ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert

9. Juli 2019
Blatt 11

Systemnahe Software II (SS 2019)

Abgabe bis zum 16. Juli 2019, 14:00 Uhr

Lernziele:

- Kommunikation und Synchronisierung über gemeinsame Speicherbereiche

Aufgabe 11: Mandelbrot Reloaded

Wie zuvor in der Aufgabe 4 geht es um die parallelisierte Visualisierung der Mandelbrotmenge. Zuvor wurde nur ein gemeinsamer Speicherbereich für den Pixel-Puffer verwendet, die Kommunikation selbst erfolgte jedoch über Pipelines. Im Rahmen dieser Aufgabe soll die Kommunikation und die dazu notwendige Synchronisierung nicht über Pipelines, sondern ebenfalls über den gemeinsamen Speicherbereich erfolgen.

Zur Synchronisierung sind Mutex- und Bedingungsvariablen der POSIX-Threads-Schnittstelle zu verwenden, wobei jeweils darauf zu achten ist, dass diese für die Nutzung durch verschiedene Prozesse konfiguriert sind. Dies leisten *shared_mutex* und *shared_cv* aus der Vorlesungsbibliothek.

Es steht Ihnen frei, das darauf aufbauende Modul *shared_domain* aus der Vorlesungsbibliothek zu verwenden oder dies selbst zu organisieren. Inzwischen wird von *shared_domain* (und dem zugehörigen Werkzeug *smrun*) auch die Reservierung von zusätzlichem Speicherplatz unterstützt, der u.a. für den Pixel-Puffer genutzt werden kann.

Wenn Sie es selbst organisieren möchten, dann dürfte dies in diesem Fall am leichtesten über einen einzigen Ringpuffer gehen (als Array von Aufträgen zusammen mit einem Mutex und zwei Bedingungsvariablen), bei dem der Master die Aufträge hineinschreibt und die Worker konkurrierend die Aufträge herausholen.

Die Parallelisierung soll wie zuvor nach dem Master/Worker-Pattern erfolgen. Wenn Sie *shared_domain* verwenden, muss es diesmal etwas anders organisiert werden:

- Die Datenstruktur für einen Arbeitsauftrag sollte auch eine Kodierung für das Ende der Aufträge vorsehen. Dies ist notwendig, damit der Worker weiß, wann er aufzuhören hat. (Zuvor stellte er das fest, wenn das Einlesen von der Pipeline nur 0 Bytes lieferte.)

- Worker lesen dann auf ihrem Kommunikationskanal einen Auftrag ein, führen diesen aus und schicken dann eine Nachricht an den Master mit ihrem *rank*. Dies wird solange wiederholt, bis das Arbeitsende signalisiert wird.

- Der Master verteilt an alle Worker jeweils einen ersten Arbeitsauftrag. Danach wartet der Master jeweils, bis einer der Worker sich auf seinem Kanal mit der Fertigstellung meldet, um dem Worker danach jeweils einen Folgeauftrag zu geben. Sobald die Arbeit erledigt ist, ist dies allen Workern mitzuteilen.

Am Ende ist ebenfalls der Einsatz eines Barriers sinnvoll, damit sichergestellt ist, dass alle Worker ihre Arbeit eingestellt haben, bevor der Pixel-Puffer ausgegeben wird.

Die Kommandozeilenverarbeitung und die Feststellung, ob genug Speicherplatz für die gemeinsamen Datenstrukturen (Pixelpuffer und ggf. die Parameter) reserviert ist, sollte dem Master-Prozess mit dem *rank* 0 überlassen werden. Wenn es Probleme gibt, dann kann der Master-Prozess Fehlermeldungen ausgeben und dann sich mit einem Exit-Code 1 verabschieden. Alle Prozesse sollten sich danach mit einem Barrier synchronisieren, d.h. die Worker beginnen erst, wenn alles korrekt vorbereitet ist. (Im Fehlerfalle werden die Worker durch *smrun* terminiert.) Hier ist ein Beispielverlauf:

```
theon$ smrun -np 4 pmandelbrot
Usage: pmandelbrot [-x value] [-y value] [-d value] [-w value] [-h value] [-m value] [-s value] [-P value] outfile
   -x value: center (real part)
   -y value: center (imaginary part)
   -d value: horizontal diameter
   -w value: width in pixels
   -h value: height in pixels
   -m value: maximal # of iterations
   -s value: color shift
   -P value: color progression
smrun: execution failed
theon$ smrun -np 4 pmandelbrot out.jpg
pmandelbrot: just 0 bytes extra space were provided but 147576 bytes are required
smrun: execution failed
theon$ smrun -np 4 -extra 147576 pmandelbrot out.jpg
theon$
```

Ihre Lösung können Sie wiederum mit Hilfe von *tar* verpacken und dann mit *submit* auf der Theon einreichen:

```
tar cvf mandelbrot.tar *.[ch] [mM]akefile
submit ss2 11 team [notes] mandelbrot.tar
```

Viel Erfolg!