# GPU implementation of spin glass simulation with parallel tempering

C. Thomasson[1], S. Hall[2], Y. Fang[2], S. Feng[2], A. Papke[2], and K.-M. Tam[2]

1. Department of Computer Science, Louisiana State University
2. Department of Physics and Astronomy, Louisiana State University

LA-SiGMA
Louisiana Alliance for Simulation-Guided Materials Applications

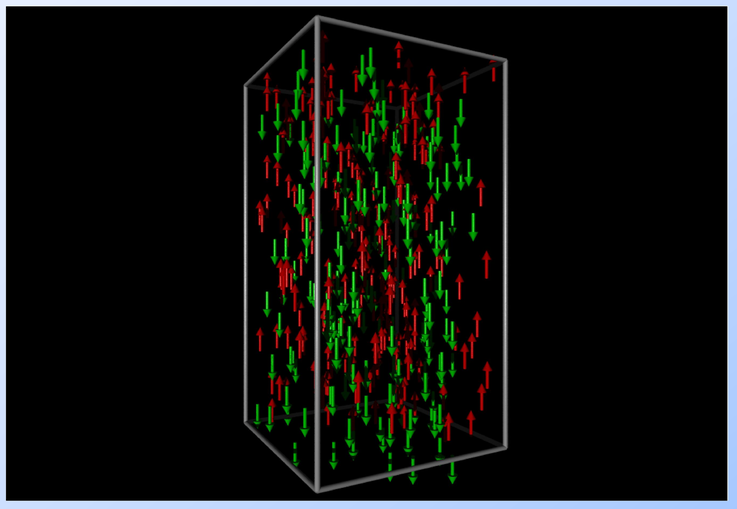
NSF

## Abstract

Most magnetic materials form magnetic ordering at low temperatures. The discovery of a metallic alloy which fails to do so has been a puzzle for the last few decades. Unfortunately, analytical calculations cannot provide a convincing answer to the problem. On the other hand, numerical Monte Carlo simulations require a long equilibration time. The parallel tempering method has been proven to be powerful method to alleviate the long equilibration time. Since the parallel tempering method is inherently parallel, this can fully utilize the architecture of the GPU's multi-threaded environment.

## Spin Glass

Magnetic materials such as Iron, Nickel, and Magnetite posses magnetic moments with a strong interaction. The interaction is predominately dominated by the electronic exchange process, which results in a short range ferromagnetic coupling. When the Temperature, and thus the thermal fluctuation is small, the magnetic moments will line up in a parallel direction. However, in spin glass materials, the interactions among magnetic moments are usually random and frustrated. This results in the random ordering pattern as shown in the figure.



K.-M. Tam and M. J. P. Gingras, arXiv:0810.0854v1

## Edwards-Anderson Model

Within the Ising model, the magnetic moments, often referred to as Spins, are coupled via the ferromagnetic coupling. The Hamiltonian which describes the energy of the model can be written as

$$H = \sum J_{i,j} s_i s_j$$

where the coupling J is a constant, usually set at -1. For the canonical spin glass model, Edwards-Anderson model, the coupling J is randomly chosen from +1 or -1.

## Monte Carlo method

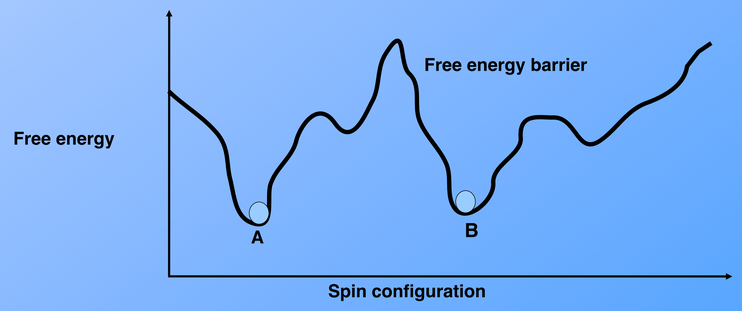
The physics of the model can be obtained by calculating the physical quantities, for example, magnetization, magnetic susceptibility, spin glass correlation length, etc, … averaged over the partition function. The partition function is given by

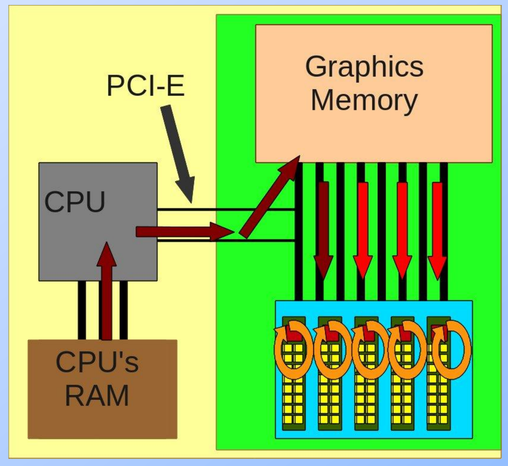
$$Z = \sum \exp\left[-\beta H(\{s\})\right]$$

The calculation of the partition function involves a summation over all the spin configurations. However, the number of spin configurations grows by $2^N$ for $N$ particles, directly performing the summation is not feasible. The Monte Carlo method is used to sample the partition function stochastically. In principle, if the number of Monte Carlo steps is large enough, reliable estimates of the physical quantities can be obtained.

## Parallel Tempering

The two essential conditions for spin glass, frustration and Disorder, present a rugged free energy landscape for different spin configurations. The Monte Carlo simulation has a very long equilibration time, especially at low temperatures. Parallel tempering is a technique which uses the idea of swapping between the spin configurations at low temperature and high temperature to shorten the equilibration time.




Traditionally used for graphics display, the GPU is also a powerful tool for computations. The GPU can be used to speed up calculations tremendously, but is lacking in memory

## GPU IMPLEMENTATION

The PTMC inherently possesses parallelization as the update of the bipartite lattice can be assigned to independent threads. The CPU initializes the joint energy J and a random configuration, then calls the GPU kernel in which one 16*16*16 3D lattice is assigned to a multiprocessor. This process has multiple replicas to fully utilize the computational power where concurrency comes from the combination of various β and J's

## COMPULATIONAL LIMITATION

1. For each spin update, the GPU reads from memory 7 times and references a pre-generated table using a random index. The arithmetic operations are relatively less complex such that the simulations bottleneck is memory access.

2. We currently fill the entire lattice into shared memory for optimization. However, this design limits the simulation scale no larger than 18*18*18. This is a trade-off between scale capability and speed.

3. The implementation needs references to a pre-generated table of random integers. Due to its irregular access pattern, a naive design of reading from the global memory is very inefficient.

## FUTURE WORKS

We aim to find the balance between calculations and memory accesses using global memory to texture memory pre-fetching, awareness of the texture cache for random table look up, and employing register or shared memory as much as possible.

## Acknowledgements

This material is based upon work supported by the National Science Foundation under award EPS-1003897

LSU