ISSN 2229-5518

A Comparative Study on Improving the Latency Time of File Access Using Standard Backpropagation Neural Networks

Mr. S. Ashok Kumar, Dr. G. M. Nasira

Abstract—The data or file will be required to be moved from secondary memory to main memory for executing certain instructions. The time taken to transfer the data is referred as latency time. The In this paper a detailed study on various file access predictors and caching techniques is discussed. The various file operations such as file prediction, file creation, file deletion, file modification and file access are considered and a study is made in this paper for early prediction of files. Various file access predictors have been proposed in identifying the immediate successor of file or data block to be accessed. Few predictions have also proposed in identifying the files that can be accessed upto five access ahead. The standard backpropagation neural implementation in file access by improving the latency time of the file access is also proposed in this paper.

Index Terms— Artificial Neural Networks, Backpropogation Algorithm, Caching, File Access, Prefetching Files, Predictors

1 INTRODUCTION

This paper attempts to undertake the study of comparing the existing file access predictors with the proposed neural network implementation in accessing the file or data from secondary memory to main memory. While executing an instruction, the computer processor search for data or file that is stored in the main memory i.e. RAM (Random Access Memory) of the computer. If the required data or file is not stored in the main memory, then the data or file has to be fetched from the secondary memory i.e. hard disk or floppy disk or compact disks (CD). This process is called as loading a program.

The problem is that the hard disk is a mechanical system, and not an electronic one. This means that the data transfer between the hard disk and the main memory is much slower than the data transfer between the processor and the main memory. [1]

The processor communicates with the main memory typically at a transfer rate of 800 MB/s (100 MHz bus), while the hard disks transfer data at rates such as 33 MB/s, 66 MB/s and 100 MB/s, depending on their technology. The latency time is thus very high for data transfer between the main memory and secondary memory. The latency time is the time between initiating a request and receiving the answer. [2]

2 FILE/DATA ACCESS

The idea of placing the likely data to be accessed together near each other has long been accepted as a wise goal in memory storage systems. Placing item nearer will reduce the latency time. To overcome latency problems two main techniques are used they are pre-fetching and predictive caching.

Caching keeps the mostly required data in the memory, likewise pre-fetching try to bring the data in memory before they are required [1].

2.1 Caching

The memory access time can be reduced if the most frequent accessed instruction and data are kept in a small memory. Such fast small memory is called as cache memory. The caching methodology operates as of the data which was accessed recently is likely to be accessed very soon again. Thus the cache memory contains the next data that might be requested.

The drawback of caching is to find out which data block has to be removed when cache memory is full. And when a new block of data is to be transferred to cache. When memory is full FIFO (First in First Out) and LRU (Least Recently Used) block replacement policies are used.

2.2 Prefetching

File prefetching is an effective technique for improving file access performance. Prefetching will fetch the required data by retrieving the data in advance. A predictive prefetching requires the prediction of sequences of file accesses far enough in advance to avoid the predictions being untimely.

The literature survey shows that a number of file access predictors are being proposed namely [1], [2]

1) First Successor (FS)
2) Last Successor (LS)
3) Stable Successor (SS)
4) Recent Popularity (RP)
5) First Stable Successor (FSS)
6) Predecessor Position (PP)
7) Prepredecessor Position (PPP)

First Stable Successor (FSS) predictor requires m successive instances of file Y immediately following in-

stances of file X before predicting that file Y is the successor of file X. Otherwise it makes no prediction. When m = 1, the FSS predictor becomes identical to the First Successor protocol and predicts that that file Y is the successor of file X once it has encountered a single access to file Y immediately following an access to file X.

Assumptions

G is file being currently accessed
F its direct predecessor
FirstStableSuccessor (*F*) is last prediction made for The successor of *F*
Last Successor (*F*) is last observed successor of *F*
Count (*F*) is a counter
M is minimum number of consecutive identical successors to declare a First Stable Successor

Algorithm

if *FirstStableSuccessor*(*F*) is undefined then if
    LastSuccessor(*F*) = *G* then
        Counter(*F*) ← *Counter*(*F*) + 1
    else
        Counter(*F*) ← 1
    end if
    if *Counter*(*F*) = *m* then
        FirstStableSuccessor(*F*) ←G
    end if
end if

Fig. 1. First Stable Successor Algorithm, The figure shows the first stable successor algorithm used for predicting the file for early file access.

3 Artificial Neural Networks

Artificial Neural Networks (ANNs) is an abstract simulation of a real nervous system that contains a collection of neuron units communicating with each other via axon connections. Neural networks take a different approach in solving a problem than that of conventional methods.

Conventional methods use algorithmic approach, where the method follows a set of instructions in order to solve the problem. Unless we know the specific steps in prior that the method needs to follow, only then the computer can solve the problem. That restricts the problem solving capability of conventional methods to solving problems. But a method would be so much more useful if they could do things that we don't exactly tell them rather train them how to do [3].

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements called neurons which works in parallel to solve a specific problem. There are two phases in neural information processing.

They are the learning phase and the retrieving phase. In the training phase, a training data set is used to determine the weight parameters that define the neural model. This trained neural model will be used later in the retrieving phase to process real test patterns and yield classification results. A detailed literature survey was made in the area of neural network which has motivated us to apply this technique to solve this problem [4].

A standard feed forward artificial neural network is shown in the Fig. 1. It has three layers. The inputs biases and weights are labeled. Each unit outputs a function of its inputs. This function is called the activation function. Selecting values for the biases and weights is done to fit the entire ANN function to data.

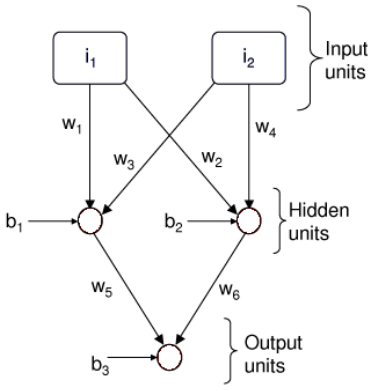

Fig. 2. Example Artificial Neural Network, showing 3 layers viz input units, hidden units and output units with artificial neurons. The figure shows the bias b and weights w applied to the network

The output of the artificial network is given by the following equation

$$\text{Output} = (b3 + w5 \cdot f(b1 + w1 \cdot i1 + w3 \cdot i2) + w6 \cdot f(b2 + w2 \cdot i1 + w4 \cdot i2))$$

Artificial Neural Network (ANN) is a small group of classic networks which are widely used and on which many others are based. These are: Back Propagation, Hopfield Networks and Competitive Networks. The Standard back-propagation (Fletcher and Powell, 1963) is the most popular method used to select values for ANN free parameters. It is done iteratively, calculating the error gradients of the data in respect to the free parameters and then updates them appropriately. The error gradients are calculated starting from the error on the outputs and works backwards. Each iteration of all the training data is called an epoch. It is a steepest decent search for a minimum value.

A Back Propagation network learns by example. You give the algorithm examples of what you want the network to do and it changes the network's weights so that, when training is finished, it will give you the required output for a particular input. The backporpogation algorithm works as follows:

1. First apply the inputs to the network and work out the output – the initial weights were random numbers.
2. Next work out the error for neuron B. The error is

$$\text{Error}_B = \text{Output}_B (1-\text{Output}_B)(\text{Target}_B - \text{Output}_B)$$

ISSN 2229-5518

3. Change the weight. Let W^+AB be the new (trained) weight and W_{AB} be the initial weight.

$$W^+AB = WAB + (Error_B \times Output_A)$$

4. Calculate the Errors for the hidden layer neurons. Unlike the output layer we can't calculate these directly (because we don't have a Target), so we back propagate them from the output layer. This is done by taking the Errors from the output neurons and running them back through the weights to get the hidden layer errors.

$$Error_A = Output_A (1 - Output_A)(Error_B W_{AB} + Error_C W_{AC})$$

5. Having obtained the Error for the hidden layer neurons now proceed as in stage 3 to change the hidden layer weights. By repeating this method we can train a network of any number of layers.

6.1 Artificial Neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire or not for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

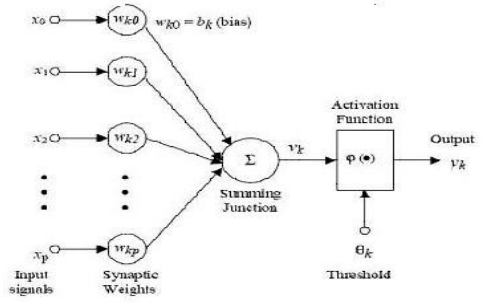

Fig. 3. Simple neuron, the figure shows a simple neuron with activation function and input weights. The output is decided based on the activation function used by the network

4 Related Work

Kroeger and Long [5] suggested that the file access prediction is not only used to prefetch the necessary files, but it is also used to find the related files which can be grouped under a single set.

Griffioen and Appleton [6] presented an analysis of the potential effects of predictive prefetching based on system performance. They proposed that there was relationship between successive pairs of file accesses.

Lei et. al. [7] proposed a new method which keeps track of different file access patterns observed under each application in a specific system. The method scheme stores file patterns as access tree.

Tait and Duchamp [8] investigated a client-side cache management technique used for detecting file access patterns and for exploiting them to prefetch files from servers. Darrell D.E.Long and et. al. has proposed to distinguish between the short term and long term file activity on a Unix system. This information helps to predict the files which are never used.

Purvi Shah, Jehan Paris, A. Amer and Darell D.E Long [9] have suggested in identifying a stable file access pattern using the first stable successor predictor. It is required since prefetching is built on long-lived clusters of related files. Natarjan Ravichandran and Jehan Paris [10] have proposed a perceptron based file predictor that predicts the files that need to be accessed upto give file accesses ahead.

5 Proposed Research

The file activity can be summarized and is presented in the Table 1

TABLE 1
VARIOUS FILE ACTIVITIES

File Activity	Particulars of the Activity
Accesses, Creations	It is the measure of total number of files and bytes.
Deletions	It is similar to access, with deletions the i-nodes are reused and tracked separately
Modifications	It is similar to access with categories for files that are modified and increased in size, decreased in size, or remained the same.
Modification Differences (Deltas)	It is similar to access with categories for files that are modified and increased in size or decreased in size. Tracks files by the amount of change. Produces a two dimensional histogram of file size versus the amount of the

Darell D.E. Long [11] in his findings suggests that the files in a computer system are relatively small and most of the file is found to be less than 8 kb size. They also suggest that only 25% of the files are larger than 8 kb.

The various file activity include, files modified, files deleted, files created and files accessed. A file modification is explained with an instance as follows. When a programmer changes a file with a text editor and saves the changes the following steps occur

a) User will command to open the required file
b) Operating System will find the i-node
c) Operating System gets the file's disk block numbers from the i-node
d) Operating System reads the disk blocks and loads the file into main memory
e) User makes the changes and saves the file
f) Operating System writes the modified file to a new set of disk blocks with a new i-node value
g) Operating System frees the old disk blocks and the old i-node is deleted

h) Operating System updates the file name to point to the new i-node.

The modification of files relates to increase in the size of file or decrease in size of the file. By statistics it is found that mostly the file has growth rather than depreciation. The file growth rate has direct implications for operating system design. When a file needs to increase in size, some operating systems allocate additional space based on the files original size, the larger the file the more space is allocated.

In order to recover the unused disk space, the operating system must keep track of the files that has received the additional spaces and must be taken back when required. Modification of file size involves the following characteristics

a) The numeric index files are modified
b) At most, half of all numeric index modifications result in a file size increase
c) Most numeric index file size increases from modifications are less than 32 KB.

The theory of locality states that the deletion rate for used files should initially be low and increase as the files get older. This will surely increase the deletion rate for used files.

Various file characteristics are considered for the analysis of file prefetching and predictive caching. All the file characteristics can be defined and can be trained by the standard backpropagation neural network algorithm so that the algorithm will provide a mean in prefetching the required file well in advance. This activity will reduce the efforts of operating systems to deal with input-output operations of a file transfer in more efficient manner.

6 CONCLUSION

A detailed study on various files access predictors and caching techniques were discussed. A detailed comparative study on various file operations such as file prediction, file creation, file deletion, file modification and file access was also carried out. A standard backpropagation neural network implementation for file access will improve the latency time of file access has been proposed.

REFERENCES

[1] A. Amer, "Predictive data grouping using successor prediction," Ph.D Dissertation, Department of Computer Science, University of California, Santa Cruz, CA, 2002

[2] Prasantha Kumar Patra, et. al, "File Access Prediction Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol 21, No. 6, June 2010

[3] Kartalopoulous, Stamatious V, "Understanding neural networks and fuzzy logic", Prentice hall 2003

[4] Foo Yoon-Pin Simon Tajkhorshid, "Hopfield Linear Programming neural networks for job-shop scheduling," IEEE International conference on Neural Networks, 1988, Vol 2, pp. 341-348

[5] T. M. Kroegaer and D.D.E Long, "Design and implementation of a predictive file prefetching algorithm," Proceeding of *USENIX*, Tech. Conference, Boston, Jun 2001

[6] Griffioen and R. Appleton, "Redoing file system latency using a predictive approach," Proceedings of *USENIX*, Tech. Conference, Boston, June 1994

[7] H. Lei and et. al, "An analytical approach to file prefetching", Proceedings of *USENIX,* Tech, January 1997

[8] Tait and Duchamp, "Detections and exploitations of file working sets", Proceedings of Conference on Distributed Computing systems, May 1991

[9] Purvi Shah et.al , "Identifying Stable File Access Patterns," 21st Proceedings of IEEE Mass storage syst. Apr 2004

[10] Natrajan Ravishandran and Jehan Paris, " Making early predictions of File Accesses," 4th Proceedings of 4th International conference on Information, Telecommunication and Technology, December 2005

[11] Darell D.E. Long, Ethan L. Miller, Tim Gibson, "Long-term File Activity and Inter-Reference Patterns," CMG, 1998

[12] Keith Smith and Margo Seltzer, "File layout and file system performance", Technical report, Harvard University, 1994