Parallel Barnes-Hut algorithm to deal with N-Body problem

Master Erasmus EPFL Student

ATTACHE William

attachew974@gmail.com
Linkedin

Departement of Mathematics - MATH454
Ecole Polytechnique Federale de Lausanne

Spring Semester 2018

Outline

Introduction

- N-Boy problem : *predict the individual motions of a group of objects interacting with each other under gravitationnal forces*
- Parallelize : Sequential $O(n^2)$ *v.s.* Barnes-Hut $O(n * ln(n))$
- Work load balancing
- Communication between processors : `MPI`
- Local Essential Tree (`LET`)

Parallelization choices

1. gprof and code profiling (20000 particules, 1000 time steps)

Method	**% of total time**	**Number of calls**
insertNode	78, 62	5483872
computeForce	9, 00	5000000
gForce	6, 77	80041997
dist	4, 91	108258504
getSubtree	3, 38	80084009
buildTree	0, 75	1000

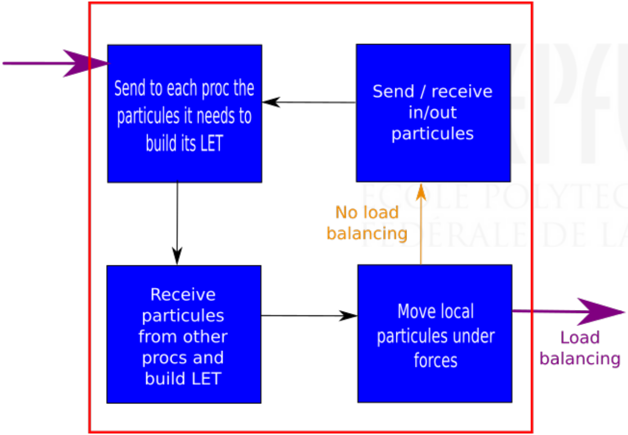
2. MPI \ OpenMP

Communications

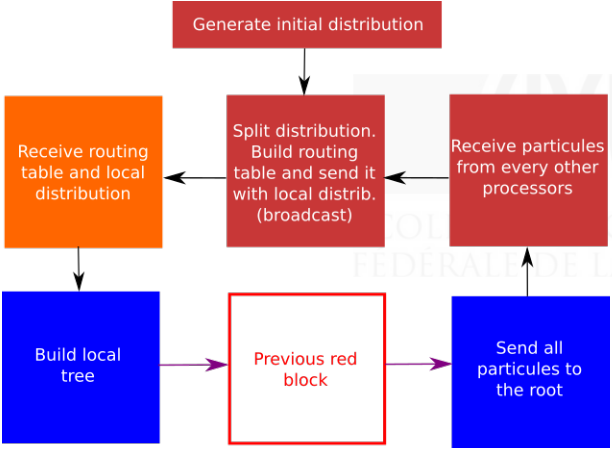
1. Routing table (*MPI_Broadcast*) & local distribution
2. Load balancing (*MPI_Send**MPI_Recv*)

Global architecture

For a processor with its local distribution.

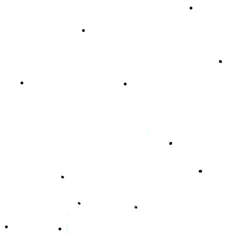


Global architecture



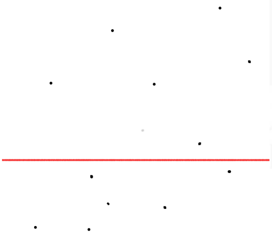
Split the work load among processors

#areas = 1 ; #leftprocs = p-1



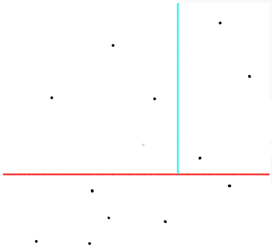
Split the work load among processors

`#areas = 2 ; #leftprocs = p-2`



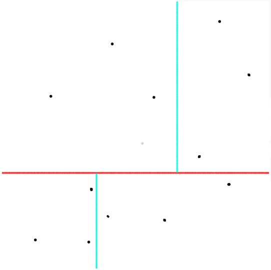
Split the work load among processors

`#areas = 3 ; #leftprocs = p-3`



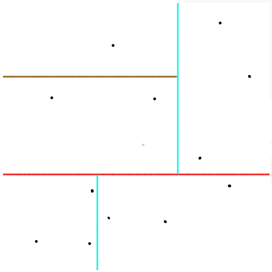
Split the work load among processors

`#areas = 4 ; #leftprocs = p-4`



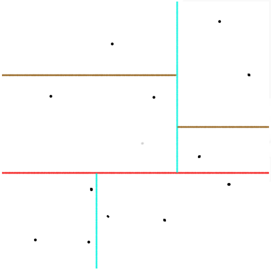
Split the work load among processors

`#areas = 5 ; #leftprocs = p-5`



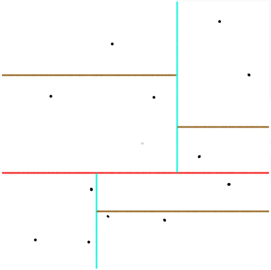
Split the work load among processors

#areas = 6 ; #leftprocs = p-6



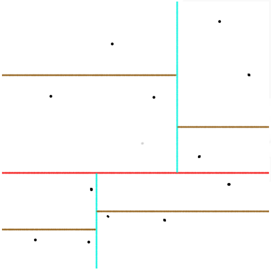
Split the work load among processors

`#areas = 7 ; #leftprocs = p-7`



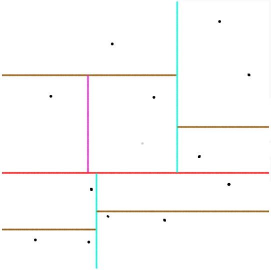
Split the work load among processors

`#areas = 8 ; #leftprocs = n-8`



Split the work load among processors

`#areas = 9 ; #leftprocs = n-9`



Main differences with Berkeley lecture notes

1. not only for $p = 2^m$ processors
2. do not need to perform a round-robin to send particules : routing table access in $O(1)$
   $\triangleright$ **main drawback** (reasonable) memory consumption to store such a table for every processor

Difficulties encountered

1. "learn" C++ and so deal with memory \ scope : `addChilds`
2. lots of *absent-mindedness* (move particules, reshape boxes)
3. particules received but coordinates changed
   → define a `MPI_Data_type` *particules* as a solution
4. *synchronization point* : impossible to run with more than 2500 particules
5. write at the right place in I \ O file

Improvements considered

1. Recompute the center of mass when a particule moves
2. Run simulations with heterogeneous distributions
3. Manage particule collisions and particules that go out
4. Add genericity to implement quick sort : define *comparison operators*.
5. Use more *auxiliary functions* in main, like fill arrays with addresses (build rooting table, back sending for load balancing) ; use more classes to structure the code.
6. Do not systematically create four childs when building the tree to save (a little) of memory

Manage particule collisions and particules that go out

▷ instead of going out, a particule could :

- bounce on the wall
- (erroneously) be considered as a point that goes away with the direction and the velocity it had when leaving the area

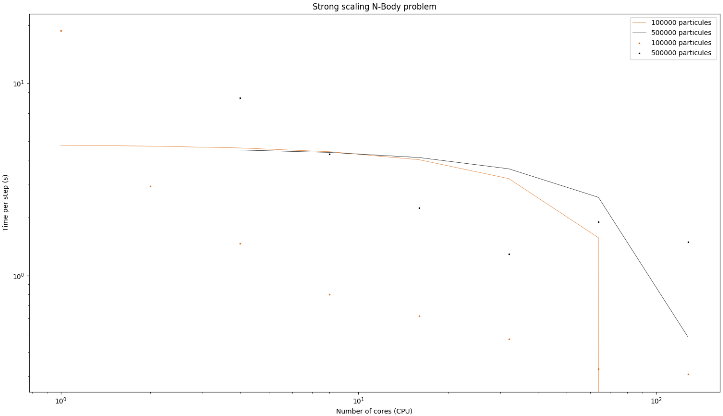
▷ collisions : for any particule, consider all particules which could have ramed it → add technical difficulties and computations

Strong scaling

▷ behaviour of the application by fixing the total size of the problem and increasing the number of processors

$$\texttt{SPEEDUP} \;:\; S_p(n) = \frac{t_p(n)}{t_1(n)}$$

Strong scaling

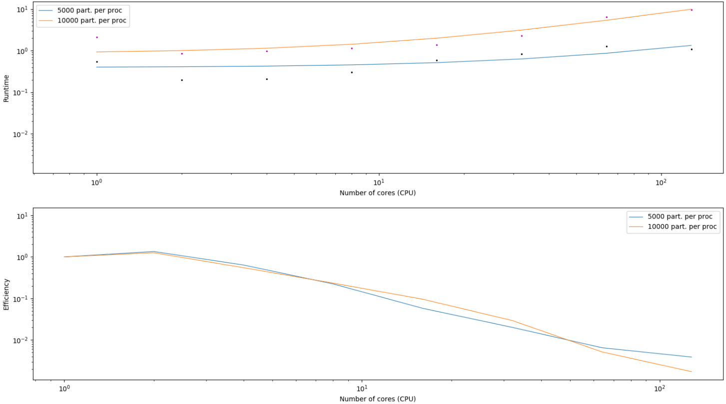


Weak scaling

▷ behaviour of the application by fixing the total size of the problem and increasing the number of processors

$$\texttt{PARALLEL EFFICIENCY} : E_p = \frac{S_p(n)}{p}$$

Weak scaling



Conclusion

- Encouraging scaling results
- But lots a work left to get a proper application
- Deal with particules collisions could obstruct scaling
- Maybe CUDA could offer better scaling results