Chapter 4

Implicit Solvation Using the Superposition Approximation (IS-SPA) Simulations on GPUs

4.1 Overview

Molecular dynamics simulation is an important tool that can provide molecular-level insight into condensed phase molecular phenomenon. Often the process of interest occurs on length- and time-scales that is computationally unfeasible to approach with MD. Implicit solvent models were developed to reduce the computational demand of simulating solvated systems by removing the solvent degrees of freedom. Implicit Solvation using the Superposition Approximation is an implicit solvent model that was developed to reduce the computational demand of simulating solvated systems while maintaining a high degree of accuracy relative to explicit solvent simulations. Advancements in the performance of molecular dynamics simulations have recently focused on the use of graphics processing units due to their parallel computing power. I present a GPU implementation of the IS-SPA molecular dynamics code. Three different parallelizations of the IS-SPA algorithm are discussed including a highly parallelized algorithm, a tiling method algorithm, and a modified tiling method algorithm that removes the use of atomic operations at the expense of the level of parallelization of the IS-SPA calculations. The tiling method algorithm outscales the other two algorithms with increasing system size and the number of Monte Carlo integration points used per solute atom. Our results demonstrate that the GPU-implementation of IS-SPA is more computationally efficient than AMBER's explicit solvent all-atom molecular dynamics simulations at concentrations below 25 mM for a system of 50 alanine dipeptide molecules performed on a GPU.

4.2 Introduction

Molecular simulation is a powerful approach to investigate condensed phase molecular phenomenon such as protein folding, allostery, and self-assembly. Many of these processes, however, occur on time- or length-scales that are still challenging to approach using explicit solvent all-atom simulations. A variety of methods have been developed to address this issue including enhanced sampling protocols, particle coarse-graining, and implicit solvation. Of these methods, implicit solvation is the most computationally appealing since it offers the possibility for drastic computational savings; for many biomolecular simulations, the solvent represents $\sim$90% of the computational cost and yet the macroscopic properties of interest typically only depend on the solute positions. Despite its appeal, implicit solvation has not become mainstream due to a lack of models that provide both computational efficiency and reasonable thermodynamic consistency.

Implicit solvation is achieved by removing explicit representation of the solvent particles but including their effect in the forces felt by the solute. A variety of theoretical approaches have been used to determine the forms of the forces that should be applied. While certainly a simplification of the plethora of available methods, I separate implicit solvent models into two types: those meant for single point calculations and those meant for simulation. The single point calculation methods are largely based on classical density functional theory and include methods such as RISM[158–160] and variational implicit solvation.[161] These methods can provide highly accurate free energies of solvation but come at a significant computational cost and are thus not typically applicable for large-scale biomolecular structural sampling. One common component of these accurate methods is the consistent treatment of the polar and non-polar components of solvation. Methods designed for molecular simulation include GB[65] for polar solvation and solvent-accessible surface area (SASA) for non-polar solvation.[162–164] While these methods can be utilized in a simulation their accuracy is application, implementation, and parameterization dependent.[165] It is suggested that these methods are insufficient for molecular aggregation phenomena.[20,64]

IS-SPA is a recently developed implicit solvent model that accurately captures polar[20] and non-polar[64] components to solvation. This method has been shown to be applicable to aggregation

phenomena.[20,64] The non-polar model demonstrated significant computational speed-up compared to both explicit solvent and RISM.[64] The treatment of both polar and non-polar solvation is more computationally expensive but is still found to outperform explicit solvent simulations at solute concentrations well above typical experimental concentrations for self-assembly applications.[20] While the algorithm has demonstrated impressive accuracy and performance, the applications have been limited to relatively small systems. It is of interest, therefore, to further develop this method for large-scale applications.

Recent performance advancements in molecular simulation have focused on GPU implementation of explicit solvent aaMD methods. The recent advancements in the AMBER implementation of GB and SASA, in particular, demonstrate the importance of considering GPU implementation for simulation-based implicit solvation methods.[166] The AMBER implementation is based on a previous implementation of GB on a GPU[167,168] that used a tiling particle decomposition to distribute the pairwise computation workload efficiently across the GPU. Recent performance results have been impressive suggesting the importance of considering this scheme in the GPU algorithm for IS-SPA.

In this chapter, I present a GPU implementation of the IS-SPA algorithm. The entire code was written from scratch and each force calculation is performed on the GPU. Three different GPU parallelization algorithms are compared and contrasted. I start by briefly outlining the theory behind IS-SPA, next I describe the three algorithms, and finally I present the results of these algorithms as applied to systems of alanine dipeptide (AP) molecules.

4.3 Theory

The goal of IS-SPA is to accurately reproduce the mean solvent force on a given solute configuration. The IS-SPA theory is described in full detail by Lake *et al.*[20,64] For polar solutes in an arbitrary solvent, the mean solvent force on an atom j, given the position $\boldsymbol{R}^N$ of the N solute atoms, in the case of pairwise additive interactions is given by

$$\langle \boldsymbol{f}_j \rangle_{solv} = \rho \iint \boldsymbol{f}_{j,solv}(\boldsymbol{r} - \boldsymbol{R}_j, \boldsymbol{\Omega}) g(\boldsymbol{r}, \boldsymbol{\Omega}; \boldsymbol{R}^N)\, d\boldsymbol{\Omega}\, d\boldsymbol{r} \tag{4.1}$$

where ρ is the solvent density, $\boldsymbol{f}_{j,solv}$ is the solvent force, g is the solvent distribution function, and $\boldsymbol{\Omega}$ represents the internal coordinates of the solvent molecule. IS-SPA makes use of several approximations to analytically integrate over $\boldsymbol{\Omega}$ such that only the position of the solvent, $\boldsymbol{r}$, needs to be sampled by MC sampling.

The first approximation presumes that the only important degrees of freedom of a solvent molecule is the alignment of its dipole moment and that the molecule is axially symmetric. This reduces the 12 degrees of freedom in $\boldsymbol{\Omega}$ for a chloroform molecule to two, represented by $\hat{\boldsymbol{p}}$. The second approximation used is the the Kirkwood superposition approximation, where the many-body distribution function is reduced to a product of two-body distribution functions g_k that also take into account the orientation of the dipole moment. Specifically,

$$g(\boldsymbol{r}, \hat{\boldsymbol{p}}; \boldsymbol{R}^N) \simeq \frac{\prod_{k=1}^{N} g_k(|\boldsymbol{r} - \boldsymbol{R}_k|) P_k(\hat{\boldsymbol{p}}; \boldsymbol{r} - \boldsymbol{R}_k)}{\int \prod_{k=1}^{N} P_k(\hat{\boldsymbol{p}}; \boldsymbol{r} - \boldsymbol{R}_k)\, d\hat{\boldsymbol{p}}} \tag{4.2}$$

where P_k is the probability distribution of the dipole moment of the solvent given the distance vector $\boldsymbol{r} - \boldsymbol{R}_k$ from atom k.

The probability distribution of the dipole moment is approximated by the probability distribution of a thermal ideal dipole in a constant electric field and is given as

$$P_k(\hat{\boldsymbol{p}}; \boldsymbol{r} - \boldsymbol{R}_k) \simeq \exp\left[-\frac{p\hat{\boldsymbol{p}} \cdot (\boldsymbol{r} - \boldsymbol{R}_k)}{T|\boldsymbol{r} - \boldsymbol{R}_k|} E_k(|\boldsymbol{r} - \boldsymbol{R}_k|) \right] \tag{4.3}$$

where p is the static dipole moment of a chloroform molecule, T is the temperature in units of energy, and E_k is the radially symmetric effective mean field along its separation vector produced by each atom.

Finally, a MC integration is used to approximate the integral of Equation 4.1. Namely,

$$\langle \boldsymbol{f}_j \rangle_{solv} \simeq \frac{1}{M} \sum_{i=1}^{M} \frac{\rho}{\Gamma(\boldsymbol{r}_i)} \boldsymbol{f}_{j,solv}(\boldsymbol{r}_i - \boldsymbol{R}_j, \boldsymbol{E}_i) \prod_{k=1}^{N} g_k(|\boldsymbol{r}_i - \boldsymbol{R}_k|) \tag{4.4}$$

where Γ represents the probability distribution from which the MC points are sampled, M is the total number of MC points for all atoms, and $\boldsymbol{E}_i$ is the sum of the electric fields produced by each atom, E_k, which comes out of the product of the probability distributions of the dipole moments defined in Equation 4.3. It is important to note that the MC integration is only performed inside a defined interaction volume around each atom. The long ranged electrostatics outside of the interaction volume are approximated by a constant density dielectric. The treatment of the long ranged electrostatics are explained in greater detail by Lake *et al.*[20]

4.4 GPU Architecture and CUDA Software

To better understand the IS-SPA CUDA implementations discussed in this chapter it is import to understand the differences between GPUs and CPUs, the architecture of GPUs, and how algorithms are implemented on GPUs using the CUDA platform. GPUs and CPUs are designed to be efficient for different types of computational tasks. CPUs have latency-optimized cores and can handle a wide-range of complex tasks quickly, but these tasks must be run in serial. Alternatively, GPUs contain thousands of throughput-optimized cores making them more efficient than CPUs for computational algorithms that process large sets of data in parallel such as in the case of IS-SPA. The NVIDIA GPU architecture is built around a scalable array of multithreaded Streaming Multiprocessors (SMs) that each consist of a variable number of CUDA cores or Stream Processors (SPs) which are the most basic processing units. Both the number of SMs and SPs per SM is GPU dependent, for example a GeForce GTX 1080 GPU card contains 20 SMs each containing 128 SPs per SM for a total of 2560 CUDA cores.

CUDA is a general purpose parallel computing platform and programming model that utilizes the parallel compute engine of GPUs. A function executed on a GPU is called a kernel. A kernel is executed in parallel by an array of threads. All threads run the same code, but each thread has a unique ID that is used to select which data to work on. A block is a group of threads which are executed together. Blocks are dynamically assigned by the GPU scheduler to SMs giving GPUs their scalability. An SM can execute more than one block concurrently, but the number of blocks

that can simultaneously execute on an SM is limited by both hardware and the amount of resources each block uses including number of threads, shared memory, and number of registers. A grid is a set of blocks which together execute the GPU operation. The number of threads per block and the total number of blocks can be changed to fit the needs of a kernel, but there are limits for each that depends on the specific GPU card. Furthermore, threads are bound together in groups of 32 called warps. All threads in a single warp can only run a single set of instructions at once. For example if one thread within a warp follows one condition of an if-else statement and another thread in the same warp follows the other condition than all 32 threads in the warp will go down both sides of the statement. This does not create a problem functionally as the threads that do not satisfy the condition become inactive when following that branch of the statement, but this could cause a reduction in performance if both sides of the statement are long. Additionally, all threads within a warp fetch data from the memory together. Therefore, it is important to consider warps when trying to optimize the efficiency of the code.

Another important factor to consider when creating an efficient application is the use of memory. Due to the hierarchical nature of the GPU architecture there is also a hierarchy of memory types in size, access speeds, and lifetimes. The smallest and fastest accessible memory type on the SMs are registers. Registers are only accessible by and have the lifetime of a thread. The next fastest memory type is shared memory which can be as fast to access as registers depending on a few factors such as bank conflicts. Each block has its own shared memory that is accessible by any thread within that block and has the lifetime of the block. Shared memory is an important factor to consider when choosing how to divide calculations into blocks. The largest and slowest form of memory is global memory. Global memory can be 150x slower than registers or shared memory and has the lifetime of the application. Global memory can be accessed by both the host CPU and the GPU device and is persistent between kernel launches. Reducing the number of calls to global memory will typically improve code performance.

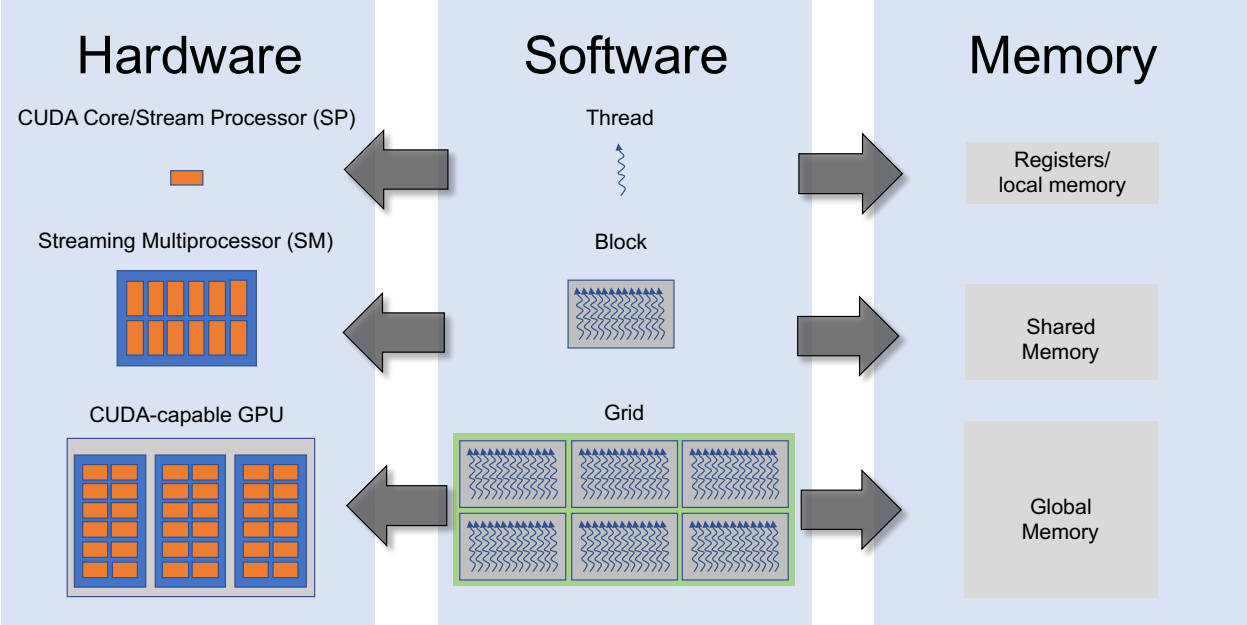

Figure 4.1: Schematic representation of the GPU architecture, the logical hierarchy of the CUDA software, and the GPU memory hierarchy.

4.5 CUDA Implementations

The CUDA algorithms that I use to perform IS-SPA implicit solvent simulations divide the calculation of the IS-SPA forces into three kernels, delineated as the MC kernel, the field kernel, and the force kernel. The MC kernel generates the positions of all of the MC points used to calculate the mean solvent force in Equation 4.4. The user defines the number of MC points per atom, N_{MC}, in the input file. The MC kernel generates N_{MC} MC points within the interaction volume of each atom producing a total of M MC points ($M = N \cdot N_{MC}$). At each MC point i, the field kernel calculates the product of the densities and the sum of the electric field, $\boldsymbol{E}_i$, produced by each atom as in Equation 4.2. Look-up tables are used to determine the density and electric field values produced at an MC point by each atom. Finally, the density and electric field are used by the force kernel to calculate the mean solvent force on each atom j by each MC point i using Equation 4.4. Both the field and force kernels calculate $N \cdot M$ atom–MC point interactions per step of the simulation making them both computationally demanding. There are numerous ways in which these kernels can be written, each with significantly differing computational efficiencies. I present three versions of the CUDA algorithm.

4.5.1 Algorithm 1: Highly Parallelized

The main goal of algorithm 1 is to fully parallelize the IS-SPA calculations such that each thread is only calculating the interactions between a single atom–MC point pair so that no loops are used. The high degree of parallelization of the IS-SPA calculations should make the CUDA algorithm quite efficient. This parallelization is done in both the field and force kernels.

The field kernel parallelizes the calculations of the electric field and density at each MC point such that each thread is only calculating the partial electric field and density that a specific atom is placing onto a specific MC point. For a particular MC point, N/W warps are created, where W is the total number of threads in a warp. If N is not a multiple of W than this leads to some warps having inactive threads. Since all threads within a warp are calculating the electric field and solvent density at a single MC point, the partial electric field and solvent densities are reduced using a warp reduction. Finally, each warp uses atomic operations to push their partial electric field and density to the global total solvent density and electric field for each MC point. Atomic operations are used to prevent race conditions where multiple threads may try to read and write to the same variable simultaneously which could cause a value to be computed incorrectly. A total of $M \cdot (N/W)$ warps are generated filling up $M \cdot (N/W)/S$ blocks, where S is the max number of warps per block.

Similarly, the force kernel parallelizes the calculation of the mean solvent forces on each atom such that each thread only calculates the partial mean solvent force from one specific MC point onto one atom. The electric field must be converted into polarization by each thread instead of being calculated one time in the field kernel due to how the field kernel is organized. For a particular atom, M/W warps are created; if M is not a multiple of W than some warps will have inactive threads. Since each warp is calculating the forces from various MC points onto one atom, the forces from each thread in the warp can be reduced using a warp reduction. Atomic operations are then used to calculate the total mean solvent force on each atom. The force kernel generates $N \cdot (M/W)$ warps across $N \cdot (M/W)/S$ blocks.

4.5.2 Algorithm 2: Tiling Method

In algorithm 2, I use a tiling method approach to calculate the mean solvent forces that is similar to the tiling method used by AMBER and Friedrich *et al.* to calculate the non-bonded forces between atoms in GB implicit solvent simulations in which the calculation of the interaction between each pair of atoms is divided into warp sized tiles that the GPU scheduler distributes across the GPU into individual blocks.[167–169] Albeit, there are a few major differences between the two methods due to differences in how the two forces are calculated. The pairwise interactions between MC point i and atom j can be represented by the matrix shown in Figure 4.2, where the atom–MC point interactions are grouped into tiles of dimension $W \times W$. The reason for the division of the calculations in this fashion is because the GPU scheduler works in terms of warps, where each warp performs the same mathematical operation on W values simultaneously.[169] Unlike calculating the non-bonded interactions between all pairs of atoms, there is no symmetry that can be exploited in the interactions between MC point–atom pairs to reduce the number of calculations needed to be performed. In determining the non-bonded forces, the diagonal tiles must be treated differently then the off-diagonal tiles and only the upper triangle of the matrix needs to be calculated due to Newton's third law of motion. In the tiling method used in calculating the mean solvent forces, every tile in the matrix must be calculated and every tile is calculated in the same way. Both the field and force kernel generate $(N/W) \cdot (M/W)$ blocks, assigning each tile to a block.

In the field kernel a tile assigns each thread in the block an MC points between i to $i + W - 1$. Each thread then loops over atoms, from atom j to $j + W - 1$, calculating the electric field and density produced by the atom at the MC point assigned to that thread as shown by the blue tile in Figure 4.2. Since this kernel is calculating a per MC point quantity, this setup avoids any race conditions within a tile. Race conditions between tiles must still be considered, therefore, atomic operations are used to update the global electric field and densities calculated from each tile. Also, the charge and position of atoms j to $j + W - 1$ are stored in shared memory.

Conversely, a tile in the force kernel assigns each thread in the block an atom between j to $j + W - 1$. Each thread loops over MC points between i and $i + W - 1$, calculating the partial

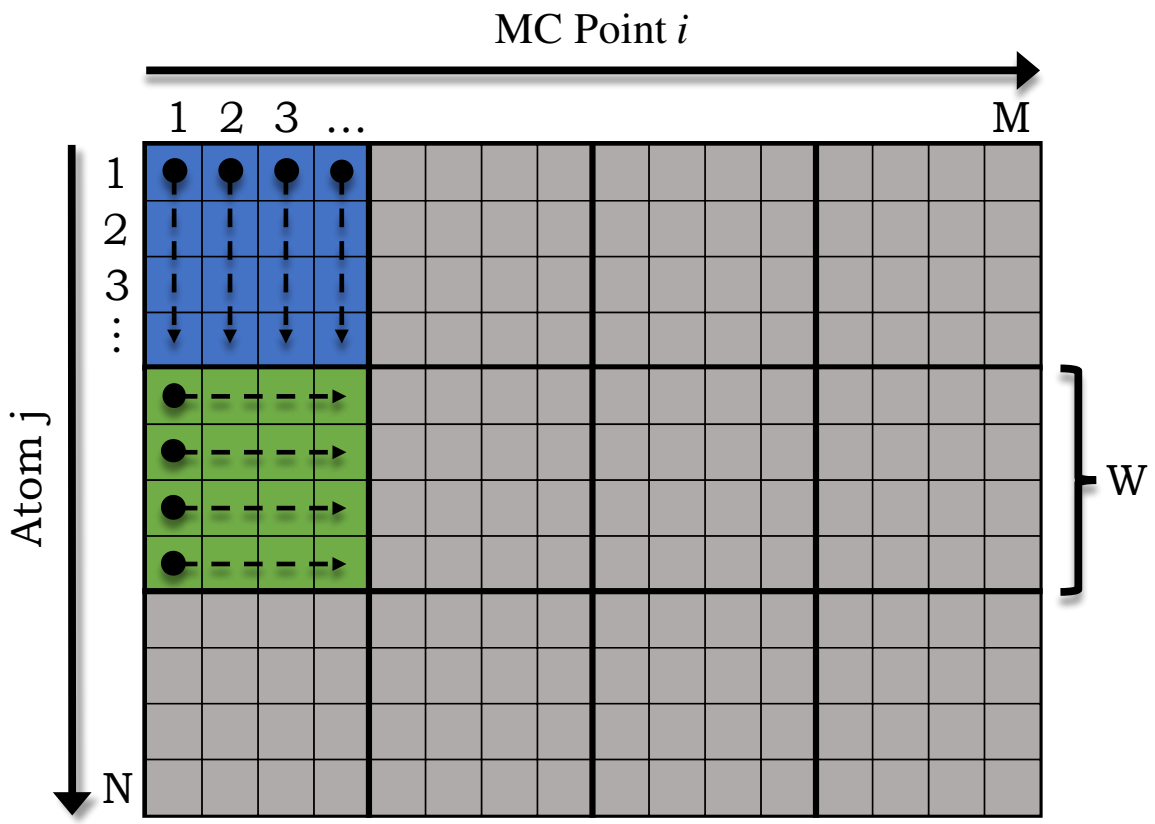

Figure 4.2: Schematic representation of the of the work-load distribution for the calculation of the IS-SPA forces using algorithm 2. Each box represents the interaction between an MC point i with an atom j. These interactions are grouped into tiles of size $W \times W$ that are each assigned to an independent warp. A tile in the IS-SPA field kernel assigns each thread an MC point i and loops over W atoms as illustrated by the blue tile. A tile in the IS-SPA force kernel assigns each thread an atom j and loops over MC points as demonstrated by the green tile.

mean solvent force that the MC point produces on the atom assigned to that thread as shown by green tile in Figure 4.2. This setup allows race conditions to be avoided within a tile because the force kernel is calculating a per atom quantity. Similar to the field kernel, atomic operations must be used to update the global IS-SPA forces due to possible race conditions between tiles. In this kernel, the electric field and density of the MC points i to $i + W - 1$ are stored in shared memory.

4.5.3 Algorithm 3: Tiling Method with no Atomic Operations

In algorithm 3, I use an approach similar to the tiling method with changes in the size of the tile such that all race conditions are avoided. It is suggested that atomic operations are a bottleneck in

terms of performance in the tiling method.[169] Therefore, this approach removes all race conditions between tiles by increasing the size of the loops performed by each tile. Tiles are still organized such that there are no race conditions within a tile.

In algorithm 2, the field kernel assigns a warp-sized block to each MC point for a total of M blocks. Each block then calculates the total electric field and density for its MC point i by looping over all atoms as shown in Figure 4.3. The loop over all atoms is divide among W threads within the warp, such that atoms $j = 0$ to $j = W - 1$ are assigned to a thread. Then the the loop increments j for each thread by W until all atoms are considered for MC point i. At the end of the loop the total electric field and density at the MC point assigned to the warp is calculated by warp reducing the W partial electric fields and densities of each thread. Since each warp is considering a different MC point, the total electric field is converted into a polarization and is then pushed to the global variable without any race conditions.

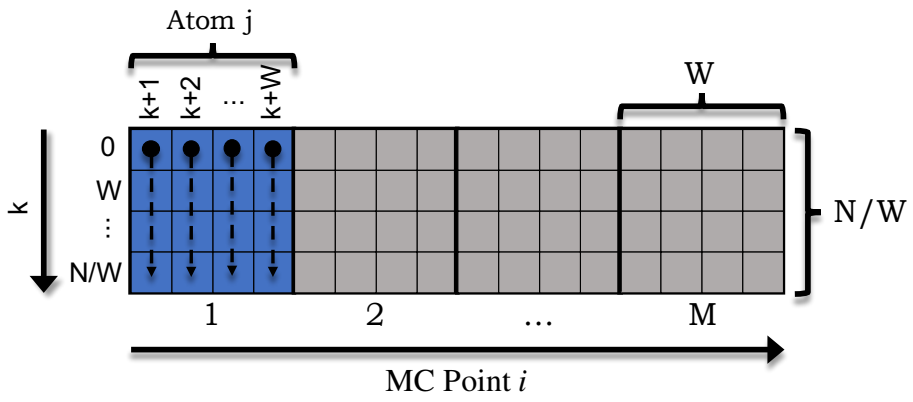

Figure 4.3: Schematic representation of the of the work-load distribution for the calculation of the electric fields and densities by the field kernel using algorithm 3. Each box represents the interaction between an MC point i with an atom j. These interactions are grouped into tiles of size $W \times N/W$ that are each assigned to an independent warp. A tile in the IS-SPA field kernel assigns each warp am MC point i and loops over all atoms as illustrated by the blue tile.

Conversely, the force kernel assigns a warp-size block to each atom for a total of N blocks. Each block calculates the mean solvent force for its atom j by looping over all MC points as shown in Figure 4.4. The loop over all MC points is divided among all W threads within the warp. The

total mean solvent force of the atom is calculated by warp reducing the W partial mean solvent forces of each thread. Since each warp is calculating the mean solvent force on a single atom, no race conditions are encountered when pushing the total force to the global variable.

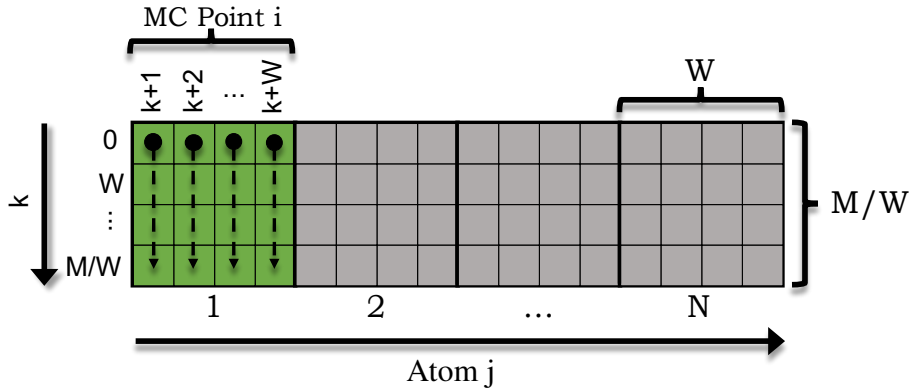

Figure 4.4: Schematic representation of the of the work-load distribution for the calculation of the IS-SPA forces by the force kernel using algorithm 3. Each box represents the interaction between an MC point i with an atom j. These interactions are grouped into tiles of size $W \times M/W$ that are each assigned to an independent warp. A tile in the IS-SPA field kernel assigns each warp an atom j and loops over all MC points as illustrated by the green tile.

4.6 Validation

IS-SPA simulations of both Lennard-Jones ions and AP molecules are performed to validate the accuracy of the IS-SPA CUDA code. The charged Lennard-Jones spheres are used as a simple test case for the IS-SPA CUDA procedure and AP is chosen as a model molecular system as it is a typical test system for solvent models.[161,170] In particular, these two systems were chosen so that the results of IS-SPA CUDA code could be compared to the results presented by Lake *et al.*[20] The IS-SPA simulations are performed in the NVT ensemble with an Andersen thermostat with a collision frequency of 33.33 ps^{-1}. The mass of the hydrogen atoms is set to 12 u to reduce the frequency of those bonds and allow the use of a 2 fs integration time step. No cutoff is used in the calculation of the non-bonded and solvent forces in the IS-SPA simulations. N_{MC} is set to 100 MC points per atom for the dimer IS-SPA simulations of both systems. A radial distance of 12 Å defines

the spherical interaction volume around each atom and is chosen based on when variations in the density and polarization tend to the bulk values.[20] The external dielectric constant of chloroform is set to 2.3473 as measured from the bulk chloroform model.[171,172] The same IS-SPA parameters are used for both the Lennard-Jones ions and the AP molecules as the IS-SPA parameters used by Lake *et al*.

4.6.1 Lennard-Jones Sphere

To validate the accuracy of the IS-SPA CUDA code, the dimerization potential of mean force (PMF) of a pair of Lennard-Jones particles in chloroform is calculated and compared to the Lennard-Jones dimer PMF presented by Lake *et al*.[20] To parallel the simulations presented in that study, the Lennard-Jones ions are given charges of $q = +1$ and $q = -1$ and Lennard-Jones parameters of $\epsilon = 0.152$ kcal/mol and $r_{min} = 7$ Å. The dimer PMF calculated by Lake *et al.* relies on grid integration to numerically solve the integral in Equation 4.1, where as the IS-SPA CUDA code utilizes MC integration to the solve the integral. In both cases, the total force on the positive and negative ions are calculated at separation distances from 3.5 Å to 15.0 Å at intervals of 0.1 Å. The red and blue IS-SPA PMF curves presented in Figure 4.5 are the integration of the positive and negative ion forces, respectively. There is complete agreement between the curves produced by the IS-SPA CUDA code and those presented by Lake *et al* demonstrating the accuracy of the IS-SPA CUDA code. There is deviation from the PMF produced by explicit solvent simulations, but that is a statement on the accuracy of IS-SPA as an implicit solvent model and not on the accuracy of the IS-SPA CUDA code.

4.6.2 Alanine Dipeptide

Further validation of the accuracy of the IS-SPA CUDA code is performed by comparison of the dimerization PMFs of AP molecules in chloroform presented in Figure 4.6. The dimerization behavior along the center-of-mass separation distance is investigated using umbrella sampling simulations performed by the IS-SPA CUDA code. Similarly, umbrella sampling simulations are performed by the IS-SPA Fortran CPU code developed by Lake *et al*.[20] These simulations are

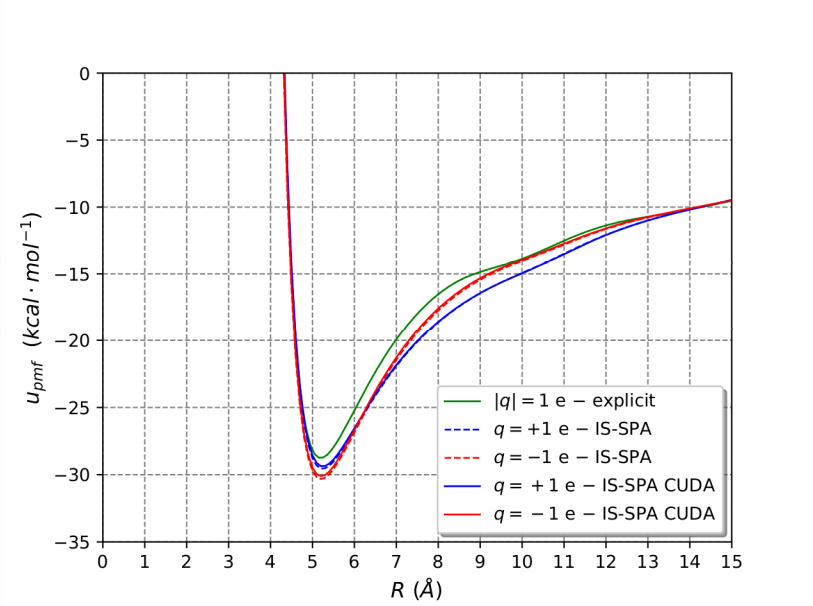

Figure 4.5: Dimerization PMFs of two Lennard-Jones ions with a charge of $q = +1$ (blue curves) and $q = -1$ (red curve) produced by the IS-SPA CUDA code (solid lines) and presented by Lake *et al.* (dotted lines). Furthermore, the explicit solvent dimerization PMF is shown by the green curve.

performed every 0.5 Å from 3.5 Å to 16.0 Å using a force constant of 20 kcal/mol/Å^2 and each window is simulated for 150 ns. There is complete agreement between the two PMFs as shown by the red and blue curves in Figure 4.6 further demonstrating the accuracy of the IS-SPA CUDA code. Again, the discrepancy between the IS-SPA PMFs and the explicit solvent PMF is due to the accuracy of IS-SPA as an implicit solvent model and not the IS-SPA CUDA code, specifically.

4.7 Performance

Although I have developed a full MD CUDA code to perform IS-SPA simulations, the work presented here focuses on optimizing the efficiency of calculating the IS-SPA solvent forces. Other parts of the code, such as the non-bonded interactions, are not highly optimized and, therefore, could provide additional increase in code performance. There are an abundance of studies in the literature that discuss the optimization of cMD simulations performed using CUDA, with a large focus on the non-bonded interactions which constitutes the largest bottleneck in cMD code performance.[161,166,168,169,171,173–177] In the IS-SPA simulations the largest bottleneck lies in

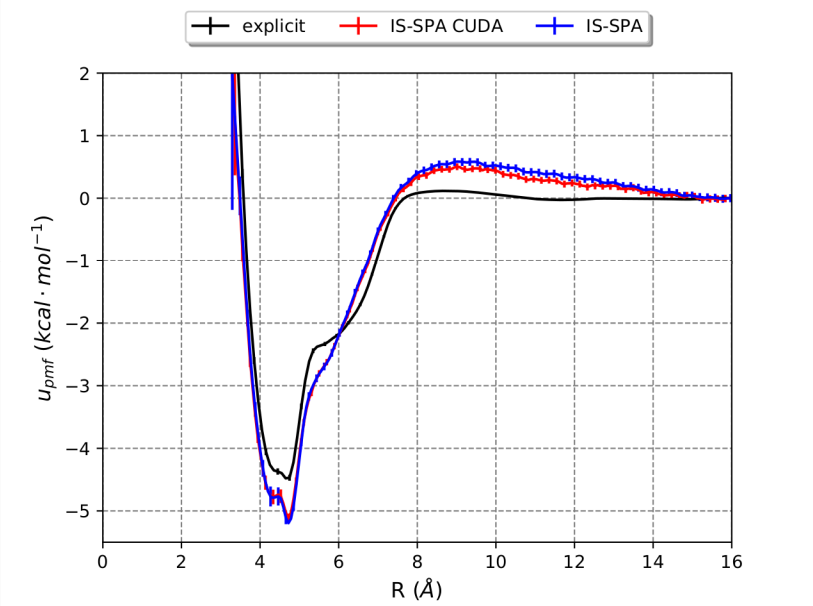

Figure 4.6: Dimerization PMFs of AP along the center-of-mass separation distance using explicit solvent, IS-SPA performed by the IS-SPA Fortran CPU code, and IS-SPA performed by the IS-SPA CUDA GPU code.

the calculation of the IS-SPA solvent forces on the solutes. Therefore, it is not our goal to fully optimize the entire MD code, rather, our goal is to highly optimize the calculation of the IS-SPA solvent forces such that it could be adapted into a highly-developed MD software. For this reason I discuss the relative code efficiency between IS-SPA algorithms, but do not compare the overall code performance with other MD codes. For the three algorithms, I compare how their performance scales with both the number of solute atoms and the number of MC points per solute atom. All performance calculations were performed on a GeForce GTX 1080 GPU card.

To determine how the performance of each algorithm scales with the total number of solute atoms I performed a series of timescaling simulations for systems of 2, 10, 25, and 50 AP molecules in chloroform. Each system was performed with $N_{MC} = 96$ for 1000 steps. To obtain each timescaling point the performance was averaged over 100 individual runs. The timescaling as a function of the number of solute atoms is shown in Figure 4.7. For all system sizes, the tiling method, algorithm 2, outperforms the other algorithms. As the system size increases the performance of algorithms 1 and 3 declines more quickly than in algorithm 2, suggesting that algorithm 2 will continue to outscale algorithms 1 and 3 at continually increasing system sizes. Algorithms 1 and

3 perform similarly for the dimer AP system, but as the system size increases the performance of algorithm 3 surpasses that of algorithm 1.

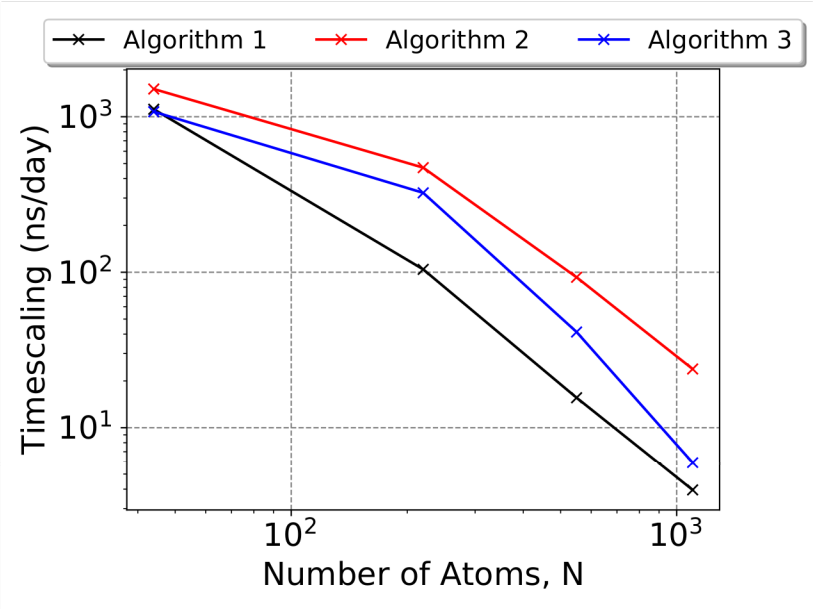

Figure 4.7: Timescaling as a function of the number of atoms, N, for $N_{MC} = 96$.

To determine how the performance of the three algorithms scale with the number of MC points per solute atom I performed timescaling simulations of AP systems with 2 and 50 AP molecules in chloroform with varying N_{MC} values. Figure 4.8(a) presents the performance of the dimer AP system with N_{MC} ranging from 8 to 256 MC points. Figure 4.8(b) shows the performance of the system with 50 AP molecules with N_{MC} ranging from 8 to 152 MC points. In both cases N_{MC} was increased incrementally by 8 MC points. For the dimer system, at low N_{MC} values algorithm 1 and 3 outperform algorithm 2. Once N_{MC} reaches a value of 40 MC points or higher algorithm 2 begins to outscale the other two, as the performance of algorithms 1 and 3 decreases sharply with increasing N_{MC}. Algorithms 1 and 3 scale similarly for the dimer system until N_{MC} is greater than 96 MC points at which algorithm 1 performs slightly better than algorithm 2. For the system of 50 AP molecules, algorithm 2 greatly outperforms algorithms 1 and 3 for all N_{MC} values, especially when N_{MC} is small. For small N_{MC}, algorithm 3 performs slightly better than algorithm 1, but the difference between them diminishes as N_{MC} increases. In general for both system sizes algorithm

2 performs significantly better than algorithm 1 and 3, which both perform similarly. Also, the performance of all three algorithms declines significantly as N_{MC} increases.

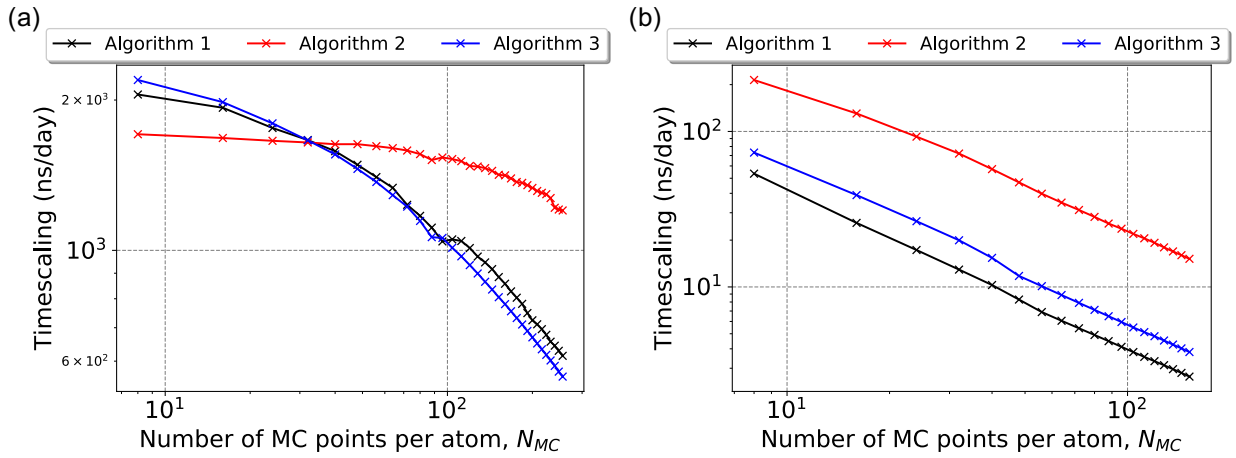

Figure 4.8: Timescaling as function of the number of MC points per atom, N_{MC}, for (a) $N = 44$ and (b) $N = 1100$.

These results demonstrate that fully parallelizing the calculation of the IS-SPA solvent forces is not optimal and that grouping things into tiles leads to better code performance. Although it is suggested that atomic operations can lead to a significant loss in performance, I show here that removing atomic operations at the expense of higher parallelism leads to lower performance of the code. Our results suggest that the code must strike a delicate balance between the use of expensive operations such as atomic operations and the level of parallelization of the calculations such as in the tiling method used by algorithm 2.

The current utility of IS-SPA lies in simulating systems of small molecules at low concentrations due to the current requirement of a fixed number of MC points per atom.[20] Table 4.1 shows the performance of explicit solvent aaMD of a system of 50 AP molecules in chloroform for a variety of concentrations and, therefore, system sizes performed on a GeForce GTX 1080 card using AMBER's CUDA MD code. It was demonstrated that algorithm 2 is the highest performing implementation of IS-SPA on a GPU for a system of 50 AP molecules, producing 24 ns of simulation a day. Comparison of the performance of the algorithm 2 implementation of IS-SPA to the performance of

the explicit solvent aaMD simulations reveals that IS-SPA begins to outperform explicit solvent simulation at concentrations below 25 mM. At concentrations below 25 mM the systems size becomes to large to be simulated efficiently. While this demonstrates where the usage of the current IS-SPA CUDA MD code becomes beneficial, the performance of the IS-SPA MD code can still be improved, further extending the utility of IS-SPA.

Explicit Solvent Performance		
Concentration (mM)	Number of Atoms	Performance (ns/day)
5	613920	9.02
10	301125	16.33
25	118665	39.78
50	57450	84.82
100	31765	156.16
500	7215	561.00
1000	4200	744.41

Table 4.1: The performance of explicit solvent aaMD simulations of a system of 50 AP molecules in chloroform for a variety of concentrations and system sizes. All simulations were performed using AMBER's highly optimized CUDA MD software on a GeForce GTX 1080 GPU card.

4.8 Conclusion

IS-SPA was developed to reduce the computational cost of simulating solvated systems by removing the solvent degrees of freedom while still capturing the force of the solvent on the solute. Until now, IS-SPA has only been implemented in a parallelizable CPU Fortran code. It has been shown that the parallel computing power of GPUs can provide significant speed up in the performance of MD simulations over CPU MD codes. Therefore, an implementation of IS-SPA in a GPU-capable CUDA code is needed to further improve the IS-SPA simulation performance. I present a full IS-SPA MD code with a focus on optimizing the calculation of the IS-SPA solvent forces. Three different algorithms are developed focusing on different methods to enhance performance. Based on the scaling of the performance of the three algorithms with the number of solute atoms

and the number of MC points per atom, algorithm 2, the tiling method, outperforms the other two algorithms as it strikes a balance between the use of atomic operations and parallelization of the IS-SPA calculations.

Future work will aim at further optimizing the IS-SPA CUDA MD code. There are at least three methods that could be utilized to improve code performance. The first is to further optimize the current version of the CUDA code. One way to enhance the performance of the code is to limit the use of atomic operations in a different manner than that done by algorithm 3 that still allows for a high level of parallelization of the IS-SPA calculations. For example, this was done in the tiling method used by AMBER in which they used buffers to avoid race conditions between tiles and, therefore, avoided the use of atomic operations.[169] Such a method could be implemented into algorithm 2 and may provide additional increase in computational performance of the IS-SPA CUDA code. A second way in which the IS-SPA MD code may be improved is to introduce a cutoff for the IS-SPA interactions, although, it is not clear if this could be implemented in a way to enhance code performance. For all three algorithms the performance decreases quickly with an increasing N_{MC} value. This suggests that third possible source of optimization is reducing the number of MC points per atom that is needed to sufficiently sample the IS-SPA solvent forces. In the current implementation of IS-SPA, 96 MC points are uniformly chosen in a spherical volume around a solute to estimate the integral in Equation 4.1. To reduce the value of N_{MC}, various sampling distributions of the MC points could be tested to find a distribution that reduces the variance of the mean force on a particular test molecule allowing for smaller values of N_{MC} to be used but still maintain a relatively low degree of error in solving the integral in Equation 4.1. This was done by Lake *et al.* in the implementation of IS-SPA for the nonpolar component of solvation, but has not been done for the extension of IS-SPA to polar solvents.[20,64]