Universität Hamburg
Fachbereich Informatik
Wissenschaftliches Rechnen
M. Kuhn, A. Fuchs

Übungsblatt 1 zur Vorlesung
Hochleistungs-Ein-/Ausgabe
im SoSe 2015
Abgabe: 17.04.2015

Dieses Übungsblatt dient der Übung der grundlegenden Ein-/Ausgabe Operationen in der Programmiersprache C.

Bei jeglichen Problemen oder Nachfragen benutzen Sie bitte die folgende Mailingliste:

`hea-15@wr.informatik.uni-hamburg.de`

Sie können sich unter folgender Adresse in die Mailingliste eintragen:

`http://wr.informatik.uni-hamburg.de/listinfo/hea-15`

1 Cluster-Kennung

Zur Bearbeitung der Übungsaufgaben benötigen Sie generell kein Konto auf unserem Cluster. Die Korrektur der Blätter erfolgt allerdings auf dem Cluster, der somit auch als Referenzsystem gilt, auf dem Ihre Abgabe final funktionieren muss.

Um ein solches Konto zu beantragen, schicken Sie eine E-Mail mit Ihrem vollen Namen und Ihrer E-Mail-Adresse in folgendem Format an `michael.kuhn@informatik.uni-hamburg.de`:

Max Mustermann <max.mustermann@domain.de>

Weitere Informationen zum Cluster finden Sie auf unserer Webseite im Beginners' Guide:

`http://wr.informatik.uni-hamburg.de/teaching/ressourcen/beginners_guide`

In den folgenden Aufgaben sollen Sie sich mit den grundlegenden Lese- und Schreiboperationen in C vertraut machen. In den Materialien auf der Webseite finden Sie ein bereits vorgegebenes Programmgerüst. Darin befindlich ist eine Funktion `calculate`, welche iterativ mathematische Operationen auf einer Matrix durchführt. Zum Ausführen des Programms müssen die Anzahl der Threads sowie der Iterationen übergeben werden.

In der Ausgabe können Sie die statistischen Auswertungen des Programmlaufs, wie die benötigte Zeit, den Durchsatz oder die IOPS nachvollziehen.

2 Checkpoint schreiben (90 Punkte)

Modifizieren Sie das gegebene Programm so, dass in jeder Iteration ein Checkpoint in die Datei `matrix.out` geschrieben wird; das Schreiben soll dabei parallel durch alle Threads erfolgen. Die Werte der Matrix sollen in jeder Iteration überschrieben werden. Sehen Sie sich dazu die Funktionen `write` und `pwrite` an und beachten Sie die parallele Umgebung (*Tipp: Thread-Safety*). Beschreiben Sie kurz die Funktionsweise beider Funktionen und begründen Sie Ihre Wahl für die gegebene Aufgabe. Integrieren Sie einen Header in die Ausgabedatei, der die Aufrufparameter des Programmlaufs – T (Threads), I (Iterationen) und I_c (die Nummer der zuletzt geschriebenen Iteration) – beinhaltet.
Implementieren Sie außerdem die Berechnung für den erreichten Durchsatz sowie die IOPS. Überlegen Sie sich hierbei auch, ob es sinnvoll ist, den Durchsatz und die IOPS auf Basis der Gesamtlaufzeit oder nur auf Basis der für die Ein-/Ausgabe benötigten Zeit zu berechnen.

3 Checkpoint lesen (60 Punkte)

Die Matrix wird derzeit statisch mit vorgegebenen Werten initialisiert. Implementieren Sie die vorgegebene Funktion `read_matrix`. Erweitern Sie hierzu die Eingabeparameter um den Pfad zu der vorhin geschriebenen Datei `matrix.out`. Sofern kein Pfad angegeben wird oder die Datei nicht existiert, soll die Matrix weiterhin statisch initialisiert werden; denken Sie an angemessene Fehlerbehandlung.

Es werden zwei hintereinander ausgeführte Programmläufe L_1 und L_2 mit den jeweiligen Aufrufparametern T_1, I_1, T_2 und I_2 betrachtet. Dabei wurde L_1 schon durchgeführt und hat einen Checkpoint mit den Werten für T_1, I_1 und I_c im Header geschrieben. Nun soll L_2 ausgeführt werden. Dabei soll der in L_1 geschriebene Checkpoint für L_2 weiterverwendet werden. Zunächst soll der Header analysiert und vier Szenarien betrachtet werden:

1. $T_1 \neq T_2$, wobei die Matrixgröße nicht von der Anzahl der Threads abhängt.
2. $I_c < I_1$
3. $I_c = I_1$ und $I_c > I_2$
4. $I_c = I_1$ und $I_1 < I_2$

Überlegen Sie sich und implementieren Sie sinnvolle und angemessene Vorgehensweisen für die ersten drei beschrieben Fälle. Begründen Sie Ihre Entscheidung.

Für den letzten Fall implementieren Sie das Einlesen der Werte aus der letzten geschriebenen Iteration. Mit diesen Werten soll im aktuellen Lauf (L_2) die Matrix initialisiert und ab der entsprechenden Iteration ($I_c + 1$) weiter berechnet werden.

4 Atomare Checkpoints (60 Punkte)

Implementieren Sie das atomare Schreiben des Checkpoints, d.h. selbst bei einem Programmabsturz während des Schreibens soll sich der Checkpoint trotzdem in einem konsistenten Zustand befinden. Konsistent bedeutet in diesem Zusammenhang, dass die geschriebenen Werte vollständig aus ein und derselben Iteration stammen und I_c den geschriebenen Werten zuzuordnen ist. Diskutieren Sie die Vor- und Nachteile von zwei möglichen Lösungsansätzen.

Abgabe

Erstellen Sie in dem Verzeichnis mit ihren C-Programmen eine Datei `antworten.txt` mit Ihren Antworten zu den Fragen. Packen Sie ein komprimiertes Archiv (`.tar.gz`) aus dem sauberen Verzeichnis (ohne Binärdateien).

Senden Sie das Archiv per E-Mail an `hea-abgabe@wr.informatik.uni-hamburg.de`.

Um das Archiv zum Versenden auf ihren Rechner zu kopieren, können sie `scp` verwenden. Näheres dazu finden Sie auf unserer Webseite.