Betriebssysteme Übung 2

Aaron Winziers - 1176638

14. Januar 2020

In dieser Ausarbeitung handelt es sich um das zweite Übungsblatt der Vorlesung zu Betriebssysteme im Wintersemester 2019. Der Sinn der Übung war es, die Performance-Verluste von verschiedenen Virtualisierungsmethoden zu analysieren und vergleichen. Hier sollten ein C-Programm, ein Java-Programm, und dasselbe C-Programm, aber auf jeweils einer Virtuellen Maschine, und innerhalb eines Containers ausgeführt und deren Laufzeiten analysiert werden.

1 Benchmark-Programme

Die Benchmark-Programme sollten zwei verschiedene Arten von Last prüfen, nämlich die CPU-Last und I/O-Last. In dieser Ausarbeitung wurden beide Arten von Lasten mittels einem einzigen Programms getestet. Dieses Programm hat jedoch mit zwei verschiedenen Funktionen die verschiedene Lasten analysiert und hat dementsprechend zwei verschiedene Zeitintervalle gemessen um diese getrennt zu halten.

1.1 CPU-Last

Um den CPU auszulasten wurde eine Funktion erstellt welche alle Primzahlen zwischen 2 und einer anzugebende Zahl `number` berechnet. Da es in dem Fall nicht darum ging die Primzahlen effizient zu berechnen, sondern sogar das Gegenteil, wurde ein sehr simples Verfahren verwendet. Um das Programm lang genug laufen zu lassen um eine Laufzeit zu haben die für den Vergleich zwischen den Virtualisierungsmethoden geeignet ist wurden die Primzahlen bis 250.000 ausgerechnet.

Abbildung 1 zeigt die in C implementierte Logik des Programms. Für die Variante die in Java implementiert wurde wurde sowohl eine identische Programmstruktur verwendet als auch das gleiche Algorithmus zum berechnen der Zahlen.

Um die Laufzeit zu erfassen wurde ein Timestamp vor und nach der Ausführung der Funktion erstellt und diese wurden anschließend verglichen.

```
bool primes[number];
for (int i = 0; i < number; i++) {
    primes[i] = true;
}
for (int i = 2; i < number; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0) {
            primes[i] = false;
            break;
        }
    }
}
```

Abbildung 1: CPU-Lastiges Programm

1.2 I/O-Last

Um möglichst viele I/O Operationen ausführen zu müssen wurde eine Funktion geschrieben die eine `.txt` Datei erstellte, 50 mal in diese Datei geschrieben hat und im Anschluss diese Datei wieder gelöscht hat. Diese Datei wurde dann 400.000 mal erstellt und gelöscht. Es wurde ein Fokus auf Schreib-Operationen gelegt, da diese meistens langsamer sind als Lese-Operationen.

Abbildung 1 zeigt die in C implementierte Logik des Programms. Für die in Java implementierte Variante wurde eine identische Programmstruktur verwendet.

Wie in der anderen Funktion wurde um die Laufzeit zu erfassen ein Timestamp vor und nach der Ausführung der Funktion erstellt und diese wurden anschließend verglichen.

```
for (int i = 0; i < number; ++i) {
    file = fopen(filename, "w");
        if (file == NULL){
        printf("Error creating file.\n");
        exit(EXIT_FAILURE);
    }
    for (int j = 0; j < 50; ++j) {
        fputs(data, file);
    }
    fclose(file);
    remove(filename);
}
```

Abbildung 2: I/O-lastiges Programm

	C-Programm		Java-Program		VM		Docker Container	
Ø Laufzeit	6487	9415	9407	14635	7535	20324	6680	10238
Ø Abweichung vom Ø	951	735	229	332	1001	821	989	726
% langsamer als C-Programm	0	0	45	55	16	116	3	9

Tabelle 1: Ausgewertete Laufzeiten

2 Hardware

Ausgeführt wurden die Testprogramme auf einem Rechner mit einem Intel i7-3770 CPU(8 Cores @ 3,40GHz) mit 16 GB Speicher und einer 500GB SSD (Schreibgeschwindigkeit 520 MB/s). Das Betriebssystem war Ubuntu 19.10 auf Linux Kernel Version 5.3.0-26-generic und der Container wurde mit Docker Version 19.03.3 erstellt und ausgeführt.

Die Verwendete VM wurde mit VirtualBox verwaltet und bekam 1 Kern, 4 GB Speicher und einer 32GB großen Virtual Disk zur Verfügung gestellt. Hier wurden nur so wenige Ressourcen zugewiesen um eine Realistische Anwendung einer VM abzubilden, da in den meisten Fällen eine VM nicht alle Ressourcen zugewiesen bekommen wird die ein Host Rechner hat.

3 Ergebnisse

Die Benchmark-Programme wurden jeweils auf den verschiedenen Platformen 10 mal ausgeführt und die Laufzeiten wurden notiert und anschließend ausgewertet. Die ausgewertete Ergebnisse sind in Tabelle 1 aufgelistet.

Zunächst war die Ausführung auf einer VM in dem CPU-Test recht ähnlich zu der Laufzeit des C-Programms und die 16% Verlangsamung kann wahrscheinlich dadurch erklärt werden, dass die Vm nur ein Kern zur Verfügung hatte. Ein sehr starker Verlust von 116% wurde jedoch bei dem I/O Test gemessen, was der größte Zeitverlust von allen Virtualisierungsmethoden war.

Das Java Programm war jeweils 45% und 55% langsamer bei den CPU- und I/O-Tests, hatte aber die geringste Abweichung von der Durchschnittszeit von allen Programmen.

Die Docker Containern waren die schnellsten unter den Virtualisierungen, mit Zeitverlusten von jeweils weniger als 10%

4 Fazit

Insgesamt haben die Ergebnisse der Ausführungen gezeigt dass in allen Fällen die Virtualisierungen langsamer sind, da aber die erstellten Benchmark-Programme einen nur sehr geringen Realitätsbezug besitzen sollten die konkrete Werte der Verlangsamungen nur als Bestätigung der Hypothese genommen werden, und nicht als tatsächliche Messung der Performance.