A Practical Comparison of N-Body Algorithms

Guy Blelloch and Girija Narlikar
guyb@cs.cmu.edu, girija@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

March 24, 1995

Abstract

This work compares three algorithms for the three dimensional N-body problem, the Barnes-Hut algorithm, Greengard's Fast Multipole Method(FMM), and the Parallel Multipole Tree Algorithm (PMTA) to determine which of the algorithms performs best in practice. Although FMM has a better asymptotic running time ($O(N)$ instead of $O(N \log N)$ for uniform distributions), the algorithm is more complicated and it is not immediately clear above what values of N it performs better in practice. We studied the dependence of accuracy on the variable parameters θ, p and α, and then compared the floating point operation counts of the three algorithms at similar levels of accuracy, for both charged and uncharged random distributions. At a high level of accuracy (RMS-error $\approx 10^{-5}$), the FMM did the least number of operations for $N > 10^4$, assuming both charged and uncharged distributions of points. At a lower level of accuracy, (RMS-error $\approx 10^{-3}$) for uncharged distributions, the FMM did not outperform Barnes-Hut even for $N > 10^8$. For charged distributions of particles, both the FMM and PMTA were comparable at low accuracy. The algorithms were implemented in the parallel language NESL.

1 Introduction

The Classical N-body problem simulates the evolution of a system of N bodies, where the force exerted on each body arises due to its interaction with all the other bodies in the system. N-body algorithms have numerous applications in areas such as astrophysics, molecular dynamics and plasma physics. The simulation proceeds over timesteps, each time computing the net force on every body and thereby updating its position and other attributes. If all pairwise forces are computed directly, this requires $O(N^2)$ operations at each timestep. Hierarchical tree-based methods have been developed to reduce the complexity, such as the Barnes-Hut algorithm [2], which is $O(N \log N)$ for uniform distributions, or the more complex Fast Multipole Method [9], which is $O(N)$ for uniform distributions. The Parallel Multipole Tree algorithm [6] is a hybrid of the Barnes-Hut and the Fast Multipole method. A recent algorithm by Callahan [8] has a time complexity of $O(N)$ for general distributions.

There have been several efforts to implement N-body code on parallel machines. The Stanford Splash benchmarks includes the Barnes-Hut algorithm as one of the applications [17]. They studied how to parallelize the code on a shared-memory model and derived speedups for up to 64 processors. Their algorithm has a serial bottleneck at the root of the tree, and

is therefore not appropriate for a much larger number of processors. A similar version, for a distributed memory machine, was implemented by Salmon [14]. Other researchers have implemented various N-body algorithms [19, 12, 3]. This previous work has shown that it is possible to get close to peak floating point performance by being careful about the communication in these algorithms [14, 3]. There has been little work, however, comparing the various n-body algorithms from a practical standpoint both in terms of error and running time. Board et.al have compared their PMTA algorithm to the FMM [6, 7] based on running times for a particular implementation. Our work extends this previous work in the following ways. Firstly, we compare all three algorithms, Barnes Hut, PMTA and FMM. Secondly, we consider both electrostatic and gravitational distributions of data, and show that the algorithms have quite different characteristics with the two types of forces. Thirdly, we use floating point operation counts to measure the work executed by the three algorithms in a manner independent of the machine and implementations. By deriving expressions for the operation counts of the algorithms we provide the ability to estimate their performances for large values of N without executing the code. This is particularly useful to help choose an algorithm and values for its parameters that will run well for a given level of accuracy and input distribution. Salmon [15] has given an upper bound for the number of interactions in the Barnes-But algorithm for both uniform and non-uniform distributions, but we give an exact expression for the expected number of interactions for uniform distributions that is much closer to the experimentally measured value. For example, at high accuracy, assuming a uniform distribution, Salmon's upper bound for $N = 10^5$ is about 4 times larger than the actual value, whereas our estimate is off by about 10%.

Our work involves the data-parallel implementation and comparison of the Barnes-Hut algorithm, the PMTA and the uniform FMM in three dimensions. We studied the dependence of the number of operations required by the algorithms and their accuracy on certain variable parameters, namely, θ in the Barnes-Hut algorithm, α in the PMTA and the number of terms p for all three algorithms. The goal was to compare the computational costs of the algorithms in practice, for various degrees of accuracy, for different sizes and distributions of input data. The FMM has two versions — a uniform version for uniform distributions of bodies, and a more complicated adaptive version for non-uniform distributions. Since we have tested the algorithms on points distributed randomly with uniform probability, we have restricted this work to the uniform version of the FMM. The Barnes-Hut and the PMTA, on the other hand, work well for both uniform and non-uniform distributions of data. We were interested in studying the trade-offs between asymptotic complexity and the hidden constants, and this work should help decide weather it is worth implementing the more complex adaptive $O(N)$ FMM instead of the simpler Barnes-Hut or PMTA algorithms for different distributions of data.

Section 2 describes the Barnes-Hut, the PMTA and the FMM in detail. The experimental results are given in section 3, in which we describe how the two algorithms were compared. Finally, the conclusions are given in section 4.

2 About the Algorithms

2.1 The Barnes-Hut Algorithm

The Barnes-Hut algorithm is based on a hierarchical octree representation of space in three dimensions. The algorithm has two phases. The first phase consists of constructing the octree by recursively subdividing the root cell containing all the particles into eight cubical subcells of equal size, until each subcell has at most one particle. Each cell contains the total mass and the position of the center of mass of all the particles in the subtree under it. In the second phase, the tree is traversed once per particle to compute the net force acting on it. We start at the root, and at each step, if the cell is *well separated* from the particle, we use the center of mass approximation to compute the force on the particle due to the entire subtree under that cell. Otherwise, each of its subcells is visited. A cell is considered *well separated* from a particle if its size, divided by the distance of its center of mass from the particle, is smaller than a parameter θ, which controls accuracy. In addition to the monopole (center of mass) approximation, higher order multipole terms can be used to increase accuracy.

2.2 The Fast Multipole Method (FMM)

The Fast Multipole Method (FMM) uses an octree similar to that of the Barnes-Hut algorithm. The uniform version builds a balanced octree. It distributes the particles into leaf cells, and computes their multipole expansions, followed by a bottom-up phase in which it constructs the multipole expansions of the parent cells by shifting and adding the expansions of its children. After the tree is built, it has a top-down phase in which the local expansion of the parent cell (which describes the potential field due to distant particles) is shifted to the center of each child, and added to the multipole expansions of the cells in the child's interaction list to form its local expansion. Finally, the local expansions at the leaf cells, along with direct interactions with particles in neighboring cells gives us the total force on each particle. The number of terms in the multipole expansions, p, controls the accuracy of the algorithm.

The primary difference between the FMM and the Barnes-Hut lies in the fact that the Barnes-Hut algorithm computes particle-cell interactions, whereas the FMM computes cell-cell interactions, thereby reducing its complexity. The level of accuracy can be adjusted for both the algorithms as required. However, the FMM has proven error bounds unlike the Barnes-Hut method.

2.3 Parallel Multipole Tree Algorithm (PMTA)

The PMTA is a hybrid of the Barnes-Hut and the FMM algorithms. It uses a rule similar to that of Barnes-Hut to determine the well-separatedness of two cells. Two cells are said to be well-separated from each other if the size of the bigger cell divided by the distance between the two cells is less than the parameter α. The tree is built as in the Barnes-Hut method, but a cell is recursively subdivided until it contains no more than m particles (instead of one particle as in the case of the Barnes-Hut algorithm). Then the tree is traversed top down for each leaf cell, and when a cell is found to be well-separated from the leaf cell, its multipole expansion is translated into a local expansion about the center of the leaf cell, and the rest of the subtree below that cell is not visited. All these translated local expansions are added and the gradient

is found to get the force due to the far field on every particle in the leaf. The particles in the leaf cell interact directly with the particles in all the leaf cells that are not well separated from it. The number of terms in the multipole expansion, p, and the separation parameter α can both be varied to control accuracy. A theoretical error bound for this algorithm is not known.

3 Experimental Analysis

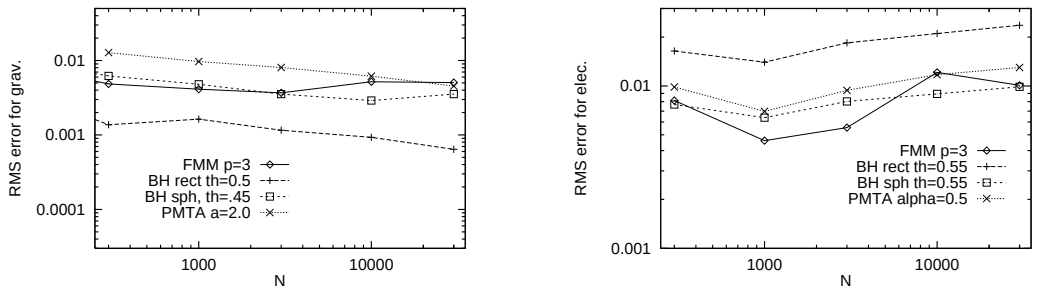
The goal of this work is to compare the constant factors in the running time of the three algorithms and their variants; in particular to determine how the constants depend on the desired accuracy. We chose to use floating-point operation count as the measure of computational work since measuring the running time on a particular machine would be machine and implementation specific. Clearly results based purely on floating-point operations will not exactly correspond to the running times on a particular machine, however, they should be quite representative. This is because the non-computational overheads of the algorithms are approximately equal. They parallelize quite easily and can take advantage of locality to reduce communication overheads. Previous implementations on parallel machines have managed to reduce the non-computational overheads of these algorithms to 15% or less [3, 19, 18]. We have assumed all floating point operations to have the same computational cost.

The original Barnes-Hut algorithm uses only the center of mass (monopole) approximation. We have also implemented two versions — one in rectilinear coordinates that uses quadrupole moments in addition to the center of mass approximation, and one in spherical coordinates that can have an arbitrary number of terms in the expansion. Both these versions increase the accuracy at the cost of computing the additional terms. At all levels of accuracy we found that they outperform the monopole version, so the results reported here are for these versions only.

We have implemented the uniform version of the FMM. Greengard defines the *near field* of a cell (the cells that are not well-separated from a given cell) as its first and second nearest neighbors. We have also implemented a variant in which the near field is taken as just the first nearest neighbors. This reduces the maximum number of cells in the interaction list of a cell from 825 to 189, but also reduces the accuracy (making it necessary to use a larger p). As it turns out, the additional work required by the extra p terms approximately balances the work saved by using fewer neighbors, so that the two versions are competitive for all the levels of accuracies that we have studied. Hence we have used the original version for comparison with the other algorithms.

Our implementations have been carried out in NESL [4], a data parallel language that supports nested data parallelism. It presents to the programmer a uniform memory-cost model of computation. Therefore issues like load balancing and data distribution, which are critical to the efficiency of the algorithm, are left to the NESL compiler to handle.

We empirically found the dependence of error on the variable parameters in the algorithms, namely, θ in the case of Barnes-Hut, α in the case of the PMTA and the number of multipole terms p for all three algorithms. Figure 1 shows the variation of error with N for fixed values of these parameters. Since they were found to vary in a similar manner, we measured errors for different values of the parameters for N in the range of 3000 to 10000 in order to find values of parameters that gave similar errors. The error we calculated is the RMS relative error which we define as the RMS of the maximum absolute error in the forces in the three dimensions for each



(a) (b)

Figure 1: Variation of sample errors with N for fixed parameter values for (a) gravitational and (b) electrostatic distributions. Errors for all the algorithms are found to vary in a similar manner, except for FMM, where the error increases when the number of levels in the tree increases, and then falls again gradually. The errors shown here are not all for the same level of accuracy.

particle, relative to the magnitude of the total force on the particle. This gives a value that is slightly higher than the RMS or mean of the absolute relative errors in all three dimensions.

3.1 Operations as a Function of θ and N for the Barnes-Hut Algorithm

In this section we first derive an expression that approximates the number of interactions as a function of θ and N. The expression is of the form:

$$b(\theta)N \log N - a(\theta)N$$

where $b(\theta) = O(1/\theta^3)$ and $a/b = O(\log \theta)$. We have also run experiments to measure the number of interactions processed by the Barnes-Hut algorithms over different values of N and θ. Our measurements fit well with the derived expression. As it turns out, for higher accuracies (lower θ) the second term almost cancels the first for most values of N that would be used in practice (up to $\approx 10^7$), such that the $N \log N$ asymptotic behavior is not applicable over this range.

We now consider how many cells each particle interacts with as a function of θ and N. The total number of interactions is N times this result. We will make some approximations in the analysis. Remember that a cell can interact directly with a particle (it is well-separated) if the ratio of the cell size to its distance from the particle is less than θ. Our analysis is based on calculating how many cells in each level of the tree a particle will interact with. The number of interactions at each level is constant from the bottom of the tree up to a fixed level, at which point none of the cells are well-separated from the particle. This is what gives the $N \log N - N$ form of the equation.

Let us assume for the sake of simplicity that the space is continuous, *i.e.*, it has no edges. In 3D the cell dimensions double at every level up the tree and the average number of particles in a cell increases 8-fold. If d is the distance between a particle and a cell of size s that is well-separated from it, then it follows that $d > s/\theta$. Similarly, the cells of size $2s$ (at the next higher level) that are well-separated from the particle lie at a distance $> 2s/\theta$ from it. Thus, the cells of size s that interact with the particle directly are more or less contained between

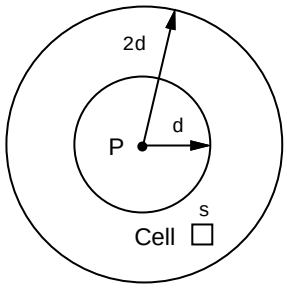


Figure 2: A cell of size s that interacts with particle P. Cells of size s inside the inner sphere will be expanded further and cells outside the outer sphere that interact with particle P will be larger than size s. The total number of cells of size s which interact with P will be $\approx (28\pi/3) \cdot d^3/s^3$. Since $\theta = s/d$, this is $28\pi/3\theta^3$.

spheres of radii s/θ and $2s/\theta$, centered around the particle (see figure 2). Hence the number of cells at that level (of size s) that interact with the particle are given by

$$f(\theta) = \frac{volume\ enclosing\ the\ cells}{volume\ of\ one\ cell} = \frac{4/3\pi[(2s/\theta)^3 - (s/\theta)^3]}{s^3} = 28\pi/3\theta^3 \ .$$

Given that a particle P interacts with $f(\theta)$ cells at each level, and that there are approximately 8^l particles in each cell at level l (l=0,1,... starting from the leaves), the total number of particles P interacts with up to level l (directly or indirectly) is $\sum_{i=0}^{i<=l} f(\theta) \times 8^i \approx f(\theta) \times 8^l$. Since there are totally N particles, all the particles will be covered upon reaching the level L given by

$$L \approx \log_8[N/f(\theta)] \ .$$

This means that P interacts with $f(\theta)$ cells at each of the L levels, giving a total of $f(\theta) \times \log_8[N/f(\theta)]$ interactions per particle and a total of

$$\begin{aligned} I(N,\theta) &= N\ f(\theta) \log_8[N/f(\theta)] \\ &= f(\theta) N \log_8(N) - f(\theta) \log_8(f(\theta)) N \end{aligned}$$

interactions across all particles. This explains why the number of interactions fits the $b(\theta)N \log N - a(\theta)N$ curve, with $a(\theta) = f(\theta)\log_8(f(\theta))$, $b(\theta) = f(\theta)$ and $a(\theta)/b(\theta) = \log_8(f(\theta))$.

Table 1 lists the values of $a(\theta)$ and $b(\theta)$ obtained from the measured number of interactions. They are close to the predicted values. It also lists the measured ratio of $a(\theta)/b(\theta)$ and the derived ratio $\log_8(f(\theta))$. Note that the slight deviation of the measured numbers from the derived expression can be explained by the following factors:

- The region containing the particles is not continuous, so the above expression is not valid for interactions with particles near the edges.
- Some cells of size s may interact with the particle even though they are more than $2s/\theta$ away from it, since their siblings and parents are less than $2s/\theta$ away from the particle.

θ	$a(\theta)$	$b(\theta)$	$a(\theta)/b(\theta)$	
	measured	*measured*	*measured*	*derived*
1.0	40.9	26.8	1.526	1.624
0.7	169.1	78	2.168	2.139
0.6	276.4	120.6	2.29	2.36
0.5	551.6	209.8	2.63	2.624
0.3	3242.7	840	3.86	3.36
0.2	11278	3077.5	3.66	3.956

Table 1: Measured values of $a(\theta)$ and $b(\theta)$ for some values of θ, the measured ratio $a(\theta)/b(\theta)$ and the derived ratio ($=\log_8(28\pi/3\theta^3)$).

- The expression for the number of cells in the region between the two spheres is not exact, and some cells may be only partially within the region.

Figure 3 shows the variation of the measured number of interactions with N for some values of θ.

3.2 Comparison of Operation Counts

We have used the number of floating point operations required to compare the work performed by the three algorithms. For the Barnes-Hut algorithm, we used the variation of the number of interactions with N for different θ (that was found in the previous section), to estimate the number of interactions needed for larger numbers, and multiplied that by the number of floating point operations needed for each interaction. We obtained a similar estimate for the PMTA in terms of α, p and m. For the FMM we added the number of floating point operations needed at each stage (for a given p), which is a reasonable estimate of the actual number since this is the uniform version of the algorithm and the distribution of points is random. The final expressions used for the floating point operation counts of the algorithms are listed in appendix A. In the case of the FMM, the optimum number of levels in the tree depends on N and p. The slope of the curve for the total work changes at values of N at which the optimal number of levels in the tree increases.

We compared the work performed by the algorithms for two different levels of accuracy. For uncharged distributions, at a lower level of accuracy (RMS error $\approx 10^{-3}$), where $\theta = 0.55$, $\alpha = 0.8$ and $p = 4$, all three do comparable work for practical values of N, with the Barnes-Hut in rectilinear coordinates doing the best for N as large as 10^8 million. Figure 4 shows the work done by the three algorithms at low accuracy for chargeless distributions. Figure 5 summarizes the results of the estimates for both levels of accuracy, for both charged(electrostatic) and uncharged(gravitational) distributions at $N \approx 10^7$. At lower accuracy (RMS error $\approx 10^{-3}$), the Barnes-Hut in rectilinear coordinates with the quadrupole moment performs the least number of operations for gravitational distributions. On the other hand, for electrostatic distributions, at low accuracy (RMS error $\approx 2 \times 10^{-3}$) both the FMM and PMTA do comparable amounts of work and do better than the Barnes-Hut versions. Finally, at high accuracy (RMS error $\approx 2 \times 10^{-5}$ for gravitational, $\approx 6 \times 10^{-5}$ for electrostatic), the FMM outperforms both the other algorithms. The rectilinear Barnes-Hut with quadrupole moments performs better for

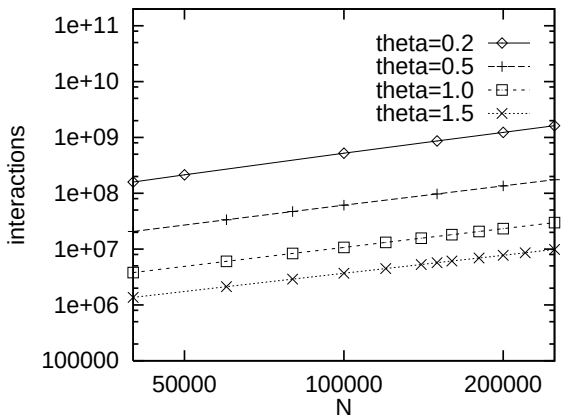


Figure 3: Measured number of interactions as a function of N in Barnes-Hut for four different values of θ.

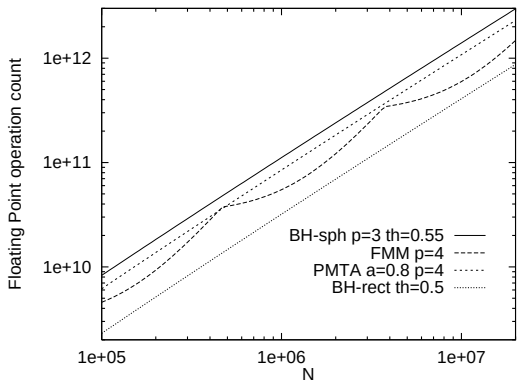


Figure 4: Floating point operation counts of the algorithms for gravitational (chargeless) distributions at low accuracy (RMS error $\approx 10^{-3}$). The bumps in the FMM curve occur due to the change in optimal number of levels in the FMM tree as N increases.

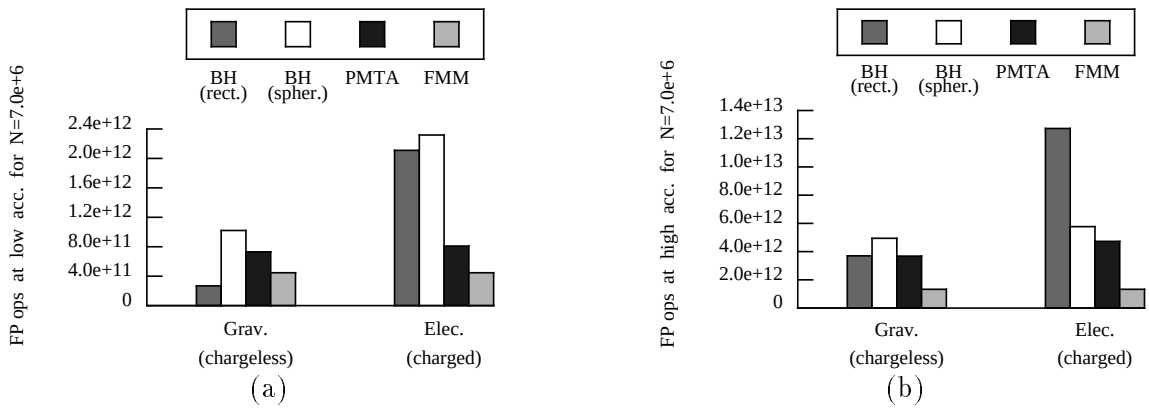


Figure 5: Floating point operation counts for the Barnes-Hut (in both rectilinear and spherical coordinates), the PMTA and the FMM for electrostatic and gravitational distributions. The figures show operation counts for $N = 7 \times 10^6$, at (a) low accuracy, and (b) high accuracy. The parameter values for the operation counts shown above are as follows. Low acc. (grav.) : BH-rect. $\theta = 0.55$; BH-sph. $\theta = 0.55$, $p = 3$; PMTA $\alpha = 0.8$, $p = 4$, $m = 80$; FMM $p = 4$. Low acc. (elec.) : BH-rect. $\theta = 0.28$; BH-sph. $\theta = 0.63$, $p = 6$; PMTA $\alpha = 1.5$, $p = 6$, $m = 190$; FMM $p = 4$. High acc. (grav.) : BH-rect. $\theta = 0.2$; BH-sph. $\theta = 0.58$, $p = 8$; PMTA $\alpha = 0.4$, $p = 6$, $m = 170$; FMM $p = 7$. High acc. (elec.) : BH-rect. $\theta = 0.135$; BH-sph. $\theta = 0.5$, $p = 7$; PMTA $\alpha = 0.4$, $p = 7$, $m = 220$; FMM $p = 7$.

gravitational distributions because the lower order terms in the expansion dominate. Since this is not the case for electrostatic distributions, more than 3 terms are needed to accurately calculate the potential and force.

A few optimizations have been suggested for multipole-based algorithms, such as the use of FFTs to reduce the cost of translating expansions [10, 16, 13] at the cost of greater memory requirements. However, it has been reported that this optimization gives a overall speedup of less than 2 at a high level of accuracy ($p = 12$), and little or no speedup at lower accuracy ($p = 4$) [5]. Hence we have not included the FFT version in our experiments. Another optimization suggested is to reduce the number of multipole-to-local translations in the FMM by using what is called *parental conversion* [11]. This optimization uses the multipole expansion of the parent cell for translation into a local expansion if all eight of its children are in the interaction-list. This reduces the maximum size of the interaction list from 825 to 189 at the loss of some accuracy. This loss of accuracy is compensated for by using one extra term in the expansion [11]. We found that at low accuracy (p=4) this gave a speedup of about $1.3 - 1.4$ and less at higher accuracy (higher p). Hence this optimization too has not been included in our experiments.

4 Conclusions

The conclusions of this work can be summarized as follows.

- The FMM always performs better than the other two algorithms, except for gravitational distributions at low accuracy, for which it performs less than twice the work performed by the Barnes-Hut algorithm.

- At low accuracy, for gravitational distributions, the operation counts of all three algorithms are nearly equivalent, with the Barnes-Hut doing best for values of N up to 10^8 and more. For electrostatic distributions, the PMTA and the FMM do nearly equivalent amounts of work and outperform the Barnes-Hut versions.
- The negative linear term plays a significant role in the complexity of the Barnes-Hut algorithm. Hence Barnes-Hut does not perform strictly as $N \log N$ for reasonable values of N and θ.
- The FMM always outperforms the PMTA, unlike the results obtained by [6, 7]. This is probably because we have a different criteria of measuring errors.
- Barnes-Hut performs better for gravitational distributions using rectilinear coordinates and electrostatic distributions using spherical coordinates. This is because we have used only up to the quadrupole moment in rectilinear coordinates. Quadrupole approximation works well for gravitational distributions since the first few multipole terms dominate, which is not true in the case of electrostatic distributions.
- Electrostatic distributions require more work for the same level of accuracy as compared to gravitational distributions.

The three algorithms have high memory requirements if we exploit all the available parallelism, hence timing fully parallel versions for large data sets was beyond the scope of this work. Time has not yet permitted us to study the performance of the algorithms on non-uniform distributions, such as the Plummer model [1].

All the algorithms are highly parallel in nature. This made it necessary to reduce some of the parallelism, especially for runs with high accuracy, since their memory requirements increase very rapidly with increasing accuracy. The Barnes-Hut in rectilinear coordinates was the simplest to code. In spherical coordinates, all three algorithms were comparable in terms of difficulty of coding. However, if the particle distributions are non-uniform, the more complex adaptive version of the FMM will have to be used.

References

[1] S.J. Aarseth, M. Henon, and R. Wielen. Numerical methods for the study of star cluster dynamics. *Astronomy and Astrophysics*, 37(2):183–187, 1974.

[2] J.E. Barnes and P. Hut. A hierarchical $O(N \log N)$ force calculation algorithm. *Nature*, 324(4):446–449, December 1986.

[3] S. Bhatt, M. Chen, C.Y. Lin, and P. Liu. Abstractions for parallel N-body simulations. In *Proceedings Scalable High Performance Computing Conference*, pages 26–29, 1992.

[4] Guy E. Blelloch. Nesl: A nested data-parallel language. Technical Report CMU-CS-93-129, CMU, School of Computer Science, April 1993.

[5] J.A. Board and W.S. Elliot. Fast fourier transform accelerated fast multipole algorithm. Technical Report 94-001, Duke University Dept of Electrical Engineering, 1994.

[6] J.A. Board, Z.S. Hakura, W.S. Elliot, D.C. Gray, W.J. Blanke, and J.F. Leathrum Jr. Scalable implementations of multipole-accelerated algorithms for molecular dynamics. Technical Report 94-002, Duke Univaersity, 1994.

[7] J.A. Board, Z.S. Hakura, W.S. Elliot, and W.T. Rankin. Scalable variants of multipole-accelerated algorithms for molecular dynamics. Technical Report 94-006, Duke University Dept of Electrical Engineering, 1994.

[8] P.B. Callahan. Optimal parallel all-nearest-neighbours using the well-separated pairs decomposition. In *34th Annual Symposium on Foundations of Computer Science*, pages 332–340, Palo Alto, 1993. IEEE.

[9] L. Greengard. *The rapid evaluation of potential fields in particle systems*. The MIT Press, 1987.

[10] L. Greengard and V. Rokhlin. A fast algorithm for particle simulation. *Journal of Computational Physics*, 73(325), 1987.

[11] J.F. Leathrum, Jr., J.A. Board, and W.S. Elliot. The parallel fast multipole algorithm in three dimensions. Technical report, Duke University Dept of Electrical Engineering, 1992.

[12] P.H. Mills, L.S. Nyland, J.F. Prins, and J.H. Reif. Prototyping N-body simulation in proteus. In *Proceedings Sixth International Parallel Processing Symposium*, pages 476–482, 1992.

[13] V.Y. Pan, J.H. Reif, and S.R. Tate. The power of combining the techniques of algebraic and numerical computing. In *32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 92)*, pages 703–713, 1992.

[14] J. Salmon. Parallel $N \log N$ N-body algorithms and applications to astrophysics. In *COMPCON Spring '91, Digest of Papers*, pages 73–78, 1991.

[15] J.K. Salmon. *Parallel hierarchical N-body methods*. PhD thesis, Caltech University, 1991.

[16] K.E. Schmidt and M.A. Lee. Implementing the fast multipole method in three dimensions. *Journal of Statistical Physics*, 63(5):1223–1235, 1991.

[17] J.P. Singh, W.D. Weber, and A. Gupta. Splash: Stanford parallel applications for shared-memory. *Computer Architecture News*, 20(1):5–44, March 1987.

[18] M. Warren and J. Salmon. Astrophysical n-body simulations using hierarchical tree data structures. In *Proceedings of Supercomputing*, 1992.

[19] F. Zhao and S.L. Johnsson. The parallel multipole method on the connection machine. *SIAM Journal on Scientific and Statistical Computing*, 12(6):1420–1437, november 1991.

A Expressions used to estimate floating point operations counts

Let

$W(x_1, x_2, \ldots)$ = Number of floating point operations performed by the algorithm to calculate force on each particle in terms of variable parameters $x_1, x_2, \ldots$,
θ = separation parameter for Barnes-Hut,
α = separation parameter for PMTA,
p = number of terms in the multipole expansion ($= 1, 2, 3, \ldots$),
l = number of levels in the FMM tree ($= 0, 1, 2, \ldots$ starting from the root), and,
N = number of particles.

$W(x_1, x_2, \ldots)$ does not include the cost of building the octree, which is negligible compared to the cost of force calculation.

A.1 Barnes-Hut in rectilinear coordinates

$$W(N, \theta) = C_q \cdot N \cdot f(\theta) \cdot \left(\log_8 \frac{N}{f(\theta)} - 1 \right) + d \cdot N \cdot f(\theta)$$

where $f(\theta)$ is defined in section 3.1 and C_q is the cost of evaluating the gradient using up to quadrupole moments.

C_q = cost of gradient for uncharged distributions using only monopole and quadrupole moments (the dipole moment vanishes if evaluated about center of mass) = 50 ,
C_q = cost of gradient for charged distributions using monopole, dipole and quadrupole moments = 70, and,
d = cost of a direct interaction = 13.

A.2 Barnes-Hut in spherical coordinates

$$W(N, \theta, p) = g(p) \cdot N \cdot f(\theta) \cdot \left(\log_8 \frac{N}{f(\theta)} - 1 \right) + d \cdot N \cdot f(\theta)$$

where $g(p)$ = cost of gradient for a multipole expansion of p terms $= 15p(p+1) + 5p + 4$.

A.3 PMTA

$$W(N, \alpha, p, m_{avg}) = t(p) \cdot \frac{N}{m_{avg}} \cdot f'(\alpha) \cdot \log_8 \left(\frac{N}{m_{avg} f'(\alpha)} \right) + d \cdot \frac{N m_{avg}}{7} \cdot f'(\alpha) + g(p) \cdot N$$

where

$t(p)$ = cost of translating a multipole expansion having p terms into a local expansion
= $3p^2(p+1)^2 + 5p(p+1)$,
m_{avg} = average population of leaf cells ($\approx m/2$), and,
$f'(\alpha)$ = average number of cells a leaf cell interacts with at each level (measured).

We have assumed that a leaf interacts with a constant $f'(\alpha)$ cells at each level, and the average radius of the sphere around the leaf cell containing all the cells that have interacted with it so far doubles at every level up the tree. This means that the volume of cells interacting with it at each level increases 8-fold. Hence the number of leafs it directly interacts with at the lowest level is $\approx (8/7 - 1) \cdot f'(\alpha) = f'(\alpha)/7$. This estimate agrees with the numbers we have measured experimentally for moderate values of N. We used $f'(1.5) = 374$, $f'(0.8) = 642$ and $f'(0.4) = 1910$.

A.4 FMM

$$W(N, p, l) = d \cdot near(l) \cdot \left(\frac{N}{8^l}\right)^2 + g(p) \cdot N + t(p) \cdot \sum_{i=2}^{l} trans(i)$$

where

$near(l)$ = total number of leaf pairs that interact directly in a tree with l levels.
$neal(0) = 1$,
$near(l) = 63 \cdot 2^{3l} - 225 \cdot 2^{2l} + 270 \cdot 2^l - 108$ for $l \geq 1$, and,

$trans(l)$ = the total number of local-to-local and multipole-to-local translations at the level l in the tree.
$trans(2) = 1352$,
$trans(l) = (1 + 875) \cdot 2^{3l} - 6750 \cdot 2^{2l} + 16740 \cdot 2^l - 13608$ for $l \geq 3$.