Otto-von-Guericke-Universität Magdeburg
Max-Planck-Institut for Dynamics of Complex Technical Systeme
Computational Methods in Systems and Control Theory

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Website: http://www.mpi-magdeburg.mpg.de/2896761/sc2

summer term 2015

Scientific Computing 2
Exercise 4
06/01/2017

Handout: 05/23/2017

Exercise 1: **(10 Points)**

We consider the sparse matrix-vector product $x = Ay$. The matrix A is stored using the compressed row storage scheme as described in the winter term's part of the lecture.

a.) Implement the sparse matrix-vector product as a C function

```
void sparse_mvp(struct sparse_matrix_st *A, double *y, double *y);
```

Parallelize the function using OpenMP in a proper way. Evaluate the different OpenMP scheduling schemes `dynamic`, `static`, `guided` and `auto`. If the scheduling scheme allows a chunk-size vary it from 1 to 4096. Show how the different scheduling schemes influence the runtime of the matrix-vector product.

b.) We assume the following hardware-setup to compute the sparse matrix-vector product:

- 2 8-core CPUs @ 2,9 GHz. peak performance per CPU 180 GFlops/s,
- memory bandwidth per CPU 21 GiB/s,
- 20 MB level-3 cache per CPU.

Determine the optimal number of threads to use and the corresponding flop-rate under the assumption that $A \in \mathbb{R}^{171657 \times 171657}$ with $5\ 144\ 734$ non-zero entries. Furthermore, we assume that the vectors x and y will stay in the CPU cache.

A skeleton code is available at: http://www2.mpi-magdeburg.mpg.de/mpcsc/lehre/2016_WS_SC/tutorial/skeleton_sparse.tar.gz. The test matrix mentioned in b.) can be downloaded from http://www2.mpi-magdeburg.mpg.de/mpcsc/lehre/2015_SS_SCII/tutorial/sparse_matrix.mtx.gz and needs to be decompressed using `gunzip`.

Exercise 2: **(12 Points)**

The Mandelbrot-set $\mathbb{M}$ is the set of all complex numbers c for which the sequence $z_0, z_1, z_2, \ldots$ defined by

$$z_{n+1} = z_n^2 + c$$

with the initial condition

$$z_0 = 0$$

does not approach infinity. The popular visualization of this set normally works as follows: All points c which belong to $\mathbb{M}$ will be colored in black and for all other points a color depending on their divergence rate is chosen.
We provide a purely sequential code http://www.mpi-magdeburg.mpg.de/mpcsc/lehre/2015_SS_SCII/tutorial/mandelbrot.tar.gz which computes the famous part $(-2, 1) \times (-1, 1) \in \mathbb{C}$ of the Mandelbrot-set as a BMP-file.

a.) Parallelize the code using OpenMP such that the rows of the picture are distributed over the processors.

b.) Parallelize the code using OpenMP such that the all points are distributed over the processors.

c.) Compare the runtime of both variants for different number of threads and scheduling schemes. Select `guided`, `dynamic`, and `static` scheduling with a chunk size of 32, 64, and 128.

In the exercise: Develop a GPU accelerated variant of the code which computes each point in parallel. Use different thread block sizes like 1×1, 4×4, and 16×16, and compare the runtime. The host is not involved in the computation of the set.

Exercise 3: **(8 Points)**

Implement the "Block Outer Prodoct" LU decomposition without pivoting. Thereby, the input matrix A should be overwritten by its factors L and U. The inner rank-k-update must be parallelized using OpenMP. The function has to fulfill the following calling sequence:

```
void LU(struct my_matrix_st *A, int k)
```

where k determines the blocking factor, i.e., the size of the rank-k update.
Determine the optmimal blocksize k for matrices of size 1000, 2000, 3000, 4000, and 5000. Futhermore, determine the speed-up and the parallel efficiency for 1, 2, and 4 threads.

Overall Points: 30