Electronics and Computer Science Faculty
of Physical and Applied Sciences
University of Southampton

Alexandros-Pana Oikonomou
1st May 2013

Solving the Conjugate Gradient Method in a SpiNNaker Machine

Project Supervisor: Jeff Reeve
Second Examiner: Marcus Breede

A project report submitted for the award of
BSc Computer Science

Abstract

SpiNNaker is an asynchronous, event-driven parallel architecture designed to simulate the human brain. It has been designed to operate as a large scale neural network in real-time using a System-on-Chip multi core system. Its architecture is different from usual parallel computers, since cores use spikes to communicate with each other. That way usual pitfalls of parallel computing, such as race conditions and deadlocks are avoided. So far the most prominent uses of this architecture have been in neuroscience and robotics. The aim of this project is to put into use SpiNNaker's architecture and bring it closer to classic computer science problems, while solving them optimally. The given algorithm to solve in this project is the conjugate gradient method, an iterative way of solving systems linear equations. The algorithm successfully runs on the simulator and reduces the time complexity of the most expensive operations of the algorithm.

Contents

Acknowledgments and Statement of Originality

I would like to thank my supervisor Jeff Reeve for his help and support throughout this project.

1 Introduction

1.1 Aim

The aim of this project is to correctly solve the Conjugate Gradient Method on a SpiNNaker chip, thus using the massive parallelism that this machine offers to reduce the time complexity of the aforementioned algorithm. This is accomplished by reducing the time complexity of the most expensive operations of the algorithm which are matrix-vector multiplication and the scalar product of vectors. The complexity is reduced dramatically, due to the abundant number of cores provided from the architecture. This report explains this project and its constituents, any background research done to launch this project, along with designe and implementation choices.

1.2 Reasons and Justification

SpiNNaker is an architecture inspired by the biology of the human brain. Its optimal configuration has over a million cores[3], which have mainly been used to simulate the neurons of the human brain and in robotics. Examples of these would be using the SpiNNaker chip to simulate thousands of spiking neurons by using over four million synapses[6], or to simulate the neurons of a retina sensor[1].

However little work had been done into using the SpiNNaker architecture to solve classic computer science problems. That is why a problem such as the Conjugate Gradient Method had been proposed, which is a very common solution to optimization problems. In addition to that, the SpiNNaker architecture offers new parallel programming paradigms, that escape some common parallel programming pitfalls such as race conditions, deadlocks, mutual exclusion etc[7].

1.3 Overview

Say something here

2 Background

2.1 The neuron

To make the explanation of the SpiNNaker architecture easier, the design from which the SpiNNaker chip was inspired will be outlined. This is no other than the human neuron.

The human neuron is an electrically excitable cell that processes and transmits information through electrical and chemical signals. Its basic costitutes are the the soma, the dendrites and the axon. The soma is the body of the neuron. A dendrite receives signals from the soma of the neuron that it belongs to or other neurons. The dendrite extends for hundreds of micrometers and branches multiple times, thus forming a dendritic tree, which connects with other neurons axons. The axon is used to transmit signals to other neurons and it extends from the soma of the neuron to a dendrite. All human neurons have only one axon. Given the above analysis the dendrites could be described as the inputs of a neuron

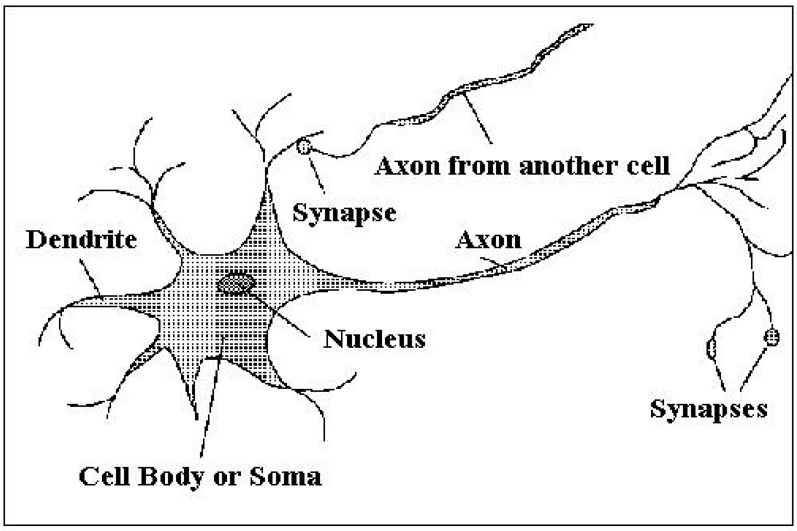

Figure 1: A neuron

One of the most important parts of the neuron structure is the synapse. The synapse is the contact between the axon of a neuron and the dendrite or the soma of another. It is where a information from one neuron is transmitted to the other. When a set of neurons are connected with each other through synapses, then they create a neural network.

Finally, the communication between neurons is accomplihed through spikes, which are either chemichal or electrical[2].

Given the terms in this section, the name of the SpiNNaker chip is deductable. It stands for Spi(king)N(eural)N(etwork) architecture.

2.2 The SpiNNaker architecture

2.3 The Conjugate Gradient Method

The conjugate gradient method is an algorithm for the numerical solution of systems of linear eqations of the type Ax=b, those whose matrix is symmetric and positive definite.

A *symmetric* matrix is a matrix which is equal to its transpose. If A is a symmetric matrix then A=A^T. The entries of the matrix are symmetric with respect to the main diagonal, so if an element of the matrix A is a, then a$_{ij}$=a$_{ji}$. A *positive definite* matrix M is a matrix which when multipled by any non-zero vector z and its transpose z^T, is always positive. In short the relationship that needs to be satisfied is zMz^T>0

It is an iterative method, which means it can be applied to sparse systems. A *sparse matrix* is a matrix which is populated primirily with zeros. Its opposite would be a *dense matrix*. It was developed by Magnus Hestenes and Eduard Stiefel and can be used to solve optimization problems[5].

2.3.1 The quadratic form

In order to explain why the Conjugate Gradient Method can solve problems whose matrix is positive-definite and symmetric an explanation of the quadratic form needs to be presented.

$$f(x) = \frac{1}{2}x^T Ax - b^T + c \tag{1}$$

The gradient $\nabla f(x)$ of the quadratic form is a vector field that points in the direction of greates increase of $f(x)$. If we were to take into account the ith component of ∇f.

$$\begin{aligned} f(x+\psi e_i) &= \frac{1}{2}(x+\psi e_i)^T A(x+\psi e_i) - b^T(x+\psi e_i) + c \\ &= \frac{1}{2}(x^T Ax + \psi e_i^T Ax + x^T A\psi e_i) - b^T(x+\psi e_i) + c \end{aligned} \tag{2}$$

Taking the definition of a derivative $f(x)$=$\frac{f(a+h)-f(a)}{h}$ we have.

$$\frac{f(x+he_i)-f(x)}{h} = \frac{\frac{1}{2}(\psi e_i^T Ax + x^T A\psi e_i) - \psi e_i^T b}{h} \tag{3}$$

Eqaution 3 is the same as the ith component of $\frac{1}{2}$(Ax+A^{T}x)-b So we have

$$\nabla f = \frac{1}{2}A^T x + \frac{1}{2}Ax - b \tag{4}$$

But if A is symmetric, then it is obvious that we have Ax=b at the minimum of f

If A is positive negative then f is concave up.

2.3.2 Conjugate Gradients

So as we said before A mus be symmetric and positive definite. Given a quadratic function as defined in Equation 1, then it turns out that
-∇f=b-Ax=r

Most optimization methods look towards a specifin direction everytime they are to move to a correct position and try to look for the best step given their gradient and direction

$$x_{k+1} = x_k + \alpha p_k \tag{5}$$

Given Equation 5, α also needs to be chosen so that $f(x_k + \alpha p_k)$ is minimized in the direction of p_k

$$0 = \frac{d}{d\alpha}f(x_{k+1}) = \nabla f(x_{k+1})^T \frac{d}{d\alpha}x_{k+1} = -r_{k+1}^T \frac{d}{d\alpha}(x_k + \alpha p_k) = -r_{k+1}^T p_k \tag{6}$$

As it is for the method of Steepest Descent, α should be set in a way that f is minimized in the direction ∇f=r

$$0 = \frac{d}{d\alpha}f(x_{k+1}) = \nabla f(x_{k+1})^T \frac{d}{d\alpha}x_{k+1} = -r_{k+1}^T \frac{d}{d\alpha}(x_k + \alpha r_k) = -r_{k+1}^T r_k \tag{7}$$

We need to pick α so that $\nabla f(x_{k+1})$ and r$_k$ are orthogonal. But since we have that $\nabla f(x_{k+1})$=-r$_{k+1}$ then

$$\begin{aligned} r_{k+1}^T r_k &= 0 \\ (b - A(x_k + \alpha r_k)^T) r_k &= 0 \\ (b - Ax_k)^T r_k - \alpha (Ar_k)^T r_k &= 0 \\ r_k^T r_k &= \alpha r_k^T A r_k \\ \alpha &= -\frac{r_k^T r_k}{r_k^T A r_k} \end{aligned} \tag{8}$$

2.3.3 The algorithm

Combining the optimal value of α and a good step direction for p_k the Conjugate Gradient Method is formed as follows

```
$r_0$=b-A$x_0$
$p_0$=$r_0$
f=0
**for** f to 1000000 **do**
    $alpha_f = \frac{r_f^T*r_f}{p_f^T*A*p_f}$
    $x_{f+1}$=$x_f$+$alpha_f$*$p_f$
    $r_{f+1}$=$r_f$-$alpha_f$*$p_f$*A
    **if** $r_{f+1}^T$*$r_{f+1}$ is small enough **then**
        break
    **end if**
    $beta_f = \frac{r_{f+1}^T*r_{f+1}}{r_f^T*r_f}$
    $r_{f+1}$=$r_{f+1}$+$beta_f$*$p_f$
    f=f+1
**end for**
```

Some notes that can be made about this algorithm are that the axis is defined by the eigenvectors of A and that the algorithm takes a conjugate step closest to r. Finally, the algorithm guarantees convergeance in at most n steps.[5][8][4]

References

[1] Sergio Davies, Cameron Patterson, Francesco Galluppi, Alexander D Rast, David Lester, and Steve B Furber. Interfacing real-time spiking i/o with the spinnaker neuromimetic architecture. In *Proceedings 17th International Conference, ICONIP 2010.*, pages 7–11. Australian Journal of Intelligent Information Processing Systems, Vol. 11, No. 1,, 2010.

[2] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[3] Javier Navaridas, Mikel Luján, Jose Miguel-Alonso, Luis A Plana, and Steve Furber. Understanding the interconnection network of spinnaker. In *Proceedings of the 23rd international conference on Supercomputing*, pages 286–295. ACM, 2009.

[4] L. Olson. Lecture 21 better iterative methods; google and markov chains, 2009.

[5] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[6] Thomas Sharp, Francesco Galluppi, Alexander Rast, and Steve Furber. Power-efficient simulation of detailed cortical microcircuits on spinnaker. *Journal of Neuroscience Methods*, 2012.

[7] Thomas Sharp, Luis A Plana, Francesco Galluppi, and Steve Furber. Event-driven simulation of arbitrary spiking neural networks on spinnaker. In *Neural Information Processing*, pages 424–430. Springer, 2011.

[8] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[3] [6] [1] [7] [2] [5] [8] [4]