

# SPECIFIC MATHEMATICAL MODEL OF MEMORY SHARING PREDICTOR

H. A. Ali,  
Computers and systems Dept.,  
Faculty Eng.  
Mansoura Univ. Mansoura, Egypt  
h\_arafat\_ali@mans.edu.eg

Mostafa S. Saleh  
Computer Science Dept.  
King Abd-Ualziz Univ.  
KSA

M. A. Abas  
Computers and systems Dept.,  
Faculty Eng.  
Mansoura Univ. Mansoura, Egypt  
mohaym\_2000@yahoo.com

## Abstract

A lot of researches had been concerned with the memory sharing predictor (MSP), they defined the main purpose of such architecture explained how the mechanism of message prediction occurs through speculative coherence protocol to avoid the remote latency in memory sharing systems. This protocol is very vital specially when the messages and the acknowledgements are sent and received through slow processors. Most of the researches which had been done on MSP concentrated on avoiding latency in memory sharing system by investigating mechanisms for message prediction through speculative coherence. The lack of these researches is in ignoring a lot of parameters which may have a great impact on the performance analysis. The main objective of this paper is to introduce a MSP mathematical model that takes into account most of the important parameters that affect the prediction. Extracted results based on simulation are compared with the most recent model. Analyzing of the comparison shows that how the prediction accuracy affects the performance of memory sharing systems. Finally this paper also suggests using a new developed and a low constant synchronization overhead operating system called K42/Tornado to increase the performance of memory sharing systems.

**Keywords:** Active memory, Memory sharing predictor, Modeling, Speculative coherence

## 1. INTRODUCTION

Memory sharing predictor is one of the most considered mechanism in active memory technology, it saves the total latency time by eliminating the overheads time at processors and directory in shared memory systems using speculation method .MSP consists of history table which contains the message memory request for each block and pattern tables which contains

the last occurred message sequence for this memory block. Each message memory block has its own pattern table ,MSP neglects the acknowledgment messages for each memory block , the messages which it predicts are read , write and upgrade [12,16].

An MSP is based on the key observation that to eliminate the coherence overhead on a remote access latency it is only necessary to predict the memory request messages (i.e., a read, write, or an upgrade). A general message predictor unnecessarily predicts the coherence acknowledgement messages (i.e., an invalidation response or a write-back) as well, even though these messages are in direct response to a coherence action and are always expected to arrive. In Figure 1 (right), the write-back message by P3 is in direct response to the invalidation message by P2. The write-back is only a response to the coherence activity invoked by the read request and is itself part of the coherence overhead [10]. Because it predicts all coherence messages, a general message predictor has several key shortcomings. First, since the protocol overlaps the invalidation messages for a block, the acknowledgments may arrive in any arbitrary order. Predicting acknowledgments may unnecessarily and severely perturb prediction of the (more fundamental) request messages if acknowledgments often arrive out of order. Second, predicting the acknowledgments unnecessarily increases the number of pattern table entries. Third, predicting the acknowledgments increases the required number of bits to encode message types in both the history and pattern tables [8]. MSP addresses the above shortcoming in a general message predictor by only predicting the request messages. Most of recent researches did not deal with the functionality of memory sharing predictor in terms of mathematics, and neglecting number of parameters that affect the accuracy of MSP. This paper fully describes the effect of the prediction on the performance and how the prediction accuracy affects the performance of memory sharing systems.

## **2. MEMORY SHARING PREDICTOR (MSP)**

Figure 1 illustrates an example of producer/consumer sharing for memory sharing predictor that eliminates half of the pattern table entries. The MSP would also eliminate all the sequences resulting from the potential re-orderings of the acknowledgments (not shown in the figure). The MSP requires two bits to encode three request message types (i.e., read, write, and

upgrade) as compared to a message predictor requiring three bits to encode three request types and two acknowledgement types (i.e., responses to read-only invalidations and write backs).

In contrast, MSP only requires two bits for the type and log (n) bits to encode a processor id for every read Request [15]. Memory sharing predictor is one project toward an effective active memory system in multiprocessors environment using a shared memory, the prediction process increases the efficiency of the system, saving number of steps in sending and receiving message requests [10,14].

Most of recent works suggested a mathematical model of MSP but neglecting many parameters that can affect the performance such as the operations that occurs in the history table, pattern table and the wire time [12,20].

It is much important to search for a suitable mathematical model to represent accurately these parameters and studying the effect of considering these parameters on calculating the overall performance of the system. considering these extra parameters is expected to help in studying the performance of the system more accurately.

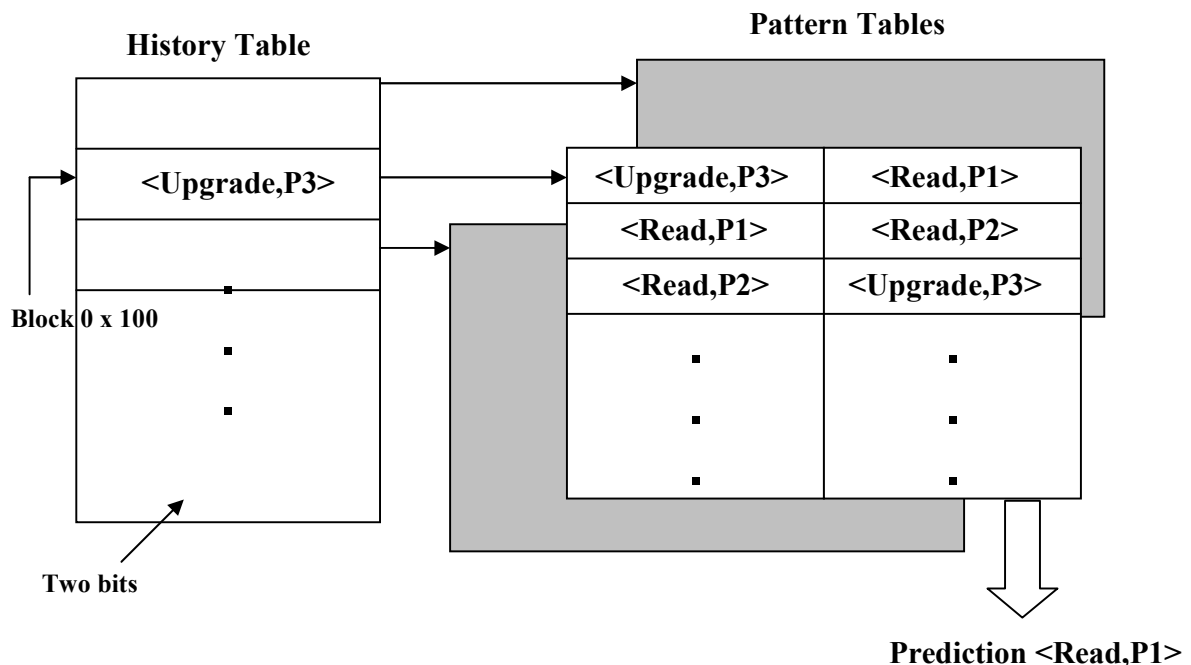
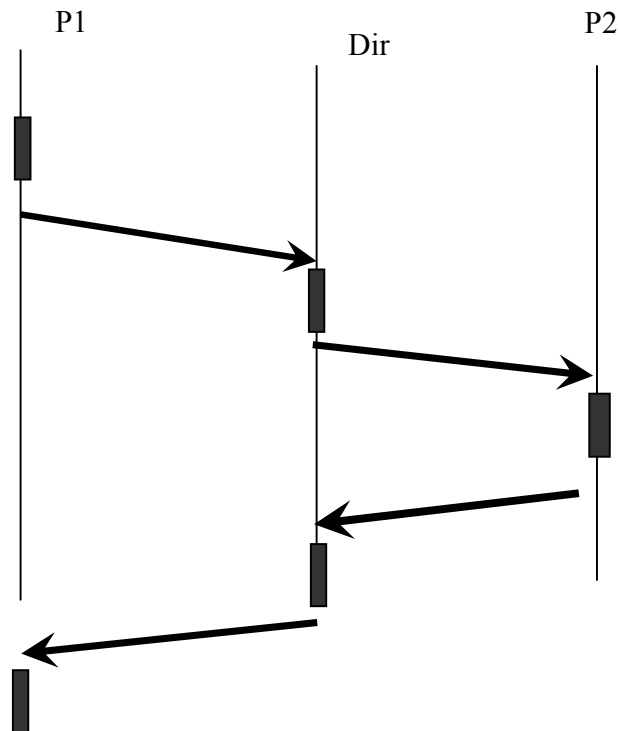


Figure 1 A Memory Sharing Predictor (MSP) [10]

### 3. MEMORY SHARING MODE WITHOUT PREDICTION

Figure 2 shows a system with two processors deals with a directory through request messages under the absence of memory sharing predictor, when P1 sends a request message to Directory there is an overhead time on P1 and a wire time during sending the message from P1 to the directory, then there is an overhead on the directory and a wire time during sending the message from the directory to P2 , then there is an overhead time on directory P2 and a wire time during sending the response message from P2 to the directory , then there is an overhead on the directory and a wire time during sending the response message from directory to P1 and then there is an overhead on P1 during the receiving of the request , the following parameters shows these operations in details.[13-14,18]



**Figure 2 shows coherence protocol action digraph without prediction**

#### 3.1 Performance Model

The previous performance model includes the following parameters:  $c$  is the application's communication ratio on the critical path,  $f$  is the fraction of speculatively-executed memory requests over all the received requests,  $p$  is the request prediction accuracy,  $l$  access and  $r$  access represent the local and remote memory latencies respectively,  $rtl$  is the ratio of remote to

local access latencies,  $n$  is the mis-speculation penalty factor, and  $N$  is the number of remote requests on the critical path where:

Comm-speedup = (communication time without speculation)/( (communication time with speculation)

$$= N r_{\text{access}} / [(1-f) N r_{\text{access}} + f N (p l_{\text{access}} + (1-p) n r_{\text{access}})]$$

$$T_{\text{lat}} \text{ (latency time without prediction)} = N r_{\text{access}} \quad (1)$$

$$= 1 / [(1-f) + f(p/rtl + n(1-p))]$$

Speedup = (total execution time without speculation / total execution time with speculation)

$$= 1 / [(1-c) + c / \text{com-speedup}]$$

$$= 1 / [(1-c) + c [(1-f) + f(p/rtl + n(1-p))]]$$

#### 4. THE PROPOSED MATHEMATICAL MODEL

A performance model based on the proposed mathematical model of memory sharing predictor is investigated. In this model the parameters that affect the remote latency time is considered [4,5,7]. Before starting our derivation three theories will be discussed in brief. Each of them will help in analyzing and understanding the whole idea of memory mapping and presentation[8]. **i) Ordered pairs theory:** is used here to represent the contents of the pattern table. The contents of pattern tables could be ordered triples or ordered quadruples .....etc, In general formula, the contents of pattern tables are described as: (< message-type, processor-receiver >, < message-type, processor-sender>, < message-type, Directory>)

For a given pattern in figure1: <Upgrade,P3>, <Read,P1> , <Read,P2>

The Memory Sharing Predictor mathematical model use the ordered triples definition to express the above pattern as follow (<Upgrade,P3> , <Read,P1> , <Read,P2>) = ((<Upgrade,P3> , <Read,P1>) , <Read,P2>) the ordered pair (<Upgrade,P3> , <Read,P1>) indicates that it is expected to see the contents of the pattern table <Read,P1> after <Upgrade,P3>

**ii) Relation theory:** is used to represent the relation between history table and pattern table, which is depicted in fig 1, this relation should be expressed as a Reflexive Relation as: for all  $a \in A$   $a R a$  , where  $R$  is reflexive if every node of  $G$  has a loop attached to it[8]. The Memory Sharing Predictor mathematical model uses the definition of Reflexive Relation to express that

the last message received in the history table refers to the same message in the pattern table, can be represented as a reflexive relation in a general formula:

For  $\langle \text{message-type, processor} \mid \text{directory} \rangle \in$  sets of received messages  $\langle \text{message-type, processor} \mid \text{directory} \rangle R \langle \text{message-type, processor} \mid \text{directory} \rangle$ .

The total time required to execute the reflexive relation in MSP ( $T_{\text{reflexive}}$ ) is represented as:  
 $T_{\text{reflexive}} = T(\langle \text{message-type, processor} \mid \text{directory} \rangle R \langle \text{message-type, processor} \mid \text{directory} \rangle)$

$T_{\text{reflexive}}$  could be represented in a digraph as a loop,  $T_{\text{reflexive}} = 2 * \pi r$  where  $r$  is the radius of the loop which represents the reflexive relation, the diameter of the loop represents the prediction accuracy, increasing the diameter refers to more delay in prediction process and less accuracy. For a given content of history table and pattern table in figure1  $\langle \text{Upgrade, P3} \rangle \in$  sets of received messages  $\langle \text{Upgrade, P3} \rangle R \langle \text{Upgrade, P3} \rangle$  is a reflexive relation and  $T_{\text{reflexive}} = T(\langle \text{Upgrade, P3} \rangle R \langle \text{Upgrade, P3} \rangle)$

Finally, **iii) The contents of pattern table:** could be expressed as an injective function [8] in a general formula

$F(\langle \text{message-type, processor} \mid \text{directory} \rangle) = \langle \text{message-type, processor} \mid \text{directory} \rangle$

The total time required to execute the injective function in MSP ( $T_{\text{func}}$ ) is represented as:

$T_{\text{func}} = T(F(\langle \text{message-type, processor} \mid \text{directory} \rangle) = \langle \text{message-type, processor} \mid \text{directory} \rangle)$

$T_{\text{func}}$  could be represented in a digraph as an arrow,  $T_{\text{func}} \approx 2M$  where  $M$  represents the remote receive overhead because wire time is very little compared to send and receive overheads.

The contents of pattern table as shown in figure1 includes  $\langle \text{Upgrade, P3} \rangle$ ,  $\langle \text{Read, P1} \rangle$ ,  $\langle \text{Read, P2} \rangle$  the injective function should be introduced as injective function  $F(\langle \text{Upgrade, P3} \rangle) = \langle \text{Read, P1} \rangle$ ,  $F(\langle \text{Read, P1} \rangle) = \langle \text{Read, P2} \rangle$ ,  $F(\langle \text{Read, P2} \rangle) = \langle \text{Upgrade, P3} \rangle$

A global overview of how MSP operates using figure 1 could be summarized as follow

1- use the ordered triples definition to express  $\langle \text{Upgrade, P3} \rangle$ ,  $\langle \text{Read, P1} \rangle$ ,  $\langle \text{Read, P2} \rangle$  as  $(\langle \text{Upgrade, P3} \rangle, \langle \text{Read, P1} \rangle, \langle \text{Read, P2} \rangle) = ((\langle \text{Upgrade, P3} \rangle, \langle \text{Read, P1} \rangle), \langle \text{Read, P2} \rangle)$

2- the reflexive relation for  $\langle \text{Upgrade, P3} \rangle \in$  sets of received messages  $\langle \text{Upgrade, P3} \rangle R \langle \text{Upgrade, P3} \rangle$ .

3-the injective function  $F(\langle \text{Upgrade}, P3 \rangle) = \langle \text{Read}, P1 \rangle$  ,  $F(\langle \text{Read}, P1 \rangle) = \langle \text{Read}, P2 \rangle, F(\langle \text{Read}, P2 \rangle) = \langle \text{Upgrade}, P3 \rangle$ .

This model is used to compare the performance of the system, contains a sharing memory and multiple processors each of them deals with the memory with request messages, with and without memory sharing predictor and indicates how the memory sharing predictor can affect on the performance of this system.[15-21]

#### 4.1 Memory Sharing Mode with prediction

Figure 3 shows coherence protocol action digraph with prediction under the existence of memory sharing predictor which is depicted in Figure 1.

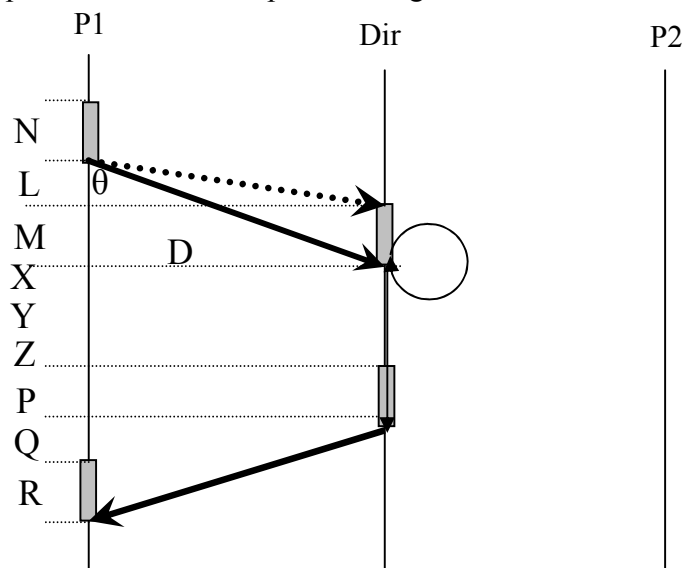


Figure 3 coherence protocol action digraph with prediction

Where:

**N** : local send overhead in P1.

**L** : wire time for the request of P1.

**M** : remote receive overhead of Directory from P1.

**X** : wire time for the request of Directory.

**Y** : remote receive overhead of P2.

**Z** : wire time for the response from P2 to Directory.

**P** : remote receive overhead of Directory from P2.

**Q** : wire time for the response from Directory to P1.

**R** : local receive overhead in P1.

**Φ** : parameter refers to the difference of wire time of the system and it varies from system to system which applies the cache coherence protocol without prediction.

$\theta$  : parameter refers to the difference of wire time of the system and it varies from system to system which applies the cache coherence protocol with prediction.  
 $D$  : the ideal wire time.

Figure 3 shows that due to prediction the overhead time in coherence protocol action is neglected this cause the existence of two angles  $\theta$  and  $\Phi$ , sending a request message from P1 to the Directory does not require an overhead time on P1 and also there is not an overhead time on the directory in the moment of receiving the request, MSP exists at the side of the directory, so there is a time required to connect the history table with the pattern table representing the reflexive relation, this time is called  $T_{reflexive}$ , from the property of reflexive relation this time is represented in the digraph as a loop [13], then there is a time representing the prediction operation called  $T_{func}$  after the prediction operation completed the response message is sent to the P1 with a wire time but without any overhead on P1, Under the assumption that:

- 1- local and remote operating system overhead and wire time are the same, then  $L = X = Z = Q$
- 2- In these systems wire time is very little compared to send and receive overheads so  $L + M \approx M$  [9].

3-the dot line depicted in figure 3 show the path of the message request of P1 from P2.

The total latency time with prediction ( $T_{latPred}$ ) [2] is calculated as follow:

$$\begin{aligned} T_{latPred} &= M + ((L+M)^2 + D^2)^{1/2} + T_{reflexive} + T_{func} + ((Q+M)^2 + D^2)^{1/2} \\ &= M + T_{reflexive} + T_{func} + 2(M^2 + D^2)^{1/2} \end{aligned} \quad (2)$$

from figure 4  $((L+M) / \cos \theta) = (D / \sin \theta)$  so,  $(M / \cos \theta) = (D / \sin \theta)$

$$D = M \tan \theta \quad (3)$$

From 2&3

$$\begin{aligned} T_{latPred} &= M + T_{reflexive} + T_{func} + 2(M^2 + M^2 \tan^2 \theta)^{1/2} \\ &= M + T_{reflexive} + T_{func} + 2M(1 + \tan^2 \theta)^{1/2} \\ &= M + T_{reflexive} + T_{func} + (2M / \cos \theta) \end{aligned} \quad (4)$$

In digraph  $T_{reflexive} = 2 * \pi r$  where  $r$  is the radius of the circle which represents the reflexive relation,  $T_{func} \approx 2M$  because wire time is very little compared to send and receive overheads.

from equation 7 :  $T_{lat} = 5M + 4(L^2 + D^2)^{1/2}$  So,  $\tan \Phi = D/L = M \tan \theta / L$

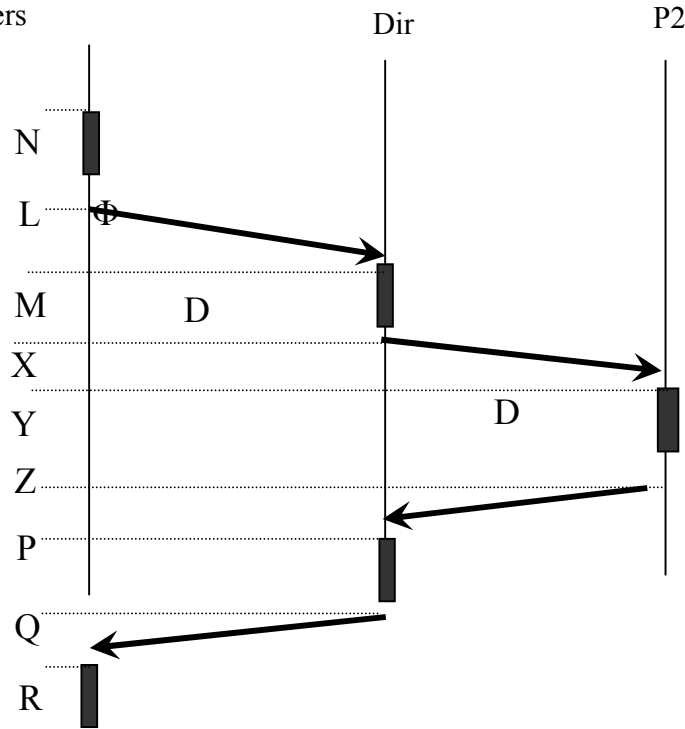
$$T_{lat} = 5M + 4((D / \tan \Phi)^2 + D^2)^{1/2} = 5M + 4M \tan \theta ((\cot \Phi)^2 + 1)^{1/2} = 5M + 4M \tan \theta / \sin \Phi$$

$$=M(5 + 4 \tan \theta / \sin \Phi) \quad (5)$$

$$T_{lat} / T_{latPred} = \text{Speedup} = [(5 + 4 \tan \theta / \sin \Phi)] / [1 + (2 * \pi r / M) + 2 + 2 / \cos \theta] \quad (6)$$

#### 4.2 Memory Sharing Mode without prediction

The following figure shows the coherence protocol action digraph without prediction using additional parameters



**Figure 4 coherence protocol action digraph without prediction using additional parameters**

So the total latency time  $T_{lat} = N + (L^2 + D^2)^{1/2} + M + (X^2 + D^2)^{1/2} + Y + (Z^2 + D^2)^{1/2} + P + (Q^2 + D^2)^{1/2} + R$

In many researches which offered the concept of remote latency assumed that local and remote operating system overhead are the same [4]. Then the above equation could be rewritten as :

$$T_{lat} = 5M + (L^2 + D^2)^{1/2} + (X^2 + D^2)^{1/2} + (Z^2 + D^2)^{1/2} + (Q^2 + D^2)^{1/2} \quad (7)$$

### 5. Simulation Analysis and Results

Validation and analysis of the proposed algorithm is done via simulation the parameters  $\Phi$ ,  $r$ ,  $M$  and  $\theta$  are generated randomly, the speedup of the system is calculated at different values of  $r$  and  $\theta$  and constant values of  $\Phi$  and  $M$ , the speedup is an indicator to the performance of the

system that refers to the ratio between Tlat and TlatPred ,this ratio increases when the performance increases and vice versa.

### 5.1 Performance Study and Analysis

Equation 6 represents the ratio of performance of memory sharing system and the role of memory sharing predictor in increasing the performance of the system; it is obvious that there is a relationship between the performance and the radius r that represents the prediction accuracy, the following results and graphs are calculated at constant wire time and overheads.

At  $\Phi = 60$  ,  $r = 0.5$  ,  $M = 0.6$

<b><math>\theta</math></b>	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.558	0.650	0.712	0.790	0.896	1.051	1.30	1.79	3.254

At  $\Phi = 60$  ,  $r = 0.55$  ,  $M = 0.6$

<b><math>\theta</math></b>	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.532	0.621	0.679	0.755	0.857	1.006	1.25	1.72	3.129

At  $\Phi = 60$  ,  $r = 0.6$  ,  $M = 0.6$

<b><math>\theta</math></b>	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.508	0.593	0.65	0.723	0.822	0.966	1.19	1.65	3.013

At  $\Phi = 60$  ,  $r = 0.65$  ,  $M = 0.6$

<b><math>\theta</math></b>	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.485	0.569	0.624	0.695	0.789	0.928	1.15	1.59	2.905

At  $\Phi = 60$  ,  $r = 0.7$  ,  $M = 0.6$

<b><math>\theta</math></b>	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.466	0.546	0.599	0.668	0.759	0.894	1.11	1.54	2.806

At  $\Phi = 60$  ,  $r = 0.75$  ,  $M = 0.6$

<b><math>\theta</math></b>	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.447	0.525	0.576	0.643	0.732	0.862	1.07	1.48	2.712

At  $\Phi = 60$  ,  $r = 0.8$  ,  $M = 0.6$

<b><math>\theta</math></b>	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.430	0.505	0.555	0.619	0.706	0.832	1.04	1.44	2.624

Figure 5 shows that at different values of wire time in case of prediction using MSP increases. The analysis of the results of the proposed mathematical model that uses extra parameters shows an important notation, with increasing the radius r of the reflexive relation (means more delay in prediction process and less accuracy), the performance decreases by an amount greater than the amount mentioned in previous mathematical model of memory sharing predictor in [12,16]. For example, the results of the proposed mathematical model show that at

prediction accuracy 70% at  $r = 0.65$  ( the ideal performance occurs at  $r = 0.5$  or at unit diameter) the performance decreases with 13%, in previous model [12,19], at prediction accuracy 70% the performance decreases with 5%. The results of the proposed mathematical model show more accurately the effect of the prediction ,the role of MSP on the performance and how the increasing of prediction accuracy affects the increasing of performance of memory sharing systems.

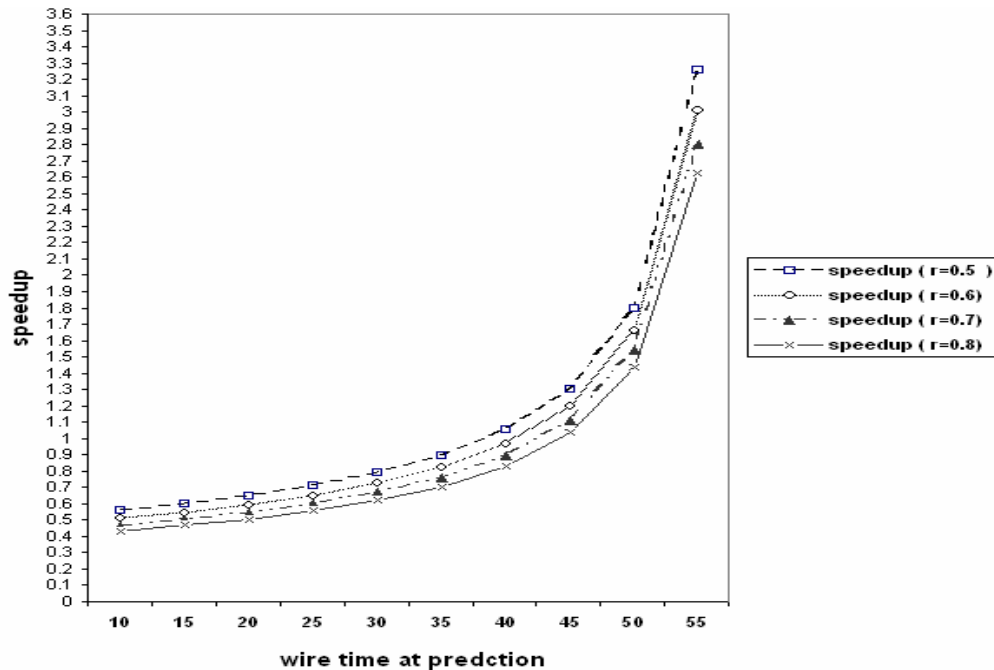
At  $\Phi = 60$  ,  $r = 0.5$  ,  $M = 0.8$

$\theta$	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.636	0.738	0.806	0.892	1.01	1.18	1.46	2	3.615

At  $\Phi = 60$  ,  $r = 0.5$  ,  $M = 1$

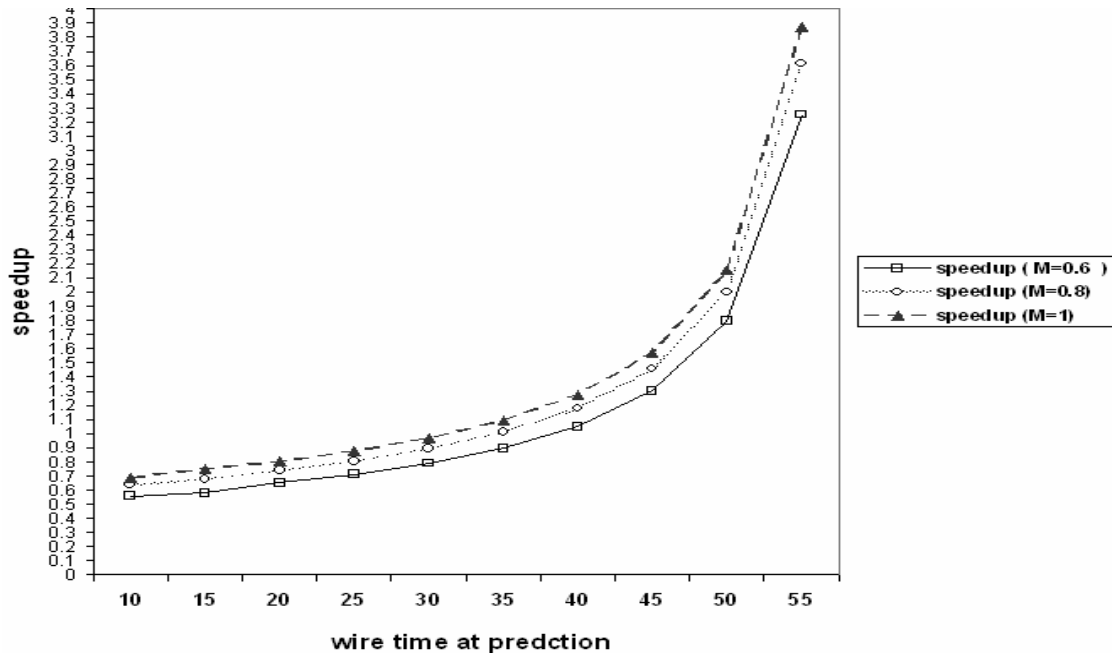
$\theta$	10	20	25	30	35	40	45	50	55
<b>Speedup</b>	0.69	0.803	0.876	0.968	1.093	1.275	1.57	2.15	3.874

Figure 6 shows the performance of the system at different values of actual overheads occur at prediction process ( $\Phi - \theta$ ) and different values of ideal overheads of the operating system M.



**Figure 5 s increasing of Speedup using prediction**

It is clear that by increasing the difference between  $\Phi$  and  $\theta$  ( $\Phi - \theta$ ) which represents an increasing in the actual overhead the performance decreases, so this paper suggests to use a low constant synchronization overhead operating system called K42/Tornado.



**Figure 6 Decreasing of Speedup by increasing the actual overhead ( $\Phi - \theta$ ) at prediction**

K42/Tornado is developed by K42, it is a new high performance, open source, general-purpose research operating system kernel for cache-coherent and large-scale shared-memory multiprocessors, such as NUMAchine, must be specifically designed for this class of system if the operating system and the applications that run on it are to perform well. For example, the large disparity between cache and main memory access times dictate that data sharing be minimized in order to minimize cache misses and reduce consistency traffic. The generally low cache hit rates expected in executing operating system code dictate that memory access locality be made high so average memory access times are low. Parallelism in servicing independent operating system requests must increase proportionally to the number of processors in the system if bottlenecks in the operating system won't start appearing as the system is scaled. The operating system must be flexible in allowing applications to customize operating system policies to suit their performance needs, since these needs tend to vary greatly and any single default policy will only perform well for a small percentage of applications.

Tornado is a new operating system is developed for NUMAchine that addresses these issues using novel approaches, some of which were developed for our previous operating system Hurricane[3,5,6,11,19]. It uses Hierarchical Clustering, a modular and flexible structure that can be applied to operating system objects or services. Using this structure, operating system service points are automatically replicated to provide for concurrency that increases linearly with the number of processors in the system, yet at a low constant synchronization overhead.

## **6.CONCLUSION**

Memory Sharing Predictor plays an important role for increasing the performance in shared memory systems by neglecting the overhead time by the coherence protocol using speculations. A new MSP mathematical model that takes into account most of parameters that affect the prediction is proposed. Comparison of these results obtained with and without prediction shows an important notation, with more delay in prediction process and less accuracy. The performance with system prediction using the proposed mathematical model is improved when compared with those calculated in previous mathematical model of memory sharing predictor. the proposed mathematical model show more accurately the effect of the prediction ,the role of MSP on the performance and how the increasing of prediction accuracy affects the increasing of performance of memory sharing systems. This paper also suggests to use a new developed and a low constant synchronization overhead operating system called K42/Tornado to increase the performance of memory sharing systems.

## **7.REFERENCES**

- [1] Anant Agarwal. Performance tradeoffs in multithreaded pro-cessors. IEEE Transactions on Parallel and Distributed Sys-tems, 3(5):525–539, September 1992.
- [2] Jonathan Appavoo, Marc Auslander, Maria Burtico, Dilma Da Silva, Orran Krieger, Mark Mergen, Michal Ostrowski, Bryan Rosenburg, Robert W. Wisniewski, Jimi Xenidis, "K42: an Open-Source Linux-Compatible Scalable Operating System Kernel", IBM Systems Journal to appear 2005.

- [3] Andrew Baumann, Jonathan Appavoo, Dilma Da Silva, Orran Krieger, and Robert W. Wisniewski, "Improving Operating System Availability With Dynamic Update", OASIS (Workshop on Operating System and Architectural Support for the on demand IT InfraStructure) pps 21-27, October 9, 2004, Boston Massachusetts.
- [4] Dilma Da Silva, Livio Soares, Orran Krieger, "KFS: Exploring Flexibility in File System Design", Brazilian Workshop on Operating Systems (WSO'2004), Salvador, Brazil, August 2004.
- [5] Erik Hagersten and Michael Koster. WildFire: A scalable path for SMPs. In Proceedings of the Fifth IEEE Symposium on High-Performance Computer Architecture, pages 172– 181, February 1999.
- [6] Ben Gamsa, Orran Krieger, Jonathan Appavoo, Michael Stumm, Tornado: Maximizing Locality and Concurrency in a Shared Memory Multiprocessor Operating System, USENIX OSDI 1999.
- [7] Steven K. Reinhardt, James R. Larus, and David A. Wood. Tempest and Typhoon: User-level shared memory. In Proceedings of the 21st Annual International Symposium on Computer Architecture, pages 325–337, April 1994.
- [8] Wolf-Dietrich Weber, Stephen Gold, Pat Helland, Takeshi Shimizu Thomas Wicki, and Winfried Wilcke. The Mercury interconnect architecture: A cost-effective infrastructure for high-performance servers. In Proceedings of the 24th Annual International Symposium on Computer Architecture, May 1997.
- [9] P. Keleher, "The Relative Importance of Concurrent Writers and Weak Consistency Models," in Proceedings of the 16th International Conference on Distributed Computing Systems, 1996.
- [10] Daehyun Kim, Mainak Chaudhuri : "Architectural Support for Uniprocessor and Multiprocessor Active Memory Systems" IEEE TRANSACTIONS ON COMPUTERS, VOL. 53, NO. 3, MARCH 2004
- [11] L. Kontothanassis and M. Scott. Using Memory-Mapped Network Interfaces to Improve the Performance of Distributed Shared Memory. In Proceedings of the 2nd IEEE Symposium on High Performance Computer Architecture, February 1996.

- [12] An –Chow Lai and Babak Falsafi. “ Memory Sharing Predictor: The Key to a Speculative Coherent DSM.”, Proceedings of the 26th International Symposium on Computer Architecture (ISCA 26), 1999.
- [13] Ammar Qusaibaty - Newton Howard: “DISCRETE STRUCTURES “,Depatement of computer science, G.W university.
- [14] J. Oplinger and M. Lam. “Enhancing software reliability using speculative threads”. In Proceedings of the Conference on Architectural Support for Programming Languages and Operating Systems, October 2002.
- [15] S. S. Mukherjee and M. D. Hill. Using prediction to accelerate coherence protocols. In Proceedings of the 25th annual international symposium on Computer architecture, pages 179–190. IEEE Press, 1998.
- [16] Shubhendu S. Mukherjee and Mark D. Hill :” Using Prediction to Accelerate Coherence Protocols.” the Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA), 1998.
- [17] Cristian T, Justin D. Smith, and Jason Hickey: “Kernel Level Speculative DSM”, DARPA,,2002
- [18] Sarita V. Adve and Kourosh Gharachorloo. Shared memory consistency models: A tutorial. IEEE Computer, 29(12):66– 76, December 1996.
- [19] Tse-Yuh Yeh and Yale Patt. Alternative implementations of two-level adaptive branch prediction. In Proceedings of the 19th Annual International Symposium on Computer Architecture, May 1992.
- [20] W. Yu, C. Amza, A. L. Cox, S. Dwarkadas, P. Keleher, H. Lu, R. Rajamony, and W. Zwaenepoel, “TreadMarks: Shared Memory Computing on Networks of Worksta-tions,” IEEE Computer, pp. 18--28, February 1996.