Rainer Spurzem

Parallel Computing with NBODY6++GPU
with and without GPU

Thursday, February 20, 2025, Uni Heidelberg, GPU Block Course

1. Compiling and Running NBODY6++GPU

1.1 Compiling and running the code on bwUniCluster

```
cp -p -i ~hd_un119/gworknb6.tar.gz .
tar xvfz gworknb6.tar.gz
cd gworknb6/build
make clean -f Makefile.gpu ; make -j6 -f Makefile.gpu
make clean -f Makefile.mpi ; make -j6 -f Makefile.mpi
ls -ltr nbody6++*
```

You find the executable files nbody6++.sse.gpu.mpi and nbody6++.sse.mpi ; move it to the Run directory:

```
mv -i nbody6++* ../Run/
```

Here are the batch job scripts mpi_script.sh and gpu_script.sh . There is one line in the header of the script files, which defines the number of tasks per node. You should change this number and start a run for each choice of:

```
ntasks-per-node=1,2,3,4
```

The standard job prepared for you uses 16000 particles, and simulation time of 5 model units (check output how many years). The input file containing some control data is N16k.inp

For our course you will need outputs of 8 jobs (mpi and gpu, 1,2,3,4 tasks per node), 16000 particles, 5 time units.

Additional voluntary study (not required):

Our runs contain astrophysical stellar evolution, with stars ranging from initially 0.08 to 100 solar masses. Data for a Hertzsprung Russell diagram (HRD,

temperature/luminosity) are in sev.83_n (here n stands for the time, like 0,1,2,3,4,5 time units). In this file you get lines with following columns:

time, index, name, stellar type, pos. in cluster, mass, log luminosity , log radius, log effective temperature (units are dimensionless code units for time and position, solar mass, solar luminosity, solar radius, temperature in Kelvin). There are two more columns in the file, which are the age and formation time of the star in Myr – we do not use them here.

Astronomers like to plot log(luminosity) as a function of effective temperature (high T left, low T right on the x-axis), that is called an HRD (Hertzsprung-Russell diagram, sometimes observers use colour-magnitude diagram, which is equivalent).

The code also produces a lot of other output files. Most of them are not interesting for us. We will look mainly at the output listing such as e.g. N16k.xxx.nnnnn.out ... Here xxx is either gpu or mpi and nnnnn is a job id number.

```
grep ADJUST N16k...xxx.nnnnn
```

You can see whether the run has done well. To extract the timing data relevant for the course exercise, find the lines below ADJUST, headed by “rank PE N Total...” (times in secs), and for a final number you can also look at the end for the line starting with “Total CPU (in minutes)”.

2 Parallel Communication Schemes and Literature

NBODY6++GPU runs in the SPMD (Single Program Multiple Data) Scheme. It means when you start the parallel NBODY6++ run with n parallel tasks (or mpi processes - by using the command mpirun -np n …), n identical copies of the program will start. In parallel sections these copies of the code share their work and communicate data with each other through a software package called MPI (message passing interface). For GPU runs also every task is using a GPU.

NBODY6++GPU uses for communication a copy algorithm (all new information is copied immediately to all nodes); other algorithms are ring algorithm or (hyper)systolic algorithm, see Dorband, Hemsendorf, Merritt, 2003 (Journ. Comp. Phys.); Makino (2002). If the number of particles per node is large enough, all algorithms scale equally well. The copy algorithm in NBODY6++ is implemented manually with `MPI_SENDRECV`. Current modern implementations of `MPI_ALLREDUCE` will be equally efficient.

3 Hands-On Experiment on parallel computer

3.1 Profiling for NBODY6++GPU

The code measures the wall clock time used for many things:

total, regular force, irregular force, adjust, initialization

Your task: Do some experiment - run 1,2,3,4 tasks (MPI processes), with and without GPU usage, as explained above. Find, cut and paste the lines below the timing header ("PE N ttot.."). Use the last one in your job (after ADJUST TIME= 5.00).

Explanation of times (all in secs) in output (line starting with 'rank PE N', numbers below):

```
Total: total     wallclock time
Reg.: regint, regular force (PAR)
Irr.: nbint, neighbour force (PAR)
Adjust: energy check (PAR)
Init.: computing of initial model (PAR)
```

(PAR) means these routines are parallelised (contain shared work and MPI functions); there are more times listed, but we do not need them here. Times needed for Reg, Irr, Init and Adjust are required to determine X (see below); Time for Total, Reg, Irr, and a prediction from Amdahl's law should be plotted as a function of number of tasks used (1,2,3,4), both for MPI and GPU jobs. What is the maximum speedup we get, without GPU, with GPU? What result comes from Amdahl's law? Note that the speed-up should be measured relative to the single node case;

3.2 Example Solution for NBODY6++GPU Tasks:

Calculate Amdahl's Law:

Let X be the part of my program (in terms of computing time) which can be parallelised. The sequential computing time T_{seq} is normalized to unity (1), and can be expressed as:

$T_{seq} = 1 = X + (1-X)$

The parallel computing time T_{par} under ideal conditions (ideal load balancing, ultrafast communication):

$T_{par} = X/p + (1-X)$ with number of tasks p (processes, in case of GPU runs also the number of GPUs, every process has one);

Then the speed-up of the program $S = T_{seq} / T_{par}$:

$S = 1 / (1-X+X/p)$

Note the limit if p is very large: $S = 1/(1-X)$. We find from our measurements with NBODY6++GPU given above: X = (Reg. + Irr. + Init. + Adjust) / Total. So, if we know X and p we can compute a predicted speedup S, and with that predict a parallel computing time by using

$T_{par,p} = T_{seq}/S$

The times $T_{par,p}$ are used to plot a line for Amdahl's law . Note that the computation of X needs to be done separately for GPU and MPI runs. We discuss this in the lecture.

If you have successfully finished the 8 runs and collected your data, your plots should ideally give curces for Total close to Amdahl's law. However, there may be deviations e.g. due to system load etc. No matter whether your results look good or "bad", it is ok to pass the course.

Summary of your tasks to pass the course; please turn in the following results to your tutors:

1.) Two plots showing the times Total, Reg, Irr – one for MPI, one for GPU jobs. Plot also the predicted times obtained from Amdahl's law.

2.) Data file(s) containing the data you have used for 1.); or a notice where we can find the file on bwUniCluster. **Please do not delete the output files of your 8 runs, because they are proof that you did the experiment.**
3.) A few (one, two, three...) sentences for interpretation: How good is Amdahl's law working? How good works GPU acceleration? Did you get outliers / bad results which do not match expectations? Anything else you like to mention. And please also questions if there are.