SIMULATION OF LRU PAGE REPLACEMENT ALGORITHM FOR IMPROVING PERFORMANCE OF SYSTEM

Mukesh Kumari¹, Ashish Kumar², Neha³

¹ Asst.Proff. in RPSGOI Mohindergarh (Haryana),
² PG (M.Tech) student PDMCET Bahadurgarh,
³ Asst. Prof. In East Point Engineering College Bangalore

msheoran132@gmail.com
ashish4ash@gmail.com
neha2204@gmail.com

Abstract—Now days, page replacement policy become the heart of caching system. Which decides which page to replace in a cache by a new one? Different page replacement algorithms have dramatically different effects on the system performance. A virtual memory system needs efficient page replacement algorithms to decide which pages to evict from memory in case of a page fault. Over the years many algorithms have been proposed for page replacement. Each algorithm attempts to minimize the page fault rate while incurring minimum overhead. In this paper we discuss at the traditional algorithms such as LRU. Least Recently Used (LRU) page replacement algorithm is most efficient algorithm compare to others algorithms.

Keywords—Page Replacement, LRU, Demand paging, locality of reference, Reference bit.

I. INTRODUCTION

Computer system physical memory sizes have increased consistently over the years, yet counter to popular conception, optimizing the allocation and management of memory continues to be important. Numerous scientific and engineering applications exist that can exhaust even large physical memory. Moreover, while physical memory is generally considered to be inexpensive, it continues to be one of the dominant factors in the cost of today’s medium to large scale computer systems, and also a major factor in energy consumption. To use memory effectively, accurate information about the memory access pattern of applications is needed. Hence we use a two – level memory hierarchy consisting of a faster but costlier main memory and a slower but cheaper secondary memory. Virtual memory systems use this hierarchy to bring parts of a Program into main memory from the secondary memory in terms of units called as pages. Pages is brought into main memory only when the executing process demands them; this is known as demand paging. A page fault is said to occur when a requested page is not in main memory and needs to be brought from secondary memory. In such a case an existing page needs to be discarded. The selection of such a page is performed by page replacement algorithms which try to minimize the page fault rate at the least overhead.

1.1 PAGE REPLACEMENT PROCESS:

Once the main memory fills up a page must be swapped out to make room for any pages to be swapped in. This is known as page replacement process.

The goal of the page-replacement algorithm is to minimize the page-fault rate. Different algorithms may be compared by computing the number of page faults on a particular reference string. Given the overhead of a page fault, small improvements in the page replacement algorithm will greatly improve the performance of the entire system. Generally, increasing the number of frames reduces the number of page faults.

The page-fault service routine will:

1. Find the location of the desired page on disk.
2. Either find a free frame or create a free frame by selecting a victim frame, writing it to disk, and updating the page and frame tables.
3. Read the desired page into the free frame.
4. Restart the process.

```
frame
```
2. LEAST RECENTLY USED (LRU) ALGORITHM

The page references that occurred in the recent past are good indicators of what page references will occur in future. That is if a page has just been referenced it is likely that it will be referenced again. This gives rise to the least recently used (LRU) algorithm.

The LRU keeps track of how recently a page has been used and keeping the pages that reference is most distant past. The LRU policy is based on the principle of locality which states that program and data references within a process tend to cluster. The Least Recently Used replacement policy selects that page for replacement which has not been referenced for the longest time. For a long time, LRU was considered to be the most optimum online policy. The problem with this approach is the difficulty in implementation. One approach would be to tag each page with the time of its last reference; this would have to be done at each memory reference, both instruction and data. LRU policy does nearly as well as an optimal policy, but it is difficult to implement and imposes significant overhead.

### LRU Replacement

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>e</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>Page Frames</td>
<td>0</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>c</td>
<td>c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. THE RULES OF LRU POLICY

The following are the rules of this page replacement policy:

1. The page that has been brought into the memory is assigned an age bit and a reference bit.
2. The age bit will keep a record of the time the page has spent in memory.
3. The value of the age bit can be any value between the number of frames and twice the number of frames in memory.
4. For every page that is admitted to the main memory the age bit corresponding to the pages in memory is decremented by some constant number.
5. Whenever a page hit occurs, its reference bit will be set and the age will be reset to the initial age value (maximum).
6. When the age of a particular page becomes 0, then its reference bit, if set, is reset. But it will continue to stay in memory and the value of age is decremented in every iteration as above.
7. When the memory is not free and there is a need for page replacement all the pages resident in memory will be checked in regards with their age bit and reference bit.
8. The obvious choice for replacement will be the page whose reference bit is not set and is the oldest of all pages.
9. The pages with their reference bit set to 1 will not be chosen for replacement as they are given a second chance.
10. If all the pages have their reference bits set then the oldest page, i.e. the page with the minimum age value will be replaced.

4. SIMULATION OF LRU ALGORITHM

Consider for 16 page frames and as shown in right side reference fields, we want to simulate LRU Page Replacement Algorithm.

Steps:
1. Select Algorithm ‘LRU’ from combo box.
2. Set Page frames ‘16’.
3. Fill reference fields to reference list.
4. Press ‘Simulate’ button.

We get page table with page hits and page faults information.

![Figure 4](image_url) is a software implementation of the LRU.
5. CONCLUSION

In this paper, we designed and implemented a LRU page replacement algorithm. The goal of that page-replacement algorithm is to minimize the page-fault rate. Small improvements in the page replacement algorithm will greatly improve the performance of the entire system. Generally, increasing the number of frames reduces the number of page faults. That algorithm also helps to improve the throughput of the system.

6. REFERENCES
