Molecular Dynamics: Barnes-Hut Algorithm

Georgios Chnitidis,

September 17, 2020

1. The difference between long and short-range forces

The main attribute of short-range forces, like the strong and weak nuclear forces, is that they are decaying fast with the distance increasing. We can represent them with a dense force matrix with $O(N^2)$ complexity and mostly very small entries. We can cut-off the contributions of far away particles, thus making the force matrix sparse. That reduces the complexity of the entire force calculation to $O(N)$.
Concerning the long-range forces, like gravity and electromagnetic forces, we have a slow decay of the force contributions with increasing distances which means that we can't just cut off the force matrix, even the far away particles need to be taken into account. If we did that we would make just to hard high error and it would be inaccurate. We can reduce the complexity for the long-range forces calculations to $O(N\log N)$ with the Barnes-Hut algorithm.

2. Description of the algorithm

The Barnes-Hut method is based on the idea of exploiting hierarchies which can reduce the complexity significantly. The scheme recursively divides the set of bodies into groups by storing them in a quadtree (or an octree for 3D) in a binary style tree, with each node representing a region of the 2D space or it corresponds to clusters of particles (pseudo particle). As a logical corollary, the clustering of particles is required where the size of clusters depends on the distance to each individual particle, the multi-level tree-based domain decomposition needs to be solved for every particles position. The steps of the algorithm are the following:

Step 1: For 2D, we subdivide our domain into quadtree. We distribute the long-range region into specific sub-domain of Ω for each particle position $x \in \Omega$.

$$\Omega^{\text{far}} = \bigcup_i \Omega_i^{\text{far}}$$

We start in the root node, then we descent into the sub-domains and subdivide until every domain contains either 0 or 1 particles.

Step 2: Next step would be subdividing and computing the centres of mass or charge.For each level of hierarchy it's computing a center of mass and it is assign a point for each sub-domain corresponding to the center of mass. Then it is computing the total mass by summing particles for each sub-domain . The decomposition depends on the size of the sub-domain.

$$\text{diam} := \sup_{y \in \Omega_i^{\text{far}}} |y - y_0^i|$$

Step 3: Finally we compute the forces. For each particle in position x, we go back to the root node and then we descent into the tree sub-domains, until the θ-rule is satisfied,

$$\frac{\text{diam}}{r} \leq \theta,$$

r is the distance of pseudo particle from x. If θ is set to zero which means that we are always going to descend towards the leaves of the tree and compute every particle interaction. In the end accumulate the corresponding partial force to the current particle. The accuracy depends on the choice of θ.

3. Parallelising the algorithm and computing the independent components

There have been many parallelizations of Barnes-Hut algorithms both on shared memory machines and distributed memory machines. In general, two similar strategies have been pursued on both architectures: the spatial partitioning and the tree partitioning approaches.
One formulation for a parallel Barnes-Hut implementation could be partitioning the domain statically and assigning the sub-domains to various processors with the combined objectives of optimizing communication and balancing load. Shipping instead of data to processors, computations to processors where data resides could be also a good implementation.
Taking in consideration of the steps of the algorithm mentioned in section 2, we can see that step 1 requires modifications on a shared data structure, which needs synchronization. For step 2, computation of the center of larger cells depend on the center of masses of their sub-cells, which introduces data dependencies, but can be parallelized on a per-level basis. And for the third step, although we need other particles' mass for computing the force, we don't actually need to modify this information, and therefore we can perfectly parallelize all these computations of the forces on different particles independently.

4. Asymptotic complexity of Barnes-Hut algorithm

By approximating the particle problem to a clustering problem an error is occurred which is depending on the θ parameter choice we mentioned in section 2 and therefore the accuracy of the algorithm is also depending on the same parameter, for which the convergence is also slower (low-order method). It can be observed that the accuracy and complexity are a trade of in this algorithm and so the complexity for small θ is growing. For $\theta \to 0$, where the accuracy is high, the algorithm degenerates to $O(N^2)$ complexity. For roughly homogeneously distributed particles the number of cells that we need to compute at each particle will be roughly $O(\log N/\theta^3)$ for 3D or $O(\log N/\theta^2)$ for 2D (the power of θ depends on the number of dimensions) and therefore the total computational effort is $O(\theta^{-3} N \log N)$.

References

[1] J. Barnes and P. Hut, *A Hierarchical $O(n\log(n))$ force calculation algorithm*, Nature, vol. 324, December 1986.

[2] A. Grama, V. Kumar, A. Sameh, *Scalable parallel formulations of the Barnes–Hut method for n-body simulations*, Elsevier, 3Parallel Computing 24 797–822, 1998.