Seminar Report

PostgreSQL Benchmark

Katrena Shihada

MatrNr: 12204741

Betruer: Jonathan Decker

Georg-August-Universität Göttingen
Institute of Computer Science

17. Juni 2024

Declaration on the use of ChatGPT and comparable tools in the context of examinations

In this work I have used ChatGPT or another AI as follows:

- ☐ Not at all
- ☐ During brainstorming
- ☐ When creating the outline
- ☐ To write individual passages, altogether to the extent of 0% of the entire text
- ☐ For the development of software source texts
- ☑ For optimizing or restructuring software source texts
- ☐ For proofreading or optimizing
- ☐ Further, namely: -

I hereby declare that I have stated all uses completely.
Missing or incorrect information will be considered as an attempt to cheat.

Inhaltsverzeichnis

1 Einführung **1**

2 Grundlagen **2**
2.1 Benchmark 2
2.2 PostgreSQL 2
2.3 Benchbase 3
2.4 TPCC 3
2.5 Boxplot 3

3 Methodik **4**
3.1 Zielsetzung 4
3.2 Hypothesen 4
3.3 Performance Tuning/Leistungsoptimierung 4
3.4 Benchmarks 5
3.5 Einstellung 7

4 Auswertung und Analyse **8**
4.1 Auswertung der Merkmale der TPCC Benchmarks 9
4.2 Auswertung der Benchmarks 14
4.3 Fazit 15

Literatur **17**

A Abbildungen **A1**

Abbildungsverzeichnis **A13**

1 Einführung

Durch die immer stärker datengetriebene Welt sind Unternhemen und Organisationen auf höhere Leistungen von Datenbankmanagementsysteme (DBMS) angewiesen, um ihre alltäglichen Geschäftzprozesse Effizient durchzuführen. Als eines der führenden Open-Source-DBMS hat sich PostgreSQL durch das Engagement der Open-Source-Gemeinschaft etabliert. PostgreSQL sieht sich trotz seiner Popularität – genauso wie jedes andere DBMS - mit der Herausforderung konfrontiert, seine Leistung kontinuierlich zu optimieren. Es muss den steigenden Anforderungen an der Verarbeitung und der Verfügbarkeit von Daten gerecht werden.

Die Optimierung der Performance von PostgreSQL ist jedoch kein triviales Vorhaben. Es gibt zahlreiche und umfangreiche Anwendungsfälle die unterschiedlichen Herangehensweisen an das Perofrmance Tuning verlangen. Die Identifizierung von Engpässen und die Entwicklung von Optimierungsstrategien erfordern ein tiefes Verständnis der zugrundeliegenden Performance-Eigenschaften von PostgreSQL. Darüber hinaus sind effektive Werkzeuge und Methoden zur Performance-Analyse erforderlich.

An dieser Stelle werden Benchmarks eingesetzt. Durch die Simulation realer Einsatzszenarien bieten Benchmarks eine strukturierte Methode zur Bewertung der Performance von PostgreSQL und zur Identifizierung von Optimierungsmöglichkeiten. Sie helfen Entwicklern und Administratoren, Performance-Probleme zu erkennen und gezielte Optimierungsmaßnahmen einzuleiten.

Das Ziel dieser Arbeit ist es, die Rolle von Benchmarks bei der Performance-Optimierung von PostgreSQL zu untersuchen. Es soll gezeigt werden, wie Benchmarks dazu beitragen können, die Performance von PostgreSQL für spezifische Anwendungsfälle zu verbessern.

Durch eine gründliche Analyse von Benchmarks und ihrer Anwendungsfälle werden die folgenden Beiträge geleistet:

- Es werden Benchmarks verwendet, um potentielle Leistungseinschränkungen in PostgreSQL zu identifizieren und zu verstehen, sowie Ihren Einfluss auf bestimmte Anwendungsfälle zu untersuchen.

- Basierend auf den Ergebnissen der Benchmarks werden gezielte Optimierungsstrategien für PostgreSQL entwickelt, um seine Leistung für spezifische Workloads zu maximieren.

In den folgenden Kapiteln werden die Grundlagen 2 zu Benchmarks und Performance-Tuning erläutert. Im weiteren Verlauf wird das Design und die Implementierung eigens entwickelter Benchmarks behandelt 3, gefolgt von der Durchführung von Performance-Tests und der Identifizierung von Optimierungsstrategien. Abschließend werden die Ergebnisse bewertet und Schlussfolgerungen 4 gezogen , die einen Ausblick auf zukünftige Entwicklungen geben.

2 Grundlagen

2.1 Benchmark

Benchmark[1] [Maj] ist eine Methode zur Bewertung der Leistung von Computersystemen, insbesondere Datenbankmanagementsysteme wie PostgreSQL. Hierfür existieren verschiedene Tools und Standards, die verwendet werden können, um die Leistung zu messen und zu bewerten. Die Durchführung verschiedener Benchmarks an einem Datenbankmanagementsystem ermöglicht es herauszufinden, welche Perfomance Tuning-Maßnahmen relevant und ideal sind für die Verbesserung einer Datenbank. Durch den Einsatz verschiedener Benchmarks können unterschiedliche Situationen analysiert werden, um die spezifischen Einstellungen zu identifizieren, die die Leistung für bestimmte Anwendungsfälle maximieren.
Daher ist die Durchführung einer Benchmarks wichtig, um die Effektivität eines Systems zu bewerten.

2.2 PostgreSQL

Die in dieser Arbeit verwendete Dantenbank Software ist PostgreSQL. Es ist ein objektrelationales Open-Source-Datenbankmanagementsystem [2] [Dif+13]. Dieses wird verwendet für transaktionale und analytische Arbeitslasten. Es implementiert und erweitert die SQL-Programmiersprache und bietet eine skalierbare, zuverlässige und erweiterbare Plattform für die Speicherung, Verwaltung und Abfrage von strukturierten Daten. PostgreSQL ermöglicht, komplexe Anwendungen und Workloads zu entwickeln und zu betreiben und wird in einer Vielzahl von Anwendungen, wie Webanwendungen, Unternehmensanwendungen oder Datenanlysen eingesetzt. Die Anfänge von PostgreSQL liegen im Jahr 1986 als Teil des POSTGRES-Projekts an der University of California in Berkeley. Seit mehr als 35 Jahren wird die Kernplattform stetig weiterentwickelt. Eine Umfrage von StackOverflow aus dem Jahr 2022 zeigt, dass PostgreSQL die am häufigsten gewählte Datenbank unter den professionellen Entwicklern ist. Es geben 46% von den 73.268 befragten Programmierern aus 180 Ländern weltweit an, dass PostgreSQL die Datenbank ist, mit der sie im vergangenen Jahr am intensivsten gearbeitet haben. Sie haben auch vor, die Datenbank im kommenden Jahr zu nutzen.[3] [Pol23] Mit seiner bewährten Architektur, Zuverlässigkeit und Datenintegrität hat sich PostgreSQL als renommierte Datenbanklösung etabliert. Die Erweiterbarkeit und die Open-Source-Community machen es möglich, dass PostgreSQL kontinuierlich leistungsstarke und innovative Lösungen liefert. Es ist plattformübergreifend und kann auf verschiedenen Betriebssystemen wie Linux, Windows, macOS und BSD genutzt werden. Mit dem Release der Version 16 im September 2023[4] [Pos24] erfüllt PostgreSQL mindestens 170 der 179 obligatorischen Merkmale für die SQL:2023 Core-Konformität. Zum Zeitpunkt der Erstellung dieses Dokuments erfüllt keine relationale Datenbank die vollständige Konformität mit diesem Standard. Durch die regelmäßigen Release-Zyklen wird PostgreSQL kontinuierliche verbessert und innoviert und bietet

[1] `https://aerospike.com/blog/best-practices-for-database-benchmarking/` zuletzt besucht 30.04.2024

[2] `https://www.postgresql.org/about/` zuletzt besucht 04.05.2024

[3] `https://linuxpolska.com/en/knowledge-base/blog/postgresql-the-database-most-frequently-chosen-by-` zuletzt besucht 04.05.2024

[4] `https://www.postgresql.org/about/` zuletzt besucht 04.05.2024

zahlreiche Möglichkeiten für ein Performance Fine Tuning, wie Arbeitspeicherzuweisung, Caching-Mechanismen und paralleler Verarbeitung.

2.3 Benchbase

Auf Benchbase[5] [Dif+13] wurden in der vorliegenden Arbeit TPCCs durchgeführt. Benchbase ist eine spezifische Softwareplattform, welche entwickelt wurde, um die Leistung von verschiedenen Datenbankmanagementsystemen unter verschiedenen Bedingungen zu testen und zu vergleichen.

2.4 TPCC

TPCC [6] [exp13] steht für Transaction Processing Performance Concil Benchmark C. TPCC ist ein Benchmark für die Online-Transaktionsverarbeitung, und gehört zu OLTP (Online Transaction Processing). TPCC ist komplexer als frühere OLTP-Benchmarks, wie z.B. TPC-A, da in TPCC mehrere Transaktionen ausgeführt werden. Es umfasst eine Mischung aus fünf gleichzeitigen Transaktionen unterschiedlichster Art und Komplexität. Diese umfassen sowohl lese- als auch aktualisierungsintensive Transaktionen. Sie stellen eine Simulation der Aktivitäten in einer komplexen OLTP-Anwendungsumgebung dar. Die Leistungskennzahl für diesen Benchmark ist ein „Business Throughput“, der die Anzahl der verarbeiteten Aufträge pro Minute in Transaktionen pro Minute (tpm) ausgibt.

2.5 Boxplot

Im Deutschen Kastengrafik genannt, bieten Sie eine übersichtliche Darstellung der Verteilung der Benchmark-Daten. Es wird das Minimum, das untere Quartil, der Median und das obere Quartil sowie das Maximum dargestellt.

- **Quartile**: Die Box selbst repräsentiert den Interquartilsabstand (IQR), der den Bereich zwischen dem ersten (Q1) und dritten Quartil (Q3) abdeckt.
- **X-Achse**: Zeigt die Werte der verschiedenen Performance-Tuning-Parameter, die auf den Benchmarks angewendet wurden.
- **Y-Achse**: Zeigt die gemessenen Werte für Latenz und Throughput.
- **Median**: Der horizontale Strich innerhalb der Box zeigt den Median der Daten an.
- **Whisker**: Die vertikalen Linien, die aus der Box herausragen, erstrecken sich bis zum Minimum und Maximum der Daten oder bis zu 1,5-mal dem IQR über die Quartile hinaus.
- **Ausreißer**: Punkte, die außerhalb der Whisker liegen, werden als potenzielle Ausreißer betrachtet und separat als Punkte dargestellt.

[5] `https://github.com/cmu-db/benchbase` zuletzt besucht 30.04.2024
[6] `https://www.tpc.org/tpcc/default5.asp` zuletzt besucht 30.04.2024

3 Methodik

3.1 Zielsetzung

Das Ziel dieser Arbeit ist es, die Performance einer PostgreSQL-Datenbank durch gezielte Performance-Tuning-Maßnahmen zu verbessern, so dass eine höhere Anzahl von Transaktionen pro Minute (tpm) für den TPCC-Benchmark erreicht werden. Durch die Anpassung des TPCC Benchmarks basierend auf TPCC default konnten eine Vielzahl von Anwendungsfällen abgedeckt werden. Dies ermöglichte, dass verschiedene Arbeitlasten simuliert und die Leistung der PostgreSQL-Datenbank analysiert werden konnte. Es soll ein fundiertes Verständnis dafür entwickelt werden, welche Feinabstimmung für welche Benchmarks und die damit verbundenen Anwendungsfälle am vorteilhaftesten ist.

3.2 Hypothesen

Es wird angenommen, dass durch die Anpassung verschiedener Konfigurationseinstellungen und Optimierungen innerhalb von PostgreSQL, wie max worker processes, shared buffers und effective cache size, die Leistung im TPCC-Benchmark verbessert werden kann. Die ausgewählten Benchmarks (siehe Kapitel 3.4) sollen repräsentativ für eine Vielzahl von Anwendungsfällen stehen. Dabei wird erwartet, dass deutlich wird, welche Feinabstimmungsmaßnahmen für welchen Benchmark und damit für welchen Anwendungsfall von Vorteil sein können.

3.3 Performance Tuning/Leistungsoptimierung

Es gibt verschiedene Möglichkeiten, die Leistung einer PostgreSQL Datenbank zu optimieren, beispielsweise durch Anpassung des Database Design, Hardware Tuning, Anpassung des Netzwerks oder des Tunings der Datenbank-Konfigurationen. Die vorliegende Arbeit konzentriert sich auf das Tuning der Datenbank-Konfigurationen, auch als PostgreSQL Performance Tuning bekannt. Dies ist ein Prozess, bei dem die Konfigurationen geändert werden, um eine bessere Leistung der Datenbank zu erreichen. Hierbei wird sich auf die folgenden Konfigurationsparameter konzentriert: *max worker processes*, *shared buffers*, *effective cache size*[7] [Alb].

- *max worker processes*: Dieser Parameter legt die maximale Anzahl der Worker Processes, die PostgreSQL starten kann, fest. Der Worker Processes umfasst die Abfrageparalität, Hilfsprozesse sowie logische Replikationen und Hintergrundaufgaben. Hintergrundprozesse werden für verschiedene Aufgaben verwendet zum Beispiel Hintergrundwartungvorgänge, Hintergrundschreiber und parallele Abfrageverarbeitung. Der Standardwert ist 8.

- *shared buffers*: Auch als Seitencache bekannt, ist ein Bereich des Arbeitsspeichers, der von PostgreSQL zur Zwischenspeicherung von häufig verwendeten Datenblöcken verwendet wird. Der Standardwert ist im Normalfall 128 MB und es wird empfohlen, ihn auf 25% des RAMs einzustellen. Wenn eine Abfrage ausgeführt wird und PostgreSQL Zugriff auf Datenblöcke benötigt, überprüft es zuerst, ob diese Blöcke bereits im *shared buffers* vorhanden sind. Wenn ja, kann PostgreSQL direkt und

[7] `https://postgresqlco.nf/doc/en/param/` zuletzt besucht 30.04.2024

somit schneller auf diese Daten zugreifen, als wenn diese auf der Festplatte gespeichert sind. Bei ausreichend Arbeitsspeicher (RAM) im System kann die Erhöhung der Anzahl der *shared buffers* zu einer Verbesserung der Ausnutzung der verfügbaren Ressourcen und zu einer Steigerung der Effizienz des Datenbankservers beitragen. Dieser kann je nach Anwendungsfall und Arbeitslast variiert werden. Die Veränderung des *shared buffers* kann also positiv die Leistung beeinflussen.

- *effective cache size*: Ist eine PostgreSQL-Konfigurationsoption, die dem Abfrageplaner eine Schätzung darüber liefert, wie viel Cache-Speicher auf dem System verfügbar ist, der verwendet werden kann, um Datenbankblöcke zwischenzuspeichern. Diese Schätzung hilft PostgreSQL, fundierte Entscheidungen darüber zu treffen, wie Abfragen ausgeführt werden sollen. Dies gilt insbesondere für die Auswahl von Index-Scans oder die Definition von Join-Strategien. Es ist wichtig zu betonen, dass *effective cache size* nur dazu dient, eine Schätzung des verfügbaren Cache-Speichers zu liefern und nicht dazu verwendet werden kann, Ressourcen direkt zuzuweisen oder zu reservieren. Vielmehr verwendet PostgreSQL diesen Wert als Hinweis darauf, wie gut Daten im Cache gehalten werden können, um die Ausführungspläne von Abfragen zu optimieren. Die tatsächliche Zuweisung von Speicherressourcen erfolgt nicht über diese Konfigurationsoption, sondern über andere Parameter wie z. B. *shared buffers*. Der Standardwert ist 4GB.

3.4 Benchmarks

Benchmarks sind eine objektive Möglichkeit, die Leistungsfähigkeit von Datenbanksystemen zu bewerten und zu verbessern, indem sie eine standardisierte Umgebung für Leistungstests bereitstellen. Durch die Analyse der Ergebnisse von Benchmarks, wie dem TPC-C, werden die Stärken und Schwächen von PostgreSQL deutlich, wodurch gezielte Maßnahmen zur Leistungsoptimierung identifiziert werden können. Dies ermöglicht es, fundierte Entscheidungen bei der Auswahl und Konfiguration von Datenbanksystemen zu treffen und sicherzustellen, dass Sie den Anforderungen von ihren entsprechenden Anwendungen gerecht werden. Die Benchmarks Terminals, Scalefactor, Time, Rate und Batchsize sind die Merkmale von TPCC-Benchmarks. Allerdings bilden Die Gewichtungen innerhalb des TPCC-Benchmarks den Fokus der Leistungsbewertung, da sie die relative Häufigkeit der verschiedenen Transaktionstypen im Benchmark-Szenario bestimmen. Insbesondere spiegeln die Gewichtungen die Mischung aus Lese- und Schreiboperationen wider, die für realistische Workloads in Datenbankmanagementsystemen typisch sind. Damit werden Anwender in die Lage versetzt, die Auswirkungen unterschiedlicher Transaktionsmuster auf die Effizienz und Skalierbarkeit von PostgreSQL zu untersuchen und Optimierungsstrategien zu entwickeln, die speziell auf die Anforderungen ihrer Anwendung zugeschnitten sind. Es wurden die nachfolgenden direkten relevanten Benchmarks für die Konfiguration von TPCC Benchmarks durchgeführt:

- *Terminals*: Ist die Anzahl der Benutzer, die gleichzeitig Transaktionen durchführen. In dieser Forschung hatte diesen Benchmarks den Wert 3.

- *Scalefactor*: Es bezieht sich auf die Größe der simulierten Datenbank. Diese Größe bestimmt das Volumen der Daten und die Merkmale während des Benchmark Test. Es wird als Zahl ausgedück. Größere Scalefactor repräsentieren größere Datenbanken mit mehr Daten. Der Wert der Terminals ist 8

- *Time*: Die Messdauer ist der Zeitraum, in dem das System kontinuierlich mit Anfragen belastet wird. Gemessen und bewertet wird während dieser Zeit die Performance des Systems. Zu Beginn der Messzeit durchläuft das System eine Aufwärmphase, in der die Caches gefüllt und optimiert werden. Die eigentliche Messung der Systemleistung beginnt erst nach dieser Aufwärmphase. Ausgeführt mit dem Wert 120

- *Rate*: Ist die Geschwindigkeit, mit der Transaktionen innerhalb der Benchmarks ausgeführt werden. Ausgeführt mit dem Wert 40000

- *Batch size*: Steht für die Anzahl der Transaktionen in einer Batchverarbeitung und beeinflusst die Gesamtleistung. Höhere Batchgrößen reduzieren die Anzahl von notwendigen Ein- und Ausgabevorgängen. Allerdings kostet ein größerer Batchsize eine längere Antwortzeit für einzelne Transaktionen. Ausgeführt mit dem Wert 2

Für TPCC wurden spezielle ausgewählte Anwendungsfälle wie Lese- und Schreiboperationen mit verschiedenen Gewichtungen berücksichtigt. Die sogenannten Weights in Benchmarks bestehen aus fünf parallelen Tansaktionen im TPCC Benchmark in der folgenden Reihenfolge: New Ordner, Payment, Order Status, Delivery, Stock Level.

- *New Order*: Diese Transaktion simuliert den Prozess der Erstellung eines neuen Auftrags durch einen Kunden. Dies ist eine Schreiboperation, indem neue Aufträge in das System eingefügt werden.

- *Payment*: Repräsentiert den Zahlungsvorgang eines Kunden für einen bestehenden Auftrag. Die Zahlung aktualisiert den Saldo des Kunden. Dies ist eine Lese- und Schreiboperation, indem Informationen zu einem Auftrag ausgelesen und die Zahlungsinformationen aktualisiert werden.

- *Order Status*: Fragt den Status einer Bestellung eines Kunden ab (Leseoperation).

- *Delivery*: Besteht aus der Bearbeitung eines Stapels von 10 neuen Aufträgen. Es ist eine Schreiboperation, indem der Lagerbestand aktualisiert wird.

- *Stock Level*: Der Geschäftsvorfall Lagerbestand ermittelt die Anzahl der kürzlich verkauften Artikel, deren Lagerbestand unter einem Schwellenwert liegt (Leseoperation).

Die Standardgewichtung im TPCC für die fünf Transaktionen lautet, wie folgt: 45,43,4,4,4 (= 45 % New Order, 43 % Payment, 4 % Order Status, 4 Delivery, 4 % Stock Level). Es handelt sich um eine Mischung von verschiedenen Transaktionstypen. Die Gewichtung (in %) bestimmt, wie oft jede Transaktion im Benchmark aufgerufen wird. In der vorliegenden Arbeit wurden die nachfolgenden fünf neuen Benchmarks mit den aufgeführten Gewichtungen ausgewählt. Ziel dieser fünf Benchmarks ist es, die Arbeitslast zu steuern und spezifische Leistungsaspekte zu testen.

- *deafult*: 45,43,4,4,4.
  Es handelt sich um die Standardgewichtung bei TPCC

- *TPCC-read-heavy*: 0,0,0,0,100.
  Bei diesem Benchmark wird eine große Anzahl von Leseoperationen in der Datenbank ausgeführt.

- *TPCC read mid*: 0,0,100,0,0.
  Hier werden mittlere bis schwere Leseoperationen auf der Datenbank duchgeführt.
- *TPCC read write balances*: 50,50,0,0,0.
  Im gleichen Maß werden Lese- und Schreiboperationen durchgeführt.
- *TPCC write mix*: 91,0,0,9,0.
  Sowohl Schreib- als auch Leseoperation werden berücksichtigt, wobei der Schwerpunkt auf den Schreibvorgängen liegt.
- *TPCC wirte new*: 100,0,0,0,0.
  Es werden ausschließlich Schreiboperationen durchgeführt.

3.5 Einstellung

Die folgende Hardware wurde eingesetzt

- 4 CPUs, 8GB RAM.
- Betriebssystem ist Ubuntu 22.04.3 LTS.

Die installierte PostgreSQL-Version ist die Version 14.

Um die Genauigkeit der Ergebnisse zu verbessern, werden zwei getrennte Maschinen für die Ausführung der Benchmarks und die Ausführung der Datenbank verwendet. Im Folgenden wird das System, auf dem der Benchmark BenchBase ausgeführt wird[8], als Client Node bezeichnet, darauf würde die 21. Java-Version ausgeführt um Benchbase zu bauen, während der Node, auf dem die Datenbank ausgeführt wird, als Datenbank Node bezeichnet wird.

[8] `https://github.com/cmu-db/benchbase/commit/752e337d839b67f36034d3cac60bf28603e1b368`
commit hash: 752e337d839b67f36034d3cac60bf28603e1b368

4 Auswertung und Analyse

In diesem Kapitel werden die Ergebnisse der Analyse vorgestellt. Sie sind durch die Erstellung und Auswertung von Grafiken, insbesondere von Boxplots, veranschaulicht. Diese Diagramme basieren auf Benchmarks und zeigen die Performance-Tuning-Tests. Die in Kapital 2.5 defenierte Ausreißer sind besonders wichtig, da sie jeweils 20% der Gesamtperformance-Ergebnisse ausmachen. Jeder Außreißer stellt einen signifikanten Anteil der Gesamtperformance dar.

Zur Bewertung der Leistungsfähigkeit wurden insgesamt 11 verschiedene Benchmarks entwickelt,die sich in zwei Gruppen unterteilen lassen, wobei die ersten 5 die Merkmale der TPCC-Benchmarks und die übrigen die neuen Benchmarks darstellen. Um die Robustheit der Daten zu gewährleisten, wurde jeder Benchmark fünfmal wiederholt. Die durchgeführten Benchmarks und die spezifischen Parameter, die bei der Durchführung der Benchmarks verwendet wurden, sind folgend zusammengefasst:

- Terminals: Der Standardwert für Terminals ist 1 und wird um 3 erhöht.
- Scalefactor: Der Standardwert für Scalefactor ist 1 und wird um 3 erhöht.
- Time: Der Standardwert für Time ist 60 und wird um 120 erhöht.
- Rate: Der Standardwert für Rate ist 10000 und wird um 40000 erhöht.
- Batchsize: Der Standardwert für Batchsize ist 128 und wird um 256 erhöht.
- TPCC read heavy 0,0,0,0,100
- TPCC read mid 0,0,100,0,0
- TPCC read write balanced 50,50,0,0,0
- TPCC write mixed 91,0,0,9,0
- TPCC write new 100,0,0,0,0

Für das Performance-Tuning wurden die Benchmarks unter kontrollierten Bedingungen durchgeführt. Nachfolgend werden die signifikanten Beobachtungen vorgestellt.

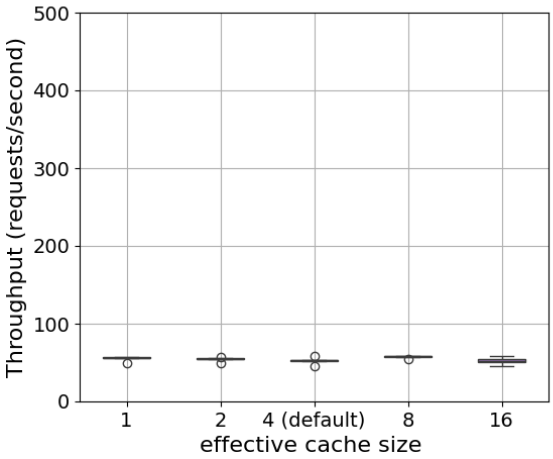
Bei den Benchmarks Scalefactor, Time, read heavy und read mid sind keine signifikanten Auswirkungen auf den Durchsatz und die Latenz zu beobachten.

4.1 Auswertung der Merkmale der TPCC Benchmarks

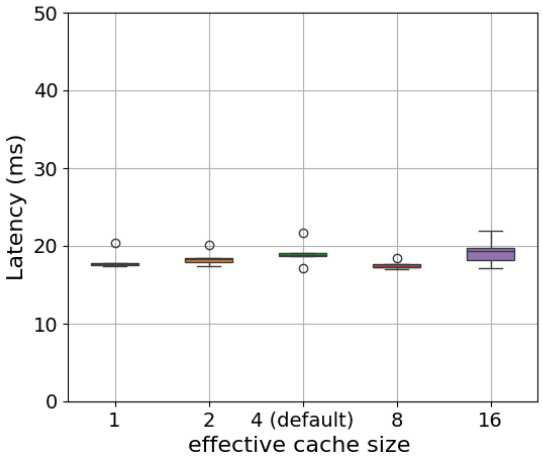
Die Auswertung von Default

Effective cache size

- Bei den Werten 1,2 und 4 liegen die Durchsatzwerte nahe beieinander. Dies könnte darauf hindeuten, dass die ausgeführten Abfragen in diesen Fällen nicht viel Cache benötigen. Bei einer effektiven cache-Größe von 16 sinkt der Durchsatz stark ab, da dieser Wert weit über dem eigentlichen Cache des Systems liegt.
- Bei einer Cache-Größe von 16 beginnt die Latenz zu schwanken, was darauf hindeutet, dass der Cache-Wert weit vom tatsächlichen RAM des Systems entfernt ist, welches sich negativ auf die Leistung auswirkt.



(a) Throughput effective cache size default

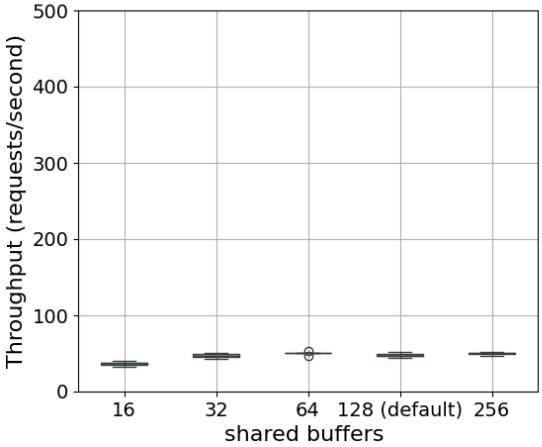


(b) Latency effective cache size default

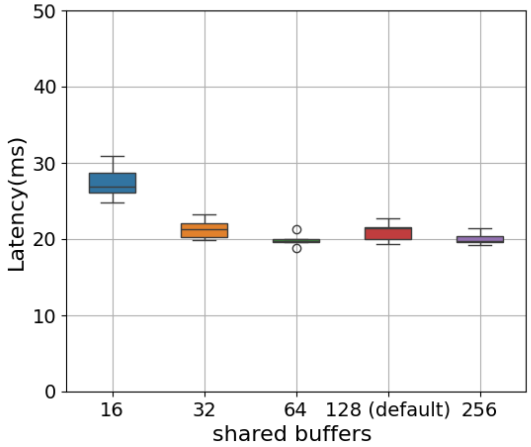
Abbildung 1: Ausführung default mit der Einstellung Performance Tuning effective cache size

Shared buffers

- Der Durchsatz des Systems sinkt deutlich, wenn der shared buffer auf einen niedrigen Wert gesetzt wird.
- Gleichzeitig wird die Latenz erhöht. Dies deutet darauf hin, dass eine zu geringe Puffergröße die Fähigkeit des Systems, die Daten effizient zu verarbeiten, beeinträchtigt.



(a) Throughput shared buffers default



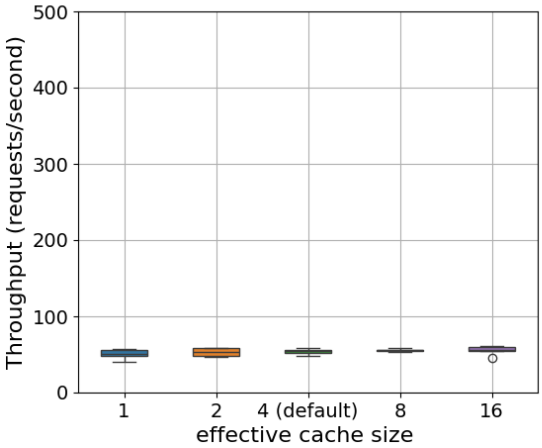
(b) Latency shared buffers default

Abbildung 2: Ausführung Default mit der Einstellung Performance Tuning shared buffers

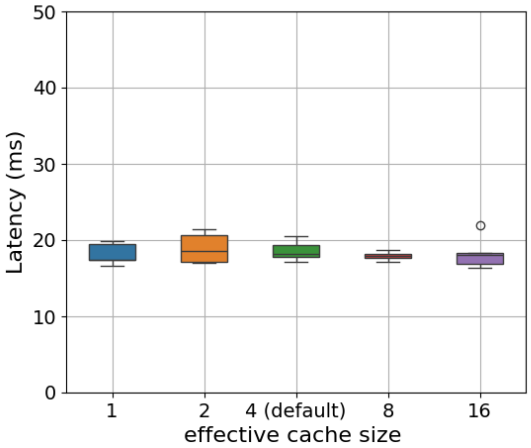
Die Auswertung von Batchsize

Effective cache size

- Bei einem höheren Batchsize sinkt die Latenz leicht, insbesondere wenn die effektive cache-Größe höher ist. Dies deutet darauf hin, dass das System bei größeren Batches effizienter arbeitet, sofern ausreichend Cache vorhanden ist.



(a) Throughput effective cache size batchsize

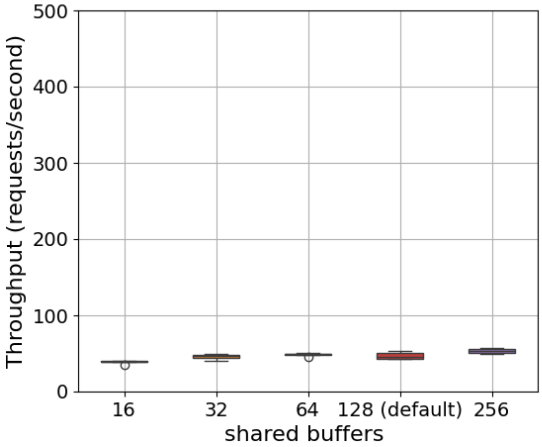


(b) Latency effective cache size batchsize

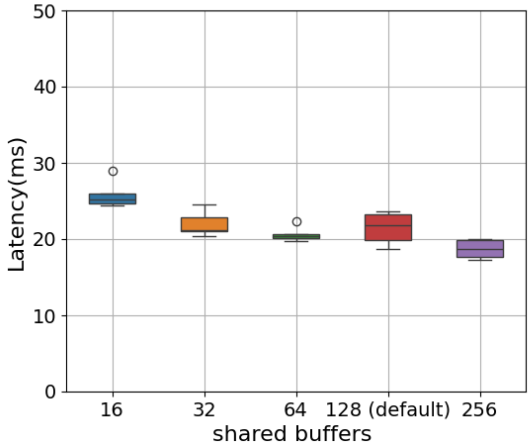
Abbildung 3: Ausführung der Batchsize mit der Einstellung Performance Tuning effective cache size

Shared buffers

- Bei höheren Batchsizes verbessert sich die Performance, wenn die Shared buffers entsprechend angepasst sind. Eine größere Puffergröße ermöglicht es dem System, die Daten effizienter zu verarbeiten und den Durchsatz zu erhöhen.



(a) Throughput shared buffers batchsize



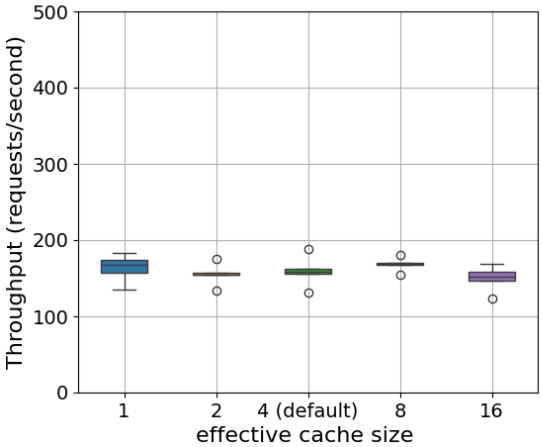
(b) Latency shared buffers batchsize

Abbildung 4: Ausführung Batchsize mit der Einstellung Performance Tuning shared buffers

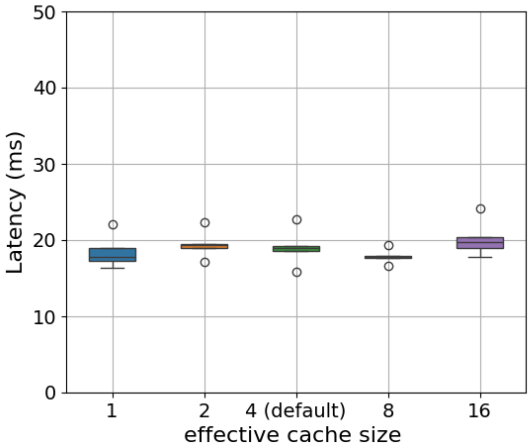
Die Auswertung von Terminals

Effective cache Size:

- Mehrere Nutzer profitieren von einer optimalen Cache-Größe. Eine gut konfigurierte Cache-Größe erhöht den Durchsatz, da mehr Daten im Cache gehalten und schneller verarbeitet werden können.
- Die Latenz bleibt relativ konstant, unabhängig von der Cache-Größe. Dies zeigt, dass die Cache-Größe primär den Durchsatz beeinflusst, während die Latenz stabil bleibt.



(a) Throughput effective cache size terminals

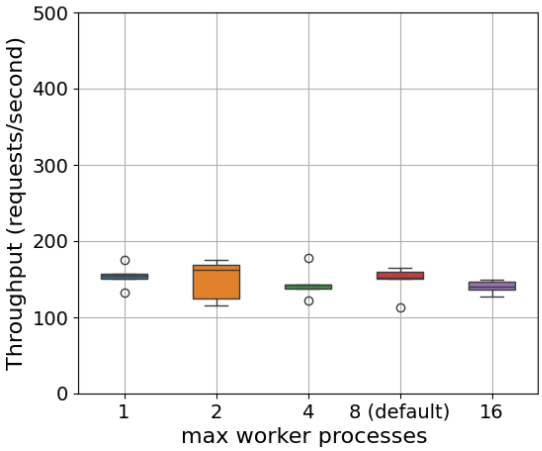


(b) Latency effective cache size terminals

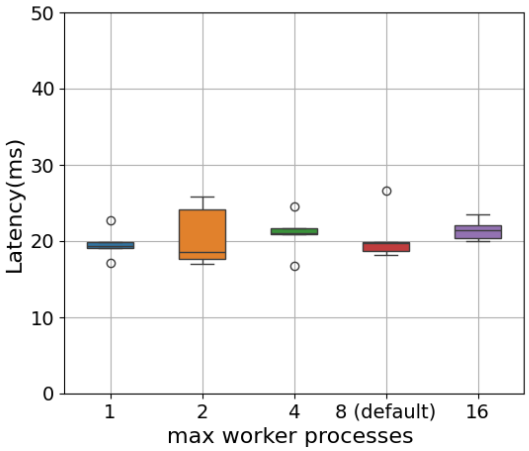
Abbildung 5: Ausführung Terminals mit der Einstellung Performance Tuning effective cache size

Max worker processes

- Die Anzahl der maximalen Worker-Prozesse reagiert ähnlich auf die Anzahl der Terminals wie die Cache-Größe. Eine Erhöhung der Worker-Prozesse kann den Durchsatz verbessern, da mehr Anfragen parallel bearbeitet werden.
- Trotz der erhöhten Worker-Prozesse bleibt die Latenz stabil. Dies bedeutet, dass die zusätzliche Parallelisierung keine negativen Auswirkungen auf die Antwortzeit hat, was positiv zu bewerten ist.



(a) Throughput max worker processes terminals

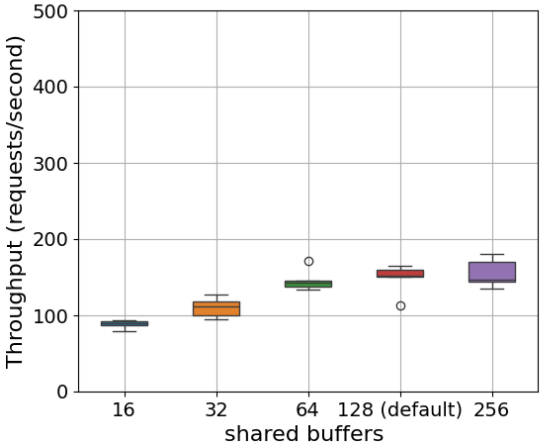


(b) Latency max worker processes terminals

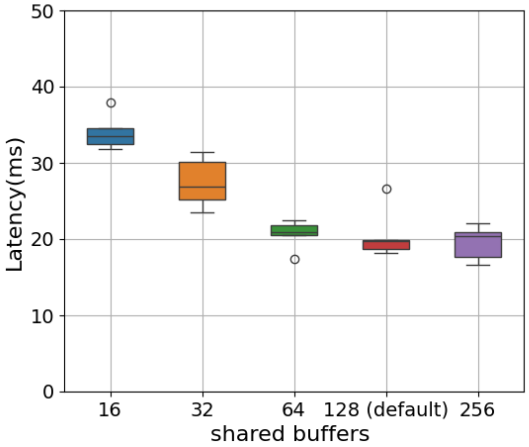
Abbildung 6: Ausführung Terminals mit der Einstellung Performance Tuning max worker processes

Shared buffers

- Eine Erhöhung der Shared buffers führt zu einer Verbesserung des Durchsatzes, besonders bei mehreren Terminals. Größere Puffergrößen ermöglichen eine effizientere Verarbeitung und Speicherung der Daten.
- Mit der Erhöhung der Shared buffers sinkt die Latenz, was ebenfalls positiv ist. Der Standardwert für Shared buffers erweist sich als ausreichend, aber höhere Werte verbessern die Performance weiter. Niedrigere Puffergrößen führen hingegen zu schlechteren Ergebnissen in Bezug auf Durchsatz und Latenz.



(a) Throughput shared buffers terminals



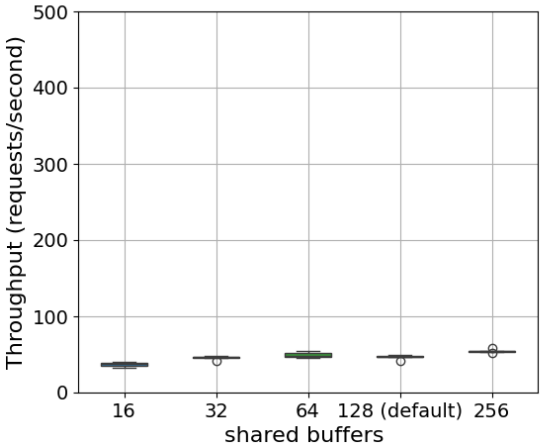
(b) Latency shared buffers terminals

Abbildung 7: Ausführung der Benchmark terminal der Einstellung Performance Tuning shared buffers

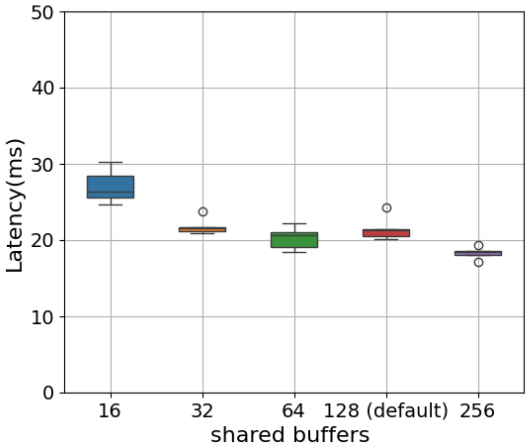
Die Auswertung von Rate

Shared buffers

- Die Shared buffers zu vergrößern führt zu einer Verbesserung des Durchsatzes. Größere Puffer ermöglichen eine effizientere Verarbeitung und Speicherung von Anfragen.
- Eine Erhöhung der Shared buffer führt zu einer Verringerung der Latenz, was auf eine verbesserte Effizienz des Systems hinweist. Geringere Puffergrößen beeinträchtigen die Performance erheblich.



(a) Throughput shared buffers rate



(b) Latency shared buffers rate

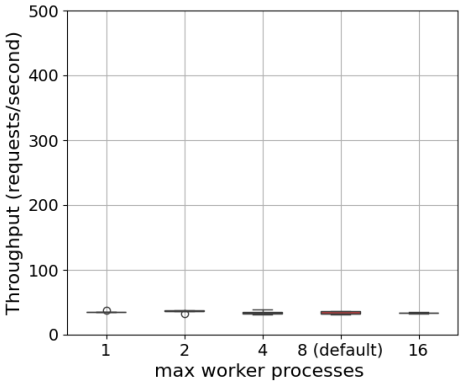
Abbildung 8: Ausführung der Benchmark rate mit der Einstellung Performance Tuning shared buffers

4.2 Auswertung der Benchmarks

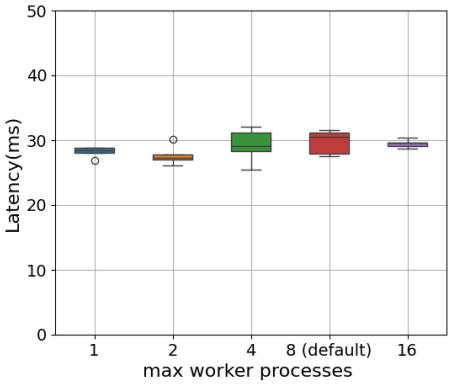
Die Auswertung von Benchmarks TPCC write mix

Max worker processes

- Der Durchsatz bei max worker processes sind sehr leichte Schwankungen und die Latenz bleibt relative konstant.



(a) Throughput max worker processes weights write mixed

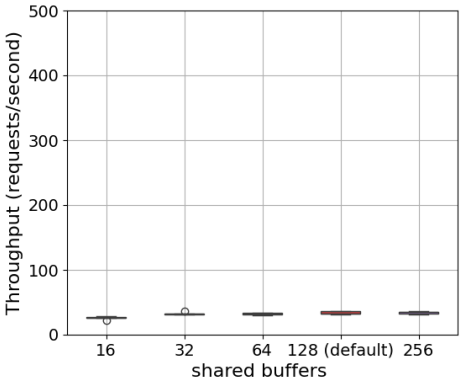


(b) Latency max worker processes weights write mixed

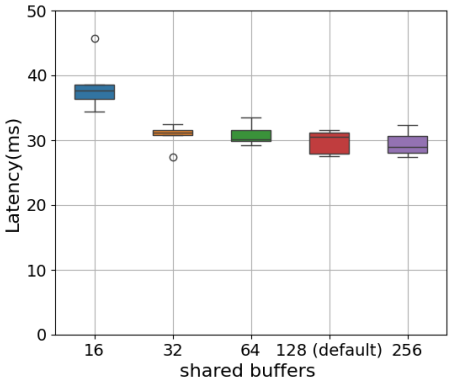
Abbildung 9: Ausführung der Benchmark weights write mix mit der Einstellung Performance Tuning max worker processes

Shared buffers

- Eine Verringerung der shared buffers führt zu einem deutlichen Rückgang des Durchsatzes. Bei niedrigeren shared buffer Werten zeigt sich eine Erhöhung der Latenz, insbesondere bei einem Wert von 16, was eine schlechtere Performance bedeutet.



(a) Throughput shared buffers weights write mixed



(b) Latency shared buffers weights write mixed

Abbildung 10: Ausführung der Benchmark weights write mix mit der Einstellung Performance Tuning shared buffers

Die Auswertung von Benchmarks TPCC read heavy und TPCC read mid:

- Für weights read heavy 20a 20b 20c 20d 20e 20f und weights read mid 21a 21b 21c 21d 21e 21f ist die signifikante Steigerung des Durchsatzes mit niedriger Latenz unter den drei Performance Tunings zu sehen. Ab einem Wert von 64 bei TPCC read heavy unter der Einstellung von shared buffers ist jedoch eine Abnahme und eine Zunahme der Latenz bei shared buffers zu beobachten.

Analyse: Die Durchführung von TPCC Benchmarks mit verschiedenen Performance-Tuning-Parametern auf Benchbase PostgreSQL haben die nachfolgenden Erkenntnisse geliefert, um das System zu optimieren.
Shared buffers: Bei Terminals, rate und batch size hat shared buffers einen positiven Effekt auf den Durchsatz und die Latenzzeit. Geringerer Buffer lieferte einen geringeren Durchsatz. Hingegen höherer Buffer einen höheren Durchsatz und geringere Latenz lieferte. Bei TPCC Read Heavy weist shared buffer einen höheren Wert als dem Standardwert auf. Es zeigt sich ein geringerer Durchsatz und eine höhere Latenz.
Effective cache size: Bei einem höheren Batchsize sinkt die Latenz leicht, insbesondere wenn die effektive Cache-Größe höher ist. Dies deutet darauf hin, dass das System bei größeren Batches effizienter arbeitet, sofern ausreichend Cache vorhanden ist. Die Auswertung der Terminals zeigt, dass bei einer optimalen Konfiguration der cache size der Durchsatz gesteigert wird, weil mehrere Terminals parallel arbeiten und so Daten schneller verarbeitet werden, die im Cache gehalten sind. Dabei ist keine Veränderung der Latenz ersichtlich. Das Performance Tuning Max Worker Processes hat keinen Einfluss auf die Benchmarks und kann daher vernachlässigt werden.

4.3 Fazit

Diese Arbeit beschäftigte sich mit der Durchführung und Analyse von TPCC Benchmarks auf Benchbase PostgreSQL unter verschiedenen Performance Tuning Parametern. Zu Beginn wurden die relevanten Benchmarks erstellt, um eine fundierte Grundlage für unsere Untersuchungen zu schaffen. Anschließend wurde ein Testsystem aufgesetzt und die entsprechenden Tests durchgeführt. Eine detaillierte Beobachtung und eine sorgfältige Dokumentation der Ergebnisse wurde während der Tests durchgeführt. Durch die Durchführung der Benchmarks mit gezielten Performance-Tuning-Maßnahmen konnte folgendes gezeigt werden: Gezielte Anpassung der shared buffer und der Batchgrößen führen zu erheblichen Leistungsverbesserungen. Durch die Optimierung dieser Parameter kann die Effizienz und Performance des Systems insbesondere bei großen Datenmengen und write-intensiven Workloads deutlich gesteigert werden. Im Gegensatz dazu haben die effektiven cache-Größen und die maximale Anzahl der Worker-Prozesse einen geringeren Einfluss auf die Performance des Systems und können auf Standardwerte belassen werden. Diese Erkenntnisse liefern wertvolle Hinweise zur weiteren Optimierung und Konfiguration von PostgreSQL-Systemen und können zur Leistungsmaximierung unter verschiedenen Workloads beitragen. Eine signifikante Verbesserung der Systemeffizienz kann durch die Fokussierung auf die richtigen Parameter, insbesondere des shared buffers, erreicht werden. Es gibt eine Reihe von Themen, die weiter untersucht werden können, um ein umfassenderes und tieferes Verständnis zu gewinnen. Bspw. können noch weitere Benchmarks erstellt werden, um ein größeres Spektrum von Anwendungsfällen abzudecken. Weiterhin können Langzeitmessungen durchgeführt werden. Eine weitere genauer Abstufung der Konfigurationsparameter, wie zum Beispiel der shared buffers und der batch size könnte helfen, die

optimalen Einstellungen noch genauer zu bestimmten. Außerdem können beim effective cache size noch größere Werte untersucht werden. Eine weitere Möglichkeit wären neue Performance Tuning Maßnahmen zu ermitteln.

Literatur

[Alb] Laurenz Albe. „PostgreSQL Documentation: max_worker_processes parameter“. en. In: (). URL: https://postgresqlco.nf/doc/en/param/ (besucht am 05. 06. 2024).

[Dif+13] Djellel Eddine Difallah u. a. „OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases“. In: *PVLDB* 7.4 (2013), S. 277–288. URL: http://www.vldb.org/pvldb/vol7/p277-difallah.pdf.

[exp13] author not explicitly. „TPCC-Benchmark: An Extensible Testbed for Benchmarking Relational Databases“. In: *PVLDB* 7.4 (2013), S. 277–288. URL: https://www.tpc.org/tpcc/default5.asp.

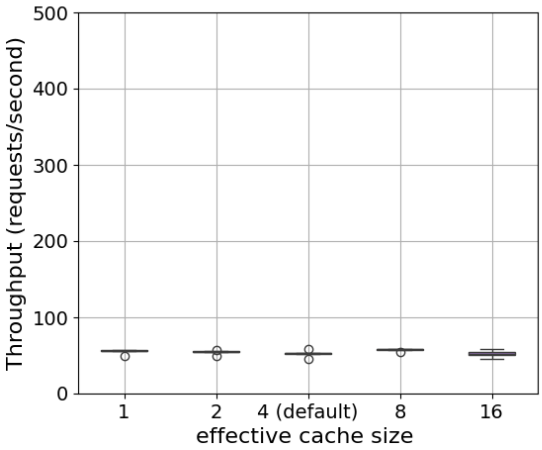
[Maj] Michael Majdalany. „Best practices for database benchmarking“. Aerospike. In: (). URL: https://aerospike.com/blog/best-practices-for-database-benchmarking/ (besucht am 07. 05. 2024).

[Pol23] Linux Polska. „PostgreSQL - the database most frequently chosen by developers and the future of DBMS | Linux Polska blog“. en-GB. In: *Linux Polska* (März 2023). URL: https://linuxpolska.com/en/knowledge-base/blog/postgresql-the-database-most-frequently-chosen-by-developers-and-the-future-of-dbms/ (besucht am 21. 05. 2024).

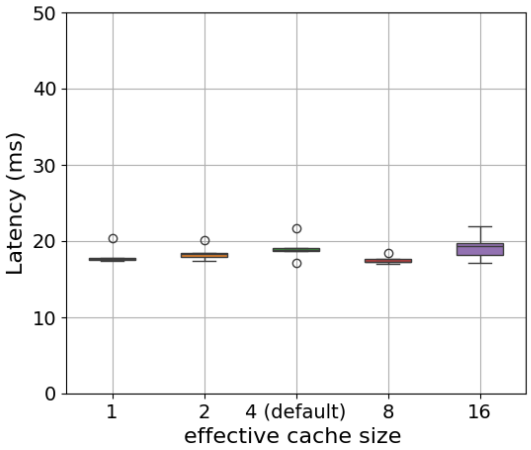
[Pos24] PostgreSQL Global Development Group. „About PostgreSQL“. In: *PostgreSQL Official Website* (2024). Accessed: 2024-06-06. URL: https://www.postgresql.org/about/.

A Abbildungen

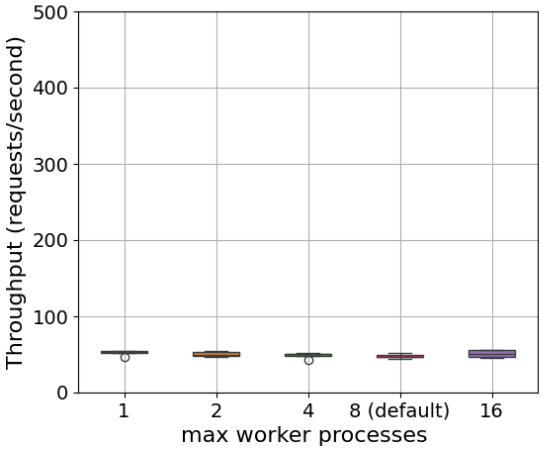
Die in Kapitel 12d beschriebenen



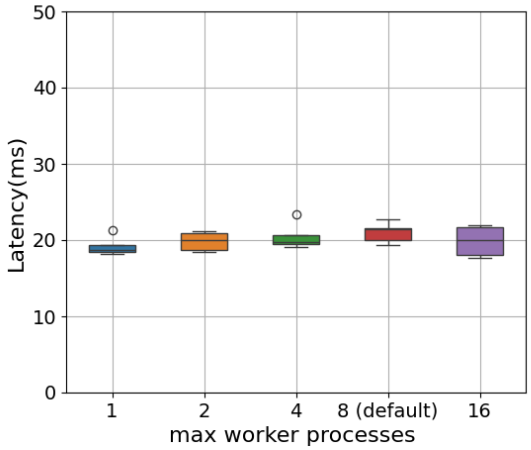
(a) Throughput effective cache size default



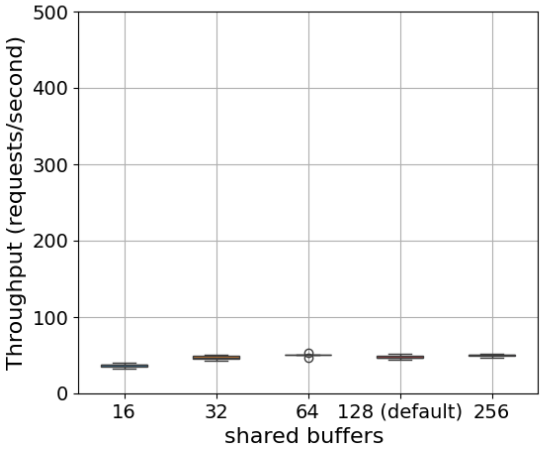
(b) Latency effective cache size default



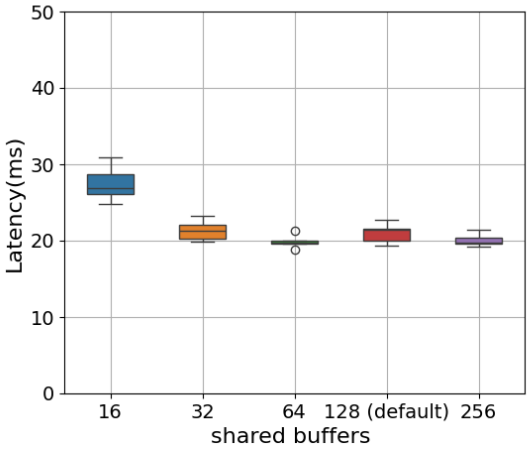
(c) Throughput max worker processes default



(d) Latency max worker processes default

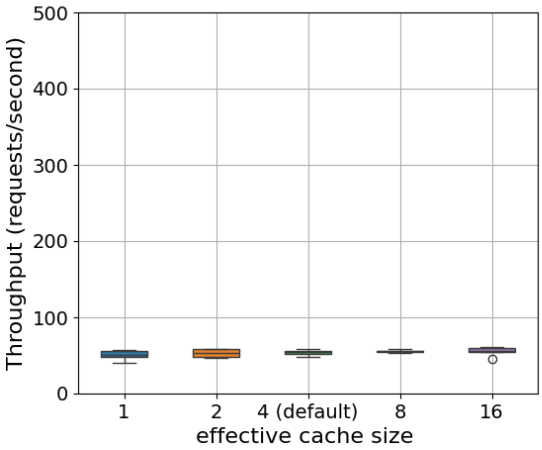


(e) Throughput shared buffers default

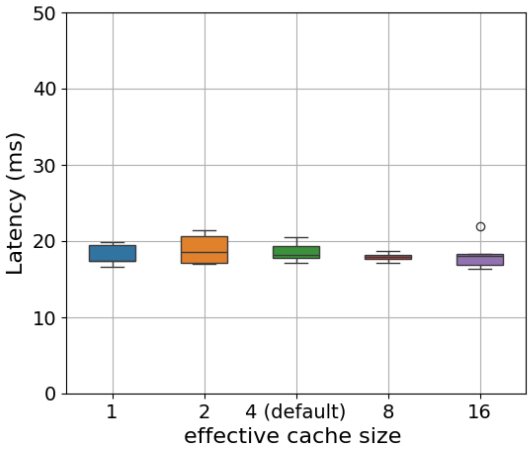


(f) Latency shared buffers default

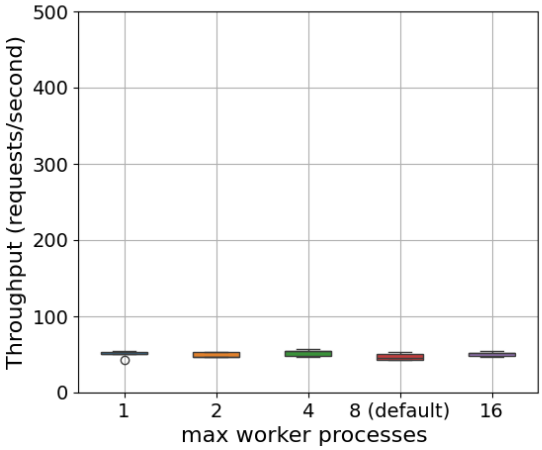
Abbildung 11: Ausführung der Benchmark Default mit der Einstellung Performance Tuning



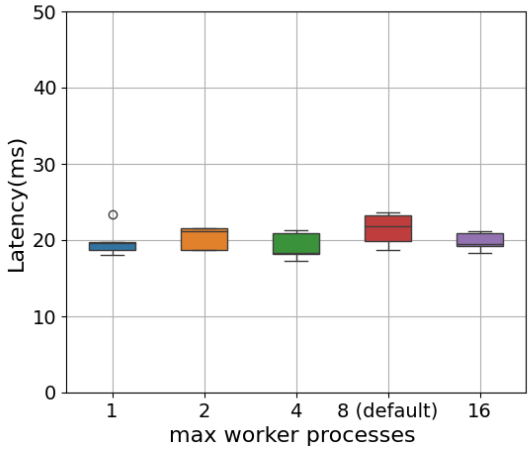
(a) Throughput effective cache size batchsize



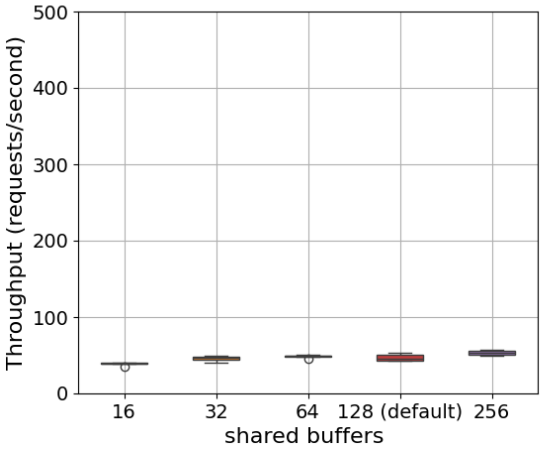
(b) Latency effective cache size batchsize



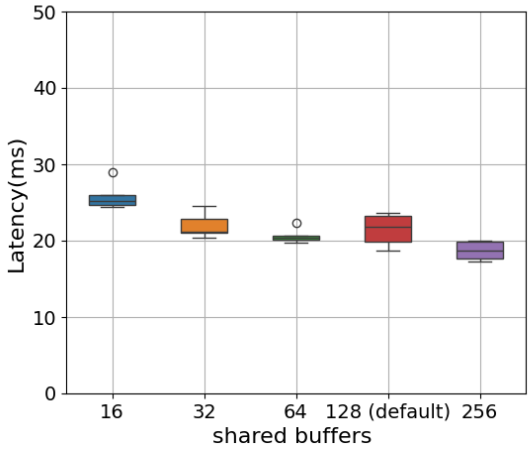
(c) Throughput max worker processes batchsize



(d) Latency max worker processes batchsize

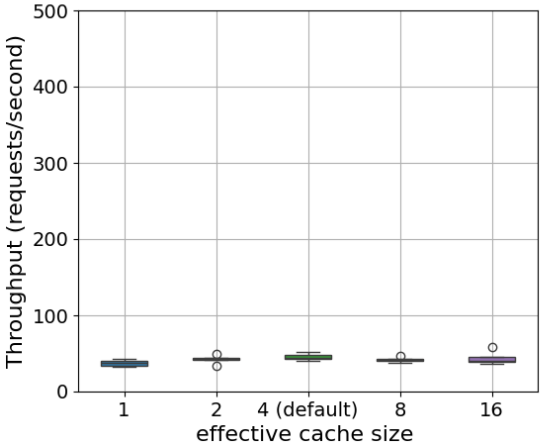


(e) Throughput shared buffers batchsize

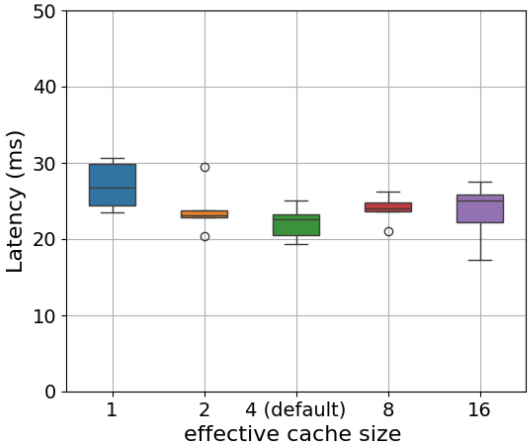


(f) Latency shared buffers batchsize

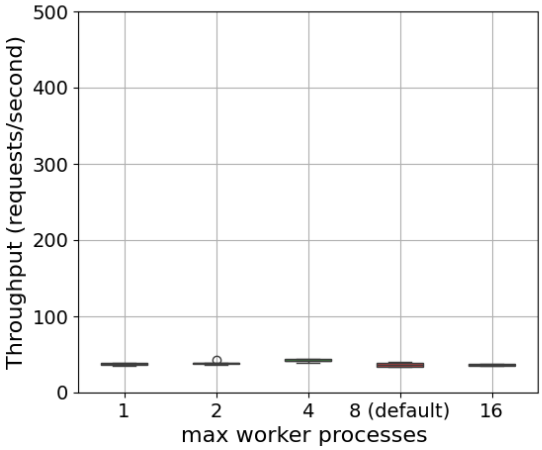
Abbildung 12: Ausführung der Benchmark Batchsize mit dem Wert 2 und mit der Einstellung Performance Tuning



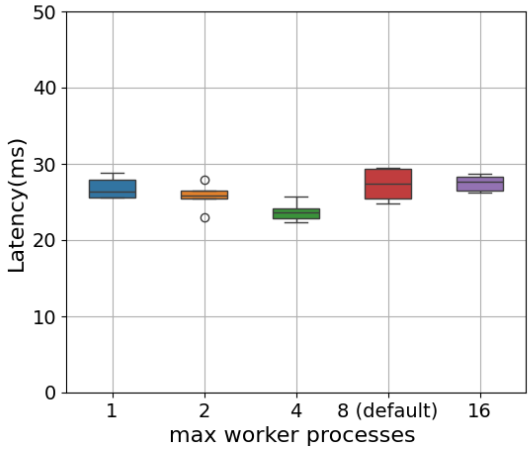
(a) Throughput effective cache size scalefactor



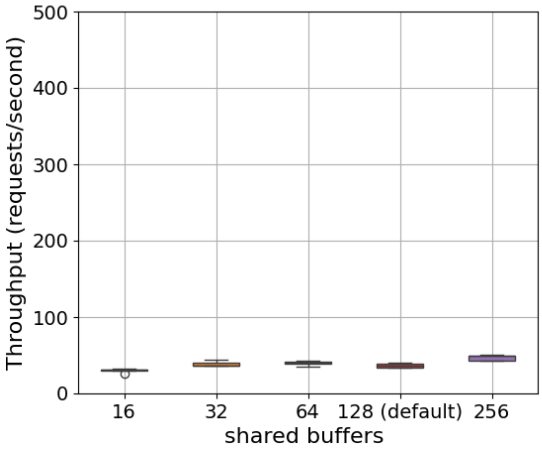
(b) Latency effective cache size scalefactor



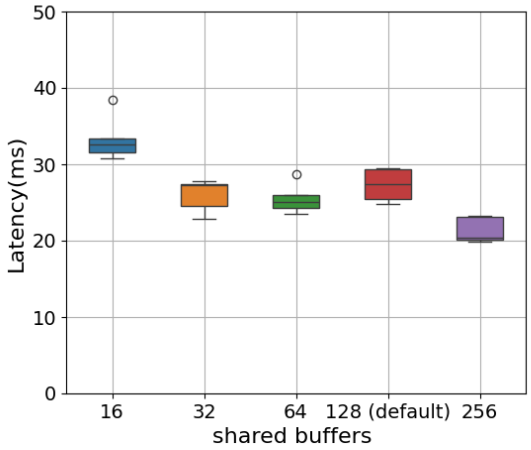
(c) Throughput max worker processes scalefactor



(d) Latency max worker processes scalefactor

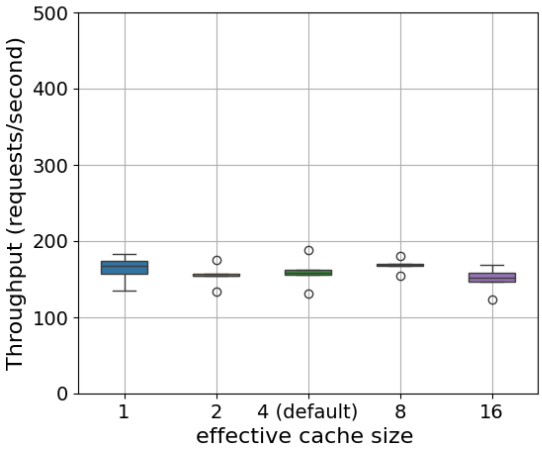


(e) Throughput shared buffers scalefactor

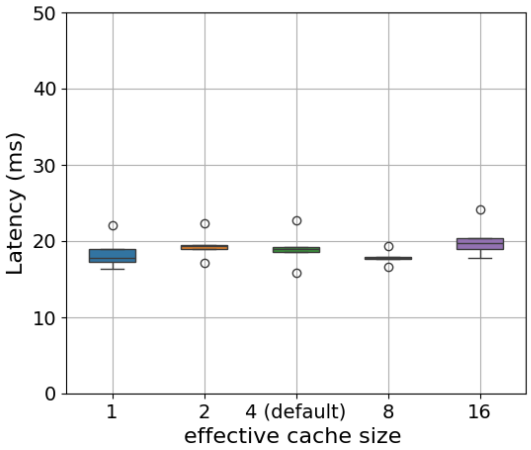


(f) Latency shared buffers scalefactor

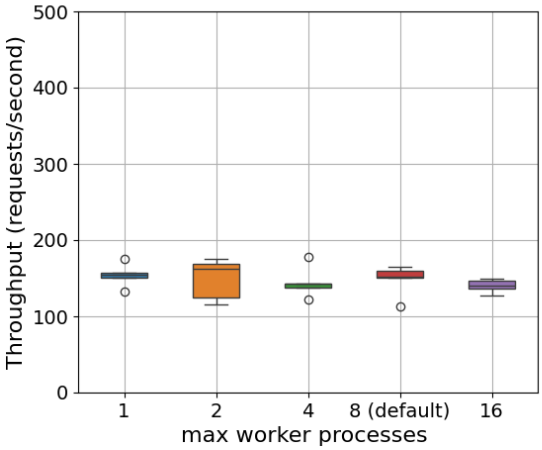
Abbildung 13: Ausführung der Benchmark scalefactor mit dem Wert 8 und mit der Einstellung Performance Tuning



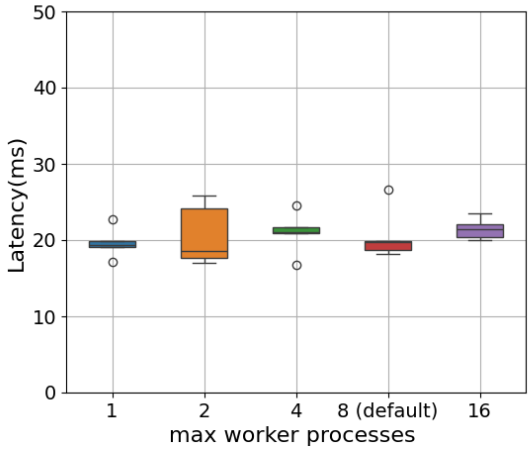
(a) Throughput effective cache size terminals



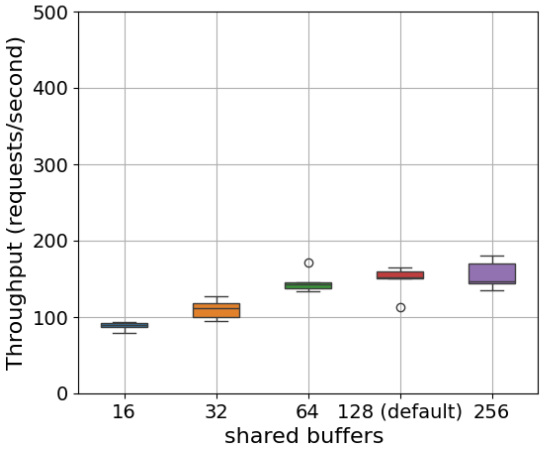
(b) Latency effective cache size terminals



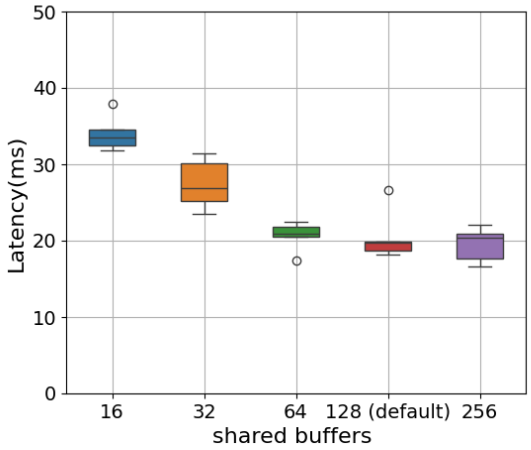
(c) Throughput max worker processes terminals



(d) Latency max worker processes terminals

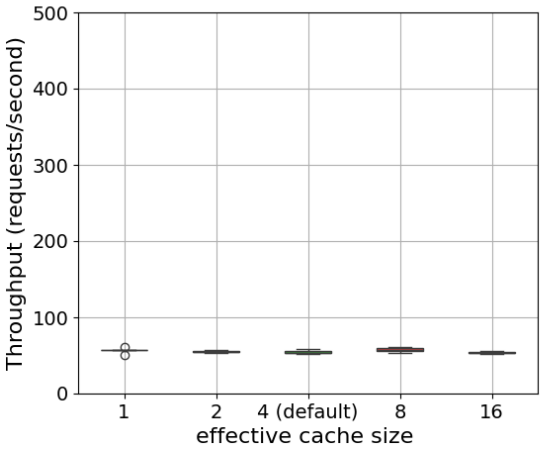


(e) Throughput shared buffers terminals

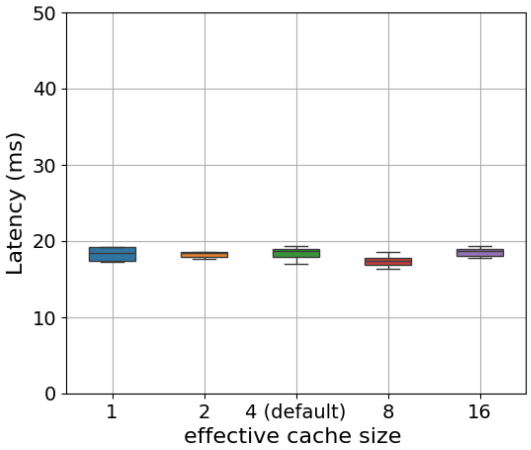


(f) Latency shared buffers terminals

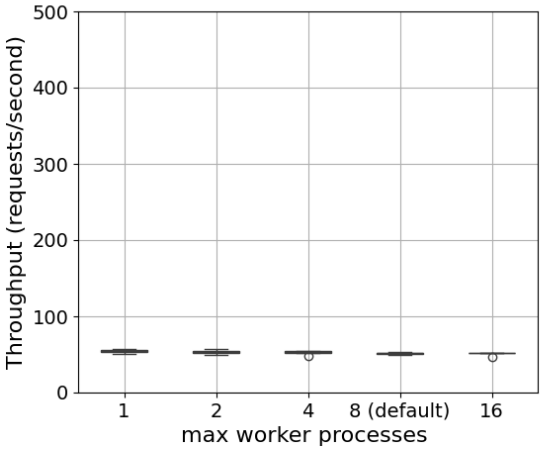
Abbildung 14: Ausführung der Benchmark terminal mit dem Wert 3 und mit der Einstellung Performance Tuning



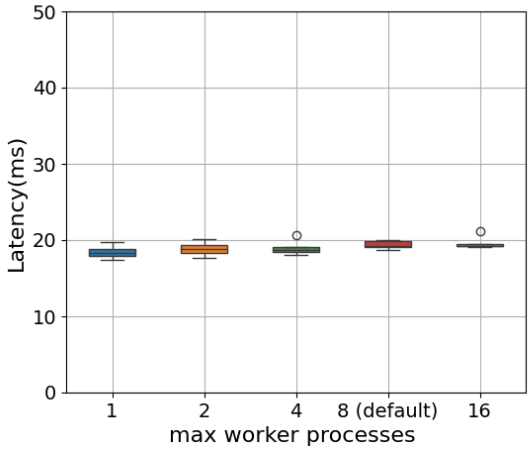
(a) Throughput effective cache size time



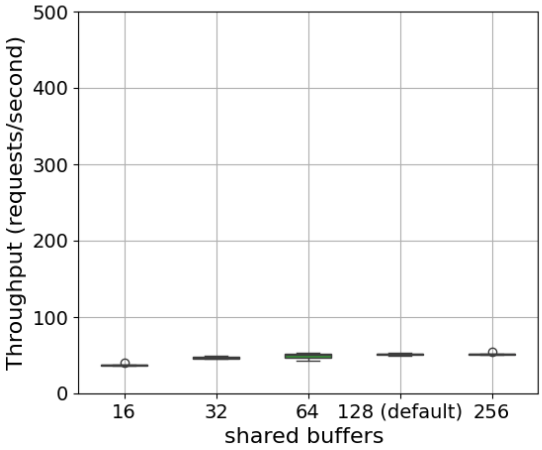
(b) Latency effective cache size time



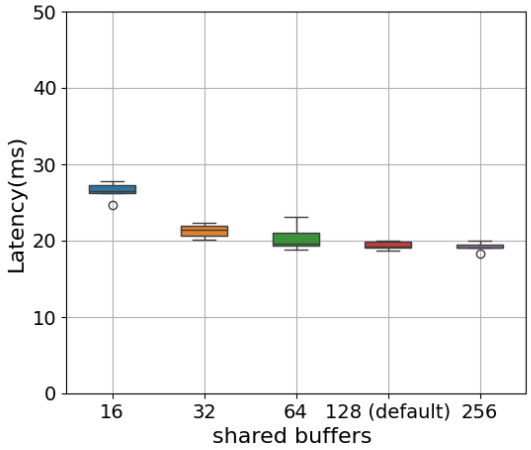
(c) Throughput max worker processes time



(d) Latency max worker processes time

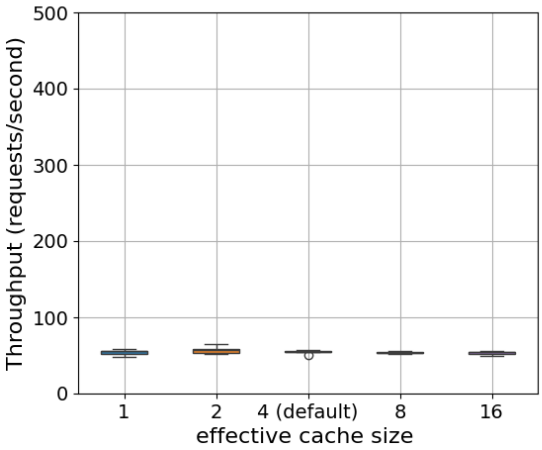


(e) Throughput shared buffers time

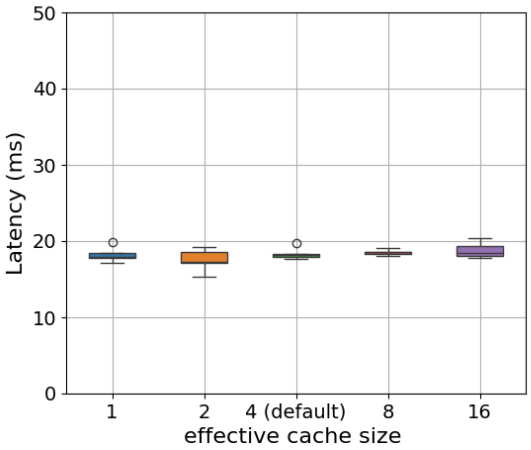


(f) Latency shared buffers time

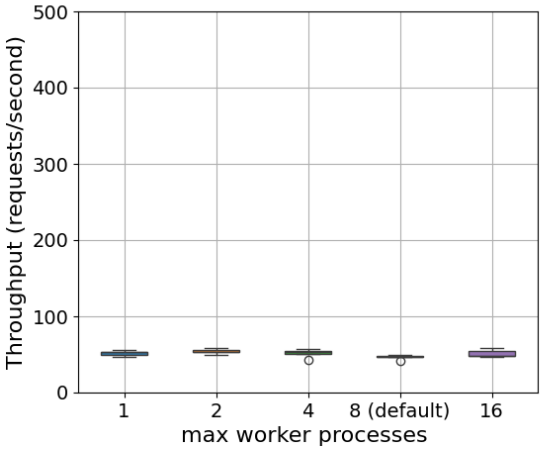
Abbildung 15: Ausführung der Benchmark time mit dem Wert 120 und mit der Einstellung Performance Tuning



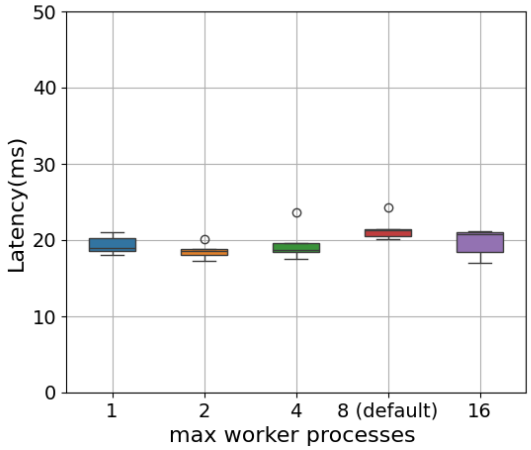
(a) Throughput effective cache size rate



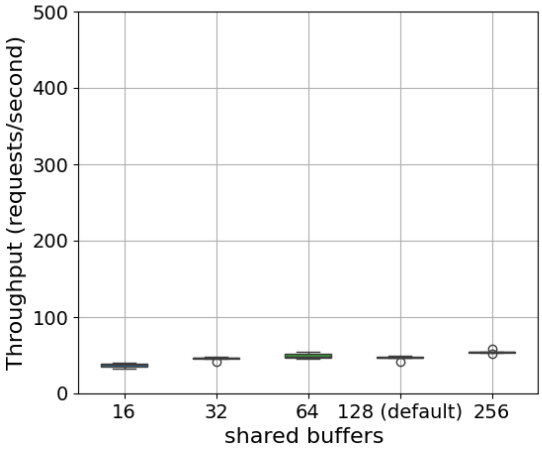
(b) Latency effective cache size rate



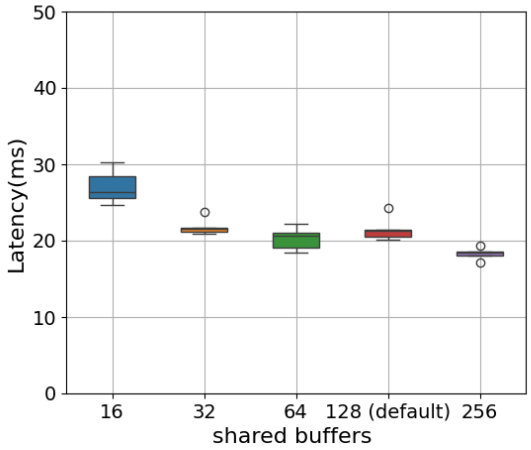
(c) Throughput max worker processes rate



(d) Latency max worker processes rate

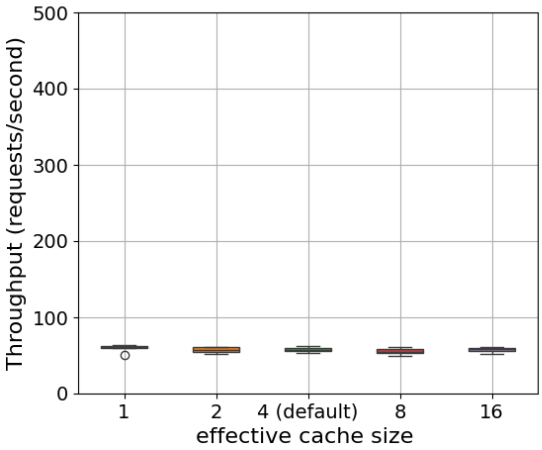


(e) Throughput shared buffers rate

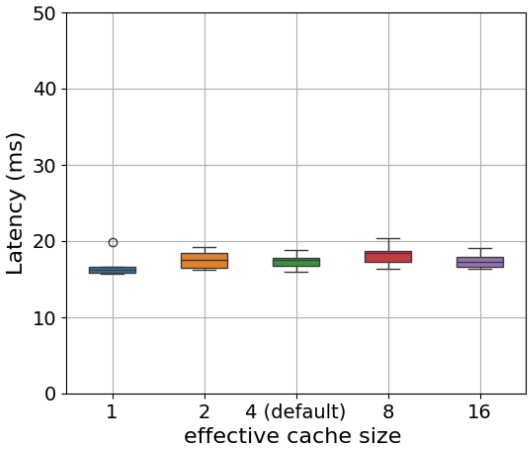


(f) Latency shared buffers rate

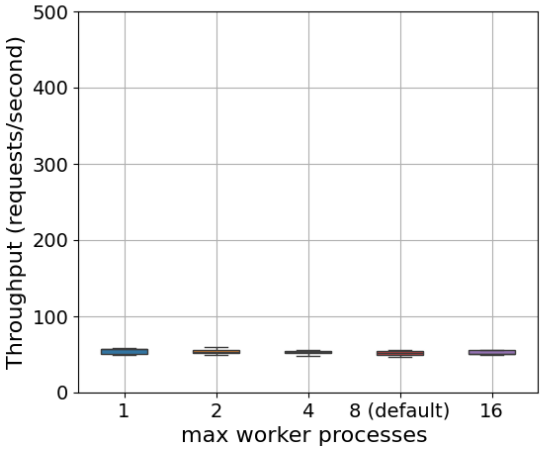
Abbildung 16: Ausführung der Benchmark rate mit dem Wert 40000 und mit der Einstellung Performance Tuning



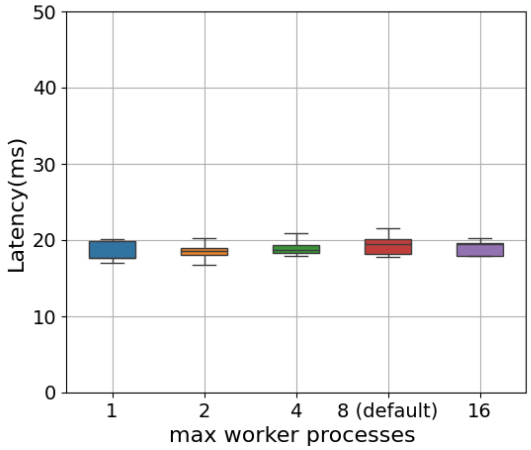
(a) Throughput effective cache size weights read write balanced



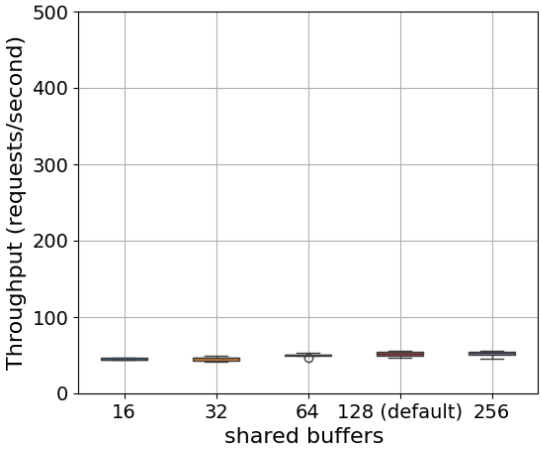
(b) Latency effective cache size weights read write balanced



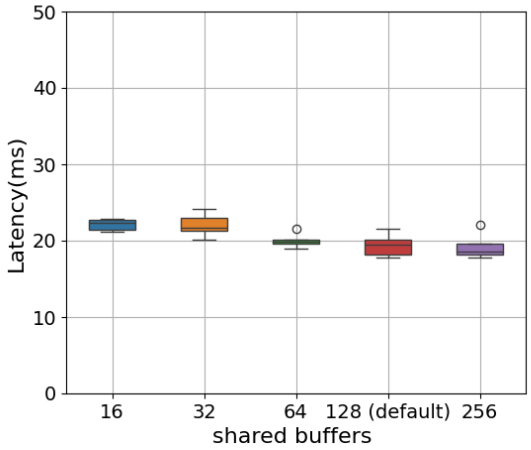
(c) Throughput max worker processes weights read write balanced



(d) Latency max worker processes weights read write balanced

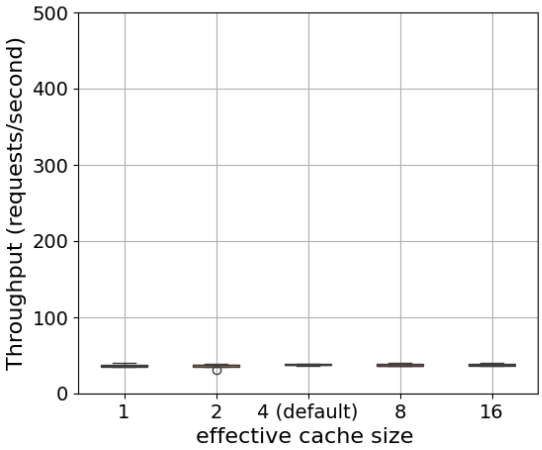


(e) Throughput shared buffers weights read write balanced

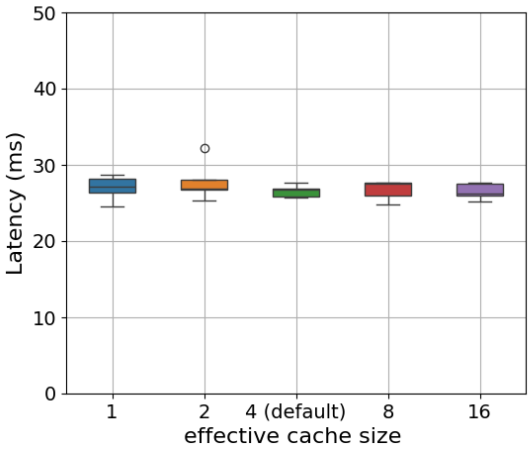


(f) Latency shared buffers weights read write balanced

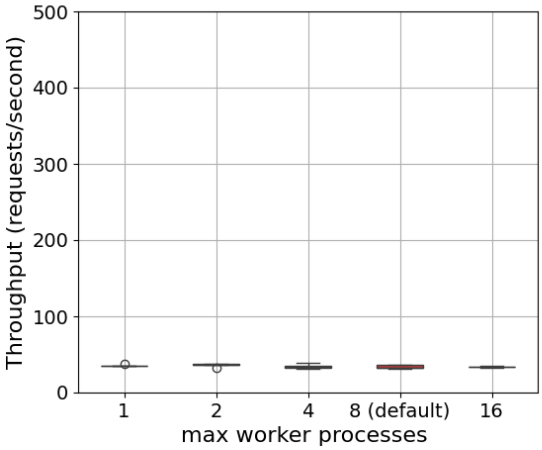
Abbildung 17: Ausführung der Benchmark weights read write balanced mit der Einstellung Performance Tuning



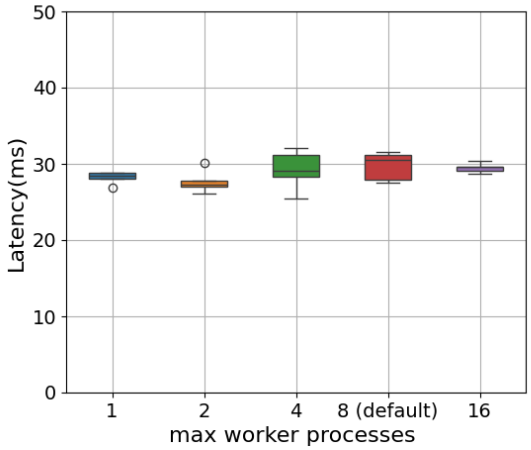
(a) Throughput effective cache size weights write mixed



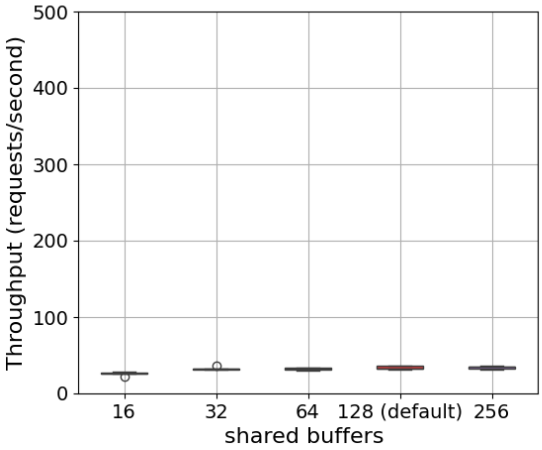
(b) Latency effective cache size weights write mixed



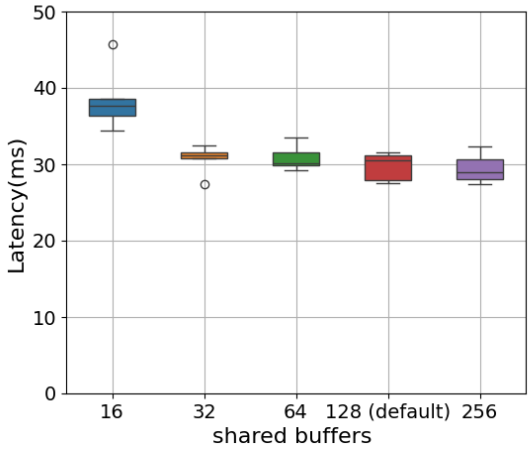
(c) Throughput max worker processes weights write mixed



(d) Latency max worker processes weights write mixed

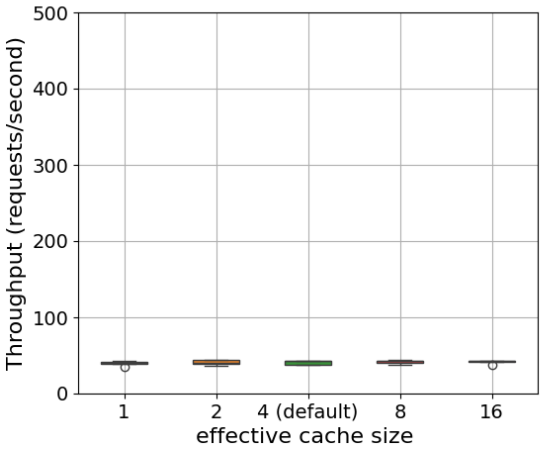


(e) Throughput shared buffers weights write mixed

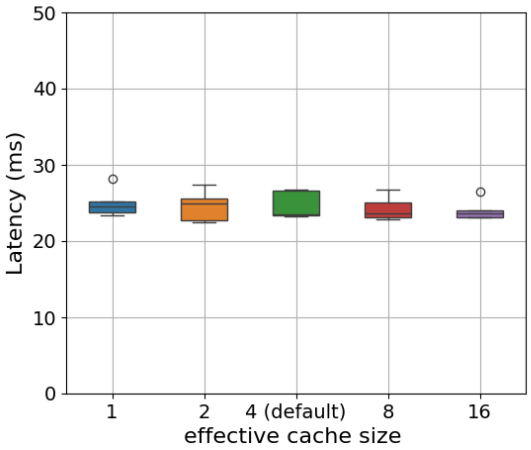


(f) Latency shared buffers weights write mixed

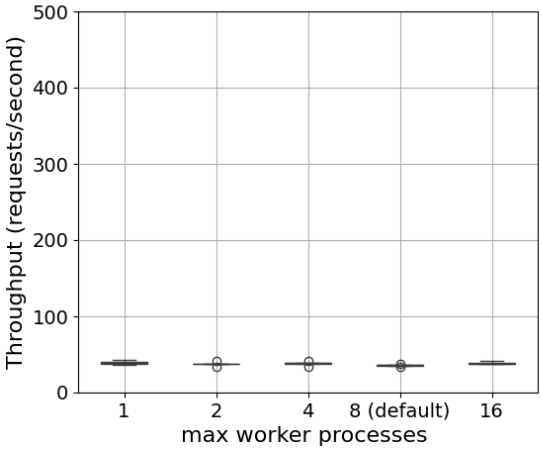
Abbildung 18: Ausführung der Benchmark weights write mix mit der Einstellung Performance Tuning



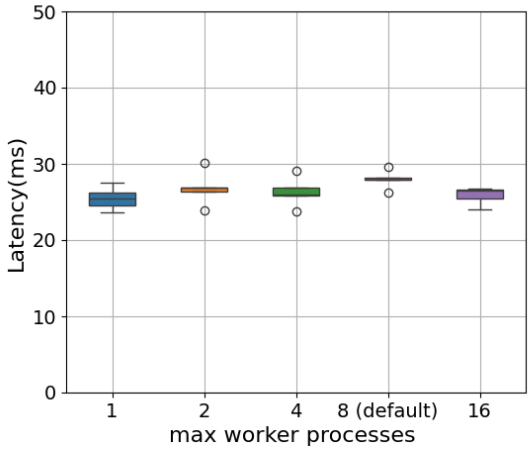
(a) Throughput effective cache size weights write new



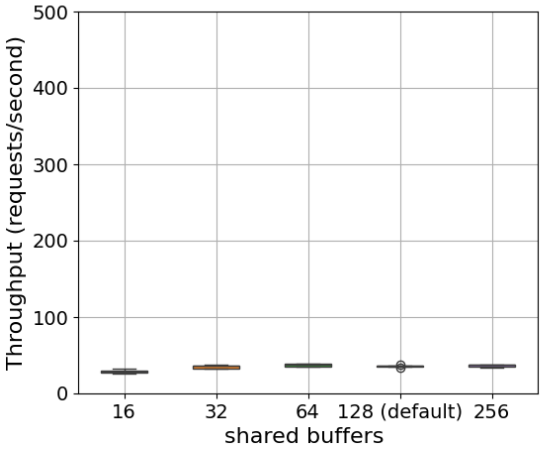
(b) Latency effective cache size weights write new



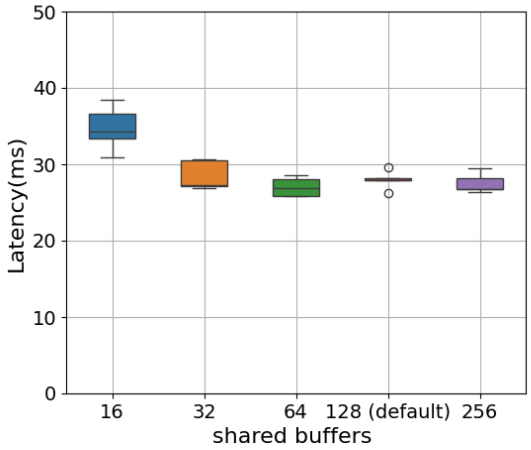
(c) Throughput max worker processes weights write new



(d) Latency max worker processes weights write new

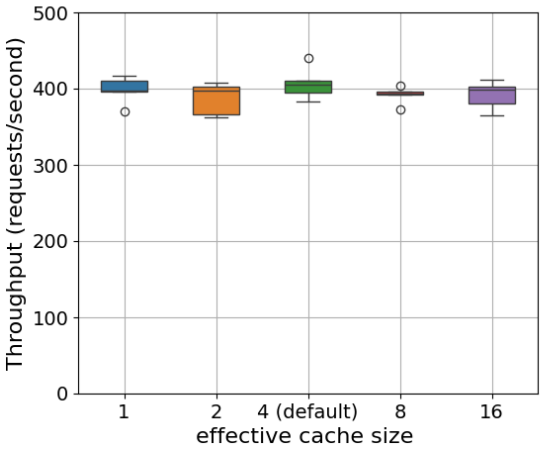


(e) Throughput shared buffers weights write new

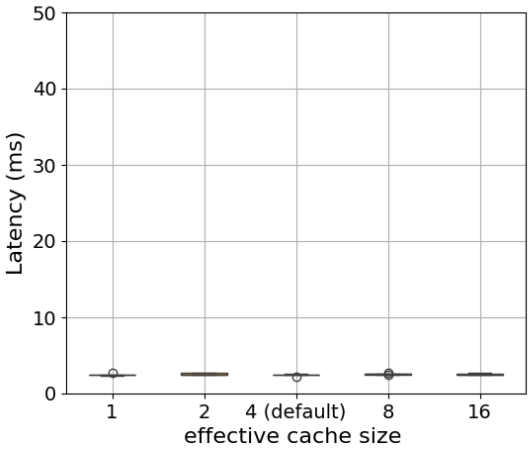


(f) Latency shared buffers weights write new

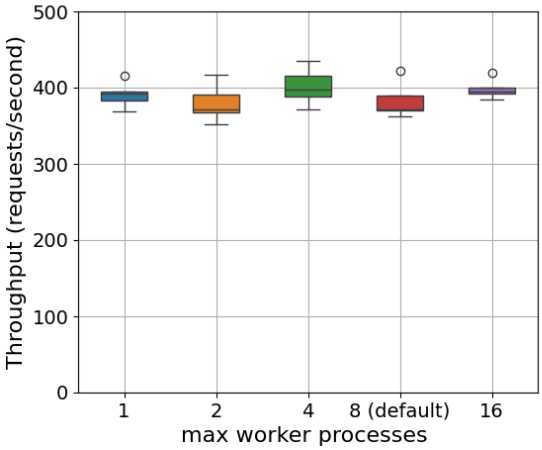
Abbildung 19: Ausführung der Benchmark weights write new mit der Einstellung Performance Tuning



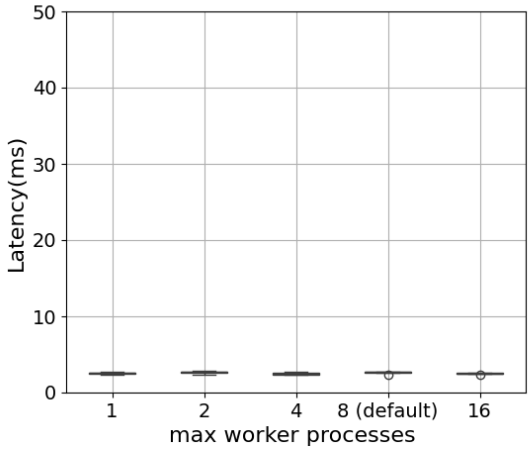
(a) Throughput effective cache size weights read heavy



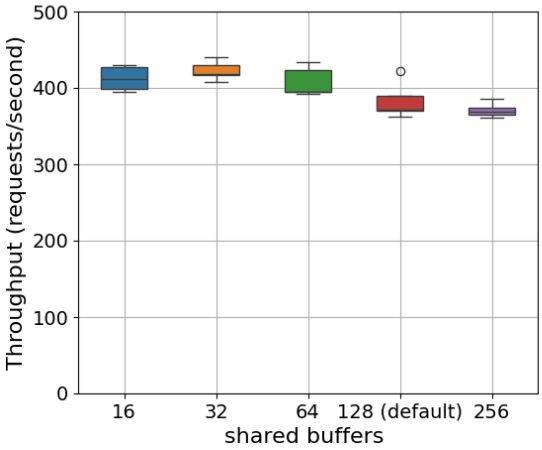
(b) Latency effective cache size weights read heavy



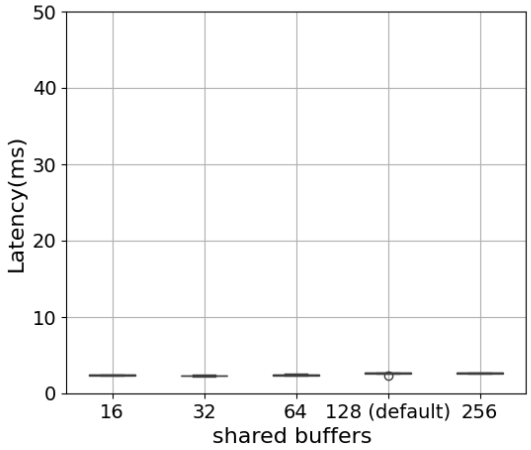
(c) Throughput max worker processes weights read heavy



(d) Latency max worker processes weights read heavy

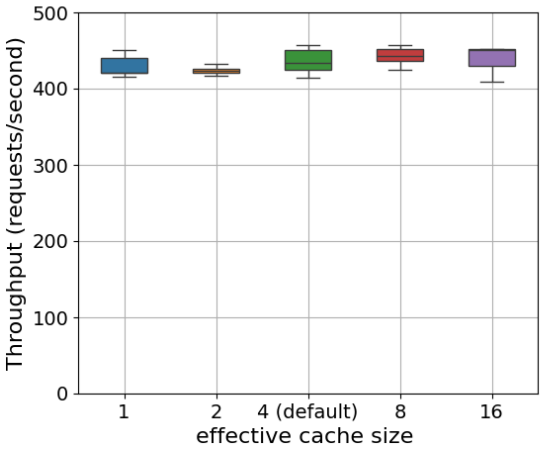


(e) Throughput shared buffers weights read heavy

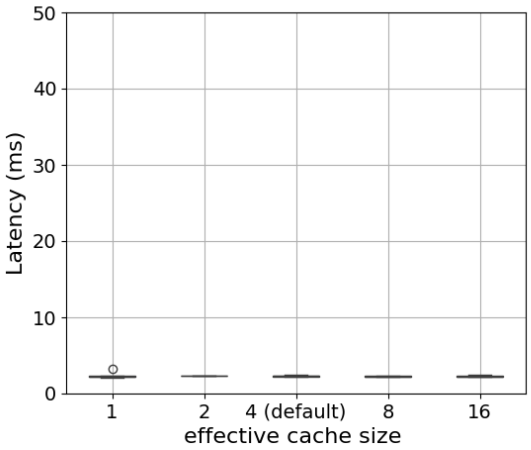


(f) Latency shared buffers weights read heavy

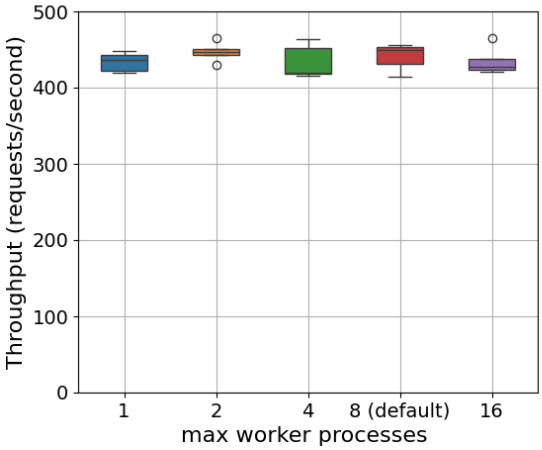
Abbildung 20: Ausführung der Benchmark weights read heavy mit der Einstellung Performance Tuning



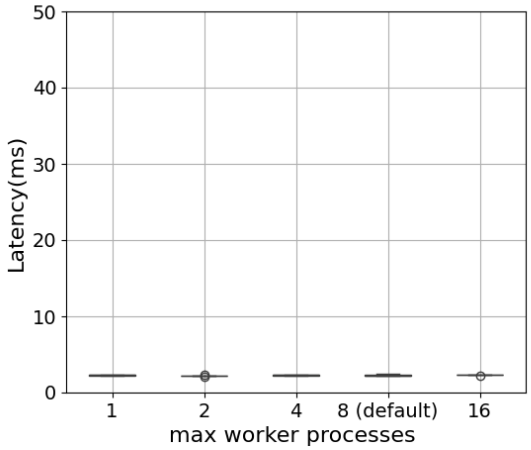
(a) Throughput effective cache size weights read mid



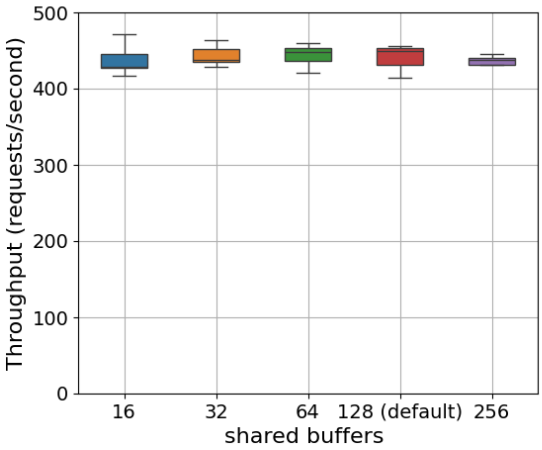
(b) Latency effective cache size weights read mid



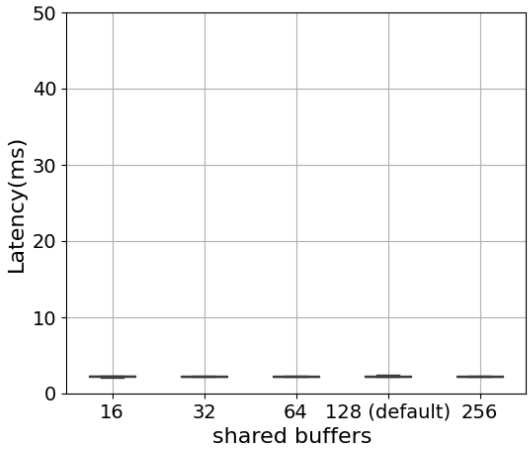
(c) Throughput max worker processes weights read mid



(d) Latency max worker processes weights read mid



(e) Throughput shared buffers weights read mid



(f) Latency shared buffers weights read mid

Abbildung 21: Ausführung der Benchmark weights read mid mit der Einstellung Performance Tuning

Abbildungsverzeichnis

1 Ausführung default mit der Einstellung Performance Tuning effective cache size . . . 9
2 Ausführung Default mit der Einstellung Performance Tuning shared buffers 10
3 Ausführung der Batchsize mit der Einstellung Performance Tuning effective cache size . . . 10
4 Ausführung Batchsize mit der Einstellung Performance Tuning shared buffers 11
5 Ausführung Terminals mit der Einstellung Performance Tuning effective cache size . . . 11
6 Ausführung Terminals mit der Einstellung Performance Tuning max worker processes . . . 12
7 Ausführung der Benchmark terminal der Einstellung Performance Tuning shared buffers . . . 13
8 Ausführung der Benchmark rate mit der Einstellung Performance Tuning shared buffers . . . 13
9 Ausführung der Benchmark weights write mix mit der Einstellung Performance Tuning max worker processes . . . 14
10 Ausführung der Benchmark weights write mix mit der Einstellung Performance Tuning shared buffers . . . 14
11 Ausführung der Benchmark Default mit der Einstellung Performance Tuning A2
12 Ausführung der Benchmark Batchsize mit dem Wert 2 und mit der Einstellung Performance Tuning . . . A3
13 Ausführung der Benchmark scalefactor mit dem Wert 8 und mit der Einstellung Performance Tuning . . . A4
14 Ausführung der Benchmark terminal mit dem Wert 3 und mit der Einstellung Performance Tuning . . . A5
15 Ausführung der Benchmark time mit dem Wert 120 und mit der Einstellung Performance Tuning . . . A6
16 Ausführung der Benchmark rate mit dem Wert 40000 und mit der Einstellung Performance Tuning . . . A7
17 Ausführung der Benchmark weights read write balanced mit der Einstellung Performance Tuning . . . A8
18 Ausführung der Benchmark weights write mix mit der Einstellung Performance Tuning . . . A9
19 Ausführung der Benchmark weights write new mit der Einstellung Performance Tuning . . . A10
20 Ausführung der Benchmark weights read heavy mit der Einstellung Performance Tuning . . . A11
21 Ausführung der Benchmark weights read mid mit der Einstellung Performance Tuning . . . A12