ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert

5. Juli 2019
Blatt 7

Parallele Programmierung mit C++ (SS 2019)

Abgabe bis zum 12. Juli 2019, 14:00 Uhr

Lernziele:

- Entwicklung eines geeigneten Protokolls für ein MPI-basiertes Teile-und-Herrsche-Pattern mit einer rekursiven und ungleichmäßig tiefen Raumaufteilung
- Effiziente Datenübertragung mit Hilfe selbstdefinierter Datentypen in MPI

Aufgabe 8: Visualisierung der Mandelbrot-Menge

Die Mandelbrot-Menge ist eine von Benoît Mandelbrot entdeckte Teilmenge der komplexen Zahlen, die auf den folgendermaßen definierten Folgen $\{z_n(c)\}_{n\in\mathbb{N}_0}$ für jeden Punkt $c \in \mathbb{C}$ beruht:

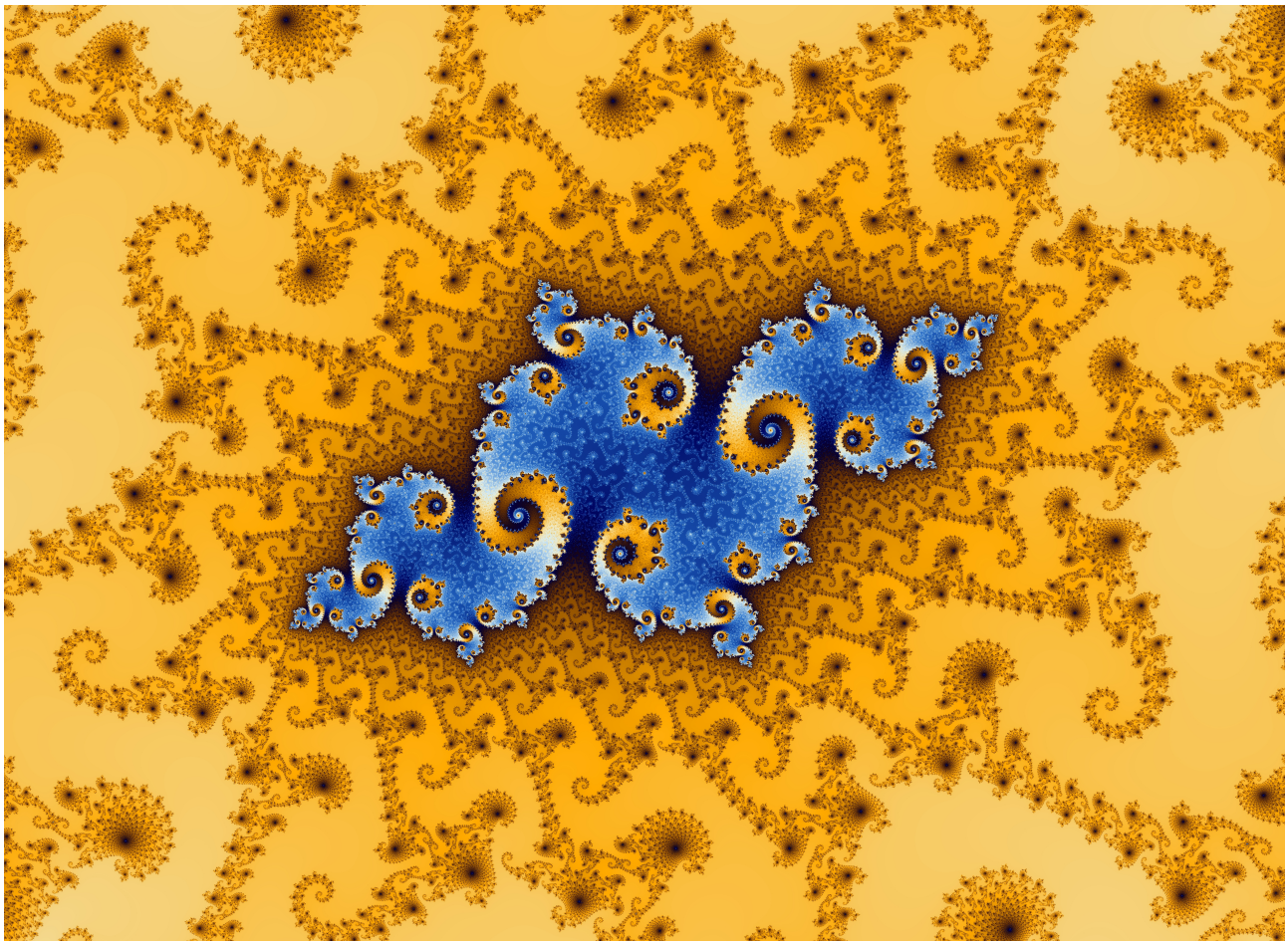
$$\begin{aligned} z_0(c) &:= 0 \\ z_{n+1}(c) &:= z_n(c)^2 + c \end{aligned}$$

Die Mandelbrot-Menge lässt sich dann wie folgt definieren:

$$M := \{c \in \mathbb{C} \mid z_n(c) \not\to \infty \text{ für } n \to \infty\}$$

Um diese Menge zu bestimmen, ist es hilfreich zu wissen, dass $|z_n(c)| \leq 2$ gilt für alle $c \in M, n \in \mathbb{N}_0$. Wenn numerisch festgestellt werden soll, ob $c \in M$, dann wird die Folge berechnet, bis entweder ein n gefunden wird mit $z_n(c) > 2$ oder n eine vorgegebene Grenze überschreitet. Das lässt sich in eine Grafik umsetzen, bei der jeder Punkt unterschiedlich (weiß oder schwarz) markiert wird, je nachdem, ob die Grenze erreicht wird oder nicht. Es ist aber entsprechend dem sogenannten „Escape-Time-Algorithmus“ auch möglich, der Zahl der Iterationen unterschiedliche Farben zuzuordnen, ggf. auch mit Farbüberläufen. Hier ist ein Beispiel mit einem bläulich/gelblichen Farbverlauf gezeigt an einem

Ausschnitt mit einem horizontalen Durchmesser von 0.000000000085 und einer Auflösung von 5200×3800 Pixeln an der Position $-0.7436438870371 + 0.13182590420531i$:



Das Verfahren zur Darstellung, die jedem Punkt c mit einen Farbwert zuordnet, lässt sich hervorragend parallelisieren, da die Berechnungen unabhängig voneinander sind. Jedoch ist eine einfache Aufteilung der gewünschten Fläche in der komplexen Zahlenebene keine sinnvolle Vorgehensweise, da die Zahl der Iterationen und damit der Rechenaufwand sehr unterschiedlich sein kann. Während am Rand des gezeigten Bilds nur sehr wenige Iterationen benötigt werden, um den Grenzwert von 2 zu erreichen, muss im Bereich von M bis zur vorgegebenen Grenze gerechnet werden. Bei Übersichtsbildern genügt 1000, bei dem gezeigten Ausschnitt wurde M auf 100000 gesetzt, um mehr Präzision in der Darstellung noch zu gewinnen.

Es liegt nahe, die vorgegebene Fläche in rechteckige Quadranten aufzuteilen. Für jede solche Teilfläche ist es hilfreich, zuerst die Pixel für den Rand zu berechnen. Wenn diese alle die gleiche Farbe haben (wie insbesondere im Inneren von M), dann kann davon ausgegangen werden, dass auch der Innenteil diese Farbe besitzt (dieser Trick geht zusammen mit der rekursiven Flächenaufteilung auf Gropp, Lusk und Skjellum zurück). Wenn der Rand nicht gleichmäßig ist, sollte der Quadrant weiter entsprechend des Teile-und-Herrsche-Prinzips rekursiv in vier Teile aufgeteilt werden, solange die entstehenden Rechtecke nicht zu klein werden.

Im Rahmen des Master/Worker-Patterns verwaltet der Master dann eine Warteschlange mit zu bearbeitenden Teilflächen. Wenn ein Worker eine solche Fläche übernimmt, wird zunächst bestimmt, ob diese sehr klein ist. Wenn ja, wird sie vollständig berechnet und an den

Master übergeben. Falls nein, wird nur der Rand betrachtet und hier das Ergebnis zurückgeliefert mit dem Hinweis, ob die Farbe einheitlich ist oder nicht. Der Innenbereich dieser Fläche wird dann vom Master wiederum in vier Quadranten zerlegt und der Warteschlange zugeführt. Dies läuft solange, bis die gesamte Fläche vollständig berechnet ist.

Bei der Kommunikation zwischen dem Master und den Workern ist ein geeignetes Protokoll zu entwickeln. Dabei ist darauf zu achten, dass unnötige Kopieraktionen unterbleiben. D.h. beim Empfang eines Resultats sollte der Master dieses bei *MPI_Recv* direkt in die Zielmatrix überführen.

Es steht Ihnen frei, ob Sie die Worker direkt die Farbwerte berechnen lassen oder nur die Zahl der Iterationen und diese anschließend in Farbwerte umrechnen. Letzteres ist naheliegend, wenn Sie keine dynamischen Farbübergänge wünschen.

Um das Resultat zu visualisieren, empfiehlt sich die Verwendung der *gdk-pixbuf*-Bibliothek. Sie können hierfür – wie in *jacobi.cpp* demonstriert, die Funktion *create_pixbuf* aus der HPC-Vorlesungsbibliothek verwenden oder selbst manuell die Konvertierung vornehmen. Wie letzteres geht demonstriert das Beispiel *gdk-pixbuf-demo.tar*. Beim Übersetzen liefert

- `` `pkg-config --cflags gdk-pixbuf-2.0` ``
  die notwendigen Übersetzungsoptionen und
- `` `pkg-config --libs gdk-pixbuf-2.0` ``
  die Optionen für den Zusammenbau.

Die nur aus Header-Dateien bestehende HPC-Vorlesungsbibliothek steht auf unseren Rechnern unter */home/numerik/pub/pp/ss19/lib* zur Verfügung oder kann alternativ heruntergeladen werden als Archiv hpc.tar.gz.

Die farbliche Gestaltung bleibt Ihnen vollkommen überlassen. Es wäre nett, wenn Sie ein Resultat, das Ihnen gefällt, zusammen mit den Parametern (Mittelpunkt, horizontaler Durchmesser, Auflösung) Ihrem Ergebnis beifügen würden. Ich zeige es dann in den Übungen. Es ist auch sinnvoll, durch Tests festzustellen, bei welcher Größe einer Teilfläche eine weitere Unterteilung nicht mehr sinnvoll ist.

Verpacken Sie all Ihre Quellen wiederum mit *tar* in ein Archiv und reichen Sie dies ein:

```
tar cvf mandelbrot.tar *.?pp *.jpg [mM]akefile
submit pp 8 mandelbrot.tar
```

Viel Erfolg!