Technische Universität München

Parallele Berechnung der Mandelbrotmenge

Einführung in die Rechnerarchitektur Großpraktikum

Maximilian Frühauf, Tobias Klausen, Florian Lercher, Niels Mündler

Abstract

Die Leistung und Geschwindigkeit des individuellen Rechenkerns stagniert seit einigen Jahren. Moderne Computer erlangen einen Großteil ihrer erhöhten Rechenleistung seit einiger Zeit nur noch durch Parallelisierung. Diese sollte jedoch geschickt gestaltet werden, um unerwünschte Seiteneffekte wie Leerlauf zu vermeiden.

In diesem Projekt wird die Mandelbrotmenge verwendet, um dem Benutzer die Effekte einer korrekten Parallelisierung zu verdeutlichen.

Mandelbrotmenge

Um die Mandelbrotmenge zu berechnen wendet man folgende Formel wiederholt auf eine komplexe Zahl $c \in C$ an: $z_{n+1} = z_n^2 + c, z_0 = 0$ In der Mandelbrotmenge befinden sich alle solche c, für die $\lim_{n\to\infty} |z_n| < \infty$.

MPI

Das Message Passing Interface ist eine weit verbreitete Spezifikation, für die Kommunikation zwischen unabhängigen Rechenknoten.

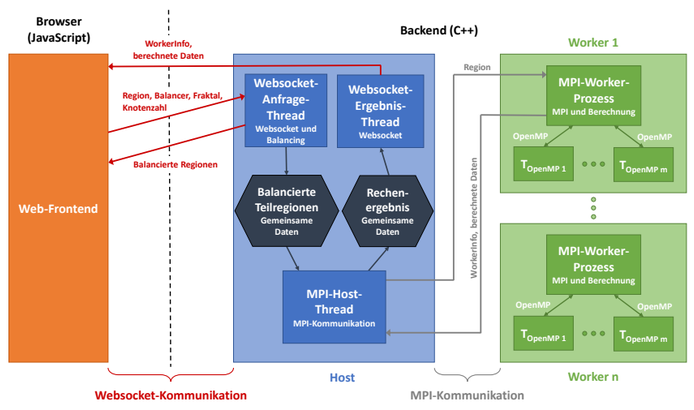
OpenMP

OpenMP2 (Open Multi-Processing) ist ein API, das auf die Parallelisierung von Schlei- fen und Programmabschnitten auf Shared Memory Systemen spezialisiert ist

SIMD

„Single Instruction, Multiple Data“ setzt auf Hardwareebene um, was der Name bereits andeutet: Eine Instruktion wird auf verschiedene Daten gleichzeitig angewendet.

Architektur



Die hohe Rechenintesivität erfordert, dass der Berechnungsteil des Projektes in einer hardwarenahen Sprache (C++) umgesetzt wird. Andererseits sollte die Benutzeroberfläche einfach zu bedienen und auf möglichst vielen verschiedenen Geräten lauffähig sein. Daher wurde sich für eine Zweiteilung entschieden, in ein Frontend (JavaScript), im Browser aufrufbar, und ein Backend, auf einem Rechencluster laufend und hardwarenah programmiert.

Lastbalancierung

Um die Effizienz der parallelen Berechnung der Mandelbrotmenge zu erhöhen, sollte die Rechenlast möglichst gleichmäßig auf die Worker verteilt werden. Für diese Verteilung sind unter anderem die folgenden Strategien implementiert:

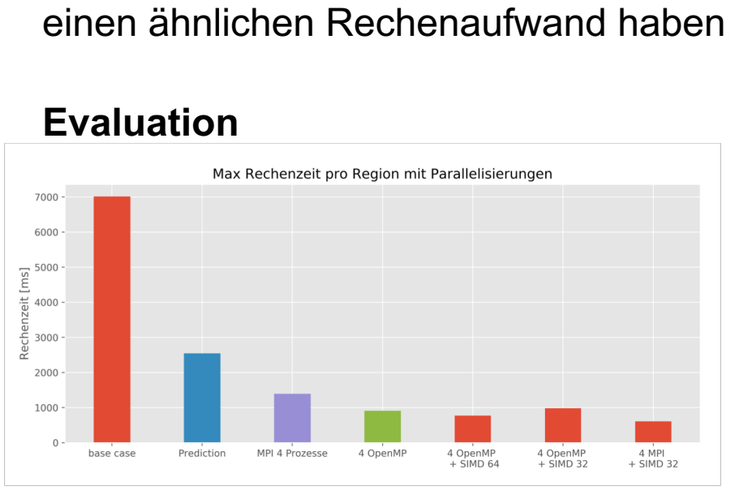
Naive Strategie

Bei der naiven Strategie wird versucht den einzelnen Workern etwa gleich große Teilregionen zuzuweisen. Dabei wird werden die Rechenzeiten der Teilregionen nicht beachtet.

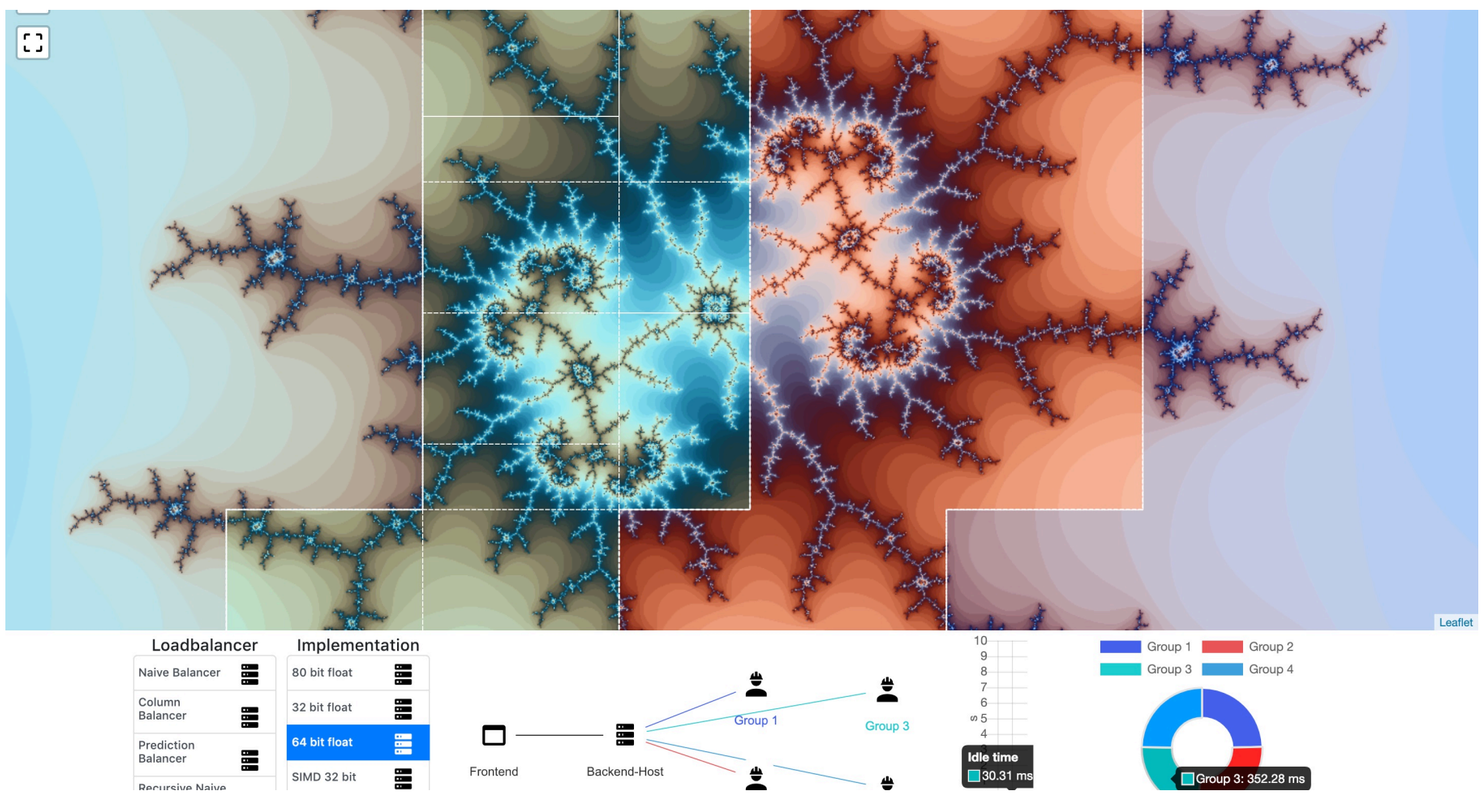
Strategie mit Vorhersage

Bei dieser Strategie basiert die Aufteilung der Region auf einer Vorhersage über die Rechenzeit. Die Teilregionen werden so gewählt, dass sie, entsprechend der Vorhersage, etwa einen ähnlichen Rechenaufwand haben.

Evaluation



Unter der Verwendung aller implementierten Parallelisierungen ließ eine Verbesserung der maximalen Rechenzeit im Vergleich zu einer naiven Lastbalancierung vom Faktor ≈ 7 erzielen, was einer absoluten Rechenzeit von 0.6s für 1280px × 720px entspricht.



Benutzeroberfläche des Frontends für eine Aufteilung mit 40 Rechenknoten