Contents

1 Einführung

Performanz spielt eine große Rolle in der Entwicklung von Software und Hardware. Die größten Probleme in den Großkonzernen sind Performanzprobleme und nicht die funktionellen Probleme. Service Ausfälle kosten sind höchst kostspielig. Deshalb werden in High Performance Computing viele Tests durchgeführt, um ihre Grenzen zu analysieren [1]. In meiner Arbeit werde ich mit dem fio Tool arbeiten, um solche Performanzänderungen zu berechnen und zu analysieren. Fio ist ein Programm mit der man die Hardware-Performanz testen kann, z.B die Brandweite beim Schreiben einer Datei, und die Performanz als log Daten ausgeben kann. Mein Programm wird mit diesen Logs arbeiten, um sie besser darzustellen und arbeiten zu können. Ziel meines Tools soll es sein, dass man den stationären Zustand analysiert.

2 Grundlagen

Mit dem fio (flexible Input/Output) Programm, kann man die Performanz auf einer bestimmten Hardware testen . fio besitzt kein GUI sondern arbeitet nur in der Konsole. Man kann für das Programm fio-Dateien schreiben oder auch in der Konsole arbeiten, um seine gewünschten Task zu testen, wie random read/write auf einer Datei. Wenn man ein Random Read testen möchte, kann die fio-Datei so aussehen.

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

In der Konsole gebt man sie als Parameter an, wenn es nur ein schneller Test sein soll. Zusätzlich können die Tests, oder auch Jobs genannt, Logs ausgeben. Mit diesen Logs wird mein Tool arbeiten. Dieser Job oben testet das Random-Read mit einer runtime von 2 Sekunden. Nun möchte man wissen wann der stationäre Zustand erreicht wurde, wenn man solche Tests länger durchlaufen lässt. Der stationäre Zustand ist der erreichte Zeitpunkt, wenn es keine starken Schwankungen mehr bei der Brandweite im Lesen und Schreiben gibt. Das Schreiben und Lesen verhält sich außerdem nicht deterministisch. Das bedeutet, auch wenn man die selbe Datei mit dem Selben Computer lesen/schreiben würde, wäre die Brandweitengeschwindigkeit immer im Durchschnitt unterschiedlich. Weitere nicht deterministische Zustände könnte sogar CPU Temperatur sein, oder CPU Scaling, parallele Prozessierung.

3 Konzept

Mein Tool soll mit den Logs von fio arbeiten.
Mit dem Parameter *write_bw_logs* = [*Dateiname*], kriege ich eine Datei mit jeweils 5 Spalten:

```
1    Time, Bandwidth, data direction, Blocksize, Offset
2    0,      59782,    0,             4096,      0
```

Time für die Zeit in Millisekunden (ms) die verlaufen ist, Bandwidth für die Brandweite in Megabyte/ms für das random read, Date direction ob gelesen (= 0) oder geschrieben (= 1) wurde und ein Offset. Die Log Daten sind immens Lang und nicht schön lesbar. Deshalb wurde schon ein Tool in Python gebaut was diese log Daten auswertet. Für Veranschaulichung habe die das fio Programm auf meinem Rechner laufen lassen mit dem Command die oben erwähnt wurde. Die log Datei wurde ausgewertet und als .svg Datei ausgegeben:

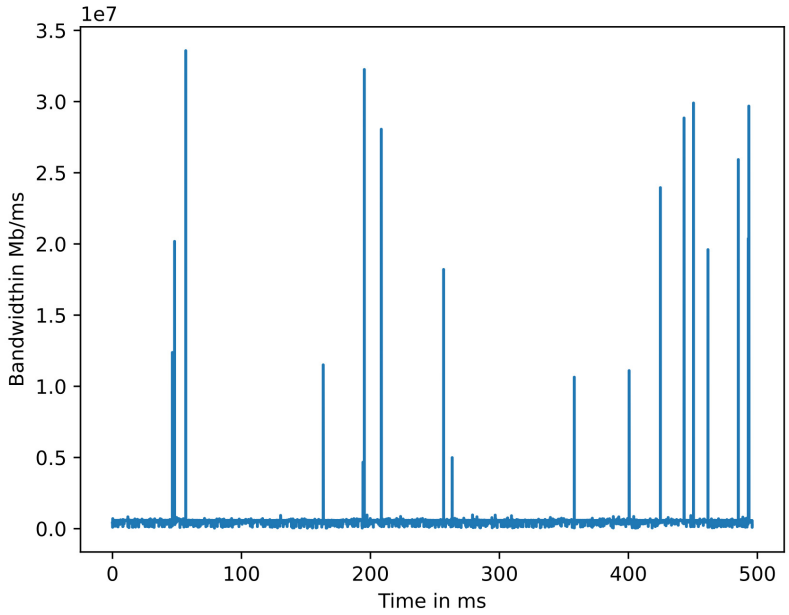

Figure 1: Plot tool Ausgabe mit Log Datei

Es wurde ein kleines Tool geschrieben. 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. Grund dafür ist, da dort die bereits die meiste Erfahrung steckt. Eine Möglichkeit, den stationären Zustand zu ermitteln, wäre das man die Standardabweichung verwendet und errechnet, wann die Abweichung klein ist.

4 Verwandte Arbeiten

5 Zeitplan

6 Referenz

- An Automated Approach for Recommending When to Stop Performance Tests [1]