Supersonic Algorithms

Anton Rodenwald

January 11, 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

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, in dem ich es mir zuerst zur Aufgabe machte, den schnellsten Weg zu finden, 10 Millionen zufällig generierte zahlen zu sortieren. Mein Fokus änderte sich dann allerdings und ich fokussierte mich schon früh auf die Aspekte der Implementation und tatsächlichen Ausführung der Programme, da Sortieralgorithmen algorithmisch bereits sehr weit erforscht sind und man in diesem Gebiet nur sehr schwer neue Erkenntnisse sammeln konnte. 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? Zum Thema der Optimierung fand ich im Bezug auf Python einige Bibliotheken im Internet, mit denen sich die Performance verbessern ließ, von denen ich einige Auswählte, um herauszufinden, welche dieser den größten Geschwindigkeitsboost bringt. Im Bezug auf C++ fand ich nicht direkt Möglichkeiten und Erklärungen, welche Modifikationen ein Program aus welchen Gründen schneller machen, weswegen ich mich hier auch vorallem anschaute, was ein C++ Program in seiner Ausführung von einem Python Program unterscheidet und wieso es meist deutlich schneller ist. Meine Erwartung war, dass C++ bei jeder Aufgabe Python um ein vielfaches übertrifft im Punkt Geschwindigkeit, da Python im allgemeinen als langsam gilt und C++ als sehr performant.

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*