Serie 2

Aufgabe 2.1 (Cache Benchmark)
Beim Verarbeiten großer Datenmengen ist es wichtig Algorithmen so zu implementieren, dass sie möglichst effizient die vorhanden Caches ausnutzen.
Um uns die Vorteile klar zu machen, die ein Cache uns liefert, wollen wir ein kleines Testprogramm schreiben, dass für unterschiedliche Eingabegrößen $n \in \mathbb{N}$ die Addition zweier Vektoren $x, y \in \mathbb{R}^n$ durchführt und dabei messen wie viele Daten wir pro Sekunden verarbeiten können.
Implementieren Sie hierzu in der Datei `cachebench.c` die Funktion `vector_add`, die für ein $\alpha \in \mathbb{R}$ die Operation

$$y \leftarrow y + \alpha x$$

ausführt.
Wenn das Programm `cachebench` gestartet wird, erwartet es einen Komandozeilenparameter, z.B.

```
./cachebench 128
```

Dabei werden die Vektoren dann aus dem $\mathbb{R}^{128}$ gewählt. Weiterhin wird diese Vektoraddition so oft ausgeführt, dass bei jeder Vektorgröße stets dieselbe Anzahl an Rechenoperationen und Datentransfers statt findet.
Variieren Sie die Größe der Vektoren n von 2^0 bis 2^{26}. Tragen Sie die gemessenen Transferraten in ein Diagramm ein. Ermitteln Sie auf diese Weise ungefähr die Größe Ihres L1-, L2- (und L3-)Caches und vergleichen Sie die gemessenen Werten mit den Herstellerangaben Ihres Prozessors. Machen Sie sich hierbei deutlich welche Größen das Programm in die Datei `data.dat` schreibt.

Aufgabe 2.2 (Matrix-Matrix-Multiplikation)
Seien $n, m, p \in N$ sowie Matrizen $A \in \mathbb{R}^{n \times m}, B \in \mathbb{R}^{m \times p}$ sowie $C \in \mathbb{R}^{n \times p}$ gegeben. Matrizen wollen wir in *column-major-order* speichern. Das bedeutet der Eintrag A_{ij} wird durch `A[i + j * n]` adressiert.

(a) Implementieren Sie in der Datei `matmul.c` die Matrixmultiplikation $C \leftarrow C + AB$ zunächst in naiver Weise mittels drei ineinander geschachtelter Schleifen, die die Schleifenvariablen `i, j, k` benutzen sollen. Der Algorithmus funktioniert unabhängig davon in welcher Reihenfolge die Schleifen geschachtelt werden. Testen Sie daher alle möglichen Kombinationen aus und diskutieren Sie, warum die jeweiligen Konfigurationen gut oder schlecht sind.

(b) Überlegen Sie sich (ohne es zu implementieren), ob die Operation $C \leftarrow C + AB^T$ besser oder schlechter hinsichtlich der Ausnutzung der Caches geeignet ist.

(c) Wir Nehmen an, die Dimensionen der Matrizen seien ein Vielfaches einer Blockgröße von `BLOCKSIZE` $\in \{32, 64\}$. Entwerfen Sie eine Variante der Matrixmultiplikation, die stets auf `BLOCKSIZE` $\times$ `BLOCKSIZE` großen Teilmatrizen operiert. Versuchen Sie dabei die zeitliche Lokalität mit dem Cache auszunutzen.

Website: https://lms.uni-kiel.de/auth/RepositoryEntry/3682631699/CourseNode/102508131085097
Abgabe: Bis Mo, 09.11.20, 16:00 Uhr über das OLAT.