Subject: alternative to array of linked list
Posted by forlano on Sun, 01 May 2016 22:52:21 GMT
View Forum Message <> Reply to Message

Hello,

I am doing experiments with molecular dynamics.
The domain of simulation is divided in cells and each cell contains a number of particles.
The particle can move from a cell to another.
Usually it is used an array of linked list. I would like to replicate this structure with U++ containers.
I do not know what to use:
Vector< Vector<int> >
Vector< Index<int> >
Map< ... >
something else?
Here the speed is very important: search quickly a particle in a given cell, delete it, move in another cell.
Does anybody have suggestions? If somebody have already done similar experiment snippet code are greatly appreciated.

Thanks,
Luigi

Subject: Re: alternative to array of linked list
Posted by Lance on Sun, 01 May 2016 23:55:36 GMT
View Forum Message <> Reply to Message

Linked list may not be a good option for this application.

In general, you will be happy with O(logN) insert,delete,search time complexity. std::set, set::map, std::unordered_set, set::unordered_map, or u++ VectorMap should all be good candidate. Maybe even Upp::InVector (I am not sure though). If class Particle is of considerable size, you can put them in either a Upp::Vector or a std::vector, and actually put its index in the vector in cell's particles container.

experiment with std::set, set::unordered_set, Upp::VectorMap etc, see which offer best performance for your application.

Subject: Re: alternative to array of linked list
Posted by mr_ped on Mon, 02 May 2016 00:32:42 GMT
View Forum Message <> Reply to Message

If the speed is very important, why do you bother about moving particles around?
I think the simulation itself is more computationally expensive, and the data structures should be optimized for that, not for particles movement solely (although that's part of it for sure).

For example, if the particles are of the same type, and have the completely same attributes, and the number of types is much lower than amount of particles, you may have for each cell just counters of each type, so moving particle between cells will be --, ++ operation.

Or if you process all the particles during simulation, and you don't must to process them per-cell in ordered way, you can store all particles in single array (vector), and just have "belongs_to_cell" index inside them, which is updated upon movement. And process them all the time in sequentially in the array.

Or if amount of particles per cell is quite average across all cells with no extremes, you can have some vector<int> in each cell with index into particle array, the vector with some reasonable buffer and pointer to last inserted particle, then do a move as old_cell.particle_index[x] = -1; // free slot, new_cell.particle_index[search_for_free_slot(last_insert_ind ex)] = particle_index;

Etc..

There's probably million possible ways, and to get some near-optimal advice you would have to show here the whole process and details.

Subject: Re: alternative to array of linked list
Posted by koldo on Mon, 02 May 2016 06:22:56 GMT
View Forum Message <> Reply to Message

Hello Forlano

What method are you using?

I have had some experience with SPH and LBM.

Subject: Re: alternative to array of linked list
Posted by forlano on Mon, 02 May 2016 06:30:44 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Mon, 02 May 2016 02:32
There's probably million possible ways, and to get some near-optimal advice you would have to show here the whole process and details.

Thanks both for reply!

Few more details. The particles are all the same and can be a huge number for example 1000 (and more if one has a super computer!). In principle the particle interacts with all others (particle i exerts a force on j and viceversa). As result of this interactions the particle can freely move around the domain.
However the interaction range of such force is just a fraction of the whole dimension of the

simulation domain (you may think a square DxD). To save a lot of computation time then come up the concept of cell. Each has dimension d=D/n, where d a bit gretear than the force range.
In this way you can scan the domain by cell, pick a particle inside it and look for other particles that stay in the same cell and in its neighbour cells (https://en.wikipedia.org/wiki/Cell_lists)
Each cell must have a container of the particle they belong to. During the upgrade process I must pick a particle from one cell (find and delete it) and move in another one (append).

I never used std::set. Perhaps it can be a good alternative to linked list.

Thanks,
Luigi

Subject: Re: alternative to array of linked list
Posted by forlano on Mon, 02 May 2016 06:36:56 GMT
View Forum Message <> Reply to Message

koldo wrote on Mon, 02 May 2016 08:22Hello Forlano

What method are you using?

I have had some experience with SPH and LBM.

Hi Koldo,

I do not know SPH and LBM. I have implemented an array of cell and each element contains a linked list of the particle id. Moreover I have an array of Particle which hold all physics stuff. But I am not satisfied by the result.
I would like to use something cleaner ans faster with find and delete.

Best regards,
Luigi

Subject: Re: alternative to array of linked list
Posted by koldo on Tue, 03 May 2016 13:36:57 GMT
View Forum Message <> Reply to Message

It sounds very similar to SPH (https://en.wikipedia.org/wiki/Smoothed-particle_hydrodynamic s), but I do not understand the role of the cells. Questions:
- Is the number of particles "almost" the same in all simulation?
- How is every solving step/iteration?. In the "explicit" case:
--- First, are computed the forces that act over every particle?
--- Second, is the movement of every particle computed depending on the forces?

If the problem is similar to this, you can:

- use a fixed list of particles.
- do a terrific parallelization

Subject: Re: alternative to array of linked list
Posted by forlano on Tue, 03 May 2016 18:30:03 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 03 May 2016 15:36It sounds very similar to SPH (https://en.wikipedia.org/wiki/Smoothed-particle_hydrodynamic s), but I do not understand the role of the cells. Questions:
- Is the number of particles "almost" the same in all simulation?
- How is every solving step/iteration?. In the "explicit" case:
--- First, are computed the forces that act over every particle?
--- Second, is the movement of every particle computed depending on the forces?

If the problem is similar to this, you can:
- use a fixed list of particles.
- do a terrific parallelization

The answer is YES.
The problem is that I have only one CPU.
Anyway I have implemented a
std::vector< std::unordered_set<int> >
and I am very happy. The code is very clean and easy to read.

Luigi

Subject: Re: alternative to array of linked list
Posted by koldo on Wed, 04 May 2016 07:53:10 GMT
View Forum Message <> Reply to Message

Are you sure that you have one CPU with one core/thread or with more than one core/thread?. If yes, the second phase could be parallelized.

If the number of particles is almost unaltered, a simple C/C++ array (malloc/new) or a U++ Buffer<> could be used.