



---

**Übungen zur Vorlesung  
“Database Systems and Modern CPU Architecture”**

Sommersemester 2009

Jan Rittinger (jan.rittinger@uni-tuebingen.de)

**1. Übungsblatt**

Ausgabe: 4. Mai 2009 · Besprechung: 6. Mai 2009

**Aufgabe 1: Hardware-Spezifikationen**

**Finden Sie die Hardware-Spezifikationen Ihres eigenen Rechners heraus.**

Im Laufe dieses Semesters werden wir uns insbesondere mit CPU- und Cache-Verhalten auseinandersetzen. Je nach Hardware können sich hier einige Aha!-Effekte ergeben und auch durchaus unterschiedliches Verhalten beobachtet werden. Von besonderem Interesse werden für uns sein:

- Prozessor (Hersteller, Bezeichnung, Taktfrequenz, Anzahl, ...)
- Caches (Anzahl, Größe, Zugriffszeiten, ...)
- Hauptspeicher (Größe, Bandbreite, ...)

Schön wäre es, wenn wir während des Semesters das Verhalten von verschiedenen, möglichst unterschiedlichen Systemen vergleichen könnten. Interessant sind dafür besonders (a) Maschinen (Laptops), die wir „live“ in der Übung ausprobieren können und (b) exotische Maschinen, die vielleicht nicht jedermann zuhause stehen hat.

**Aufgabe 2: Loop Swapping**

Nehmen wir an, eine Anwendung möchte die (zugegeben in dieser Form nicht wirklich sinnvolle) Operation ausführen, alle Einträge einer Matrix aufzusummieren. Das läßt sich z. B. durch folgendes Stück Code bewerkstelligen:

```
1  sum = 0;  
2  for (r = 0; r < rows; r++)  
3      for (c = 0; c < cols; c++)  
4          sum += src[r][c];
```

Alternativ läßt sich die gleiche Aufgabe natürlich auch lösen, indem man die beiden Schleifen vertauscht ausführt, also die Matrix *spaltenweise* (anstelle *zeilenweise*) abarbeitet:

```
1  sum = 0;
2  for (c = 0; c < cols; c++)
3      for (r = 0; r < rows; r++)
4          sum += src[r][c];
```

### Welche Auswirkung auf die Laufzeit erwarten Sie durch diese Änderung des Codes?

Auf der Kurswebseite finden Sie ein kleines C-Programm, das im Kern aus zwei Abschnitten besteht, die die obigen zwei Varianten implementieren. Die Größe der Matrix läßt sich dabei in Form von Kommandozeilenparametern übergeben, z. B.

```
$ ./loop-swapping 100 1000000
```

### Untersuchen Sie die Laufzeit beider Varianten in Abhängigkeit der Anzahl Zeilen und Spalten Ihrer Matrix. Interpretieren Sie Ihre Beobachtungen.

Einige Hinweise dazu:

- Der Code auf der Webseite sollte einigermaßen POSIX-konform sein und daher auf den meisten Architekturen kompilieren. Gegebenenfalls helfe ich gerne, den Code auf Ihre Plattform zu portieren.
- Schalten Sie beim Kompilieren bitte sämtliche Debugging-Optionen Ihres Compilers ab. In der ausführbaren Datei hinterlegte Debugging-Informationen könnten die Hardware-sensitiven Effekte überlagern. Beim Testen hat sich bei mir das Anschalten sämtlicher Optimierungen (GCC: `-O9`) als vorteilhaft erwiesen.
- Deaktivieren Sie bei Ihren Messungen bitte konkurrierende CPU-intensive Prozesse.
- Um möglichst lange Laufzeiten (und damit aussagekräftige Messungen) zu erzielen, sollten Sie die Zeilen- und Spaltenanzahl möglichst groß wählen, jedoch nicht größer als der zur Verfügung stehende Hauptspeicher. Auf meiner 4 GB-Maschine hat sich  $Zeilen \times Spalten = 600000000$  als sinnvoll erwiesen. Ändern Sie zwischen den Messungen dann nur das *Verhältnis* der beiden Parameter, halten Sie jedoch das Produkt konstant.