Heterogeneous Computing
Ergebnisbericht

Tillmann Faust

2.6.2024

Aufgabe 1

Zum Lösen der ersten Aufgabe wurde Versucht mithilfe von Chat GPT-4o eine Python Anwendung zu erstellen, welche in der Lage ist die wav-Datei einzulesen, die sliding-window-fft auf durch den Nutzer bestimmt großen Blöcken (im Programm standardmäßig auf 1024 gesetzt) durchzuführen, die Ergebnisse der block-ffts auf jedem Block zu speichern und die Ergebnisse anschließend per pyplot darzustellen. Die erste Iteration des Programms gab den Querschnitt, also die Verteilung des Frequenzen als Graph zurück, der anschließende Versuch das Programm so abzuwandeln, dass die einzelnen Frequenzen sichtbar sind scheiterte. Das abgegebene Programm führt die sliding-window-fft wie erläutert durch, das Ausgegebene Diagramm ist entspricht allerdings nicht den Erwartungen.

Aufgabe 2

Zur Beobachtung des Speicherverbrauchs wurde die Python eigene Bibliothek 'tracemalloc' verwendet. Das Programm wurde so modifiziert, dass an Schlüsselstellen, d.h. bei Programmstart, nach einlesen der wav-Datei, nach jeder fft und nach Durchlauf des Programms die vergangene Zeit seit Start des Programms und der verbrauchte Speicher mit tracemalloc ermittelt und in einer Liste gespeichert wurde.Der Speicherverbrauch beträgt zu Beginn des Programms 0MB und erreicht seinen höchsten Punkt nach ca. 550 Sekunden mit ca. 3,367GB: Insgesamt stieg der Speicherverbrauch linear an. Nach Durchlauf des Programms wurde der ermittelte Speicherverbrauch in einem Scatterplot (Siehe fig. 1) Plot graphisch dargestellt.

Es ist anzumerken, dass der durch tracemalloc ermittelte Speicherplatz den im Taskmanager angegbenen ca 10,5GB Speicheraufwand widerspricht. Dies kann verschiedene Ursachen haben. Unter anderem berücksichtigt tracemalloc nur den direkten Speicherplatzverbrauch der Anwendung, was bedeutet, dass z.B. durch Module verwendete Python- und C-Bibliotheken, nicht berücksichtigt werden. Da tracemalloc nur den direkten Speicherplatz betrachten, kann es

passieren, dass Speicherblöcke nicht vollständig belegt, aber dennoch blockiert werden. Das Programm benötigt dadurch reell mehr Speicherplatz, als die im Programm verwendeten Datenstrukturen.

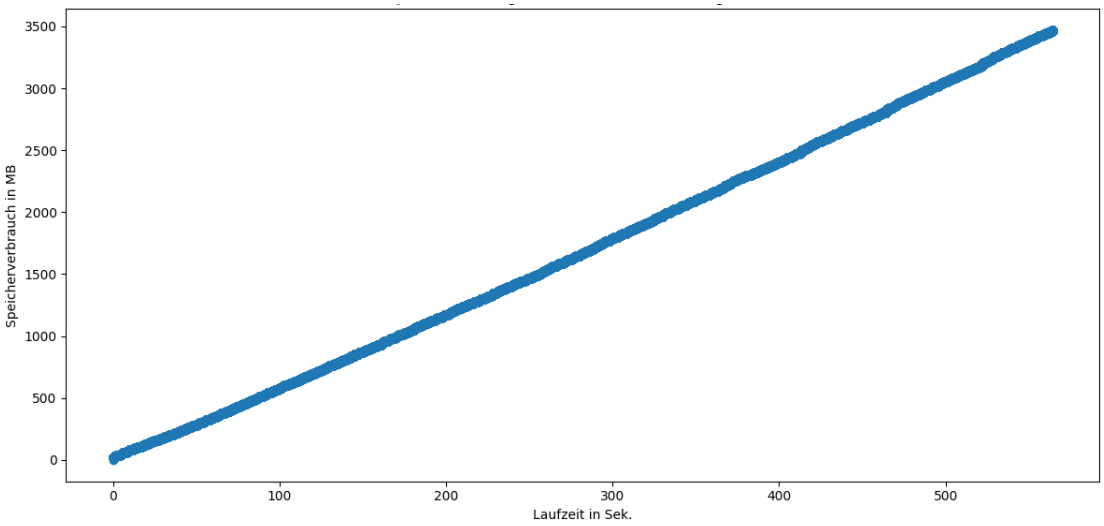

Figure 1: Speicherverbrauch des Python-Programms über die Laufzeit

Aufgabe 3

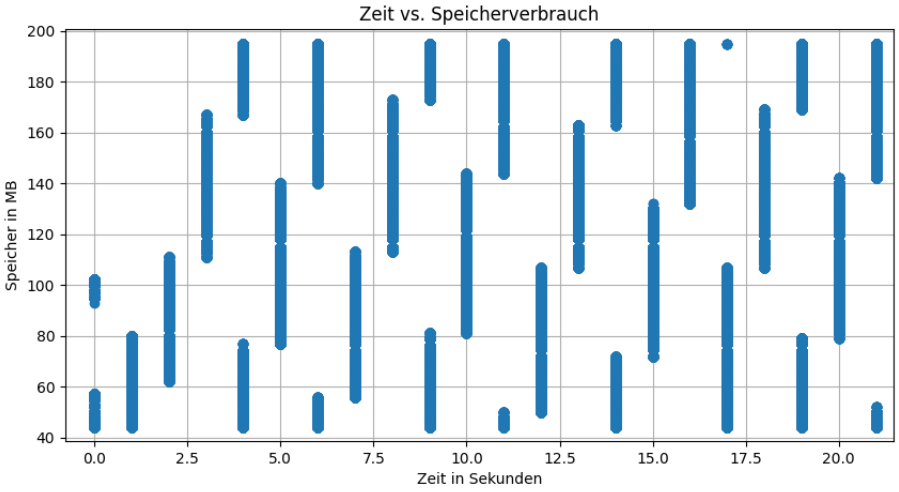

Figure 2: Speicherverbrauch des Java-Programms über die Laufzeit

Zum Vergleich des Speicherverbrauchs der Python Implementation mit anderen Sprachen wurde die sliding-window-FFT ebenfalls in Java implementiert und der Speicherverbrauch über Abfragen an den Garbage Collector realisiert. Das Ergebnis ist zum einen, dass die Realisierung der FFT in Java signifikant schneller läuft (ca. 120 Sekunden, statt über 550 Sekunden in Python), zum anderen überschreitet der Speicherverbrauch nie die 250MB, sondern wächst immer bis ca. 200MB und fällt anschließend wieder ab. Ein möglicher Grund dafür ist, dass die berechneten Werte im Laufe des Programms nicht weiter verwendet werden und der Garbage Collector den Speicher daher wieder freigibt, bevor ein starkes Wachstum im Speicherplatzverbrauch erkennbar ist. Einen Ausschnitt des Musters beim Speicherplatzverbrauch über die ersten 23 Sekunden der Laufzeit ist in Figur 2 erkennbar.