Institut für Informatik — TUM
Scientific **C**omputing in **C**omputer **S**cience
R. Wittmann

WT 2015

Tutorial
(Advanced Programming)
Worksheet 10:

In this worksheet we will look into some basic methodologies to optimize an algorithm for better hardware efficiency. For this, we will take one of the best optimized problems to date, namely the matrix-matrix multiplication. Please download the stub implementation `mm_mult.cpp` from moodle and work through the following assignments. The stub implementation offers a predefined test setup to evaluate the performance of your routines. However, please make sure that you have C++11 features enabled in your compiler settings. In the given stub implementation you will also find an example implementation of a matrix-matrix multiplication in `mm_mult_rck` which is close to its mathematical definition:

$$C = A \cdot B, \text{ with } C \in \mathbb{R}^{rows \times columns}, A \in \mathbb{R}^{rows \times k}, B \in \mathbb{R}^{k \times columns}$$

Assignment 1: Roofline Model

In this assignment we will do a performance analysis of the most basic reference matrix-matrix multiplication. You can find the implementation inside the provided stub implementation. We will now define some properties of a system on which this algorithm should run:

- single core CPU: 2 GHz
  - vectorized double-precision floating point unit
    - width 4
    - +,-,* (2 cycle latency,no pipelining)
    - fused multiply add: a+=b*c (3 cycle latency,no pipelining)
  - 2 load and 1 store units
    - cache access: pipelined (5 cycle latency)
    - memory access: no pipelining (10 cycle latency)
  - Cache: 2 GHz
  - Main Memory: 800 MHz
  - both types of memory can load and store 2 vector registers in each memory cycle

First, we need to determine some performance properties of our floating point unit:

- What is the theoretical maximum floating point performance for basic arithmetic operations?

- What is the theoretical maximum floating point performance for fused multiply add operations?

Second, we have to determine some performance properties of our memory system (cache + main memory):

- What is the theoretical bandwidth of our memory?
- How much bandwidth can be saturated by our cpu?

We are now able to determine the theoretical performance of our reference matrix matrix multiplication:

- How much floating point operations are necessary for our computation?
- How much values need to be transferred from/to memory in the worst case (e.g. no optimizations)?
- What is the computational intensity of this algorithm with respect to read and writes?
- What performance can we expect from this algorithm when cache is not involved? What is the limiting component in our system?
- What performance can we expect from this algorithm when all matrices fit in cache? What is the limiting component in our system?

Assignment 2: Code transformations - Loops and Branches

For these first transformations, please disable any form of compiler optimizations for now. Otherwise, you may not be able to see the expected performance differences. The corresponding test function is provided through `testFunction`.

1. Explain the used memory layout, e.g. how are the rows and columns stored in memory and in what order are they read and written in the involved matrices.
2. Analyze the code and eliminate the primary performance bottleneck in the provided implementation. How does the performance improve?
3. The loop ordering in the provided implementation is (row, column and k) = (rck). Run some experiments with different loop orderings and check the performance results. Implement at least the following loop orderings: (column, k, row) and (k, row, column). As homework, you may also experiment with all 6 different possible loop combinations.

Assignment 3: Compiler optimizations

Finally, you may now activate compiler optimizations again. Increase the overall optimization level (O0,O1,O2,O3) and look how the overall performance changes. Which loop orderings benefit most/least in your case? Do you have an explanation for this behavior?

Now to the fun part: compare your results with those of your fellow students! Who can perform the largest matrix multiplication in the shortest amount of time?