Design of Virtual Memory Compression System for the Embedded System

Seung-Ju Jang
Department of Computer Engineering, Dong-eui University, Korea
sjjang@deu.ac.kr

ABSTRACT

The embedded system has less fast CPU and lower memory than PC(Personal Computer) or Workstation system. Therefore, the embedded operating system is designed to efficiently use the limited resource in the system. Virtual memory management of the embedded Linux have a low efficiency when page fault is occurred to get a data from I/O device. Because a data is moving from the swap device to the main memory. This paper suggests the virtual memory compression algorithm for improving in virtual memory management and capacity of space. This paper suggests the CAMD compression algorithm that achieves significant improvement for the embedded system.

Keywords

Embedded System, Virtual Memory System, Virtual Memory Compression Algorithm

1. Introduction

The virtual memory concept is a successful mechanism that is an essential part of memory subsystem in the kernel. The virtual memory mechanism overcomes insufficient physical memory for all processes. The virtual memory mechanism divides the physical memory into several pages. When the physical memory is fully used by all processes, some pages of a process should be swapped out to the swap space. The embedded Linux has also this mechanism.

If certain computer does not have enough memory, residing pages of data and program should be get out of memory. Moreover it occurs frequently page fault for a data of a process. It will down the computer performance entirely. The page fault of a kernel makes an IO operations and process sleep. The sleeping process loses ownership of a CPU. At this time, if the system has no free memory space, virtual memory handler should pick some pages and get out those to swap space. The page replacement algorithm is the LRU(Least Recently Used), the LFU(Least Frequently Used) and the FIFO(First In First Out) [14,15].

Whenever those pages are being needed, those pages will be swapped into the memory at any time. Swapping pages will make IO operation that will down the system performance. If we have a solution of reducing of swapping pages, we can apply this algorithm to an Operating System. However, there is not properly solution to reduce frequency of swap. Therefore, this paper proposes new algorithm that is the compressed algorithm to reduce frequency of swap for a page. The suggesting algorithm will give fast performance of the system [6].

The proposing algorithm is called the CAMD(Compression Algorithm for in-Memory Data) algorithm which is based on the dictionary of a LZ style. The dictionary size is sixteen entries.. The CAMD algorithm compresses 32 bits data unit. The basic size of a compressed data is 32 bits. This procedure will repeat until end of the page that is 4KB in Linux O.S. First, The dictionary is being initialized. Next, the 32 bits input data is categorized. It is categorized three types that are 0x0, 0xffffffff, or others. The data size is small, because the memory data size is small. The data of memory almost is composed of 0 or 1. The CAMD algorithm is highly available for this characteristic memory data. Normal types of data that is besides 0 or 1 type of data is compressed by using the dictionary.

The CAMD algorithm will make effect of large memory, as well. Therefore, the CAMD algorithm reduces frequency of swap of pages. In addition the mobile devices does not have enough memory, the CAMD algorithm give effect of large memory. The CAMD algorithm can enhance memory space.

Chapter 2 shows the related studies. Chapter 3 is the system design of the CDMA algorithm. Chapter 4 explains implementation of the CDMA algorithm. Chapter 5 gives the experimental results. Finally chapter 6 concludes this paper.
2. Related Research

The compression system of the virtual memory system was suggested by Fred Douglos, 1993. Fred Douglos experimented the virtual memory system on the DECStation 5000/200 by using Sprite OS [4]. His experiment was focused on swap out page. The swap out page was compressed into compression area in memory. The swap page does not need to be swapped out. That page just moves to the compression area in memory. It eliminates IO operation. He used the LZRW1 compression algorithm [10]. He proved that the compressed cache is dependent on application program and compression rate is important. He proved that the compressed cache is more speedy two or three times than non-compressed cache system. But his research has a restriction that it is applied to specific application that has a special mechanism of swap. His result cannot satisfy reasonable performance to all application programs.

Doug Banks and Stemm are proposed the compressed memory system of the portable device(embedded system). This system cannot be satisfying reasonable performance for all application programs. However, the portable system has a little function, there is no need of large capacity in the portable. Key point of his research is that reducing the program and data sizes. The secondary devices are connected with network. His research is focused on the communication speed. Therefore, it is not a general solution. It is just experimental algorithm [1]. We cannot use this algorithm to the Workstation and PC. He uses the specific O.S that is NACHOS 3.2. The NACHOS 3.2 has not common virtual memory structure [1].

3. Design of the Compressed Virtual Memory System

3.1 Architecture of the Compressed Virtual Memory System

Some processes must wait long time caused of IO operation by dirty page on demand paging or swapping mechanism. At this time due to this I/O operation, every system has performance degradation. The compression mechanism of the virtual memory can eliminate IO operation due to compressed cache memory. The swapping page does not need to swap to the secondary device. It is just moved to the compressed cache memory area. Therefore, it reduces a number of IO operations [3].

![Figure 1. Architecture of Compression of the Virtual Memory](image)

Figure 1 show the architecture of compression virtual memory system. The area of main memory is divided into two parts which are compressed cache area and non-compressed cache area. The swapping out pages are moved into compressed cache memory. On the contrary swapping in pages are moved into non-compressed cache memory. When the compressed cache memory has no space, the swap out pages must move to the secondary devices. At this time the swap out pages is decompressed as normal page and moved into the secondary space [2].

The key of the compression algorithm is the performance that should be faster than read and write operation of swap device. Earlier version of the most of compression algorithm is doing compression or decompression as character unit. The compression rate is also not pretty well. Therefore, it is not applicable to the virtual memory compression algorithm. As time passing, the compression algorithm is progressive in compression rate and performance [9, 13]. The proposing algorithm, the CDMA, is solved these problems.

3.2 The CAMD Algorithm

The compression algorithm should be considered the factor, which is compression rate, performance, and page size. The page size of the embedded system is smaller than PC and Workstation [15]. The Linux Operating System has a 8KB page size for Alpha AXP processor. And the Linux Operating System has a 4KB page size for Intel X86 processor. While the page size of the embedded system has smaller page size than that of the PC. The page size of the embedded system is 512B or 1KB. The compression algorithm should be efficient to the small size of data for the embedded system. In addition, the portable device has a small memory space. The PDA system shares the RAM with the secondary memory. Therefore, the RAM space has a swap space, as well. If the swap space can be compressed, it gives effect of large memory space in the PDA system.
The compression algorithm is divided into two sorts that are no-loss data compression and loss data compression. The algorithm of the no-loss data compression is RLC(Running Length Coding), VLC(Variable Length Coding), Huffman coding, DBC(Dictionary Based Coding), LZ(Lempel-Ziv)-base [12]. The proposing CAMD algorithm of this paper is based on LZW compression algorithm. The LZ algorithm makes a character as a code that is saved into the index table. A duplicating character is changed into the code. Therefore, the duplicating character has a small code value. That mechanism can reduce the code size.

The proposing CDMA algorithm is based on the dictionary. The sixteen buffers are used in the CDMA algorithm. Fig. 2 shows the flow chart of the CAMD algorithm. The input data is classified 0x0, 0xffffffff, or other case. The 0x0 input value has 0 compressed value by masking 0x0 and 0x00000001. The 0xffffffff input value has 1 compressed value by masking 0x00000001 and 0xffffffff. The result data of the compression is 0x00 or 0x01 that is saved into “Compress_Buffer”.

\[
\text{Compressed Buffer} = (32 \text{ bits input value} \& 0x00000001)
\]

**Figure 2. Flow Chart of the Compression Algorithm**

The CAMD algorithm is working like Figure 2. The dictionary is important in the virtual memory compression algorithm. The CAMD algorithm is used 16 buffers for the dictionary entry to compare a data. The recent used data is maintained as queue. The recent used data is put into the first position of the queue of the 16 buffers. If the queue has no empty entry, the CAMD algorithm removes the last entry of the data in the queue. If the entry has an empty entry, the new data will be put in this entry. This mechanism is similar to the LRU(Least Recently Used) page replacement. This mechanism will hit frequently used data in the queue entry. It makes high compression rate of the CAMD algorithm. But this mechanism makes down the performance for the compression due to rearrange the queue entry.

In order to remove a traditional compressed algorithm, the CAMD algorithm is considered the two points that are the compression rate and performance. Therefore, the CAMD algorithm is used the hash function to calculate the index value of the input value. The form of the compression data of the CAMD algorithm is the following. The compressed data is composed of the control bits and compressed data. The input value is non-loss data. The output value is divided into two parts that are control bits and compressed data as follows. If the input value is not in the dictionary, two control bits (11) and 32 data bits are combined as 34 bits output data. Next the input value is enrolled into the dictionary table. The index value of the dictionary is determined as the four bits of the input value. The output data is divided into two parts that are two control bits and 32 data bits as shown in the following table.

| Compressed_Buffer = 0x3 |
| Compressed_Buffer = 32 bits input data |

The CAMD algorithm is checking that the input value is enrolled at the dictionary table. If the input value is already enrolled at the dictionary table, the output code is six bits that are composed of two control bits(0x2) and four index bits. The output form is the following table that is control two bits(0x20) and index value of the dictionary. The two values of the control bits and dictionary index are ORing.

\[
\text{Compressed Buffer} = (0x20 \mid \text{index value of the dictionary})
\]
Table 1. Control Bits for the CAMD Algorithm

<table>
<thead>
<tr>
<th>Control bits</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>Output is 0</td>
<td>Output is 1</td>
<td>Index value</td>
<td>Input data</td>
</tr>
<tr>
<td>To be read data</td>
<td>nothing</td>
<td>nothing</td>
<td>4</td>
<td>32</td>
</tr>
</tbody>
</table>

The CAMD algorithm makes high compression rate and good performance in the virtual memory system. Therefore, the CAMD algorithm has a high compression rate and good performance by reducing output data bit.

4. Implementation of the Compression Memory System

This section shows how implements the compression system of the virtual memory. The compression algorithm of the virtual memory is implemented into the memory module of the kernel. The old page should be swapped out to the secondary memory, if there is no empty page of the memory. The selecting procedure of the swapping page is determined at the try_to_swap_out() function [8]. The selecting page should be moved to the swap cache or swap space. And next, shrink_cache() function moves the selecting page to the swap space [7].

Figure 3 shows the entire procedure of the swap out. The incoming page of the swap cache is moved to the swap device by calling the writepage() at the shrink_cache(). The writepage() is a pointer data structure of function, This function pointer is an address swap_writepage() function. The swap_writepage() calls rw_swap_page() function. At this point, the swap out page is compressed by the compress_page() function. The compressed page is moved to the swap cache memory. If there is no space of the swap cache, the page replacement algorithm selects the removing page to the secondary device by the rw_swap_page() function. The removing page is decompressed and moved to the secondary device.

The swap in procedure is simple. When page fault is occurred at certain page, the CPU seek the indicated page at the swap cache first. And next if it cannot find the page at the swap cache, it seeks the page at the compressed cache. Finally when it cannot find the page at the compressed cache, it seeks the page at the swap device as Figure 4.

5. The Experiment and Performance

The experiment is doing for two cases to compare the case of compressed page and normal page. The CAMD algorithm is implemented into the Linux kernel of the virtual memory. To do experiment of the embedded system, first the CAMD algorithm is implemented into the Linux kernel.

The “Fillmem” test program which was programmed by ourself was used to test of allocation and release physical memory. This program makes forced swapping of a certain page. The experiment was done by changing the size of the physical memory. The performance and usage of the CPU was evaluated by this experiment. The experiment was estimated the real time for the compression virtual memory system. The physical memory size is changed 10Mbytes to 450 Mbytes.

The size of the compressed memory space was set up 8Mbyte to 128Mbytes. Figure 5 shows that the size of the compressed memory is 8Mbytes. By this experiment the performance of the compressed memory system is worse than non-compressed memory system. The X-axis of Figure 4 indicates memory size. The performance of the compressed memory system is the same for both systems until the memory size is 360Mbytes. Entire memory sizes are 385Mbytes. Certain daemon process consumes 25Mbytes memory space. The remaining memory space is 360Mbytes. Therefore, the threshold point of the memory is 360Mbytes. If a task needs more than 360Mbytes memory space, the compressed memory system has a performance benefit. The reason why the performance degradation of the compressed memory system is the overhead of the swapping page if there is no compressed cache space in memory. If the memory space is growing up more and more, the performance of the “Fillmem” program is fast.

Figure 4. The Performance for 8MBytes Compressed Memory

When the total memory space is 390 Mbytes, 410 Mbytes, and 430 Mbytes, the performance is fast suddenly. Because growing memory space makes
reducing the swap out pages. Figure 5 and Figure 6 show the memory space was changed 16Mbytes to 32Mbytes.

**Figure 5.** The Performance for 16Mbytes Compressed Memory

**Figure 6.** The Performance for 32Mbytes Compressed Memory

Figure 5 and Figure 6 show the memory space was changed 16Mbytes to 32Mbytes.

**Figure 7.** The Performance for 64Mbytes Compressed Memory

Figure 7 shows the performance result of the experiment for 64 Mbytes memory space. Figure 7 shows that compressed memory system is better than non-compressed memory system. The optimized compressed memory size is dependent on the physical memory and requiring memory for some processes that include daemon processes. If the compressed cache is flexibly changed under certain circumstances, it will make the best performance in the virtual memory system.

The 1/6 compressed cache memory makes the best performance of virtual memory in the experiment system. Figure 8 shows that the elapsed time is over 8 seconds in the case of 128Mbyte compressed cache memory size. Figure 7 shows that the elapsed time is not over 8 seconds. In conclusion the virtual memory performance of the 128Mbytes compressed cache memory is more fast than 64Mbyte compressed cache memory.

**Figure 8.** Elapsed Time for the 128Mbytes Compressed Cache Memory

Figure 9 shows the result of the performance among the LZ algorithm, LZO compression algorithm, and the
CAMD algorithm. The proposing CAMD algorithm is better than the LZO algorithm.

![Performance Result for the CAMD Algorithm and the LZO Algorithm](image)

Figure 9. Performance Result for the CAMD algorithm and the LZO Algorithm

The performance of the virtual memory system is dependent on the compressed cache memory size. The performance for 8, 16, and 32 Mbytes compressed cache memory is not good against the non-compressed cache memory. Because the compressed cache memory space is not enough. The more compressed space in the virtual memory system, the better performance. If having much occurrence of swap to the secondary, the performance of the compressed memory system is better. The CAMD algorithm eliminates the frequency of the access of the secondary device. It gives high performance of virtual memory system.

6. Conclusion

This paper proposes the compressed algorithm, the CAMD algorithm, which is reduced frequency of the swap out pages and gives effect of the large size of memory cache. The CAMD algorithm makes high performance for the embedded system. The memory is divided into two parts, which are the compressed area and the uncompressed area. The page that must move to the swap device can be saved into the compressed area. This idea reduces the access frequency of the secondary device.

The CAMD algorithm compresses the page data. It is categorized three types that are 0x0, 0xffffffff, or others. The page data size becomes the small length, because the CAMD algorithm produces small code length. The data of page memory is almost composed of 0 or 1. The CAMD algorithm is highly available this characteristic memory data. Normal types of data that is besides 0 or 1 type of data is compressed by using the index of the dictionary.

We did performance test for the compressed virtual memory and the uncompressed virtual memory. By this test, the more compressed cache space, the better performance in the virtual memory system.

In the future, the pretty good compressed algorithm of the virtual memory page data makes good performance of the virtual memory system. The CAMD algorithm should be upgraded in the future in terms of performance and compression rate. Moreover, the compressed cache must be changed flexibly as needed.

References

[7] Sumit Roy, Raj Kumar, Milos Prvulovic, "Improving System Performance with Compressed Memory", 2001 IEEE.

[16] Abraham Silberschatz, Greg Gane, Petter Baer Galvin
"Operating System Concept", 5th Edition