8-2009

Acceleration of Spiking Neural Networks on Multicore Architectures

Rommel Jalasutram
Clemson University, rjalasu@clemson.edu

ACCELERATION OF SPIKING NEURAL NETWORKS ON MULTICORE ARCHITECTURES

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

---

by
Rommel Jalasutram
August 2009

Accepted by:
Tarek Taha, Committee Chair
Walter Ligon
Mark Smotherman

ABSTRACT

The human cortex is the seat of learning and cognition. Biological scale implementations of cortical models have the potential to provide significantly more power problem solving capabilities than traditional computing algorithms. The large scale implementation and design of these models has attracted significant attention recently. High performance implementations of the models are needed to enable such large scale designs. This thesis examines the acceleration of the spiking neural network class of cortical models on several modern multicore processors. These include the Izhikevich, Wilson, Morris-Lecar, and Hodgkin-Huxley models. The architectures examined are the STI Cell, Sun UltraSPARC T2+, and Intel Xeon E5345. Results indicate that these modern multicore processors can provide significant speed-ups and thus are useful in developing large scale cortical models.

The models are then implemented on a 50 TeraFLOPS 336 node PlayStation 3 cluster. Results indicate that the models scale well on this cluster and can emulate 10^8 neurons and 10^{10} synapses. These numbers are comparable to the large scale cortical model implementation studies performed by IBM using the Blue Gene/L supercomputer. This study indicates that a cluster of PlayStation 3s can provide an economical, yet powerful, platform for simulating large scale biological models.

DEDICATION

This thesis is dedicated to my loving parents, my brother and my friends Ahalya, Sumod, Archana, Biswa, Achal and Vishrut.

ACKNOWLEDGMENTS

The work presented in this thesis would not have been possible without the help and support of many. I would like to acknowledge the members of my committee, and especially Dr Tarek Taha for his guidance, help and support. Next, I would like to acknowledge my fellow graduate students who have spent a lot of time discussing about the work contained herein. I would especially like to thank Sumod Mohan, Pavan Yalamanchali, Mohammad Ashraf and Kenneth Rice in this regard.

Finally, I would also like to acknowledge Dexter Stowers and Dr Wayne Madison for the funding support I received that greatly aided me during this research.

TABLE OF CONTENTS

Table of Contents (Continued)

Page

LIST OF TABLES

LIST OF FIGURES

List of Figures (Continued)

Figure Page

CHAPTER ONE

INTRODUCTION

The human brain can perform complex cognitive tasks at a much faster rate than silicon based processors, despite the fact that neurons are much slower than the transistors used to design the processors. This is primarily because of the massive parallel processing employed in the neocortex, which is the main part of the brain dealing with learning and cognition. This is the outer layer of the human brain and is approximately the size of a large unfolded dinner napkin. It is estimated to consist of approximately 10^{11} neurons and 10^{14} connections between the neurons. Each neuron is connected to a large set of neurons through extensions called dendrites and axons. Neurons communicate with each other by sending electrical pulses. These pulses are generated by the exchange of ions between the neurons.

There has been a strong interest amongst researchers to develop large parallel implementations of neuron models on the order of animal or human brains. At this scale, the models have the potential to provide much stronger inference capabilities than current generation computing algorithms [1]. A large domain of applications would benefit from the stronger inference capabilities including speech recognition, computer vision, textual and image content recognition, robotic control, and data mining.

Large scale models however require significant computing power to implement. Several research groups are currently examining large scale implementations of neuron based models [25] [23] and cortical based models [17] [42]. Such large scale implementations require high performance resources to run the models at reasonable

speeds. Lansner et. al. [2] has shown that mouse sized cortical models developed on a cluster of commodity computers are computationally bound rather than communication bound. Thus the acceleration of neuron models on modern multicore architectures could provide significant benefits for the development of large scale cortical models. Multicore processors are currently the norm in the computing industry because it is difficult to increase the performance gain of single core processors as we have reached the limits on frequency scaling. However, the implementation of several recent cortical models on clusters of multicore processors has not yet been investigated.

This thesis examines the acceleration of spiking neural network models on modern multicore processors. Spiking neural network models are the third generation of neural networks and are considered to be one of the most biologically accurate models. Izhikevich compares a set of 11 spiking neuron models [3] in terms of their biological accuracy and computational load. He shows that four of the more biologically accurate models include (in order of biological accuracy) the Hodgkin-Huxley [4], Izhikevich [5], Wilson [6], and Morris-Lecar [7] models. Of these, the Hodgkin-Huxley model is the most computationally intensive, while the Izhikevich model is the most computationally efficient.

The four spiking neural networks examined are based on a character recognition model. The character recognition model was taken from [15]. This model was adapted from the two layer spiking neuron network model developed by Gupta and Long [8]. While Gupta utilized the integrate and fire model, this work utilizes the four more biologically accurate spiking models identified by Izhikevich [3]. Izhikevich points out

that the commonly used Integrate and Fire model is one of the least biologically accurate spiking neuron models. He states that “the model cannot exhibit even the most fundamental properties of cortical spiking neurons, and for this reason it should be avoided by all means” [3]. The models utilize pulse coding to mimic the high speed recognition taking place in the mammalian brain [9] and spike timing dependent plasticity (STDP) for training [10]. The models are trained to recognize a set of 48 images of characters.

The multicore architectures examined in this study include the 8+1 core Sony/Toshiba/IBM (STI) Cell broadband engine [11], the Intel Xeon E5345 processor, and the Sun UltraSPARC T2+ processor [12]. The platform used to examine the Cell was a Sony PlayStation 3. In case of the Xeon, the platform used was a dual socket × quad-core Intel Xeon E5345. The UltraSPARC T2+ platform used was a dual-socket × eight-core UltraSPARC T2+ T5140 server.

IBM is currently utilizing a 32,768 processor Blue Gene/L to simulate a spiking network based model [25], while EPFL and IBM are utilizing an 8,192 processor Blue Gene/L system to simulate a sub-neuron based cortical model [23]. The PetaVision project announced in June 2008 is utilizing the Roadrunner supercomputer to model the human visual cortex [13].

The Air Force Research Laboratory (AFRL) in Rome, NY has set up a 336 STI Cell multicore processor based Sony PlayStation 3’s (PS3s) primarily to examine the large scale implementations of neuromorphic models [40]. This cluster is capable of providing a performance of 51.5 TF and cost about $361K to set up (of which only 37%

is the cost of PS3s). This is significantly more cost effective than an equivalently performing cluster based on Intel Xeon processors [40]. This thesis also examines the scaling and performance of the spiking neural models on this cluster.

The main contributions of this work are:

1. A study of the parallelization of four biologically accurate neuron models. Both thread and data level parallelization are examined.

2. An evaluation of the multicore implementations of the models. The performance of the models on three multicore platforms (a Cell based Sony PS3, a Sun dual-socket × eight-core UltraSPARC T2+ T5140 server, and a dual socket × quad-core Intel Xeon E5345) is examined. Several network sizes were implemented to examine the effect of scaling the models.

3. A study of the scalability of these models on the AFRL PS3 cluster.

Results indicate that optimized parallel implementations of the models can provide significant speed-ups on multicore architectures. The highest speed-ups were observed for the Hodgkin-Huxley and Morris-Lecar models. For the largest size implemented, these were 7.87 and 8.08 on the Intel Xeon platform utilizing 8 cores, 7.01 and 5.74 on the PS3 utilizing 6 cores, and 1.48 and 2.66 on Sun UltraSPARC T2+ T5140 utilizing 15 cores. The speed-ups are with respect to a vectorized single thread implementation on the Intel Xeon. Please note that [14] [15] present preliminary versions of the results in the thesis. The speed-up on all the processing platforms increased and saturated as the network size was increased for the Morris-Lecar and Hodgkin-Huxley

models. For the Izhikevich and Wilson models, the speed-up decreased with the increase in network size. This was due to the fact that the models were memory intensive and had a low compute-to-I/O ratio.

Large scale implementation results indicate that the models scale almost linearly on the PS3 cluster. Equivalents of 10^8 neurons were modeled along with about 10^{10} synapses. A mouse cortex in comparison contains about 1.6×10^7 neurons and 1.6×10^{11} synapses [17]. The Blue Gene was able to simulate a rat cortex (55×10^6 neurons and 4.42×10^{11} synapses) at near real time. However the cost of AFRL cluster is significantly lower than the one used in [25]. This indicates that the 336 node PS3 cluster provides a highly economical, yet powerful, platform for neuromorphic simulations.

This thesis is organized in the following manner: chapter 2 discusses the related work, chapter 3 and 4 discuss the spiking models and the multicore architectures considered. Chapter 5 and 6 discuss the network, parallelization techniques and optimizations used. Chapter 7 describes the experimental set up. Chapter 8 presents the results for individual multicore architectures. Chapter 9 describes the cluster implementation in detail and presents the results and its biological relevance. Chapter 10 concludes the thesis.

CHAPTER TWO

RELATED WORK

Several groups have examined the acceleration of applications on multicore processors. Williams et. al. [27] examined the acceleration of several scientific computing kernels on the Cell processor and found good performance gains over other non-multicore architectures. Other studies have also shown that the Cell processor can provide high performance as well [28], [29]. Williams et. al. have also compared the performance of some recent multicore architectures for sparse matrix-vector multiplication applications and have found that the Cell and Sun UltraSPARC T2+ processors provide good speed-ups [30]. Khan et. al. [31] describes simulations of an ARM based multicore processor for the acceleration large scale spiking neural network models. Till date, there has been no study examining the acceleration of spiking network models on various multicore processors.

Several groups are currently developing biological scale implementations of spiking networks, but are generally not examining the applications of these systems (primarily as they are modeling large scale neuronal dynamics seen in the brain). The Swiss institution EPFL and IBM are developing a highly biologically accurate brain simulation [23] at the sub neuron level. They have utilized the Hodgkin Huxley and the Wilfred Rall [24] models to simulate up to 100,000 neurons on an IBM BlueGene/L supercomputer. At the IBM Almaden Research Center, Ananthanarayanan and Modha [25] utilized the Izhikevich spiking neuron models to simulate 55 million randomly

connected neurons (equivalent to a rat-scale cortical model) on a 32,768 processor IBM BlueGene/L supercomputer. Johansson et. al. simulated a randomly connected model of 22 million neurons and 11 billion synapses using an 8,192 processors IBM BlueGene/L supercomputer [2]. Izhikevich developed a very large model of the thalamocortical system and studied its behavior [26]. The neural connections in these studies are random and the networks do not identify any patterns.

Several groups have studied image recognition using spiking neural networks. In general, these studies utilized integrate and fire model. Johansson and Lansner developed a large cluster based spiking network simulator of a rodent sized cortex [17]. They tested a small scale version of the system to identify 128×128 pixel images. Baig [18] developed a temporal spiking network model based on integrate and fire neurons and applied them to identify online cursive handwritten characters. Gupta and Long [8] investigated the application of spiking networks for the recognition of simple characters. Other applications of spiking networks include instructing robots in navigation and grasping tasks [19], recognizing temporal sequences [20][21], the robotic modeling of mouse whiskers [22]. Thorpe developed SPIKENET [16], a large spiking neural network simulation software. The system can be used for several image recognition applications including identification of faces, fingerprints, and video images.

CHAPTER THREE

MODELS EXAMINED

Spiking neural models capture neuronal behavior more accurately than traditional neural models. A neuron consists of three functionally distinct parts called dendrites, axons, and a soma. Each neuron is typically connected to over 10,000 other neurons [32]. The dendrites of a neuron collect input signals from other neurons, while the axons send output signals to other neurons. Signals coming in along dendrites cause changes in the ionic levels within the soma, which in turn causes the neuron's membrane potential to change. If this membrane potential crosses a certain threshold, the neuron is said to have "fired" or "spiked". In these events the membrane potential rises rapidly for a short period of time (a spike) and causes electrical signals to be transmitted along the axons of the neuron to other neurons connected to it [33]. Spiking is the primary mechanism by which neurons send signals to each other. Over the last 50 years, several models have been proposed that capture the spiking mechanism within a neuron.

In this paper, four of the more biologically accurate spiking neuron models (as listed by Izhikevich [3]) are examined. These are the Hodgkin-Huxley [4], Izhikevich [5], Wilson [6], Morris-Lecar [7] models. The Hodgkin–Huxley model is considered to be one of the most biologically accurate spiking neuron models. All four of the models can reproduce almost all types of neuron responses that are seen in biological experiments. All but the Izhikevich model are based on biologically meaningful parameters (such as activation of Na and K currents, and inactivation of Na currents). Table 3.1 compares the computation properties of the four models. The Hodgkin–Huxley model utilizes

exponential functions, while the Morris-Lecar model uses hyperbolic functions. These contribute to the higher flops needed for these two models. Note that the four models are not tuned to replicate one specific type of neuron. Thus the number of simulation cycles for the models do vary. This however does not impact the inference carried out by the models in this study.

Table 3.1. Spiking Network Properties

Model	Differential Equations	Variables updated each cycle	Flops / neuron (Euler)	Flops / neuron (Runge-Kutta)	Cycles / recognition
Izhikevich	2	2	13	70	12
Wilson	4	7	37	152	29
Morris-Lecar	2	5	187	297	15
Hodgkin-Huxley	4	16	265	442	373

Two common methods to implement the differential equations in these models include the Euler and the Runge-Kutta approaches. While the Runge-Kutta approach provides more accurate results, the Euler method is the most common approach for implementing the differential equations as it has a significantly lower computational load. This study primarily utilizes the Euler approach, although the Runge-Kutta approach is examined as well. The flop counts for both the Euler and the Runge-Kutta approach are listed in Table 3.1. These values are based on the implementations of the four models.

Izhikevich Model

Izhikevich proposed a new spiking neuron model in 2003 [5] that is based on only two differential equations (eq. 1, 2). This model requires the least computations of all the models examined, because it needs fewer flops per neuron update and requires fewer neuron updates to be carried out per simulation run time (since the simulation time step is higher). However the model can still reproduce almost all types of neuron responses that are seen in biological experiments. The four constant parameters (*a*, *b*, *c* and *d*) can be initialized differently to allow modeling of various neural responses. A time step of 1 ms was utilized (as was done by Izhikevich in [5]). Figure 3.1 shows the spikes produced with this model.

$$\frac{dV}{dt} = 0.04V^2 + 5V + 140 - u + I \qquad (1)$$

$$\frac{du}{dt} = a(bV - u) \qquad (2)$$

$$\text{if } V \geq 30 \text{ mV, then } \begin{cases} V \leftarrow c \\ u \leftarrow u + d \end{cases}$$

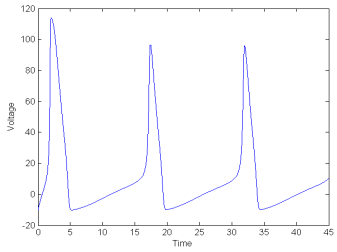

Figure 3.1. Spikes produced with the Izhikevich Model

The following parameters were used for the Izhikevich model: Excitatory neurons: *a* = .02, *b* = 0.2, *c* = -55, *d* = 4; Inhibitory neurons: *a* = 0.06, *b* = 0.22, *c* = -65, *d* = 2, time step = 1 ms.

Wilson Model

The Wilson model [6], proposed in 1999, requires four differential equations (equations 3 to 6). The model has more number of parameters than the Izhikevich model. Tuning these parameters allow the model to exhibit almost all neuronal properties. Three of the parameters in the differential equations (T∞, R∞, and m∞) also need to be evaluated each cycle, thus adding a set of three more equations. A time step of 0.01 ms was utilized to update the four differential equations. Figure 3.2 shows some typical spikes produced with this model.

$$\frac{dH}{dt} = (1/45)(-H + 3T) \tag{3}$$

$$\frac{dT}{dt} = (1/14)(-T + T_{\infty}) \tag{4}$$

$$\frac{dR}{dt} = (1/\tau_R)(-R + R_{\infty}) \tag{5}$$

$$\frac{dV}{dt} = (\frac{1}{C})(-m_{\infty}(V - E_{Na}) - 26R(V + E_K) - g_T T(V - E_{Ca}) - g_H H(V + E_K) + I) \tag{6}$$

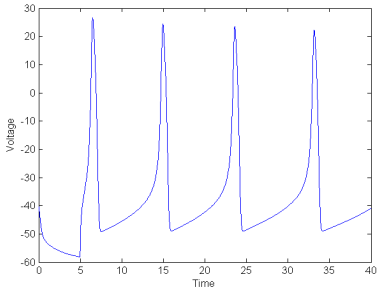

Figure 3.2. Spikes produced with the Wilson model

The following parameters were used for the Wilson model: g_T = 0.1 seimen, g_H = 5 seimen, τ_R = 4.2 ms, C = 1 micro farad, E_{Na} = 0.5, E_K = 0.95, E_{Ca} = 1.2, V = -0.6 mV, R = 0, T = 0, H = 1, time step = 0.01 ms.

Morris-Lecar Model

Cathy Morris and Harold Lecar proposed a two dimensional conductance-based spiking model in 1981 [7]. The model consists of two differential equations (eq. 7, 8). Three of the parameters in the differential equations (m_∞, w_∞, and τ_w) need to be evaluated each cycle, thus adding a set of three more equations. These three equations involve hyperbolic functions, thus making it computationally more expensive than the Izhikevich and Wilson models. This computational load is lower than the Hodgkin-Huxley model however, thus making it popular in neuro-computation communities. A time step of 0.01 ms was utilized to update the two differential equations. Figure 3.3 shows some typical spikes produced with this model.

$$\frac{dv}{dt} = (\frac{1}{C})(I - g_{Ca} m_\infty (V - V_{Ca}) - g_K w(V - V_K) - g_{Leak}(V - V_{Leak})) \quad (7)$$

$$\frac{dw}{dt} = (\frac{1}{\tau_w})(w_\infty - w)\phi \quad (8)$$

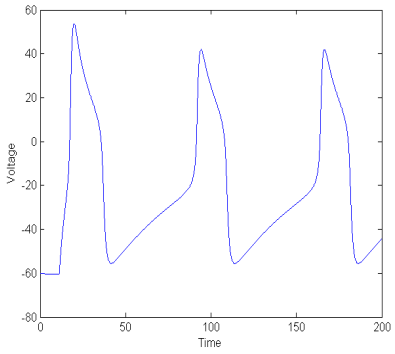

Figure 3.3. Spikes produced with the Morris-Lecar model

The following parameter values were used for the Morris-Lecar model: C = 7, V_K = -84 mV, g_K = 8 mV, V_{Ca} = 120 mV, g_{Ca} = 4.4 seimen, V_{Leak} = -60, g_{Leak} = 2 seimen, V_1 = -1.2, V_2 = 18, V_3 = 2, V_4 = 30, φ = 0.04, Time step = 0.01 ms.

Hodgkin-Huxley Model

The Hodgkin–Huxley model [4] was a seminal work in neuron modeling. It consists of four differential equations (eq. 9-12). A set of 10 more equations have to be evaluated each cycle to update parameters used in the differential equations. Four of these equations utilize exponential functions. This makes the Hodgkin-Huxley model the most complex of the four models studied. A time step of 0.01 ms was utilized to update the four differential equations as this is the most commonly used value. Figure 3.4 shows the spikes produced with this model. This model has also been used in the detailed large scale neural simulations being carried out by IBM and EPFL [23].

$$\frac{dv}{dt} = (\frac{1}{C})\{I - g_K n^4 (V - E_K) - g_{Na} m^3 h (V - E_{Na}) - g_L (V - E_L)\} \quad (9)$$

$$\frac{dn}{dt} = (n_\infty (V) - n) / \tau_n (V) \quad (10)$$

$$\frac{dm}{dt} = (m_\infty (V) - m) / \tau_m (V) \quad (11)$$

$$\frac{dh}{dt} = (h_\infty (V) - h) / \tau_h (V) \quad (12)$$

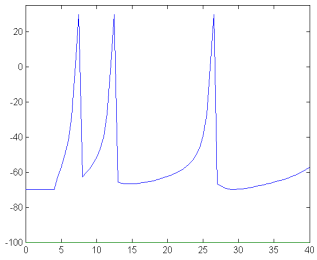

Figure 3.4. Spikes produced with the Hodgkin-Huxley model

The following parameter values were used for the Hodgkin-Huxley model: g_K = 36 seimen, g_{Na} = 120 seimen, g_L = 0.3 seimen, E_K = -12 mV, E_{Na} = 115 mV, E_L = 10.613 mV, V = -10 mV, V_K = 0 mV, V_{Na} = 0 mV, V_L = 1 mV, time step = 0.01 ms.

CHAPTER FOUR

MULTICORE ARCHITECTURES EXAMINED

Due to microelectronics constraints, such as wire delays and power densities, modern processors are increasing performance by exploiting parallelism rather than increasing clock frequencies. As a result, multicore processors have become widespread. It is expected that in the future, processors with hundreds of cores will become available. In addition to multiple cores, some processors are exploiting vector parallelism to improve performance.

This thesis examines three of the leading chip multicore processor (CMP) designs in context of neuromorphic algorithms – in particular biologically inspired spiking neural networks. This chapter briefly describes the processors and platforms with their individual architectural features: the dual socket × quad-core Intel Xeon E5345 (Clovertown); the dual socket × eight core hardware multithreaded Sun UltraSPARC T2+ T5140 (Victoria Falls); and the heterogeneous single socket × eight-SPE Cell processor ased Sony PlayStation 3. Overviews of the configurations and characteristics appear in Table 4.1.

Table 4.1. Architectural summary of evaluated platforms. Top: per core characteristics. Bottom: SMP characteristics

Core Architecture	Intel Core2	Sun UltraSPARC T2+	IBM PPE	SPE
Type	Superscalar out-of-order	MT dual issue	MT dual issue	SIMD dual issue
Clock (GHz)	2.33	1.16	3.2	3.2
Local store	-	-	-	256 KB
L1 Data Cache per core	32 KB	8 KB	32 KB	-
L2 Cache per core	-	-	512 KB	-
System	Xeon E5345 (Clovertown)	UltraSPARC T2+ T5140 (Victoria Falls)	Sony PlayStation 3	
# Sockets	2	2	1	
Cores per Socket	4	8	1	8
DRAM Capacity	12 GB	64 GB	2 GB	
DRAM bandwidth (GB/s)	21.33 (read)	42.66 (read)	25.6	
	10.66 (write)	21.33 (write)		
Threading	Pthreads	Pthreads	Libspe2	
Compiler	gcc	cc	Gcc	spu-gcc

Intel Xeon E5345 (Clovertown)

Two dual-core Xeon chips are paired onto a single multi-chip module. Each core is based on Intel's Core2 micro-architecture (Woodcrest). It utilizes lower voltage and is more power efficient. Each core thus runs at a lower frequency (2.33 GHz). The

individual cores can fetch and decode up to four instructions per cycle, and can execute six micro-ops per cycle. Each core has a 128b adder and a 128b multiplier. This enables Xeon to support Single Instruction Multiple Data (SIMD). Streaming SIMD Extensions 3 (SSE3) instructions make use of the 128b registers. Its peak double precision performance is 9.3 GFlops/s. Each core includes a 32 Kb, 8-way L1 cache, each chip (two cores) has a shared 4MB, 16 way L2 cache.

This thesis evaluates the performance of Clovertowns' available on the Palmetto Cluster at Clemson University. This Xeon platform is a two processor based platform. Thus, the platform can use up to eight cores (four from each processor) for computation. A simplified version of the Clovertown architecture as seen in [41] is shown in Figure 4.1.

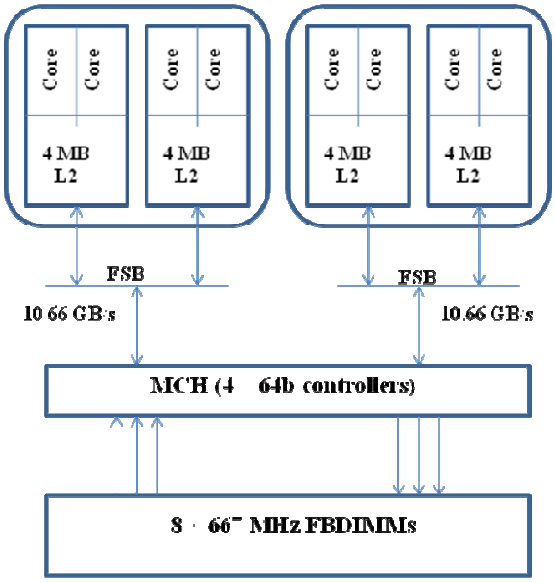

Figure 4.1. Dual-socket × quad-core Intel Xeon E5345 (Clovertown) processor architecture

Sun UltraSPARC T2+ T5140 (Victoria Falls)

Being the industry's first "system-on-a-chip" (SoC), the UltraSPARC T2 Plus [12] packs the most cores and threads available on any general purpose processor. The processor has eight cores, each capable of supporting two groups of four hardware thread contexts (referred to as Chip Multi-Threading or CMT). Thus each core can support up to eight threads. Unlike other architectures which improve performance by utilizing larger registers, this processor scales its performance through multi-threading. The SoC has 10 GB Ethernet networking.

The UltraSPARC T2+ has a dedicated Floating Point Unit (FPU) for each core (shared among eight threads). The FPU does not have a fused multiply add (FMA) functionality. Per-core and per-socket performance is 1.16 GFlops/s and 9.33 GFlops/s. Each core has 8kB write-through L1 cache, 16 KB of instruction cache and is connected to a 4MB shared L2 cache via an on-chip crossbar switch. The UltraSPARC T2+ has no hardware pre-fetching. Also, software pre-fetching places the data in L2 cache. The platform utilized for experimentation was a Sun Enterprise T5140 server. It consists of two UltraSPARC T2+ processors each operating at 1.2 GHz. Thus, the system has 16 available cores (eight from each processor). A simplified version of the Sun UltraSPARC T2+ T5140 architecture as seen in [41] is shown in Figure 4.2.

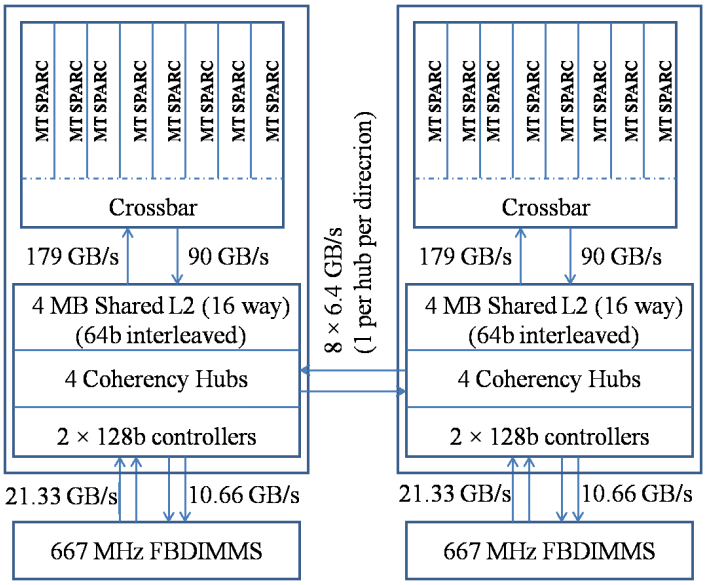

Figure 4.2. Dual-socket × eight-core Sun UltraSPARC T2+ T5140 (Victoria Falls) processor architecture

STI Cell

The STI Cell Broadband Engine [11] processor is the heart of the Sony PlayStation 3 (PS3) video gaming console. The architecture is heterogeneous. It consists of a Power Processing Element (PPE) and eight Synergistic Processing Elements (SPEs). Of the eight SPEs, only six are enabled on the PlayStation 3 platform. The PPE handles the operating system and administrative functionalities. The SPEs each use a disjoint software controlled local memory. Each SPE has 256 KB of local memory.

The software controlled DMA (Direct Memory Access) engine is efficient and helps fetch data asynchronously from DRAM into the local store. As fetching the data is software controlled, programming the Cell architecture is more complex than

conventional architectures. This approach eliminates conflict misses and write fills, but capacity misses must be handled by the programmer due to the limited amount of local store space available.

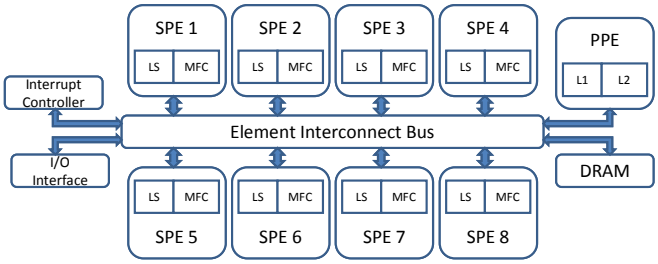

Figure 4.3. STI Cell processor architecture

Each SPE has 128b wide registers. This enables the Cell to take advantage of vector processing as well. The PPE and SPE operate at 3.2 GHz. Each SPE is capable of processing up to four instructions in parallel each cycle (eight if considering FMA). The SPEs utilize in-order execution and have no branch prediction hardware. High compute-to-I/O ratios are needed to achieve the full potential of the Cell processor [34].

One of the key features of the cell processor is to transfer complexity from hardware to software. Thus the cores utilize in-order execution with no branch prediction hardware. Instead of a processor controlled cache, the local store is programmer managed. This ensures that only necessary data is brought in. The Cell platform used in this study is a Sony PlayStation 3 cluster available at the Arctic Region Supercomputing Center (ARSC). A simplified version of Cell processor architecture is shown in Figure 4.3.

CHAPTER FIVE

NETWORK DESIGN

Gupta and Long [8] presented a character recognition network based on the integrate-and-fire neuron model [35]. That network was adapted in this thesis to evaluate the four spiking neuron models under consideration. Four versions of the image recognition network were developed (corresponding to the four spiking models studied) where the main difference was in the equations utilized to update the potential of the neurons. The parameters utilized in each case are specified in chapter three.

The network consisted of two layers, where the first layer acted as input neurons and the second layer as output neurons. Input images were presented to the first layer of neurons, with each image pixel corresponding to a separate input neuron. Thus the number of neurons in the first layer is equal to the number of pixels in the input image. Binary input images were utilized in this study. The number of output neurons was equal to the number of training images. Each input neuron was connected to all the output neurons. A prototype of this network is shown in Figure 5.1.

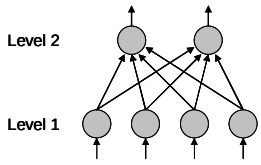

Figure 5.1. Network used for testing spiking models

Each neuron has an input current that it uses to evaluate its membrane potential. If the membrane potential crosses a certain threshold during a cycle, the neuron is considered to have fired or spiked. In case of a level one neuron, the input current is zero if the neuron's corresponding pixel in the input image is "off". If the pixel is "on", a

constant current is supplied to the input pixel. A level two neuron's overall input current is the sum of all the individual currents received from all the level one neurons connected to it. This input current for a level two neuron is given by equation 13 below:

$$I_j = \sum_i w(i,j) f(i) \tag{13}$$

where

w is a weight matrix where *w(i,j)* is the input weight from level one neuron *i* to level two neuron *j*.

f is a firing vector where *f(i)* is 0 if level one neuron *i* does not fire, and is 1 is the neuron does fire.

The elements of the weight matrix *w* are determined through a training process where a set of training images are presented sequentially to the input neurons. The weight matrix elements of each output neuron are updated using the STDP rule each cycle. The weight matrix thus obtained is used to determine the input current to each of the output neuron.

In the testing phase, an input image is presented to the input neurons and after a certain number of cycles, one output neuron fires, thus identifying the input image. During each cycle, the level one neurons are first evaluated based on the input image. As and when a level one neuron fires, its weight corresponding to each output neuron is added to the current of each of the output neurons it is connected to. At the end of the cycle, the input current for two level neurons is the sum of the weights from the level one neurons that have fired the current cycle. This input current is used in the following cycle to evaluate the neuron membrane potentials of the level two neurons. This process is

described in detail in algorithm 1. This thesis studies the acceleration of the recognition phase of each network on multicore processors.

Algorithm 1: The testing phase for the spiking neuron image recognition model

```
1.  Repeat till a level two neuron fires:
2.      For all level one neurons:
3.          Read input current
4.          Calculate neuron membrane voltage
5.          If neuron fires, upgrade the level 2 input current
    —Barrier—
6.      For all level two neurons:
7.          For each non zero number of firing from level one (from previous cycle),
8.              Calculate total level 2 input current
9.          Calculate neuron membrane voltage
10.         If neuron fires, output is produced
    —Barrier—
```

CHAPTER SIX

PARALLELIZATIONS AND OPTIMIZATIONS

Network Parallelization

All four models studied in this paper are evaluated using the network developed in chapter five. In general one thread was created for each core at the start of the program on the different multicore architectures examined (STI Cell, Intel Xeon E5345, and Sun UltraSPARC T2+). Since the UltraSPARC T2+ supported multiple threads per processor it was tested it with multiple threads per core. Each thread was assigned tasks intermittently by the master thread and was terminated only at the end of the program. Creating threads only once and letting them run for the lifetime of the program significantly reduces the thread creation overhead; this issue has been discussed at great length in several recent papers including [11] [36].

Since all the nodes in each layer are independent of each other, they can be evaluated in parallel. Individual threads were created for each core on all the processors studied. The neurons in the first layer (input layer) were distributed evenly across the threads generated. Since all the neurons utilize the same set of computations, processors with SIMD support available (Cell and Xeon) exploited data level parallelism by evaluating sets of four neurons simultaneously. As the level two neurons are only 48 in number, they were evaluated on the PPE in case of Cell and on the first thread in case of Xeon and UltraSPARC T2+. This improved the scalability of the model.

As shown in Algorithm 1, two barriers were utilized to allow all the threads to complete the nodes in each layer of the network. A firing vector was used in the

preliminary implementations. It stored the indices of the level one neurons fired in the previous cycle. This was later used to calculate the current for the level two neurons. This approach required the sharing of the firing vector among all the threads. While migrating to a large scale cluster implementation, this approach was a lot harder to implement. Thus this approach was discarded.

In the current approach, if a level one neuron fired, its corresponding weights were added to the level two current data structure for the corresponding thread. At the start of the next cycle, the currents from all the threads are summed by which ever processor was evaluating the level two neurons. This eliminated the need for having an extra data structure like the firing vector and eliminated concurrency issues. It also improved the scalability of the implementation.

The Cell implementation used mailboxes for synchronization while the Sun and Intel implementations used pthread barriers.

Optimizations

This section explains the optimizations that were considered and the order in which they were considered. The optimizations are designed to be effective both on conventional cache-based multicore processors and the Cell architecture. The techniques implemented include multi-threading, vectorization, double buffering, software pipelining, pre-fetching and loop optimizations.

Multi-Threading

The first technique in the optimization process utilizes the fact that all the processors are multicore by exploiting thread-level parallelism. The neurons at level one are distributed evenly across the threads. This ensures that each thread is fairly load balanced. The implementation does this dynamically depending on the size of the input image. Threading reduces the instruction latency thus improving the performance. The threads are created using the POSIX thread library on the UltraSPARC T2+ and Xeon architectures while the Cell implementation utilized the libspe2 library.

Vectorization

The Cell SPE and the Xeon architectures have 128b registers. This enables them to exploit any data-level parallelism present in the application. Four single precision floating point operations or two double precision floating point operations can be evaluated with a single instruction on these registers. All the loads are 16 bytes and must be 16 byte aligned. It is to be noted that it takes 6 cycles for a single precision floating point operation and 13 cycles for a double precision floating point operation on the Cell. Thus the implementations used single precision floating point values. The Cell can perform a fused multiply-add (FMA).

On the Intel, SSE3 instructions were used for vectorizing the operations. On x86 architectures, SSE3 provides an unaligned load but adds some performance penalty. Thus all the data structures were 16 byte aligned. The Morris-Lecar and the Hodgkin-Huxley models make use of hyperbolic and exponential functions. As the SSE3 library did not

have exponential functions, the implementation made use of the Universal SIMD Library developed by Helmut Dersch [37].

Loop Optimization

Maximizing in-core performance is essential. The Cell processor does not have branch prediction hardware. In order to minimize the branch prediction on the Cell, the loops were unrolled by a factor of eight. This technique is useful for the Cell as it utilizes in-order execution, but is of little use in case of out-of-order superscalar processors. This also meant that the function calls had to be in-lined. The code on the Cell was further optimized by minimizing unnecessary branches.

Even Pipeline															
1545:fma$13,$63,$83,$13	x									x	x	x	x	x
1546:fma$14,$58,$83,$14	x	x									x	x	x	x
1547:fma$15,$57,$83,$15	x	x	x									x	x	x
1548:fma$16,$59,$83,$16	x	x	x	x									x	x
1550:fma$25,$25,$76,$42	x	x	x	x	x									x
1552:fms$2,$84,$2,$26	x	x	x	x	x	x								
1554:fa$17,$81,$17		x	x	x	x	x	x							
1556:fma$18,$18,$76,$43			x	x	x	x	x	x						
1557:fma$19,$19,$76,$44				x	x	x	x	x						
1558:fma$20,$20,$76,$45					x	x	x	x	x					
1559:fma$21,$21,$76,$46						x	x	x	x	x				
1563:fma$22,$22,$76,$47							x	x	x	x	x			
1564:fma$23,$23,$76,$48								x	x	x	x	x		
1565:fma$24,$24,$76,$49									x	x	x	x	x	
1569:fms$3,$84,$3,$27										x	x	x	x	x
1573:fms$4,$84,$4,$28	x										x	x	x	x
1577:fms$5,$84,$5,$29	x	x										x	x	x
1581:fms$6,$84,$6,$30	x	x	x										x	x
1585:fms$7,$84,$7,$31	x	x	x	x										x
1589:fms$8,$84,$8,$32	x	x	x	x	x									x

Figure 6.1. Assembly code for the Wilson model on Cell after loop unrolling, data pre-fetching and software pipelining.

On the Cell, it takes 6 cycles to fetch data into the register. This meant that utilizing indexed data structures would lead to a high percentage of stalls in the instruction pipeline. Thus, the required data was pre-fetched. Software pipelining was

used in combination with loop unrolling to minimize these stalls. Figure 6.1 shows a snippet of the assembly code for the Wilson model after data pre-fetching, loop unrolling and software pipelining. It can be seen from the figure that after these optimizations, there are no stalls in the instruction pipeline.

Memory optimizations

Data transfer on the Cell is explicitly managed by the programmer as opposed to the operating system. The local store's memory is limited to 256 KB. On an average, 16 bytes of memory is required to store the state of a neuron. Thus when evaluating a network of neurons, we need to explicitly DMA in the required data, evaluate the neuron equations, and DMA out the data. If we are to wait for the data to be brought in before we start performing computations, we are limited by the DMA. Thus double buffering was implemented to hide I/O latency with computation. In this process, once the data required for the first iteration has been obtained, the first iteration of the loop can be evaluated simultaneously with DMA data transfer for the second iteration of the loop. This ensures that one is not waiting on data to perform computations.

CHAPTER SEVEN

EXPERIMENTAL SETUP

Three hardware platforms were utilized in this study: one was Intel Xeon E5345 based, one was STI Cell based, and one was Sun UltraSPARC T2+ based. The Intel platform had two quad-core Xeon E5345 processors, used 12 GB of DRAM and was running RedHat 4.1.2-44. Thus, the platform was capable of utilizing eight cores. The STI platform utilized was a Sony PlayStation 3 cluster available at the Arctic Region Supercomputing Center (ARSC), Alaska. The PlayStation 3 has one Cell processor on which six of the eight cores are available for use and contains 256 MB of DRAM. This platform was running Fedora Core 9 with IBM Cell SDK 3.1. The Sun UltraSPARC T2+ platform utilized was a Sun SPARC Enterprise T5140 running Solaris 10. This system contained two UltraSPARC T2+ processors and used 64 GB of DRAM. This system is thus capable of utilizing 16 cores (eight from each processor) and each core supports up to a maximum of eight threads. All the programs were compiled with –O3 optimizations. On the UltraSPARC platform, one core was used for running the operating system while the remaining cores were used to run the spiking neuron models. Each thread was bound to a specific core to ensure process affinity thereby ensuring optimum performance.

Seven networks with varying input image sizes were utilized to examine each of the spiking neural network models. The sizes of level 2 and level 1 neurons for different input sizes are shown in Table 7.1. The overall network structure was kept similar to the design mentioned in Figure 5.1 with two layers of nodes per network. Each of the level 2 neurons was connected to all of the neurons from level 1. The number of input (level 1)

neurons was equal to the number of pixels in the input image. The number of output (level 2) neurons was equal to the number of training images categories.

Table 7.1. Spiking Network Configurations Evaluated

Input image size	Level 1 neurons	Level 2 neurons	Total neurons
480 × 480	230,400	48	230,448
720 × 720	518,400	48	518,448
960 × 960	921,600	48	921,648
1200 × 1200	1,440,000	48	1,440,048
1680 × 1680	2,822,400	48	2,822,448
2160 × 2160	4,665,600	48	4,665,648
2400 × 2400	5,760,000	48	5,760,048

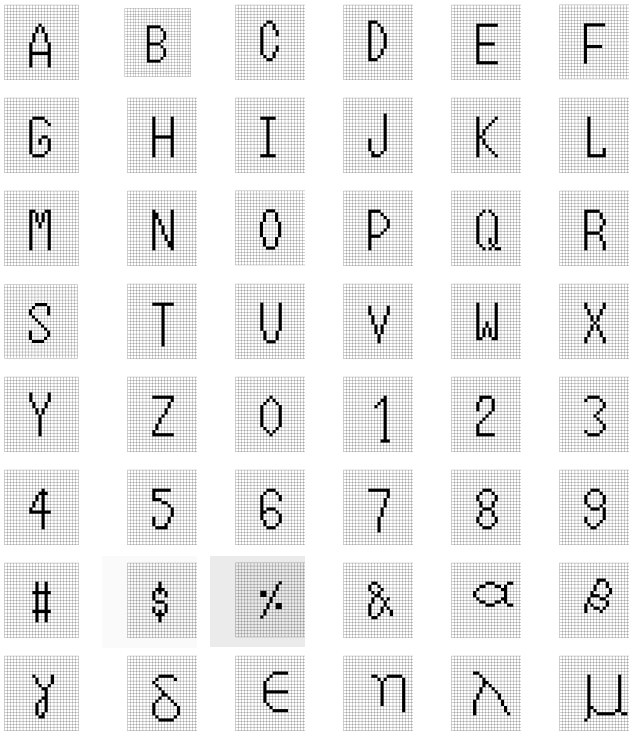

Figure 7.1. Training images utilized. There are 48 24 × 24 pixel images

In this study all the spiking networks of different sizes were trained with to recognize the same number of images. A set of 48 24×24 pixel images were generated initially and were then scaled linearly to the different network sizes. The images represented the 26 upper case letters (A-Z), 10 numerals (0-9), 8 Greek letters, and 4 symbols. Figure 7.1 shows the training images used. The four models were initially developed and tested and trained in MATLAB before being converted to C.

CHAPTER EIGHT

RUN TIME PERFORMANCE

All the models were implemented on the three computing platforms to evaluate the performance of one processor per platform. Unless mentioned, the Euler method was utilized to evaluate the differential equations for each of the models. A vectorized single thread version of the program was developed and tested on the Intel Xeon E5345. The speed-up of the four models running on each of the three computing platforms is shown in Figure 8.1. These speed-ups are relative to the single thread SSE3 implementations of the models on the Intel Xeon E5345.

The Izhikevich and Morris-Lecar models require fewer parameters than the Wilson or Hodgkin-Huxley models to store the state of a neuron (see Table 3.1). Thus, the memory requirement for the former models is less when compared to the latter models. As a result, on the PS3, the largest network implemented for the Izhikevich and Morris-Lecar models had 5,760,048 neurons (2400 × 2400), while the largest network implemented using the Wilson and Hodgkin-Huxley models had 4,665,648 neurons (2160 × 2160). This was due to insufficient virtual memory on the PPE of the PS3. The largest size implemented for each of the models on Xeon and UltraSPARC T2+ platforms had 5,760,048 neurons (2400 × 2400).

Table 8.1 gives the compute-to-I/O ratios for all the models on the PS3. This indicates the number of flops performed per byte of data fetched. It is 0.45 and 0.77 for Izhikevich and Wilson models and 4.6 and 5.52 for Morris-Lecar and Hodgkin-Huxley models. Low compute-to-I/O ratios hinder the performance of multicore processors.

Table 8.1. Compute to I/O ratios (flops per data byte fetched) for models on the PlayStation 3

Model	Euler	Runge-Kutta
Izhikevich	0.45	2.37
Wilson	0.77	3.15
Morris-Lecar	4.60	7.30
Hodgkin-Huxley	5.52	9.21

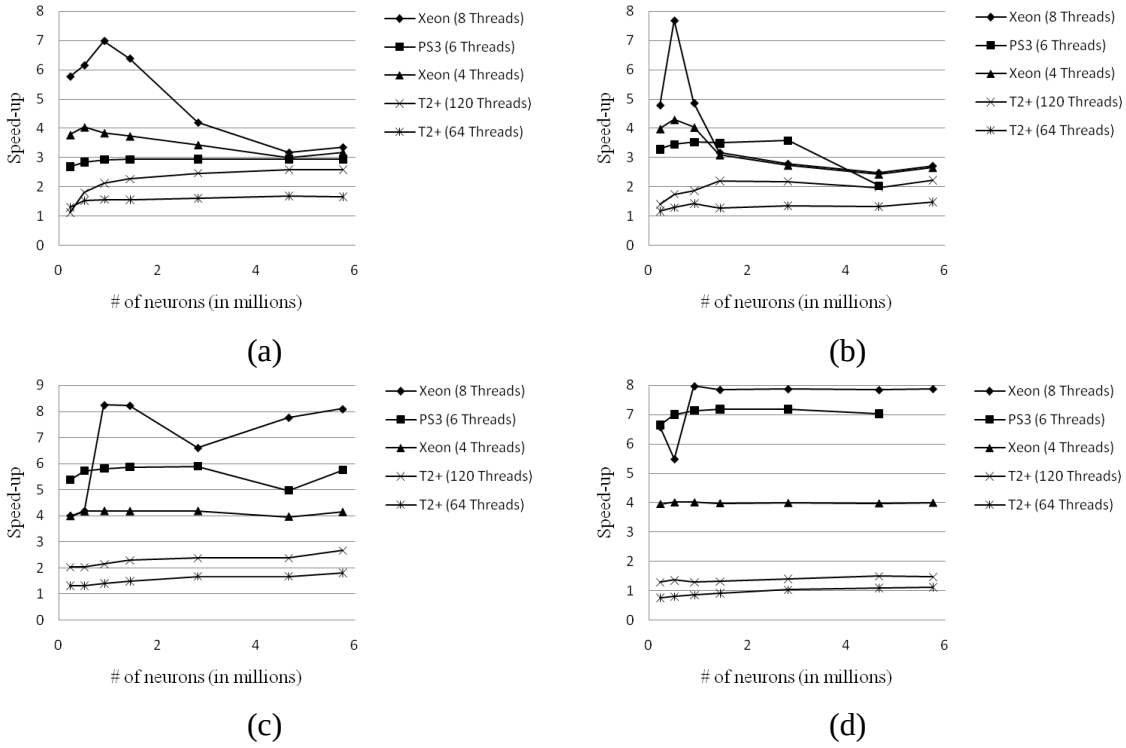

Figure 8.1. Speed-up for Intel Xeon E5345 platform (8 threads), Sony PS3 (6 threads), Intel Xeon E5345 platform (4 threads), Sun UltraSPARC T2+ T5140 (120 and 64) threads over vectorized Intel Xeon single thread for the (a) Izhikevich, (b) Wilson, (c) Morris-Lecar, and (d) Hodgkin-Huxley models.

Figures 8.1 a and 8.1 b show the speed-ups of the Izhikevich and Wilson models on all the platforms with the increase in input image size (network size). On the Cell (6 threads) and UltraSPARC T2+ platforms (64 and 120 threads), the speed-up increases and eventually saturates. In case of the Wilson model, the speed-up decreases only for the 2160 × 2160 (4,665,648 neurons) network on the Cell platform. This probably might be due to hard disk access or excessive cache misses. The Xeon platform speed-up for these models increases initially but eventually decreases. It is to be noted that these models are more memory intensive as they have a low compute-to-I/O ratio (less than one). It has been shown that for similar applications, the Xeon chipset's capabilities limit multi-socket scaling on memory intensive applications [41]. The speed-up declines faster in case of the Wilson model on the Xeon platform as it requires more data to store the state of each neuron. As expected, for these models, maximum speed-up obtained for the largest size was on the Xeon platform using 8 threads while the least speed-up obtained was on the UltraSPARC T2+ platform using 64 threads (eight cores).

Figures 8.1 c & 8.1 d show the speed-ups for the Morris-Lecar and Hodgkin-Huxley models on all the platforms. These models have a high compute-to-I/O ratio (greater than one). The speed-up increases with size and saturates on all the platforms. It is seen that for larger network sizes, these models give the highest performance gain. As expected, for these models, the performance gain is close to 4 and 8 in case of the Xeon platform running 4 and 8 threads. Similarly, the gain on the Cell platform (PS3) is close to 6 for the Morris-Lecar model while it is close to 7 for the Hodgkin-Huxley model. The performance is higher in case of the Hodgkin-Huxley model as it has more equations

making use of the fused multiply add functionality available on the Cell platform. On the other hand, the gain observed on UltraSPARC T2+ platform for Hodgkin-Huxley is lower than that of Morris-Lecar inspite of having the highest compute-to-I/O ratio. This is primarily due to the limited number of dedicated FPUs (one per core shared among eight threads) available on the UltraSPARC T2+. Thus on this platform, the Hodgkin-Huxley model has the least performance gain amongst all the models.

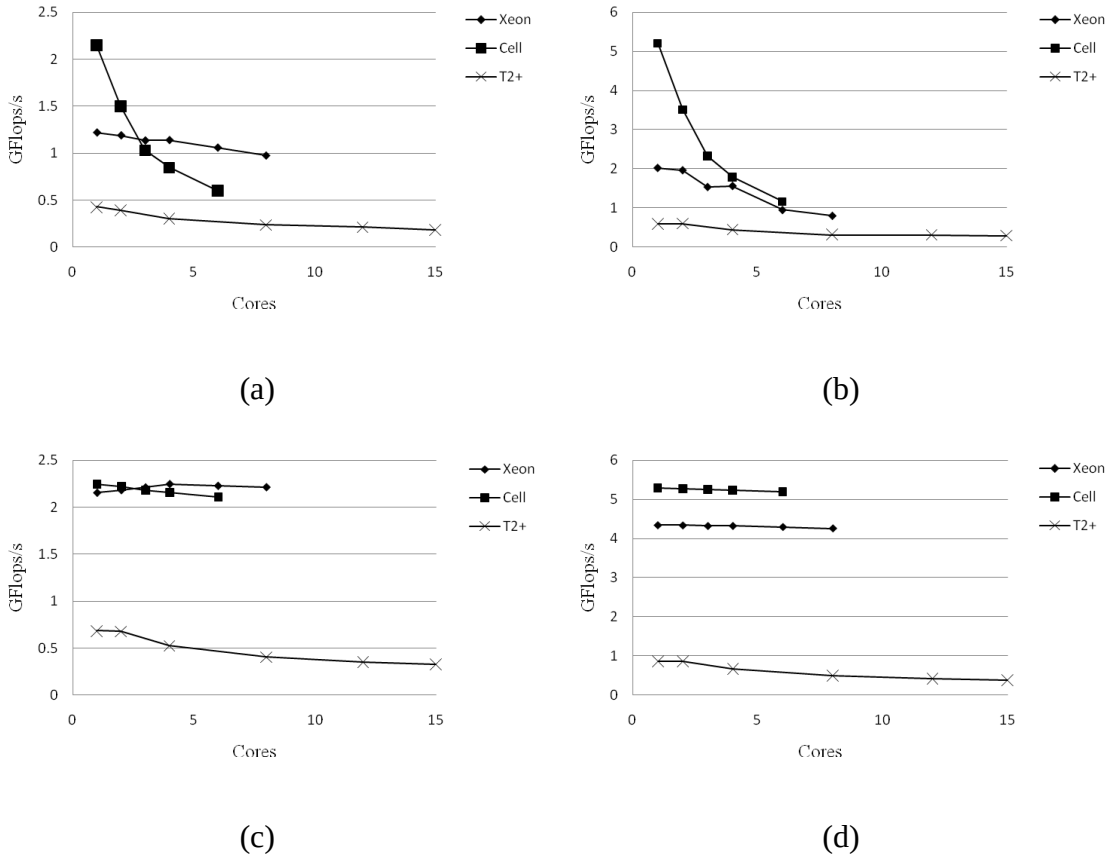

Figure 8.2. Per-core efficiency of (a) Izhikevich, (b) Wilson, (c) Morris-Lecar, and (d) Hodgkin-Huxley models with network size 1200 × 1200 on the Xeon, Cell and UltraSPARC T2+ platforms.

Figure 8.2 shows the per-core efficiency of the platforms for a network with 1,440,048 neurons (1200 × 1200) of all the models. This network size was chosen as this

was the largest network implemented on a single core on each of the platforms. On the UltraSPARC T2+ platform, note that there are always eight threads per-core.

Figures 8.2 a and 8.2 b show the per core efficiency of all the platforms for the Izhikevich and Wilson models. For these models, a single core on the Cell (PS3) platform is faster than that on the Xeon platform. The per-core efficiency on all the platforms declines with the increase in the number of cores for these models. In case of the Sun platform, the decline is fairly gradual. But in case of Xeon and Cell platforms, the decline in performance is very steep. It is seen that the per-core efficiency on the Cell platform declines much faster than on the Xeon platform. This indicates that memory intensive applications with a low compute-to-I/O ratio, do not achieve significant gains with the increase in the number of cores. This also explains why there is a decline in performance gain with the increase in network size for Izhikevich and Wilson models as observed in Figures 8.1 a and 8.1 b.

Figures 8.2 c and 8.2 d show the per core efficiency of all the platforms for the Morris-Lecar and Hodgkin-Huxley models. These models have a high compute-to-I/O ratio. As expected, the per-core efficiency remains almost constant on all the platforms for these models. Such models can make use of all the cores to achieve the best performance. For these models, the I/O overhead is a lower fraction of the overall runtime for these models.

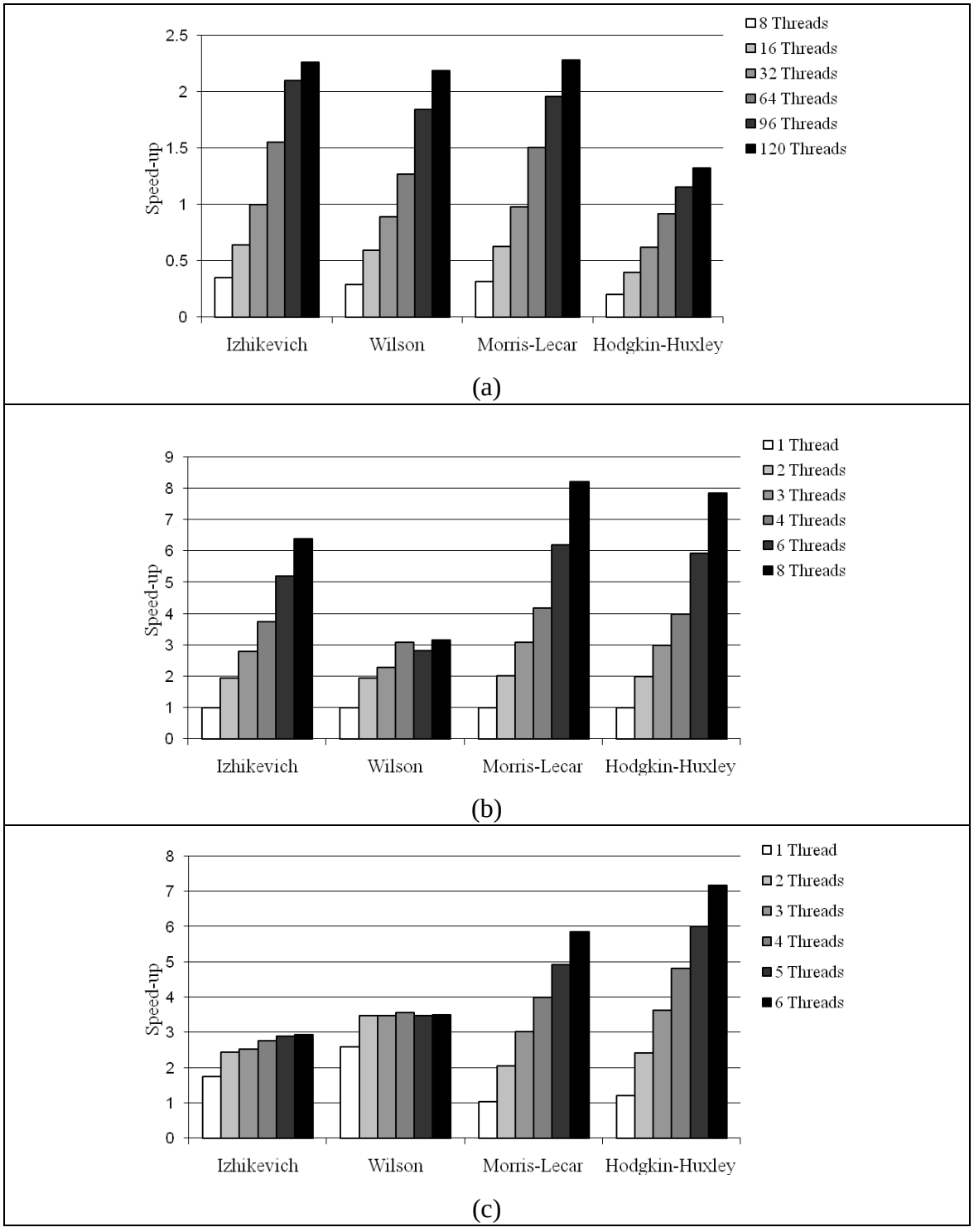

Figure 8.3. Variation in speed-up of the 1200×1200 network with the number of threads for the (a) Sun UltraSPARC T2+, (b) Intel Xeon, and (c) Cell platforms.

Figure 8.3 shows how the speed-up varies as the number of threads is increased on the different platforms. On Sun, Intel and Cell platforms, the speed-up increases with the number of threads for all the models. On the Cell platform, the performance gain with the increase in threads is less or almost negligible for the Izhikevich and Wilson models. This is primarily because the short runtimes of the code allow the extra DMAs from six cores to cause contentions. Figure 8.4 shows how the overall DMA time varies with the number of SPEs on the Cell processor based Sony PlayStation 3 for the 480 × 480 Wilson model. As expected, the computation time decreases with the increase in the number of SPEs, but the time for DMA does not decrease. Thus, these models are limited by DMA contentions.

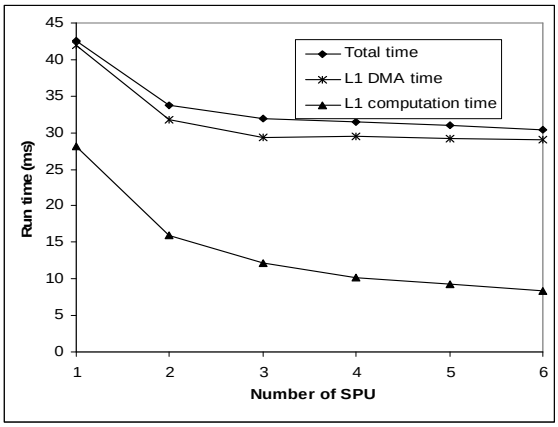

Figure 8.4. Variation in the runtime of the 480 × 480 Wilson model on the Cell processor based PlayStation 3.

The Runge-Kutta approach provides a higher level of accuracy in resolving the differential equations in the models, but requires significantly higher runtime. Table 3.1 compares the flops needed per neuron under two different approaches. Table 8.1 provides the compute-to-I/O ratio for each of the models when using this model to evaluate differential equations. As shown in Figure 8.5, the Runge-Kutta approach generally provided a higher speed-up over the Euler approach. This is primarily due to a higher compute-to-I/O ratio in this approach. The speed-up is over a PPE serial implementation.

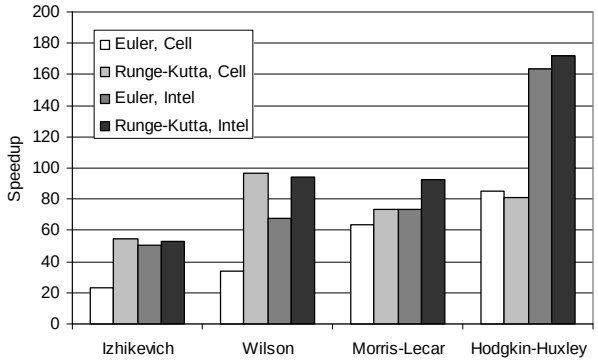

Figure 8.5. Speed-up of the four models over the Cell PPU when using the Euler and Runge-Kutta approaches.

CHAPTER NINE

LARGE SCALE IMPLEMENTATION

The AFRL Cluster

The AFRL cluster utilized in this study consists of 336 Sony Playstation 3s (PS3s). Each PS3 contains 256MB of RDRAM and a 40GB hard drive. As shown in Figure 9.1, the 336 PS3s were grouped into 14 sub-clusters, with each sub-cluster consisting of 24 PS3s, a dual quad-core Xeon head node, and a high speed Ethernet switch. The sub-clusters were connected through a central high speed Ethernet switch. The peak performance of the cluster is 51.5 TF. The cluster uses openMPI 2.4.1 for communication between the PS3s. Each PS3 was running on Fedora 7 equipped with IBM Cell SDK 3.1. A detailed description of the cluster is presented in [40].

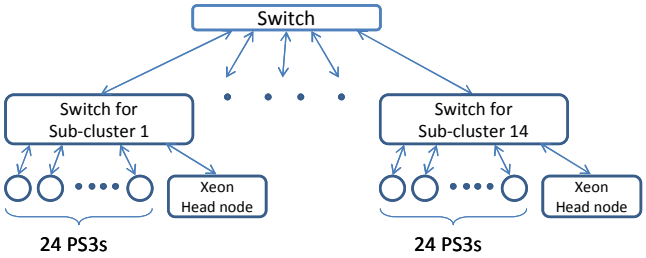

Figure 9.1. AFRL Cluster PS3 organization

Implementation

The testing phase of the network described in chapter five was implemented on the PS3 cluster. Since all four spiking network models were implemented using the same image recognition network structure, the parallelization approach and optimizations for all the models were the same. All the neurons at any particular level of the model run in parallel and are independent of each other. This allows the neurons of a given level to be split evenly across all the available SPEs in the full set of PS3s used. Additionally, since all the neurons utilize the same set of computations, vectorization was used to evaluate four neurons at a time on each SPE.

Eleven networks with varying input image sizes were developed in order to examine the performance and scalability of the four models on the AFRL cluster. Table 9.1 shows the input image size, number of level one and level two neurons and the corresponding number of synapses for the networks. The number of output neurons is equal to the number of training categories. In this study, all the networks are trained to recognize the same set of input images (scaled to appropriate sizes). The set of 48, 24x24 images which were generated initially [15] were scaled linearly depending on the required input image size.

Table 9.1. Networks used for Cluster Implementation of Spiking Neural Network Models

Size of Input Image	Level 1 neurons	Level 2 neurons	Synapses
1200 × 1200	1,440,000	48	69,120,000
2400 × 2400	5,760,000	48	276,480,000
3600 × 3600	12,960,000	48	622,080,000
4800 × 4800	23,040,000	48	1,105,920,000
8160 × 8160	66,585,600	48	3,196,108,800
8400 × 8400	70,560,000	48	3,386,880,000
12000 × 12000	144,000,000	48	6,912,000,000
14400 × 14400	207,360,000	48	9,953,280,000
16800 × 16800	282,240,000	48	13,547,520,000
18000 × 18000	324,000,000	48	15,552,000,000
20400 × 20400	416,160,000	48	19,975,680,000

Experimental Setup

On the AFRL Cluster utilized, approximately 300 out of 336 PS3s were available for use. The studies utilized only the PS3s on the cluster and did not run any code on the Xeon headnodes. In the runs performed, no impact was seen in using the PS3s from different sub-clusters on the overall runtime. This indicates that the MPI overhead for using PS3s in different sub-clusters and within one sub-cluster were similar.

Results

This section considers the scalability of the four spiking neural network models with variations in the number of PS3s. All the runs utilized the two layer configuration described in chapter five. Figure 9.2 shows the performance of the cluster with a fixed set of neurons assigned to each PS3. Thus varying the number of PS3s would proportionately change the overall number of neurons in the networks modeled. The results show that the neurons per second throughput for the cluster scaled up almost linearly with the number of PS3s. The four models have different flop counts per cycle. Additionally, as mentioned in chapter three, the four models do not simulate the same type of neuron; thus the number of simulation cycles needed for the four models to generate an inference is different. These contribute to the difference in the neurons per second throughput of the four models. The Izhikevich model required the least runtime and so had the highest neurons per second throughput. The Hodgkin-Huxley model was at the other end of the throughput scale.

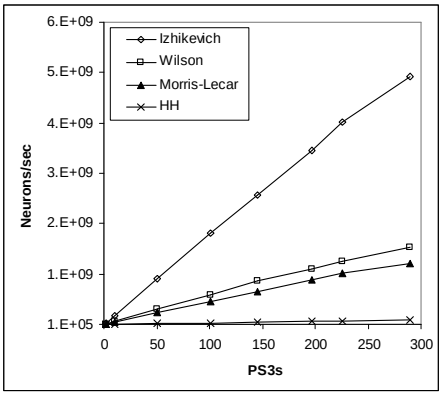

Figure 9.2. Runtime for varying number of PS3 and network size for same number of neurons/PS3 (1,440,000).

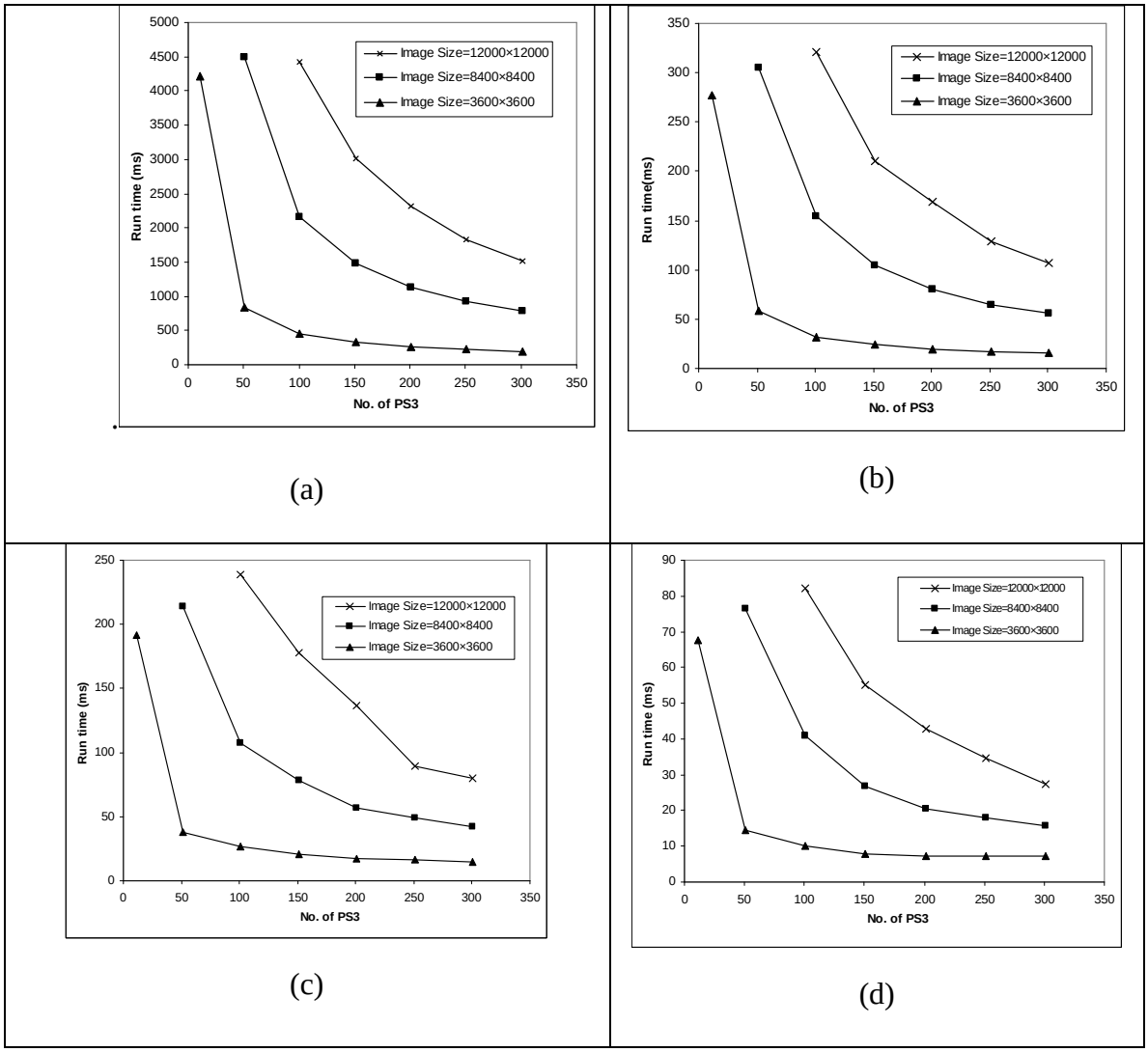

Figure 9.3. Runtime for the Spiking Neural Models for varying number of PS3s a) Hodgkin-Huxley b) Morris c) Wilson and d) Izhikevich

The scalability of networks with fixed numbers of neurons was examined with variations in the number of PS3s. Figure 9.3 shows the change in runtime for networks of three sizes for the four spiking neuron models. The three networks examined had the following number of level 1 neurons: 12,960,000 (3600×3600), 70,560,000 (8400×8400), and 144,000,000 (12000×12000). The number of level 2 neurons was always fixed at 48.

In all cases it is seen that the runtime decreases with the number of PS3s utilized. As expected, smaller networks reached a saturation point with fewer PS3s than the larger networks

The Cell processor contains eight SPEs, of which six are available on the PS3. The effect of changing the number of SPEs on the overall runtime was investigated with variations in the number of PS3s. Figure 9.4 shows the results of this study for the four spiking neuron models. As expected, in both the Hodgkin-Huxley and Morris-Lecar models (Figures 9.4 a and 9.4 b respectively), the runtime decreases proportionately with the number of SPEs utilized. This indicates that these two models were able take full advantage of all the SPEs available on the PS3s.

As shown in Figures 9.4 c and 9.4 d, the Wilson and Izhikevich models respectively reach saturation points at three SPEs – there is no significant improvement in performance by increasing the number of SPEs utilized. Chapter eight discusses this issue in detail. This is primarily due to limitations in the DMA bandwidth of the SPEs. A similar DMA saturation effect is seen in [38] [39]. Thus with these two models, it may be useful to run other (non-memory intensive) tasks on three of the SPEs on each Cell processor in the cluster.

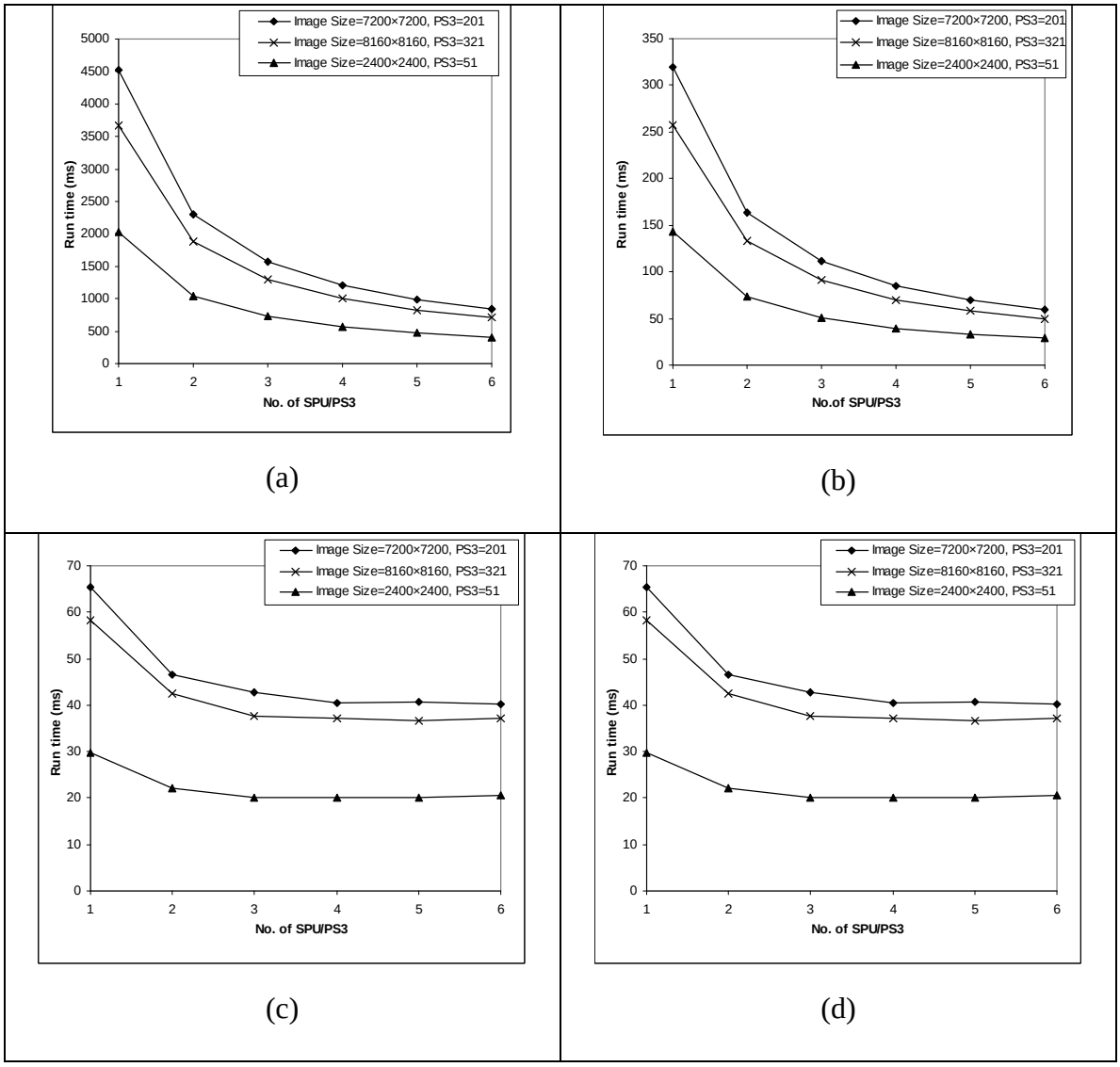

Figure 9.4. Runtimes of the Spiking Neural Models with varying number of SPEs on the PS3s a) Hodgkin-Huxley b) Morris-Lecar c) Wilson d) Izhikevich

Biological Relevance

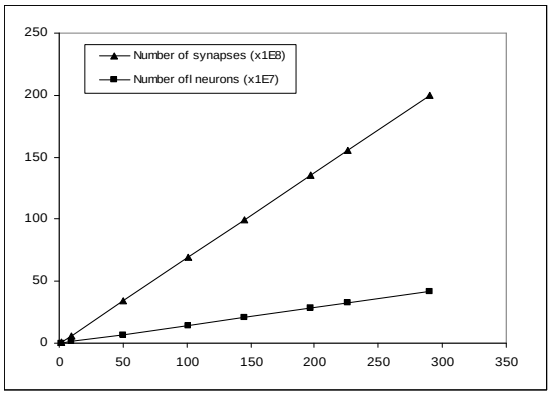

Figure 9.5. Maximum neurons and synapses processed for varying number of PS3s

The human cortex contains approximately 10^{11} neurons [23] and 1.5×10^{14} synapses whereas a mouse cortex has 1.6×10^{7} neurons and 1.6×10^{11} synapses [17]. Figure 9.5 shows the maximum number of neurons and synapses that were modeled with the spiking network models for varying numbers of PS3s. With 300 PS3s, up to 4.16×10^{8} neurons and 2×10^{10} synapses were modeled. Table 9.2 summarizes these results.

Table 9.2. Components of different systems

System	Neurons	Synapses
Human cortex	10^{11}	1.5×10^{14}
Mouse cortex	1.6×10^{7}	1.6×10^{11}
Spiking neuron models	4.16×10^{8}	2×10^{10}

Although the number of neurons (or equivalent neurons) modeled is close to biological scales, it is important to note that several biological properties were not captured in the models implemented. The models implemented considered only the "recognition phase", and thus did not model spike-timing-dependent plasticity (STDP).

Additionally, the two layer spiking network models were far removed from the highly interconnected neural structure seen in the cortex. However the results do indicate that large clusters of PS3s can provide a good platform for biological scale cortical models.

CHAPTER TEN

CONCLUSION

An image recognition model was utilized to analyze the performance acceleration of four spiking network models on modern multicore processors. The spiking network models were parallelized using both data and thread level parallelism.

The results of this work show that modern multicore processors can provide significant speed-ups for spiking neural network models. Results show that the architectures scale well and provide significant speed-ups for models with high compute-to-I/O ratio. The Sun UltraSPARC T2+ platform provided a lower performance improvement than both the Cell and Intel Xeon processors. This is likely to be due to the lack of SIMD operations on the Sun UltraSPARC T2+. Of the four models examined the Hodgkin Huxley and Morris-Lecar models provided the highest speed-ups for the larger models, while the Izhikevich and Wilson model provided the lowest speed-up. This was due to the low compute-to-I/O ratio in the latter.

From the scaling study on the cluster, it can be seen that the 336 PS3 cluster provides a highly economical, yet powerful, platform for neuromorphic simulations. The system is capable of producing up to 50 TFlops. The four models under study were scaled up on a cluster of 336 PS3s at the AFRL facility in Rome, NY. Results indicate that the models were fully scalable across the cluster. Additionally, two of the four models were scalable across the six SPEs available on each Cell processor in the cluster.

The largest spiking network model implemented contained 4.16×10^{8} neurons and 2×10^{10} synapses. Given that the human brain contains about 10^{11} neurons, this is a large

number of components that the cluster was capable of modeling. As a simplistic comparison, image recognition (for the largest image size tested) required about 227ms in the spiking network, and about 100ms in the human brain.

In a recent study [2], a 32,768 processor IBM BlueGene supercomputer was able to simulate a rat scale cortex (55×10^6 neurons and 4.42×10^{11} synapses) at near real time. This model did implement learning and was more biologically accurate than the models implemented in the study. However the cost of the BlueGene system is significantly higher (approximately 2-3 orders more) than the system we utilized. The AFRL cluster cost $337k, of which the PS3s cost about $133k. Since we were able to model a similar scale cortical system (although our model was much simpler) it indicates that a cluster of PS3s can be an economical platform for simulating large scale neuromorphic models.

It is important to note that the networks implemented are extremely simplistic. Possible future work in this area could be to examine implementations of more biologically realistic networks and include learning in the implementations. Additionally, the application of similar large cortical models to different domains could be examined. Also, the effect of various compilers on the performance of the models on each of the architectures could be studied. In particular, the effect of *icc* compiler on the performance of the Intel architectures could be investigated for each of the models.

REFERENCES

[1] T. Dean, "A Computational Model of the Cerebral Cortex," *Proceedings of the Twentieth National Conference on Artificial Intelligence* (AAAI-05), pp. 938-943, 2005.

[2] M. Djurfeldt, M. Lundqvist, C. Johansson, M. Rehn, O. Ekeberg, and A. Lansner, "Brain-scale simulation of the neocortex on the IBM Blue Gene/L supercomputer," *IBM Journal of Research and Development*, 52(1-2), 31–41, Jan.-Mar. 2008.

[3] E.Izhikevich, "Which Model to Use for Cortical Spiking Neurons?" *IEEE Transactions on Neural Networks*, 15(5), 1063-1070, 2004.

[4] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and application to conduction and excitation in nerve," *Journal of Physiology*, 117, 500–544, 1952.

[5] E. M. Izhikevich, "Simple Model of Spiking Neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569-1572, November, 2003.

[6] H. R. Wilson, "Simplified dynamics of human and mammalian neocortical neurons," *J. Theor. Biol.*, vol. 200, pp. 375–388, 1999.

[7] C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber," *Biophys. J.*, vol. 35, pp. 193–213, 1981.

[8] A. Gupta, L. Long, "Character Recognition using Spiking Neural Networks," International Joint Conference on Neural Networks, Aug. 2007.

[9] A. Delorme and S. Thorpe, "Face processing using one spike per neuron: resistance to image degradation," *Neural Networks*, vol. 14, pp. 795–804, 2001,

[10] Y. Dan and M. Poo, "Spike time dependent plasticity of neural circuits," *Neuron*, vol. 44, pp. 23–30, 2004.

[11] M. Gschwind, H. P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki, "Synergistic Processing in Cell's Multicore Architecture," *IEEE Micro*, 26(2), 10–24, Mar. 2006.

[12] Sun Microsystems, "UltraSPARC T2™ Supplement to the UltraSPARC Architecture 2007", http://opensparc-t2.sunsource.net/specs/UST2-UASuppl-current-draft-HP-EXT.pdf, 2007.

[13] J. Rickman, "Roadrunner supercomputer puts research at a new scale," Jun. 2008, http://www.lanl.gov/news/index.php/fuseaction/home.story/story_id/13602.

[14] T. M. Taha, P. Yalamanchili, M. A. Bhuiyan, R. Jalasutram, and S. Mohan, "Parallelizing Two Classes of neuromorphic Models on the Cell Multicore Architecture," to be presented at the International Joint Conference on Neural Network, Atlanta, Georgia, June 2009.

[15] M. A. Bhuiyan, R. Jalasutram, and T. M. Taha, "Character recognition with two spiking neural network models on multicore architectures," to be presented at the *IEEE Symposium on Computational Intelligence for Multimedia Signal and Vision Processing*, Nashville**,** Tennessee, March 2009.

[16] A. Delorme and S. J. Thorpe, "SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons," *Network-computation in neural systems*, 14(4), 613–627, Nov. 2003.

[17] C. Johansson and A. Lansner, "Towards Cortex Sized Artificial Neural Systems," *Neural Networks*, 20(1), 48–61, Jan. 2007.

[18] A. R. Baig, "Spatial-temporal artificial neurons applied to online cursive handwritten character recognition," in European Symposium on Artificial Neural Networks, pp. 561–566, April 2004.

[19] C. Panchev and S. Wermter "Temporal sequence detection with spiking neurons: towards recognizing robot language instructions," *Connect. Sci.*, 18(1): 1-22, 2006.

[20] D. V. Buonomano and M. M. Merzenich, "A neural network model of temporal code generation and position invariant pattern recognition," *Neural Computation*, vol. 11, pp. 103–116, 1999.

[21] T. Ichishita, R. Fujii, "Performance evaluation of a temporal sequence learning spiking neural network", *Proceedings of the 7th IEEE International Conference on Computer and Information Technology*, Oct. 2007.

[22] K-Team, Inc. Online Available: http://www.k-team.com/

[23] H. Markram, "The Blue Brain Project," *Nature Reviews Neuroscience*, 7, 153–160, 2006.

[24] W. Rall, "Branching dendritic trees and motoneuron membrane resistivity," *Experimental Neurology*, 1, 503–532, 1959.

[25] R. Ananthanarayanan and D. Modha, "Anatomy of a Cortical Simulator," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Supercomputing 2007), Reno, NV, November 2007.

[26] E. Izhikevich and G. Edelman, "Large-Scale Model of Mammalian Thalamocortical Systems," *Proceedings of the National Academy of Sciences*, 105(9), 3593–3598, Mar. 2008.

[27] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, K. Yelick, "Scientific Computing Kernels on the Cell Processor", *International Journal of Parallel Programming (IJPP)*, Vol. 35, No. 3, 2007.

[28] Y. Liu, H. Jones, S. Vaidya, M. Perrone, B. Tydlitát, A. K. Nanda, "Speech recognition systems on the Cell Broadband Engine processor", *IBM Journal of Research and Development*, Volume 51 Issue 5, 2007

[29] Scarpazza, D.P.; Villa, O.; Petrini, F. "Efficient Breadth-First Search on the Cell/BE Processor**",** *IEEE Transactions on Parallel and Distributed Systems*, Volume 19, Issue 10, Page(s):1381 – 1395, Oct. 2008.

[30] S. Williams et al., Optimization of sparse matrix–vector multiplication on emerging multicore platforms, *Parallel Comput. (2009)*, doi:10.1016/ j.parco.12.006, 2008

[31] M. M. Khan, D. R. Lester, Luis A. Plana, Alexander D. Rast, X. Jin, E. Painkras, Stephen B. Furber, "SpiNNaker: Mapping neural networks onto a massively-parallel chip multiprocessor," *IJCNN 2008*, pp. 2849-2856, June 2008

[32] E. M. Izhikevich., "Dynamical Systems in Neuroscience", MIT press, Cambridge, Massachusetts, 2007

[33] W. Gerstner, W. Kistler, "Spiking Neuron Models, Single neurons, Populations, Plasticity," Cambridge University Press, 2002

[34] A. Buttari, J. Dongarra, and J. Kurzak, "Limitations of the Playstation 3 for High Performance Cluster Computing," University of Tennessee Computer Science Technical Report, CS-07-594, May 2007.

[35] A. N. Burkitt, "A review of the integrate-and-fire neuron model: II. Inhomogeneous synaptic input and network properties," Biological Cybernetics, Vol. 95, Number 2, pp 97-112, August, 2006

[36] Y. Xia and V. Prasanna, “Parallel Exact Inference on the Cell Broadband engine processor,” *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, 2008.

[37] http://webuser.hs-furtwangen.de/~dersch/libsimdmath.pdf

[38] David Krolak, “Unleashing the Cell Broadband Engine Processor, The Element Interconnect Bus”, IBM white paper, http://www.ibm.com/developerworks/power/library/pa-fpfeib/

[39] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, K. Yelick, "Stencil Computation Optimization and Autotuning on State-of-the-Art Multicore Architectures", in Supercomputing (SC), 2008.

[40] Richard Linderman, “Early experiences with algorithm optimizations on clusters of playstation 3’s,” DoD HPCMP Users Group Conference, Jul. 2008.

[41] Samuel Williams, Jonas Carter, Leonid Oliker, John Shalf, Katherine Yelick, “Optimization of a lattice Boltzmann computation on state-of-the-art multicore platforms”, Journal of Parallel Distributed Computing, 2009.

[42] Q. Wu, P. Mukre, R. Linderman, T. Renz, D. Burns, M. Moore and Qinru Qiu, “Performance optimization for pattern recognition using Associative Neural Memory,” Proceedings of the 2008 IEEE International Conference on Multimedia & Expo, Jun. 2008.