Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
Michael Kuhn, Ruben Felgenhauer

Übungsblatt 2 zur Vorlesung
Hochleistungs-Ein-/Ausgabe
im SoSe 2019
Abgabe: 26.04.2019, 10:00

In den folgenden Aufgaben werden Sie das in den Materialien bereitgestellte Programm analysieren und optimieren. Modifizieren Sie das Programm zuerst so, dass nach jedem `pwrite`-Aufruf eine Synchronisation mittels `fsync` ausgeführt wird, um sicherzustellen, dass die Checkpoints auch tatsächlich auf das Speichergerät geschrieben werden. Speichern Sie das angepasste Programm als `checkpoint-fsync.c`. Analysieren Sie nun das Programm unter Zuhilfenahme der folgenden Werkzeuge.

1 `iostat` (30 Punkte)

Lassen Sie sich während des Programmlaufs die Statistiken mittels `iostat` ausgeben. (Hinweis: `iostat` ist auf dem Cluster nur auf den Rechenknoten verfügbar.) Beschreiben Sie ausführlich, was sie in der Ausgabe sehen können. Lassen Sie sich anzeigen, wie viele MB pro Sekunde geschrieben werden und vergleichen Sie diesen Wert mit Ihrer Ausgabe. Protokollieren Sie die Ergebnisse und erklären Sie eventuelle Abweichungen.

Hinweis: Verwenden Sie ein lokales Verzeichnis wie z. B. `/tmp`, da die Home-Verzeichnisse mittels NFS eingebunden sind.

2 VampirTrace-Analyse und Optimierung (150 Punkte)

In dieser Aufgabe soll das auf dem Cluster verfügbare Leistungsanalysewerkzeug Vampir[1] verwendet werden, um das Laufzeitverhalten des Programms auszuwerten. Gehen Sie dazu wie folgt vor:

1. Kompilieren Sie das Programm mit dem VampirTrace-Compiler-Wrapper `vtcc`. Laden Sie zuerst mit `spack load -r vampirtrace` das VampirTrace-Paket.
2. Setzen Sie die beiden Umgebungsvariablen `VT_IOTRACE_EXTENDED=yes` und `VT_MAX_FLUSHES=0` und führen Sie das Programm aus. Hierbei entstehen so genannte Spurdaten (Traces) für die spätere Visualisierung.
3. Visualisieren Sie die Spurdaten in Vampir (`vampir checkpoint.otf`). (Hinweis: Hierfür können Sie das auf dem Cluster installierte X2Go verwenden, das eine deutlich höhere Leistung als das übliche X-Forwarding bietet.)

Beschreiben Sie nun die wesentlichen Merkmale des Programms und legen Sie Ihren Beschreibungen aussagekräftige Screenshots bei. Gehen Sie dabei besonders detailliert auf das E/A-Verhalten ein. Bewerten Sie dieses und begründen Sie Ihre Meinung.

Überlegen Sie sich nun, wie Sie das Programm hinsichtlich der E/A-Leistung verbessern können. Implementieren und erläutern Sie Ihre Optimierungen als `checkpoint-optimiert.c` (achten Sie auf gute Dokumentation). Beachten Sie hierbei, dass Ihre optimierte Version `fsync` nicht notwendigerweise nach jedem `pwrite`-Aufruf ausführen muss. Es muss allerdings für jede Iteration sichergestellt werden, dass sie auf das Speichergerät geschrieben wurde.

[1] `http://www.vampir.eu/`

3 Leistungsvergleich (60 Punkte)

Werten Sie nun das Programm mittels aussagekräftigen Messungen aus. Nehmen Sie dabei Messungen für die folgenden Varianten vor:

1. Das Originalprogramm aus den Materialien,
2. das Originalprogramm mit Synchronisation
3. und das von Ihnen optimierte Programm.

Verwenden Sie dazu jeweils 1–12 Threads und wählen Sie eine geeignete Iterationszahl aus, wobei die Mindestlaufzeit jeweils 120 Sekunden betragen soll. (Hinweis: Für die unterschiedlichen Messungen können unterschiedliche Iterationszahlen gewählt werden. Als Richtwerte empfehlen sich folgende Iterationszahlen: Originalprogramm 100.000, Originalprogramm mit Synchronisation 4.000 und optimiertes Programm 10.000.)

Visualisieren Sie Ihre Ergebnisse in geeigneten Diagrammen und schreiben Sie jeweils Ihre Erkenntnisse auf. Die IOPS- und Durchsatzwerte der drei Varianten mit unterschiedlichen Threadzahlen sollen dabei möglichst einfach verglichen werden können.

Abgabe

Erstellen Sie im Verzeichnis mit den drei Programmen (`checkpoint.c`, `checkpoint-fsync.c` und `checkpoint-optimiert.c`) außerdem eine Datei `antworten.pdf`. Packen Sie ein komprimiertes Archiv (`.tar.gz`) aus dem sauberen Verzeichnis (ohne Binärdateien).

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