SUPPORTING INFORMATION:

Using the k-d Tree Data Structure to Accelerate Monte Carlo Simulations

Qile P. Chen,[†,‡] Bai Xue,[‡] and J. Ilja Siepmann[*,†,‡]

Department of Chemical Engineering and Materials Science, University of Minnesota, 421 Washington Avenue SE, Minneapolis, Minnesota 55455-0132, United States, and Department of Chemistry and Chemical Theory Center, University of Minnesota, 207 Pleasant Street SE, Minneapolis, Minnesota 55455-0431, United States

E-mail: siepmann@umn.edu

S.1. CONTENTS

This Supporting Information provides a description of the implementation details for the *k*-d tree data structure, a discussion about the effects of changing the cutoff distance on the CPU timings, and numerical data for all figures presented in the main manuscript.

[*]To whom correspondence should be addressed
[†]CEMS
[‡]CHEM

S.2. IMPLEMENTATION DETAILS FOR THE k-D TREE DATA STRUCTURE

S.2.1. k-d Tree Construction. The k-d tree data structure needs to be initialized at the start of the simulation when an initial configuration is either read from a file or generated by the software. In addition, the k-d tree needs to be reconstructed periodically to ensure the balance of the tree and, hence, efficient insertions, deletions, and range searches (see discussion in Section S.2.4). During the tree initialization, the memory is allocated for every node, and then parent-child points for each node are assigned. For the tree reconstruction, the memory does not need to be reallocated because the "in-place" sorting algorithm is used (see below). The k-d tree can be perfectly balanced at time of the initialization or reconstruction by always first inserting the node whose coordinate in the cutting dimension is the median among all the other nodes. This balancing procedure can be achieved using the following steps:

1. Split all the nodes that have not yet been inserted into two halves based on the cutting dimension coordinate.

2. Place the median node into the insertion queue.

3. Recursively carry out step (1) for the smaller and the greater half.

4. Insert all the nodes in the insertion queue to the tree.

The split is performed via the "in-place" `quickselect` algorithm and it scales as $\mathcal{O}(N)$,[1] resulting in an overall scaling of $\mathcal{O}(N\log_2 N)$ for the tree construction. The treatment of the periodic boundary condition (see below) might introduce beads that are periodic images and, hence, there can be a tie when it comes to sorting or comparison in the k-d tree. In this case, the coordinate of the next cutting dimension can serve as a tie breaker (e.g. comparing y coordinates if x coordinates are equal) to ensure the balance of the tree.

The treatment of the periodic boundary condition can make a significant impact on the performance. To reduce the number of nodes stored in the k-d tree, periodic images are stored in the k-d

tree only if their coordinates in the i^{th} dimension, r_i ($i = x, y, z$), satisfies the following condition:

$$-r_{\text{cut}} - 2\Delta r_{\text{intra}} \leq r_i \leq r_{\text{cut}} + 2\Delta r_{\text{intra}} \tag{1}$$

This approach is used when molecules are "wrapped" into the central box on a COM basis, i.e., the coordinate of a particular bead may exceed the box length as long as the COM of the molecule does not. For other "wrapping" options, similar approaches can be adopted accordingly. The current approach ensures that only periodic images that may interact with nodes in the central box are stored in the k-d tree. Because of the storage of coordinates for additional periodic images, the faster range search of this extended k-d tree comes with the price of higher memory usage. Nevertheless, this does not affect the performance of the MCCCS–MN program for the system sizes considered here.

S.2.2. Node Insertion and Update. A new node is inserted into the tree by first traversing the tree through a series of comparisons and then being added as a new leaf in the appropriate position. The coordinate update during the course of a simulation is achieved by first deleting the old node and then inserting the new one. The deletion involves finding the node that has the minimum coordinate in the same cutting dimension in the right sub-tree (or equivalently, the maximum node in the left sub-tree), exchanging the to-be-deleted node with this minimum node (or the maximum node), and then iterating the previous two steps recursively until the to-be-deleted node has no child node and can now be removed. The node update can result in an imbalance of the tree. For local Monte Carlo (MC) moves, such as a translation or a rotation of the molecule, local updates may be designed to maintain the balance of the tree. However, it is difficult to generalize the design of this type of local operation to be applicable for non-local MC moves, such as the particle transfer move between simulation boxes in the Gibbs ensemble or the aggregation-volume bias MC move that utilizes large displacements to efficiently sample aggregation within a simulation box.[2] Therefore, a global tree reconstruction is performed here to maintain the balance of the tree after multiple node updates (as described in Section S.2.4).

S.2.3. Range Search. The range search for particles within r_{cut} from any arbitrary position in the simulation box (or reference position) can be performed via the following steps:

1. Examine the current node, and include it in the "found neighbor" list if it is within r_{cut} from the reference position.

2. Compute the distance from the reference position to the left and right sub-trees of the current node. The distance from a node to a sub-tree is defined as the shortest distance between the reference coordinate and the bounding cube of this sub-tree, and it is zero when the reference coordinate is within the bounding cube. This is the closest possible distance between the reference position and every particle in the sub-tree.

3. Recursively examine the sub-tree that is "closer" in distance from the reference position starting from step (1).

4. After all of the "closer" trees are examined, recursively examine the "farther" sub-tree starting from step (1) when the distance from the reference coordinate to the sub-tree is smaller or equal to r_{cut}.

S.2.4. Scaling for a Balanced Tree and Tree Reconstruction. When the k-d tree is perfectly balanced, then the height of the tree satisfies $H_{\text{min}} = \lceil \log_2 N_{\text{tree}} \rceil$, where H_{min} is the smallest integer number larger than $\log_2 N_{\text{tree}}$ and where N_{tree} is the number of interaction sites including the ones in the periodic images that might interact with the interaction sites in the original simulation box. In this case, insertions and deletions of a single bead scale as $\mathscr{O}(\log_2 N_{\text{tree}})$. Thus, the k-d tree insertions and deletions are more expensive than the coordinate update in the standard array implementation ($\mathscr{O}(1)$). However, the range search using the k-d tree scales asymptotically as $\mathscr{O}(\log_2 N_{\text{tree}})$, that is significantly improved over the $\mathscr{O}(N)$ scaling for the array implementation.

To maintain a well-balanced tree, the k-d tree needs to be reconstructed during the simulation. In this work, the point of recronstruction depends on the ensemble. Since an accepted volume move results in a change of all intermolecular distances, the tree is reconstructed at this point for

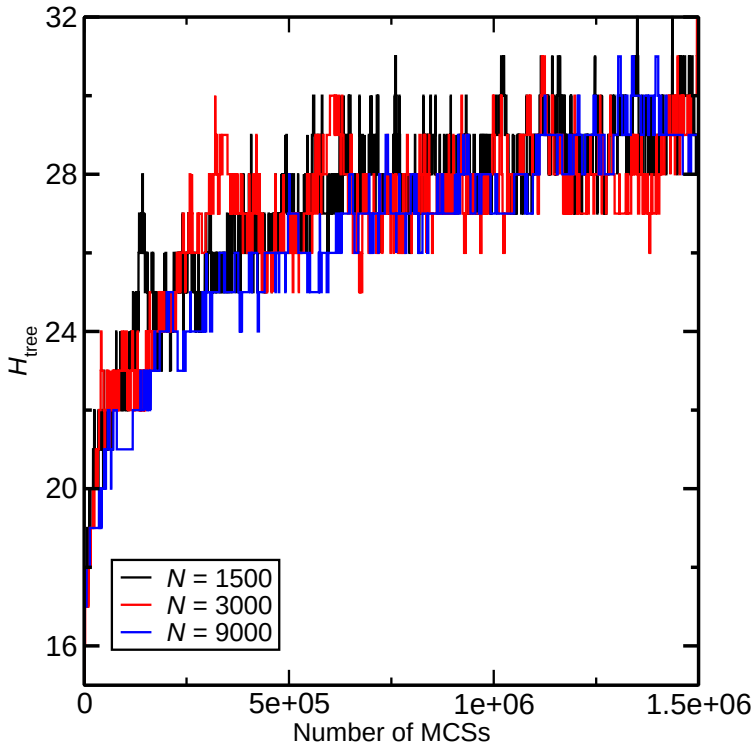

Figure S1: The height of the tree as a function of the number of MC steps for a simulation without tree reconstruction: Lennard-Jonesium in the NVT ensemble ($T^* = 0.72$, $\rho^* = 0.75$, and $r^*_{\rm cut} = 4$).

all simulations carried out in the isobaric-isothermal and Gibbs ensembles. For simulations using a fixed volume of the simulation box, the tree is reconstructed when the tree height exceeds a user specified threshold value.

Figure S1 indicates the evolution of the tree height as a function of the number of MC steps for simulations of Lennard-Jones particles in the NVT ensemble with three different system sizes. The tree height is found to grow initially exponentially and to reach a plateau at around twice the balanced tree height $H_{\rm min}$ (with $H_{\rm min} = 15$, 15, and 16 for $N = 1500$, 3000, and 9000, respectively). The effect of an imbalanced tree on the performance can be assessed through the dependence of the CPU time on the reconstruction threshold used for the simulations. Here, the k-d tree is reconstructed if $H/H_{\rm min}$, the ratio between the tree height and the balanced tree height, exceeds a user specified threshold value. As can be seen in Figure 2, the efficiency is not strongly sensitive to the threshold value. The timing data indicate that the CPU time initially decreases as the recon-

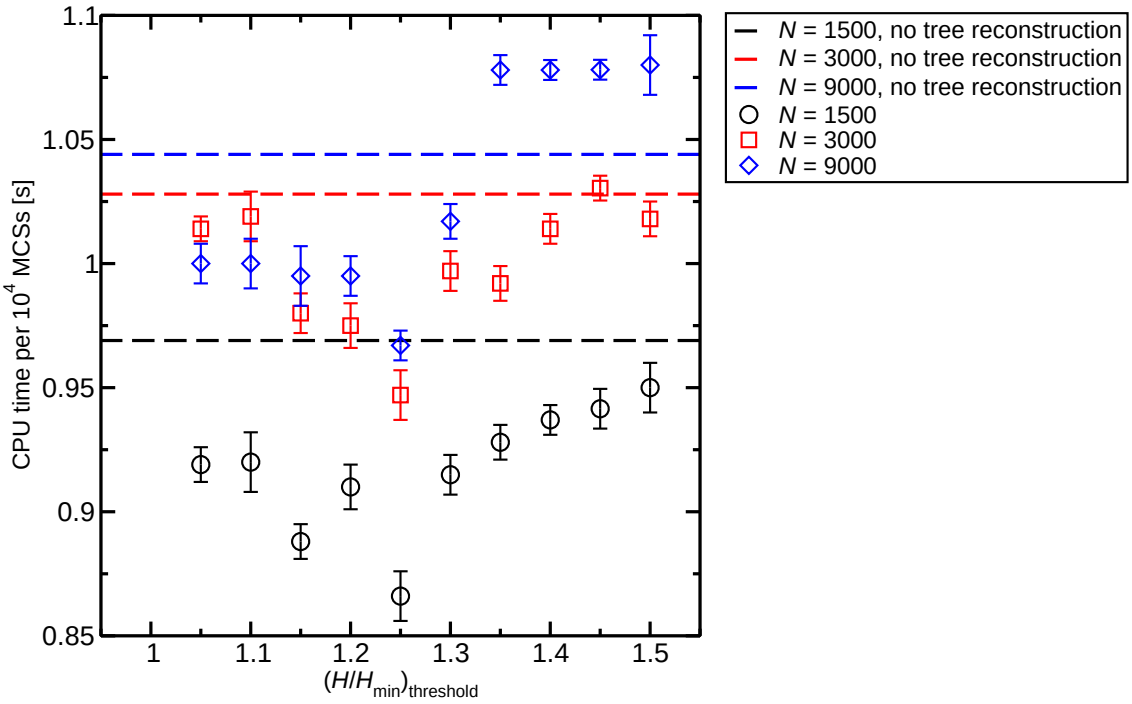

Figure S2: CPU timings for simulations of Lennard-Jonesium in the NVT ensemble ($T^* = 0.72$, $\rho^* = 0.75$, and $r^*_{\text{cut}} = 4$) as a function of the tree reconstruction threshold, $(H/H_{\text{min}})_{\text{threshold}}$. The tree is reconstructed if the reduced height of the tree, H/H_{min}, exceeds this threshold value.

struction threshold is increased (i.e., the reconstruction frequency is decreased) and subsequently the CPU time increases when the threshold value becomes too large. This behavior is caused by two competing factors: the additional CPU time spent for reconstructing the tree and the CPU time saved when performing tree operations on a better balanced tree. The optimal performance is found here for a threshold value of 1.25. For this threshold value, the CPU time is about 10% smaller compared to that of a simulation without tree reconstruction for the current length of simulations, but the difference may become slightly larger for longer simulations. Therefore, for all of the simulations in the canonical ensemble, the tree reconstruction threshold is set at 1.25.

From Figure S1, it can also be deduced that the threshold value of 1.25 leads to a tree reconstruction frequency of every few MC cycles, with values of 3, 1, and 4 MC cycles for $N = 1500$, 3000, and 9000, respectively. Since the frequency of volume moves for simulations in the isobaric-isothermal and Gibbs ensembles is set to yield about one accepted volume move per MC cycle, there is no need to reconstruct the k-d tree for simulations in which volume moves are used.

S.3. EFFECTS OF CHANGING THE CUTOFF DISTANCE

Figures S3 and S4 show the CPU time as a function of the number of molecules, N, for simulations of n-butane and TIP4P water, respectively, in the NpT ensemble using smaller r_{cut} values of 10 and 12 Å. For n-butane, the simulation cost decreases significantly for the volume, translational, and rotational moves compared to simulations with $r_{\text{cut}} = 14$ Å (see Figure 4). In contrast, the CPU time for CBMC moves decreases only slightly because a smaller inner cut-off value of 5 Å is already used for the energy calculation of all the trial sites utilized for the Rosenbluth weight, and the choice of r_{cut} affects only the energy computation of the fully-grown n-butane molecule used for the acceptance of the CBMC move.[3] Despite these changes, the trends described in the Section 3.3 of the main text remain valid: the use of the k-d tree method still results in an efficiency increase for the total set of moves when $N \geq 3000$.

For TIP4P water, the CPU time of all the moves at $N = 2000$ and 4000 decreases on average by 40% for the k-d tree method and 20% for the COM cutoff method when a smaller r_{cut} at 10 Å is used instead of $r_{\text{cut}} = 14$ Å. As the system size increases, the expense of the reciprocal space calculation in Ewald summation increases due to the presence of more vectors in the reciprocal space (the number of vectors in one dimension is inverse proportional to r_{cut}). Therefore, when $N < 4000$, the timing differences between all the methods are similar to those for a larger r_{cut} value of 14 Å. When $N > 4000$, due to the increase of the Ewald summation (equivalent to an overhead), the relative differences between all the methods become smaller. For example, for the complete set of moves, the time for the k-d tree method is only 7% greater than that for the COM cutoff method for the system with 8000 water molecules and $r_{\text{cut}} = 10$ Å, whereas it is 50% greater for $r_{\text{cut}} = 14$ Å.

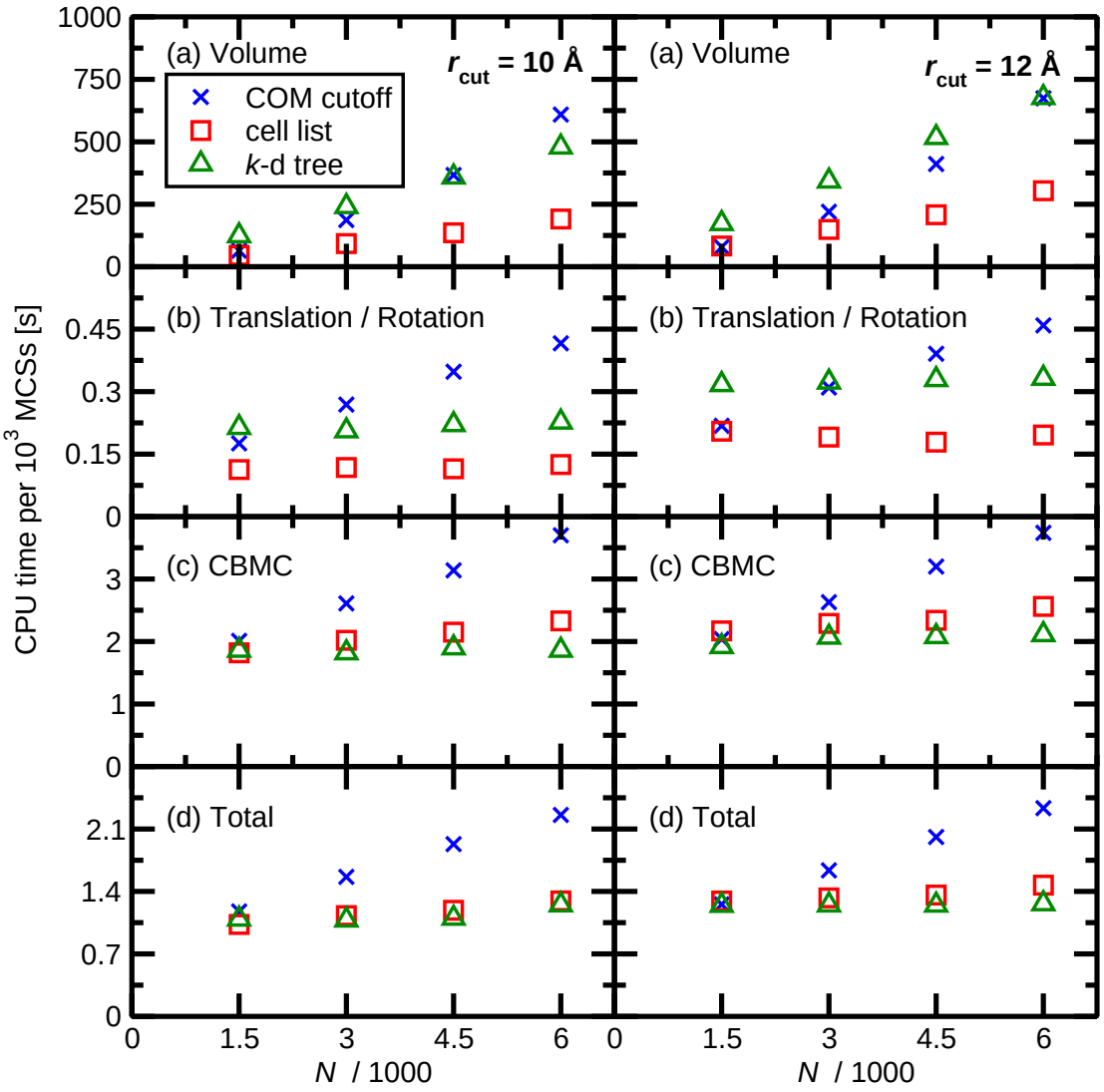

Figure S3: CPU timings for simulations of n-butane in the NpT ensemble ($T = 255$ K, $p = 1$ bar) using $r_{\text{cut}} = 10$ Å (left column) and 12 Å (right column) as functions of the number of molecules, N: (a) volume moves, (b) translational or rotational moves, (c) CBMC conformational moves, and (d) total set of moves.

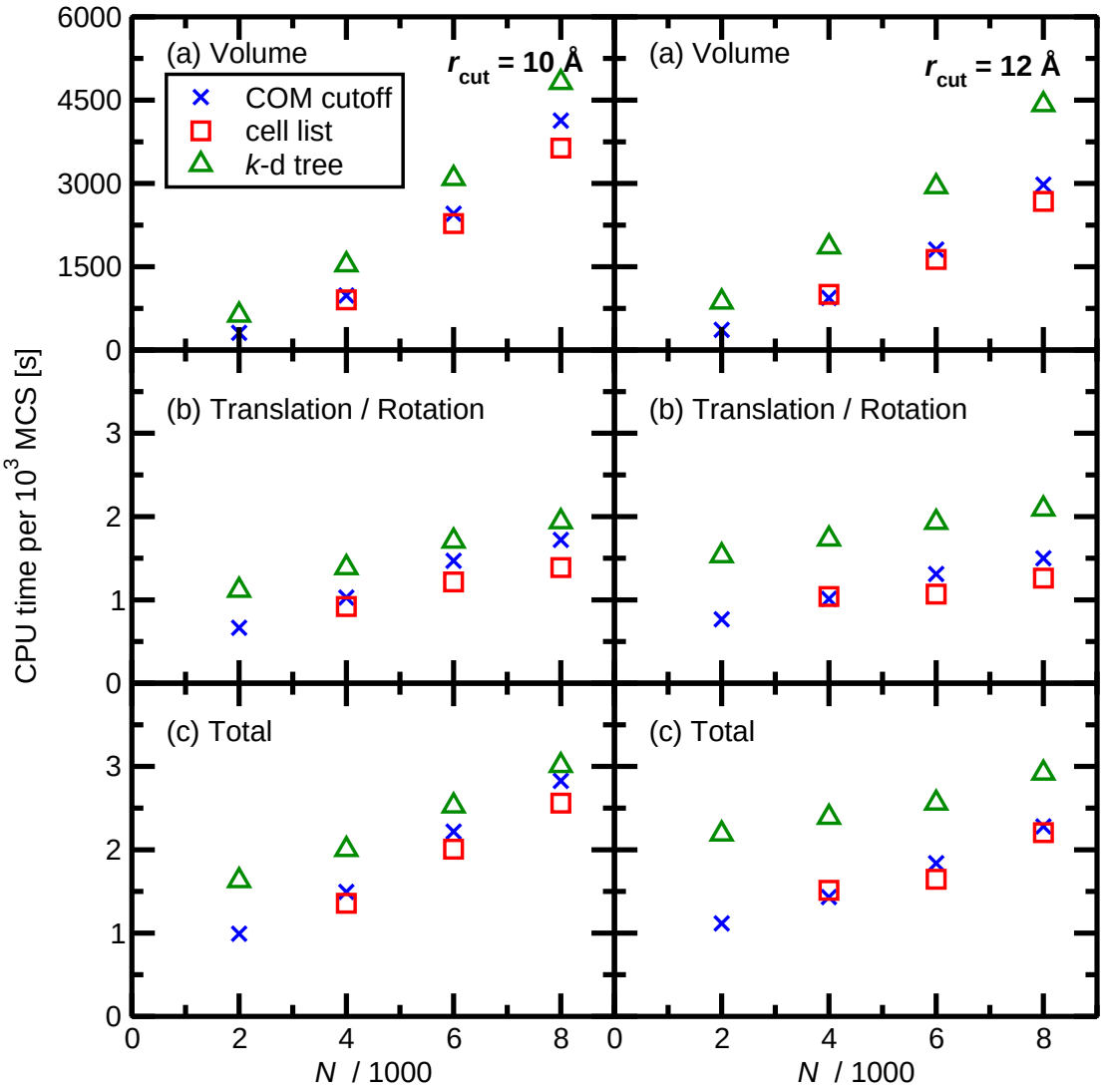

Figure S4: CPU timings for simulations of TIP4P water in the NpT ensemble ($T = 298$ K, $p =$ 1 bar) using $r_{\mathrm{cut}} = 10$ Å (left column) and 12 Å (right column) as functions of the number of molecules, N: (a) volume moves, (b) translational or rotational moves, (c) CBMC conformational moves, and (d) total set of moves.

S.4. NUMERICAL DATA

The following abbreviations are used in the data tables: MCS for Monte Carlo step, LOOP for the complete loop method, COM for the center-of-mass cutoff method, CELL for the linked-cell list method, and TREE for the k-d tree method.

The subscripts for the timing data denote the statistical uncertainty in the last digit given as the 95% confidence limit.

Table S1. CPU Timings for Simulations of Lennard-Jonesium in the NVT Ensemble ($k_{\mathrm{B}}T/\varepsilon = 0.72$, $\rho\sigma^3 = 0.75$, $r_{\mathrm{cut}}/\sigma = 4$).

N	$L_{\mathrm{box}}/r_{\mathrm{cut}}$	Time per 10^4 MCSs [s]		
		LOOP	CELL	TREE
400	2.1	0.475_1	N/A	0.872_4
1000	2.8	0.851_5	N/A	0.880_3
1500	3.2	1.162_3	1.042_5	0.866_4
3000	3.9	2.03_1	1.75_1	0.947_5
4500	4.6	2.66_1	1.250_6	0.916_2
6000	5.0	3.05_2	1.689_4	0.952_1
7500	5.4	3.68_3	1.251_8	0.978_3
9000	5.7	4.35_3	1.52_1	0.967_4
10500	6.0	4.99_2	1.594_7	0.953_5
12000	6.3	5.63_3	1.212_6	0.967_5
15000	6.8	6.92_5	1.410_8	1.020_6
20000	7.4	9.02_6	1.290_7	1.032_4
25000	8.0	11.25_8	1.042_9	1.035_5

Table S2. CPU Timings for Simulations of Lennard-Jonesium in the NVT-Gibbs Ensemble ($k_{\mathrm{B}}T/\varepsilon = 0.72$, $r_{\mathrm{cut}}/\sigma = 4$, and 20% of the Molecules in the Vapor Phase).

N	$L_{\mathrm{liq}}/r_{\mathrm{cut}}$	$L_{\mathrm{vap}}/r_{\mathrm{cut}}$	LOOP	TREE (both)	TREE (liq)	CELL (liq)
			Volume Moves [10^{-1} s]			
1500	2.8	12.6	0.32_1	0.56_1	N/A	N/A
3000	3.6	15.9	1.32_2	1.12_3	1.03_1	1.21_1
4500	4.1	18.2	2.31_1	1.67_4	1.60_2	1.21_3
6000	4.5	20.0	4.15_2	2.37_3	2.33_3	2.08_5
7500	4.8	21.6	6.20_4	2.92_2	3.14_2	3.26_4
9000	5.1	22.9	8.39_5	3.45_5	4.05_4	3.43_5
10500	5.4	24.1	11.0_3	4.03_5	4.75_2	4.29_6
12000	5.7	25.2	16.0_4	4.90_4	6.05_3	6.01_6
15000	6.1	27.2	N/A	6.30_7	8.20_4	8.30_1
			Translational Moves [10^{-4} s]			
1500	2.8	12.6	0.93_1	0.94_1	N/A	N/A
3000	3.6	15.9	1.67_1	1.07_1	0.995_5	1.45_1
4500	4.1	18.2	2.35_1	0.93_1	1.152_5	1.19_1
6000	4.5	20.0	3.13_2	0.94_1	1.23_1	1.50_1
7500	4.8	21.6	3.75_4	0.96_1	1.365_8	1.87_1
9000	5.1	22.9	4.48_3	1.00_1	1.43_2	1.62_1
10500	5.4	24.1	5.21_4	1.11_1	1.52_1	1.66_1
12000	5.7	25.2	5.87_1	1.02_1	1.60_2	2.07_2
15000	6.1	27.2	N/A	1.23_1	1.71_1	2.01_1
			Swap Moves [10^{-4} s]			
1500	2.8	12.6	9.93_2	1.40_1	N/A	N/A
3000	3.6	15.9	19.2_3	1.65_1	8.65_3	17.5_2
4500	4.1	18.2	29.0_3	1.67_1	7.74_3	12.6_4
6000	4.5	20.0	38.7_5	2.68_1	9.63_5	19.9_2
7500	4.8	21.6	48.5_6	2.63_2	12.2_1	26.3_3
9000	5.1	22.9	58.2_5	2.90_2	14.3_2	23.1_4
10500	5.4	24.1	67.3_3	3.12_2	16.5_1	27.2_4
12000	5.7	25.2	78.2_5	2.99_1	18.7_2	30.5_6
15000	6.1	27.2	N/A	3.05_2	24.5_3	31.7_5
			Total Move Set [10^{-4} s]			
1500	2.8	12.6	3.67_1	1.81_1	N/A	N/A
3000	3.6	15.9	4.25_3	1.81_1	2.12_1	3.87_1
4500	4.1	18.2	6.10_2	1.79_1	2.48_2	3.20_5
6000	4.5	20.0	6.92_3	1.81_1	2.35_3	3.70_6
7500	4.8	21.6	7.61_4	1.77_1	2.43_1	4.19_7
9000	5.1	22.9	8.90_3	1.79_1	2.56_2	3.51_7
10500	5.4	24.1	9.97_4	2.01_1	2.84_2	3.80_5
12000	5.7	25.2	11.0_1	2.03_1	3.10_3	4.26_1
15000	6.1	27.2	N/A	2.08_1	3.55_3	3.89_4

Table S3. CPU Timings for Simulations of TraPPE–UA *n*-Butane in the NpT Ensemble ($T = 255$ K, $p = 1$ bar, $r_{\text{cut}} = 14$ Å).

N	$L_{\text{box}}/L_{\text{cell}}$	LOOP	COM	CELL	TREE
		Volume Moves [10^{-1} s]			
1500	3.3	N/A	1.00_1	1.08_1	2.43_7
3000	4.2	N/A	2.62_1	1.94_4	4.82_{14}
4500	4.8	N/A	4.76_1	3.30_2	7.3_2
6000	5.3	N/A	7.58_4	3.88_1	9.4_3
		Trans/Rot Moves [10^{-4} s]			
1500	3.3	15.3_2	2.74_5	2.61_1	4.51_5
3000	4.2	30.0_1	3.66_3	2.49_1	4.70_7
4500	4.8	44.8_1	4.48_5	2.84_2	4.87_5
6000	5.3	59.1_3	5.15_2	2.55_3	5.16_7
		CBMC Moves [10^{-3} s]			
1500	3.3	N/A	2.11_1	2.21_3	2.11_3
3000	4.2	N/A	2.72_6	2.30_2	2.19_6
4500	4.8	N/A	3.26_7	2.68_3	2.21_{11}
6000	5.3	N/A	3.79_1	2.62_3	2.29_3
		Total Move Set [10^{-3} s]			
1500	3.3	N/A	1.33_3	1.374_{14}	1.44_4
3000	4.2	N/A	1.724_2	1.414_6	1.45_2
4500	4.8	N/A	2.06_2	1.63_3	1.44_2
6000	5.3	N/A	2.38_2	1.56_2	1.43_2

Table S4. CPU Timings for Simulations of TraPPE–UA *n*-Butane in the NpT Ensemble ($T = 255$ K, $p = 1$ bar, $r_{\text{cut}} = 12$ Å).

N	$L_{\text{box}}/L_{\text{cell}}$	LOOP	COM	CELL	TREE
			Volume Moves [10^{-1} s]		
1500	3.7	N/A	0.79_1	0.83_1	1.73_1
3000	4.7	N/A	2.20_1	1.49_2	3.43_3
4500	5.4	N/A	4.11_1	2.08_2	5.17_2
6000	5.9	N/A	6.74_9	3.04_1	6.74_4
			Trans/Rot Moves [10^{-4} s]		
1500	3.7	N/A	2.18_2	2.05_1	3.17_1
3000	4.7	N/A	3.10_1	1.91_2	3.23_3
4500	5.4	N/A	3.91_2	1.79_1	3.29_3
6000	5.9	N/A	4.59_2	1.96_2	3.32_3
			CBMC Moves [10^{-3} s]		
1500	3.7	N/A	2.04_1	2.17_1	1.92_1
3000	4.7	N/A	2.63_2	2.29_4	2.07_3
4500	5.4	N/A	3.20_3	2.34_2	2.08_4
6000	5.9	N/A	3.74_1	2.563_2	2.11_3
			Total Move Set [10^{-3} s]		
1500	3.7	N/A	1.255_{13}	1.293_{15}	1.245_7
3000	4.7	N/A	1.636_5	1.328_8	1.249_{11}
4500	5.4	N/A	2.01_3	1.36_3	1.247_5
6000	5.9	N/A	2.333_5	1.471_4	1.263_7

Table S5. CPU Timings for Simulations of TraPPE–UA *n*-Butane in the NpT Ensemble ($T = 255$ K, $p = 1$ bar, $r_{\text{cut}} = 10$ Å).

N	$L_{\text{box}}/L_{\text{cell}}$	LOOP	COM	CELL	TREE
		Volume Moves [10^{-1} s]			
1500	4.2	N/A	0.64_1	0.453_4	1.24_1
3000	5.3	N/A	1.87_1	0.925_1	2.41_1
4500	6.1	N/A	3.67_5	1.36_1	3.59_2
6000	6.7	N/A	6.09_8	1.91_1	4.79_2
		Trans/Rot Moves [10^{-4} s]			
1500	4.2	N/A	1.76_1	1.13_2	2.14_1
3000	5.3	N/A	2.69_6	1.18_1	2.06_1
4500	6.1	N/A	3.48_1	1.15_1	2.21_3
6000	6.7	N/A	4.16_1	1.25_1	2.27_3
		CBMC Moves [10^{-3} s]			
1500	4.2	N/A	2.01_2	1.82_1	1.86_7
3000	5.3	N/A	2.61_1	2.02_1	1.82_2
4500	6.1	N/A	3.14_2	2.15_4	1.90_8
6000	6.7	N/A	3.70_3	2.33_1	1.86_3
		Total Move Set [10^{-3} s]			
1500	4.2	N/A	1.176_4	1.030_1	1.09_1
3000	5.3	N/A	1.563_{11}	1.130_2	1.08_1
4500	6.1	N/A	1.93_2	1.189_2	1.10_1
6000	6.7	N/A	2.257_5	1.294_8	1.25_1

Table S6. CPU Timings for Simulations of TraPPE–UA Ethanol in the NpT Ensemble ($T = 323$ K, $p = 1$ bar).

N	L_{box}/L_{cell}	LOOP	COM	CELL	TREE
			Volume Moves [s]		
1500	2.9	N/A	0.233_1	N/A	0.449_5
3000	3.6	N/A	0.697_2	0.723_2	1.043_{14}
4500	4.2	N/A	1.572_7	1.468_7	2.027_{11}
6000	4.5	N/A	2.876_5	2.651_{14}	3.338_{14}
			Trans/Rot Moves [10^{-3} s]		
1500	2.9	1.753_3	0.670_7	N/A	1.03_1
3000	3.6	3.36_3	1.000_7	0.980_6	1.30_1
4500	4.2	4.77_2	1.257_1	1.099_5	1.48_2
6000	4.5	6.37_3	1.617_1	1.366_{11}	1.76_1
			CBMC Moves [10^{-3} s]		
1500	2.9	N/A	2.592_2	N/A	2.92_5
3000	3.6	N/A	3.39_2	3.542_7	3.11_3
4500	4.2	N/A	4.11_2	3.488_{13}	3.30_3
6000	4.5	N/A	4.89_3	4.1_2	3.30_3
			Total Move Set [10^{-3} s]		
1500	2.9	N/A	2.12_5	N/A	2.40_2
3000	3.6	N/A	2.919_7	3.00_2	2.77_4
4500	4.2	N/A	3.76_5	3.27_2	3.25_3
6000	4.5	N/A	4.445_3	3.78_2	3.51_3

Table S7. CPU Timings for Simulations of TIP4P Water in the NpT Ensemble ($T = 298$ K, $p = 1$ bar, $r_{\mathrm{rcut}} = 14$ Å).

N	$L_{\mathrm{box}}/L_{\mathrm{cell}}$	LOOP	COM	CELL	TREE
			Volume Moves [s]		
2000	2.5	N/A	0.475_{1}	N/A	1.208_{8}
4000	3.1	N/A	1.112_{4}	1.199_{3}	2.399_{10}
6000	3.6	N/A	1.864_{4}	2.020_{13}	3.55_{4}
8000	3.9	N/A	2.920_{11}	3.206_{3}	5.154_{13}
			Trans/Rot Moves [10^{-3} s]		
2000	2.5	1.923_{5}	0.990_{5}	N/A	2.12_{3}
4000	3.1	3.472_{4}	1.191_{6}	1.236_{4}	2.36_{5}
6000	3.6	4.97_{1}	1.377_{4}	1.34_{1}	2.29_{4}
8000	3.9	6.57_{2}	1.562_{2}	1.58_{1}	2.61_{5}
			Total Move Set [10^{-3} s]		
2000	2.5	N/A	1.485_{3}	N/A	3.10_{2}
4000	3.1	N/A	1.720_{7}	1.84_{3}	3.14_{1}
6000	3.6	N/A	1.875_{3}	1.998_{6}	3.04_{1}
8000	3.9	N/A	2.339_{3}	2.460_{3}	3.51_{4}

Table S8. CPU Timings for Simulations of TIP4P Water in the NpT Ensemble ($T = 298$ K, $p = 1$ bar, $r_{\mathrm{rcut}} = 12$ Å).

N	$L_{\mathrm{box}}/L_{\mathrm{cell}}$	LOOP	COM	CELL	TREE
			Volume Moves [s]		
2000	2.8	N/A	0.363_2	N/A	0.865_3
4000	3.5	N/A	0.937_6	1.002_2	1.858_{13}
6000	4.2	N/A	1.808_2	1.631_1	2.941_{11}
8000	4.4	N/A	2.978_6	2.677_1	4.419_{15}
			Trans/Rot Moves [10^{-3} s]		
2000	2.8	N/A	0.767_9	N/A	1.53_1
4000	3.5	N/A	1.012_3	1.041_4	1.73_5
6000	4.2	N/A	1.311_1	1.069_4	1.93_3
8000	4.4	N/A	1.50_2	1.261_5	2.09_6
			Total Move Set [10^{-3} s]		
2000	2.8	N/A	1.116_2	N/A	2.19_1
4000	3.5	N/A	1.432_3	1.514_{11}	2.39_1
6000	4.2	N/A	1.838_1	1.646_1	2.56_2
8000	4.4	N/A	2.28_2	2.205_1	2.92_3

Table S9. CPU Timings for Simulations of TIP4P Water in the NpT Ensemble ($T = 298$ K, $p = 1$ bar, $r_{\mathrm{rcut}} = 10$ Å).

N	$L_{\mathrm{box}}/L_{\mathrm{cell}}$	LOOP	COM	CELL	TREE
			Volume Moves [s]		
2000	3.3	N/A	0.309_{1}	N/A	0.629_{3}
4000	4.1	N/A	0.981_{1}	0.904_{3}	1.536_{3}
6000	4.9	N/A	2.454_{4}	2.277_{5}	3.089_{15}
8000	5.2	N/A	4.132_{5}	3.636_{3}	4.818_{1}
			Trans/Rot Moves [10^{-3} s]		
2000	3.3	N/A	0.665_{6}	N/A	1.115_{6}
4000	4.1	N/A	1.028_{3}	0.920_{4}	1.387_{15}
6000	4.9	N/A	1.469_{3}	1.215_{3}	1.706_{16}
8000	5.2	N/A	1.72_{2}	1.388_{4}	1.939_{6}
			Total Move Set [10^{-3} s]		
2000	3.3	N/A	0.992_{2}	N/A	1.630_{3}
4000	4.1	N/A	1.494_{3}	1.357_{1}	2.003_{3}
6000	4.9	N/A	2.218_{2}	2.007_{11}	2.526_{5}
8000	5.2	N/A	2.827_{2}	2.56_{2}	3.011_{2}

Table S10. CPU Timings for Simulations of TIP4P Water in the Gibbs Ensemble ($T = 298$ K, 20% of the Molecules in the Vapor Phase, and $r_{\mathrm{cut}}^{\mathrm{vap}}/L_{\mathrm{vap}} = 0.4$).

N	$L_{\mathrm{box}}/L_{\mathrm{cell}}$	LOOP	COM	CELL	TREE
			Volume Moves [s]		
2000	2.3	N/A	0.390_5	N/A	1.045_{10}
4000	2.8	N/A	0.948_3	N/A	2.246_{11}
6000	3.2	N/A	1.747_5	N/A	3.76_2
8000	3.6	N/A	2.578_{12}	N/A	5.055_{12}
			Trans/Rot Moves [10^{-3} s]		
2000	2.3	N/A	0.789_7	N/A	1.93_5
4000	2.8	N/A	0.981_6	N/A	2.17_3
6000	3.2	N/A	1.196_4	N/A	2.35_6
8000	3.6	N/A	1.366_4	N/A	2.39_6
			Swap Moves [10^{-3} s]		
2000	2.3	N/A	4.44_7	N/A	2.25_3
4000	2.8	N/A	7.82_4	N/A	2.41_1
6000	3.2	N/A	10.83_5	N/A	2.75_4
8000	3.6	N/A	13.51_6	N/A	3.41_5
			Total Move Set [10^{-3} s]		
2000	2.3	N/A	2.68_2	N/A	2.84_1
4000	2.8	N/A	4.20_6	N/A	3.11_5
6000	3.2	N/A	5.73_8	N/A	3.54_4
8000	3.6	N/A	6.89_3	N/A	3.87_{10}

Table S11. CPU Timings for Simulations of TraPPE–UA $C_{36}H_{74}$ in the NpT Ensemble ($T = 440$ K, $p = 1$ bar).

N	L_{box}/r_{cut}	L_{box}/L_{cell}	COM	TREE
			Volume Moves [s]	
100	3.4	1.1	0.176_1	0.183_2
200	4.3	2.2	0.525_1	0.349_1
300	5.0	2.5	0.790_2	0.507_3
400	5.5	2.8	1.017_{11}	0.665_4
			Trans/Rot Moves [10^{-3} s]	
100	3.4	1.1	7.017_3	4.86_5
200	4.3	2.2	10.52_2	4.87_4
300	5.0	2.5	10.70_4	4.90_6
400	5.5	2.8	10.63_8	4.92_7
			CBMC Moves [10^{-2} s]	
100	3.4	1.1	1.465_9	1.008_5
200	4.3	2.2	1.58_1	1.11_1
300	5.0	2.5	1.66_2	1.11_2
400	5.5	2.8	1.67_1	1.11_1
			Total Move Set [10^{-2} s]	
100	3.4	1.1	1.645_6	1.205_6
200	4.3	2.2	2.09_1	1.208_4
300	5.0	2.5	2.12_2	1.20_1
400	5.5	2.8	2.06_1	1.21_1

Table S12. CPU Timing Decomposition from the Allinea MAP Profile for Simulations of Lennard-Jonesium in the NVT Ensemble ($N = 12000$, $k_{\mathrm{B}}T/\varepsilon = 0.72$, $\rho\sigma^3 = 0.75$, $r_{\mathrm{cut}}/\sigma = 4$).

CPU Time	LOOP	CELL	TREE
a. Overhead [s]	8.4	1.6	2.4
a1. energy	6.4	1.1	0.9
a2. pressure	2.0	0.5	0.4
a3. data structure initialization	< 0.01	< 0.01	1.1
a4. other	< 0.01	< 0.01	< 0.01
b. Translation [10^{-4} s per MCS]	5.51	1.24	1.03
b1. energy	5.51	1.23	0.99
b1-1. range search	5.41	1.13	0.89
b1-2. potential calculation	0.10	0.10	0.10
b2. coordinate update	< 0.01	0.01	0.04
b3. other	< 0.01	< 0.01	< 0.01

Table S13. CPU Timing Decomposition from the Allinea MAP Profiler for Simulations of TraPPE–UA Ethanol in the NpT Ensemble (N = 3000, $T = 323$ K, $p = 1$ bar).

CPU Time	LOOP	COM	TREE
a. Overhead [s]	7.2	5.1	4.5
a1. energy	4.6	4.7	2.6
a2. pressure	2.6	0.4	0.8
a3. data structure initialization	< 0.01	< 0.01	1.1
a4. other	< 0.01	< 0.01	< 0.01
b. Translation/Rotation [10^{-3}s per MCS]	3.30	0.66	1.12
b1. direct space energy	3.08	0.45	0.90
b1-1. range search	2.92	0.29	0.74
b1-2. potential calculation	0.16	0.16	0.16
b2. reciprocal space energy	0.22	0.21	0.20
b3. coordinate update	< 0.01	< 0.01	0.01
b4. other	< 0.01	< 0.01	< 0.01
c. CBMC [10^{-3} s per MCS]	N/A	3.35	3.06
c1. direct space energy	N/A	2.01	1.70
c1-1. range search	N/A	1.56	1.25
c1-2. potential calculation	N/A	0.45	0.45
c2. reciprocal space energy	N/A	0.43	0.41
c3. coordinate update	N/A	< 0.01	0.03
c4. other	N/A	0.91	0.92
d. Volume [s per MCS]	N/A	0.70	1.24
d1. direct space energy	N/A	0.36	0.81
d1-1. range search	N/A	0.26	0.70
d1-2. potential calculation	N/A	0.10	0.11
d2. reciprocal space energy	N/A	0.34	0.36
d3. coordinate update	N/A	< 0.01	0.07
d4. other	N/A	< 0.01	< 0.01

Complete Author Information for Reference (28)

Siepmann, J. I.; Martin, M. G.; Chen, B.; Wick, C. D.; Stubbs, J. M.; Potoff, J. J.; Eggimann, B. L.; McGrath, M. J.; Zhao, X. S.; Anderson, K. E.; Rafferty, J. L.; Rai, N.; Maerzke, K. A.; Keasler, S. J.; Bai, P.; Fetisov, E. O.; Shah, M. S.; Chen, Q. P.; DeJaco, R. F.; Chen, J. L.; Bai. X. *Monte Carlo for Complex Chemical Systems — Minnesota*, Versions 16.1; Minneapolis, MN, 2016.

References

(1) Hoare, A. R. Algorithm 65: Find. *Commun. ACM* **1961**, *4*, 321-322.

(2) Chen, B.; Siepmann, J. I. A Novel Monte Carlo Algorithm for Simulating Strongly Associating Fluids: Applications to Water, Hydrogen Fluoride, and Acetic Acid *J. Phys. Chem. B* **104**, *36*, 8725-8734

(3) Vlugt, T. J. H.; Martin, M. G.; Smit, B.; Siepmann, J. I.; Krishna, R. Improving the Efficiency of the Configurational-Bias Monte Carlo Algorithm. *Mol. Phys.* **1998**, *94*, 727–733.