A Methodology for Distributed Virtual Memory Improvement

Sahel Alouneh[1], Sa'ed Abed[2], Ashraf Hasan Bqerat[2], and Bassam Jamil Mohd[2]

[1] Computer Engineering Department, German-Jordanian University, Jordan
sahel.alouneh@gju.edu.jo
[2] Computer Engineering Department, Hashemite University, Jordan
{sabed,bassam}@hu.edu.jo, aihetchbe@hotmail.com

Abstract. In this paper, we present a methodology for managing the Distributed Virtual Memory (DVM). The methodology includes distributed algorithms for DVM management to detect the memory status which will enhance previous DVM techniques. The DVM data structure tables are similar to those found in current Conventional Virtual Memory (CVM) with some modifications. Finally, we evaluate our methodology through experimental results to show the effectiveness of our approach.

Keywords: Distributed Virtual Memory (DVM), Conventional Virtual Memory (CVM), Page faults.

1 Introduction

Distributed Virtual Memory (DVM) is a technique which exploits all of the first storage devices (commonly RAMs), in such way that it maximizes the utilization of these devices as much as possible depending on the techniques used in Page out, Page replacement and Page fault. For example, if a page has to get out of the memory of a certain node, instead of sending it to its Hard Disk (HD), which will consume a massive time in storing and retrieving this page, this node may ask other nodes in the system to store its page in their memory in order to get it back when needed. Of course the enhancement here comes from the fact that memory to memory transfer time is much less than memory to HD transfer time. The aim of this work is to make load balancing through the distribution of some processes of a certain node to other nodes which may have some unused resources and thus leads to an increased throughput of the system.

DVM is used in multi-node systems which contains many nodes. The node is a single independent computer (which has its own CPU, Memory and input/output subsystems). So, our targeted system should contain the following:

- Nodes,
- Backbone network,
- A protocol which controls the communication process between these nodes.

A. Abd Manaf et al. (Eds.): ICIEIS 2011, Part III, CCIS 253, pp. 378–384, 2011.

Based on this, DVM technique adds another level of storage devices named as the memory of other nodes as shown in Figure 1.

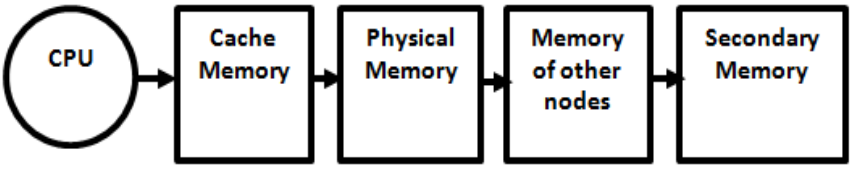

Fig. 1. Memory hierarchy of the DVM

DVM was introduced by many researchers at the level of processes and at level of pages. In this paper, we are interested in DVM at the level of pages. Memory sharing at the level of pages was firstly introduced by Li [1, 2], Clancey and Francioni [3], Abaza and Malkawi [4, 5], later on Abaza and Fellah [6] and also by others [7].

In [4], the author proposed a new way to manage virtual memory through the use of a node page table (NPT) instead of using process page table as in Conventional Virtual Memory (CVM). The node page table keeps information about the pages that are currently resides at the node and the pages departed the node to an immediate neighbor. Each entry of NPT contains the name of the owner process (PID), virtual page number (VPN), the physical address of the page if currently located in memory, and the destination of the page if transferred to another node.

Later on an optimization was carried out by Qanzu'a [8] to replace the NPT by two new structures which are: PPT (Process page table) and PrPT (Processor Page Table). He simulated his work and proved that it has improved the throughput of the whole system compared to [4].

Our work mainly states the problem of the second back storage devices which commonly are HDs (Hard desks) which use the magnetic nature to encode and store data in terms of north and south poles. Due to mechanical and physical limitations, these kinds of storage devices are much slower compared to CPU speed, caches speed and memory (most of times it is RAM) speed which is used as a first storage device commonly. Of course, we have a lot of options to go over this problem such as:

- Increasing cache size and memory size (need Hardware).
- Using some techniques for the pages and processes to decide which one should reside in memory and which should not, aiming to decrease the thrashing level as much as possible which leads to minimize the necessity of the HD (needs massive processing time) such as Least Recently Used (LRU) technique.
- Using DVM technique which we will spotlight on and introduce some methods that were done by many researchers and compare with them.

Our optimization will be modeled in terms of decreasing the traffic over the backbone network and decreasing the time needed to find a page that is reclaimed by its own original node.

We have deferent data structures that may implement DVM such using NPT as proposed in [4] and its optimization method in [8] which uses Process Page Table (PPT) and Processor Page Table (PrPT). Our work will have two steps: first, we will use the approach in reference in [8] and then divide the system into clusters. Second, we will add another memory level which is a cache for each cluster. A question may arise, what the benefits shall we have by adding a cache for each cluster? The points below summarize these benefits:

- Caches are used for the aim of:
- Reducing access time to find the node which may have a page related to another node.
- Reducing the traffic on local lines inside each cluster.
- The optimization comes from reducing the number of pages moving between different nodes which results in reducing the traffic.
- The amount of optimization will depend on caching techniques.

Thus, considering the above points will result in the following advantages:

1. Increase the scalability of the system by a massive factor.
2. Increase throughput through the time saved in caches' hits.

The structure of this paper is organized as follows: Section 2 describes the proposed methodology. Section 3 evaluates the proposed methodology through some experimental results. Finally, Section 4 concludes the paper with directions for future research.

2 Design Methodology

The DVM technique is used to increase the system throughput with some drawbacks such as: high traffic density and other problems that will be discussed later. Our methodology is based on adding another level of memory which is the cluster cache by dividing the system into clusters and each cluster has its own cache (memory). We will show the criteria step by step based on enhancements on previous techniques presented in reference [8].

Figure 2 illustrates our methodology and compares it with CVM and DVM techniques as well as shows different approaches that organize paged memory. For CVM, when a process, during execution, needs a page it directly asks for it in the cache memory (1). Then, if cache miss happens, it will ask the physical memory for such a page (2). If also page is not found, the process will go directly to the second storage device (5) to bring the page which will be so slow. Thus, pages in CVM are exchanges between memory and the second storage device. While in DVM, the process asks for the page in other nodes' memory (4) before going to the second storage device (5) in case the page is not found in other nodes. Thus, pages in DVM are exchanged firstly between memory and other nodes' memory, and secondly, between memory and second storage device. In our model, the process will ask for the page following 1,2,3,4,5 route.

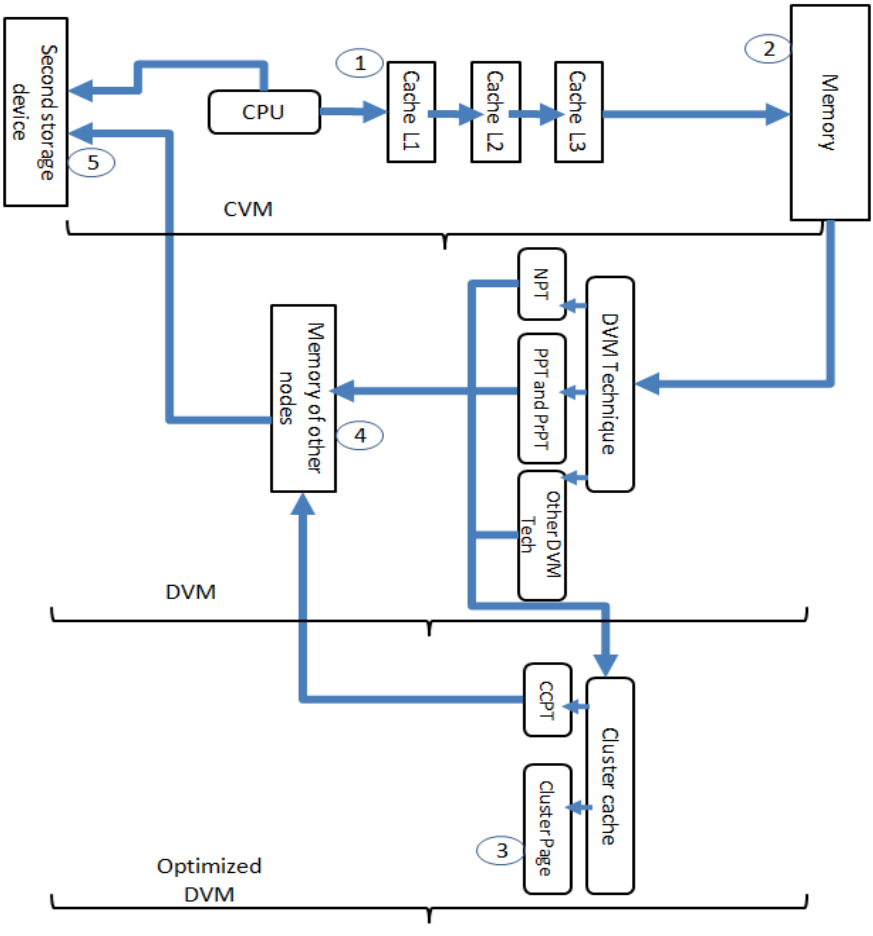

Fig. 2. CVM, DVM and optimized DVM Models

The methodology is based on dividing the whole system into different groups and making one of its node as a master node which has to be the most advanced and capable node. This master node has some additional functionality to be done. First, we will take off 10% of its memory and name it a cluster cache which seems reasonable especially with current computer capabilities which has RAM larger than 2GB. This cluster cache has to be accessible by all nodes of the local cluster. Also, it will contain the LLN table and it will have some pages of the pages of the local nodes.

Our methodology for page look up is illustrated in the following algorithm. When a process p is asking for page x which resides at node N, firstly, check page x at node n which is the local node of process p. If page x is in memory of node n, then page is found otherwise, check the node's cluster cache to see if page resides in it. If so, then the page is found and cache hit is made, otherwise check the LLN table to see if page

is at one of other nodes either in its cluster or out of it. If so, then page is found otherwise, page is residing in node's n HD. Thus, in our methodology it is clear that the aim of placing a cluster cache is to increase the throughput by cache hits which are clearly a distinguished approach in DVM.

Algorithm: A process P is asking for page X which resides in node N.

1. *Process startCheck page X at node N*

   If (Page X is in memory of node N) then
   Page found go to START
   Else
2. *Check the nodes cluster cache to see if page resides in it*

   If (Page X is in cluster cache) then
   page is found go to START
   Else
3. *Check the nodes cluster cache (Cluster Page Table CCPT) to see if page is at one of its node*

   If (Page X is in memory of node M<> N) then
   page is found go to START
   Else
4. *Check the other clusters' caches by send messages to see if the page resides in it*

   If (Page X is in cluster cache) then
   page is found go to START
   Else
5. *Check the other clusters' caches by send messages to see if the page is in its*

   (Cluster Page Table CCPT) cache to see if page is at one of its node
   If (Page X is in memory of node k<> N& M) then
   Page is found goto START
   Else
   Goto HD of Node N to bring the page
   END

3 Simulation Results

Simulation has been carried out for our methodology in terms of page fault rate and the results proved our enhancement over [8] and also over CVM.

In our simulation, we considered a system with 12 nodes; each 4 of them formulate a cluster. So, we have 3 clusters with 3 master nodes. They are connected in a star topology. Each node has 50 frames of memory. And has large HD. The cluster cache is at each master node with 5 frames size so the master node has 45 frames. Also, work load varies from 82 to 263 pages.

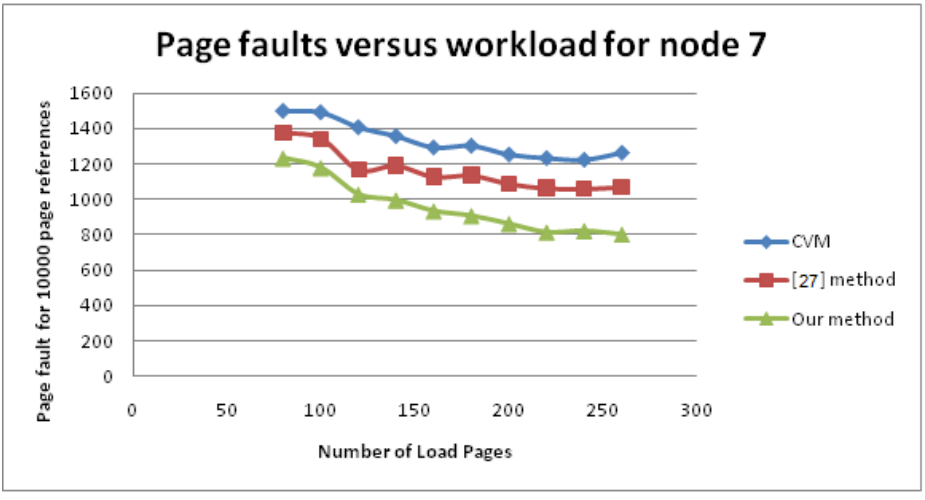

Fig. 3. Page Faults for Every Approximately 10000 Page References

Figure 3 shows the page fault rate compared to CVM for node 7. X-axis shows number of pages of the work load and Y-axis shows number of page faults for approximately 10000 page references. Notice that the enhancement that our methodology has over [8] method in terms of page faults. Also, as the work load increases, fault rate decreases compared to CVM and [8] method. So our methodology becomes more efficient as work load increases, until it shows some stability at the right most side. But on the other hand, as work load increases context switching will increase, so there is a trade off. From simulation we can notice that page faults rate decreased at average of 25% of CVM, and 15% of [8] method, which is considered as a good enhancement.

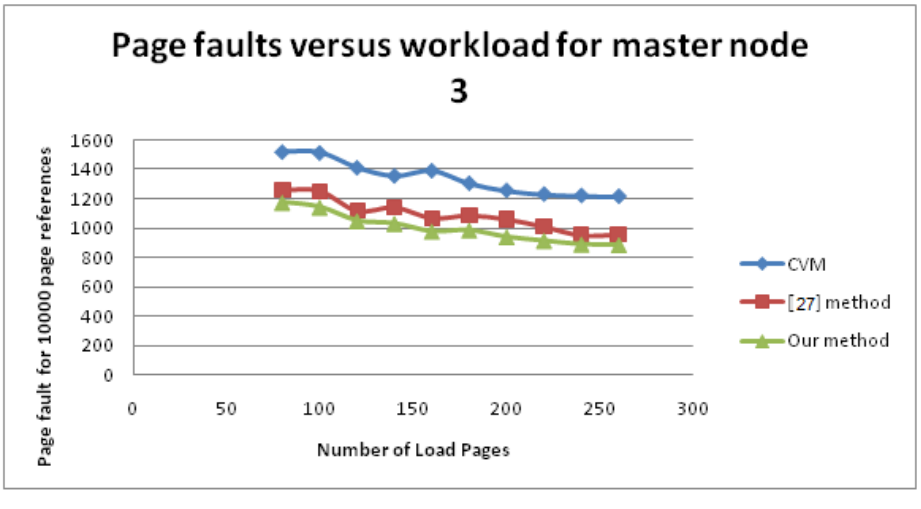

Fig. 4. Master Node Page Faults for Every Approximately 10000 Page References

Figure 4 shows the page fault rate to work load for node 3 which is a master node of cluster 1. Using our methodology, [8] methodology and CVM. Notice that the page fault rate compared to CVM has been decreased by average of 12.5 %. The enhancement over [8] is not obtained since the master node has smaller memory equals to 45 frames which have an effect over number of page faults.

4 Conclusion and Future Work

As was introduced, our work mainly depends on dividing the whole system into clusters and adding a cache for each one. As seen in the simulation, our methodology has optimized the DVM by increasing system throughput and decreasing the traffic in backbone network and as well the scalability of the system has been increased by a significant ratio. The experimental results based on benchmarks, have shown that the page fault rate has decreased by 25% compared to CVM and 15% compared to [27] methodology.

In the future, we might generalize our method by adding up the cache to the core switch commercially which will be built into it. So, the DVM technique accordingly will be applied and the throughput will be increased. Moreover, we intend to formalize the time complexity of the proposed algorithm and do more comparisons with our techniques.

References

1. Kai, L.: IVY: A Shared Virtual Memory System for Parallel Computing. In: International Conference on Parallel Processing, vol. 2, pp. 94–101 (1988)
2. Barrera III, J.S.: Odin: A Virtual Memory System for Massively Parallel Processors. Microsoft Research, Microsoft Corporation One Microsoft Way Redmond, WA 98052
3. Clancey, P.M., Francioni, J.M.: Distribution of Pages in a Distributed Virtual Memory. In: International Conference on Parallel Processing, pp. 258–265 (1990)
4. Abaza, M.: Distributed Virtual Memory Systems' Ph.D Thesis, The University of Wisconsin-Milwaukee (July 1992)
5. Malkawi, M., Knox, D., Abaza, M.: Dynamic Page Distribution in Distributed Virtual Memory Systems. In: Proceedings of the Forth ISSM International Conf. on Parallel and Distributed Computing and Systems, pp. 87–91 (1991)
6. Fellah, A., Abaza, M.: On page blocks in distributed virtual memory systems. In: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 605–607 (1999)
7. Geva, M., Wiseman, Y.: Distributed Shared Memory Integration. In: IEEE International Conference on Information Reuse and Integration (IRI), August 13-15, pp. 146–151 (2007)
8. Qanzu'a, G.E.L.: Practical Enhancements of Distributed Virtual Memory. M.S Thesis, Jordan University of Science and Technology (March 1996)
9. Abaza, M., Fellah, A.: Distributed virtual memory in the CSMA/CD environment. In: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), August 20-22, vol. 2, pp. 778–781 (1997)
10. Fellah, A.: On virtual page-based and object-based memory managements in distributed environments. In: IEEE Pacific Rim Conference on Communications, Computers and signal Processing (PACRIM), vol. 1, pp. 311–314 (2001)
11. Fellah, A., Abaza, M.: On page blocks in distributed virtual memory systems. In: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 605–607 (1999)