IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 08, Issue 9 (September. 2018), ||V (V) || PP 41-46

Adaptive Directory based Cache Coherency Model to Optimize the Network Bandwidth in Distributed Applications

Subrahmanya B[1], Dr. V. Vijayakumar [2]
[1](Research Scholar (PPCOMP.SCI 0343), Dept CS, Rayalaseema University, A.P, India)
[2](Dean, Dept CS&E, Anurag Group of Institutions, Hydrabad, Telangana, India)
Corresponding Author: Subrahmanya B

Abstract: Cache management has undergone various dimensions in the Computational field, by keeping the objective of computational performance for High Performance Computation(HPC). Cache Management in Multiprocessor Systems leads to a new problem called Cache Coherency Problem. This problem basically addresses the issue of maintaining consistency among Memory and Cache contents. Even in Distributed Systems, Cache Coherency Problems needs to be addressed and Directory Based Cache Coherency Model is one such solution for this Problem.
This work basically on implementing a Directory Based Cache Coherency Model on a Hierarchically designed Distributed System and analyzing the network performance in terms of various Network Metrics mainly, In Throughput, Out Throughput, In Out Throughput and Dropped Packets at a given nodes interface using a network simulator NCTUns 5.0. Later improving this model by adding Adaptive nature, then analyzing and comparing the network performance in terms of same Network Metrics, with that of non-adaptive implementation. Performance gain with Adaptive Model will be quantified in terms of percentage.

Keywords : Distributed System, High Performance Computation, Coherency Problem, Adaptive, Computational, Multiprocessor Systems, In Throughput, Out Throughput, In Out Throughput.

Date of Submission: 29-09-2018 Date of acceptance: 14-10-2018

I. INTRODUCTION

Cache memory is used at CPUs to reduce the average memory access time and hence to achieve HPC. In a shared memory multiprocessor system, there will be multiple levels of caches in the memory hierarchy. Each processor may read data and store it in its cache. This result in copies of the same data being cached in different processors at the same time and it leads to Coherency Problem. At any time, a read by another processor should return the last value written, which is the latest value of data. To avoid the problem of reading stale data, all processors with copies of the data must be notified of the changes by any one processor. Maintaining this consistency among multiple copies of same data is called Coherency Problem[13]. To address this issue there are few Cache Coherence Protocols[10]. They are classified as Snoopy Protocol[10] and Directory Protocols[10] based on the technique by which they implement. A directory-based protocol is being implemented on an arbitrary interconnection network as a solution for cache consistency problems in Distributed Applications. This Directory Protocols have the advantage of scalability factor and also suits best for Distributed Applications[9]. The directory works as a look-up table for each processor to identify coherence and consistency of data which is currently being updated. For each data item, an identified node called Home Node, maintains a directory to store the details of its data in terms of value, sates and sharer details. The protocol enforces every data requests to go through its Home Node so as to maintain data consistency. This work is basically implementation of a Directory Based Cache Coherency Protocol on a hierarchically designed Distributed System and analyzing its performance in terms of In Throughput, Out Throughput, In Out Throughput and Droped Packets. Later the same model is improved by adding adaptive nature and its performance is analyzed and compared with that of non-adaptive model in terms of same network metrics.

II. METHODOLOGY

Using the network simulator *NCTUns 5.0*, a hierarchical Distributed Computing System has been designed as shown in the figure 1. With the nodes being organized in a hierarchical fashion, as far as possible, the different fragments of large databases are made available at respective zones locally so as to improve the Applications Locality of Reference and hence the system performance. Here the nodes are named with label D, M, W, T, B, H, and C just to represent different state capitals of country India.

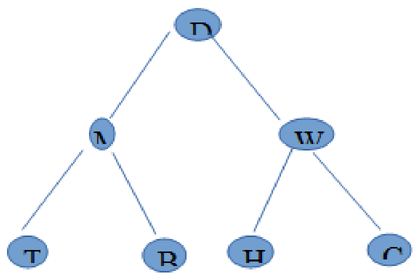


Figure 1. Hierarchical Distributed System

Since Directory Based Cache Coherency Model has been enforced to implement Cache Coherency, each of the nodes here maintains a directory for the fragments of data which has been kept at resprctive nodes. At any point of time, the model identifies a given node at six different states as mentioned below.

- *Read Requesting Node*: Any node which is sending a Read request message on the data
- *Readx Requesting Node*: Any node which is sending a Readx (Exclusive Read) request message on the data
- *Home Node*: It is that node where the data in question, happens to be stored.
- *Owner Node*: It is that node where the data of some Home Node happens to be copied and also modified to a new value but not written back to its Home Node.
- *Sharing Node*: It is that node where the data of some Home Node happens to be copied only for reading it.
- *Recent Owner/Sharing Node*: It is that node having the copy of DATA which is currently being requested and also nearest to the requesting node as far as possible. This special node is going to be identified only with our Adaptive Model. In fact it is a node having the current copy of DATA apart from its Home Node.

There are four different cases of data requests which needs to be considered for the implementation -like, Case 1, Case 2, Case 3 and Case 4 as illustrated below.

Case 1: A Read Request on a Clean Block
Following Messages Required for the Directory Coherency in Case 1.
1---> {R.DATA} - Read Request to Home Node from data requesting node
2---> {DATA} - Data reply to from Home Node
Case 2: A Readx Request on a Clean Block
Following Messages Required for the Directory Coherency in Case 2.
1---> {Rx.DATA} - Readx Request to Home Node from data requesting node
2---> {DATA} - Data reply from Home Node

Case 3: A Read Request on a Dirty Block
Following Messages Required for the Directory Coherency in Case 3.
1---> {R.DATA} - Read Request to Home Node from data requesting node
2---> {R.DATA} - Read Request to Owner Node from Home Node
3---> {DATA} - Data Reply to Home Node from Owner Node
4---> {DATA} - Data Reply to the requesting node from Home Node
(Note: Message 3 will be dispatched only after committing the transaction at Owner Node W)

With adaptive model in Case 3, the message no 2 needs to be be modified by including the ID of DATA requesting Node and message no 3 will include TS, Time Stamp value of Owner Node. If the data requesting node falls under sub tree of Owner Node, then DATA will be written to the requesting node immediately with its TS by the Owner Node itself. Requesting node will forward this TS as an acknowledge message to Home Node for receiving the DATA.

Case 4: A Readx Request on a Sharing Block
Following Messages Required for Directory Coherency in Case 4.
1---> {Rx.DATA} - Readx Request to Home Node from data requesting node
2---> {I.DATA} - Invalidation Requests to all Sharing Nodes from Home Node
3---> {ACK} - Invalidation Ack to Home Node from all Sharing Nodes
4---> {DATA} - Data to requesting node from Home Node
5--> {F.DATA, ID}

With adaptive model in Case 4, the message no 2 needs to be modified by including the ID of DATA requesting Node. If the requesting node falls under sub tree of any sharer node, then Home Node will send additional {F.DATA, ID} message to the Recent Sharing Node so as to forward the copy of its DATA along with its Time Stamp TS, which has been locally copied by this nearest sharer node. Immediately after forwarding the DATA, Recent Sharing Node sends its TS to Home Node as acknowledgement. After receiving the DATA with TS, Readx requesting node also forwards this TS to Home Node as a acknowledge of received DATA. So additional SYNC message (no 5) will be sent with the adaptive model.

III. IMPLEMENTATION

In order to implement we have used *NCTUns 5.0* network simulator because our major concern is to measure the network performance in terms of network metrics, In Throughput, Out Throughput, In Out Throughput and Number of Packets Dropped at a specific interface of given of given Home Node for varying Bandwidth and Bit Error Rate (BER). In this implementation, the node C is Read/Readx Requesting Node and the node M is Home Node for all four cases. Similarly the node W is Owner Node in Case 3 and the node B as well as the node W are Sharer Nodes in Case 4. For the implementation, a standard DATA size of 1024 Bytes for any PAYLOAD message *{DATA}* and size of 128 Bytes for all SYNC messages like *{R.DATA}*, *{I.DATA}* etc has been considered. Results for two different Bit Error Rates (BER), i.e. at 0 BER value as well as at 0.04 BER value for the network links have been recorded. The Bandwidth value is kept initially at 10 Mbps and network performance has been analyzed, in terms different metrics, at a specific interface of given Home Node. Similar observations are made by changing the bandwidth value to 5 Mbps and also with 2.5 Mbps. The Same environment has been applied even for Adaptive model and its results are compared with that of non- adaptive model implementation. The entire implementation has been done by taking in to consideration a steady network load by applying traffic between the nodes with in left half sub tree and right half sub tree of the distributed system.

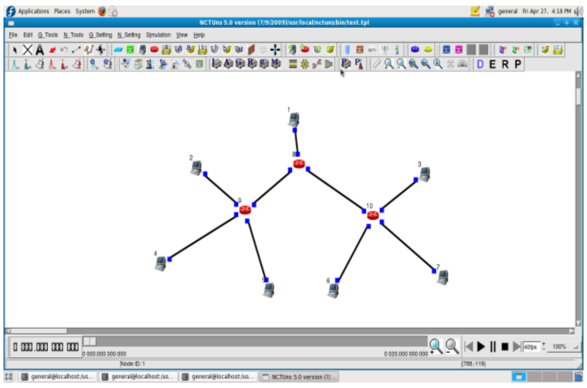

NCTUns 5.0 version (7/9/2009)/usr/local/nctuns/bin/test.tpl
File Edit G_Tools N_Tools G_Setting N_Setting Simulation View Help
D E R P
1
2
3
4
6
7
9
10

Figure 2. NCTUns 5.0 Simulator

IV. RESULTS

Figure 3 to Figure 6 shows the comparison of results with adaptive versus non adaptive models, in terms of different identified network metrics, using the simulator NCTUns 5.0. Here X axis shows time in seconds where as Y axis shows data packet transfer in kilo bytes per second (Kbps). Plot in blue corresponds to the adaptive model and plot in red corresponds to the non adaptive model.

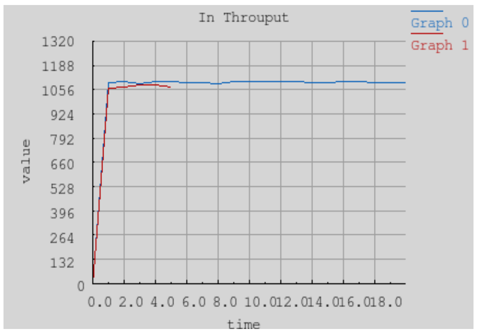


Figure 3. In Throughput Comparison.

Results on In Throughput comparison shows that an average gain of 3.3 % is recorded with adaptive model.

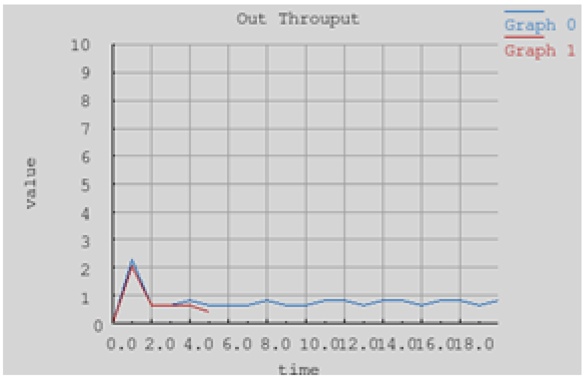


Figure 4. Out Throughput Comparison.

Results on Out Throughput comparison shows that an average gain of 6.5 % is recorded with adaptive model.

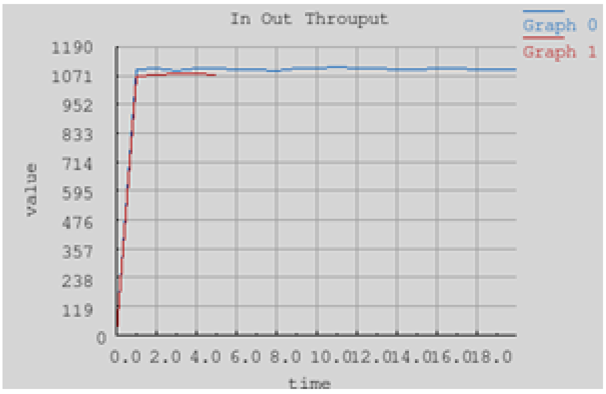


Figure 5. In Out Throughput Comparison.

Results on In Out Throughput comparison shows that an average gain of 2.8 % is recorded with adaptive model.

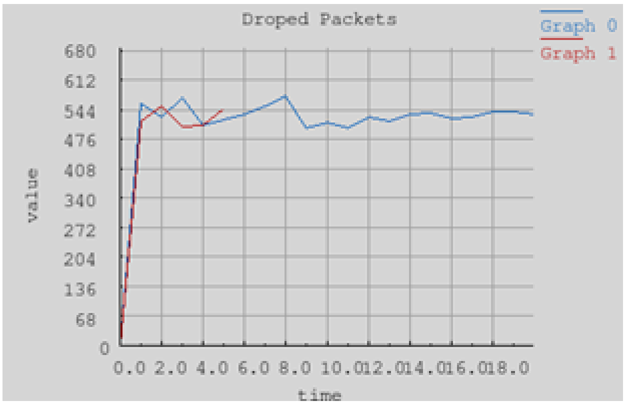
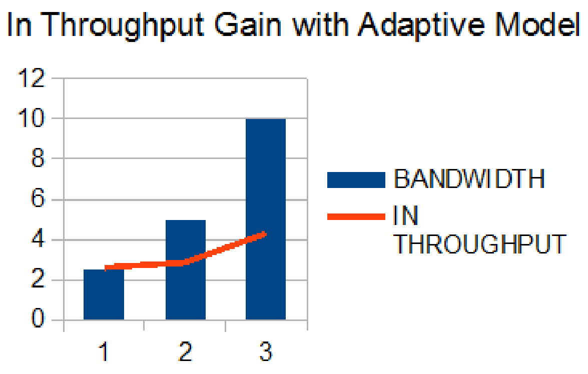
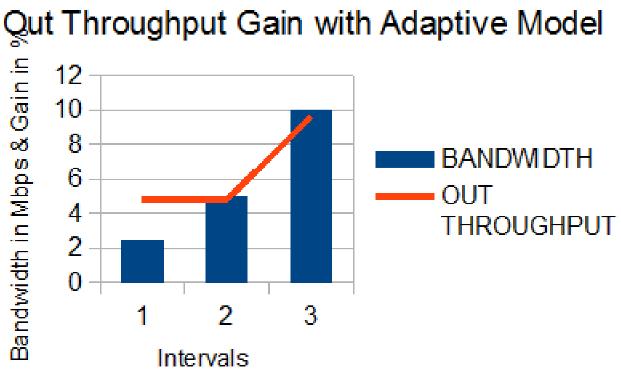


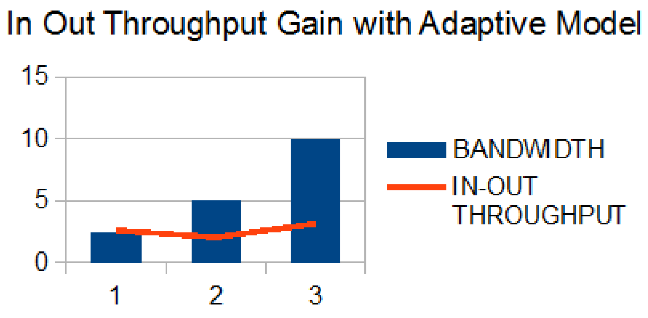
Figure 6. Dropped Packets Comparison.

Results on Dropped Packets comparison shows that an average gain of 2.03 % is recorded with adaptive model.

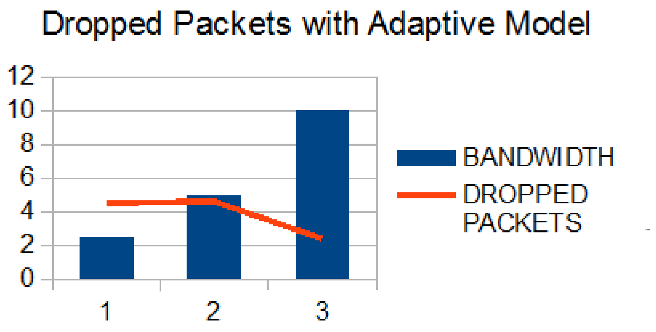


X Axis: Intervals, Y Axis: Bandwidth in Mbps & Gain in %





X Axis: Intervals, Y Axis: Bandwidth in Mbps & Gain in %



X Axis: Intervals; Y Axis: Bandwidth in Mbps & Dropped Packe

V. CONCLUSION

On considering the overall results recorded in terms of all the four Network Metrics that has been considered in this work, it has been found that, adaptive model results with an average gain of 3.6% compared to that of non-adaptive model. The gain achieved with the adaptive model is high at higher Bandwidth value and decreases as Bandwidth value decreased. Also the gain is marginal at very low Bit Error Rate value and increases as links Bit Error Rate increased from a ideal value of 0 to a practical value of 0.04. But on increasing this error rate beyond a critical value, the gain should also fall drastically, which can be considered as a Thrashing Effect.

REFERENCES

[1]. Shou-Chih Lo, Jhih-Sian Hu, Varsha A. Kshirsagar, Neighborhood-Based In-Network Caching for Information-Centric Networks. Int. J. Communications, Network and System Sciences, 2017, 10, 76-87

[2]. Sarah Tasneem 1 , Reda Ammar 2, Performance Study of a Distributed Web Server: An Analytical Approach. Journal of Software Engineering and Applications, 2012, 5, 855-863.

[3]. Rizik M. H. Al-Sayyed, Fawaz A. Al Zaghoul, Dima Suleiman, Mariam Itriq, Ismail Hababeh. New Approach for Database Fragmentation and Allocation to Improve the Distributed Database Management System Performance. Journal of Software Engineering and Applications, 2014, 7, 891-905

[4]. Hang Qin, Li Zhu Adaptive Cache Allocation with Prefetching Policy over End-to-End Data Processing. Journal of Signal and Information Processing, 2017, 8, 152-160

[5]. Arif Sari, Murat Akkaya, Fault Tolerance Mechanisms in Distributed Systems. Int. J. Communications, Network and System Sciences, 2015, 8, 471-482

[6]. Dimitris Tsaliagos, Design and Implementation of a Directory based Cache Coherence Protocol, Technical Report FORTH-ICS/TR-418 May 2011

[7]. Michel Dubois, Member- IEEE, and Faye A. Briggs, Member-IEEE, Effects of cache coherency in Multiprocessors, Technical Report FORTH-ICS/TR-206 May 2011

[8]. Subrahmanya Bhat & Dr. K. R. Kamath. (August,2016). Directory Organizations in Cache Coherency Systems for High Performance Computation. International Journal of Current Research and Modern Education (IJCRME), I(I), 868-871, ISSN (Online): 2455 – 5428

[9]. Subrahmanya Bhat & Dr. K. R. Kamath. (May 2016). Directory Based Cache Coherency Protocol In Multi-Core System For High Performance Computation. International Journal of Current Research and Modern Education (IJCRME), (I/I), 257-261, ISSN : 2455 – 5428.

[10]. Subrahmanya Bhat B and Dr. K.R Kamath. (July 2015). Cache Hierarchy In Modern Processors And Its Impact On Computing. International Journal of Management, IT and Engineering (IJMIE), 5(7), 248-253, ISSN: 2249-0558, I.F. 5. 299

[11]. Subrahmanya Bhat B and Dr. K.R Kamath. (2015). Directory Based Cache Coherency, Organization, Operations And Challenges In Implementation –Study. International Journal of Advanced Trends in Engineering and Technology (IJATET)., ISSN (Online): 2456 - 4664 (www.dvpublication.com), 1(1), 30-33.

[12]. Subrahmanya Bhat, & Dr. K. R. Kamath. (2016). Effective Learning With Usage of Simulators – A Case of Nctuns Simulator In Computer Networks. International journal of Scientific Research and Modern Education, I (I), 415-420. ISSN-2455 – 5630.

[13]. S.Y. Wang and H.T. Kung. (October 2002). “A New Methodology for Easily Constructing Extensible and High-Fidelity TCP/IP Network Simulators,” Computer Networks, Vol. 40, Issue 2, pp. 257-278.

[14]. S.Y. Wang. (July 2003). “NCTUns 1.0,” in the column “Software Tools for Networking,” IEEE Networks, Vol. 17, No. 4.

[15]. S.Y. Wang, C.L. Chou, C.H. Huang, C.C. Hwang, Z.M. Yang, C.C. Chiou, and C.C. Lin. (June 2003). “The Design and Implementation of NCTUns 1.0 Network Simulator,” Computer Networks, Vol. 42, Issue 2, pp.175-197

[16]. S.Y. Wang, C.L. Chou, C.C. Lin. (2007). “The Design and Implementation of the NCTUns Network Simulation Engine,” Elsevier Simulation Modelling Practice and Theory, 57 -81.

Subrahmanya B. " Adaptive Directory based Cache Coherency Model to Optimize the Network Bandwidth in Distributed Applications” IOSR Journal of Engineering (IOSRJEN), vol. 08, no. 9, 2018, pp. 41-46.