Parallel computing with LMGC90

November 6, 2020

Introduction

Part of the developments of LMGC90 are currently dealing with the optimization of tasks using parallel computing. However, given the complexity of the code and the structure of the Contact Dynamics Method (i.e., the algorithm behinds LMGC90), not all the can be coded for multithreading processing.

Between the most time-consuming operations the code runs, we find on top: 1) the solver of interactions (the resolution of contact forces and particles' velocities), and 2) the contact detection. The first is parallelized but not the second.

Other parts of the code, such as the reading/writing of files, the preparation of the simulation (this is the computation of rigid body properties, external forces) are not parallelized either.

Now, how many processors is it worth using with LMGC90? Bellow, you find a figure from the Ph.D. dissertation of Mathieu Renouf (Optimisation numérique et calcul parallèle pour l'étude de milieux divisés bi- et tridimensionnels, 2004) presenting the speed-up (SP) of LMGC90 as a function of the number of processors (P). The different lines correspond to different data structures the code can use. For the moment, the most stable and that we usually use is the SDL ("Stored Delassus Loops"). As you can see, the speed-up quickly reaches a plateau around 4. That means that, at most, we can divide the computing time by 4 times when using around 16 processors. It is probably not worth using 16 processors for just a speed-up of 4 when using 4 processors already splits the computing time by almost 2. So, I recommend using not more than 6 processors for your simulations when you are running parallel jobs with LMGC90.

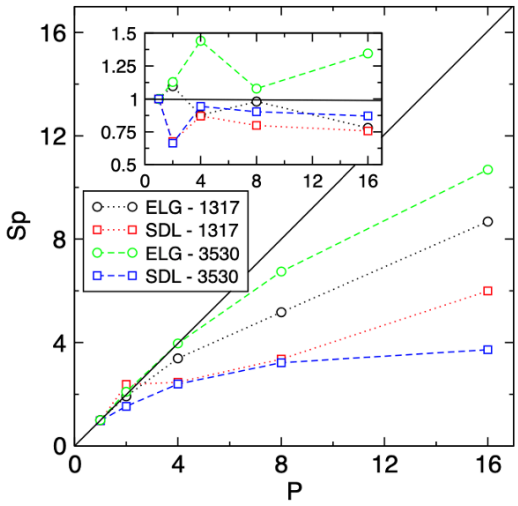

Figure 1: Speed up scaling with number of cores

Compiling a parallel version of LMGC90

To run simulations in parallel, you need to make a new compilation of LMGC90. Below, You will find a comparative step-by-step procedure on how to compile and run a single processor simulation and a multithreading simulation.

Just for your information, there are several methods to run algorithms in parallel. LMGC90 uses an implementation called OpenMP.

Single processor

1. Open a terminal and go to your the directory containing the sources lmgc90_user_xxx.

2. Type on your terminal:

```
mkdir build
cd build
cmake ..
make
```

Multi processor

1. Open a terminal and go to your the directory containing the sources lmgc90_user_xxx.

2. Type on your terminal:

```
mkdir build_parallel
cd build_parallel
cmake ..  -DWITH_OPENMP=True
make
```

Now you have 2 installations of LMGC90 in your system. It is very important to differentiate each one when you want to run a simulation.

Before, you have set your system to automatically recognize your installation with these lines (see compilation instructions in the lmgc90 wiki page):

```
if [ -z ${PYTHONPATH} ]; then
  export PYTHONPATH=mysourcepath/build
else
  export PYTHONPATH=${PYTHONPATH}:mysourcepath/build
fi
export PATH=${PATH}:mysourcepath/src/addons
```

To run a single-processor simulation, you have to write on the terminal (before running your scripts) the line corresponding to:

```
export PYTHONPATH=${PYTHONPATH}:yoursourcepath/build
```

To run a multithreading simulation, you first need to write on the terminal these two lines:

```
  export OMP_SCHEDULE=STATIC
  export OMP_NUM_THREADS=4
```

The first line sets that you will be using a given number of processors, and you do not allow the algorithm to use more than that.

The second line sets the number of processors that you want to use. In this example, we set the number of processors to 4.

Then, you have you set your PYTHONPATH to the multithreading compilation. Something like:

```
export PYTHONPATH=${PYTHONPATH}:yoursourcepath/build_parallel
```

And then you can run your scripts normally. You can also check on your computer tasks manager the number of processors your script is using. It can vary between 1 and the number of processors max (OMP_NUM_THREADS) that you set, depending on the routines the code is running at that moment.