Scientific Software
Session 5: Fortran 95, Matrix competition

Pieter Appeltans, Emil Loevbak& Joris Tavernier Leuven, November 7, 2018

Introduction

In this session you must write a Fortran 90/95 program that multiplies two $N \times N$ matrices in double precision as fast as possible. The complexity of this calculation is $\mathcal{O}(N^3)$, but depending on the implementation, there will be huge differences in execution time. In this exercise session the effect of following points will be examined:

- the way Fortran stores matrices;
- the memory hierarchy of a PC[1];
- the used compiler and compiler options (check the documentation for the available optimisation flags) [2];

During the session

Download following files from Toledo:

- `dmr.f90`: program that executes the timings of the different implementations and returns a control output for every implementation.
- `matrixop.f90`: module that contains the different implementations of the matrix-matrix multiplication. You should complete these implementations yourself. The description of these implementations can be found in the following section.

Complete `matrixop.f90` and compile and link both files. Experiment with the size of the matrices and blocks. Study the result.
Remark 1: To compare your results easily, you must execute the experiments on the PC's in the PC rooms of the department.
Remark 2: Because the exercise session is closely related to the assignment, you should work individual.

Description different implementations

1. `a_maal_b_ijk, a_maal_b_ikj, a_maal_b_jik, a_maal_b_jki, a_maal_b_kij, a_maal_b_kji`:
   With three nested loops: with `ijk` we mean that `i` changes in the outermost loop and `k` in the innermost.
2. `a_maal_b_ikj_vect, a_maal_b_jki_vect, a_maal_b_kij_vect, a_maal_b_kji_vect`:
   With two nested loops: you replace the inner loop by a vector operation.
3. `a_maal_b_ij_dot_product, a_maal_b_ji_dot_product`:
   With two nested loops in which you use the `dot_product` function.
4. `a_maal_b_transp_ij_dot_product, a_maal_b_transp_ji_dot_product`:
   Same as before, but where you first transpose A explicitly.
5. `a_maal_b_blas`:
   By means of a BLAS routine: do not forget to link BLAS itself.
6. `a_maal_b_blocks`:
   With a blockwise multiplication (for convenience, suppose that the block size divides N).
7. `a_maal_b_matmul`:
   Use the intrinsic `matmul` function that is also based on a blockwise multiplication.
8. Another variant that has a better complexity then $\mathcal{O}(N^3)$

[1] `https://en.wikipedia.org/wiki/Memory_hierarchy`
[2] Nagfor: `https://www.nag.com/nagware/np/r61_doc/nagfor.html#OPTIONS`,
Gfortran: `https://gcc.gnu.org/onlinedocs/gcc-4.9.3/gcc/Optimize-Options.html`,
Ifort: `https://software.intel.com/en-us/fortran-compiler-developer-guide-and-reference-optimization-options`

Task

In this assignment we build on the exercise session. Answer following questions. Try to use the bullet points from the introduction in your answer.

- Is there a difference between the available compilers (ifort, nagfor, gfortran)?
  Note: Starting from here you should use gfortran
- Which optimisation flag gives the best results?
- Which (and why) method with three nested loops is the fastest?
- Which (and why) method with dot products (both A and A^T) is the fastest? Why would you transpose A in implementation 4?
- Does BLAS (implementation 5) utilize the different cores in your machine and how did you check this?
- What is the influence of the block size in implementation 6? What is the optimum size given the amount of L2 cache (*tip*: `$ lscpu`) in your machine? Does it still matter which routine you use for the multiplications on block level?
- Discus briefly (maximal 1 or 2 sentences) the practical relevance of the algorithm in implementation 8.
- Compare the optimal method with three loops, the optimal method that uses dotproducts, the blockwise multiplication with optimal blocksize, the BLAS-routine and the intrinsic `matmul` function. Plot one or more simple graph(s) that give the MFlop/s in funciton of N, for N in a meaningful range. Like mentioned in the previous exercise session, the the execution time of the same method can vary over different runs. Therefore do for each N at least three timings. Your figure should not only give the average value, but also the minimal and maximal value.

Method

- Measure the processor time needed for the matrix multiplication for the above methods and for N in an interesting range. You do (at least) three time measurements per value of N on a machine in the computer classes and calculate the number of Mflop/s for each of these measurements.
- Write those numeric results to one or multiple data file(s).
- Plot the results with your favorite plotting program (for example Matlab, Octave, Gnuplot or similar). Choose the way you present your results (X and Y range, scale, line and/or point types, color, ...) such that the graph(s) are clear and allow an easy interpretation of the results. An example in Matlab will be provided on Toledo.

Remark: Check that there are no other users on the PC, when experimenting. Use the commandline commands `users` and `htop`

Practical information

You should submit the following:

- The Fortran code that you have used to generate your results. Give at the start of your code the commands you used to compile and link the files. When there are more then a few instructions, use a Makefile.
- A PDF with the desired figures and answers. Your discussion has to fulfil following points:
  - Answer the question in the Task section
  - Include the time you spend on the assignment
  - Make a separate section for your figures and results.
  - Focus on the *interpretation* of the results!
  - Limit your discussion (excluding figures and results) to 2 pages.
  - Specify which computer you used. Include in your answers (where necessary) the specifications of the PC (`$ lscpu` en `$ cat /proc/cpuinfo`). You could for example compare the Mflop/s with the theoretic maximum of the machine (look up this information on the internet using the serial number of your processor)

Hand in your solution in zip format with as name `lastname_firstname_studentnumber.zip` on Toledo before 21 November at 14h.