Übungen zur Vorlesung

Paralleles Höchstleistungsrechnen

Dr. S. Lang, J. Pods

Abgabe: 5. Februar 2013, 12 Uhr, per E-mail an `jurgis.pods@iwr.uni-heidelberg.de`

Übung 25 N-Körper-Problem mit PThreads und CUDA (10 Punkte)

Auf der Hompage finden Sie ein Update des Archivs `nbody.zip`. Hier sind die Kernel `nbody_pthread.c` und `nbody_cuda.cu` unter Verwendung von `PThreads` bzw. CUDA ausimplementiert, außerdem enthält das Archiv die Lösung des MPI-Kernels `nbody_mpi.c`. Im Pool können Sie wie immer mit `make` kompilieren, damit wird auch das CUDA-parallele Programm erstellt. Im Pool sind Nvidia GPUs eingebaut, und zwar die GeForce 9400 GT (`lspci | grep VGA` für mehr Infos). Machen Sie sich mit den neuen Codes vertraut.

Aufgaben

1. Führen Sie Simulationen mit den gleichen Parametern, die Sie für das letzte Übungsblatt gewählt haben, aus. Mehr als 2 Threads machen im Pool wie gehabt wenig Sinn. Messen Sie wiederum die FLOPs und vergleichen Sie die Performance der sequentiellen und parallelen Varianten. Praktisch sind Plots der FLOPs über der Problemgröße bzw. des Speed-Ups. Welche Variante hat bei Ihnen am Besten funktioniert?
2. Kommentieren Sie kurz die *execution configuration*, d.h. Erklären Sie die drei Parameter in den spitzen Klammern beim Kernelaufruf `acceleration_kernel<<<dimGrid,dimBlock,BLOCKSIZE *sizeof(float4)>>>(j,xd,ad);` in Zeile 98.
3. Warum wurde der Parameter $\epsilon 2$ im Plummer-Potential eingeführt? Warum kann nicht der ursprüngliche Wert von $1e-14$ verwendet werden (siehe die Vanilla-Variante), falls auf der GPU gerechnet wird?
4. Kommentieren Sie kurz die Genauigkeit der Ergebnisse eines Simulationslaufs mit den unterschiedlichen sequentiellen und parallelen Varianten, aber gleichen Paramatern N, ... Nutzen Sie dazu das Skript `fuzzy_diff` und analysieren Sie die Ausgabe-Dateien zu mehreren festen Zeitpunkten. Nehmen die Unterschiede mit steigender Zeit zu, d.h. akkumulieren sich die Differenzen? Was ist die höchste „gemessene“ Abweichung, die Sie erhalten haben?

Freiwilliger Zusatz

Im Makefile sind die Optionen `--use_fast_math` für den CUDA-Kernel und `-O3 -ffast-math -funroll-loops -fexpensive-optimizations` für die anderen Kernel gesetzt. Lesen Sie nach, was diese Flags bewirken und überlegen Sie, welchen Einfluss Sie auf das Ergebnis haben könnten. Wiederholen Sie einige Simulationen ohne diese Flags (optimiert nur mit `O3`), und diskutieren Sie die Performance in diesem Fall.

Die Karte im Pool hat theoretisch eine Leistungsobergrenze von etwa 60 GFLOPs. Erreicht wurden in diesem Test aber nur etwa 20 GFLOPs. Versuchen Sie, diese Rate zu verbessern. Ein Punkt ist klar: Die CUDA-Rechnung im Pool macht real mehr Flops als für die Messung angenommen. Hier kann man also die tatsächliche FLOP-Rate messen. Dennoch könnten Sie sich noch weitere Optimierungen überlegen.

Hinweise

- Der Pfad zum CUDA-Compiler ist im Makefile explizit gesetzt, er liegt unter `/usr/local/cuda/bin/nvcc`. Wenn Sie auf anderen Rechnern kompilieren wollen, müssen Sie das `Makefile` anpassen oder den Pfad lokal setzen: `export PATH=${PATH}:/usr/local/cuda/bin/`. Im Pool

müssen Sie außerdem in der Shell den Pfad zur Bibliothek `libcudart.so.4` setzen: `export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/local/cuda/lib64/`. Dies ist zwar nicht ganz sicher, auf Grund fehlender Root-Rechte kann ich die passende Konfiguration im Moment aber nicht vornehmen.

- Die CUDA-Rechnungen können sehr schnell sein. In diesem Fall werden als Wall-Time $0s$ gemessen und die Flop-Rate ist ein `inf`. Achten Sie auf hinreichend großes N, um sinnvolle Werte zu erhalten.

Semester-Abschluß

Dieses war das letzte Aufgabenblatt in diesem Semester. Insgesamt gab es 165 Punkte, d.h. zur Zulassung für die Prüfung oder den unbenotenen Schein waren 82 Punkte notwendig.