LabX-10, BY 17 December 2021

How to do the exercises in this set

We are now increasing the usage of the bash scripts and pick up additional utility commands. Please consult the material in the Files section of canvas for this course, it may help with the syntax in the examples of the exercises. Use the C book of Kernighan & Ritchie (see Pages/Textbooks on canvas[1]). In canvas Files you have a summary of C language in slides file 3.1_Linux-C[2] . Also use the summary slides in file 3.1_Linux_Bash[3], the summary or the Makefile syntax is in file 3.6_Linux-Make[4].

This exercise is for CUDA and matrix multiplication.

On the delivery of your results: When you are ready with your program assignment, please create a directory with your name and copy the files there:

```
mkdir /trinity/home/LINUX_SUPERCOMPUTERS/yourname/   #it should exist already from exe1

mkdir /trinity/home/LINUX_SUPERCOMPUTERS/yourname/labx7

cp yourprogram /trinity/home/LINUX_SUPERCOMPUTERS/yourname/labx7/yourprogram
```

After you have finished copying submit also to canvas, so we can grade the arrivals. You do not have to send us e-mails when you complete this exercise.

Of course, if there are any problems or questions, please email me and/or Andrey:

"Igor Zacharov" I.Zacharov@skoltech.ru

"Andrey Kardashin" Andrey.Kardashin@skoltech.ru

[1] http://cslabcms.nju.edu.cn/problem_solving/images/c/cc/The_C_Programming_Language_(2nd_Edition_Ritchie_Kernighan).pdf

[2] https://skoltech.instructure.com/files/257760/download?download_frd=1

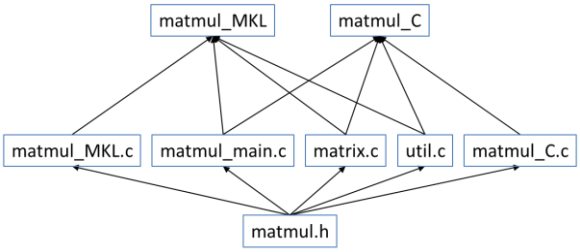
[3] https://skoltech.instructure.com/files/262799/download?download_frd=1

[4] https://skoltech.instructure.com/files/267631/download?download_frd=1

Matrix multiplication project

Copy the file from sandbox ~/../LINUX_SUPERCOMPUTERS/programs/lab10_CPU.tar.gz to your directory on Zhores.

1) The project structure is described by a make file and it is shown on the right. There are two executables, one which is using the "naïve" matrix multiplication (matmul_C) with a triple for and another that calls DGEMM from Intel scientific library MKL (matmul_MKL).



All executables are obtained with `make` (or `make all`). Individual executables can be obtained with `make matmul_C` and `make matmul_MKL` commands. This is visible from the structure of the Makefile. The meaning of different files is as follows:

The matrix.c contains the storage allocation and initialization parallelized with OpenMP. Note the code for the allocation of 2-dimensional matrices. It consist of these steps (see matrix.c):

After this initialization the matrices can be used as if declared with `double a[n][n]`:

```
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
            for(k=0; k<n; k++)
                c[i][j] += a[i][k]*b[k][j];
```

```
typedef matrix_t **double;
typedef rowmat_t  *double;   //in matmul.h

int storage_size = n*(n+1)*sizeof(double);
matrix_t a = _mm_malloc(storage_size, 64);
rowmat_t ra = a+n; //ra->first element of 2D array
// initialize the position of each row:
for(i=0; i<n; i++) a[i] = ra + n*i;
```

a) check the validity of this approach by running the commands and comparing the matrix print out. This can be done looking at the results of the multiplication for small size matrices (eg. n=8). The program accepts 2 types of arguments (see file util.c):

```
usage: matmul_C [-l <1/2/3/m>] [-s <size>]
```

```
tar -xf matmul.tar.gz
source setenv.sh
make
setenv OMP_NUM_THREADS=1
setenv MKL_NUM_THREADS=1
./matmul_C -s 8
./matmul_MKL -s 8
```

The –l flag puts **double** NxN matrices inside the cache level (1, 2, 3) or in memory (m). Alternatively, explicit matrix size NxN (with –s N) can be specified.

The matrices for the comparison are printed out if N <=10. Make the screen conveniently large. The resulting C matrix when computing matmul_C (the 3-x for loop) should be equal to matmul_MKL when using the cblas_dgemm. Look for two cases: PRECISION DOUBLE and PRECISION SINGLE. This is regulated in the Makefile (The PRECISION variable setting). To compare between SINGLE and DOUBLE you have to recompile everything (change Makefile, make clean; make).

Also: the sizes of the matrices that fit into the caches l (1/2/3 and memory) for SINGLE precision are wrong in matmul.h file. What should be the size of the single precision square matrices such as to fit into the caches?

Recalculate and correct the default matrix sizes for single precision in file matmul.h !

b) A quiz question:

There is a difference in results which you print (when n<=10) if you set OMP_NUM_THREADS >1

WHY? (make appropriate comment/warning in the file responsible for the effect and correct such that the resulting C matrix is the same independent of OMP_NUM_THREADS setting).

c) Testing larger size: run make test. Write the performances for different #threads in a table:

Таб. 1 Comparison Performance [MFLOP/s] for "3-x for" matrix multiplication and cblas_Xgemm (X=d or c).

program (N = 1000)	Precision	OMP/MKL Threads				
		1	2	4	8	16
matmul_C	DOUBLE					
	SINGLE					
matmul_MKL	DOUBLE					
	SINGLE					

d) Determine the run parameters (memory size and runtime limits) necessary to run matmul_MKL with N=30000 (Double Precision). [Hint: you can do it with srun interactively).

Take the runSERIES.sh bash script from LabX_9 and modify it to submit runMKL.sh (part of this project) to launch matmul_MKL –s 30000 (modify runMKL.sh also). Thus, submit with this the runSERIES script all the jobs to test with Threads=1, 2, 4, 8, 16 into the cpu queue. It may go faster if you submit the jobs with Threads=1,2 into the htc queue.

You are successful when you submit the appropriate runSERIES.sh script and the table:

Таб. 2 Scalabiity of matmul_MKL with OpenMP threads with the MKL library.

program (N = 30000)	Precision	OMP/MKL Threads				
		1	2	4	8	16
matmul_MKL	DOUBLE					
	SINGLE					

The GPU project

Copy the file from sandbox ~/../LINUX_SUPERCOMPUTERS/programs/lab10_GPU.tar.gz to your directory on Zhores. The two hello programs we have discussed during the class and you can repeat compiling them and running after allocating resources:

```
tar -xf lab10_GPU.tar.gz
source setenv.sh     #from CPU part
make

salloc -p gpu_devel -N 1 -n 1 -c 1 --gpus=1 --mem=6G
```

After the resources have been allocated, `srun hello` and srun hello_v to make sure it works.

There is a new program that computes matrix multiplication matmul_GPU. It is extracted from the GPU sample/0_Simple directory, therefore not too clean and somewhat "fragile". Not all matrix sizes are working. These tested are shown in run.sh script.

a) Adjust the run.sh script to a reasonable size matrix (i.e. size 9600 takes about 1 GB and 2 minutes run on VT100, 5 minutes on GTX-1080 Ti) and make sure you get the output correctly. The program does its own check ("Result=PASS") .

Prepare a test script with matmul_MKL for the same size and increasing #threads, up to the maximum of 32 threads to compare the performance result with the GPU. It should give the table:

Таб. 3 MKL performance for increasing number of threads to compare with the GPU result.

program (N = 9600)	Precision	OMP/MKL Threads					
		1	2	4	8	16	32
matmul_MKL	SINGLE						
GPU V100	SINGLE						
GPU GTX-1080 Ti	SINGLE						

The GPU GTX-1080 Ti can be found with salloc –p gpu_devel, while the VT100 with salloc –p gpu.