Parallelizing a micromagnetic program for use on multi-processor shared memory computers

Michael J. Donahue
National Institute of Standards and Technology
Gaithersburg, Maryland 20899–8910
Email: michael.donahue@nist.gov

***Abstract*—Parallelization of a finite difference micromagnetic program on shared memory computer systems is studied. Efficiency is found to be limited by memory bandwidth, and techniques are introduced to reduce memory traffic. Computations are sped up by a factor of three with four processor cores; a factor of five is possible on some systems. This corresponds to a Karp-Flatt serial fraction of 5-10% for small core counts.**

I. Introduction

Micromagnetic simulations are time consuming, mostly due to the expense of the long-range self-magnetostatic (demagnetization) calculation. Many efforts have been made to speed up these computations by parallelizing the code. Most of this work has been done on finite element codes [1]–[3], which, because of the techniques used to compute the demagnetization field, are both more amenable to and more in need of parallelization, but finite difference codes have also been parallelized [4], [5]. Historically most of these examples have been aimed at computer clusters rather than shared memory machines.

In recent years multi-core and multi-processor computers have become ubiquitous, so it is natural to examine the possibility of parallelizing micromagnetic codes for such shared memory architectures. Although memory bandwidth and latency in shared memory systems is faster than the node-to-node communication in computer clusters, nonetheless memory speed has not kept pace with the increase in the number of processing units. In many scientific codes performance is restrained more by main memory access speed than core arithmetic limitations. When parallelizing such codes, it is important to reduce memory accesses as much as possible. This paper details the implementation of a parallel version of the widely used OOMMF [6] finite difference micromagnetic program, with emphasis on reducing memory traffic.

II. Computation Details

The computations in micromagnetic programs divide naturally into two parts. One part computes the magnetic energy (and associated field) for a given magnetization state, and the second uses the results of the first to advance the magnetization from one state to the next. The latter (state-to-state transition computation) may implement either a dynamic simulation by integrating the Landau-Lifshitz-Gilbert equation, or else a quasi-static simulation by tracking magnetic equilibria via energy minimization. Either way, the computation time is dominated by the first part (energy and field computation), which is itself dominated by the time required to compute the self-magnetostatic field. See Table I for some representative times.

A. Computation of Local Energy Terms

The canonical energy terms in a micromagnetic calculation are exchange, magneto-crystalline anisotropy, Zeeman, and demagnetization (self-magnetostatic). The first three are local (or nearly local) in nature, and allow for a straightforward parallelization. The micromagnetic spin array is partitioned into separate blocks, and the local (non-demagnetization) energy terms are computed independently in each block, with one thread of execution per block. If the blocks are chosen small enough that the magnetization data for each block will fit into processor cache, then that data will be loaded only once for all the local energy terms.

B. Computation of Demagnetizing Energy

The non-local nature of the demagnetizing field makes parallelizing this term difficult, because a simple spatial decomposition of the spin array leads to expensive communication overhead between the partitions. The OOMMF micromagnetic program computes the demagnetization field using fast Fourier transforms (FFT's) [7], which is possible because the self-magnetostatic field can be naturally expressed as a convolution of the magnetization spin array with a fixed kernel (the interaction matrix) derived from the geometry of the simulation volume. The computation involves applying a three-dimensional FFT to the spin array, multiplying the transformed spin array by the transform of the interaction matrix, and applying an inverse three-dimensional FFT. The three-dimensional FFT is decomposed into iterated one-dimensional FFT's along each of the x-, y-, and z-directions. In the first stage, for each y and z an FFT is performed along x. Each FFT is performed on one core; in other words, the FFT's are partitioned by (y, z)-coordinate. In the second stage, the spin array is partitioned by (x, z) and FFT's are taken along y. This partitioning is effectively in x-transform space. The partitioning for the third stage is by (x, y), in xy-transform space, and the FFT's are taken along z. The multiplication of the spin array transform by the interaction matrix transform is partitioned across the full transform space. The process is then reversed to take the inverse FFT.

One performance issue arises from the naturally periodic nature of the FFT. To use an FFT to compute the convolution of non-periodic sequences, it is necessary to zero-pad a finite sequence to twice its length. For multi-dimensional convolutions, zero-padding is required in each dimension. Thus, for three-dimensional FFT's this increases the problem size by a factor of eight. If the original dimensional lengths of the input array are N_x, N_y, and N_z, and we assume that the computational cost of a one-dimensional FFT of size n is $Cn \log n$ (where C is a small constant), then the computational cost for a three-dimensional FFT of a fully zero-padded $2N_x \times 2N_y \times 2N_z$ array is $8CN_xN_yN_z \log 8N_xN_yN_z$. This is over eight times larger than the cost if zero-padding were not required.

The FFT of an entirely zero sequence is zero; this fact can be used to greatly reduce the cost of computing a multi-dimensional FFT of a zero padded array. For ease of discussion, consider first the two-dimensional schematic in Fig. 1. In a naive computation, the first stage (x-axis FFT's) consists of $2N_y$ FFT's of length $2N_x$, for a cost of $4CN_xN_y \log 2N_x$. The second stage (y-axis FFT's) would consist of $2N_x$ FFT's of length $2N_y$, for a cost of $4CN_xN_y \log 2N_y$. So the total cost would be $4CN_xN_y \log 4N_xN_y$. However, it is not necessary to compute the x-axis FFT's in the upper light gray rectangle of Fig. 1, since it is known a priori that the FFT's there will be zero. This reduces the cost of the first stage of the computation by half. If x and y are ordered so that $N_x \geq N_y$, then the total computational cost is reduced by at least 25%. Extending this method to three dimensions yields even larger savings. In that case the reduced cost is

$$2CN_xN_yN_z \log 2N_x + 4CN_xN_yN_z \log 2N_y + 8CN_xN_yN_z \log 2N_z.$$

If the axes are ordered such that $N_x \geq N_y \geq N_z$, then the savings is a minimum of 41.6%, which occurs when $N_x = N_y = N_z$. The savings can be significantly larger for flat plate or long needle geometries. For the 500 x 250 x 8 geometry used in the tests below, the savings is about 52%.

Of course, not only is there no need to compute an FFT of a known zero sequence, there is also no need to read it from or write it to memory. The savings in memory traffic is comparable to the computational savings, and is of primary importance for efficient parallelization. Moreover, if N_z is small enough that the z-axis FFT's fit into processor cache, then the memory loads and stores surrounding the multiplication of the spin array and interaction matrix transforms can be eliminated. Instead of computing the forward z-axis FFT's, multiplication of transforms, and inverse z-axis FFT's as three separate stages, each individual z-axis FFT can be immediately followed by the multiplication of transforms for that line, and then the inverse z-axis FFT can be performed. If this "inner" portion of the demagnetization computation fits into cache, then a significant savings in memory traffic is achieved. Part of the savings is because the working memory set size for the spin array transform drops from $8N_xN_yN_z$ to $4N_xN_yN_z$. (The "upper-z half" of the spin array transform is only held temporarily in cache, and is not stored in main memory.)

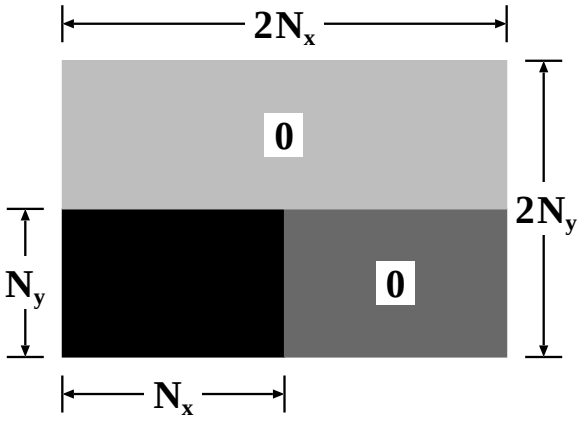

Fig. 1. Schematic illustrating efficient FFT computation with large scale zero-padding. In the x-axis FFT's, the black and dark gray regions are computed, whereas the light gray region may be ignored. For the subsequent y-axis FFT pass, the entire rectangle must be processed.

The results presented in this paper were obtained using an FFT algorithm coded specifically for this application. However, the techniques presented above for reducing memory traffic can be applied using any one dimensional FFT algorithm, such as one from a system or vendor library.

III. Computation Results

To test the effects of the above techniques, a 2000 nm $\times$ 1000 nm $\times$ 32 nm magnetic slab was simulated, with 4 nm cubic cells. This system consists of one million spins, which appears to be a reasonable benchmark size. As noted by Gustafson [8], parallel efficiency generally improves with increasing problem size, but that effect is not studied in this work. The energy terms included exchange, cubic anisotropy, Zeeman, and demagnetization. Two hardware configurations were tested: a 3 GHz two-way quad-core Intel Xeon X5365 machine (8 cores total) running Mac OS X 10.5.6, and a 2.6 GHz eight-way dual-core AMD Opteron 885 machine (16 cores total) running Linux x86_64 2.6.27. The Xeon machine has a single memory bus shared by all cores; the Opteron has separate memory for each processor, and high-speed interconnects between the processors. The somewhat higher total memory bandwidth of the latter system is evident in the results (Table I), which show the Opteron host, with slower processors, running slightly faster than the Xeon host when eight cores runs are compared.

Table I shows the component and total computational time per field evaluation (equivalently, time per state-to-state transition) as a function of the number of processor cores. The "Non-demag" column is the time for all the local energy terms. The "Demag-inner" piece, described in the previous section, is a component of the "Demag Total" time. The "Step Total" column includes all the energy computation times and the overhead time for a conjugate gradient energy minimization solver. Normalized speed-up curves for these data are presented in Figs. 2 and 3. The dashed lines in these

TABLE I
TOTAL AND COMPONENT ELAPSED (WALL) TIME IN SECONDS PER FIELD EVALUATION FOR A ONE MILLION CELL MICROMAGNETIC SIMULATION.

Intel Xeon X5365, 3 GHz				
Cores	Non-demag	Demag-inner	Demag Total	Step Total
1	0.096	0.199	0.591	0.826
2	0.051	0.109	0.318	0.453
3	0.036	0.078	0.230	0.334
4	0.029	0.062	0.188	0.277
5	0.028	0.052	0.176	0.261
6	0.026	0.048	0.172	0.253
7	0.025	0.045	0.172	0.252
8	0.023	0.044	0.171	0.248

AMD Opteron 885, 2.6 GHz				
Cores	Non-demag	Demag-inner	Demag Total	Step Total
1	0.135	0.300	0.852	1.154
2	0.069	0.157	0.439	0.616
3	0.048	0.109	0.310	0.441
4	0.037	0.086	0.249	0.360
5	0.033	0.068	0.211	0.317
6	0.027	0.059	0.175	0.272
7	0.025	0.054	0.166	0.261
8	0.023	0.051	0.149	0.239
9	0.024	0.047	0.145	0.239
10	0.023	0.043	0.137	0.228
11	0.023	0.040	0.133	0.225
12	0.023	0.040	0.126	0.217
13	0.023	0.037	0.125	0.217
14	0.023	0.038	0.124	0.217
15	0.023	0.038	0.127	0.221
16	0.023	0.039	0.125	0.218

figures represent rough fits to Amdahl's law [9], which states that an algorithm with proportion P that can be parallelized, and proportion $(1-P)$ that can't, will show a speed improvement on n processors of $1/((1-P)+P/n)$. The quoted serial fraction is essentially the Karp-Flatt metric [10], which corresponds to $1-P$ in Amdahl's law.

In both cases, we see that the "Demag inner" portion of the computation, which in the single core case accounts for approximately one fourth of the total processing time, scales well—through all eight cores on the Xeon host and up through twelve cores on the Opteron machine. The other components don't scale quite as well, but still there is a total factor of three speed-up with four cores on the Intel Xeon host, and a factor of five speed-up with ten cores on the AMD Opteron host. In these tests the non-energy computation portion of the state-to-state transition has not been parallelized. In the single core case it represents a relatively small fraction of the total compute time, but the importance of this term grows as more cores are added. In principle, the parallelization approach used for the non-demagnetization energy terms should be applicable to this component as well.

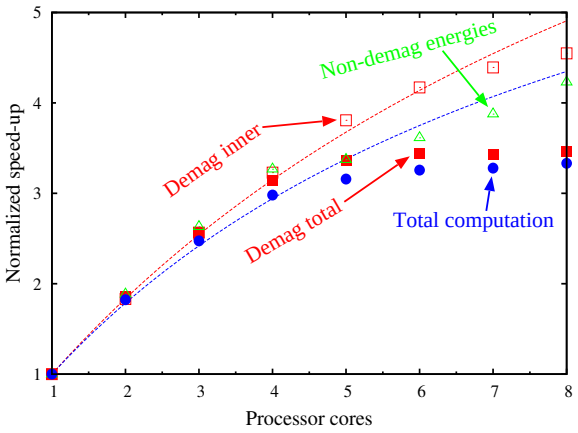

Fig. 2. Relative speed-up on Xeon host for component terms demag inner (open squares), total demag (closed squares), local energies (open triangles), and step total (closed circles). Dashed lines are rough fits to Amdahl's law; the upper curve has 9% serial fraction, the lower curve has 12% serial fraction.

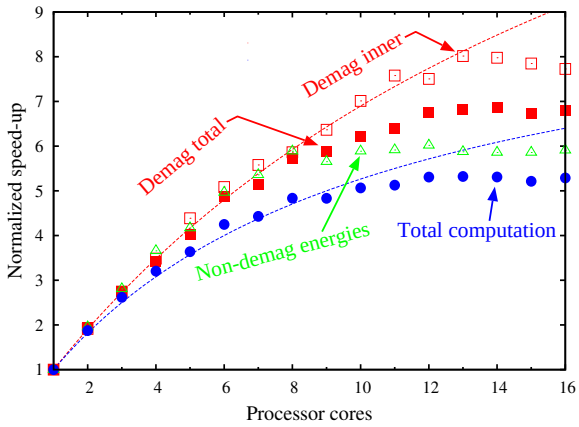

Fig. 3. Relative speed-up on Opteron host for component terms demag inner (open squares), total demag (closed squares), local energies (open triangles), and step total (closed circles). Dashed lines are rough fits to Amdahl's law; the upper curve has 5% serial fraction, the lower curve has 10% serial fraction.

DISCLAIMER

The mention of specific products, trademarks, or brand names is for purposes of identification only. Such mention is not to be interpreted in any way as an endorsement or certification of such products or brands by the National Institute of Standards and Technology. All trademarks mentioned herein belong to their respective owners.

REFERENCES

[1] B. Yang and D. Fredkin, "Dynamical micromagnetics by the finite element method," *IEEE Trans. Magn.*, vol. 34, pp. 3842–3852, 1998.

[2] W. Scholz, J. Fidler, T. Schrefl, D. Suess, R. Dittrich, H. Forster, and V. Tsiantos, "Scalable parallel micromagnetic solvers for magnetic nanostructures," *Comp. Mat. Sci.*, vol. 28, pp. 366–383, 2003.

[3] K. Takano, E.-A. Salhi, M. Sakai, and M. Dovek, "Write head analysis by using a parallel micromagnetic FEM," *IEEE Trans. Magn.*, vol. 41, pp. 2911–2913, 2005.

[4] R. C. Giles, P. R. Kotiuga, and M. Mansuripur, "Parallel micromagnetic simulations using Fourier methods on a regular hexagonal lattice," *IEEE Trans. Magn.*, vol. 27, pp. 3815–3818, 1991.

[5] Y. Kanai, M. Saiki, K. Hirasawa, T. Tsukamomo, and K. Yoshida, "Landau-Lifshitz-Gilbert micromagnetic analysis of single-pole-type write head for perpendicular magnetic recording using full-FFT program on a PC cluster system," *IEEE Trans. Magn.*, vol. 44, pp. 1602–1605, 2008.

[6] M. J. Donahue and D. G. Porter, *OOMMF User's Guide, Version 1.0*, Interagency Report NISTIR 6376, National Institute of Standards and Technology, Gaithersburg, MD (Sept 1999).

[7] T. G. Stockham, "High speed convolution and correlation," *Joint Computer Conference Proceedings*, vol. 28, pp. 229–233, 1966.

[8] J. L. Gustafson, "Reevaluating Amdahl's law," *Communications of the ACM*, vol. 31, pp. 532-533, 1988.

[9] G. M. Amdahl, "Validity of the single-processor approach to achieving large-scale computing capabilities," in *Proc. Am. Federation of Information Processing Societies Conf.*, vol. 30, AFIPS Press, Reston, Va., 1967, pp. 483–485.

[10] A. H. Karp and H. P. Flatt, "Measuring Parallel Processor Performance," *Communication of the ACM*, vol. 33, pp. 539–543, 1990.