Utilizing Hierarchical Clusters in the Design of Effective and Efficient Parallel Simulations of 2-D and 3-D Ising Spin Models

Gayathri Muthukrishnan

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

May 12, 2004
Blacksburg, Virginia

Keywords: Ising model, hierarchical clusters, LogP model, parallel computing, performance analysis & prediction

Utilizing Hierarchical Clusters in the Design of Effective and Efficient Parallel Simulations of 2-D and 3-D Ising Spin Models

Gayathri Muthukrishnan

(Abstract)

In this work, we design parallel Monte Carlo algorithms for the Ising spin model on a hierarchical cluster. A hierarchical cluster can be considered as a cluster of homogeneous nodes which are partitioned into multiple supernodes such that communication across homogenous clusters is represented by a supernode topological network. We consider different data layouts and provide equations for choosing the best data layout under such a network paradigm. We show that the data layouts designed for a homogeneous cluster will not yield results as good as layouts designed for a hierarchical cluster. We derive theoretical results on the performance of the algorithms on a modified version of the LogP model that represents such tiered networking, and present simulation results to analyze the utility of the theoretical design and analysis. Furthermore, we consider the 3-D Ising model and design parallel algorithms for sweep spin selection on both homogeneous and hierarchical clusters. We also discuss the simulation of hierarchical clusters on a homogeneous set of machines, and the efficient implementation of the parallel Ising model on such clusters.

Acknowledgements

I am grateful to my advisor and committee chair, Dr. Eunice Santos, for her numerous suggestions, guidance and support throughout the work of this thesis. I also thank Dr. Mark Jones and Dr. Calvin Ribbens for their valuable time and guidance, and for serving as members on the committee.

I thank the laboratory for computation, information and distributed processing(LCID) in the department of Computer Science at Virginia Tech for providing me with the resources needed for producing the results in this work.

The results in this thesis were funded by the National Science Foundation, Army Research Office and graduate assistantships from the Virginia Polytechnic Institute and State University.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer simulations have enabled the solution of a wide range of problems that would otherwise have remained inaccessible, and have also led to new insights and new lines of enquiry. This progress has, of course, been closely associated with developments in computer technology, which have brought about many-fold increases in computer speed, memory size and flexibility. With the growing widespread use of parallel computer systems, many new possibilities have come to the forefront, making parallelization a viable approach to obtain results in an efficient manner.

The ease of interconnecting a set of computers to form a homogeneous cluster has further necessitated the study of efficient parallelization of algorithms to yield faster results to large scale applications. Moreover, with the variations in existing resources and networks, and the need to leverage resources, it is not unusual to utilize groups of homogeneous clusters (sub-networks) that are connected via a high-latency network [3, 13]. As such, it is an important and interesting problem to focus on efficient parallelization of applications on heterogeneous clusters. Hence, in this thesis, we leverage the use of hierarchical clusters, a prevalent form of heterogeneous clusters, to design effective and efficient parallel simulations of the Ising model, which is an important spin model of magnetism.

Spin models of magnetism are used as the basis for most of the discussion of simulation in statistical physics, for a number of reasons:

- They are relevant to a variety of other systems linked by the theory of critical phenomena and phase transitions, e.g., thermodynamics and quantum field theory.

- They form an interesting application for parallel computation. Much work has been done on parallel computers (and has helped drive their development).

- They are simple models of real physical systems.

The Ising spin model is a particular spin model that plays an important role due to its relative simplicity on one hand and the richness of its physical properties on the other hand[17]. The Ising spin model is widely used in statistical physics to study the properties and dynamic behaviors of magnetic particles in a magnetic field or a heat bath.

Simulating the Ising model on a computer is a fundamental problem in computational science. The system size required in such simulations is dictated by the corresponding temperature and applied field, and in cases where finite-size effects are important (e.g., near phase transitions), it is convenient to be able to simulate relatively large systems over long time scales[26]. Moreover, parallelizing the simulation is desired due to its large-scale nature. Many applications are data parallel in nature and involve computation and communication involving multi-dimensional arrays of data. The Ising model is one such interesting problem to explore whose results can be mapped to many other data parallel applications.

Efficient parallelization of the model on a homogeneous set of computers has been studied in many places including recent work in [26]. In this work, we study the efficient simulation of the Ising model on hierarchical clusters.

In order to design realistic and efficient simulations, it is vital that a realistic, general, yet easily-usable model for hierarchical clusters be used. We note that the major issue in hierarchical clustering is the differences in network latencies. As such, a model which realistically models homogeneous clusters could be modified to model hierarchical clusters. The LogP model[12] is a general, realistic and practical parallel machine model that models the important characteristics of a homogeneous cluster of computers. Hence, we utilize a modified version of the LogP model in our design and analysis.

Next, the three-dimensional Ising model is important as real systems are three-dimensional and has been studied in many places including [9, 2]. Therefore, the need to design simulations for 3-D Ising models is clear. Hence, in this work, we design parallel algorithms for a 3-D Ising model on a homogeneous cluster and show how to extend it to a hierarchical cluster.

In particular, we focus on the efficient parallelization of:

1. the 2-D Ising model on hierarchical clusters
2. the 3-D Ising model on homogeneous as well as hierarchical clusters.

By efficiency, we are referring to performance across one iteration of the simulation rather than speed of convergence.

We consider the sweep spin selection scheme of the Ising model and analyze our results on the LogP model, which is modified to suit the hierarchical cluster. The utility of our theoretical analysis and prediction is shown by providing simulation data and analysis.

As stated previously, results for 2-D Ising on homogeneous clusters were presented in [26]. The focus of this thesis is on 2-D Ising simulations on hierarchical clusters, and on 3-D Ising simulations on both homogeneous and hierarchical clusters. As such, we will provide a brief overview of the results for 2-D Ising on homogeneous clusters, and explain the additional issues needed in order to address the problem domain of this thesis. We also note that preliminary results from work in this thesis have been published in [27].

The contents of this thesis are as follows. Chapter 2 describes the parallel sweep selection algorithms for the 2-D Ising model with emphasis on hierarchical clusters. Chapter 3 describes the efficient parallelization of the 3-D Ising model on homogeneous as well as hierarchical clusters. Chapter 4 presents the concluding remarks and suggestions for future work in the area.

In the remainder of this chapter, we provide background on the Ising model and its parallel simulation, and a discussion of the LogP parallel machine model and its modification for hierarchical clusters.

1.1 The Ising model and its simulation

The ferromagnetic Ising model [11] is a prototype for describing a wide variety of interacting systems composed of many discrete entities. Apart from ferromagnets, the Ising model can be readily adapted to simulate the behavior of certain other systems as well. More common among these are the lattice gas[24] and the binary alloy[15]. Applications of the Ising model include its use in the study of alloys in components [14] and load balancing in parallel computing [16].

Invented by W. Lenz in 1920 as a simple model of a ferromagnet, the Ising model [18] represents a collection of interacting spins placed on a d-dimensional lattice, and is far and away the most influential model of a system capable of a phase transition[8]. The properties of the model were first computed by E. Ising in 1925. In this model, the sites of a lattice are occupied by magnetic moments or spins, denoted S_i, that assume only either of two states - *up* or *down*. Each spin S_i interacts with its nearest neighbors and may interact with an external magnetic field[15].

The Ising model is not only the simplest to analyze but also unifies the study of phase transitions in systems as diverse as ferromagnets, gas-liquids, liquid mixtures, binary alloys, etc. To study the statistical mechanics of the Ising model, the kinetic energy of the atoms occupying the various lattice sites can be disregarded, for the phenomenon of phases transitions is essentially a consequence of the interaction energy among the atoms; in the interaction energy again, only the nearest-neighbor contributions are included, in the hope that the farther-neighbor contributions would not affect the results qualitatively [24]. To be able to study properties such as magnetic susceptibility, the lattice is subject to an external magnetic field. The spin then possesses an additional potential energy.

The Ising model can be described by the Hamiltonian:

$$H = \frac{1}{2} \sum_{i,j} G_{ij}\, S_i\, S_j - B \sum_{i} S_i, \tag{1.1}$$

where $S_i = \pm 1$ and where

$$G_{ij} = \begin{cases} J & \text{if } i, j \text{ are neighbors} \\ 0 & otherwise \end{cases} \tag{1.2}$$

J is the interaction strength and B is the magnetic field [8, 24].

Its thermodynamic properties can be calculated from the associated canonical partition function

$$Z = \sum_{\{S_i\}} \exp\{-\beta H\}, \tag{1.3}$$

where β is proportional to the inverse temperature and the sum is performed over all possible spin configurations[26].

Some of the thermodynamic quantities which are of interest in studying the Ising model are the magnetization, defined as $M = \sum_i S_i$, the energy given by $E = -J\sum_{i,j} S_i S_j$ where S_i is spin at site i, $< i, j >$ are nearest neighbors in the lattice, J is the interaction strength[20]. If $J > 0$, the state of lowest energy is when all spins are aligned. This is called a ferromagnet.

Figure 1.1 is an example of a 2-D Ising spin model. In this thesis, a ferromagnetic ($J > 0$),

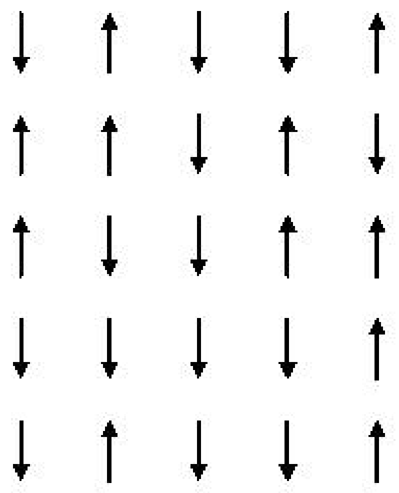


Figure 1.1: An example of 2D Ising spin model[24]

two-state spin system with nearest-neighbor interactions, on a d-dimensional lattice for $d = 2$ and $d = 3$ in the absence of a magnetic field is considered. The two states of the system are represented by +1 for *up* spin and -1 for *down* spin.

Approaches used in studying the Ising model include the method of series expansions, the renormalization group method and Monte Carlo methods[24].

As the name implies, Monte Carlo methods employ random numbers to simulate statistical fluctuations in order to carry out numerical integrations and computer simulations of systems with large numbers of degrees of freedom[24]. More discussion on these methods are available in [4, 5, 6].

The Metropolis Monte Carlo algorithm[21] is widely used to simulate the Ising model. The practical implementation of the Metropolis algorithm is quite straightforward and is one of the main reasons for its great success[8]. In the algorithm, while selecting a single spin, the spin can either be selected at random, or each spin in the sample may be selected in turn, called sweep selection. Either of these procedures will ensure that the necessary accessibility criterion of the system is satisfied[8]. The energy change on reversing a spin involves only the values of neighboring spins. Hence, parallelization of the Metropolis algorithm can be simply done.

The steps in the sequential Metropolis MC simulation [21] can be described as follows:

1. Initialize the lattice to a random initial state where each element has a value of -1(down) or +1(up).

2. Select a spin and decide whether to flip it or not based on the change in the interaction energy value of the spin. If the energy value decreases, the spin is flipped. If not, it is flipped with a probability $P(\delta E) = e^{\frac{-\delta E}{K_b T}}$ where δE is the change in energy, K_b is the Boltzman constant and T is the temperature.

3. Repeat the previous step till the system reaches equilibrium state.

The MC algorithm satisfies the detailed balance condition which enables the system to reach equilibrium over time. Periodic boundary conditions are also imposed on the system to avoid extended defects[1].

The detailed balance condition is a sufficient condition for an equilibrium (time independent) probability distribution. According to this condition, the probability of a system being in any state A and changing to any other state B is equal to the probability of it being in state B and changing to state A.

We have a valid Monte Carlo algorithm if:

1. We have a means of generating a new configuration B from a previous configuration A such that the transition probability $P(A \rightarrow B)$ satisfies detailed balance.

2. The generation procedure is ergodic, i.e. every configuration can be reached from every other configuration in a finite number of iterations.

The Metropolis algorithm satisfies the first criterion for all statistical systems. The second criterion is model dependent, and not always true (e.g. at T=0)[16, 8].

Another important condition that must be maintained while the Metropolis algorithm is implemented is the periodic boundary condition, where the lattice wraps around on itself to form a torus. Thus the sites on the uppermost and lowermost boundaries of a two-dimensional lattice are considered to be neighbors, as are the sites on the left- and right-hand edges [16]. In the three-dimensional lattice, the uppermost and the lowermost planes are regarded as neighbors in the energy calculation, as are the back and front planes, and also the leftmost and rightmost planes of the lattice. This has been shown to give the smallest finite size effects, and diminishes the disturbance from the boundaries of the system [7].

1.2 Parallelization of the simulation on hierarchical clusters

Parallelization of the Monte Carlo algorithm is desired to efficiently simulate large-scale Ising systems. Various papers have focused on parallelizing the MC simulations for 2-D Ising models for specific parallel machines [16, 28, 29]. Recently, research in efficient MC simulations for the 2-D Ising model on homogeneous clusters has been introduced [26] representing the change to clusters as one of the main environments for parallel simulations.

Heterogeneous clusters are used as environments for parallel simulations to leverage a variety of existing resources and networks. This necessitates the problem of focussing on efficient parallelization of applications on heterogeneous clusters. Hierarchical clusters are a prevalent form of heterogeneity where groups of homogeneous clusters (sub-networks) are connected via a high-latency network. Hence, in this thesis, we focus on the design, analysis, and simulation of parallel MC algorithms on hierarchical clusters.

A hierarchical cluster is viewed as consisting of two components: (a) a collection of homogeneous clusters each containing P_2 processors, each connected via a similar subnetwork structure, and (b) each homogeneous cluster is then viewed as a supernode connected via a high-latency network, such that communication across clusters require communication on this network tier. In this work, we assume that the homogeneous clusters are identical. A simple example of when a hierarchical cluster is used is when two or more identical clusters of nodes are separated by a large distance geographically and hence connected by a larger latency link. In certain cases, though the clusters are not far apart geographically, they may be interconnected through additional intermediate switches which cause the link between the clusters to have a higher latency than the links within a cluster. While hierarchical clusters are a special form of heterogeneous clusters, they proved to be a difficult problem domain and, as we shall present, required different data layouts with specific decompositions, and simulation methods to yield efficient run-times across a Monte Carlo iteration.

1.3 The LogP Parallel Model and Modification

The LogP model [12] is a general parallel machine model that provides a framework for the design and analysis of parallel and distributed algorithms. In comparison with other parallel machine models , such as PRAM, BSP, and specific network topologies, the LogP is better as it satisfies all the three criteria of being general, practical and realistic. For instance, the PRAM is simple but not realistic as it imposes the use of a shared memory and does not consider network topology. The BSP model does not have sufficient parameters to characterize important network properties. Hence, LogP has been widely used to analyze parallel algorithms. We base our analysis on this model and also use a modification to the model to analyze results on a hierarchical cluster. In this model, the letters L, o, g, and P represent 4 parameters. These parameters are,

L: the network latency of transmitting a unit length message from its source module to its target module.

o: the overhead, defined as the length of time that a processor is engaged in the transmission or reception of each message.

g: the gap, defined as the minimum time interval between consecutive message transmissions or consecutive message receptions to a processor.

P: the number of processor/memory modules.

The LogP model defines important properties of the network via the L, o and g parameters. It assumes that there are a finite number of processors P, each with local memory. Each processor has its own clock, hence synchronization and communication is performed independently using the local clock. Thus, a homogeneous cluster can be represented by the LogP model where the processors communicate by sending and receiving messages.

In the LogP model, L is the delay involved in sending a unit length message. We consider a byte as the length unit. o is the overhead for pushing the message out into the network or pulling the message in from the network. Hence, the cost of sending a 1-byte message is simply $C(1) = L + 2o$. Assuming that the underlying network sends and receives messages in a pipelined or overlapped scheme as in modern communication schemes, the $(k+1)$th byte is sent shortly after sending the kth byte. The interval between sending the (k+1)th byte and the kth byte is defined as g in the LogP model, which is a property of the network and it represents the reciprocal of the bandwidth. Hence, the cost of sending a k-byte message is $C(k) = L + 2o + (k-1)g$.

Designing efficient and optimal methods of communication such as broadcasting has been much researched on the LogP [19, 25] and these results provide insights in the design of communication patterns.

1.3.1 Hierarchical cluster

Since we view a hierarchical cluster as a collection of identical homogeneous clusters, the homogeneous clusters are represented by LogP parameters. In order to represent the high latency of the supernode network, it is important to represent the differences in latency. Therefore, a hierarchical cluster can be defined as a cluster of homogeneous nodes which are partitioned into two or more supernodes where the nodes within a supernode are interconnected by links with latency L_2 while the supernodes are connected by links of latency L_1 where $L_1 > L_2$. In this work, we call the cluster with nodes interconnected by the smaller latency links as a supernode and denote the smaller latency by L_2. The link between nodes in two different supernodes are considered to have a latency of L_1 where $L_1 > L_2$.

1.4 Parallelization of 2-D Ising model on a homogeneous cluster

The parallel implementation of the MC algorithm is essentially applying the algorithm simultaneously on different parts of the lattice. Strategies involving multiple spin flips developed include the cluster algorithms by Swendsen and Wang [28], and Wolf [29]. Parallel implementation of the Metropolis algorithm have been suggested using checkerboard pattern decomposition in [16]. Portable methods for the parallelization of the algorithm have been designed on the LogP model in [26].

1.4.1 Spin selection schemes

The MC algorithm requires the selection of spins from the lattice. Based on the randomness of the selection, there are two kinds of selections possible, namely the sweep spin selection and the random spin selection. The sweep selection scheme can be defined as follows[26]:

Definition 1.4.1 *Given a 2-D lattice of size $h_1 \times h_2$, the sweep selection scheme selects spin $S(\lfloor \frac{t}{h_2} \rfloor, t \text{ mod } h_2)$ at time step t for $0 \leq t < h_1 h_2$.*

Intuitively, the sweep spin selection scheme can be defined as selecting spins from the lattice one by one from the left to the right and then from the top to the bottom. The sweep selection is easy to implement and parallelize. In the simulation of the 2-D Ising model using the Monte Carlo algorithm, a MC step is defined as the selection of every spin on the lattice once. Thus, in the case of the sweep spin selection scheme, one MC step is the sweep over the whole lattice.

The random selection scheme, on the other hand, can be described as selecting spins uniformly randomly on the lattice. Formally, the random selection scheme can be defined as follows[26]:

Definition 1.4.2 *Given a 2D lattice of size $h \times h$, the random selection scheme selects spins* *uniformly at random on the lattice. One MC step of the random selection scheme consists of h^2 such selections.*

Algorithms to parallelize both the sweep and random spin selection schemes were designed in [26], and it was shown that the efficiency of random selection scheme per Monte Carlo iteration was not comparable to the efficiency of that of sweep selection scheme. We note that the communication overhead for the random selection scheme cannot be improved significantly for hierarchical clusters, but the efficiency of sweep selection scheme can be improved. Thus, we design layouts for the sweep selection scheme in this thesis. Our design is based on modifications of design techniques presented in [26] in order to perform optimally on hierarchical networks.

1.4.2 Domain decomposition

Parallel sweep spin selection schemes typically use two different methods of domain decomposition; namely, blocked data layouts and stripped data layouts [26]. Following are the definitions of the two different methods of domain decomposition.

Definition 1.4.3 *Given a 2-D lattice of size $S \times S$ and P processors denoted by $P_0, P_1, \cdots, P_{p-1}$, a blocked domain decomposition method(i.e., blocked data layout) is defined as partitioning the lattice into P contiguous sublattices each of size $\frac{S}{\sqrt{P}} \times \frac{S}{\sqrt{P}}$, where processor P_i is assigned the spins at site (j,k) for $(i \ mod \ \sqrt{P})\frac{S}{\sqrt{P}} \leq j < ((i \ mod \ \sqrt{P}) + 1)\frac{S}{\sqrt{P}}$, and $(\lfloor \frac{i}{\sqrt{P}} \rfloor)\frac{S}{\sqrt{P}} \leq k < (\lfloor \frac{i}{\sqrt{P}} \rfloor + 1)\frac{S}{\sqrt{P}}$.*

Definition 1.4.4 *Given a 2-D lattice of size $S \times S$ and P processors denoted by $P_0, P_1, \cdots, P_{p-1}$, a stripped domain decomposition method (i.e., stripped data layout) is defined as partitioning the lattice into P contiguous sublattices each of size $\frac{S}{P} \times S$[or $S \times \frac{S}{P}$], where processor P_i is assigned the spins at site (j,k) for $0 \leq j < S$ [or $i\frac{S}{P} \leq j < (i+1)\frac{S}{P}$], and $i\frac{S}{P} \leq k < (i+1)\frac{S}{P}$ [or $0 \leq k < S$].*

In the following chapter, we present the main results of the design of parallel algorithms for the sweep spin selection scheme on the blocked and stripped data layouts for a 2-D Ising model, from [26]. We then derive guiding equations to choose the best data layout for implementing the algorithm on a hierarchical cluster.

Chapter 2

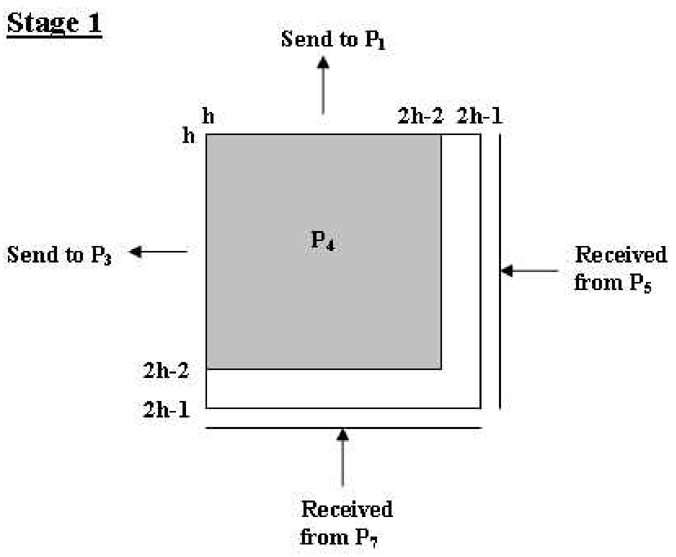
Parallel sweep selection algorithms for the 2-D Ising spin model

In this chapter, we discuss the design issues in efficiently simulating 2-D Ising spin models on hierarchical clusters. We provide data layout analysis, guiding equations, algorithms, and simulation results. Our results and presentation in this and the remaining chapters are based on efficiency analysis across one Monte Carlo iteration.

Detailed algorithm design and analysis for the parallelization of Metropolis Monte Carlo algorithm with sweep selection on both blocked data layout and stripped data layout for homogeneous clusters were presented in [26]. In order to discuss the results of this thesis, particular results from [26] were scrutinized and/or used, and thus we begin by discussing those results below.

2.1 Previous Results in Parallelization on Homogeneous Clusters

When multiple simultaneous spin flips occur on the lattice, the detailed balance condition requires that neighboring spins should not be flipped at the same time [16]. Hence, parallelizing the Monte Carlo algorithm requires that the processors should avoid flipping adjacent boundary spins simultaneously. To achieve this, a Monte Carlo step is split into stages.



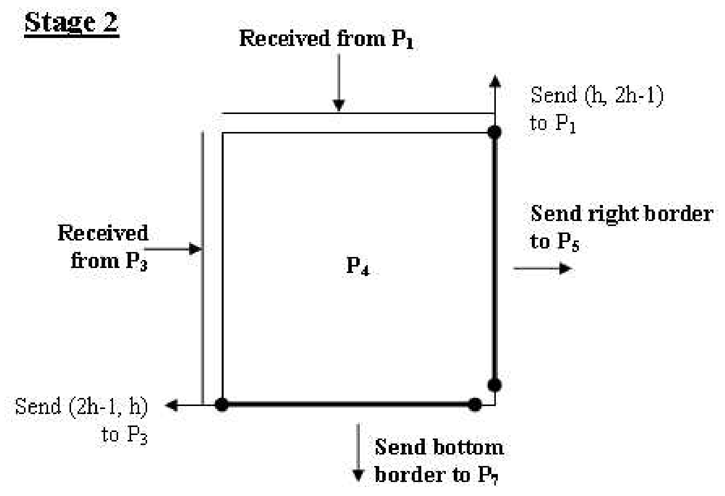
The shaded area represents the region in which P_4 executes its local spin update operations $\{(h, h), (2h-2, 2h-2)\}$

Figure 2.1: Stage 1 of blocked sweep selection[26]

The stages of computation and communication involved in the parallel algorithm when blocked data layout is used are described in [26]. The stages are summarized in figures 2.1 - 2.3[26]. As the process is symmetric in regards to processors, the figures focus only on processor P_4, which is the central processor when 9 processors are used. In the design, processor P_4 is assigned a subblock containing rows h to $2h-1$ and columns h to $2h-1$, where $h = \frac{S}{3}$. In order to flip a spin at site (h, j), it needs to know the value of 4 neighbor spins, which are $(h, j-1), (h+1, j), (h, j+1)$ and $(h-1, j)$. The first three lattice sites are in processor P_4, but the last one, $(h-1, j)$, is in processor P_1. Clearly communication between P_1 and P_4 is needed. Furthermore, in order to keep P_1 and P_4 from flipping adjacent boundary spins at the same time, synchronization between them is also necessary. This yields the three stages as shown in the figures. The pseudo code of the algorithm from [26] is as follows:

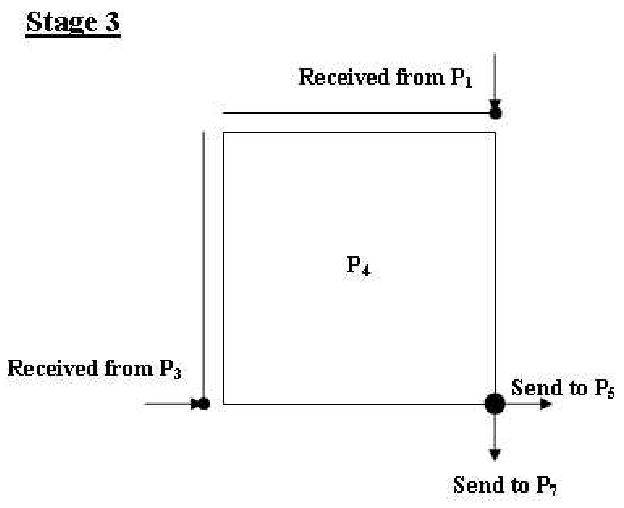
Algorithm 2.1.1 *Blocked Sweep Selection Scheme*

```
**Begin**
**For** *each processor* $P_i$ **DO**
```



P_4 executes its local spin update operations on two borders: the bottom border and the right border.

Figure 2.2: Stage 2 of blocked sweep selection[26]



P_4 executes its local spin update operations on the lower right corner $(2h-1, 2h-1)$.

Figure 2.3: Stage 3 of blocked sweep selection[26]

```
/*Each processor initializes its sub-lattice of size $h \times h$ where $h = \frac{S}{\sqrt{P}}$, where each element of the sub-lattice is randomly assigned a value of -1 or +1. $S$ is the width of the lattice and $P$ is the number of processors. The processor also sends its right and lower borders to its right and lower neighbors respectively, and receives the top and left boundaries from its neighbors.*/
Initialize();
$K = 0$;
While $K <$(Total MC steps) DO
Begin
//Stage 1
/*The processor sweeps over the sub-lattice $(0,0)-(h-2,h-2)$, when it selects the spins on the sublattice one after the other and computes the potential energy change for each selection and decides whether to flip the spin or not*/
Sweep( $(0,0),(h-2,h-2)$ );
/*The processor sends row 0 and column 0 to the upper and left neighbors respectively.*/
Send ( row 0, upper );
Send ( column 0, left );
/*The processor receives spins from its right and lower neighbors and stores them in the right array and the bottom array respectively.*/
Receive( right, right array );
Receive( lower, bottom array );
//Stage 2
/*The processor sweeps over the bottom row and the right column, except for the spin at $(h-1,h-1)$.*/
Sweep( $(h-1,0),(h-1,h-2)$ );
Sweep( $(0,h-1),(h-2,h-1)$ );
/*The processor sends the bottom row and the right column to the lower and right processors respectively.*/
Send ( row $h-1$, lower);
Send ( column $h-1$, right);
```

```
/*The processor sends the spin at $(h-1,0)$ to its left neighbor, and the spin at $(0,h-1)$ to its upper neighbor.*/
Send ( $(h-1,0)$, left );
Send ( $(0,h-1)$, upper );
/*The processor receives spins from its left neighbor and upper neighbor and stores them in the left array and the top array respectively.*/
Receive( left, left array );
Receive( upper, top array );
/*The processor receives one spin from its right neighbor into the $(h-1)th$ position of its right array, and one spin from its lower neighbor into the $(h-1)th$ position of its bottom array.*/
Receive( right, right array[$h-1$] );
Receive( lower, bottom array[$h-1$] );
//Stage 3
/*The processor updates the spin at $(h-1,h-1)$.*/
Sweep( $(h-1,h-1),(h-1,h-1)$ );
/*The processor sends the spin at $(h-1,h-1)$ to its right and lower neighbors.*/
Send ( $(h-1,h-1)$, right );
Send ( $(h-1,h-1)$, lower );
/* The processor receives one spin from its upper neighbor and stores it in the $(h-1)th$ position in its top array, and receives one spin from its left neighbor and stores it in the $(h-1)th$ position in the left array.*/
Receive( upper, top array[$h-1$] );
Receive( left, left array[$h-1$] );
$K = K + 1$;
ENDWhile
ENDFor
ENDAlgorithm
```

The design of the sweep selection scheme on stripped data layout is very similar to that of blocked data layout, except that each processor has only 2 neighbors. To satisfy the detailed balance condition, as in the blocked data layout, one MC step is divided into stages. Instead of using 3 stages, there are only 2 stages for the stripped case. Considering a stripped data layout using 4 processors, the stages can be described on processor P_0 as follows:

In stage 1, processor P_0 locally updates rows 0 to $(h-2)$ in its own sublattice. It then sweeps from left to right and then up to down. After local updates, the processor sends out spins of row 0 to its upper neighbor P_3, while receiving spins from its lower neighbor P_1, and stores the received spins into the bottom boundary array.

In stage 2, processor P_0 locally updates the bottom row (row $(h-1)$ and sweeps from left to right. After local updates, it sends the bottom row to its lower neighbor P_1, while receiving spins from it upper neighbor P_3, and stores the received spins into the top boundary array.

The pseudo code of the algorithm from [26] is as follows:

Algorithm 2.1.2 *Stripped Sweep Selection Scheme*

```
Begin
For each processor $P_i$ DO
/*Each processor initializes its sub-lattice of size $h \times S$ where $h = \frac{S}{P}$, where each element of the sub-lattice is randomly assigned a value of -1 or +1. S is the width of the lattice and P is the number of processors. The processor also sends its bottom border to its lower neighbor and receives spins from the top neighbor to the top array.*/
Initialize();
$K = 0$;
While $K <$(Total MC steps) DO
Begin
//Stage 1
/*The processor sweeps over the sublattice $(0,0)-(h-2,S-1)$, when it selects the spins on the sublattice one after the other and computes the potential energy change for each selection and decides whether to flip the spin or not*/
Sweep( $(0,0),(h-2,S-1)$ );
/*The processor sends row 0 to its upper neighbor*/
```

```
Send ( row 0, upper );
/*The processor receives spins from its lower neighbor and stores them in the bottom array*/
Receive( lower, bottom array );
//Stage 2
/*The processor sweeps over the bottom border*/
Sweep( $(h-1,0),(h-1,S-1)$ );
/*The processor sends the bottom row to its lower neighbor*/
Send ( row $h-1$, lower);
/*The processor receives spins from its upper neighbor and stores them in the top array*/
Receive( upper, top array );
$K = K + 1$;
ENDWhile
ENDFor
ENDAlgorithm
```

The analysis of the algorithm on the blocked data layout and the stripped data layout from [26] can be summarized as follows:

- Consider a blocked data layout. Let the sublattice size in each processor in the blocked data layout be $h \times h$. The communication cost involved in the phases of the parallel algorithm designed for the blocked layout is derived on the LogP model as follows:
  - The cost of sending 2 messages each of size $h-1$ is $T_1 = L + 4o + 2(h-2)g$.
  - The cost of sending 2 messages of size $h-1$ followed by 2 messages of size 1 is $T_2 = L + 8o + 2(h-2)g$.
  - The cost of sending two messages of size 1 is $T_3 = L + 4o$.

  Hence, the communication cost involved in one Monte Carlo step (where a Monte Carlo step is defined as the selection of h^2 spins from the sublattice) is

$$T_{comm} = 3L + 16o + (4h-8)g \tag{2.1}$$

where $h = \frac{S}{\sqrt{P}}$.

- The communication cost involved in a Monte Carlo step on the stripped data layout is

$$2(L + 2o) + 2g(S - 1) \tag{2.2}$$

- From the Equations [2.1] and [2.2], given P processors and assuming a lattice of size $S \times S$,

  If $P = 4$, then $T(B) > T(S)$.

  If $P > 4$, then $T(B) \geq T(S)$ if and only if

$$S \leq \frac{\sqrt{P}(L + 12o - 6g)}{2(\sqrt{P} - 2)g}.$$

- For $P = 4$, blocked data layout is slower than stripped data layout. Clearly, choose stripped data layout.

- For $P > 4$, when the lattice size S is less than or equal to $\frac{\sqrt{P}(L+12o-6g)}{2(\sqrt{P}-2)g}$, blocked data layout is slower than stripped data layout, otherwise the reverse is true.

It should be noted that if the spin system is partitioned such that each processor has an equal number of spins, then the computation time across processors is equal. Furthermore, iterations of the simulation cannot be overlapped forcing clear barriers of coarse iterations of alternating computation and communication where the next wave of communication cannot begin until all computation from the previous iteration is completed. Therefore, differences in parallel running time become wholly dependent on communication. Communication, in general, is significantly affected by the decomposition of data and becomes one of the major design considerations in our simulation designs.

2.2 Sweep selection on 2-D Ising model for a hierarchical cluster

While there have been interesting results on layouts and run-time for this problem domain for 2-D Ising systems on a homogeneous cluster, it is not a straightforward extension to hierarchical clusters. It would seem at first glance that decomposing supernodes and processors within supernodes can create two tiers of tandem decomposition. In fact, it is more complicated than that, a data layout needs to consider the decomposition of the dimensions of the Ising model together with the hierarchical nature of the network. In this thesis, we leverage the results for 2-D Ising systems on homogeneous clusters, in order to obtain results for hierarchical clusters. Many of the results in this section have been published in [27].

In Ising simulations, a particularly important aspect in the design is the partitioning of the spin system (i.e. the layout) for use among the processors. The layouts designed for homogeneous networks [26] that yield efficient simulations have the simulations utilize equal length messages that communicate across all the links. When this layout is used in a heterogeneous network where the nodes are connected by links with large variations in latency, the communication time will be dependant on the communication across the longest latency link. As such, the choice of data layout will have a significant impact on run-time making this an important area that we explore.

For 2-D Ising, clearly the lattice can be decomposed in a number of ways. Suppose we subdivide the P processors into $\widehat{P}$ groups of processors consisting of $\bar{P}$ processors each. Examples include:

- blocked layout among $\widehat{P}$ groups of processors followed by stripped layout among each group of $\bar{P}$ (referred to as **blocked-stripped** layout),
- stripped layout among the $\widehat{P}$ groups followed by blocked layout among the $\bar{P}$ processors (referred to as **stripped-blocked** layout),
- blocked layouts for both tiers (referred to as **blocked-blocked** layout), and
- stripped layouts for both tiers (referred to as **stripped-stripped** layout.

Let P_1 be the number of supernodes and P_2 be the number of nodes within a supernode ($P = P_1 P_2$). The case $\widehat{P} = P_1$ and $\bar{P} = P_2$ is intuitive. However, it should also be clear that groups of supernodes can be used to decompose one dimension, thus we have allowed for greater generality in our hierarchical layouts.

Clearly, we are leveraging generic layouts in a hierarchical environment. However, coupling all the pairs of variation is overly simplistic. Some issues:

- Stripped-Blocked layout cannot be done since in order to do a block partitioning, the sublattice must be square. If stripped layout is used as the first decomposition, all sublattices will be rectangular (non-square).
- Blocked-Stripped layout can only be done if $\widehat{P}$ is a square number.
- Blocked-Blocked can only be done if both $\bar{P}$ and $\widehat{P}$ are square numbers.
- Stripped-Stripped allows many variations. Most importantly, the two sets of stripping need not be along the same dimension. Therefore, there are two possibilities:
  - stripping on the same dimension (i.e. horizontal-horizontal, vertical-vertical which are symmetric to each other)
  - stripping in two different dimensions (i.e. horizontal-vertical, vertical-horizontal which are symmetric to each other)

Upon further examination, it is clear that we can formally define applicable hierarchical layouts based on these variations. In fact, the variability is based on only one parameter which we call β. The discussion of generalized hierarchical data layouts now follows.

> Given P processors and a lattice of size $S \times S$, each processor is assigned a sublattice of size $\frac{S}{\beta} \times \frac{\beta S}{P}$.

If $\beta = 1$ or $\beta = P$, each processor will be assigned a sublattice of size $S \times \frac{S}{P}$. This layout is the stripped data layout from homogeneous clusters.

If $\beta = \sqrt{P}$, each processor will be assigned a sublattice of size $\frac{S}{\sqrt{P}} \times \frac{S}{\sqrt{P}}$. This layout is the blocked data layout in homogeneous clusters.

For a homogeneous cluster, for $\beta > 1$, $\beta = \sqrt{P}$ will be the best layout as this divides the lattice into square blocks and hence the length of the borders are minimum which minimizes the communication time.

However, for a hierarchical cluster, we derive the value of β that minimizes communication time based on the network parameters and show that it may not yield good results to implement the data layout of $\beta = \sqrt{P}$ suggested for the homogeneous cluster, on this cluster. Clearly, β should be a multiple of P_1.

In order to understand the design issues needed in a hierarchical cluster, we begin by focusing our design of parallel MC simulations to only two supernodes. This problem in and of itself provides interesting design issues. We will then show how to extend the results to a hierarchical cluster with an arbitrary number of supernodes.

2.2.1 Two supernodes

Consider a network with $P_1 = 2$ supernodes, where each supernode has $P_2 = \frac{P}{2}$ processors. Let the latency in transferring a message between any two processors within a supernode be L_2 and the latency in transferring a message between the two supernodes be L_1, where $L_2 < L_1$.

Focusing on the design of the algorithm, let us begin by considering the first phase of the algorithm where two adjacent borders of the sublattice are communicated to two neighbors. Using a blocked layout in this case, the border over the link of latency L_1 is communicated to the neighbor before the border over the link of latency L_2 is communicated, to minimize the communication time. Let h be the sublattice size in the blocked layout where $h = \frac{S}{\sqrt{P}}$. If $L_2 + (h-2)g + o < L_1$, the destination processor will start receiving the message at $L_2 + 3o + (h-2)g$ and will receive for $(h-2)g$. Further, if $L_2 + 2o + 2(h-2)g < L_1$, the processor that receives border spins over two links of latencies L_1 and L_2 during a communication phase, has to wait for $L_1 - L_2 - 2o - 2(h-2)g$ before receiving the spins over the link of latency L_1 after receiving the spins over the link of latency L_2. The total communication time in phase 1 in this case will be $L_2 + 3o + (h-2)g + (h-2)g + L_1 + o - L_2 - 3o - 2(h-2)g + o + (h-2)g$, which is $L_1 + 2o + (h-2)g$.

Hence the communication time will be reduced if a layout where the length of the borders of the sublattice are such that a processor does not have to wait to begin receiving the spins over the link with latency L_1 after receiving the spins over the link with latency L_2. Let β be the number of rows of processors to which the lattice is divided in this case such that each processor is assigned a sublattice of size $\frac{S}{\beta} \times \frac{\beta S}{P}$. We call this layout the β-partitioning layout. The optimal value of β to be used for the network considered can be derived by equating the wait time of the processor to zero. This is stated in the following theorem.

Theorem 2.2.1 *Given a lattice of size $S \times S$ and a hierarchical cluster with $\frac{P}{2}$ processors in each supernode with latencies of L_1 and L_2 between the supernodes and within a supernode respectively where $L_1 - L_2 \geq 2o + 2g(\frac{S}{\sqrt{P}} - 2)$, the number of rows of processors(β) into which the lattice should be divided to minimize the waiting time of a processor when receiving data is given by*

$$\beta = \frac{(L_1 - L_2 - 2o + 4g)P - \sqrt{(L_1 - L_2 - 2o + 4g)^2 P^2 - 4S^2 g^2 P}}{2Sg} \tag{2.3}$$

Proof. When a β-partitioning layout is considered, to nullify the waiting time when a processor is receiving data, L_1 should be equal to $L_2 + 2o + (\frac{\beta S}{P} - 2)g + (\frac{S}{\beta} - 2)g$

$\Rightarrow \frac{\beta S}{P} + \frac{S}{\beta} = \frac{L_1 - L_2 - 2o}{g} + 4$

$\Rightarrow Sg\beta^2 - (L_1 - L_2 - 2o + 4g)P\beta + SgP = 0$

$\Rightarrow \beta = \frac{(L_1 - L_2 - 2o + 4g)P \pm \sqrt{(L_1 - L_2 - 2o + 4g)^2 P^2 - 4S^2 g^2 P}}{2Sg}$.
As $\beta <= \sqrt{P}$, $\beta = \frac{(L_1 - L_2 - 2o + 4g)P - \sqrt{(L_1 - L_2 - 2o + 4g)^2 P^2 - 4S^2 g^2 P}}{2Sg}$. □

The sweep spin selection algorithm for blocked data layout designed in [26] is modified for the data layout suggested here as follows: At every communication step in the three phases, the communication that involves sending a message over the link of latency L_1 is done before the other communication which involves sending a message over the link of latency L_2. Similarly, the receiving of a message over the link of latency L_2 is done before receiving a message over the link of latency L_1.

The communication time involved in this case can be derived as follows. The destination processor will start receiving at $L_2 + 3o + (\frac{\beta S}{P} - 2)g$ and will receive for $(\frac{S}{\beta} - 2)g$. As $L_2 + 3o + (\frac{\beta S}{P} - 2)g + (\frac{S}{\beta} - 2)g = L_1 + o$, the processor will receive for $o + (\frac{\beta S}{P} - 2)g$. Hence the communication time for phase 1 will be $L_2 + 4o + 2(\frac{\beta S}{P} - 2)g + (\frac{S}{\beta} - 2)g$.

The time taken for communication during one MC step when the blocked data layout is implemented on the hierarchical cluster can be derived as follows:

$$T_1 = L_1 + 2o + (h - 2)g$$

$$T_2 = L_1 + 4o + (h - 2)g$$

$$T_3 = L_1 + 2o$$

Hence,

$$T_{comm} = 3L_1 + 8o + 2(\frac{S}{\sqrt{P}} - 2)g \tag{2.4}$$

The communication time involved in a β-partitioning layout, where β is optimized according to Equation [2.3], during one MC step, is derived as follows:

$$T_1 = L_1 + 2o + (\frac{\beta S}{P} - 2)g$$

$$T_2 = L_1 + 4o + (\frac{\beta S}{P} - 2)g$$

$$T_3 = L_1 + 2o$$

Hence,

$$T_{comm} = 3L_1 + 8o + 2(\frac{\beta S}{P} - 2)g \tag{2.5}$$

Thus, the data layout designed here performs better than standard blocked layout by $2Sg(\frac{\sqrt{P}-\beta}{P})$. Furthermore, the optimal size for β can be determined theoretically for hierarchical clusters of two supernodes.

Stripped data layout for hierarchical clusters

When the higher latency L_1 exceeds a threshold value, it can be shown that the stripped data layout will be the best layout in such a case and the algorithm designed for a stripped data layout for homogeneous clusters can be used on the hierarchical clusters.

The algorithm on a stripped data layout involves two stages. The communication time involved in the first stage on a hierarchical cluster using stripped data layout will be $L_1 + 2o + (S-1)g$. Hence the total communication for one Monte Carlo step of the algorithm on stripped data layout for a hierarchical cluster will be

$$T_{comm} = 2L_1 + 4o + 2(S-1)g \tag{2.6}$$

Comparing Equations 2.5 and 2.6, it can be seen that the stripped data layout performs better than a β-partitioning layout, where $\beta > 1$ when

$$3L_1 + 8o + 2(\frac{\beta S}{P} - 2)g - (2L_1 + 4o + 2(S-1)g) > 0$$

From this inequality, we can derive that the stripped data layout performs better than any other layout when

$$L_1 > 4o + 2g(S - \frac{S\beta}{P} + 1)$$

Thus, when the higher latency of the hierarchical cluster is greater than $4o+2g(S-\frac{S\beta}{P}+1)$, the stripped data layout is the best layout to yield the minimum run time in parallelizing the Monte Carlo algorithm.

2.2.2 P_1 supernodes

Consider a network with P_1 supernodes where each supernode has P_2 processors. Let the latency between two processors in different supernodes be L_1 and the latency between two processors in the same supernode be L_2, where $L_1 > L_2$.

The lattice can be divided in a stripped layout or a blocked layout among the supernodes. In the stripped layout, the lattice is divided into P_1 contiguous sublattices each of size $\frac{S}{P_1} \times S$, while in the blocked layout, the lattice is divided into P_1 contiguous sublattices each of size $\sqrt{P_1} \times \sqrt{P_1}$, where each sublattice is assigned to a supernode. In a stripped layout, the P_2 processors in each supernode can be assigned a sublattice of size $\frac{S}{P_2} \times \frac{S}{P_1}$ where $\frac{S}{P_2}$ is the length of the border between two processors in different supernodes. In a blocked layout, the P_2 processors in each supernode can be assigned a sublattice of size $\frac{S}{\sqrt{P_1 P_2}} \times \frac{S}{\sqrt{P_1 P_2}}$. Furthermore, the results can be generalized to $\bar{P}$ and $\widehat{P}$ and we determine the optimal β. Below is a listing a results we have derived:

- If $P_2 > P_1$, the stripped layout would be better than the blocked layout as the length of the border between two processors in different supernodes will be lesser in the stripped layout than in the blocked layout. That is, $\frac{S}{P_2} < \frac{S}{\sqrt{P_1 P_2}}$.

- If $P_2 < P_1$, the blocked layout would be better than the stripped layout as the length of the border between two processors in different supernodes will be lesser in the blocked layout than in the stripped layout. That is, $\frac{S}{\sqrt{P_1 P_2}} < \frac{S}{P_2}$.

- If $P_2 = P_1$, either of the layouts could be chosen as in both cases, the length of the border between processors are the same and the number of neighbors to each processor is also the same.

- If $P_2 >> P_1$, the number of rows in which the P_2 processors are arranged could be more than one. Let the number of rows be β. The value of β is derived similarly to the value of β derived in Section 2.2.1.

As we can see from the above results, we have been able to consider what types of data layouts perform best under different ratios of P_1 and P_2 in order to minimize running time.

Even within 2-D Ising systems, there are many possible variations. Therefore, determining the hierarchical layouts that optimize run-time for simulations is particularly relevant and interesting.

Generalized d-dimensional Ising systems are of interest to statistical physicists. In particular, 2-D and 3-D systems are studied in-depth. In the next chapter, we will focus on 3-D systems in order to determine the additional levels of complexity that will affect running time with the future goal of generalizing to arbitrary dimensions.

2.3 Simulation results

We now provide simulation results to compare against our theoretical analyses. The simulations were run on a homogeneous 64-node cluster with 2.66GHz Pentium 4 processors connected by a 10Mbps Ethernet backbone interconnect. The hierarchical cluster was simulated on the homogeneous cluster by delaying the message communication between nodes not in the same supernode to simulate a latency of L_1 over L_2. The algorithm was implemented using LAM/MPI[10]. The values of L,o, and g for the cluster were measured and the values(in μs) are: $L = L_2 = 350$, $o = 20$, and $g = 1$.

2.3.1 Simulation environment

To implement the algorithm on a hierarchical cluster, we simulate the hierarchy on a homogeneous set of machines. We now describe the design of the simulation environment, which has been presented in [23]. The design uses the message passing interface (MPI) library and extends the communication functions provided in the library to support the hierarchy desired. The simulation of a hierarchical cluster on a homogeneous cluster can be described as follows:

Simulation of two supernodes

The homogeneous cluster can be considered to be consisting of two equal groups of processors where each group is a supernode in the hierarchical cluster. The sending of messages from a node to any other node in the cluster uses the MPI send function. But the receiving of messages invokes a user-defined receive function which incorporates the hierarchy. The hierarchy of multiple latencies can be implemented as follows. When the processor receives a message for the first time in a communication phase, it records the current time at the processor as the firstreceivetime. In the receive function, if the sender of the message lies in the same supernode as the receiver, then the MPI receive function is invoked to receive the message immediately, as this implies receiving over the link of latency L_2. Otherwise, the receiver invokes a sleep function to make the processor wait for a time interval which equals the difference between the latency difference defined for the simulation, and a value that is determined as the difference between the current time and the firstreceivetime. The processor then receives the message from the buffer. This ensures that the higher latency of L_1 is implemented on top of the lower latency of L_2, which is the latency of the homogeneous cluster.

The receive function used to implement the hierarchy can be described in pseudo-code as follows:

```
Begin
    If $firstreceivetime = 0$ //this is the first time the receive is invoked in the communication phase
        firstreceivetime := current time at the processor
    If the sending processor is not in the same supernode
    Begin
        $current\_time$ := current time at the processor
        $sleep\_time := Latency\_Difference - current\_time + firstreceivetime$
        If $sleep\_time < 0$, $sleep\_time := 0$
        Make the processor wait for $sleep\_time$
        Receive the message
    Else
        Receive the message immediately
    End If
End Function
```

Extending the simulation to a finite number of supernodes

This implementation can be extended to multiple supernodes by implementing the latency difference corresponding to the latency of the link between the sending and receiving nodes involved, in the receive function.

The simulation of the parallel phases of the Ising model on the hierarchical cluster thus boils down to simulating the corresponding phases on a homogeneous cluster.

Implementation of the algorithm on a homogeneous cluster

The size of the sublattice at each processor is dynamically determined based on the lattice size and the number of processors in the system. Similarly the neighbors of each processor are determined independently from the value of β and the rank of the processor.

The parallel sweep or random selection algorithm is then executed by the processors by choosing spins in a sweep or random fashion from their corresponding sublattices. Communication of border spins occurs alternately with the phases of computation.

The algorithm can be used to measure the static and dynamic properties of the Ising model which includes the total energy and magnetism of the system at various Monte Carlo iterations. The measurement of the total energy of the system shows the decrease in energy as the iterations progress which characterizes the behavior of the Ising model.

The physical properties of the medium including the temperature, the size of the Ising lattice and the number of Monte Carlo iterations required are defined in the program and can be varied based on the requirement of the system.

2.3.2 Performance comparison

We measure the communication time involved in the sweep selection algorithm using layouts with various possible values of β for 20 Monte-Carlo steps on 16, 36 and 64 processors. The results obtained are shown in Tables 2.1, 2.2 and 2.3.

As stated previously, the computation time involved in the parallel run-time of the algorithm will be the same across all processors since they are all of the same computational speeds and are allocated the same amount of data. Moreover, there is an inherent barrier between iterations ensuring non-overlapping whatsoever of iterations and it is easily shown that the processors perform in lock-step in each iteration, that is, a processor will not start on the next iteration unless every processor has completed the previous iteration. Hence it would suffice to measure the communication time taken by the algorithm for layouts considering different values of β, where β is a positive even value less than or equal to $\sqrt{P}$ where P is the number of processors. When $\beta = \sqrt{P}$, the layout is the blocked data layout which is used in the homogeneous cluster.

The communication times shown in Tables 2.1, 2.2 and 2.3 are measured over various values of lattice sizes and latency differences ($L_1 - L_2$), which are input to the program. The theoretical value of β for the cluster can be calculated using the equation derived in Section 2.2 and is also shown in the table. As the results show, when the value of β used in the layout is equal to the theoretical value of β for the setup, the communication time is minimized.

The results also show that the improvement in the communication time obtained using a data β-partitioning layout where β is optimized using Equation 2.3 is more significant for higher lattice sizes. This is true especially in comparison with the blocked data layout as the difference in communication time is proportional to $2Sg(\frac{\sqrt{P}-\beta}{P})$ as shown in Section 2.2.1, which is higher for higher lattice sizes.

Tables 2.4, 2.5 and 2.6 show the comparison of the results of running parallel Ising simulations designed for the hierarchical cluster against the theoretical values, for 16, 36 and 64 processors respectively. In these tables, we compare the performance improvement in implementing the algorithm with data layout optimized for a hierarchical cluster over the blocked data layout on the hierarchical cluster for the 2-D Ising model with the theoretical value of the expected performance improvement. These results evaluate the accuracy of the simulated hierarchical environment by calculating the performance improvement in using the optimized data layout over the blocked data layout, which is then compared with the theoretical value of the improvement expected for the given parameters. The improvement in communication times is measured in executing 20 Monte-Carlo iterations of the sweep selection algorithm of the 2-D Ising model on a hierarchical cluster using the β-partitioning layout over the blocked data layout. The values of lattice sizes and latency differences ($L_1 - L_2$) in μs are input to the program, from which the theoretical value of β can be determined using Equation 2.3.

The choice of the latency difference in the input to the simulation reflects the flexibility offered by the simulation environment to simulate different values of the higher latency L_1. In the experimental runs, the lattice size and the latency difference used yields practical values of β that describes the optimal data layout to be used for the input parameters of the cluster. Hence, in the experiments on 16 processors shown in Tables 2.1 and 2.4, the value of the lattice size and the latency difference($L_1 - L_2$) input to the program were chosen such that the theoretical value of β is 2, and the communication time using blocked data layout is compared with the β-partitioning layout. Similarly, in the results obtained for 36 processors shown in Table 2.2 and 2.5, the values of the difference in latency ($L_1 - L_2$) corresponding to the various lattice sizes were chosen to obtain a β of 2 or 4, that is compared with the blocked data layout. For the results on 64 processors as shown in Tables 2.3 and 2.6, the values chosen yield a β of 2 or 4, that is compared with the blocked data layout, which has a β of 8.

We also measure the parallel run time involved in the sweep selection algorithm which includes the communication as well as the computation time. The run time results are shown in Tables 2.7 and 2.8 measured for 36 and 64 processors respectively. The run time is measured using layouts with various possible values of β for 20 Monte-Carlo steps on 36 and 64 processors which are divided into two supernodes. Again, the parallel run time of the algorithm is measured for layouts considering different values of β, where β is a positive even value less than or equal to $\sqrt{P}$ where P is the number of processors. When $\beta = \sqrt{P}$, the layout is the blocked data layout which is used in the homogeneous cluster.

The comparison of the performance improvement obtained in the run time is again comparable with the theoretical value expected as shown in 2.9 and 2.10. The run times shown are measured over various values of lattice sizes and latency differences ($L_1 - L_2$), which are input to the program. The theoretical value of β for the cluster can be calculated using the equation derived in Section 2.2 and is also shown in the table. As the results show, when the value of β used in the layout is equal to the theoretical value of β for the setup, the run time is lesser than considering a homogeneous layout. This proves the analysis provided in Section 2.2.

It can be seen that the performance improvement obtained in measuring the communication time compares with the theoretical improvement, just like the performance improvement obtained in measuring the run time does. Hence measuring the communication time alone does predict our analysis accurately proving our hypothesis in not including the computations in the parallel time analysis.

In comparison of the communication time with the run time obtained, though the com-

Lattice size	$L_1 - L_2$	Theoretical β	$\beta = 4$	$\beta = 2$
21600	13536	2	4.5	4.34
27000	16911	2	5.25	5.12
32400	20286	2	7.4	7.19
54000	33786	2	12.57	12.33
81000	50661	2	22.8	22.42
108000	67536	2	39.17	38.67

Table 2.1: Communication time on a hierarchical cluster for 16 processors (in seconds)

Lattice size	$L_1 - L_2$	Theoretical β	$\beta = 6$	$\beta = 4$	$\beta = 2$
21600	12036	2	8.34	8.43	8.23
21600	7836	4	7.93	7.87	7.94
27000	15036	2	9.26	9.56	9.11
27000	9786	4	8.5	8.4	8.48
32400	18036	2	10.89	11.1	10.7
32400	11736	4	9.02	8.94	9.22
54000	30036	2	17.57	17.78	17.27
54000	19536	4	18.32	18.14	18.32
81000	45036	2	25.3	25.17	24.93
81000	29286	4	22.52	22.27	22.51
108000	60036	2	32.1	31.77	31.58
108000	39036	4	28.96	28.63	29.1

Table 2.2: Communication time on a hierarchical cluster for 36 processors (in seconds)

putation time involved in the Ising model is more significant than the communication time, it should be noted that the computation time and communication time depend on independent factors, the computation time dependant on the processing speed of the processors, while the communication time on the speed of the network connecting the processors. Hence an improvement in communication time may yield significant results depending on the environment and the applications.

We note that quality of approach is based on the method (sweep selection) and simulations have already been performed [26] to show that the approach maintains the necessary domain criteria in a parallel environment.

Further results have been obtained to validate the performance of stripped data layout

Lattice size	$L_1 - L_2$	Theoretical β	$\beta = 8$	$\beta = 4$	$\beta = 2$
21600	11511	2	9.45	9.4	9.36
21600	6786	4	7.96	7.88	8.19
28000	14911	2	11.72	11.7	11.54
28000	8786	4	11.37	11.3	11.43
36000	19161	2	14.96	14.9	14.78
36000	11286	4	13.72	13.65	13.7
56000	29786	2	17.02	16.9	16.73
56000	17536	4	15.28	15.07	15.4
80000	42536	2	22.34	22.3	21.97
80000	25036	4	20.98	20.7	21.1
108000	57411	2	26.14	26.02	25.7
108000	33786	4	24.02	23.78	24.1

Table 2.3: Communication time on a hierarchical cluster for 64 processors (in seconds)

Lattice size	$L_1 - L_2$	Performance improvement by simulation	Theoretical improvement
21600	13536	0.16	0.108
27000	16911	0.13	0.135
32400	20286	0.21	0.162
54000	33786	0.24	0.27
81000	50661	0.38	0.405
108000	67536	0.5	0.54

Table 2.4: Comparison of the performance improvement in communication time obtained by simulation and theoretically on a hierarchical cluster for 16 processors (in seconds)

over a β-partitioning layout for larger values of the latency L_1. These results are shown in Tables 2.12 and 2.13.

The threshold values of latency difference over which the stripped layout yields the best result for the cluster are shown in Table 2.11.

The values of communication times determined on the hierarchical cluster for higher values of latency differences for the stripped data layout and the β-partitioning layout for 16 and 36 processors are given in Tables 2.12 and 2.13. The last column in the tables show the difference in communication time between using a β-partitioning layout with $\beta = 2$ and the stripped layout.

Lattice size	$L_1 - L_2$	Performance improvement by simulation	Theoretical improvement
21600	12036	0.11	0.096
21600	7836	0.06	0.048
27000	15036	0.15	0.12
27000	9786	0.1	0.06
32400	18036	0.19	0.144
32400	11736	0.08	0.072
54000	30036	0.3	0.24
54000	19536	0.18	0.12
81000	45036	0.37	0.36
81000	29286	0.25	0.18
108000	60036	0.52	0.48
108000	39036	0.33	0.24

Table 2.5: Comparison of the performance improvement in communication time obtained by simulation and theoretically on a hierarchical cluster for 36 processors (in seconds)

As these results show, when the latency difference exceeds the threshold value as shown in Table 2.11, the stripped data layout performs better than the layout where β is two. It suffices to consider a β of two as we have shown that for higher latencies, the value of β to be considered to obtain the minimum communication time is two. It can also be noted that the difference in performance increases with increasing values of latency difference.

Lattice size	$L_1 - L_2$	Performance improvement by simulation	Theoretical improvement
21600	11511	0.09	0.081
21600	6786	0.08	0.054
28000	14911	0.18	0.105
28000	8786	0.07	0.07
36000	19161	0.18	0.135
36000	11286	0.07	0.09
56000	29786	0.29	0.21
56000	17536	0.21	0.14
80000	42536	0.37	0.3
80000	25036	0.28	0.2
108000	57411	0.44	0.405
108000	33786	0.24	0.27

Table 2.6: Comparison of the performance improvement in communication time obtained by simulation and theoretically on a hierarchical cluster for 64 processors (in seconds)

Lattice size	$L_1 - L_2$	Theoretical β	$\beta = 6$	$\beta = 4$	$\beta = 2$
21600	12036	2	264.9	264.88	264.82
21600	7836	4	258.6	258.49	258.66
27000	15036	2	406.87	406.77	406.73
27000	9786	4	399.7	399.62	399.74
32400	18036	2	579.79	579.76	579.62
32400	11736	4	575.75	575.65	575.78
54000	30036	2	1577.32	1577.2	1577.09
54000	19536	4	1568.72	1568.62	1568.72
81000	45036	2	3541.85	3541.9	3541.4
81000	29286	4	3529.9	3529.76	3529.92

Table 2.7: Run time on a hierarchical cluster for 36 processors (in seconds)

Lattice size	$L_1 - L_2$	Theoretical β	$\beta = 8$	$\beta = 4$	$\beta = 2$
21600	11511	2	144.4	144.5	144.32
21600	6786	4	142.82	142.76	142.9
28000	14911	2	249.94	249.87	249.82
28000	8786	4	244.71	244.57	245.09
36000	19161	2	411.23	411.1	411.02
36000	11286	4	406.13	406.06	406.14
56000	29786	2	970.63	970.62	970.4
56000	17536	4	968.17	968.02	968.2
80000	42536	2	1961.41	1961.3	1961.1
80000	25036	4	1958.89	1958.62	1959.05
108000	57411	2	3471.29	3470.93	3470.8
108000	33786	4	3461.49	3461.3	3461.6

Table 2.8: Run time on a hierarchical cluster for 64 processors (in seconds)

Lattice size	$L_1 - L_2$	Performance improvement by simulation	Theoretical improvement
21600	12036	0.08	0.096
21600	7836	0.11	0.048
27000	15036	0.14	0.12
27000	9786	0.08	0.06
32400	18036	0.17	0.144
32400	11736	0.1	0.072
54000	30036	0.23	0.24
54000	19536	0.1	0.12
81000	45036	0.45	0.36
81000	29286	0.14	0.18

Table 2.9: Comparison of the performance improvement in run time obtained by simulation and theoretically on a hierarchical cluster for 36 processors (in seconds)

Lattice size	$L_1 - L_2$	Performance improvement by simulation	Theoretical improvement
21600	11511	0.08	0.081
21600	6786	0.06	0.054
28000	14911	0.12	0.105
28000	8786	0.14	0.07
36000	19161	0.21	0.135
36000	11286	0.07	0.09
56000	29786	0.23	0.21
56000	17536	0.15	0.14
80000	42536	0.31	0.3
80000	25036	0.27	0.2
108000	57411	0.49	0.405
108000	33786	0.19	0.27

Table 2.10: Comparison of the performance improvement in run time obtained by simulation and theoretically on a hierarchical cluster for 64 processors (in seconds)

Number of processors	Lattice size	Threshold value of $L_1 - L_2$
16	5200	8832
16	10800	18632
36	10800	20132
36	21600	40532

Table 2.11: Threshold values of latency difference for stripped data layout

Lattice size	$L_1 - L_2$	$beta = 2$	stripped layout	performance difference
5200	5000	1.31	1.5	-0.19
5200	10000	1.55	1.52	0.03
5200	20000	1.57	1.53	0.04
5200	40000	2.61	1.72	0.89
5200	80000	5.04	1.78	3.26
5200	100000	6.28	1.79	4.49
10800	10000	2.68	2.83	-0.15
10800	20000	2.93	2.92	0.01
10800	30000	3.01	2.92	0.09
10800	40000	3.08	2.94	0.14
10800	100000	6.33	3.62	2.71

Table 2.12: Communication time in seconds using stripped layout and β-partitioning layout for 16 processors

Lattice size	$L_1 - L_2$	$beta = 2$	stripped layout	performance difference
10800	10000	3.83	4.9	-1.07
10800	20000	4.88	4.92	-0.04
10800	30000	5.37	5.13	0.24
10800	40000	5.76	5.34	0.42
10800	60000	6.17	5.67	0.5
10800	100000	6.86	6	0.86
21600	20000	7.46	8.48	-1.02
21600	40000	8.74	8.99	-0.25
21600	50000	9.64	9.19	0.45
21600	60000	10.16	9.56	0.6
21600	80000	10.65	9.62	1.03
21600	100000	10.91	9.75	1.16
21600	200000	12.42	10.1	2.32

Table 2.13: Communication time in seconds using stripped layout and β-partitioning layout for 36 processors

Chapter 3

The 3-D Ising Model

The 3-D Ising model, being the most practical model of ferromagnetic systems, is of particular interest to statistical physicists and has been studied in many places including [9, 2]. It has proved useful in several applications including series expansions and real-space renormalization group calculations. Moreover, the 3-D Ising model has been shown to provide better series estimates of the critical energy than the 2-D Ising model[22]. Hence, simulations of the 3-D Ising model are vital to analyzing the properties of the model.

Further, the 3-D Ising model involves larger lattices than the 2-D model as the lattices are three-dimensional, which necessitates the need for efficient parallelization of the simulations. We show that the results obtained on the 2-D Ising model for the parallelization on homogeneous and hierarchical clusters, can be extended to the 3-D model, thus proving that our analysis forms the basis for the parallelization of higher-dimensional Ising models.

In the 3-D Ising model, layouts must be considered that may subdivide up to the three dimensions in the system. Again, we must couple this with the multi-latency values in a hierarchical cluster. We first discuss the efficient parallelization of the 3-D Ising model on a homogeneous cluster and then look at extending the algorithm to a hierarchical cluster. The theoretical results in this chapter have been published in [27].

3.1 Parallelization on a homogeneous cluster

For parallelizing the 3-D Ising model on a homogeneous cluster, three main models of data layout division can be identified, two of which are analogous to the data layouts considered on a 2-D model. Consider a 3-D lattice of size $S \times S \times S$ and P processors denoted by $P_0, P_1, \cdots, P_{p-1}$. The three cases are:

- Partition the lattice into P contiguous sublattices each of size $\frac{S}{P} \times S \times S$[or $S \times \frac{S}{P} \times S$ or $S \times S \times \frac{S}{P}$], where processor P_i is assigned the spins at site (j, k, l) for $i\frac{S}{P} \leq j < (i+1)\frac{S}{P}$ [or $0 \leq j < S$], $0 \leq k < S$ [or $i\frac{S}{P} \leq k < (i+1)\frac{S}{P}$], and $0 \leq l < S$ [or $i\frac{S}{P} \leq l < (i+1)\frac{S}{P}$].

- Partition the lattice into P contiguous sublattices each of size $\frac{S}{\sqrt{P}} \times \frac{S}{\sqrt{P}} \times S$, where processor P_i is assigned the spins at site (j, k, l) for $(i \ mod \ \sqrt{P})\frac{S}{\sqrt{P}} \leq j < ((i \ mod \ \sqrt{P}) + 1)\frac{S}{\sqrt{P}}$, $(\lfloor\frac{i}{\sqrt{P}}\rfloor)\frac{S}{\sqrt{P}} \leq k < (\lfloor\frac{i}{\sqrt{P}}\rfloor + 1)\frac{S}{\sqrt{P}}$, and $0 \leq l < S$.

- Partition the lattice into P contiguous sublattices each of size $\frac{S}{\sqrt[3]{P}} \times \frac{S}{\sqrt[3]{P}} \times \frac{S}{\sqrt[3]{P}}$, where processor P_i is assigned the spins at site (j, k, l) for $((i \ mod \ P^{\frac{2}{3}}) \ mod \ \sqrt[3]{P})\frac{S}{\sqrt[3]{P}} \leq j < (((i \ mod \ P^{\frac{2}{3}}) \ mod \ \sqrt[3]{P}) + 1)\frac{S}{\sqrt[3]{P}}$, $(\lfloor\frac{i \ mod \ P^{\frac{2}{3}}}{\sqrt[3]{P}}\rfloor)\frac{S}{\sqrt[3]{P}} \leq k < (\lfloor\frac{i \ mod \ P^{\frac{2}{3}}}{\sqrt[3]{P}}\rfloor + 1)\frac{S}{\sqrt[3]{P}}$, and $(\frac{i}{P^{\frac{2}{3}}})\frac{S}{\sqrt[3]{P}} \leq l < (\frac{i}{P^{\frac{2}{3}}} + 1)\frac{S}{\sqrt[3]{P}}$.

The layouts in the three cases for 64 processors are shown in figures 3.1, 3.2 and 3.3. As stated previously, the computation time will be exactly the same across processors, and there are inherent barriers in iterations. Therefore, again, the changes in parallel running time are wholly dependent on communication. This again places the spotlight on data layout.

We note that we will consider the three different types of data layouts, provide algorithms, and determine theoretical results on our hierarchical model. We note that we first obtain results on communication for homogeneous clusters (using LogP) and then account for the hierarchical multiple-latency constraints.

Consider the first case where each processor has two neighbors. A parallel Monte Carlo algorithm for sweep selection on this layout is an extension of the algorithm on the stripped layout on the 2-D model suggested in [26] and will have the following computation and communication phases:

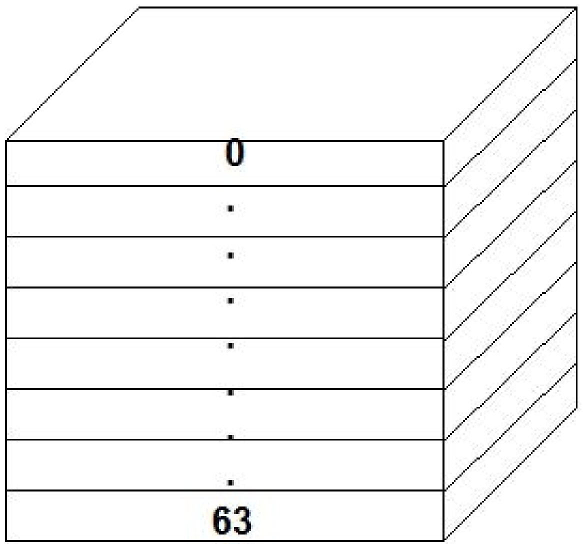


Figure 3.1: 3D layout of the first case for 64 processors

Algorithm 3.1.1 *Sweep Spin Selection Scheme on 3-D Ising model - case 1*

```
Begin
For each processor $P_i$ DO
/*Each processor initializes its sub-lattice of size $h \times S \times S$ where $h = \frac{S}{P}$, where each element of the sub-lattice is randomly assigned a value of -1 or +1. S is the width of the lattice and P is the number of processors. The processor also sends its bottom row to its lower neighbor and receives spins from the upper neighbor and stores in the top matrix.*/
Initialize();
$K = 0$;
While $K <$(Total MC steps) DO
Begin
//Stage 1
/*The processor sweeps over the sublattice $(0,0,0)-(h-2,S-1,S-1)$, when it selects the spins on the sublattice one after the other and computes the potential energy change for each selection and decides whether to flip the spin or not*/
Sweep( $(0,0,0),(h-2,S-1,S-1)$ );
/*The processor sends row 0 to its upper neighbor*/
```

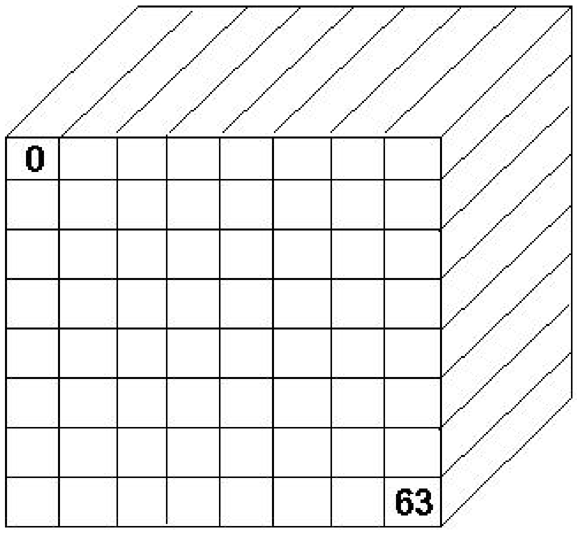


Figure 3.2: 3D layout of the second case for 64 processors

```
Send ( row 0, upper );
/*The processor receives spins from its lower neighbor and stores them in the bottom matrix*/
Receive( lower, bottom matrix );
//Stage 2
/*The processor sweeps over the bottom border*/
Sweep( $(h-1,0,0),(h-1,S-1,S-1)$ );
/*The processor sends the bottom row to its lower neighbor*/
Send ( row $h-1$, lower);
/*The processor receives spins from its upper neighbor and stores them in the top matrix*/
Receive( upper, top matrix );
$K = K + 1$;
ENDWhile
ENDFor
ENDAlgorithm
```

The time taken for communication in one phase in algorithm 3.1.1 is $L + 2o + (S^2 - 1)g$. The total communication time involved in parallel sweep selection on the layout considered

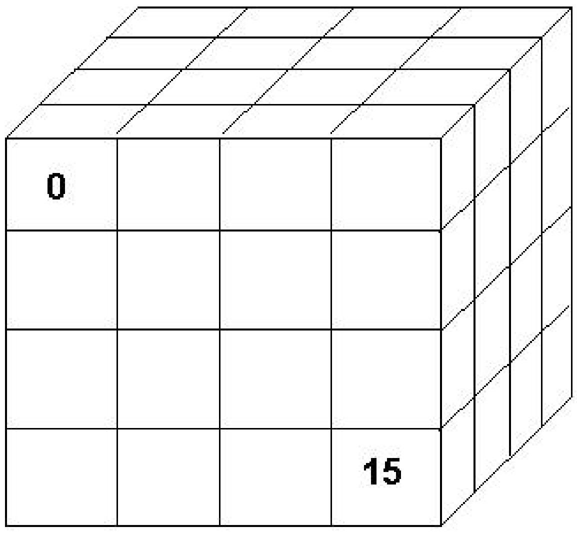


Figure 3.3: 3D layout of the third case for 64 processors

in case 1 is:

$$T_{comm} = 2(L + 2o + (S^2 - 1)g) \tag{3.1}$$

Consider the second case where each processor has 4 neighbors as in the case of blocked data layout on a 2-D model. The algorithm is similar to the algorithm for sweep selection on a blocked data layout for a 2-D model as suggested in [26]. The computation and communication steps of the algorithm are as follows:

Algorithm 3.1.2 *Sweep Spin Selection Scheme on 3-D Ising model - case 2*

```
Begin
For each processor $P_i$ DO
/*Each processor initializes its sub-lattice of size $h \times h \times S$ where $h = \frac{S}{\sqrt{P}}$, where each element of the sub-lattice is randomly assigned a value of -1 or +1. S is the width of the lattice and P is the number of processors. The processor also sends its right and lower borders to its right and lower neighbors respectively, and receives the top and left boundaries from its neighbors.*/
Initialize();
$K = 0$;
While $K <$(Total MC steps) DO
```

```
Begin
//Stage 1
/*The processor sweeps over the sub-lattice $(0,0,0)-(h-2,h-2,S-1)$, when it selects
the spins on the sublattice one after the other and computes the potential energy change for
each selection and decides whether to flip the spin or not*/
Sweep( $(0,0,0),(h-2,h-2,S-1)$ );
/*The processor sends row 0 and column 0 to the upper and left neighbors respectively.*/
Send ( row 0, upper );
Send ( column 0, left );
/*The processor receives spins from its right and lower neighbors and stores them in the right
matrix and the bottom matrix respectively.*/
Receive( right, right matrix );
Receive( lower, bottom matrix );
//Stage 2
/*The processor sweeps over the bottom row and the right column, except for the spins at
$(h-1,h-1)$.*/
Sweep( $(h-1,0,0),(h-1,h-2,S-1)$ );
Sweep( $(0,h-1,0),(h-2,h-1,S-1)$ );
/*The processor sends the bottom row and the right column to the lower and right processors
respectively.*/
Send ( row $h-1$, lower);
Send ( column $h-1$, right);
/*The processor sends the spins at $(h-1,0)$ to its left neighbor, and the spins at $(0,h-1)$
to its upper neighbor.*/
Send ( Array at row $h-1$ and column 0, left );
Send ( Array at row 0 and column $h-1$, upper );
/*The processor receives spins from its left neighbor and upper neighbor and stores them in
the left matrix and the top matrix respectively.*/
Receive( left, left matrix );
Receive( upper, top matrix );
```

```
/*The processor receives spins from its right neighbor into the $(h-1)th$ row of its right matrix, and spins from its lower neighbor into the $(h-1)th$ row of its bottom matrix.*/
Receive( right, row $h-1$ of right matrix );
Receive( lower, row $h-1$ of bottom matrix );
//Stage 3
/*The processor updates the spins at $(h-1, h-1)$.*/
Sweep( $(h-1, h-1, 0), (h-1, h-1, S-1)$ );
/*The processor sends the spins at $(h-1, h-1)$ to its right and lower neighbors.*/
Send ( Array at row $h-1$ and column $h-1$, right );
Send ( Array at row $h-1$ and column $h-1$, lower );
/* The processor receives spins from its upper neighbor and stores them in the $(h-1)th$ row in its top matrix, and receives spins from its left neighbor and stores them in the $(h-1)th$ row in the left matrix.*/
Receive( upper, row $h-1$ of top matrix );
Receive( left, row $h-1$ of left matrix );
$K = K + 1$;
ENDWhile
ENDFor
ENDAlgorithm
```

The communication time involved in the three phases in algorithm 3.1.2 are:

$$T_1 = L + 4o + 2(\frac{S^2}{\sqrt{P}} - S - 1)g$$

$$T_2 = L + 8o + (\frac{2S^2}{\sqrt{P}} - 4)g$$

$$T_3 = L + 4o + (2S - 2)g$$

Hence the total communication time of sweep selection on the data layout considered in case 2 is:

$$T_{comm} = 3L + 16o + 4(\frac{S^2}{\sqrt{P}} - 2)g \tag{3.2}$$

Consider the third case where the lattice is divided into equal cubic blocks with $\frac{S}{\sqrt[3]{p}} \times \frac{S}{\sqrt[3]{p}} \times \frac{S}{\sqrt[3]{p}}$ spins in each block and each processor has 6 neighbors. A parallel Monte Carlo algorithm for sweep spin selection on this layout has four stages of computation and communication as described below:

Algorithm 3.1.3 *Sweep Spin Selection Scheme on 3-D Ising model - case 3*

```
Begin
For each processor $P_i$ DO
/*Each processor initializes its sub-lattice of size $h \times h \times h$ where $h = \frac{S}{P^{\frac{1}{3}}}$, where each element of the sub-lattice is randomly assigned a value of -1 or +1. S is the width of the lattice and P is the number of processors. The processor also sends its right, lower and back borders to its right, lower and back neighbors respectively, and receives the top, left and front boundaries from its neighbors.*/
Initialize();
$K = 0$;
While $K <$(Total MC steps) DO
Begin
//Stage 1
/*The processor sweeps over the sub-lattice $(0,0,0) - (h-2, h-2, h-2)$, when it selects the spins on the sublattice one after the other and computes the potential energy change for each selection and decides whether to flip the spin or not*/
Sweep( $(0,0,0), (h-2, h-2, h-2)$ );
/*The processor sends row 0, column 0 and depth 0 matrices to the upper, left and front neighbors respectively.*/
Send ( row 0 matrix, upper );
Send ( column 0 matrix, left );
Send ( depth 0 matrix, front );
/*The processor receives spins from its right, lower and back neighbors and stores them in the right matrix, bottom matrix and the back matrix respectively.*/
Receive( right, right matrix );
Receive( lower, bottom matrix );
Receive( back, back matrix );
//Stage 2
/*The processor sweeps over the bottom row, right column and the back face, except for the spins along the lower-right, lower-back and back-right borders.*/
Sweep( $(h-1,0,0),(h-1,h-2,h-2)$ );
Sweep( $(0,h-1,0),(h-2,h-1,h-2)$ );
Sweep( $(0,0,h-1),(h-2,h-2,h-1)$ );
/*The processor sends the bottom row, right column and the back face to the lower, right and back processors respectively.*/
Send ( row $h-1$, lower);
Send ( column $h-1$, right);
Send ( depth $h-1$, back);
/*The processor sends the spins at the bottom-front border to its front neighbor, the spins at the bottom-left border to its left neighbor, the spins at the right-front border to its front neighbor, the spins at the top-right border to its upper neighbor, the spins at the left-back border to its left neighbor, and the spins at the top-back border to its upper neighbor.*/
Send ( Array at row $h-1$ and depth 0, front );
Send ( Array at row $h-1$ and column 0, left );
Send ( Array at column $h-1$ and depth 0, front );
Send ( Array at row 0 and column $h-1$, upper );
Send ( Array at column 0 and depth $h-1$, left );
Send ( Array at row 0 and depth $h-1$, upper );
/*The processor receives spins from its left, upper and front neighbors and stores them in the left, top and front matrices respectively.*/
Receive( left, left matrix );
Receive( upper, top matrix );
Receive( front, front matrix );
/*The processor receives spins from its back neighbor into the $(h-1)$th row of its back matrix, spins from its right neighbor into the 0th column of its right matrix, spins from its back neighbor into the $(h-1)$th column of the back matrix, spins from its lower neighbor into the
```

```
(h − 1)th row of the bottom matrix, spins from the right neighbor into the (h − 1)th column
of the right matrix, and spins from the lower neighbor into the (h−1)th column of its bottom
matrix.*/
Receive( back, row $h-1$ of back matrix );
Receive( right, column 0 of right matrix );
Receive( back, column $h-1$ of back matrix );
Receive( lower, row $h-1$ of bottom matrix );
Receive( right, column $h-1$ of right matrix );
Receive( lower, column $h-1$ of bottom matrix );
//Stage 3
/*The processor updates the spins at the lower-right, lower-back and right-back borders except
for the spin at $(h-1, h-1, h-1)$.*/
Sweep( $(h-1, h-1, 0), (h-1, h-1, h-2)$ );
Sweep( $(h-1, 0, h-1), (h-1, h-2, h-1)$ );
Sweep( $(0, h-1, h-1), (h-2, h-1, h-1)$ );
/* The processor sends the spins along the lower-right border to the lower and right proces-
sors, the spins along the lower-back border to the lower and back processors, and the spins
along the right-back border to the right and back processors.*/
Send ( Array at row $h-1$ and column $h-1$, lower );
Send ( Array at row $h-1$ and column $h-1$, right );
Send ( Array at row $h-1$ and depth $h-1$, lower );
Send ( Array at row $h-1$ and depth $h-1$, back );
Send ( Array at column $h-1$ and depth $h-1$, back );
Send ( Array at column $h-1$ and depth $h-1$, right );
/*The processor sends the spin at lower-right-front corner to the front neighbor, the spin at
lower-left-back corner to the left processor, and the spin at top-right-back corner to the upper
processor.*/
Send ( Spin at $(h-1, h-1, 0)$, front );
Send ( Spin at $(h-1, 0, h-1)$, left );
Send ( Spin at $(0, h-1, h-1)$, upper );
```

```
/*The processor receives spins from its upper neighbor into row $(h-1)$ and column $(h-1)$ of the top matrix, spins from the left neighbor into row $(h-1)$ and column $(h-1)$ of the left matrix, spins from the front neighbor into row $(h-1)$ and column $(h-1)$ of the front matrix.*/
Receive( upper, row $h-1$ of top matrix );
Receive( left, row $h-1$ of left matrix );
Receive( upper, column $h-1$ of top matrix );
Receive( front, row $h-1$ of front matrix );
Receive( front, column $h-1$ of front matrix );
Receive( left, column $h-1$ of left matrix );
/*The processor receives a spin from its back neighbor into $(h-1, h-1)$ position in back matrix, a spin from its right neighbor into $(h-1, h-1)$ position in right matrix, and a spin from its lower neighbor into $(h-1, h-1)$ position in bottom matrix.*/
Receive( back, back matrix[$h-1$, $h-1$] );
Receive( right, right matrix[$h-1$, $h-1$] );
Receive( lower, bottom matrix[$h-1$, $h-1$] );
//Stage 4
/*The processor updates the spin at $(h-1, h-1, h-1)$*/
Sweep( $(h-1, h-1, h-1), (h-1, h-1, h-1)$ );
/*The processor sends the spin at $(h-1, h-1, h-1)$ to its right, lower and back neighbors.*/
Send ( Spin at $(h-1, h-1, h-1)$, right );
Send ( Spin at $(h-1, h-1, h-1)$, lower );
Send ( Spin at $(h-1, h-1, h-1)$, back );
/*The processor receives a spin from its left, upper and front neighbors and stores it in the $(h-1, h-1)$ position in the left, top and front matrices respectively. */
Receive( left, left matrix[$h-1$, $h-1$] );
Receive( upper, top matrix[$h-1$, $h-1$] );
Receive( front, front matrix[$h-1$, $h-1$] );
$K = K + 1$;
ENDWhile
```

```
ENDFor
ENDAlgorithm
```

Considering a homogeneous cluster, the communication times of the four phases of algorithm 3.1.3 on the LogP model are:

$$T_1 = L + 6o + 3((\frac{S}{\sqrt[3]{P}} - 1)^2 - 1)g$$

$$T_2 = L + 18o + 3((\frac{S}{\sqrt[3]{P}} - 1)^2 - 1)g + 6(\frac{S}{\sqrt[3]{P}} - 2)g$$

$$T_3 = L + 18o + 6(\frac{S}{\sqrt[3]{P}} - 2)g$$

$$T_4 = L + 6o$$

Hence, the total communication time of one Monte Carlo step in a 3-D Ising model using the layout in case 3 is:

$$T_{comm} = 4L + 48o + 6(\frac{S^2}{P^{\frac{2}{3}}} - 4)g \qquad (3.3)$$

3.1.1 Comparison of the Three Cases on LogP

Comparing equations 3.1, 3.2 and 3.3, the following guiding equations are obtained.

Theorem 3.1.1 *Given P processors and a lattice of size $S \times S \times S$,*

- *If $P <= 4$, $T_{comm}\{case1\} < T_{comm}\{case2\}$*

- *If $P > 4$, $T_{comm}\{case1\} > T_{comm}\{case2\}$ when*

$$S^2 > \frac{(L + 12o - 6g)\sqrt{P}}{2g(\sqrt{P} - 2)}$$

- *If $P > 12$, $T_{comm}\{case2\} > T_{comm}\{case3\}$ when*

$$S^2 > \frac{(L + 32o - 16g)P^{\frac{7}{6}}}{g(4P^{\frac{2}{3}} - 6P^{\frac{1}{2}})}$$

For homogeneous clusters, the results are able to be derived fairly easily. We now move to hierarchical clusters and obtain theoretical run-times.

3.2 3-D Ising model on a hierarchical cluster

The optimal data layout for sweep selection on the 3-D model on a hierarchical cluster can be derived using a method similar to that used for the 2-D model. We consider the algorithm used in case 3 of the homogeneous model and extend it to hierarchical clusters, because we have shown from the guiding equations that the third case performs better for higher layouts and more number of processors. The communication involved in phase 1 of the algorithm considered in case 3 of the homogeneous model when implemented on a hierarchical cluster with $L_1 > L_2$ is obtained as follows. The border over the link of latency L_1 is communicated to the neighbor before the border over the links of latency L_2 is communicated, to minimize the communication time in this case. Let h be the sublattice size in the layout considered in case 3 where $h = \frac{S}{\sqrt[3]{P}}$. If $L_2 + 4o + 3((h-1)^2 - 1)g < L_1$, the processor that receives one of the borders over the link of latency L_1 has to wait for $L_1 - L_2 - 4o - 3((h-1)^2 - 1)g$ before receiving the spins over the link of latency L_1 after receiving the spins over the links of latency L_2. The communication time in phase 1 in this case is $L_1 + 2o + ((h-1)^2 - 1)g$.

Hence the communication time will be reduced if a layout where the length of the borders are such that a processor does not have to wait for the spins over the link with latency L_1 after receiving the spins over the links with latency L_2. Let β be the number of rows of processors to which the lattice is divided where $\sqrt{\frac{P}{\beta}}$ is the number of processors in a row along a depth, such that each processor is assigned a sublattice of size $\frac{S}{\beta} \times \frac{\sqrt{\beta}S}{\sqrt{P}} \times \frac{\sqrt{\beta}S}{\sqrt{P}}$.

To nullify the wait time,

$$L_1 = L_2 + 4o + ((\frac{\sqrt{\beta}S}{\sqrt{P}} - 1)^2 - 1)g + 2((\frac{S}{\beta} - 1)(\frac{\sqrt{\beta}S}{\sqrt{P}} - 1) - 1)g$$

$$\Rightarrow (\frac{\sqrt{\beta}S}{\sqrt{P}} - 1)(\frac{\sqrt{\beta}S}{\sqrt{P}} + \frac{2S}{\beta} - 3) = \frac{L_1 - L_2 - 4o + 3g}{g}$$

$$\Rightarrow g\beta^2 S^2 - 4g\sqrt{P}S\beta\sqrt{\beta} - (L_1 - L_2 - 4o)P\beta + 2g\sqrt{P}S^2\sqrt{\beta} - 2gSP = 0 \qquad (3.4)$$

Solving Equation 3.4 for β will give the optimal value of β that minimizes the communication time involved in phase 1 of the algorithm.

3.2.1 Communication time

The time taken for communication when the algorithm designed in 3.1.3 is implemented on the hierarchical cluster can be derived as follows:

$$T_1 = L_1 + 2o + ((\frac{S}{\sqrt[3]{P}} - 1)^2 - 1)g$$

$$T_2 = L_1 + 6o + ((\frac{S}{\sqrt[3]{P}} - 1)^2 - 1)g + 2(\frac{S}{\sqrt[3]{P}} - 2)g$$

$$T_3 = L_1 + 6o + 2(\frac{S}{\sqrt[3]{P}} - 2)g$$

$$T_4 = L_1 + 2o$$

Hence,

$$T_{comm} = 4L_1 + 16o + 2((\frac{S}{\sqrt[3]{P}} - 1)^2 - 1)g + 4(\frac{S}{\sqrt[3]{P}} - 2)g \qquad (3.5)$$

The communication time involved in a β-partitioning layout where β is optimized according to Equation [3.4] is derived as:

$$T_1 = L_1 + 2o + ((\frac{\sqrt{\beta}S}{\sqrt{P}} - 1)^2 - 1)g$$

$$T_2 = L_1 + 6o + ((\frac{\sqrt{\beta}S}{\sqrt{P}} - 1)^2 - 1)g + 2(\frac{\sqrt{\beta}S}{\sqrt{P}} - 2)g$$

$$T_3 = L_1 + 6o + 2(\frac{\sqrt{\beta}S}{\sqrt{P}} - 2)g$$

$$T_4 = L_1 + 2o$$

Hence,

$$T_{comm} = 4L_1 + 16o + 2((\frac{\sqrt{\beta}S}{\sqrt{P}} - 1)^2 - 1)g + 4(\frac{\sqrt{\beta}S}{\sqrt{P}} - 2)g \tag{3.6}$$

Thus, the data layout designed here performs better than the data layout used in algorithm 3.1.3 by $2S^2g(\frac{\sqrt[3]{P}-\beta}{P})$.

For 3-D Ising, we have been able to determine the best layouts based on problem size, and parameters for the hierarchical cluster and determine a guiding equation for decomposition.

3.3 Simulation results

We now provide simulation results to compare the implementation of the three cases of the 3-D Ising model against our theoretical analyses. The simulations were run on a homogeneous 64-node cluster with 2.66GHz Pentium 4 processors connected by a 10Mbps Ethernet backbone interconnect. The algorithms were implemented using LAM/MPI[10]. The values of L,o, and g for the cluster were measured and the values(in μs) are: $L = 316$, $o = 20$, and $g = 1$.

The threshold values of lattice sizes obtained by theorem 3.1.1 are as follows:
When $P = 36$, if $S > 21$, case 2 is faster than case 1.
When $P = 64$, if $S > 20$, case 2 is faster than case 1, and if $S > 87$, case 3 is faster than case 2.

Table 3.1 shows the run-time in seconds obtained on 36 processors when the algorithms using layouts defined in cases 1 and 2 are implemented for different values of lattice sizes. The last column shows the difference in run-time between the implementation of cases 1 and 2. As the value of the lattice sizes used are greater than the threshold value for 36 processors, which is 21, case 2 is always faster than case 1. Moreover, the performance difference increases with increasing values of lattice sizes.

Table 3.2 shows the run-time in seconds obtained on 64 processors when the algorithms using layouts defined in cases 1, 2 and 3 are implemented, for different values of lattice sizes. The difference in run-times between the cases are shown in the last two columns. As the results show, case 2 performs better than case 1 as the lattice sizes used are greater than the threshold value of 64. Also, case 3 gives the best performance as the lattice sizes used are

Lattice size	Case 1	Case 2	$Case1 - Case2$
72	2.65	1.71	0.94
144	6.9	5.35	1.55
216	16.11	10.08	6.03
288	23.05	17.96	5.09
360	35.05	26.82	8.23
720	110.92	98.98	11.94

Table 3.1: Run-time for 20 MC iterations of the 3-D Ising model on 36 processors (in seconds)

above the threshold value of 87.

Lattice size	Case 1	Case 2	Case 3	$Case1 - Case2$	$Case2 - Case3$
128	6.8	4.5	4.02	2.3	0.48
192	13.92	11.1	9.95	2.82	1.15
256	25.19	23.67	21.8	1.52	1.87
384	44.5	41.1	38.9	3.4	2.2
512	56.59	53.63	51	2.96	2.63
640	83.18	79.8	76.4	3.38	3.4

Table 3.2: Run-time for 20 MC iterations of the 3-D Ising model on 64 processors (in seconds)

These performance results prove the validity of the theoretical results obtained on the 3-D model for a homogeneous cluster.

Chapter 4

Conclusions

In this work, we have analyzed the sweep selection algorithms for the 2-D Ising models on a parallel hierarchical cluster modeled by modifying the LogP model. We have also analyzed the parallelization of the sweep selection for 3-D Ising spin models on both homogeneous and hierarchical clusters. Moreover, we have provided guiding equations to choose the best data layout to implement the sweep selection algorithms for the 2-D and the 3-D Ising model on a homogeneous as well as a hierarchical cluster. The derivation of the guiding equations for the 2-D and 3-D Ising model on a hierarchical cluster show that the derivations can be easily extended to higher dimensional Ising models. Our performance analysis is based on run-time across a Monte Carlo iteration. Finally, results from implementation of the algorithm using both blocked data layout and the layout considering optimized number of rows, on a hierarchical cluster, have been obtained on the 2-D model, which prove the correctness of the theoretical results obtained in the work. This shows that the modeling and design of algorithms presented in this work are capturing realistic performance criteria and can provide good prediction of parallel running time. We have also discussed the simulation of hierarchical clusters on a homogeneous cluster and tested the accuracy of the environment by simulations. The environment is easily extendable and can be used in analyzing many other applications on such clusters.

We also show that an improvement in communication time as obtained in our analysis on hierarchical clusters may yield significant results depending on the environment and the data parallel application used.

Hierarchical clusters are particularly important as many clusters are composed of multiple processors on a board that are then networked together to form a cluster.

Our work clearly form the basis of many useful future work. The design and analysis of the parallelization of the 2-D and the 3-D Ising models on homogeneous and hierarchical clusters, shows that this work forms the basis for extension to multi-dimensional Ising systems, which will prove as a useful future work in the area, as hypercubic Ising models and in general multi-dimensional Ising models are of great research interest to statistical physicists. Moreover, the design on a hierarchical cluster with two tiers considered in this work is fundamental to designing data layouts for generic hierarchical clusters. Hence, future work can also be done in designing data layouts for efficient parallelization of the Ising model on generic hierarchical clusters which includes different latencies between the supernodes and unequal number of nodes in each supernode. Further work can also be done in designing efficient layouts for hierarchical clusters with heterogeneous nodes. This work can also be applied to the efficient parallelization of many other applications in computational sciences.

Bibliography

[1] N. W. Ashcroft and N. D. Mermin, *Solid State Physics.* Saunders, 1976.

[2] C. F. Baillie, R. Gupta, K. A. Hawick, and G. S. Pawley, “Monte Carlo renormalization-group study of the three-dimensional Ising model,” *Physical Review B*, vol. 45, pp. 10438–10453, 1992.

[3] D. Basak and D. K. Panda, “Designing Clustered Multiprocessor Systems under Packaging and Technological Advancements,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 9, pp. 962–978, 1996.

[4] K. Binder, *Monte Carlo Methods in Statistical Physics*, vol. 7. Berlin: Springer-Verlag, 2nd ed., 1986.

[5] K. Binder, *Applications of Monte Carlo Method in Statistical Physics*, vol. 36. Berlin: Springer-Verlag, 2nd ed., 1987.

[6] K. Binder, *The Monte Carlo Method in Condensed Matter Physics, Topics in Applied Physics*, vol. 71. Berlin: Springer-Verlag, 1992.

[7] K. Binder and D. W. Heermann, *Monte Carlo Simulation in Statistical Physics, An Introduction*, vol. 80 of *Springer Series in Solid-State Sciences.* Springer-Verlag, 3rd ed., 1997.

[8] J. J. Binney, N. J. Dowrick, A. J. Fisher, and M. E. J. Newman, *The Theory of Critical Phenomena: An Introduction to the Renormalization Group.* Clarendon Press, Oxford, 1992.

[9] H. W. J. Blote, E. Luijten, and J. R. Heringa, “Ising universality in three dimensions: a Monte Carlo study,” *J. Phys. A: Math. Gen.*, vol. 28, pp. 6289–6313, 1995.

[10] G. Burns, R. Daoud, and J. Vaigl, “LAM: An Open Cluster Environment for MPI,” *Proceedings of Supercomputing Symposium*, pp. 379–386, 1994.

[11] P. M. Chaikin and T. C. Lubensky, *Principles of Condensed Matter Physics.* Cambridge University Press, New York, 1995.

[12] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, E. E. Santos, K. E. Schauser, R. Subramonian, and T. von Eicken, “LogP: A Practical Model of Parallel Computation,” *Communications of the ACM*, vol. 37, no. 11, 1996.

[13] S. P. Dandamudi and D. L. Eager, “Hierarchical interconnection networks for multicomputer systems,” *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 786–797, 1990.

[14] A. Gonis, P. P. Singh, P. E. A. Turchi, and X. G. Zhang, “The Use of the Ising Model in the Study of Substitutional Alloys,” *Physics Review*, vol. B51, no. 2122, 1995.

[15] J. D. Gunton and M. Droz, *Introduction to the Theory of Metastable and Unstable States.* New York: Springer-Verlag, 1983.

[16] D. W. Heermann and A. N. Burkitt, *Parallel Algorithms in Computational Science*, vol. 24 of *Springer Series in Information Sciences.* Springer-Verlag, 1991.

[17] K. Huang, *Statistical Mechanics.* New York: J. Wiley and Sons Inc., 1963.

[18] E. Ising *Z. Physik*, vol. 31, no. 253, 1925.

[19] R. M. Karp, A. Sahay, E. E. Santos, and K. E. Schauser, “Optimal Broadcast and Summation in the LogP Model,” *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1993.

[20] S. E. Koonin, *Computational Physics.* Benjamin/Cummings, 1986.

[21] N. Metropolis and A. W. Rosenbluth, “Equation of State Calculations by Fast Computing Machines,” *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[22] O. G. Mouritsen, *Computer Studies of Phase Transitions and Critical Phenomena.* Springer Series in Computational Physics, Springer-Verlag, 1984.

[23] G. Muthukrishnan and E. E. Santos, “On Simulating Hierarchical Clusters for Performance of Ising Spin Systems,” *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 2004, (to appear).

[24] R. K. Pathria, *Statistical Mechanics.* Pergamon Press, New York, 1972.

[25] E. E. Santos, “Optimal and Near-Optimal Algorithms for k-Item Broadcast,” *Journal of Parallel and Distributed Computing*, vol. 57, no. 2, pp. 121–139, 1999.

[26] E. E. Santos, J. M. Rickman, G. Muthukrishnan, and S. Feng, “Efficient Algorithms for Parallelizing Monte Carlo Simulations for 2-D Ising Spin Models,” *Parallel Algorithms and Applications*, to appear.

[27] E. E. Santos and G. Muthukrishnan, “Efficient Simulation Based on Sweep Selection for 2-D and 3-D Ising Spin Models on Hierarchical Clusters,” *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004.

[28] J.-S. Wang and R. H. Swendsen, “Cluster Monte Carlo Algorithms,” *Physica A*, vol. 167, pp. 565–679, 1990.

[29] U. Wolf, “Collective Monte Carlo Updating for Spin Systems,” *Physical Review Letters*, vol. 62, pp. 361–364, 1989.