Institut für Informatik — TUM
Scientific Computing in Computer Science
K. Rahnema, M. Schreiber

WT 2012

Tutorial (Advanced Programming) Worksheet 7:

Matrix multiplications are among the most basic and most frequently used linear algebra operations. Among others, they are used for higher order computations in discontinuous Galerkin methods to compute time-step updates.

In this worksheet we consider especially matrices which have a relatively large size exceeding L3 cache size.

Assignment 1: Code skeleton

Develop a skeleton which handles the program parameters in the following way: The first parameter specifies the matrix-size. You are free to assume, that the matrix size is e. g. a power of 2 with a minimal size of e. g. 8×8. The second program parameter specifies the type of matrix multiplication to run (see assignments below).

In case that invalid parameters are given, some helpful information has to be be printed to the terminal.

Write two helper functions *start_time_measuring* and *stop_time_measuring* measuring the elapsed time. The function *stop_time_measuring* should return the elapsed time.

Assignment 2: Straightforward matrix-matrix multiplication

Develop a matrix-matrix multiplication which multiplies two matrices by using the typical three for-loops. Think about the way how you iterate over the arrays (e. g. row or column in the innermost loop) to optimize the data access on current CPU generations.

Assignment 3: Block-wise matrix-matrix multiplication

Cache-blocking should be used in this assignment to gain more performance for your matrix-matrix multiplication. Sketch the matrix-matrix multiplication on a sheet and think about the L1 cache utilization. Do you see any problems regarding cache utilization? Create an additional version of the matrix-matrix multiplication which uses blocking in the innermost loops by adding 2 additional for-loops. The size of the innermost blocks should be specified as a 3rd program parameter.

Use the "Performance analysis tools for Linux" or cachegrind[1] to compare the cache hit-rate for the block-wise matrix-matrix multiplication.

[1]`http://valgrind.org/docs/manual/cg-manual.html`

Assignment 4: Performance optimized matrix-matrix multiplications

Activate compiler options to support SIMD instructions (e. g. -msse4.2). Use the perf tool or cachegrind again to compare the results with the previous matrix-matrix multiplications.

Questions:

Answer the following questions:

- Assignment 2: When looking at different FLOPs created during the matrix-matrix multiplication for different sizes of blocks, can you guess the L2 cache size?
- Write down one operation of the matrix-matrix multiplication with pseudo-SIMD instructions. What is better in terms of performance? A vector-scalar multiplication or a vector dot-product?
- Evaluate the quantitative complexity of the $O(n^3)$ operations in terms of CPU-cycles.
- Which optimizations for sparse Toepliz matrices[2] (e. g. 4 on the main diagonal, -1 on the upper and lower secondary diagonal and two other diagonals -1) are possible?
- What challenges for sparse matrices exist when SIMD operations should be used?

[2] `http://en.wikipedia.org/wiki/Toeplitz_matrix`