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 Python Bibliotheken ”numpy” und ”numba” der Bibliothek ”cython” deutlich überlegen waren, doch mir wurde vorallem klar, dass alle diese Bibliotheken sich auf die Ausführung von immer mehr hochperformantem C Code verlassen, also die Ausführung von Python Code nicht optimiert wurde. Im Vergleich zu C++ wird der große Unterschiede zwischen interpretierten und kompilierten Sprachen klar und welche Optimierungsmöglichkeit bei diesen unterschiedlichen Konzepten möglich sind. Insgesamt zeigte sich, dass sich fast alle getesteten Sprachen auf das Aufrufen von schnellen, bereits kompilierten C/C++ Funktionen verlassen und somit eher diese die untere Grenze der Optimierung darstellen, weil die Sprache beim Aufrufen dieser Funktionen beliebig sein kann.

Inhaltsverzeichnis

tbd

1. Einleitung

Motivation
Fragestellung
Fokus Implementierung, nicht Algorithmik
Aktuelle Lage
Anknüpfung an andere meinungen, hypothese

Vorgehensweise, Materialien, Methode

testen
zeitmessung
[1]
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*