Supersonic Algorithms

Anton Rodenwald

January 10, 2023

Kurzfassung

Nachdem wir im Informatikunterricht der SEK II Sortieralgorithmen behandelt hatten, stellte ich mir die Frage, wie man am schnellsten eine Liste von 10 Millionen zufällig Generierten Zahlen sortieren kann und welche Programmiersprache und welche Techniken man nutzen sollte. Daraus entwickelte sich dann die etwas allgemeinere Fragestellung, nämlich welche Optimierungen erhöhen die Ausführgeschwindigkeit von Programmen am meisten und wieso? Mir war bekannt, dass Python, was wir im Unterricht verwendet hatten, als eine der langsamsten Sprachen gilt, weswegen ich neben Python auch noch C++ wählte, was allgemein als eine der schnellsten Sprache gilt. Ich implementierte anschließend verschiedene Variationen der Quicksort und anderer Algorithmen und testete so, in welchem Maß Optimierungsansätze die Performance beeinflussten. Dabei kam ich zu dem Ergebnis, dass die besten Python Bibliotheken zur Optimierung ”numpy” und ”numba” waren, wobei C++ trotzdem schneller war, womit sich meine Hypothese bestätigte. Dies erklärte ich mir dadurch, dass die Python eine Interpretierte und C++ eine kompilierte Sprache ist, diese beiden also gänzlich verschiedenen und somit auch die Möglichkeiten zur Optimierung total verschieden sind. Schlussendlich gelang es mir noch unter Nutzung von AVX2, in C++ eine 4x schnellere Version als die standardmäßig Vorhande zu entwickeln, einem speziellen Befehlssatz, was mir zeigte, dass es im Gebiet der Codeoptimierung noch viel zu entdecken gibt.

Inhaltsverzeichnis

Einleitung
Vorgehensweise, Materialien, Methode

1. Einleitung

Im Informatik Leistungskurs des 12 Jahrgang beschäftigten wir uns nach den Herbstferien mit der Laufzeit von Algorithmen am Beispiel der Quicksort, einem Sortieralgorithmus. Nach diesem thematischen Impuls ergab sich mein Projekt zur Erforschung von Sortieralgorithmen, wobei ich mich schnell auf die Aspekte der Implementation und tatsächlichen Ausführung von Algorithmen fokussierte, da Sortieralgorithmen algorithmisch bereits sehr weit erforscht sind. Ich entschied mich deswegen, nicht nach besseren Algorithmen zu suchen, sondern nach Wegen, mein ursprüngliches Program in Python in seiner Ausführung zu beschleunigen und so vorteilhafte Wege der Geschwindigkeitsoptimierung zu entdecken. Ich stellte mir die Frage, welche Optimierungen die Ausführgeschwindigkeit von Programmen am meisten erhöhen und wieso? Außerdem wollte ich ein möglichst schnelles Program zur Sortierung implementieren, weswegen ich neben Python, einer interpretierten und als langsam geltenden Sprache, noch C++, eine hochperformante, kompilierte Sprache wählte. Zusätzlich entschied ich mich noch, in einigen anderen Sprachen meine Quicksort zu implementieren, weil mich ein Vergleich verschiedner Sprachen interessierte.

Vorgehensweise, Materialien, Methode

testen
zeitmessung
software versionen
python, c++, java, javascript, lua, go, julia

Ergebnisse

Diskussion

Zusammenfassung

Literaturverzeichnis

References

[1] Pierre Terdiman, *Radix Sort Revisited*, http://codercorner.com/RadixSortRevisited.htm *Zugriff am: 10.1.23*, *Veröffentlicht am 4.01.2000*

[2] Michael Herf, *Radix Tricks*, http://stereopsis.com/radix.html *Zugriff am: 10.1.23*, *Veröffentlicht im December 2001*