PDC Summer School 2010
High-performance: Unofficial MATMUL competition

August 24, 2010

Following the exercise on performance engineering and OpenMP parallellization we present a competition for the fastest implementation of matrix-matrix multiplication. The prize is a unique coffee cup (seen below), as well as honor and good graces!

1 Objective

The winner will be the person, or possibly group of at most two students, who provides the fastest MATMUL routine for a range of matrix sizes. We will let all matrices be $N \times N$, and let the number of flops be $2N^3$.

2 Rules

1. Deadline: Wednesday Aug. 25:th 2010, 18:00 CET.
   EXTENDED to midnight, i.e. 23:59 Aug. 25:th.
2. You may use C/C++/C99, FORTRAN (any version) or assembly language.
3. The official machine for judging the results will be a Harpertown node on Ferlin. **All 8 cores on this machine may be used**. The official compiler will be Intel (version 11.1).
4. The submitted code must execute without any errors and not leak any memory.
5. Source code must be submitted. Benchmarking will be done manually. The code must be the original work of the authors.

6. The program must take the matrix size as the first and only argument on the command line, and must output the matrix size, the elapsed time and the flop/s estimate like this:

```
a11c21n09:hpc_matmul > ./my_matmul 1000
1000    0.862177        2.319710E+09
```

7. The accuracy attained in the multiplication must correspond to 64-bit (double precision). Check your results!

8. No external, or library, subroutines may be called. Certain compilers have been known to recognize matrix multiplication code, replacing it with a call to a vendor-tuned library. In this (very rare) case, your entry will be considered invalid in this competition.

9. In C, the input matrices must be row major. In Fortran, column major. That is, any reordering of matrix elements (such as transposition or recursive block partitioning) must be done inside the timed multiplication routine.

10. The input matrices, i.e. A and B in $C \leftarrow A \times B$, may not be altered. That is, the input matrices must be `const` as in this C example declaration:

```
void mul(double* result, const double* A, const double* B, int N);
```

11. You may *not* execute code that tampers with the system in any way, including the clock and timing facilities.

12. Scoring will be done on the sizes $N = 400$, 800, 1200, 1600, 2400, 3200, 4800, and 9600. You may not specialize code for any or all of these sizes. That is, the same code should execute for all sizes. Do not hard-code for any or all of these sizes. You may assume that all sizes are divisible by some factor and use that in the implementation.

13. Scoring: For each matrix size the contestant with the most efficient code gets one point. The winner is the contestant with the most points. To break a tie, the one who scored a point on the largest matrix wins.

14. Common sense applies in extension to these rules. Seriously.