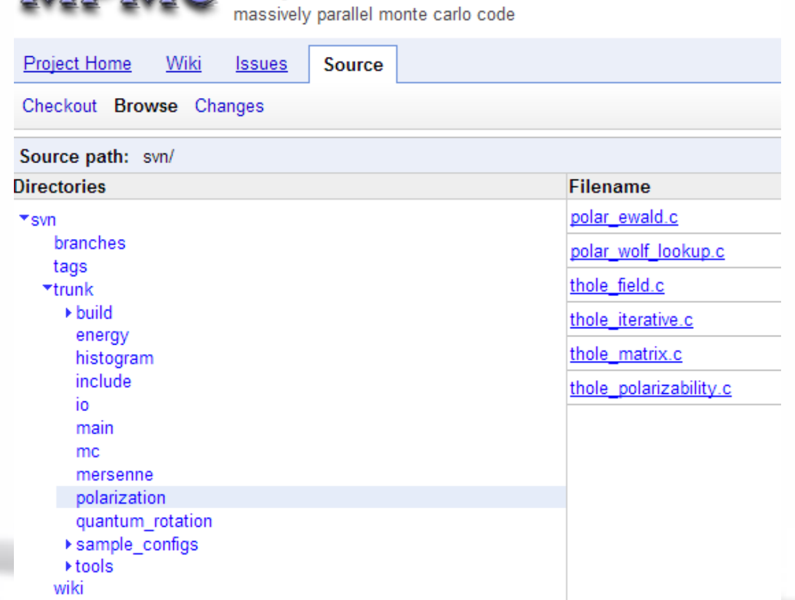
FMRI RET 2014-Solving Many-Body Polarization on GPU's

Pedro Villavicencio[1], Mentors[2]: Brant Tudor, Christian R. Cioce, Brian Space

1. Steinbrenner High School; 2. Department of Chemistry, University of South Florida

Abstract


MPMC mpmc
massively parallel monte carlo code

Source path: svn/

Directories: svn, branches, tags, trunk, build, energy, histogram, include, io, main, mc, mersenne, polarization, quantum_rotation, sample_configs, tools, wiki

Filename: polar_ewald.c, polar_wolf_lookup.c, thole_field.c, thole_iterative.c, thole_matrix.c, thole_polarizability.c

Massively Parallel Monte Carlo, an in-house computer code available at http://code.google.com/p/mpmc/, has been successfully utilized to simulate interactions between gas phase sorbates and various metal-organic materials. In this regard, calculations involving polarizability were found to be critical, and computationally expensive. Although GPGPU routines have increased the speed of these calculations immensely, in its original state, the program was only able to leverage a GPU's power on small systems. In order to study larger and evermore complex systems, an attempt to modify the program model using new parallel programming techniques was performed. It was expected that the new techniques, while providing similar or better performance, will solve limitations related to convergence times, system size and scalability. After implementation, the new techniques yielded results that were more or less five percent accurate, lower than the ideal. In this project, another attempt is made to improve the accuracy of the model which will open a path to subsequently solve the aforementioned limitations.

Objectives

Convert Gauss-Seidel CPU-based process to GPU-based parallel process
Verify the accuracy of the GPU polarization results to one percent.
Decrease the Gauss-Seidel convergence time.

Background

The Space group is a theoretical chemistry group concerned primarily with computer simulation of condensed phase phenomena. Current focus is on the development of highly accurate potential energy functions for environmentally relevant gases, such as carbon dioxide, hydrogen, nitrogen, methane, oxygen and associated oxides. The potentials are used in molecular simulations of sorption of such gases within metal-organic materials, or MOMs; MOMs offer great potential as H_2 storage, CO_2 capture and $N_2/CO_2/CH_4/O_2$ separation platforms …. Computer simulation is a highly effective tool to intelligently engineer MOMs

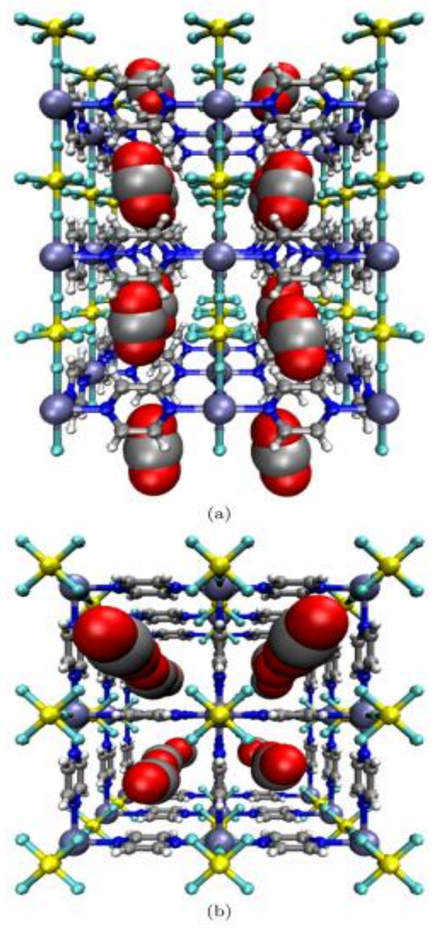
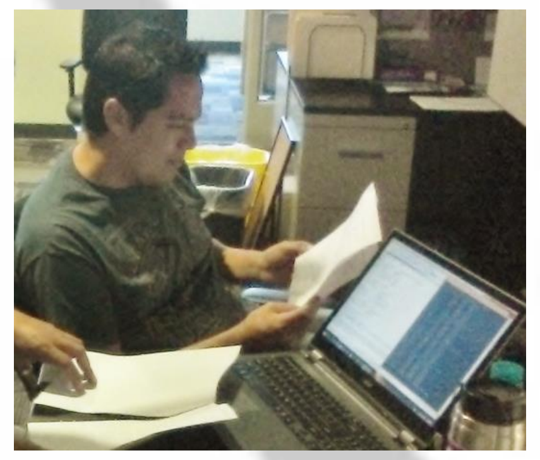
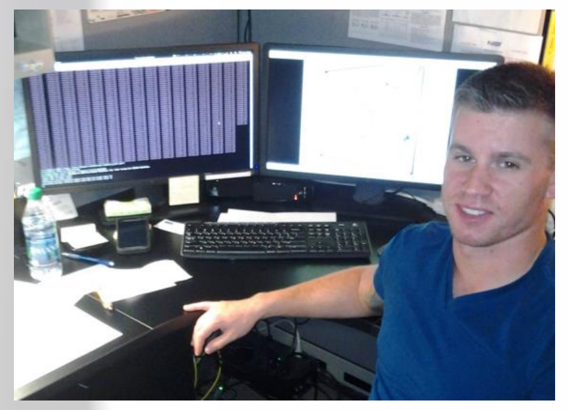


Figure 1.(a) The side view and (b) top view of the modeled 3 × 3 × 3 unit cell system of $[Zn(pyr)_2(SiF_6)]$ at CO_2 saturation, which corresponds to 1 CO_2 molecule per unit cell. Note the terminal pyrazine units were removed for clarity. MOM atom colors: Zn = purple, Si = yellow, F = cyan, N = blue, C = gray, H = white. CO2 molecule atom colors: C = gray, O = red.

Source: The Journal of Physical Chemistry - Computational Studies of CO2 Sorption and Separation in an Ultramicroporous Metal−Organic Material: **Katherine A. Forrest, Tony Pham, Adam Hogan, Keith McLaughlin, Brant Tudor, Patrick Nugent, Stephen D. Burd, Ashley Mullen, Christian R. Cioce, Lukasz Wojtas, Michael J. Zaworotko, and Brian Space**





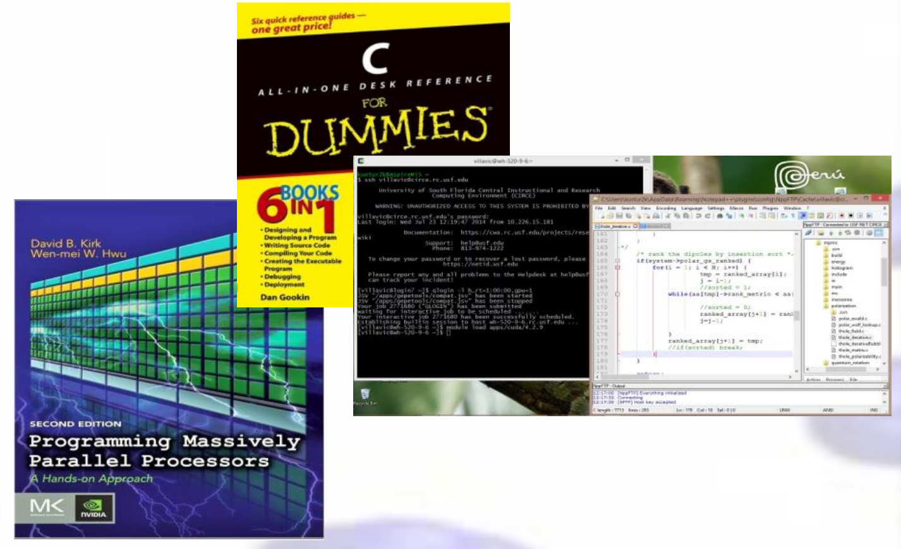
Polarization calculations in MPMC are conducted using the Thole-Applequist model. In this model, the dipole for a molecule is then treated as a collection of N atomic-point dipoles, which are summed to give the net dipole for the set.

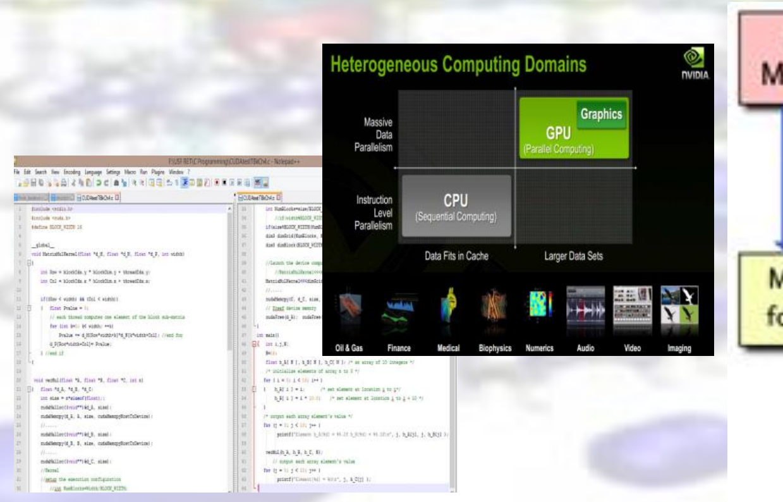
$$\vec{\mu}_{mol} = \sum_i^N \vec{\mu}_i = \sum_i^N \alpha_i \vec{E}_i^{stat} \quad \left\{ \vec{\mu} = \begin{pmatrix} \vec{\mu}_1 \\ \vec{\mu}_2 \\ \vdots \\ \vec{\mu}_N \end{pmatrix} \quad \vec{E}^{stat} = \begin{pmatrix} \vec{E}_1^{stat} \\ \vec{E}_2^{stat} \\ \vdots \\ \vec{E}_N^{stat} \end{pmatrix} \xrightarrow{\quad} \text{if matrices A and B are defined as } A = \left[(\alpha^{\circ})^{-1} + T_{ij}^{\alpha\beta}\right] \quad B = A^{-1} \right\} \quad A\vec{\mu} = \vec{E}^{stat} \quad \vec{\mu} = B\vec{E}^{stat}$$

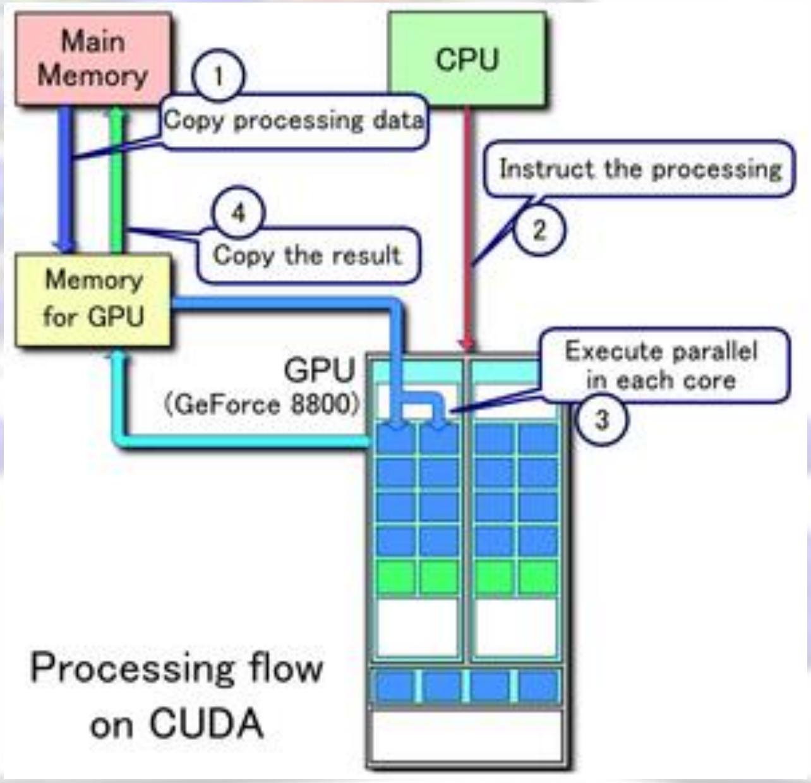
The size of matrices required to model typical MOF systems renders the computation required for matrix inversion impractical. MPMC solves these equations by guessing at the value of each point dipole and solving [the last] equation[s] iteratively. Iteration's accuracy and speed relies on convergence. Still, a test for convergence of the GPGPU polarization calculation was not implemented in the original version of MPMC. Hence, the computation would run for a preset number of iterations and results were delivered without any way of estimating their accuracy.

Approach

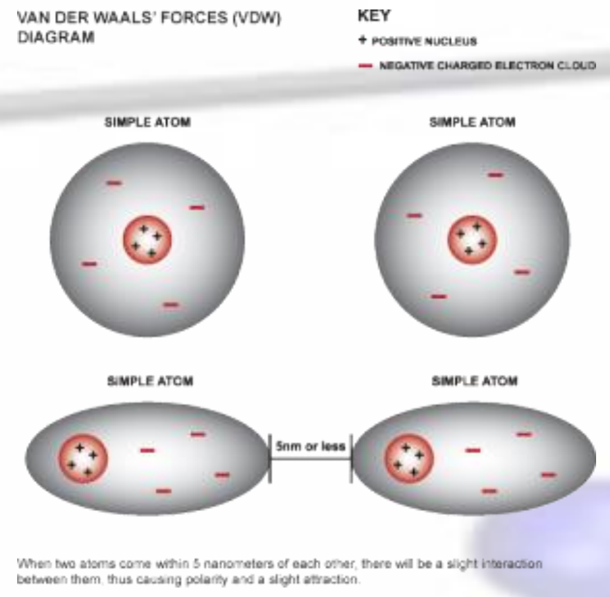
MPMC uses C language for the CPU-based routines while CUDA is the superset language used to port routines to the GPU for parallel processing. The working eco-system uses the Linux/Unix operating system.


Six quick reference guides — one great price!
C ALL-IN-ONE DESK REFERENCE FOR DUMMIES
6 BOOKS IN 1
Dan Gookin
David B. Kirk
Wen-mei W. Hwu
SECOND EDITION
Programming Massively Parallel Processors
A Hands-on Approach
MK
NVIDIA





A review of UNIX and the C language was initiated along with self-teaching of CUDA parallel programming.



When two atoms come within 5 nanometers of each other, there will be a slight interaction between them, thus causing polarity and a slight attraction.

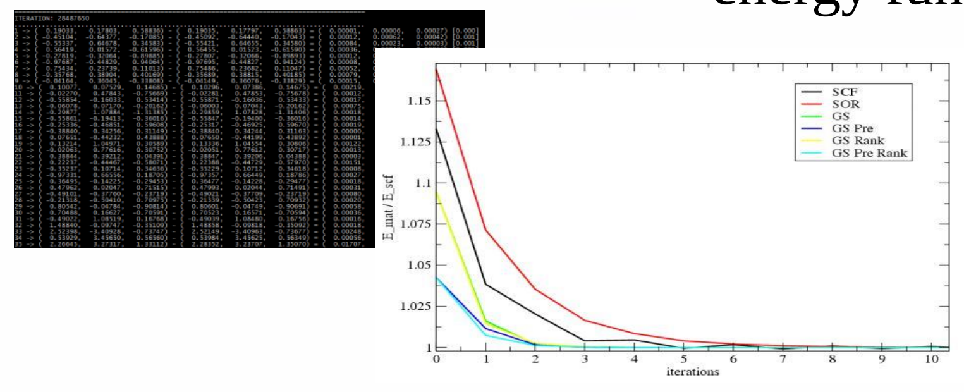
It was important to understand the algorithms used by the theoretical chemistry group in MPMC to represent concepts such as Lennard-Jones parameters, Van der Waals' forces, many-body polarization, Gauss-Seidel, total-system potential energy function among others.

$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] = \varepsilon\left[\left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}\right],$$

Lennard-Jones parameters, representing van der Waals interactions between sorbate molecules and the MOM atoms, were taken from known force fields.

$$\langle N \rangle = \frac{1}{\Xi}\sum_{N=0}^{\infty} e^{\beta\mu N}\left\{\prod_{i=1}^{3N}\int_{-\infty}^{\infty} dx_i\right\} N e^{-\beta U(x_1,\ldots,x_{3N})}$$

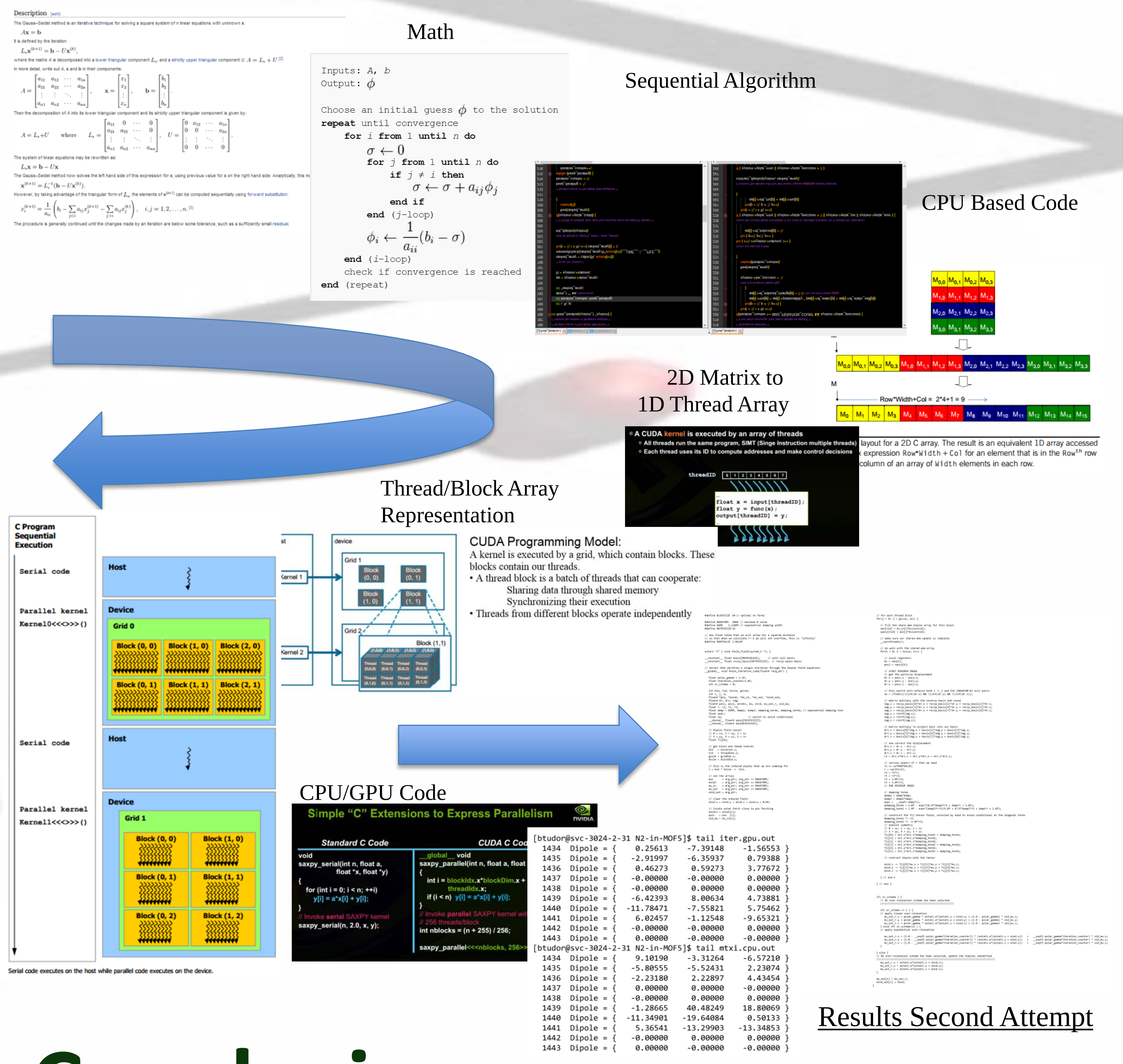
The following statistical mechanical expression was numerically calculated using GCMC, where Ξ is the grand canonical partition function, β represents the quantity 1/kT (T is the temperature, and k is the Boltzmann constant), and μ represents the chemical potential of the gas reservoir.



The iterated energy vs. the exact matrix inversion energy as a function of iterative convergence for various SCF methods.

With benchmark results on convergence and accuracy it was decided to replace a module from the old MPMC GPU version with the Gauss-Seidel Preconditioned method adapted from the original MPMC CPU program.

Gauss – Seidel CPU to CPU/GPU Conversion



The Gauss–Seidel method is an iterative technique for solving a square system of n linear equations with unknown x: $Ax = b$. It is defined by the iteration $L_* x^{(k+1)} = b - U x^{(k)}$, where the matrix A is decomposed into a lower triangular component L_*, and a strictly upper triangular component U: $A = L_* + U$. The procedure is generally continued until the changes made by an iteration are below some tolerance, such as a sufficiently small residual.

```
Inputs: A, b
Output: $\phi$

Choose an initial guess $\phi$ to the solution
repeat until convergence
    for i from 1 until n do
        $\sigma \leftarrow 0$
        for j from 1 until n do
            if $j \neq i$ then
                $\sigma \leftarrow \sigma + a_{ij}\phi_j$
            end if
        end (j-loop)
        $\phi_i \leftarrow \frac{1}{a_{ii}}(b_i - \sigma)$
    end (i-loop)
    check if convergence is reached
end (repeat)
```

A CUDA kernel is executed by an array of threads
- All threads run the same program, SIMT (Singe Instruction multiple threads)
- Each thread uses its ID to compute addresses and make control decisions

```
float x = input[threadID];
float y = func(x);
output[threadID] = y;
```

layout for a 2D C array. The result is an equivalent 1D array accessed expression Row*Width + Col for an element that is in the Row^th row column of an array of Width elements in each row.

CUDA Programming Model:
A kernel is executed by a grid, which contain blocks. These blocks contain our threads.
- A thread block is a batch of threads that can cooperate:
  Sharing data through shared memory
  Synchronizing their execution
- Threads from different blocks operate independently

Serial code executes on the host while parallel code executes on the device.

```
void
saxpy_serial(int n, float a,
             float *x, float *y)
{
  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}
// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);
```

```
__global__ void
saxpy_parallel(int n, float a, float
{
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}
// Invoke parallel SAXPY kernel wi
// 256 threads/block
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>
```

```
[btudor@svc-3024-2-31 N2-in-MOF5]$ tail iter.gpu.out
 1434  Dipole = {    0.25613    -7.39148    -1.56553 }
 1435  Dipole = {   -2.91997    -6.35937     0.79388 }
 1436  Dipole = {    0.46273     0.59273     3.77672 }
 1437  Dipole = {   -0.00000    -0.00000     0.00000 }
 1438  Dipole = {   -0.00000    -0.00000     0.00000 }
 1439  Dipole = {   -6.42393     8.00634     4.73881 }
 1440  Dipole = {  -11.78471    -7.55821     5.75462 }
 1441  Dipole = {    6.02457    -1.12548    -9.65321 }
 1442  Dipole = {   -0.00000    -0.00000     0.00000 }
 1443  Dipole = {    0.00000    -0.00000    -0.00000 }
[btudor@svc-3024-2-31 N2-in-MOF5]$ tail mtxi.cpu.out
 1434  Dipole = {    9.10190    -3.31264    -6.57210 }
 1435  Dipole = {   -5.80555    -5.52431     2.23074 }
 1436  Dipole = {   -2.23180     2.22897     4.43454 }
 1437  Dipole = {    0.00000     0.00000    -0.00000 }
 1438  Dipole = {   -0.00000    -0.00000     0.00000 }
 1439  Dipole = {   -1.28665    40.48249    18.80069 }
 1440  Dipole = {  -11.34901   -19.64084     0.50133 }
 1441  Dipole = {    5.36541   -13.29903   -13.34853 }
 1442  Dipole = {   -0.00000     0.00000     0.00000 }
 1443  Dipole = {    0.00000    -0.00000    -0.00000 }
```

Results Second Attempt

Conclusions

- Benchmark results showed flaws on the first version of the GPU code.
- Conversion of the Gauss-Seidel iterative module is still underway; each attempt is benchmarked to determine improvements on accuracy, speed and convergence.
- A recommendation to convert a sort method from Bubble Sort to Insertion Sort algorithm will be considered as a future implementation.
- Addressing MPMC's growth by system size and scalability will follow.

Referenced Resources

1.) "Mpmc - Massively Parallel Monte Carlo Code - Google Project Hosting." *Mpmc - Massively Parallel Monte Carlo Code - Google Project Hosting*. N.p., n.d. Web. 26 July 2014. <https://code.google.com/p/mpmc/>.

2.) Belof, Jonathan L. *Theory and Simulation of Metal-organic Materials and Biomolecules*. Diss. U of South Florida, 2009. N.p.: n.p., n.d. Print.

4.) Tudor, Brant, and Brian Space. "Solving the Many-Body Polarization Problem on GPUs: Application to MOFs."*Journal of Computational Science Education* 4.1 (n.d.): n. pag. Print.

6.) Puiu, Tibi. "Direct Measurement of Van Der Waals Force Made for the First Time." *ZME Science*. Physics, 5 July 2013. Web. 26 July 2014. <http://www.zmescience.com/science/physics/direct-measurement-of-van-der-waals-force-made-for-the-first-time/#!bkAgQc>.

7.) Forrest, Katherine A., Tony Pham, Adam Hogan, Keith McLaughlin, Brant Tudor, Patrick Nugent, Stephen Burd, Ashley Mullen, Christian R. Cioce, Lukasz Wojtas, Michael J. Zaworotko, and Brian Space. "Computational Studies of CO2 Sorption and Separation in an Ultramicroporous Metal-Organic Material." *The Journal of Physical Chemistry* (n.d.): n. pag. *Pubs.acs.org/JPCC*. Web. 26 July 2014. <pubs.acs.org/JPCC>.

8.) Cioce, Christian. "MOF Flue." *YouTube*. YouTube, n.d. Web. 26 July 2014. <http://www.youtube.com/watch?v=_ph3ceMm75Q&feature=youtu.be>.

9.) Kirk, David, and Wen-mei Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. Amsterdam ; Boston: Elsevier/Morgan Kaufmann, 2013. Print.

10.) Tumeo, Antonino. *Massively Parallel Computing with CUDA.* Presentation File. Politecnico di Milano. 26 July 2014.

11.) "CUDA." *Wikipedia*. Wikimedia Foundation, 25 July 2014. Web. 26 July 2014. <http://en.wikipedia.org/wiki/CUDA>.

12.) Gookin, Dan. *C All-in-one Desk Reference for Dummies*. Hoboken, NJ: Wiley Pub., 2004. Print.

For more information about the program visit: http://fmri-ret.eng.usf.edu/. The Research Experience for Teachers (RET) at the Functional Materials Research Institute at USF is funded by the National Science Foundation under award number 1301054.