IWR, University of Heidelberg — Winter term 2014/15
Exercise Sheet 12 — 22. Januar 2015

Exercise for Course

Parallel High-Performance Computing

Dr. S. Lang

Return: 29. Januar 2015 at the beginning of the exercise or earlier

---

Task 25 N-Body Problem with PThreads and CUDA (10 points)

You receive an update of the zip file `nbody2.zip`. Here the kernels `nbody_pthread.c` and `nbody_cuda.cu` are implemented, furthermore the archive contains the solution of the MPI kernel `nbody_mpi.c`. In the pool you can compile as usual with `make`, to build also the CUDA parallel programm. In the pool NVidia GPUs are available, in the detail GeForce 630 GT (`lspci | grep VGA` for more information). Please get used to the new codes.

Tasks

1. Perform simulations with the same parameters, that you have choosen for the last exercise sheet. More than 4 threads don't make sense in the pool. Measure the MFLOPs and compare the performance of the sequential and the parallel variants. Useful are plots of the MFLOPs rate over the problem size resp. the speedup. Which variant has worked best in your opinion?
2. Comment shortly on the *execution configuration*, this means explain the three parameters in the brackets of the kernel call `acceleration_kernel<<<dimGrid,dimBlock,BLOCKSIZE*sizeof(float4)>>>(j,xd,ad);` in line 98.
3. Why has the parameter $\epsilon 2$ been introduced in Plummer potential? Why can't the original value of $1e-14$ been used (see the vanilla variant), if you compute on the GPU?
4. Comment shortly on the accuracy of the results of a simulation run with the different sequential and parallel variants, but same paramater $N, \ldots$. Use for this the script `fuzzy_diff` and analyse the output file for several fixed time points. Do the differences increase with time, this means accumulate the differences? What is the largest „measured" deviation that you have achieved?

Free Willi Extension

In the Makefile the options `--use_fast_math` are set for the CUDA kernel and `-O3 -ffast-math -funroll-loops -fexpensive-optimizations` for the other kernels. Examine the modification with these flags and think what influence they might have on the result. Repeat some simulations without these flags (thus optimized only with `-O3`), and discuss the performance in this case.

The graphics card in the pool has theoretically a performance limit of about 360 GFLOPs. In the test we only get about 150 GFLOPs. Try to improve this rate. One thing becomes obvious: The CUDA calculation in the pool performs for real more FLOPs than assumed in the measurements. Here you could also measure the real MFLOPs rate. However you could still think about further optimisations.

Hints

- The path for the CUDA compiler is explicitly set in the Makefile, it resided in `/usr/lib/nvidia-cuda-toolkit/bin/nvcc`. If you want to compile on other machines, you have to adapt the `Makefile` or set the path locally: `export PATH=${PATH}:/usr/lib/nvidia-cuda-toolkit/bin/`.
- CUDA calculations can be very fast, the measured time interval is then small. In this case you can measure $0s$ as wallclock time and the MFlops rate is a `inf`. Watch out for a reasonably large N, to get meaningful values.