INFORMATIK FAKULTÄT

TECHNISCHE UNIVERSITÄT MÜNCHEN

Seminararbeit

Experimenteller Vergleich von Methoden zur parallelen Berechnung der Mandelbrotmenge

Pascal Ginter

INFORMATIK FAKULTÄT

TECHNISCHE UNIVERSITÄT MÜNCHEN

Seminararbeit

Experimenteller Vergleich von Methoden zur parallelen Berechnung der Mandelbrotmenge

Titel der Abschlussarbeit

Author:	Pascal Ginter
Supervisor:	Supervisor
Advisor:	Advisor
Submission Date:	25.06.2019

I confirm that this seminararbeit is my own work and I have documented all sources and material used.

Munich, 25.06.2019 Pascal Ginter

Acknowledgments

Abstract

Die Berechnung der Mandelbrot Menge ist ein Problem, welches sich aufgrund der Unabhängigkeit der Berechnung der einzelnen Punkte zur Parallelisierung ausgezeichnet anbietet. Aufgrund der unregelmäßigen Verteilung der Rechenzeit pro Punkt gibt es jedoch große Unterschiede. Das Problem für parallele Algorithmen ist somit, die zu berechnenden Punkte möglichst gleichmäßig auf die verschiedenen Threads zu verteilen. Diese Arbeit vergleicht den Dynamic Load Balancing Algorithmus und den Estimation Based Load Balancing Algorithmus.

Contents

1 Einführung

1.1 Komplexe Zahlen

1.1.1 Definition

Die Menge der komplexen Zahlen $\mathbb{C}$ erweitert die Menge der realen Zahlen $\mathbb{R}$ um die imaginäre Einheit i. Diese Einheit ist interessant, weil sie folgende Eigenschaft besitzt: $i^2 = -1$. Die Menge $\mathbb{C}$ ist definiert durch $\mathbb{C} = \{x + i \cdot y | x, y \in \mathbb{R}\}$. Dies ist interessant, weil dadurch Gleichungen die in $\mathbb{R}$ keine Lösungen besitzen, in $\mathbb{C}$ gelöst werden können. Anwendung für die komplexen Zahlen finden sich in der Physik und der Elektrotechnik, namentlich ist hierbei der Wechselstrom zu nennen.

Im Rahmen dieser Arbeit sind nur wenige Aspekte der komplexen Zahlen tatsächlich relevant, die nötigen Formel sind hier drunter kurz ohne weitere Erklärungen aufgelistet:

$$z_1 + z_2 = (x_1 + i \cdot y_1) + (x_2 + i \cdot y_2) = (x_1 + x_2) + i \cdot (y_1 + y_2)$$
$$|z| = |x + i \cdot y| = \sqrt{a^2 + b^2}$$
$$z^2 = (x + i \cdot y)^2 = x^2 - y^2 + i \cdot 2 \cdot x \cdot y$$

1.1.2 2-dimensionale Darstellung

Man kann die komplexen Zahlen im 2-dimensionalen Plan darstellen. Eine Achse repräsentiert dabei den realen Teil der Zahl, der andere den imaginären. Dies wird bei der Darstellung der Mandelbrotmenge noch wichtig werden.

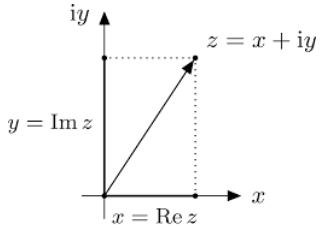

Figure 1.1: Der Plan der komplexen Zahlen [mat]

1.2 Die Mandelbrot-Menge

1.2.1 Definition

Die Mandelbrot-Menge $\mathbb{M}$ ist eine Teilmenge der komplexen Zahlen. Sie erhält sich durch die quadratische Differenzgleichung für einen Punkt c des komplexen Plans:

$$z_{n+1} = {z_n}^2 + c \mid z_0 = c$$

$$\mathbb{M} = \{c | \lim_{n \to \infty} z_n \leq 2\}$$

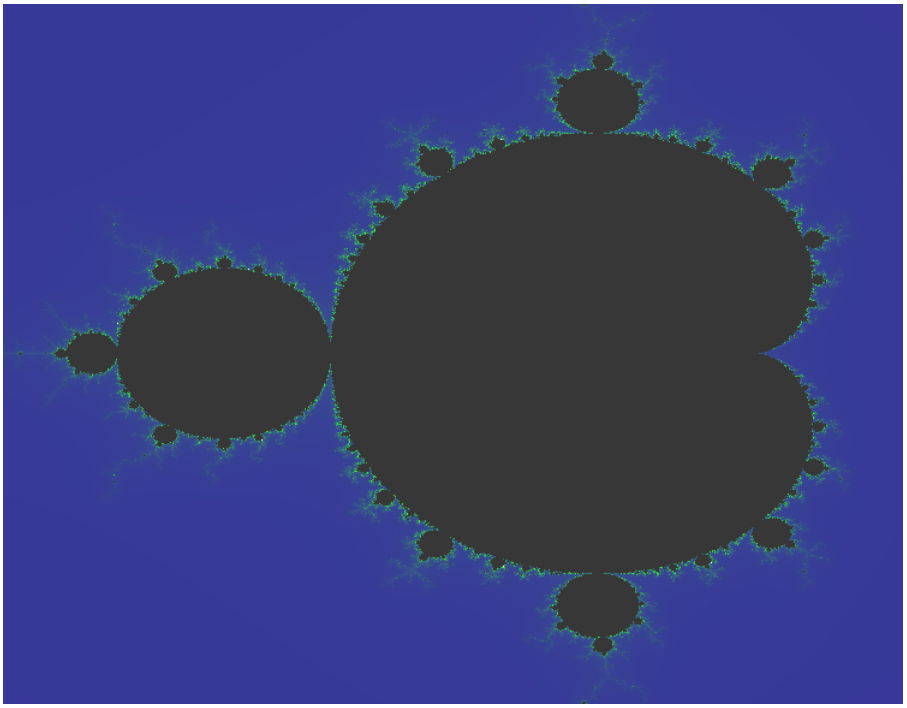

Figure 1.2: Darstellung der Mandelbrot-Menge

1.2.2 Sequentieller Algorithmus

Der Sequentielle Algorithmus ist die einfachste Art und Weise die Mandelbrotmenge zu berechnen. Der Algorithmus berechnet einfach Punkt für Punkt. Weil es praktisch nicht möglich ist, eine Folge bis in unendliche zu berechnen, wird ein Limit I an Iterationen festgelegt, ab der angenommen wird, dass der Punkt zur Mandelbrotmenge gehört.

Algorithm 1 Berechnung der gekappten Anzahl an Iterationen die für einen beliebigen Punkt der Mandelbrotmenge benötigt werden um den Wert 2 zu übertreffen

Require: c die komplexe Zahl, I die Anzahl erlaubter Iterationen
  $c \leftarrow 0$
  $i \leftarrow 0$
  while $|c| \leq 2$ **and** $i < I$ **do**
    $c \leftarrow c^2 + z$
    $i \leftarrow i + 1$
  end while
  return i

Falls der Rückgabewert des Algorithmus I ist, gehört der Punkt zur Mandelbrot-Menge. Andernfalls wird der Rückgabewert I benutzt, um die graphische Darstellung der Menge ansprechender und farbenreicher zu gestalten.

1.2.3 Verteilung der Rechenlast

A. Douady und J. Hubbard [DH82] haben gezeigt, dass die Mandelbrot-Menge eine Fläche ist. Weil der sequentielle Algorithmus für Punkte der Mandelbrot-Menge mehr Iterationen durchführt als für Punkte außerhalb, braucht es länger um diese Punkte zu bestimmen. Weil die Mandelbrot-Menge eine Fläche ist folgt, dass der Algorithmus für bestimmte Bereiche des komplexen Raumes mehr Zeit für die Berechnung braucht als für alle. Weil ein die Ausführung eines jeden Programms so lange dauert, bis auch der letzte Thread seine Aufgabe abgeschlossen und sich beendet hat, gilt es bei der Entwicklung paralleler Algorithmen zur Berechnung der Mandelbrot-Menge zu beachten, dass die Laufzeit der einzelnen Threads möglichst identisch ist.

2 Algorithmen zur parallelen Berechnung

2.1 Static Load Balancing

Der Static Load Balancing Algorithmus ist die naive Herangehensweise. Der zu berechnende Bereich wird für die Berechnung mit n Threads in n gleichgroße Teile aufgeteilt. In der Implementierung dieser Arbeit wird der Einfachheit halber der Bereich in n vertikale Streifen aufgeteilt.

2.2 Dynamic Load Balancing

Der Dynamic Load Balancing Algorithmus von Mirco Tracolli [Tra16] versucht, die Ungleichheit der nötigen Rechenleistung pro Bereich dadurch auszugleichen, dass der Bereich stärker unterteilt wird. Bei n Threads und einem Unterteilungsfaktor u wird der Bereich in $u \cdot n$ gleichgroße Bereiche aufgeteilt. Jedem Thread wird zu Beginn einer der Bereiche zugeteilt. Solange es noch nicht berechnete Bereiche gibt, erhält ein Thread bei Beendigung seiner vorherigen Aufgabe direkt einen neuen Bereich. Auch die Implementierung dieses Algorithmus unterteilt den zu berechnenden Bereich wieder in vertikale Streifen.

2.3 Prediction Based Load Balancing

Der Prediction Based Load Balancing Algorithmus wurde von Maximilian Frühauf, Tobias Klausen, Florian Lercher und Niels Mündler entwickelt [Frü+19]. Dieser Algotithmus berechnet für jede Spalte einen zufällig ausgewählte Wert als Stellvertreter. Basierend auf der Anzahl an Iterationen, die für die Berechnung des Stellvertreters notwendig waren wird der Rechenaufwand für die jeweilige Spalte geschätzt und die Spalten so zwischen den Threads aufgeteilt, dass jeder Thread eine möglichst ähnliche voraussichtliche Laufzeit hat.

3 Vergleich der verschiedenen Methoden

3.1 Details zu Implementierung und Testumgebung

Sämtliche Algorithmen wurden mithilfe von Java openjdk 11.0.2 implementiert. Die Tests wurden unter Ubuntu 18.04.2auf einem i7-8550u Prozessor mit 4 Kernen durchgeführt.

3.2 Versuchsaufbau

Zum Vergleichen haben die drei vorgestellten parallele Algorithmen sowie der sequentielle Algorithmus, die gleichen 10.000 zufällig gewählten Bereiche der Mandelbrotmenge mit einer Genauigkeit von 256 Iterationen und einer Auflösung von $1920 \cdot 1080$ Punkte berechnet. Die parallelen Algorithmen wurden auf 4 Kernen ausgeführt und der Dynamic Load Balancing Algorithmus mit einem Unterteilungsfaktor von 16. Für die Berechnung eines jeden Bildes wurde für die 3 Methoden die Ausführungszeit gemessen und in Verhältnis zu der gemessenen Zeit des sequentiellen Algorithmus gespeichert. Der Mittelwert der proportionalen Ausführungszeit wird benutzt um die Effizienz der parallelen Algorithmen zu vergleichen.

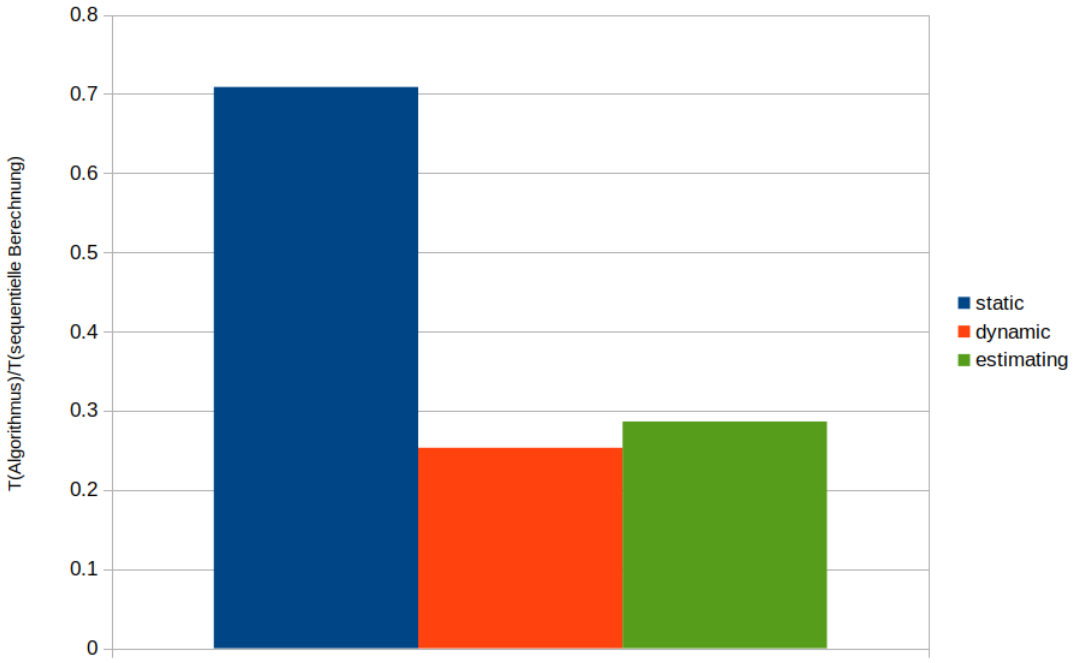

Figure 3.1: Ausführungszeiten der verschiedenen parallelen Algorithmen in Verhältnis zum sequentiellen Algorithmus

3.3 Ergebnisse und Diskussion

Bei Idealer Aufteilung der Rechenlast, würde ein paralleler Algorithmus mit 4 Threads exakt 25 % der Zeit des sequentiellen Algorithmus' brauchen. Diesem Idealzustand kommen sowohl der Dynamic Load Balancing Algorithmus (25,31%) als auch der Estimation Based Balancing Algorithmus (28,62%) sehr nahe. In der aktuellen Implementierung ist der Dynamic Load Balancing Algorithmus die bessere Wahl, sein Ergebnis erreicht fast den Idealwert und übertrifft die Erwartungen bei Weitem. Der Unterschied zwischen Idealzeit und der vom Prediction Based Load Balancing Algorithmus erreichten Zeit ist unter anderem der Berechnung der Vorhersage geschuldet, welche nur schwer parallelisiert werden kann.

List of Figures

List of Tables

Bibliography

[DH82] A. Douady and J. Hubbard. *Itération des polynômes quadratiques*. 123-126. C. R.Acad. Sci. Paris, 1982.

[Frü+19] M. Frühauf, T. Klausen, F. Lercher, and N. Mündler. *Parallele Berechnung der Mandelbrotmenge*. `https://github.com/eragp/mandelbrot/tree/master/doc`. Online, eingesehen am 19.06.2019. 2019.

[mat] mathe-online.at. *2D-Visualisierung komplexer Funktionen*. `https://www.mathe-online.at/nml/materialien/innsbruck/komplex2d/Komplexe-Funktionen-2D.pdf`. Figur 1, Online, eingesehen am 19.06.2019.

[Tra16] M. Tracolli. *Parallel generation of a Mandelbrot set*. `http://services.chm.unipg.it/ojs/index.php/virtlcomm/article/view/112`. Online, eingesehen am 19.06.2019. 2016.