Parallel Implementation of Approximate Atomistic Models of the AMOEBA Polarizable Model

Omar Demerdash[1] and Teresa Head-Gordon[1,2,3,4*]

[1]Department of Chemistry, [2]Department of Bioengineering, [3]Department of Chemical and Biomolecular Engineering, [4]Chemical Sciences Division, Lawrence Berkeley National Laboratory, University of California, Berkeley, CA 94720

In this work we present a replicated data hybrid OpenMP/MPI implementation of a hierarchical progression of approximate classical polarizable models that yields speedups of up to ~10 compared to the standard OpenMP implementation of the exact parent AMOEBA polarizable model. In addition, our parallel implementation exhibits reasonable weak and strong scaling. The resulting parallel software will prove useful for those who are interested in how molecular properties converge in the condensed phase with respect to the MBE, it provides a fruitful test bed for exploring different electrostatic embedding schemes, and offers an interesting possibility for future exascale computing paradigms.

***Corresponding author**
thg@berkeley.edu

1. INTRODUCTION

One of the most tractable classical models for condensed phase simulation is the assumption of a pairwise additive, fixed charge force field. Such models are widely available and often highly successful for many chemical systems, due to the many years devoted to optimization of their parameters, and benefits of long sampling trajectories arising from their highly scalable software implementations. Even so, pairwise additive treatments do often break down for heterogeneous environments[1-3] and lack transferability[4-6], although it can't always be anticipated why or for which chemical systems. In principle, mutually polarizable models offer a significant improvement in the physics of classical force fields[6-32]. However, the corresponding complexity of an advanced polarizable model increases significantly enough so that they are more difficult to parameterize, statistical convergence of condensed phase observables is harder to achieve, and optimal software implementations become more elusive. Therefore well-defined approximations to full classical polarization are of interest.

We have used the many-body expansion (MBE) to the total potential energy of an N-body system

$$U = U_1 + U_2 + U_3 + \cdots \tag{1}$$

to define a set of hierarchical approximations to the full mutual polarization of a classical inducible point dipole model, AMOEBA.[33] We showed that when Eq. (1) is truncated at the level of trimers of water molecules,

$$U_1 = \sum_{i=1}^{N} U(i), \quad U_2 = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} U(i,j) - U(i) - U(j), \tag{2}$$

$$U_3 = \sum_{i=1}^{N-2}\sum_{j=i+1}^{N-1}\sum_{k=j+1}^{N} U(i,j,k) - U(i,j) - U(i,k) - U(j,k) + U(i) + U(j) + U(k)$$

it captures direct polarization exactly, i.e., where the induced dipoles respond only to the permanent electrostatic field.[6,33,34] At this level of approximation it requires extensive reparameterization to recapture the missing mutual polarization response, as we have shown for the so-called inexpensive iAMOEBA model.[34] In turn, by including mutual polarization up through trimers of water molecules, and where the induced dipole interactions of the small subsystems are generated in isolation of its surrounding, the 3m-AMOEBA recaptures some of this missing mutual polarization energy[6], but now with large errors in the polarization forces.[33] We demonstrated how to improve the MBE convergence of gradients by embedding the polarization response of dimers and trimers within a more complete representation of the fixed electrostatics of the entire system.[33] We then introduced a practical scheme for representing electrostatic embedding by fragmenting the system into M large clusters, i.e. to define a body as being comprised of 100's of water molecules, and show that the resulting 3M-AMOEBA model improves accuracy of not just energy but forces as well.[33]

The very nature of the MBE approximation increases computational cost but at the same time admits a trivial parallel implementation, as the polarization energy, gradient, and virial of the subsystems are independent of one another. We have reported on the CPU parallelization of the direct polarization iAMOEBA in other work.[35] In this paper, we provide a detailed account of the parallel implementation of the approximate 3m-AMOEBA and 3M-AMOEBA mutual polarization models in the reference TINKER7 code. The paper is outlined as follows. In Section 2 we introduce the AMOEBA model[11,36] and its approximations whereby a body is defined as either one water molecule, which defines the 3m-AMOEBA model, or the system is fragmented into M large clusters to define the 3M-AMOEBA model. In Section 3 we describe their hybrid MPI/OpenMP replicated data strategy with an optimized load-balancing scheme implemented on a modern CPU hardware architecture. In Section 4 we present timings and weak and strong scaling profiles for the parallel implementations of the 3m-AMOEBA and 3M-AMOEBA approximate models. In Section 5 we discuss future directions and potential uses of the approximate AMOEBA models and provide a brief summary of results.

2. AMOEBA POLARIZATION MODEL

The classical polarization model AMOEBA (Atomic Multipole Optimized Energetics for Biomolecular Applications) belongs to the class of molecular mechanics force fields that aims for higher accuracy than fixed partial charge potentials due to explicit accounting of many-body polarization.[11,36] AMOEBA has the following physical functional form for the interactions among atoms

$$U = U_{bond} + U_{angle} + U_{b\theta} + U_{oop} + U_{torsion} + U_{VDW} + U_{elec}^{perm} + U_{elec}^{ind} \tag{3}$$

where the first five terms describe the valence interactions and the last three terms are the nonbonded terms, including van der Waals (vdW), electrostatic contributions from permanent atomic multipoles up through quadrupoles, as well as polarizable dipoles. The bottleneck in the evaluation of Eq. (1) resides in the last two terms comprising the fixed electrostatics as well as the N-body polarization term, which involves an extensive amount of algebra in the calculation of an iterative solution of the inducible dipole moments to self-consistency.[6]

The current reference implementation of the AMOEBA model in the TINKER7 package is ~100 times the cost of a fixed monopole model when optimally implemented in standard community codes such as Amber[37], NAMD[38], and OpenMM[39]. The primary computational cost of the AMOEBA model is the N-body interaction of an inducible dipole interacting with the electric field arising from the fixed multipoles and other inducible dipoles. We can express the total polarization energy, U_{pol}, as the induible dipole $\vec{\mu}_{ind}$ dotted into the electric field, $\vec{E}$

$$U_{pol} = -\frac{1}{2}(\vec{\mu}_{ind})^T \cdot \vec{E} \tag{4a}$$

where

$$E_i = \sum_{j}^{13N} T_{ij} M_j \tag{4b}$$

The elements of $\vec{\mu}_{ind}$ are defined as

$$\mu_{i,\gamma} = \alpha_i \left(\sum_{j}^{13N} T_{ij,\gamma} M_j + \sum_{k}^{3N} T'_{ik,\gamma\delta} \mu_{k,\delta} \right) \qquad \gamma, \delta = x, y, z \tag{4c}$$

where α_i is the isotropic polarizability of atom i, T_{ij} is the interaction tensor between atoms i and j containing derivatives of $1/r_{ij}$ according to the permanent multipole expansion, and M_j are the permanent multipole moments; the T and M tensors in the first term of (3c) encompass the 13 permanent multipole moments for the AMOEBA potential (q, μ_x, μ_y, μ_z, Q_{xx}, Q_{xy}, Q_{xz}, Q_{yx}, Q_{yy}, Q_{yz}, Q_{zx}, Q_{zy}, Q_{zz}). Thus, the first term in Eq. (4c) corresponds to the direct polarization response, and when the last term is ignored, it defines the functional form of the iAMOEBA model[34]. The last term in Eq. (4c) represents the electric field at atom i due to the induced dipoles at all other atomic sites, μ_j, where T_{ij}' is the interaction tensor between atoms i and j containing derivatives of $1/r_{ij}$ according to the inducible dipole-dipole interactions. This must be solved self-consistently and contributes to the expense of the AMOEBA model. Under the 3m-AMOEBA and 3M-AMOEBA models, N is replaced by the fragment size n, and the computational cost of the polarization calculation is reduced by the trivial parallelization of the independent subsystems.

3. PARALLEL IMPLEMENTATION METHOD

The computational efficiency of the 3m-AMOEBA and 3M-AMOEBA models rests not only on the reduced expense of the polarization calculations of the individual subsystems, but also on the rapid decay of 2- and 3-body energies with inter-body distance and the consequent amenability of distance cutoffs, and thus, neighbor lists. It is clear that while an individual calculation under the 3-body approximation is cheaper, if one did not exploit distance cutoffs, one would be faced with an O(n^3) calculation (n=total number of bodies) that would not be faster than the original fully mutually polarizable implementation even with the best parallelization strategy. When one makes use of distance cutoffs and neighbor lists, the O(n^3) computational complexity is reduced to a linear O(k*n)

calculation, where the prefactor k depends on the distance cutoff for the triplets since their factorial increase dominates the cost. Due to our use of distance cutoffs, we must therefore apply a smoothening function to the polarization energy, gradient, and virial in a short window just inside the cutoff distance ($r_{smoothen} < r < r_{cut}$) to ensure conservation of energy and a stable temperature, using well known techniques developed by Brooks and co-workers[40].

3m-AMOEBA Model. We use a parallel, hybrid MPI/OpenMP, replicated data (atom-decomposition) scheme wherein the atomic coordinates are known to all tasks and the final energy, force, and internal virial are accumulated on a single task. We implemented the atom-decomposition scheme instead of the other well-known parallel strategy of spatial decomposition, since the relevant 3-body interactions are not easily partitioned into spatial domains since they are more diffuse than pairs, thereby precluding a straightforward spatial decomposition approach. Instead, the 1-, 2-, and 3-body polarization energies, gradients, and internal virials are accumulated in a single loop structure using nested pairwise neighbor lists. Furthermore, we can evaluate the polarization energy and forces for pairs and triplets of water molecules exactly by simple matrix inversion, with no need to use a SCF solver; indeed in our development of 3m-AMOEBA we found that matrix inversion was faster than either Cholesky factorization or SCF for the small subsystem sizes.

We can also avoid a reciprocal-space calculation of polarization and perform a real-space only calculation without any significant loss of accuracy, rendering each call to calculate the polarization energy, gradient, and internal virial of the subsystems much more lightweight. An added benefit of the single-molecule body is that the 1-body energy becomes zero in the absence of reciprocal space in the AMOEBA force field. Early on in our development of the 3m-AMOEBA model we ensured that neglect of reciprocal space introduced errors in the polarization energy that were an order or magnitude or less than the errors associated with the 3-body approximation itself, which we reported in previous work[6]. Errors associated with neglect of Ewald for the 3m-AMOEBA approximation are presented in Table S1. Moreover, in our appraisal of the robustness of the MBE for condensed-phase properties of water, we found that neglect of Ewald for polarization under the standard, full N-body AMOEBA model gave a correct O-O radial distribution function and accurate densities; this will be reported on in more detail in future work featuring the performance of the MBE in the prediction of condensed-phase properties of water.

A FORTRAN-like pseudocode version of OpenMP parallelization on a single MPI task for 3m-AMOEBA is illustrated in Figure 1. However, a complete and efficient parallel implementation requires a scheme whereby work associated with the calculation to be parallelized is apportioned among the MPI tasks as equally as possible using load balancing to ensure better strong scaling. The rationale underlying the load-balancing scheme that we implemented becomes clear when one inspects the pseudocode for the polarization calculation in Figure 1. For example, when we consider a given molecule represented by one iteration of the outer do loop in Figure 1, we see that the work or "load" associated with that molecule is determined by the number of neighbors of that molecule, represented by the second do loop, as well as the number of neighbors of those neighbors, which is represented by the innermost loop. Our load-balancing strategy therefore recognizes the need to partition the work according to the number of neighbors of each molecule or body and the neighbors of those neighbors. Our load balancing among multiple MPI tasks is shown in FORTRAN-like pseudocode in Figure 2.

As dictated by Amdahl's law, we parallelized the next most expensive components of the calculation, namely the real-space contribution to the permanent multipole electrostatic energy, gradient, and virial as well as the van der Waals energy, gradient, and virial. For these other types of non-covalent interactions, the load balancing is straightforwardly determined based on the total number of neighbors of each atomic site. Unlike an OpenMP-only code, we can have the covalent interactions execute on one MPI task (e.g., task_id=0), the reciprocal-space portion of permanent multipole electrostatics on another task (task_id=1), the real-space permanent electrostatics on a separate set of MPI tasks not equal to 0 or 1, and the vdW on yet another separate set of tasks. The neighbor list update is the next most expensive operation after the energy/force/gradient calculation, and in our implementation all relevant neighbor lists (polarization, real-space electrostatics, and vdW) are built *de novo* on the appropriate tasks at the start of the simulation, and the relevant portions are updated for the vdW and real-space electrostatics on each task according to the atoms for which a given task is responsible. The entire polarization neighbor list must be updated for all tasks, as is evident from the implementation of polarization outlined in Figure 1.

The full pseudocode for a single MD time step using the hybrid OpenMP/MPI replicated data strategy for 3m-AMOEBA is shown in Figure 3. The equations of motion are integrated on a single MPI task, the so-called "master" task. Due to the fact that we have a replicated data parallel implementation, there are a few global communications that must be performed at each time step, such as the updated coordinates that must be broadcast to all tasks, or in a constant pressure (NPT) simulation, where the periodic box dimensions must be broadcast as well at each time step. The total polarization energy, gradient, and virial, as well as the corresponding quantities for the real-space permanent electrostatics and vdW, must be summed on the master task.

3M-AMOEBA Model. As we have shown elsewhere[33], the level of accuracy for 3m-AMOEBA gradients is quite poor, requiring higher order terms in the MBE in Eq. (1). However, if we embed the polarization response of the water dimer and trimer systems within a larger electric field environment, we can greatly reduce the errors in forces.[33] We can recover a pragmatic approximation to the electrostatic embedding result by changing the definition of a body to one defined by a larger cluster of tens to hundreds of water molecules, *M*, which renders a more accurate model for forces in 3M-AMOEBA. A trivial parallel implementation is also possible for 3M-AMOEBA, but in this case we have to contend less with a curse of numbers, as in 3m-AMOEBA, but now rather with much larger subsystem sizes. Figure S1 shows the simultaneous accumulation of 1-body, dimer, and trimer contributions to the polarization and permanent electrostatics, which differs from that in Figure 1 for 3m-AMOEBA. The subsystem-specific 1-body results must be saved for the 2-body and 3-body terms, and the relevant dimer results must be saved to calculate the 3-body term. The final polarization energy, gradient, and virial is assembled after the 2-body and 3-body terms have been calculated according to Eq. (2).

The need to enforce simultaneous execution of the larger subsystem calculations informs the load balancing strategy for 3M-AMOEBA (Figure S2). Now, we see that the load balancing is determined by the cluster size, and that there is an inevitable need to apportion the neighbors of a given cluster among tasks. A complete MD time step is displayed in pseudocode in Figure S3 for 3M-AMOEBA, in which there are a few apparent distinctions at this level from the implementation of 3m-AMOEBA, namely, that we must allocate and zero out matrices storing subsystem specific 1-body and dimer terms for the to be used for the evaluation of 2- and 3-body terms per Eq. (2). Secondly, the larger subsystem sizes under 3M-AMOEBA necessitate the use of subsystem-specific neighbor lists for each of the monomer, dimer, and trimer subsystems.

In principle we can formulate ideal serial timings for the 3M-AMOEBA model, $t_{3M\text{-}AMOEBA}$, i.e. without parallelization as

$$t_{3M-AMOEBA} \sim M * t_{n,AMOEBA} + M^2 * t_{2n,AMOEBA} + M^3 * t_{3n,AMOEBA} \tag{5}$$

where $t_{n,AMOEBA}$ and where $t_{2n,AMOEBA}$ corresponds to the regular 12-core OpenMP timings of permanent electrostatics and polarization for AMOEBA using TINKER7 for a system size of *n=N/M*; the timing $t_{3n,AMOEBA}$ corresponds to just the polarization calculation. It is interesting to note that we can use a coarse-grained representation of the PME grid (but with a larger real-space cutoff) without a great deal of error for the larger sub-systems, which reduces the cost of these calculations, however it is also true that this approximation will worsen as cluster size *n* decreases back to a single water molecule. For parallel timings we assume that we have $12*(M+M^2+M^3)$ available cores, such that each monomer, dimer, and trimer cluster calculation is performed on a single MPI task.

Table 1 shows the resulting hypothetical parallel timings when the system is fractionated into 10, 20, and 30 clusters, yielding speed ups of up to ~2.75-8.0 for this modest size case (7000 water molecules) compared with the AMOEBA model evaluated with OpenMP in TINKER7. Note that the number of required cores grows as $\sim 12*M^3$, and while it would present an interesting hypothetical case for an exascale calculation, it is not tenable on current HPC architectures. However, the use of cluster distance cutoffs would restrict the growth of M^3 calculations to M, and thus reduce the core count to 12*(M+M^2+~M). Under these assumptions we can estimate parallel timings with cluster cutoffs according to

$$t_{3M-AMOEBA} \sim M * t_{n,AMOEBA} + M^2 * t_{2n,AMOEBA} + M * t_{3n,AMOEBA} \qquad (6)$$

Again, given perfect load balancing and assuming Eq. (6), we achieve a hypothetical speed up of 3.75-10.7 compared with the AMOEBA model evaluated with OpenMP in TINKER7 on the 7000 water molecule case (Table 1). These hypothetical timings are useful for compariosns to the actual timings of the parallel implementation described in Figures S1-S3 given in the next section.

4. TIMING RESULTS

The 3m-AMOEBA and 3M-AMOEBA models were implemented in the TINKER7 software package using the hybrid MPI/OpenMP replicated data strategy presented in Section 3. Parallel timings in nanoseconds per day were generated from short molecular dynamics simulations of 100-2000 time steps using the velocity Verlet integrator, a 1 fs time step, 10^{-5} D convergence for the induced dipoles for the 3M-AMOEBA and AMOEBA calculations, on system sizes ranging from 1600 to 288,000 water molecules. In the comparisons of the mutual polarization models below, we run using the optimal set up of each software implementation. The 3m-AMOEBA timings were obtained from runs on anywhere between 700 up to 6,144 cores, while the 3M-AMOEBA model runs were obtained on 3600 cores. The reference AMOEBA timings were obtained using the number of threads that gave the best speedup in TINKER7 (12 OpenMP threads on a node where 12 cores share local cache memory). We also compare our performance to the Amber MPI implementation of AMOEBA (pmemd.amoeba) that uses MPI and spatial decomposition. For the timings of AMOEBA in Ambers pmemd.amoeba, we systematically varied the number of cores in a range of 12 to 384, with 1 MPI task/core in this MPI-only implementation. We found that speedups could be obtained for up to 48-96 cores for most systems, and up to 192 cores for the largest system of 288,000 water molecules.

Table 2 provides the timing and relative speed up results for 3m-AMOEBA compared to the full AMOEBA model as implemented in the other CPU-based implementations in TINKER7 and Amber. In this work, we restrict our comparisons to CPU-based implementations. A fair comparison with the leading GPU-based implementation of AMOEBA in OpenMM would necessitate an implementation of our approximate models in a comparable GPU-based implementation. Hence, to ensure a like-to-like comparison, we compare timings with CPU-based implementations only. It should be noted that the 3m-AMOEBA model is quite different than AMOEBA, but the point is to show that an approximate but potentially accurate model like 3m-AMOEBA (after reparameterization as we did for iAMOEBA.[34]) could serve as a replacement of the AMOEBA force field due to improvements in computational speedups. We observe speedups for 3m-AMOEBA for systems of at least 4800 atoms and larger; for 3m-AMOEBA the speedups range from ~1.4 for 1600 waters and up to ~6.9 for the largest box of 288,000 water molecules.

Table 3 provides the timing and relative speed up results for 3M-AMOEBA model under different fragmentation schemes defined by M, using cluster distance cutoffs, compared to the full AMOEBA model as implemented in TINKER7 and Amber. 2-body cutoffs in 3M-AMOEBA were defined according to the distance between the centroids of the two clusters. Similarly, 3-body cutoffs were based again on inter-centroid distances, but here the cutoff was based on the sum of the 2 shortest inter-centroid distances of the 3 possible distances. Again although 3M-AMOEBA is an approximation to the AMOEBA potential, polarization energies are reproduced within a fraction of 1% and gradients errors are on average as low as ~20-30% compared to the parent gradients. While we observe speedups of 1.8-3.4 for 3M-AMOEBA when systems are fragmented into 10 clusters, we find significant increase in the speed up factor as M increases, yielding 4.9-7.7 increases in ns/day when M=20, and factors of 3.8-10.8 when M=30 clusters.

In addition to simply the speedup, other metrics must be obtained in order to assess the efficiency of our implementations. We first assessed the weak scaling, which is a measure of the computational cost as a function of simultaneously increasing the system size and the number of cores. More specifically, this is an assessment of how the cost of a calculation grows as the number of cores and the system size are simultaneously increased. In the case of 3m- and 3M-AMOEBA, the complexity is theoretically ~O(N) due to our use of distance cutoffs with neighbor lists. In light of this, for the evaluation of weak scaling, we increased the number of cores in a manner directly proportional to the system size, maintaining a constant ratio of system size to number of cores. In the case of ideal weak scaling, the computational cost should remain bounded. We see that for both 3m-AMOEBA (Figure 4a) and 3M-AMOEBA (Figure 4b), the cost remains bounded until ~2000-2500 cores, after which the cost grows, likely due to the increase in the communication cost. Strong scaling, in contrast, is a measure of the decrease in computational cost (speedup) as a function of increasing number of cores for a fixed system size. The speedups for the 3m-AMOEBA implementation scale linearly or very close to linearly for over 1000 cores, and then diverging from linear scaling significantly at ~1500 cores (Figure 5a); we even observe supra-linear strong scaling for 3M-AMOEBA (Figure 5b). We point out that the scaling is similar for the 21,000 and 32,000 atom systems, which are commensurate with systems that are considered routinely tractable in modern MD simulations using fixed charge force fields.

CONCLUSION

In this work we presented a hybrid MPI/OpenMP implementation of two approximate polarization models, 3m-AMOEBA and 3M-AMOEBA, using the method of atom decomposition. We show that our parallel implementation yields speed ups by factors of ~2-11 compared to the parent AMOEBA potential depending upon whether we are using small or large fragments; both models are shown to exhibit excellent weak and strong scaling.

This hierarchy of more and more accurate classical polarizable models up through the parent AMOEBA potential offers a way to address several interesting questions. Although the MBE has been known since the 1970's [41,42], it remains a current topic of interest for those developing QM-based fragment approaches and embedding schemes.[43-50] However the QM community almost exclusively looks at energy convergence, virtually ignoring the poor convergence of the MBE we have observed for a classical potential for gradients. Thus the parallelization strategy that has improved timings by up to an order of magnitude for a simple classical model may aid in the analysis of how different embedding schemes and different definition of the size of bodies converge better in the MBE formalism. In addition, we plan to explore how different water properties in the condensed phase converge under the 3m-AMOEBA and 3M-AMOEBA in future work.

ACKNOWLEDGEMENTS. We thank the National Science Foundation grant CHE-1265731 for the software developed in TINKER and CHE-1363320 for the MBE work. We thank the Edinburgh team: Weronika Filinger, Mario Antonioletti, Lorna Smith, Arno Proeme, and Neil Chue Hong, for their help during the last 3 years in regards parallelization.

REFERENCES

[1] R.C. Remsing, M.D. Baer, G.K. Schenter, C.J. Mundy, J.D. Weeks, Journal of Physical Chemistry Letters 5 (2014) 2767.
[2] P. Jungwirth, B. Winter, Annual Review of Physical Chemistry, 2008, p. 343.
[3] M. Vazdar, E. Pluharova, P.E. Mason, R. Vacha, P. Jungwirth, Journal of Physical Chemistry Letters 3 (2012) 2087.
[4] P. Nerenberg, B. Jo, C. So, A. Tripathy, T. Head-Gordon, Journal of physical Chemistry B 116 (2012) 4524.
[5] P.S. Nerenberg, T. Head-Gordon, J Chem Theory Comput 7 (2011) 1220.
[6] O. Demerdash, E.H. Yap, T. Head-Gordon, Annu Rev Phys Chem 65 (2014) 149.
[7] A. Warshel, S. Kuwajima, Journal of Physical Chemistry 94 (1990) 460.
[8] S.W. Rick, S.J. Stuart, J.S. Bader, B.J. Berne, Journal of Molecular Liquids 65-6 (1995) 31.
[9] P.N. Day, J.H. Jensen, M.S. Gordon, S.P. Webb, W.J. Stevens, M. Krauss, D. Garmer, H. Basch, D. Cohen, J Chem Phys 105 (1996) 1968.
[10] J.L. Gao, Journal of Physical Chemistry B 101 (1997) 657.
[11] P.Y. Ren, J.W. Ponder, Journal of Computational Chemistry 23 (2002) 1497.
[12] G. Lamoureux, A.D. MacKerell, B. Roux, Journal of Chemical Physics 119 (2003) 5185.

[13] J.P. Piquemal, B. Williams-Hubbard, N. Fey, R.J. Deeth, N. Gresh, C. Giessner-Prettre, J Comput Chem 24 (2003) 1963.
[14] G.A. Kaminski, H.A. Stern, B.J. Berne, R.A. Friesner, J Phys Chem A 108 (2004) 621.
[15] S. Patel, C.L. Brooks, Journal of Computational Chemistry 25 (2004) 1.
[16] A.G. Donchev, V.D. Ozrin, M.V. Subbotin, O.V. Tarasov, V.I. Tarasov, P Natl Acad Sci USA 102 (2005) 7829.
[17] E. Harder, B.C. Kim, R.A. Friesner, B.J. Berne, J Chem Theory Comput 1 (2005) 169.
[18] Z.X. Wang, W. Zhang, C. Wu, H.X. Lei, P. Cieplak, Y. Duan, J Comput Chem 27 (2006) 781.
[19] G.S. Fanourgakis, S.S. Xantheas, Journal of Chemical Physics 124 (2006) 174504.
[20] K.M. Benjamin, A.J. Schultz, D.A. Kofke, Journal of Physical Chemistry C 111 (2007) 16021.
[21] T.D. Rasmussen, P.Y. Ren, J.W. Ponder, F. Jensen, Int J Quantum Chem 107 (2007) 1390.
[22] N. Gresh, G.A. Cisneros, T.A. Darden, J.P. Piquemal, J Chem Theory Comput 3 (2007) 1960.
[23] D.P. Geerke, W.F. van Gunsteren, J Phys Chem B 11 (2007) 6425.
[24] W.L. Jorgensen, K.P. Jensen, A.N. Alexandrova, J Chem Theory Comput 3 (2007) 1987.
[25] M. Salanne, R. Vuilleumier, P.A. Madden, C. Simon, P. Turq, B. Guillot, J Phys-Condens Mat 20 (2008) 494207.
[26] P. Cieplak, F.Y. Dupradeau, Y. Duan, J.M. Wang, J Phys-Condens Mat 21 (2009).
[27] M. Tafipolsky, B. Engels, J Chem Theory Comput 7 (2011) 1791.
[28] L.-P. Wang, J. Chen, T. van Voorhis, J Chem Theory Comput 9 (2013) 452.
[29] P.E.M. Lopes, J. Huang, J. Shim, Y. Luo, H. Li, B. Roux, A.D. MacKerell, J. Chem. Theoy Comput. 9 (2013) 5430.
[30] Y. Shi, P. Ren, M. Schnieders, J.P. Piquemal, Rev. Comp. Chem. 28 (2015) 51.
[31] D.J. Arismendi-Arrieta, M. Riera, P. Bajaj, R. Prosmiti, F. Paesani, J. Phys. Chem. B 120 (2016) 1822.
[32] M. Kohagen, P.E. Mason, P. Jungwirth, J Phys Chem B 120 (2016) 1454.
[33] O.N. Demerdash, T. Head-Gordon, J. Chem. Theory Comput. revisions (2016).
[34] L.P. Wang, T. Head-Gordon, J.W. Ponder, P. Ren, J.D. Chodera, P.K. Eastman, T.J. Martinez, V.S. Pande, J Phys Chem B 117 (2013) 9956.
[35] A. Albaugh, H.A. Boateng, R.T. Bradshaw, O. Demerdash, J. Dziedzic, Y. Mao, D.T. Margul, J. Swails, Q. Zeng, D.A. Case, P. Eastman, J.W. Essex, M. Head-Gordon, V.S. Pande, J.W. Ponder, Y. Shao, C.-K. Skylaris, I.T. Todorov, M.E. Tuckerman, T. Head-Gordon, J Phys Chem B (Feature article) submitted (2016).
[36] P.Y. Ren, C.J. Wu, J.W. Ponder, J Chem Theory Comput 7 (2011) 3143.
[37] D.A. Case, T.E. Cheatham III, T. Darden, H. Gohlke, R. Luo, K.M. Merz Jr., A. Onufriev, C. Simmerling, B. Wang, R.J. Woods, Journal of Computational Chemistry 26 (2005) 1668.
[38] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kale, K. Schulten, Journal of Computational Chemistry 26 (2005) 1781.
[39] P. Eastman, M.S. Friedrichs, J.D. Chodera, R.J. Radmer, C.M. Bruns, J.P. Ku, K.A. Beauchamp, T.J. Lane, L.-P. Wang, D. Shukla, T. Tye, M. Houston, T. Stich, C. Klein, M.R. Shirts, V.S. Pande, J. Chem. Theory Comput. 9 (2013) 461.
[40] P.J. Steinbach, B.R. Brooks, Journal of Computational Chemistry 15 (1994) 667.
[41] D. Hankins, J.W. Moskowitz, J Chem Phys 53 (1970) 4544.
[42] H. Stoll, H. Preuss, Theoretica Chimica Acta 46 (1977) 11.
[43] T.A. Barnes, J.D. Goodpaster, F.R. Manby, T.F. Miller, J Chem Phys 139 (2013).
[44] P.J. Bygrave, N.L. Allan, F.R. Manby, J Chem Phys 137 (2012).
[45] E.E. Dahlke, D.G. Truhlar, J Chem Theory Comput 3 (2007) 46.
[46] M.J. Gillan, D. Alfe, P.J. Bygrave, C.R. Taylor, F.R. Manby, J Chem Phys 139 (2013).
[47] K.U. Lao, K.-Y. Liu, R.M. Richard, J.M. Herbert, J. Chem. Phys. 144 (2016) 164105.
[48] R.M. Richard, K.U. Lao, J.M. Herbert, J. Chem. Phys. 141 (2014) 014108.
[49] S.H. Wen, G.J.O. Beran, J Chem Theory Comput 7 (2011) 3733.

[50] H.R. Leverentz, K.A. Maerzke, S.J. Keasler, J.I. Siepmann, D.G. Truhlar, Physical Chemistry Chemical Physics 14 (2012) 7669.

TABLES

Table 1. *Hypothetical timings for 3M-AMOEBA for N=7000 water molecule system.* The standard timing for 1000 steps of MD for this system is 478.2 s using the canonical OpenMP TINKER7 code using 12 threads. The hypothetical timings are based on Eq. (5) and optimal OpenMP timings taken from TINKER7 for each n: $t_{175,AMOEBA}$=18 s, $t_{350,AMOEBA}$=36 s, $t_{700,AMOEBA}$=60 s, $t_{1400,AMOEBA}$=120 s, $t_{2100,AMOEBA}$=180 s. Number of cores assumes that each MPI task completes one individual fragment calculation (monomer, dimer, trimer), of which there are $M+M^2+M^3$ (no M cluster cutoffs) or $M+M^2+\sim M$ (with M cluster cutoffs) such tasks. All timings are based on a Cray XC30 using 12-core Intel "Ivy Bridge" processor at 2.4 GHz, 24 cores per node.

Fragmentation of N=7000 molecule system	**Hypothetical timings (s) and core count No M cluster cutoffs**		**Hypothetical timings (s) and core count With M cluster cutoffs**	
	Number of cores	**Timings (speed ups)**	**Number of cores**	**Timings (speed ups)**
M=10, n=700	13320	174.4 (2.8)	1440	128.3 (3.7)
M=20, n=350	101040	126.5 (3.8)	5280	62.1 (7.7)
M=30, n=175	335160	59.6 (8.0)	11520	44.8 (10.7)

Table 2. *Timings on water boxes ranging from 4,800-864,000 atoms for the mutual polarization models 3m-AMOEBA and AMOEBA.* Timings (speed ups) are reported in nanoseconds/day based on running the same fixed number of molecular dynamics steps, and using up to 6144 cores. The number of cores to achieve Break even, 2X and 4-5X faster are reported as well. All timings are based on a Cray XC30 using 12-core Intel "Ivy Bridge" processor at 2.4 GHz, 24 cores per node. Timings for AMOEBA run in Tinker 7 were obtained from runs on 12 threads (1 thread/core) using the standard OpenMP parallelized implementation. Timings for AMOEBA run in Amber were obtained with the MPI-parallelized, distributed-memory code pmemd.amoeba. We systematically varied the number of cores in the pmemd.amoeba runs in a range of 12 to 384, and we report in parentheses next to the speedups below the number of cores that gave the best timing and the associated speedup.

	Timings in ns/day (speedups achieved)		
System	**AMOEBA Tinker7**	**3m-AMOEBA Tinker7**	**AMOEBA Amber**
1600	0.818	1.158 (1.4)	1.899 (2.3; 48 cores)
7000	0.180	0.770 (4.3)	0.395 (2.2; 48 cores)
32000	0.036	0.202 (5.6)	0.082 (2.3; 96 cores)
96000	0.010	0.063 (6.3)	0.025 (2.5; 96 cores)
288000	0.0026	0.018 (6.9)	0.008 (3.1; 192 cores)
	Number of cores to achieve speed-ups with 3m-AMOEBA		
System	**Break Even**	**2X faster**	**4-5X faster**
7000	768	1536	6144
32000	768	1536	6144
96000	768	1536	3072

Table 3. *Actual timings for 3M-AMOEBA using an MPI wrapper around the standard OpenMP parallel implementation in TINKER7.* All timings are based on a fixed number of MD steps using 3600 cores. M is the number of k-means clusters. Further details are reported in Table 1.

Water system (N)	**Timings in ns/day (speedups achieved)**			
	AMOEBA (OpenMP)	**3M-AMOEBA (MPI/OpenMP)**		
		M=10	**M=20**	**M=30**
1600	0.8147	2.7698 (3.4)	4.0199 (4.9)	3.1354 (3.8)
7000	0.1806	0.5179 (2.9)	0.7046 (3.9)	1.2847 (7.1)
32000	0.0362	0.1151 (3.2)	0.2196 (6.1)	0.2726 (7.5)
96000	0.0099	0.0249 (2.5)	0.0581 (5.9)	0.0909 (9.2)
288000	0.0026	0.0047 (1.8)	0.0203 (7.7)	0.0287 (10.8)

FIGURE CAPTIONS

Figure 1. *Pseudocode for the calculation of a portion of the 2- and 3-body polarization energy, gradient, and virial on a single MPI task for 3m-AMOEBA.* Note the absence of 1-body terms owing to the fact that they are zero in the absence of Ewald in the AMOEBA force field.

Figure 2. *Pseudocode for the load-balancing strategy across MPI nodes for the 3m-AMOEBA polarization calculation.*

Figure 3. *Pseudocode for a single MD timestep in the 3-body approximation, 3m-AMOEBA.*

Figure 4. *Weak scaling results for the (a) 3m-AMOEBA and (b) 3M-AMOEBA models.* All timings based on a Cray XC30 using 12-core Intel "Ivy Bridge" processor at 2.4 GHz, 24 cores per node.

Figure 5. *Strong scaling results for the (a) 3m-AMOEBA and (b) 3M-AMOEBA models.* All timings based on a Cray XC30 using 12-core Intel "Ivy Bridge" processor at 2.4 GHz, 24 cores per node.

```
C OpenMP directives defining shared and private variables, and variables that are accumulated.

!$OMP PARALLEL default(private) shared(
!$OMP&  eng_temp, d_eng_temp, vir_temp,
!$OMP&  start,last, coordinates,cutoff_distances,
!$OMP&  smoothening_distances, task_id,
!$OMP&  num_neighbors, neighbors)
!$OMP DO reduction

C  Loop over portion of fragments (moli1) given to each task as determined by load  balancing.

          do mol1=start(task_id),last(task_id)

C  Loop over neighbors of molecule with index “moli1”.

          do j=1,num_neighbors(moli1)
                moli2=neighbors(j,moli1)
                if  (r < 2-body_cutoff) then
                   call polarization(moli1,moli2)
                   if (r > 2-body smoothening dist.)
                        <calc. smoothening function terms and derivatives thereof>
                   end if
                   <Add 2-body energy, gradient virial to eng_temp, d_eng_temp, and vir_temp,
                     with smoothening, if applicable>
                   <Save 2-body energy, gradient, virial without smoothening, since it will be used
                     later in the calculation of 3-body contributions to eng_temp, d_eng_temp, and
                     vir_temp>
                end if

C  Loop over neighbors of molecule with index “moli2”.

                do k=1,num_neigbors(moli2)
                     moli3=neighbors(k,mol2)
                     if  (r < 3-body cutoff) then
                        call polarization(moli1,moli2,moli3)
                        <Save trimer energy, gradient, virial for moli1, moli2, moli3 >

C The unaccounted for 2-body terms beyond above must be calculated and then subtracted to yield
C the 3-body energy, gradient, and virial under the many-body expansion.

                        call polarization(moli1,moli3)
                        call polarization(moli2,moli3)
                        <Subtract 2-body terms corresponding to moli1+moli2, moli1+moli3,
                          moli2+moli3 from trimer energy,gradient, virial to yield 3-body term>
                        if  (r > 3-body smoothening dist.)
                           <calc. smoothening function terms and derivatives thereof>
                        end if
                        <Apply smoothening if necessary.>
                        <Add 3-body energy, gradient, virial to eng_temp, d_eng_temp, and vir_temp,
                          with smoothening, if necessary>
                     end if
                 end do
             end do
         end do
!$OMP END DO
!$OMP END PARALLEL
```

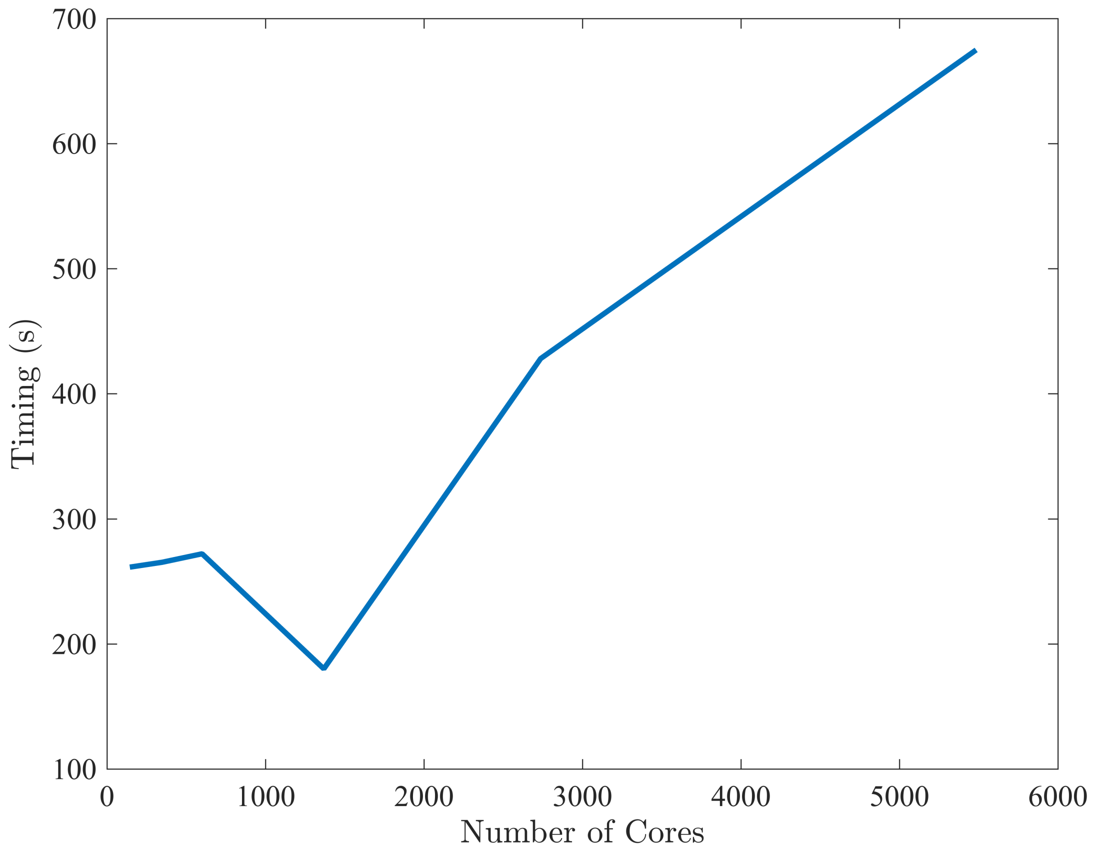
Figure 1. Demerdash and co-workers

```
C   Determine the total load and then what the ideal maximal load per task should be.

          total_load = 0
          do i=1,total_number_of_molecules
              do j=1,num_neigbors(i)
                  k=neighbor(j,i)
                  total_load = total_load + num_neighbors(k)
              end do
          end do

          max_load=total_load / number_of_MPI_tasks

          if (mod(tot_load, number_of_MPI_tasks).gt. 0 ) then
               max_load = max_load +1
          end if

C   Determine the limits of the outermost loop of the polarization calculation for each MPI task
C   based on the max_load calculated in the step above.

          molecule_counter = 1
          MPI_task_counter  = 0

          do while (molecule_counter .lt. total_number_of_molecules)
              load=0
              do while (load .lt. max_load)

                  if  (load .eq. 0) then
                      start(MPI_task_counter) = molecule_counter
                  end if

                   do j=1,num_neighbors(molecule_counter)
                        k=neighbor(j,molecule_counter)
                        load=load+num_neighbors(k)
                   end do
                   molecule_counter=molecule_counter+1

                  if  (molecule_counter .eq. total_number_of_molecules+1) then
                       goto 1
                  end if
              end do
1             continue
              last(MPI_task_counter)=molecule_counter-1
              MPI_task_counter = MPI_task_counter + 1
          end do
```

Figure 2. Demerdash and co-workers

```
if  (tasked.eq.master):
  call 1/2-step velocity Verlet
end if

<zero out eng/grad/virial and allocate gradient vectors if not already done>
<mpi_bcast(x,y,z)>
<rotate poles from local to new global xyz frame>
<update relavent portion of neighbor lists (based on prior call to 'neighbor list load balance')>
<call pairwiseadditive_grad>

  subroutine pairwiseadditive _grad

     if  (tasked.eq.master) then
          call covalent
     else if (tasked.eq.1) then
          call permanent electrostatic Particle Mesh Ewald
    else if (tasked.gt.1.and.taskid.lt.numtasks_emreal+2) then
          call load-balanced real-space permanent multipole electrostatics
    else if (tasked.gt.numtasks_emreal+2.and.taskid.lt.numtasks_emreal+2+numtasks_vdw) then
            call load-balanced vdW
        end if
        mpi_ireduce(vdw, Perm Elec)

  return
  end

<call 3-body mutual_polarization>

        mpi_ireduce(Polarisation)

<call mip_isend/irecv(PME perm. and polz)>
<call mpi_wait>

if  (taskid.eq.master) then
    update energy, viral, and gradient
    perform full-step velocity Verlet
end if
```

Figure 3. Demerdash and co-workers



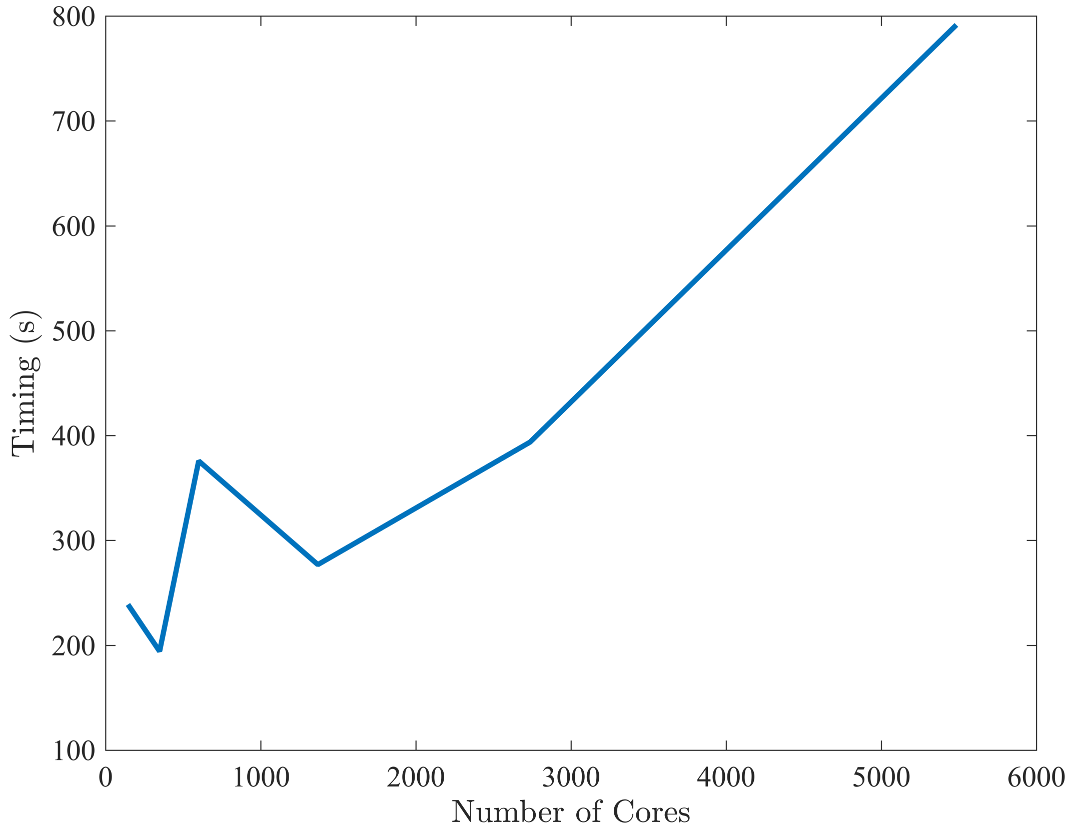
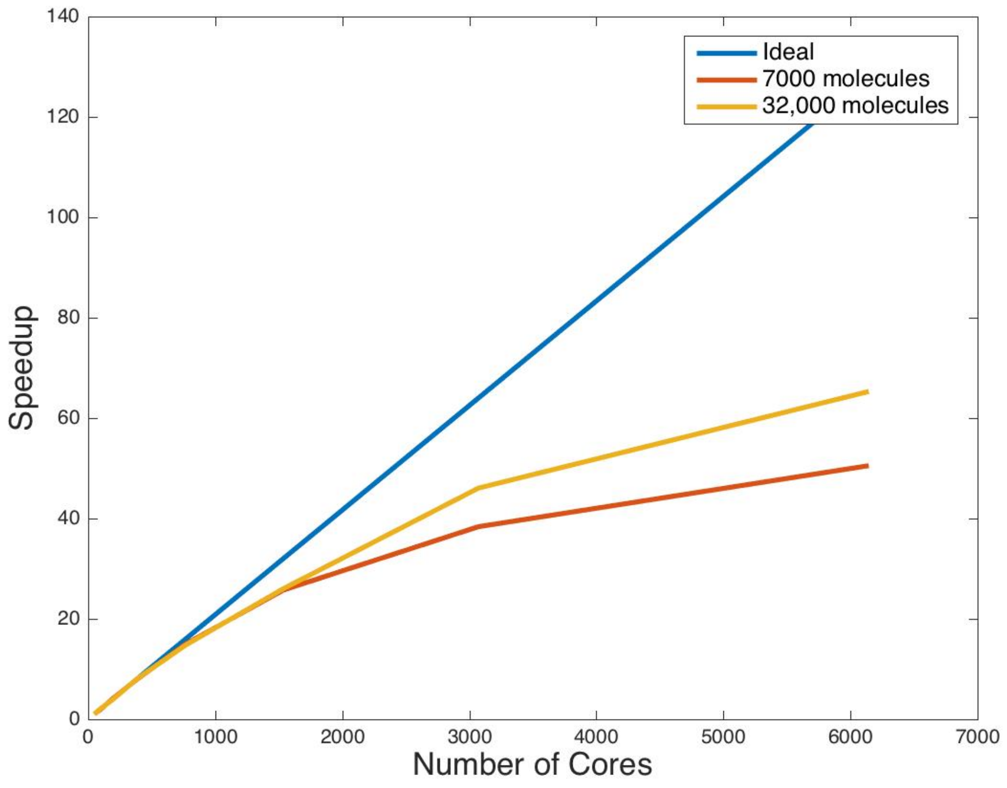


Figure 4. Demerdash and co-workers



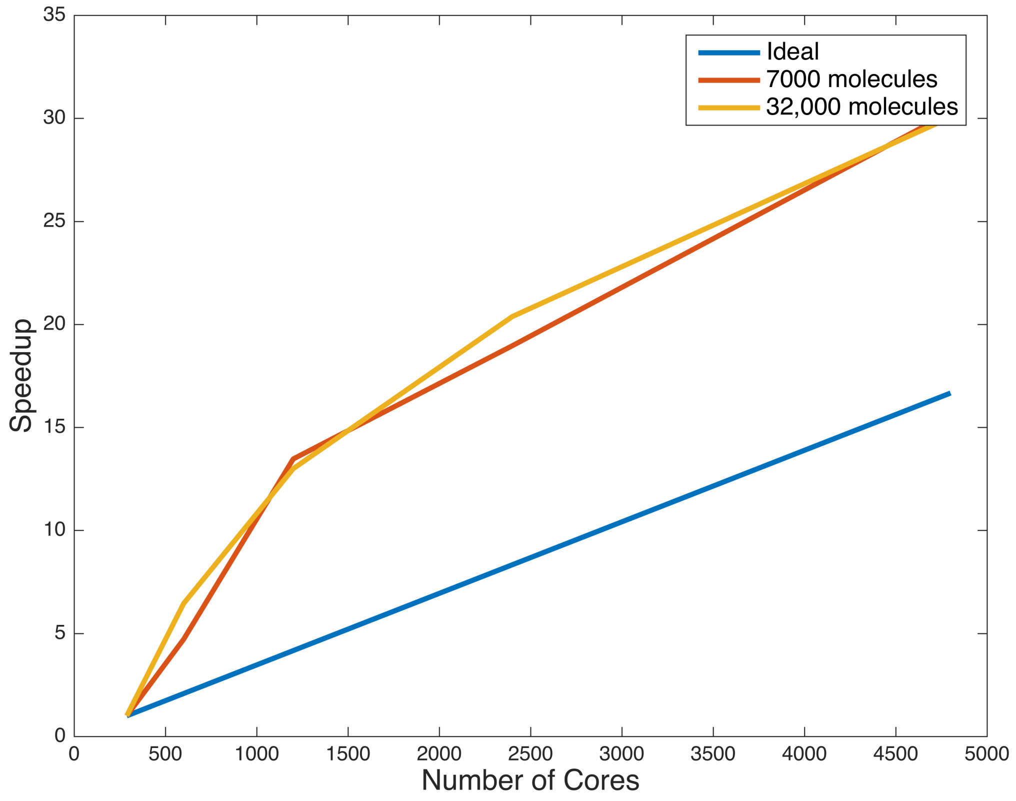


Figure 5. Demerdash and co-workers