How to start an Nbody job on kepler2 for students was created by Rainer Spurzem

(1) Getting started:

* after login to kepler2, do another ssh in the terminal window to wn12: ssh wn12

It should work without password. We need this because of some problem on the main node kepler2; for compiling the code kepler2 has a problem, we need to go to one of the worker nodes. Anyone will be ok, let's use wn12.

* You should see a prompt like [yourname@wn12 ~]$; make sure you are in your home directory (use pwd command). Note that the files and directories are the same on kepler2 and wn12, same data. Create an Nbody subdirectory; and go there:

mkdir Nbody ; cd Nbody ; pwd

* copy the tar package from my home directory to this place:

cp -p -i /home/Tit1/spurzem/Nbody/Nbody6++GPU-Dec2020.tgz . (look in man cp page if you do not know what means -p -i, it is very much recommended)

ls -lart

tar xvfz Nbody6++GPU-Dec2020.tgz

* You should see a new subdirectory Nbody6++GPU-Dec2020 ; just go there with

cd Nbody6++GPU-Dec2020

now you should see the source code in ./src/Main/ (use

ls -lart ./src/Main/

to see); and some files in the current directory, like Makefile and configure.

* Before proceeding you need to double check:

Your file /home/Tit4/lecturenn/.bashrc (nn is your account number) should contain at the end two lines:

module load cuda/7.5 (this loads the nvcc cuda c compiler for GPU computing)
ulimit -s unlimited (this allows for large memory allocation)

This has been done for you already by the system.

* make sure you are in /home/Tit4/lecturenn/Nbody/Nbody6++GPU-Dec2020 (check with pwd)

Now enter the command:

./configure --with-par=b1m --enable-simd=sse --enable-mcmodel=large

(This enables simulations with up to one million particles and many binaries, like up to 10%; you can use actually smaller particle numbers, which are selected in the

input file, see below).

After it is finished:

make clean ; make -j

NOTE ADDED for HDF5 (PLEASE SKIP this step, we will not do it now)
--------------cut--------------------------
You need to edit manually the file build/Makefile - because the configure script does not work well for hdf5 on kepler:

HDF5_FLAGS = -D H5OUTPUT -I/usr/include/openmpi-x86_64/ -L/usr/lib64/openmpi/lib/ -lhdf5_fortran
FFLAGS = ... ${HDF5_FLAGS}
-------------end cut-----------------------
* If all of this is successful, you should find in ./build/nbody6++.sse.gpu.mpi an executable file - it is the file to run the code.

See with

ls -lart ./build/

(2) Prepare a run:

.... and here are the next few steps, everything can be done on kepler2 now, no need to do ssh to wn12.

cd Nbody/Nbody6++GPU2020

Copy the executable file with new name to the run directory 100k_test:

cp -p -i build/nbody6++.sse.gpu.mpi 100k_test/nbody6++.sse.gpu.mpi.b1m

Now, if you go to 100k_test you should have three files: an N-body input file (N100k.inp), a batch job script nb6++gpu.sbatch.KEPLER), and the executable file, which you produced in the last. I recommend to give it the name nbody6+ +.sse.gpu.mpi.b1m , it means it allows for a large number of binaries.

With these data in 100k_test you are able to start a test run, with 100k single particles (no binaries yet); it will run for 100 time units and produce a lot of output files. How to start it? You should be inside 100k_test; then you say

sbatch < nb6++gpu.sbatch.KEPLER

With squeue you should see a message that your job is running. With ls -lart you will see many output files, one of them is an output file like N100k.somenumber.out . This gives informations about the progress of the run. It may take a few hours.

In Nbody6++GPU-Dec2020/doc/ you find a manual, it gives some informations about the .inp and the .out files. Do not worry - there is a LOT of informations and a lot of data; for beginning only few of them are important.

(3) The Input File:

Nbody6++GPU-Dec2020/100k_test/N100k.inp:

```
1 1.0E8 1000.0 40 40 0
100000 1 10 43532  80 1 10
0.01 0.01 0.15 1.0 1.0 100.0 1.0 1.0 0.7
0 1 1 0 1 0 4 0 0 2
0 1 0 2 2 0 0 0 3 6
1 0 2 0 0 2 0 0 0 2
1 0 2 1 1 0 1 1 2 0
0 0 0 0 0 2 3 0 1 0
2.5E-6 8.E-5 0.1 1.0 1.0E-06 0.01 0.125
2.35 150.0 0.08 0 0 0.001 0 1.0
0.5 0.0 0.0 0.0
1.78E11 13.3

KSTART, TCOMP, TCRITp, isernb,iserreg,iserks                  (nbody6.F)
N, NFIX, NCRIT, NRAND, NNBOPT, NRUN, NCOMM                    (input.F)
ETAI, ETAR, RS0, DTADJ, DELTAT, TCRIT, QE, RBAR, ZMBAR        (input.F)
(KZ(J),J=1,40)                                                (input.F)
(BK(J),J=1,10)                                                (input.F)
DTMIN, RMIN, ETAU, ECLOSE, GMIN, GMAX, SMAX                   (input.F)
ALPHA, BODY1, BODYN, NBIN0, ZMET, EPOCH0, DTPLOT              (data.F)
Q, VXROT, VZROT, RSPH2                                        (scale.F)
NBIN, SEMI0, ECC0, RATIO,RANGE, NSKIP, IDORM                  (binpop.f)
GMG, RG0                                                      (xtrnl0.F) KZ(14)=2
GMG, DISK, A, B, VCIRC, RCIRC, GMB, AR, GAM,RG(1-3),VG(1-3)   (xtrnl0.F) KZ(14)=3
NBIN, SEMI0, ECC0, RATIO,RANGE, NSKIP, IDORM                  (binpop.f)
KZ(8)=1,KZ(8)=3
```

(4) Looking At Output

1) Regularizations - whenever two particles come close to each other, such that their timestep becomes too small for the big N-body system, they are regularized. More informations in my lecture later in September. Practically it means that the two bodies disappear from the big N-body system, and only their center-of-mass (c.m.) comes in as a pseudo-particle. The internal motion is followed in a transformed four-dimensional space using quaternions. Perturbations from outside particles are taken into account. The code uses lines like

NEW KSREG.... and END KSREG to signal that a KS pair has been built (KS = Kustaanheimo-Stiefel regularization).

```
 NEW KSREG   TIME[NB] 6.2175781250E+01 NM1,2,S=         8     77471    100008
KW1,2,S=  14   0   0 IPAIR        2 DTAU  2.17E-03 M1,2[NB]  6.13E-04  2.14E-06
R12[NB]  3.21E-07 e,a,eb[NB]=  1.1037E-01  2.8924E-07 -2.27E-03 P[d]=  8.89E-01 H -
1.06E+03 GAMMA  1.65E-22 STEP(ICM)  1.95E-03 NPERT    0 NB(ICM)   81 M1,2[*]
3.59E+01  1.25E-01 RAD1,2,S[*]  1.52E-04  1.44E-01  1.42E+01 RI,VI[NB]=  1.81E-01
7.01E-01
```

TIME: Nbody units; NM1,2,S: Name of 1,2,c.m.; KW1,2,S: stellar type of 1,2,c.m.; DTAU: Step in KS time variable s or tau; M12; R12: masses and separation in N-body units; e,a,eb: eccentricity, semi-major axis, binding energy in N-Body units (note if hyperbolic encounter we have: e>1, a<0, eb>0); P[d]: Period in days; H: energy per mass in N-body units; GAMMA, STEP(ICM), NPERT, NB(ICM): perturbation, step of

c.m. in Hermite, M12[*],RAD12*: Masses and Radii in solar units, separation in solar units, RI,VI: position and velocity of system in cluster (N-body units).

Most of these events are hyperbolic fly-by's; sometimes a longer lasting binary is formed. The output lines contain detailed information what kind of particles are in a two-body encounter, and give most useful parameters, I will show and discuss it with you.

If you should some day in the future have a deeper look in the code an important issue of the data structure is related to KS regularization:

Let's say in the beginning we have our particle data in a vector 1...N for N particles (say X(1,I), I = 1,...N; which is the x-coordinate of particle I). If a KS-Pair (binary) is formed, the two members are put to the first two places in the vector; and the coordinates of the c.m. pseudoparticle come at N+1. So, if NPAIRS is the number of KS pairs we have in the vector:

position 1 to 2*NPAIRS: all members of KS binaries;
2*NPAIRS+1 to N: all single particles
N+1 to N+NPAIRS: all c.m. pseudo-particles
We have auxiliary variable IFIRST = 2*NPAIRS+1 and NTOT = N + NPAIRS .

So, you find in the code two kinds of loops:

from 1,N : all particles, no matter whether they are inside a KS binary or not
from IFIRST,NTOT: all single particles plus c.m. particles

2)a) Lagrangian Radii

Lagrangian radii are spherical shells, containing a fixed fraction of total mass. The half-mass radius(50% Lagr. radius) is frequently used, but we have currently 18 different valus, defined in lagr.f of mass fraction, ranging from 0.1 % to 100%. The code prints lines like (take these lines out of the big output file, use e.g. unix grep, into a new file for plotting).

Heading Line (fraction of total mass M/MT, last column core radius)
```
   TIME   M/MT:   1.00E-03 3.00E-03 5.00E-03 1.00E-02 3.00E-02 5.00E-02 1.00E-01
2.00E-01 3.00E-01 4.00E-01 5.00E-01 6.00E-01 7.00E-01 8.00E-01 9.00E-01 9.50E-01
9.90E-01 1.00E+00      <RC
```

Lagrangian Radii Lines:
```
0.0000D+00 RLAGR: 6.75E-02 9.38E-02 1.02E-01 1.30E-01 1.90E-01 2.34E-01 3.07E-01
4.29E-01 5.40E-01 6.58E-01 7.82E-01 9.25E-01 1.15E+00 1.46E+00 2.12E+00 2.88E+00
4.72E+00 6.10E+00 3.17E-01
```

at every N-body time (the interval depends on DELTAT input parameter). These lines can be collected in a file (e.g. using linux grep) and used to plot Lagr. Radii as function of time (the first number in the line). Also other quantities of the Lagr. shells are printed, like SIGR, SIGT, VROT and a few more. I want to show it to you in the output file and discuss.

If your N-Body output file is N100k.19491.out you get these lines in a file:

```
grep RLAGR  N100k.19491.out > RLAGR.data
```

RLAGR.data contains all the lines, ready for plotting Lagrangian radii as a function of time (column 1).

2b) Average Mass:

Printed is the average stellar mass in N-body units (use ZMBAR to scale to solar masses) inside the Lagrangian mass shells, as defined in 2a). You find the line in the code:

0.0000D+00 AVMASS: …

If your N-Body output file is N100k.19491.out you get these lines in a file:

grep AVMASS N100k.19491.out > AVMASS.data

AVMASS.data contains all the lines, ready for plotting average masses in Lagrangian mass shells as a function of time (column 1).

3) Stellar Evolution File

The code produces files with names like sev.83_n , where again n is the time (n=0 initial model, n=1 at one N-body time unit and so on). These files contain stellar evolution data for all stars, for example columns 6,7,8,9 are mass in solar masses, and luminosity (in log10 solar units), stellar radius (in log10 solar units), Teff (in log10 Kelvin). These data can be used to plot directly Hertzsprung Russel Diagrams, using x-axis column 9, y-axis column 7. Note: Astronomical habit is to plot the x-axis such that small Teff is left, and large Teff is right.

Note also that column 5 contains the distance of the star from the cluster center, and column 4 the type of star, from which you can read whether we have main sequence star, or a red giant, or black hole or whatever (see our manual).

(5) Home Work

(if possible by Oct. 17, if longer time needed let me know)

* Have in your account the N-body run output file, produced by your run, 100k particles, ready for up to 100 time units.
* Have a plot of Lagrangian radii as a function of time (it is ok to plot 5 selected Lagrangian radii, not all)
* Have a plot of average masses as a function of time (it is ok to plot 5 selected Lagrangian radii, not all).
* Have two “colour-magnitude” diagrams, one for t=0, another for t=100 (can be in one plot). “Colour-magnitude” is astrophysical observers terminology; for us it means: plot log(luminosity) as a function of log(Teff), with high temperaturs to the left, low temperatures to the right. Plot a dot for every star.