Institut für Informatik
Fakultät für Mathematik und Informatik
Abteilung Datenbanken

Ermittlung des stationären Zustands beim Festplattenbenchmarking

Exposé

vorgelegt von:
xxxx xxxxx

Matrikelnummer:
xxxxxxx

Betreuer:
Prof. Dr. Erhard Rahm
XXXXX

Contents

Abstract

Mit Benchmarks kann man die Performanz von Computersystemen, wie etwa die Rechenleistung und die I/O-Geschwindigkeit messen. Diese Messungen können stark oder schwach schwanken, je nachdem, ob ein Warm-Up erfolgte. Für die Ermittlung des stationären Zustandes sind geringe Schwankungen bei den Messungen notwendig. In dieser Arbeit wird mit dem Benchmark-Tool Fio gearbeitet, und es wird ein Analysetool entwickelt, das diesen Zustand ermittelt. Dabei wird statistische Analyse angewendet, um diesen stationären Zustand zu bestimmen. Das Programm wird in Java geschrieben.

1 Einführung

Die Performanz eines Rechners ist die Leistungsfähigkeit und Effizienz bei der Ausführung von Aufgaben [6]. Um herauszufinden welche Rechner die beste Performanz liefert, sind Messungen notwendig, um sie vergleichen zu können. Diese Messungen und Vergleiche werden mit Benchmarks durchgeführt [14]. Mit ihnen wird die Rechenleistung oder auch Lese-/Schreibgeschwingkeit der Hardware oder auch Software mit standardisierten Tests gemessen [5]. Ein Beispiel, bei dem Software getestet wird, sind virtuelle Maschinen wie die Java Virtual Machine [7] oder virtuelle Maschinen die auf reale Rechner basieren. Hier wird die Performanz von virtuellen CPUs oder des Hypervisor analysiert [10]. Auch für Cloud Computing Platformen wird Benchmarking betrieben, um auch Performanz zwischen verschieden Cloudplatformen vergleichen zu können. Diese Benchmarks verwenden z.B die Turnaround-Zeit als Metrik, also die Zeit, die ein Prozess benötigt, um vollständig durchzulaufen [11]. Somit gibt es ein weites Spektrum an Auswahl von Systemen, wo Benchmarking betrieben wird. Solche Grenzen könnten die Hardware (z. B. CPU-Taktrate oder Bandbreitengeschwindigkeit beim Lesen oder Schreiben), aber auch Software (Virtuelle Maschinen) sein. Und mit der Verbesserung von solchen Speichermedien, wie HDD zu SSD, und CPU-Leistung wird die I/O Geschwindigkeit immer relevanter. Studien haben schon bewiesen, dass dies eine Ursache für Bottlenecks in der Performanz ist [16].

In Großkonzernen sind oft Performanzprobleme die eigentlichen Hindernisse, und nicht die funktionalen Probleme. Serviceausfälle sind höchst kostspielig und sollten minimiert werden [2].

Ein Beispielprogramm für so ein Benchmark Tool ist das Fio (flexible I/O tester) [3]. Fio ist ein Tool, mit dem man die Bandbreitengeschwindigkeit vom Lesen/Schreiben testet und sie als eine Textdatei (von fio Logdatei genannt) ausgeben kann. Diese Geschwindigkeit kann stärker oder schwächer vom eigentlichen WERT schwanken. Das Ziel meines Tools ist es, den stationären Zustand zu ermitteln. Ein Zustand, wo die Rechenleistung oder Lese-/Schreibgeschwindigkeit sich nicht mehr beim Messen stark ändert [4]. Um den stationären Zustand zu ermitteln, werden verschiedene Methodiken aus der Varianzanalyse verwendet. Der Tukey-HSD-Test, t-Test und Mann-Whitney-Test werdem zusammen mit den Werten aus den Logdateien des fios genutzt.

2 Grundlagen

2.1 Festplattenbenchmarking

Das Fio Tool ermöglicht das Testen auf bestimmter Hardware oder Software. Die I/O-Geschwindigkeit wird hier in Mebibyte pro Sekunde angegeben. Das Fio bietet auch die Möglichkeit diese Einheit zu ändern. Das Programm selbst besitzt kein GUI, sondern arbeitet nur in der Konsole. Man kann für das Programm sogenannte .fio Dateien schreiben. Oder man arbeitet direkt in der Kon-

sole, um seine gewünschten Tests durchzuführen. Wenn man nun ein Random Read testen möchte, kann die .fio Datei wie folgt aussehen (Listing 2):

```
1  ; fio-rand-read.job for fio Test
2
3  [global]
4
5  name=rand-read # Name des Jobs
6  rw=randread # Was soll der Job testen, randread = random read
7  runtime=2s # Wie lange soll der Job laufen
8  size=128m # Groesse der Datei, m fuer Megabyte
9  write_bw_log=mytest # Name der Log-Datei
```

Listing 1: Beispiel für eine .fio Datei (# heißt Kommentar)

In der Konsole kann man sie auch einzeln als Parameter angeben, wenn es nur ein Test sein soll. Diese Tests, oder auch Jobs genannt, geben Logs aus, mit denen mein Tool arbeiten wird. Dieser Job oben testet das Random Read mit einer Laufzeit (Runtime) von 2 Sekunden und liest eine Datei mit einer Größe von 128 Mebibyte.

2.2 Stationärer Zustand

Bei Hardware oder Software ist anfangs wiederholtes testen notwendig, um sie analysieren zu können. Erst nach diesen *Warm-ups* wird der "steady state of peak performance" erreicht. Ein Beispiel, wo solche Warmups erforderlich sind, ist die JIT (Just-In-Time) Kompilierung in Java. Der Bytecode wird während der Laufzeit des Programms kompiliert. Damit verbessert man die Performanz von Java-Anwendungen [9]. Doch erst nach diesen Warm-Ups ist es möglich den stationären Zustand zu ermitteln, da die Performanz weniger schwankt [4]. Der stationäre Zustand ist der Zeitpunkt, wenn es keine starken Schwankungen mehr bei der Bandbreitengeschwindigkeit im Lesen oder im Schreiben gibt. Auch wenn der Warm-up erfolgt ist, wird Der Nichtdeterminismus dabei ein Hindernis sein. Auch wenn man das Lesen/Schreiben auf demselben System mit derselben Datei testet, ist die Bandbreitengeschwindigkeit im Durchschnitt nie die gleiche. Sie wird immer abweichen. Ursachen dafür könnten schon verschiedene CPU-Temperaturen sein, aber auch CPU-Scaling oder parallele Prozessierung. [8]. Das fio selbst arbeitet nicht nur mit einem Thread, sondern es arbeitet mit Multithreads, was nicht-deterministische Zustände hervorrufen kann. Da sich die Geschwindigkeit nie konstant einem Wert nähert, sondern stets abweicht, soll mein Programm in Zukunft mit statistischen Tests den stationären Zustand ermitteln und dabei den Nichtdeterminismus berücksichtigen. Solche Abweichungen können in wenigen Prozentbereichen liegen. Es wird aber nicht ausreichen, ein paar Tests durchzuführen und danach die Logs davon auszuwerten. Diese Logs wären nämlich nicht akkurat genug. Ein größeres Problem wird es sein - wenn der Warmup durchgelaufen ist - welche Logs die besten Informationen besitzen. Die Auswahl der Logs und deren Auswertung könnten signifikante Unterschiede beinhalten, die die Evaluierung verändern könnten [2].

2.3 Statistische Tests

Eine Möglichkeit, den stationären Zustand zu ermitteln, ist, dass man die Standardabweichung verwendet und errechnet, wann die Abweichung klein genug ist. Zusätzlich werden Konfidenzintervalle benutzt, um den Nichtdeterminismus mitzubeachten. Eine weitere Methode, die durchgeführt wird, ist die Verwendung der Varianzanalyse - *Analysis of Variance* (ANOVA) [8]. Da der stationare Zustand nicht eindeutig bestimmt werden kann, sollen diese statistischen Methoden dabei helfen. Mit ANOVA wird untersucht, ob die berechneten Mittelwerten aus den verschiedenen Logdaten statistisch signifikant sind. Mit statistisch signifikant ist hier gemeint, ob zwischen den durchgelaufenen Tests/Jobs des Fios eine Abhängigkeit besteht [15]. Wenn das nicht der Fall ist, sind die Jobs nicht statistisch signifikant [13]. Dies kann mit einem Signifikanzniveau festgelegt werden, das die maximale Wahrscheinlichkeit angibt, mit der die statistische Signifikanz fälschlicherweise angenommen wurde.

Dabei sind Falsch-Positive-Annahmen (Typ-I-Fehler) zu berücksichtigen. Die Analysen liefern nicht immer eine verlässliche Aussage darüber, ob der stationäre Zustand tatsächlich erreicht wurde oder ob es sich um eine falsche Annahme handelt.

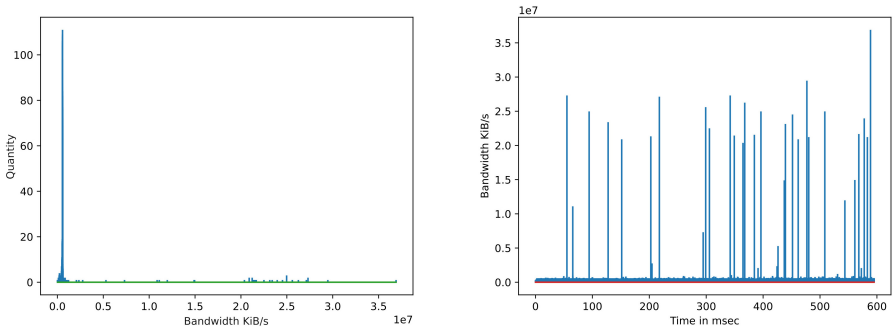
In der Arbeit werden der t-Test, der Tukey-HSD-Test und der Mann-Whitney-Test genutzt. Der Tukey-HSD-Test verwendet die Differenz zwischen zwei Mittelwerten, um ihre statistische Signifkanz zu ermitteln. Der HSD-Wert wird mittels mittlere quadratische Abweichung (MSE) berechnet und mit der Anzahl der Gruppen die getestet wurden. Die minimalste Differenz wird als HSD bezeichnet. Wenn aber die Differenz der Mittelwerte aus den Tests größer als der HSD-Wert ist, sind diese Werte statistisch signifikant. Zusätzlich versucht der Tukey HSD Test die Fehlerrate von Falsch-Positiven-Annahmen zu korrigieren. Die t-Tests arbeiten ähnlich wie die Tukey HSD Tests. Der Unterschied besteht darin, dass sie die Fehlerrate nicht anpassen [12] [1]. Beim Mann-Whitney-Test die werden zubestimmenden Werte einem Rang zugeordnet und einer Gruppe zugeordnet. Danach wird statistische Signifilanz nach den Rängen getestet und nicht nach Mittelwerrten.

3 Konzept

Das fio Tool bekommt den Parameter $–write\_bw\_logs = [Dateiname]$ dazu, um die Logs als Datei auszugeben. Diese Logs geben den kompletten Verlauf des Jobs wieder. Die ersten Zeilen des Logs könnten so aussehen, wie in Listing 2:

```
1    [Time, Bandwidth, data direction, Blocksize, Offset]
2    0, 59782, 0, 4096, 0
3    0, 54353, 0, 4096, 0
4    1, 45545, 0, 4096, 0
```

Listing 2: Erste Zeilen des Logs (Bezeichnungen sind nicht im Log enthalten)



(a) Bandwidth Frequency

(b) Bandwidth per Msec

Figure 1: fio Job: randread Bandbreite und Frequenz/Quantität

Time für die Zeit in Millisekunden (ms) die verlaufen ist, Bandwidth für die Bandbreitengeschwindigkeit in Kibibyte/s, der dritte Wert für die data direction, ob gelesen (= 0) oder geschrieben (= 1) wurde und ein Blocksize und ein Offset. Die Logdaten selbst sind immens lang und nicht schön lesbar. Deshalb wurde schon ein kleines Tool in Python programmiert, was diese Logdaten zu einem Graph umwandelt. Das Tool nimmt die Zeit als X-Achse und die Bandbreitengeschwindigkeit als Y-Achse. Wenn man sich die Logdaten als Text nochmal anschaut, sieht man, dass die Bandbreitengeschwindigkeit sich schneller innerhalb einer 1 ms ändert. Das Python-Tool umgeht das Problem so, dass die Werte, die mehrfach doppelt vorkommen, linear auftrennt. Unten wird es nochmal veranschaulicht (Listing 3), wenn die Zeit sich dreimal wiederholt, mit 23ms.

```
1     Drei Zeilen aus der Logdatei
2     [Time, Bandwidth, data direction, Blocksize, Offset]
3     23,      59782,    0,            4096,    0
4     23,      43534,    0,            4096,    0
5     23,      54364,    0,            4096,    0
6
7     Erste beide Spalten     ->       X-Y-Koordinaten
8     X: 23      Y: 59782     ->     X: 23.0     Y: 59782
9     X: 23      Y: 43534     ->     X: 23.3     Y: 43534
10    X: 23      Y: 54364     ->     X: 23.6     Y: 54364
```

Listing 3: Aufsplitten der Zeit um ein Grafik zu bilden

In Figure 1 ist einmal die Bandbreitengeschwindigkeit pro Millisekunde dargestellt (a) und in der anderen Abbildung (b) zeigt es die Häufigkeiten, wie oft diese Baandbreitengeschwindigkeit erreicht wurde. Das fio wurde wie oben in Listing 1 ausgeführt.

Der Computer, der verwendet wurde, hat ein 11th Gen Intel(R) Core(TM) i5-11400 @ 2.60GHz Prozessor und eine ADATA SWORDFISH 1TB SSD als

Datenträger. Das kleine Tool diente mehr zum Einstieg. Die eigentliche Arbeit wird es sein, das Tool weiter auszubauen, das nicht nur veranschaulicht, sondern auch den stationären Zustand berechnet. Das Programm wird in Java geschrieben, da bereits entsprechende Vorkenntnisse vorhanden sind. So soll mein Tool mit den Logdateien statistisch arbeiten.

4 Verwandte Arbeiten

In der Arbeit von Barrett et al [4]. wurde tiefer auf Nichtdeterminismus von VMs eingegangen. Dort wurde versucht, nicht-determinisitsche Faktoren so gering wie möglich zu halten, da sonst der stationäre Zustand nicht eintreten kann, auch wenn gleiche Workloads wiederholt durchlaufen werden. Auch andere Hardware und Betriebssysteme haben wenig Einfluss auf die Zeit von Warmups. Li et al. [10] arbeiteten ebenfalls mit VMs und testen die Performanz von diesen Maschinen mit verschiedener Anzahl an virtuellen CPUs. Und eine große Anzahl an VCPUs verbessert nicht zwingend die Performanz. Georges et al [7]. verwendet statistische Methoden, wie Varianzanalyse und Tukey HSD, um eine akkurate Auswertung von den Experimenten berechnen zu können. Sie arbeiten ebenfalls mit VMs und VM Warmups. Gründe für nicht-deterministisches Verhalten in VMs könnte Garbage Collection, Thread Scheduling oder auch Just-in-Time Compilation/Optimization sein [8]. AlGhmadi et al. erarbeiteten, wann Performanztests nach langem Testen redundant werden und wie viele Tests genug sind.

5 Zeitplan

Im ersten Teil der eigentlichen Arbeit werden die Grundlagen behandelt. Zuerst werden auf die Begrifflichkeiten wie Benchmarking, Warm-Up, stationärer Zustand und fio-Tool erklärt. Nach den Erklärungen werden die Methodiken der statististischen Analyse definiert, die das Analysetool verwenden wird. Im nächsten Kapitel werden die verwandten Arbeit beschrieben, die in der Arbeit verwendet wurden. Im dritten Kapitel wird der Aufbau sowie die Verwendung des Analystools eingegangen. Nach dem Analysetool, werden die Experimente durchgeführt. Diese werden im vierten Kapitel für die Evaluierung verwendet. Das letzte Kapitel ist die Zusammenfassung und Ausblick.

References

[1] Tukey's range test. https://en.wikipedia.org/wiki/Tukey%27s_range_test. Zugriff am 11. November 2024.

[2] Hammam M. Alghmadi, Mark D. Syer, Weiyi Shang, and Ahmed E. Hassan. An automated approach for recommending when to stop performance tests. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 279–289, 2016.

[3] Jens Axboe. Github—axboe/fio: Flexible i/o tester. *Retrieved Nov*, 10:2022, 2021.

[4] Edd Barrett, Carl Friedrich Bolz-Tereick, Rebecca Killick, Sarah Mount, and Laurence Tratt. Virtual machine warmup blows hot and cold. *Proc. ACM Program. Lang.*, 1(OOPSLA), October 2017.

[5] Isabelle Bruno. *Benchmarking*, pages 363–368. Springer Netherlands, Dordrecht, 2014.

[6] Delst. Definition performance. https://www.delst.de/de/lexikon/performance/. Zugriff am 12. November 2024.

[7] Andy Georges, Dries Buytaert, and Lieven Eeckhout. Statistically rigorous java performance evaluation. In *Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications*, OOPSLA '07, page 57–76, New York, NY, USA, 2007. Association for Computing Machinery.

[8] Andy Georges, Dries Buytaert, and Lieven Eeckhout. Statistically rigorous java performance evaluation. *SIGPLAN Not.*, 42(10):57–76, October 2007.

[9] IBM. The jit compiler. https://www.ibm.com/docs/en/sdk-java-technology/8?topic=reference-jit-compiler. Zugriff am 19. November 2024.

[10] Yuegang Li, Dongyang Ou, Congfeng Jiang, Jing Shen, Shuangshuang Guo, Yin Liu, and Linlin Tang. Virtual machine performance analysis and prediction. In *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*, pages 1–5, 2020.

[11] Aniruddha Marathe, Rachel Harris, David K. Lowenthal, Bronis R. de Supinski, Barry Rountree, Martin Schulz, and Xin Yuan. A comparative study of high-performance computing on the cloud. In *Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '13, page 239–250, New York, NY, USA, 2013. Association for Computing Machinery.

[12] Novustat. Mittelwertvergleich nach anova. https://schmidtpaul.github.io/crashcouRse/appendix_posthoc.html. Zugriff am 12. November 2024.

[13] Novustat. Post hoc test: Signifikante unterschiede finden und erklären. https://novustat.com/statistik-blog/post-hoc-test-signifikante-unterschiede-finden-und-erklaeren.html. Zugriff am 11. November 2024.

[14] takevalue Consulting GmbH. Definition benchmarking. https://www.takevalue.de/glossar/benchmark/. Zugriff am 12. November 2024.

[15] wikipedia. Statistische signifikanz. https://de.wikipedia.org/wiki/Statistische_Signifikanz. Zugriff am 19. November 2024.

[16] Qiumin Xu, Huzefa Siyamwala, Mrinmoy Ghosh, Tameesh Suri, Manu Awasthi, Zvika Guz, Anahita Shayesteh, and Vijay Balakrishnan. Performance analysis of nvme ssds and their implication on real world databases. In *Proceedings of the 8th ACM International Systems and Storage Conference*, SYSTOR '15, New York, NY, USA, 2015. Association for Computing Machinery.