ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert

21. Mai 2019
Blatt 4

Systemnahe Software II (SS 2019)

Abgabe bis zum 28. Mai 2019, 14:00 Uhr

Lernziele:

- Anwendung des Master/Worker-Pattern
- Kommunikation und Synchronisierung mit Hilfe eines gemeinsamen Speicherbereiches und einer Pipeline

Aufgabe 4: Visualisierung der Mandelbrotmenge

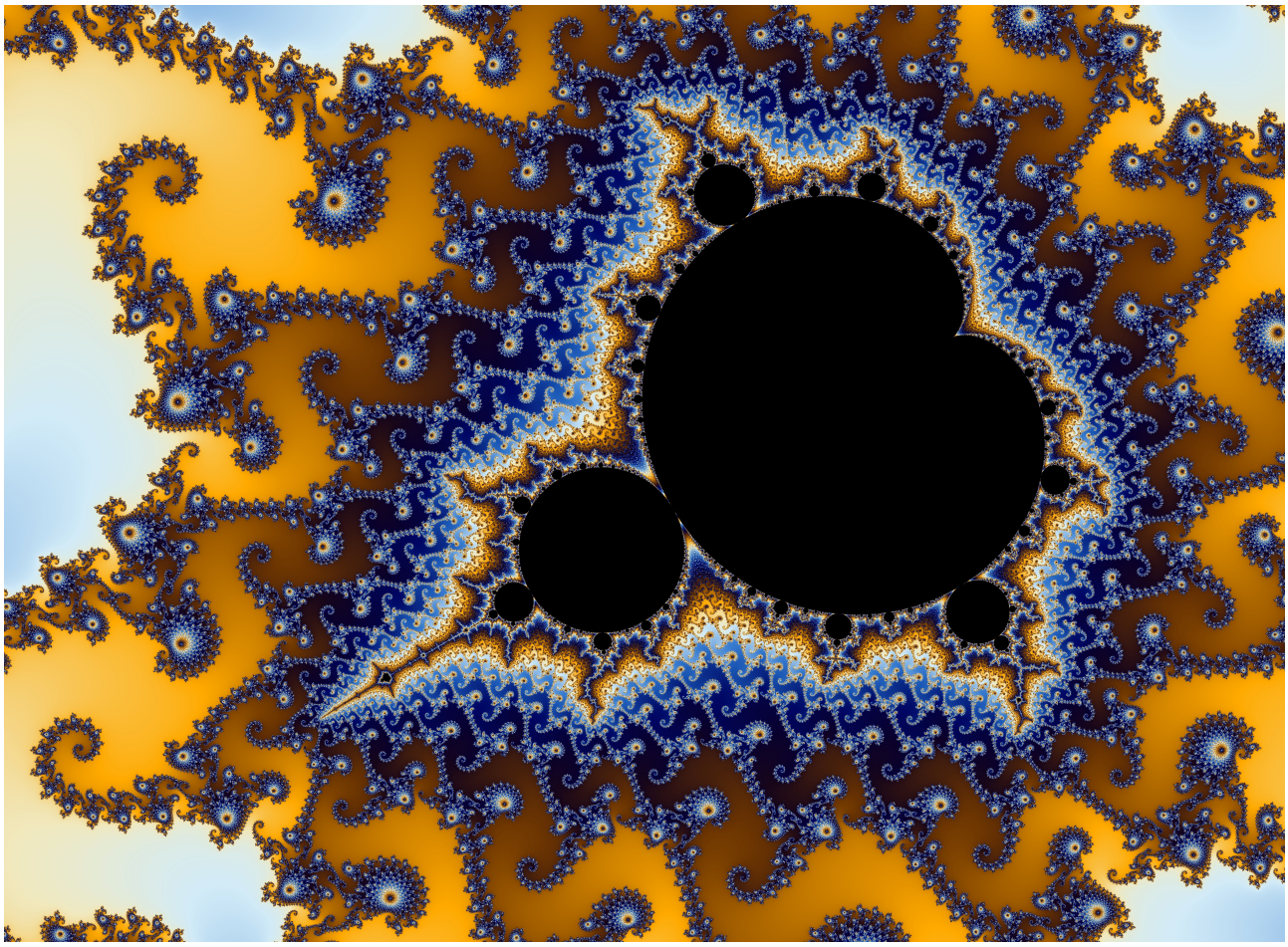
Die Mandelbrot-Menge ist eine von Benoît Mandelbrot entdeckte Teilmenge der komplexen Zahlen, die auf den folgendermaßen definierten Folgen $\{z_n(c)\}_{n\in\mathbb{N}_0}$ für jeden Punkt $c \in \mathbb{C}$ beruht:

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

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 Ausschnitt, bei dem die Punkte, bei der die maximale Zahl von Iterationen erreicht worden ist, schwarz dargestellt worden ist:



Um das Resultat zu visualisieren, empfiehlt sich die Verwendung der GDK-Pixbuf-Bibliothek. Wie dies funktioniert, zeigt *image.c*. Da der Pixel-Puffer in einen gemeinsamen Speicherbereich gelegt wird, empfiehlt es sich jedoch, die Funktion *gdk_pixbuf_new_from_data* zu verwenden, siehe hier.

Bei Übersichtsbildern wie dem gezeigten ist 1000 eine gute Grenze, bei ins Detail gehenden Bildern muss diese aber noch eventuell deutlich nach oben gesetzt werden, damit mehr Präzision in der Darstellung gewonnen wird. Wenn Sie eine detailliertere farbliche Aufteilung wünschen, kann es auch reizvoll sein, statt 2 einen deutlich höheren Grenzwert zu nehmen (etwa 2^{16}). Die farbliche Gestaltung bleibt Ihnen vollkommen überlassen.

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.

Da die einzelnen Teilbereiche einen sehr unterschiedlichen Rechenaufwand benötigen, ist es nicht sinnvoll, die Gesamtaufgabe entsprechend des Fork-und-Join-Patterns zu Beginn aufzuteilen und dann rechnen zu lassen. Das würde dazu führen, dass einige der Prozesse früh fertig werden, während andere noch länger zu arbeiten haben.

Deswegen ist im Rahmen des Übungsblattes das Master/Worker-Pattern auf folgende Weise zu realisieren:

- Der Hauptprozess legt zu Beginn einen gemeinsamen Speicherbereich für den Pixel-Puffer an und eine einzige Pipeline, in der der Master schreibt und aus der die Worker konkurrierend lesen.
- Der Hauptprozess erzeugt die gewünschte Zahl an Workern und übernimmt selbst die Rolle des Masters.
- Die Worker lesen konkurrierend aus der Pipeline einen Auftrag, erledigen diesen und wiederholen dies, bis zum Eingabe-Ende bei der Pipeline. Dann terminieren sie.
- Der Master teilt das Gesamtproblem in Teilprobleme auf und schreibt die Teilproblem-Spezifikationen in die Pipeline.
- Sobald der Master damit fertig ist, schließt er sein Ende der Pipeline und wartet darauf, dass die Worker-Prozesse fertig sind. Dann ist der Pixel-Puffer fertig gefüllt und kann ausgegeben werden.

Diese Art der Umsetzung des Master/Worker-Patterns ist möglich, weil bis zu einer gewissen Grenze Schreib- und Lese-Operationen bei Pipes atomar erfolgen. Diese Grenze ist systemabhängig und liegt bei Linux bei 4096 Bytes, bei Solaris bei 5120 Bytes und bei OS X bei 512 Bytes (siehe `https://github.com/afborchert/pipebuf`). Die Spezifikation eines Auftrags benötigt jedoch nur eine kleine **struct** mit der Angabe der Zeile und Spalte, ab wo begonnen werden soll, sowie die Höhe und Weite. Dies passt in jedem Fall unter dem Limit.

Den gemeinsamen Speicherbereich können Sie mit dem Systemaufruf *mmap* einrichten, wobei Sie bei dem Parameter *prot* mit *PROT_READ*|*PROT_WRITE* Lese- und Schreibzugriff ermöglichen sollten und bei den *flags* die Kombination *MAP_SHARED*|*MAP_ANON* angeben. Die Angabe von *MAP_ANON* erlaubt den Verzicht auf einer Angabe eines Dateideskriptors, bei *filedes* ist dann -1 anzugeben. Stattdessen wird dann Speicher belegt, der nicht aus einer Datei stammt. (Dies ist äquivalent zum Abbilden der Gerätedatei */dev/zero*.) Zwar wird *MAP_ANON* nicht im POSIX-Standard angegeben, dieses Flag ist aber inzwischen universell auf allen modernen Unix-Varianten zu finden.

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

```
tar cvf mandelbrot.tar *.c *.h [mM]akefile
submit ss2 4 team [notes] mandelbrot.tar
```

Viel Erfolg!