LBL-32512
UC-404

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Materials Sciences Division

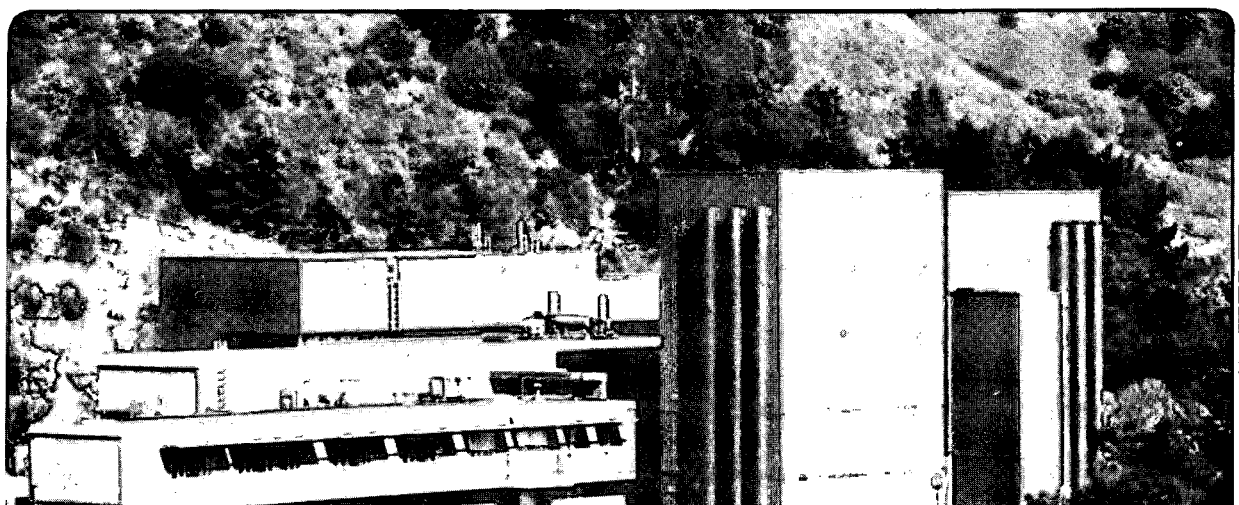
National Center for Electron Microscopy

Presented at the Materials Research Society Symposium,
San Francisco, CA, April 26–May 1, 1992,
and to be published in the Proceedings

Monte Carlo Simulations on SIMD Computer Architectures

C.P. Burmester, L.T. Wille, and R. Gronsky

March 1992



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

LBL-32512

Monte Carlo Simulations on SIMD Computer Architectures

C.P. Burmester, L.T. Wille*, & R. Gronsky

Materials Science Division
National Center for Electron Microscopy
Lawrence Berkeley Laboratory
University of California, Berkeley, CA 94720

*Florida Atlantic University
Dept. of Physics
Boca Rato, FL 33431

Materials Research Society Symp., San Francisco, 4/26-5/1/92

This work was supported in part by the Director, Office of Energy Research, Office of Basic Energy Sciences, Materials Science Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098, and by an Internal Research Grant from the Division of Sponsored Research at Florida Atlantic University.

Monte Carlo Simulations on SIMD Computer Architectures

C. P. Burmester[1], L. T. Wille[2], and R. Gronsky[1]

[1]Department of Materials Science and Mineral Engineering, University of California at Berkeley, CA; and Materials Science Division, Lawrence Berkeley Laboratory, Berkeley, CA 94720.

[2]Department of Physics, Florida Atlantic University, Boca Raton, FL 33431.

ABSTRACT

Algorithmic considerations regarding the implementation of various materials science applications of the Monte Carlo technique to single instruction multiple data (SIMD) computer architectures are presented. In particular, implementation of the Ising model with nearest, next nearest, and long range screened Coulomb interactions on the SIMD architecture MasPar MP-1 (DEC mpp-12000) series of massively parallel computers is demonstrated. Methods of code development which optimize processor array use and minimize inter-processor communication are presented including lattice partitioning and the use of processor array spanning tree structures for data reduction. Both geometric and algorithmic parallel approaches are utilized. Benchmarks in terms of Monte Carlo updates per second for the MasPar architecture are presented and compared to values reported in the literature from comparable studies on other architectures.

INTRODUCTION

Computational materials science, fast becoming a standard tool in materials research, is primarily limited in its ability to model systems of macroscopic size over realistic time scales by computer processing speed. While major advances have been made in processor speed with serial computers through design and miniaturization, further processor enhancement can achieve relatively minor gains before encountering the limits imposed by atomic scale and light-speed information transfer. Advances in parallel architectures involving distributed computation over many processors operating in parallel offer a means to achieve many orders of magnitude increased processing power with existing computer technology. Of the types of parallel architectures available, Single Instruction Multiple Data, or "SIMD", architectures are particularly applicable to materials science simulations. Many materials science simulation techniques operate on systems which involve multiple data points under the influence of a single global equation of state and, hence, are ideal candidates for data parallelism.

MasPar ARCHITECTURE OVERVIEW

The work presented here was implemented on the MasPar 1 series (DEC mpp-12000) massively parallel SIMD architecture computer[1]. On such architectures, a serial stream of program instructions is broadcast to all parallel processing elements which execute the instructions synchronously on local multiple data. The MasPar MP-1 series computers are scalable from one thousand to sixteen thousand processing elements. These processing elements are organized into a two dimensional mesh with toroidal boundary conditions. Each processing element is directly connected to its four nearest and four next-nearest neighbors forming an "X"-network of directional synchronous inter-processor communications with a bandwidth of 23,000 MBytes per second. Longer range and arbitrary direction communications are implemented by means of a global router having a bandwidth of 1,300 MBytes per second. Parallel instructions are broadcast by the Array Control Unit (ACU), which consists of a 14 MIPS control processor, thirty-two 32-bit registers, and 1 MByte of RAM. Each processing element consists of a 1.8 MIPS 4-bit control processor, forty 32-bit registers, and 64 KBytes of RAM. The complete MasPar system consists of the Data Parallel Unit (DPU), described above, and a DEC VAXstation 5200 running Ultrix 3.1 and DECwindows used as a "front end" interface. The codes presented were developed in MPL[2], a massively parallel extension of the Kernighan and Ritchie C programming language. Many of the standard UNIX programming functions have also been provided in parallel implementations.

MONTE CARLO SIMULATION IN MATERIALS SCIENCE

The Monte Carlo simulation technique was first introduced by Metropolis et al. to study the equilibrium properties of model systems[3]. However, this technique has since proven to be widely applicable to the study of both equilibrium and dynamic properties of many diverse materials systems. The classical Monte Carlo technique generates a sequence of microstates, $[\Omega]$, in an ergodic fashion such that thermodynamic averages over the "time" sequence of microstates tend asymptotically to the average over the true equilibrium ensemble as the sequence or "chain" length increases. Such a chain of microstates is generated in practice by considering candidate excitations of the system leading to a trial transition from some microstate Ω_i to Ω_j and accepting such microstate transitions with a probability

$$\Pi(\Omega_i \rightarrow \Omega_j) = D^* \frac{P(\Omega_j)}{P(\Omega_i)} \tag{1}$$

where D^* is the degeneracy ratio of the microstates and $P(\Omega_i)$ is the canonical density function

$$P(\Omega_i) = Z^{-1} exp\,[-H(\Omega_i)/k_B T]. \tag{2}$$

Here, $H(\Omega_i)$ is the Hamiltonian for the system specifying the configurational energy, T is the absolute temperature, k_B is the Boltzmann constant, and Z is the canonical partition function which, for a system involving discrete states (e.g. a lattice based simulation), takes the form

$$Z = \sum_{[\Omega]} exp\,[-H(\Omega_i)/k_B T]. \tag{3}$$

An equilibrium value of a thermodynamic quantity, f, is obtained by a weighted average over the " time" sequence of microstates

$$\langle f \rangle = \sum_{[\Omega]} f(\Omega_i)\, exp\,[-H(\Omega_i)/k_B T]. \tag{4}$$

Monte Carlo simulations of different systems are distinguished by the selection of two factors: (1) Hamilitonian, and (2) static lattice structure and boundary conditions. The Hamiltonian specifies the energy of the system as a function of all the degrees of freedom of the system and quantifies the range and nature of the interactions. The lattice symmetry and boundary conditions determine the possible structural transformations by group-subgroup relationships and the nature and existence of free surfaces and interfaces. Perhaps one of the most widely used system models is the spin-½ Ising model. It has been applied to a broad variety of problems including tumor growth[4], galaxy formation[5], binary alloys[6], and catalysis[7]. The most basic form of the Ising model exhibiting phase transitions is the two-dimensional nearest-neighbor square lattice model. The Hamilitonian for this system is given by

$$H = \tfrac{1}{2} \sum_{<ij>} V \sigma_i \sigma_j \tag{5}$$

where V is the pair interaction energy and $\sigma = \pm 1$ represents a two-state site occupation. The Ising model can be extended to handle more complicated systems by including longer range interactions. One example of this is the nearest and anisotropic next-nearest neighbor Ising model proposed by Wille et al.[8] to model oxygen ordering in the basal plane of $YBa_2Cu_3O_{7-\delta}$

$$H = \tfrac{1}{2} \sum_{<NN>} V_1 \sigma_i \sigma_j + \tfrac{1}{2} \sum_{<NNN\,\dagger>} V_2 \sigma_i \sigma_j + \tfrac{1}{2} \sum_{<NNN>} V_3 \sigma_i \sigma_j \tag{6}$$

where the † is taken over next-nearest neighbors separated by a copper cation. Even further range interactions can be considered as in the case of the screened Coulomb potential used to explain observed oxygen superstructures in this system[9]; important to the understanding of detailed dependence of critical temperature on basal plane oxygen content. While the underlying model and lattice can remain the same, the choice of Hamilitonian dictates the approach in developing the most efficient parallel algorithm. In what follows, general methods of developing parallel algorithms are applied to the Monte Carlo simulation of the Ising model for each of the above three cases.

PARALLEL MONTE CARLO - FUNDAMENTAL CONSIDERATIONS

The central issue in constructing a correct parallel Monte Carlo algorithm is to distribute the calculation in the most efficient way amongst the processors without violating the detailed balance criterion

$$P(\Omega_i)\Pi(\Omega_i \rightarrow \Omega_j) = P(\Omega_j)\Pi(\Omega_j \rightarrow \Omega_i). \quad (7)$$

which ensures that simulation time averages will converge to ensemble averages in the limit of long times. In a serial algorithm, this is achieved in a straightforward way by accepting each trial update with the appropriate probability, typically, as outlined by Metropolis[3], et al.

$$\Pi_{ij} = D^* exp[-\Delta E_{ij}/k_B T]. \quad (8)$$

where ΔE_{ij} is the difference in configurational energy between the two microstates, i and j. However, for parallel Monte Carlo algorithms, an additional constraint must be satisfied, namely, that all parallel updates are spatially independent. A set of spatially independent sites is defined as a set of sites which do not interact with each other through the action of the system Hamiltonian. For example, in the two dimensional nearest neighbor Ising model, each site interacts with its four nearest neighbors. Updating nearest neighbors in parallel leads to the creation of a chain with non-ergodic sequences in violation of the detailed balance condition. This can be understood qualitatively by considering a one dimensional chain of nearest neighbor interacting Ising spins whose ground state is ferromagnetically aligned. During an update, a site will tend to adopt the alignment of its neighbors. At some finite temperature, adjacent sites which are anti-ferromagnetically ordered will be created. If updates are executed in parallel on all sites, sites in anti-ferromagnetic sequences will oscillate between spin-up and spin-down states and never achieve an aligned ground state. This "blinking-state" or "parallel-resonance" condition has the effect of confining the system to an artificial and limited region of phase space and may lead to invalid statistics[10].

For systems which involve relatively short-range interaction, as in (5) and (6), the most effective method of parallel implementation which avoids the parallel-resonance condition is *geometric decomposition*[11-13]. In this scheme, the system is divided into maximal sets of spatially independent sites which are updated in parallel. In the case of the nearest neighbor Ising model, this amounts to a " checkerboard" decomposition where every other site in each lattice direction is updated simultaneously. Geometric decomposition has the advantage of involving a nearly direct mapping of the algorithm to the parallel array and is thus simple to implement. In the limit where system interactions extend to the size of the system, each site interacts with all others and a geometric decomposition would degrade to a serial algorithm. In such cases, an *algorithmic decomposition* scheme is employed[12,13]. In this approach, the parallel aspects of the algorithm are distributed to all the processors. For instance, in the calculation of a Coulomb interaction, each site in the parallel array determines its contribution to the energy of interaction with a given site simultaneously. In this fashion, a calculation involving $O(N^2)$ steps is reduced to one involving $O(\log_2 N)$ steps.

PARALLEL MONTE CARLO - BASIC ALGORITHMS

A first-approach implementation of the nearest neighbor Ising model is straightforward on the MP-1 architecture. As the processor array is organized as a two dimensional mesh, a direct mapping of system sites to processors is made. For a 16*k* processor machine, this gives rise to a system of 128 by 128 sites. The periodic boundary conditions are built into the hardware. Each site is given a "parity" to correspond to the checkerboard decomposition. At each cycle, all sites of the same parity are updated in parallel. This is accomplished by use of a plural *if-then* test. At any one time, the MasPar maintains a current "active set" of processors which receive and execute plural instructions. The action of a plural *if-then* test is to set all processors whose local data evaluate the plural test to *TRUE* to *active* status and all others to *inactive* status for the instructions contained in the range of the logical test. Plural *while, do,* and, *for* loops are implemented in an analogous fashion. Once one set of equal parity sites have completed their update, the sites of opposite parity are updated. The acceptance of site updates with probabilities consistent with (8) is facilitated in the MasPar environment by the availability of parallel implementations of standard UNIX library functions including, in this case, a routine which returns separate streams of random deviates on all active processors. A test series of simulations for the parallel algorithm was performed with the nearest-neighbor zero field Ising model for which exact analytical results are available. For a system size of 128 by 64 sites, a temperature versus specific heat plot [Figure 1] indicates that the critical temperature occurs at $2.28\ k_B T/V$ for $V = 1.0$, consistent in both value and form with the exact solution by Onsager[14].

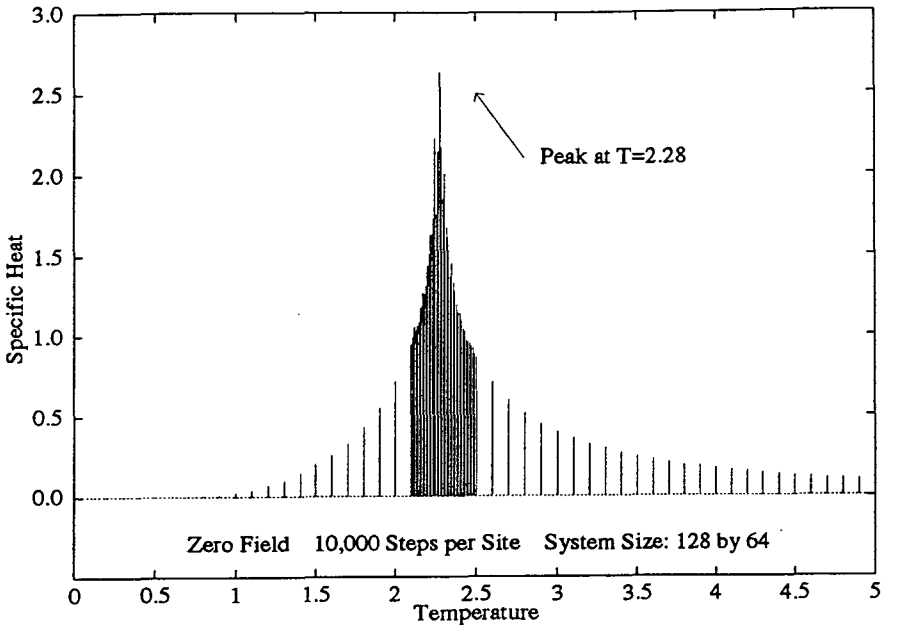


Figure 1: Specific heat versus temperature plot produced with the parallel nearest neighbor Ising model Monte Carlo simulation. Each vertical line represents the value obtained from one simulation run over 10,000 Monte Carlo steps per site after equilibrium had been obtained as determined by course grain energy averages. The critical point for this system size is at 2.28 $k_B T/V$.

In a standard Monte Carlo study, the system size is typically varied to elucidate the effect on the critical values. Simulating system sizes smaller than the processor array is accomplished by running on a subset of the available processors. Simulating system sizes larger than the available processors is accomplished by partitioning the larger lattice onto the smaller processor array. Partitioning is also useful in increasing the efficiency of the overall algorithm. In the checkerboard decomposition, at least half of the processors are idle at any given time. While this could constitute an acceptable use of the processor array, the number of idle processors increases rapidly with the extent of inter-site interactions such that a simulation using (6) with nearest and next nearest neighbor interactions would leave one fourth of the processors idle. In addition, it is generally true that inter-processor communications are much slower than intra-processor memory accesses. For instance, on the MasPar MP-1 an inter-processor or "xnet[1]" read expends 80 DPU clock ticks while an intra-processor or register read expends only 14 clock ticks. Thus, the more communications necessary to calculate interactions that are managed within a processor, the more efficient the overall algorithm. Partitioning is accomplished by declaring sub-arrays of sites on each processor. During each step, every processor updates the equivalent site in its local partition in parallel. If the size of the partition is such that it contains the range of interactions for one site, then all processors can operate in each cycle. The increase in performance obtained by partitioning is illustrated in Figure 2. Beyond partitions of 16 sites, the algorithm performance approaches intrinsic processor element speed. In a recent report of a worldline quantum Monte Carlo implementation on the Connection Machine[15], Somsky and Gubernatis report a similar increase in algorithm performance with "Virtual Processor Ratio" (VPR), a function on the CM-2 which allows one processor to simulate the function of many. They found an optimal increase in speed using a VPR of 16, analogous to the value found here, illustrating the general nature of partition optimization. While partition optimization leads to a more efficient use of the machine, it is worth noting that while an individual simulation step will take less time, this is accomplished with a concurrent increase in the number of sites resulting in longer overall run times. Thus partitioning results in a real-time savings only when there is a need to simulate system sizes larger that the physical processor array. The nearest and anisotropic next nearest neighbor Ising model (6) was implemented using a partitioning scheme. Example results from the parallel algorithm correctly identify the transitions from the Orthorhombic to cell-doubled Orthorhombic to Tetragonal oxygen ordered structures associated with the loss of superconductivity in the high-T_c $YBa_2Cu_3O_{7-\delta}$ [Figure 3].

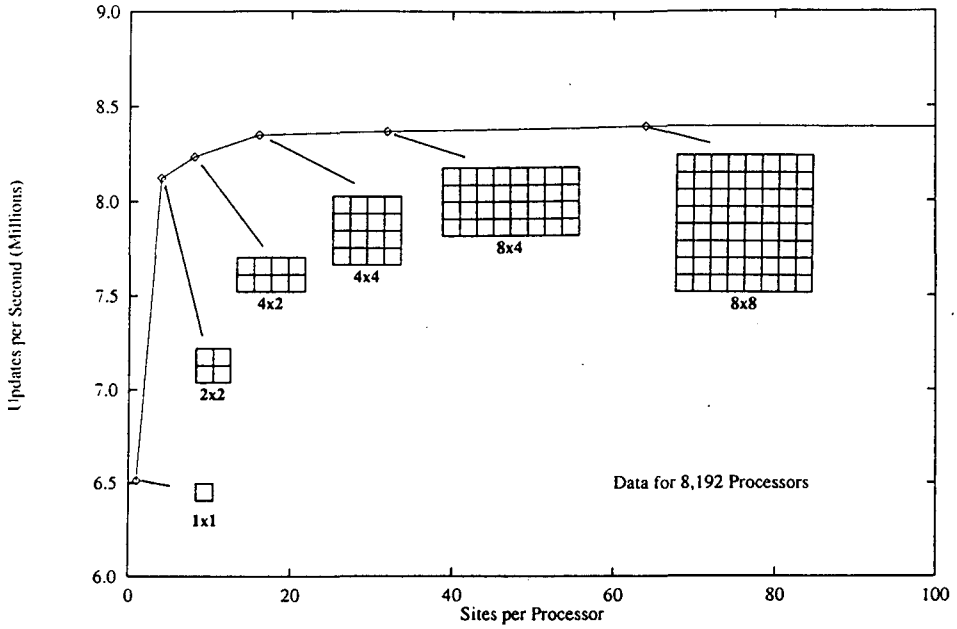


Figure 2: Illustration of partition optimization. Monte Carlo updates per second versus sites per processor for 8,192 processors. Performance increases dramatically at first and then asymptotically approaches the intrinsic processing element speed.

The ultimate bottleneck in any Monte Carlo simulation arises from the need to extract thermodynamic information from the whole processor array at each lattice update to allow the calculation of thermodynamic averages. Extracting information on a site by site basis would involve $O(N)$ steps for each lattice update. However, a general method for *data reduction* over parallel processors utilizes a binary tree reduction method which spans the processor array involving $O(\log_2 N)$ steps to extract information. This optimization leads to a critical increase in speed for normal Monte Carlo system sizes, for example; an increase of over 600 times is obtained over the discrete sum for 8,192 sites. A binary tree approach can be used to efficiently implement any general operation over the whole array and is thus useful for lattice spanning operations. The binary tree reduction method is also useful in *algorithm decomposition* parallel implementations needed when the extent of site interactions make it inefficient to use geometric decomposition methods, as is the case with a screened Coulomb Hamilitonian.

As Monte Carlo simulations are in general concerned with the calculation of critical points, the phenomenon of critical slowing down is of pivotal importance. Recently, Swendsen and Wang proposed a physical cluster update algorithm which dramatically reduces the effect of critical slowing down[16] and thus allows a faster determination of critical points. The central feature of this algorithm is the determination of physical clusters which are then updated in parallel in a way which produces an ergodic sequence that is consistent with the Metropolis algorithm thus leading to valid Monte Carlo statistics. The process of cluster identification typically involves $O(N)$ steps in a serial algorithm. Again, the time this process takes can be reduced in a parallel algorithm to $O(log_2 N)$ steps. While many parallel algorithms for cluster identification have been developed[11,12,17-19], one of the most general if not completely optimal methods is that of "label-diffusion". In this method, all sites are initially given a unique label; in the case of the MasPar, the unique processor number. At each cycle, sites determine in parallel whether they belong to the same cluster as their neighbors. If so, the lowest label of the neighbor pair "diffuses" and supersedes the higher label. This processes continues until no further label diffusions occur. At this point, all sites belonging to the same cluster are labeled in an identical fashion and can be updated in parallel.

BENCHMARKS

Timing of the parallel implementation of the nearest neighbor Ising code was performed using a geometric decomposition scheme, a partition size of 4 sites per processor, and a system lattice of 256 by 256 sites. On the MP-1216 with 16K processors, this algorithm achieved 15.5 Million Monte Carlo

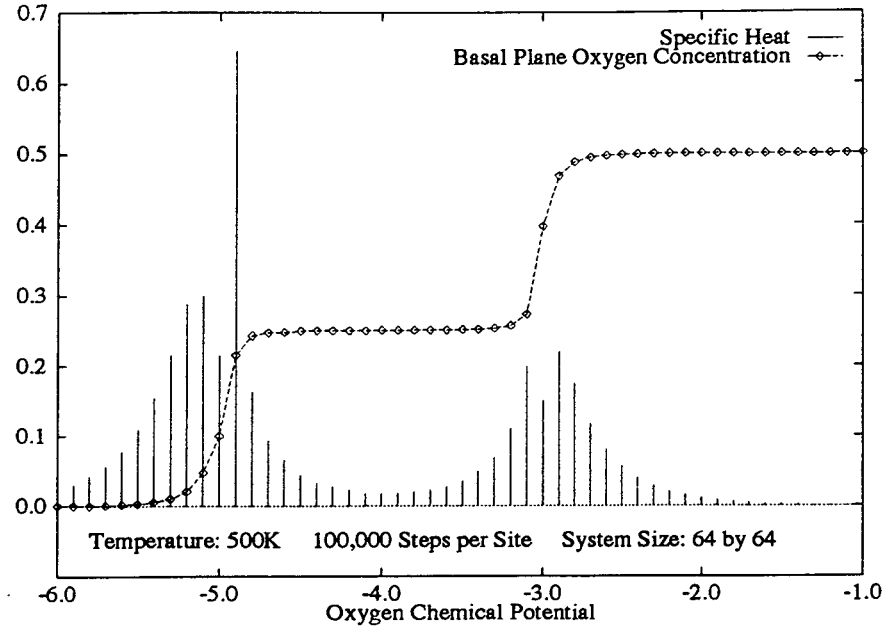


Figure 3: Plot of specific heat and basal plane oxygen concentration versus oxygen partial pressure in the $YBa_2Cu_3O_{7-\delta}$ system at a constant temperature of approximately $540K$ obtained with a geometric partitioned parallel decomposition algorithm. Peaks in the specific heat reveal the location of transitions from the single cell to cell-doubled orthorhombic phases and the cell-doubled to tetragonal phases consistent with experimental and earlier serial Monte Carlo results.

Updates per Second (MMUPS). This value compares well with performance reported[15] for simple nearest neighbor Monte Carlo algorithms on other parallel architectures with the same simulation lattice size and, if applicable, same partition size: Connection Machine CM-2 (64K processors, VPR = 4), 17.1 MMUPS; Cray Y-MP, 6.3 MMUPS; Cray X-MP, 4.6 MMUPS.

ACKNOWLEDGEMENTS

The work at Berkeley is supported by the Director, Office of Basic Energy Sciences, Materials Sciences Division of the U. S. Department of Energy under Contract Number DE-AC03-76SF00098. L. T. W. is supported by an Internal Research Grant from the Division of Sponsored Research at Florida Atlantic University. The authors would like to thank MasPar Inc., Sunnyvale, CA for access to $8k$ and $16k$ processor machines used in developing and evaluating codes used in this study. Special thanks are due to D. Rector for technical information and J. D. Becher for assistance in algorithm optimization.

REFERENCES

1. T. Blank, *Proceedings of the CompCon Spring 1990, 35th IEEE Computer Society International Meeting* (February 1990) 20.
2. J. D. Becher and K. M. Hansen, *MasPar Computer Corporation Technical Report* **TR001.1091** (November 1991).
3. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *J. Chem. Phys.* **21** (1953) 1087.
4. D. Chowdhury and D. Stauffer, *J. Stat. Phys.* **59** (1990) 1019.
5. L. Schulman and P. E. Seiden, *Science* **233** (1986) 425.
6. M. K. Phani, J. L. Lebowitz, M. H. Kalos and C. C. Tsai, *Phys. Rev. Lett.* **42** (1979) 577.
7. K. Binder and D. P. Landau, *Surf. Sci.* **61** (1976) 577.
8. L. T. Wille, A. Berera, and D. de Fontaine, *Phys. Rev. Lett.* **60**, (1988) 1065.
9. D. Adelman, C. P. Burmester, L. T. Wille, P. A. Sterne, and R. Gronsky, *submitted.*
10. D. P. Landau and D. Stauffer, *Journal de Physique* **50** (1989) 509.
11. D. W. Heermann and A. W. Burkitt, *Parallel Computing* **13** (1990) 345.
12. D. W. Heermann and A. W. Burkitt, *Parallel Algorithms in Computational Science,* Springer Series in Information Sciences, **24**, Springer-Verlag, New York (1990).
13. J. F. W. Slaets and G. Travieso, *Comp. Phys. Comm.* **56** (1989) 63.
14. L. Onsager, *Phys. Rev.* **65** (1944) 117.
15. W. R. Somsky and J. E. Gubernatis, *Comp. in Phys.* **6** (1992) 178.
16. R. H. Swendsen and J. Wang, *Phys. Rev. Lett.* **57** (1986) 2607.
17. R. Dewar and C. K. Harris, *J. Phys. A: Math. Gen.* **20** (1987) 985.
18. R. C. Brower, P. Tamayo, and B. York, *Jour. of Stat. Phys.* **63** (1991) 73.
19. F. V. Babalievski, *Comp. Phys. Comm.* **67** (1992) 453.

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
TECHNICAL INFORMATION DEPARTMENT
BERKELEY, CALIFORNIA 94720