM wie hier, Node.js und Python 3. Die Skripte erwarten den Arbeitsordner /root/studien/haertetest, laden Java und dieselben Programmversionen, solange diese Downloads verfügbar sind, prüfen die Prüfsummen, erzeugen die Welt und führen den Messplan aus. Seed, Reihenfolge, Stufen und Regeln stehen in plan.json. Andere Hardware, andere Builds oder eine andere Auslastung des Wirtssystems führen zu anderen absoluten Werten.

Datei	Größe	SHA-256
daten.csv Messwerte je Stufe (CSV)	35,6 KB	a2c2bfdfcc8ddfce2f6eac4044c15aeb ed57f4d0fab95b8a48af72c47e2f9e9e
daten.json Messwerte je Stufe (JSON)	94,6 KB	f798d400aba35617cf30f8c0cd2ad07c f6aace6e19e02300a0a1b822c6ac980e
laeufe.csv Ergebnis je Lauf	2,7 KB	3a7f0af91987e982f8e95f97c60ef833 0ea9ba8a9f4688e76c64f8514b3486d1
zusammenfassung.csv Ergebnis je Kombination	1,1 KB	8ba8269c39e20d767c2a8c87dfb805fd 0d937bb59bd448680f23401f0e66bd88
vergleiche.csv Paarvergleiche	4,7 KB	99575f03f6a3f2075bcd20426a18e86d 6d14ad5410618ebb101050c3fabcda9d
p99-aikar-g1.csv p99 Aikar-Flags und Standard-G1	0,5 KB	4d0b6c017068a81642cddc339b1623f2 71d5d543abaf0ea9e4722152907e64ac
proben.csv Einzelproben	517,9 KB	7fa08cf87e94c1e58c680ae9cec668ba 6d56da228bda6be08849d8232db27c33

Datei	Größe	SHA-256
plan.json Messplan	5,9 KB	cc335b804f52374f6c77afda5c55dd19 dbf5fa03d595c8f3a9bb72cedd569192
messprotokoll.txt Messprotokoll	8,1 KB	f234fa1d0a16b312777dd0b93fb12f0e b13161fbdbb13125501e034837705a64
herkunft.txt Herkunft der Programme	1,4 KB	5f9e921b1c4fe697d2e676b7b5777eee ae784a4c4418b211230e7ac333bb676d
liesmich.txt Liesmich	1,4 KB	8a444fae54933207d6f896459ac23511 c541a656d0fc2da2a26b2f69091f0935
01-einrichten.sh Einrichtung	3,4 KB	27627241f4e7964be26e9b3ac2b3c1de 74482e323348569f6e7ba6e9290b7ea9
02-vorlagen.py Welt und Vorlagen	4,1 KB	fc70d06d68b18d70425e0987794f06da 257659424a168433c2048577258d4cd5
03-plan.py Messplan	3,5 KB	773d40d0d4dd1323c748a8c383ecf1ff b0fb91299cbea545f21144dfa083a034
gemeinsam.py Gemeinsame Teile	5,3 KB	da16b51f5b78fd55c070c1eaa9b43485 cd7fd1654ce8c47e71df87b26bcb7e20
messlauf.py Messlauf	10,9 KB	395ec326a334855d03a333879da16f90 0730d0307f06065f078f79af8de21637
hauptlauf.py Hauptmessung	4,6 KB	7b1602088aba30734e706e35ab894d4c 918a91eec8b0846a68a2ee392c6cec2f
bots.mjs Lastgenerator	14,3 KB	6fa44c58b3c56a31fb77fb19602cc06a ab93f08e8e8984a40bb95f2ad1064a10
welt-pruefen.py Weltprüfung	1,5 KB	e0bb7dcc1e2327e9b916355b3e3dba55 2751a2f6c13989ffd07f10c234c0940e
auswertung.mjs Auswertung	10,0 KB	42dc3452a2061bad7ede2d84610cdf8c 8ef3adb5e7c6c7b8593c502aa2b85a33
baue-studie.mjs Studienbau	39,2 KB	c0e76feab583afa11baf3bbce105203f 4a3ebcf57e66adf44380bb7854a8b90b

Felder der Datendateien

Feld	Einheit	Bedeutung
lauf		Kennung des Messlaufs mit Position in der zufälligen Reihenfolge
kombination		Kennung aus Programm und JVM-Profil
anzeige		Lesbarer Name der Kombination
programm		Server-Programm
jvm		JVM-Profil

Feld	Einheit	Bedeutung
wiederholung		Nummer der Wiederholung, 1 bis 3
versuch		Versuch innerhalb der Wiederholung, 1 ist der erste
reihenfolge		Position in der festgelegten zufälligen Reihenfolge
gueltig		Lauf erfüllt alle vorab festgelegten Gültigkeitsregeln
lauf_gueltig		Der zugehörige Lauf ist gültig
spieler	Bots	Zielzahl der Bots in dieser Stufe
spieler_min	Bots	Kleinste im Messfenster per Spielerliste bestätigte Botzahl
spieler_verbunden	Bots	Per Spielerliste bestätigte Botzahl in dieser Probe
unter_grenze		p95-MSPT im Mittel des Messfensters unter 50 ms und alle Bots verbunden
p50_ms	ms	Median der Tickzeit; in Stufen Mittelwert der 100-Tick-Blöcke
p95_ms	ms	95. Perzentil der Tickzeit; in Stufen Mittelwert der 100-Tick-Blöcke
p95_max_ms	ms	Höchster p95-Wert eines 100-Tick-Blocks im Messfenster
p99_ms	ms	99. Perzentil der Tickzeit; in Stufen Mittelwert der 100-Tick-Blöcke
mspt_mittel_ms	ms	Mittlere Tickzeit laut /tick query
tps	Ticks/s	Tatsächliche Taktrate aus dem Fortschritt der Spielzeit; in Stufen Mittelwert des Messfensters
tps_min	Ticks/s	Niedrigste Taktrate einer 5-Sekunden-Probe im Messfenster
ram_gb	GiB	Belegter Arbeitsspeicher des Java-Prozesses (RSS)
cpu_server_pct	%	Mittlere Auslastung der vier Server-Kerne
cpu_bots_pct	%	Mittlere Auslastung der drei Kerne des Lastgenerators
steal_pct	%	CPU-Steal über alle acht Kerne
bots_verzug_p99_ms	ms	Höchster p99-Verzug der Zeitgeber im Lastgenerator
getrennt	Bots	Bis zu dieser Stufe unerwartet getrennte Bots

Feld	Einheit	Bedeutung
proben		Zahl der 5-Sekunden-Proben im Messfenster
zeit_unix	s	Zeitpunkt der Probe als Unix-Zeit
erste_ueberlast	Bots	Erste Stufe, deren mittlere p95-MSPT über der Grenze von 50 ms liegt
kapazitaet	Bots	Vorab festgelegte Kapazität; in der Zusammenfassung kleinster Wert der drei Läufe
tps_bei_kapazitaet	Ticks/s	Mittlere Taktrate in der Stufe der Kapazität
kapazitaet_tps	Bots	Höchste Stufe, bis zu der alle Stufen im Mittel des Messfensters mindestens 19,5 TPS hielten; in der Zusammenfassung kleinster Wert
kapazitaet_tps_190	Bots	Wie kapazitaet_tps, mit 19,0 TPS als Grenze
kapazitaet_tps_198	Bots	Wie kapazitaet_tps, mit 19,8 TPS als Grenze
p95_bei_50_ms	ms	p95-MSPT in der Stufe mit 50 Bots
p99_bei_50_ms	ms	p99-MSPT in der Stufe mit 50 Bots
tps_bei_50	Ticks/s	Taktrate in der Stufe mit 50 Bots
stufen_gemessen		Zahl der gemessenen Stufen des Laufs
ungueltig_grund		Grund bei ungültigen Läufen, sonst leer
im_programmvergleich		Kombination gehört zum Vergleich der Programme
im_jvm_vergleich		Kombination gehört zum Vergleich der JVM-Profile
laeufe_gueltig		Gültige Läufe der Kombination
laeufe_ungueltig		Ungültige Läufe der Kombination
kapazitaet_median	Bots	Median der Kapazität über die Läufe
kapazitaet_max	Bots	Größter Kapazitätswert der Läufe
kapazitaet_tps_median	Bots	Median von kapazitaet_tps
kapazitaet_tps_max	Bots	Größter Wert von kapazitaet_tps
tps_bei_kapazitaet_min	Ticks/s	Niedrigste Taktrate an der Kapazität über die Läufe
p95_bei_50_min_ms	ms	Kleinster p95-Wert bei 50 Bots
p95_bei_50_max_ms	ms	Größter p95-Wert bei 50 Bots
p99_bei_50_min_ms	ms	Kleinster p99-Wert bei 50 Bots
p99_bei_50_max_ms	ms	Größter p99-Wert bei 50 Bots
ram_bei_50_gb	GiB	Mittlerer RSS bei 50 Bots

Feld	Einheit	Bedeutung
cpu_bei_50_pct	%	Mittlere Auslastung der Server-Kerne bei 50 Bots
vergleich		Programmvergleich oder JVM-Vergleich
kennzahl		Verglichene Kennzahl
regel		Ob die Kennzahl vorab festgelegt oder nachträglich ergänzt wurde
a		Erste Kombination
a_min		Kleinster Wert der ersten Kombination
a_max		Größter Wert der ersten Kombination
b		Zweite Kombination
b_min		Kleinster Wert der zweiten Kombination
b_max		Größter Wert der zweiten Kombination
unterschied_belegt		Die Spannen der Wiederholungen überlappen sich nicht
besser		Kombination mit dem günstigeren Wert, nur bei belegtem Unterschied
aikar_min	ms	Kleinster p99-Wert mit Aikar-Flags in der Stufe
aikar_max	ms	Größter p99-Wert mit Aikar-Flags in der Stufe
g1_min	ms	Kleinster p99-Wert mit Standard-G1 in der Stufe
g1_max	ms	Größter p99-Wert mit Standard-G1 in der Stufe
aikar_niedriger		Alle drei Aikar-Werte liegen unter allen drei G1-Werten
g1_niedriger		Alle drei G1-Werte liegen unter allen drei Aikar-Werten

8 Quellen

[1] PaperMC (2026): Aikar's flags.
https://docs.papermc.io/paper/aikars-flags/
Abgerufen am 21.11.2026

[2] Minecraft Wiki (2026): Commands/tick.
https://minecraft.wiki/w/Commands/tick
Abgerufen am 21.11.2026

[3] OpenJDK: JEP 490: ZGC: Remove the Non-Generational Mode.
https://openjdk.org/jeps/490
Abgerufen am 21.11.2026

[4] Lithium-Projekt auf Modrinth (2026): Lithium.
https://modrinth.com/mod/lithium
Abgerufen am 21.11.2026

[5] PurpurMC: Purpur Documentation.
https://purpurmc.org/docs/purpur/
Abgerufen am 21.11.2026

9 Offenlegung

Eigeninteresse

Nexthosting verkauft selbst Gameserver, darunter Server für Minecraft, und hat deshalb ein wirtschaftliches Interesse an diesem Thema. Keines der getesteten Programme stammt von Nexthosting. Die Messung lief auf einer von Nexthosting bestellten Test-VM. Die Studie bewertet keine Angebote von Hostern. Verantwortlich für diese Studie ist Nexthosting, Kontakt: support@nexthosting.net.

Autorenschaft

Erik Friedl, Nexthosting. Herausgeber ist Nexthosting.

Lizenz

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

10 So zitierst du diese Studie

APA

Friedl, E. & Nexthosting. (2026). Server-Härtetest: Wie viele Protokoll-Bots tragen Vanilla, Paper, Purpur und Fabric mit Lithium? (Version 1.0.0) [Bericht]. Nexthosting. https://nexthosting.net/studien/server-haertetest-minecraft

BibTeX

```
@techreport{nexthosting2026serverhaertetestminecraft,
  author     = {Erik Friedl and {Nexthosting}},
  title      = {Server-Härtetest: Wie viele Protokoll-Bots tragen Vanilla, Paper, Purpur und Fabric mit
Lithium?},
  institution= {Nexthosting},
  type       = {Studie},
  year       = {2026},
  version    = {1.0.0},
  url        = {https://nexthosting.net/studien/server-haertetest-minecraft},
  urldate    = {2026-11-22},
  language   = {german}
```

```
}
```

11 Änderungen und Korrekturen

Version	Datum	Art	Änderung
1.0.0	22.11.2026	Fassung	Erste Fassung.