Technische Universität München
Lehrstuhl Informatik V
M. Bader, A. Pöppl

WS 2016/17
Worksheet 4
December 14th, 2016

Tutorial: HPC - Algorithms and Applications WS 16/17

Complete the following assignments (alone or in a group), and send your source code via e-mail to `poeppl@in.tum.de` until Sunday, January 8th 2017.

Worksheet 4: Heat Equation in CUDA

Remark: In the template for the exercise, you are provided with a **CMake** build file as an alternative to the classical Makefile. CMake is a cross-platform configuration tool, and should be able to generate **Makefile** as well as **Visual Studio** projects.

T4.1: cuBLAS + ELLPACK

a) Write an ELLPACK matrix-vector multiplication kernel for the Heat Equation example code.

   i) Open `kernels.cu` and implement the `ell_mat_vec_mm` kernel:

   - Define grid and block size for the call to `ell_mat_vec_mm`
   - Assign a matrix row to each thread and let the kernel perform the inner loop.
   - Take care that access to the arrays `indices` and `data` is coalesced, access to `x` may be uncoalesced.

   ii) Open `poisson.c` and implement `poisson_ellpack`.

   - Instructions for cuBLAS context creation and destruction are already added.
   - Use cuBLAS in order to execute vector algebra required for simulating the discretized heat equation.

b) Execute the code using `./sparse -H [<matrix size>]`. If you have gnuplot installed you can uncomment `#define GNUPLOT` in `poisson.c` for visual output. Textual output for a matrix of size 64×64 should be:

```
t = 0.0000, err = 1.4142e+00, Temperature at x = 0.5: 1.0000e+00
t = 0.6450, err = 9.8181e-03, Temperature at x = 0.5: 6.9696e-01
```

```
t = 1.2900, err = 5.1307e-03, Temperature at x = 0.5: 3.8419e-01
t = 1.9350, err = 2.8166e-03, Temperature at x = 0.5: 2.1122e-01
t = 2.5800, err = 1.5484e-03, Temperature at x = 0.5: 1.1613e-01
(...)
```

H4.1: cuSPARSE

Write a cuSPARSE matrix-vector multiplication kernel for the Heat Equation example code.

a) Open `poisson.c` and implement `poisson_cusparse`. A sparse matrix in CSR format already exists.

- Add instructions for cuSPARSE context creation and destruction.
- Convert the CSR matrix to HYB format using cuSPARSE
- Call cuSPARSE for matrix-vector multiplication in the time step loop
- Use cuBLAS instructions for vector algebra, as in T4.1

b) Execute the code using `./sparse -H -C [<matrix size>]` and check if the result is consistent with assignment T4.1b. How do performance of the ELLPACK and the cuSPARSE kernel compare for different matrix sizes?