Databasesystems 2
Forum: https://forum-db.informatik.uni-tuebingen.de/c/ss18-db2

Assignment 4 (15.05.2018)

Submission: Tuesday, 29.05.2018, 10:00 AM

1. [10 Points] **Bélády's Anomaly**

   In 1969, the Hungarian computer scientist Lázló Bélády had proven that with increasing size of the buffer pool, there may still be an increase in page faults compared to smaller buffer pools. This phenomenon is known as *Bélády's Anomaly*.

   (a) You are given the following replacement strategies:

   - FIFO (**F**irst **I**n **F**irst **O**ut): The first page loaded into the buffer is also to be replaced first.
   - LRU: Replaces the **L**east **R**ecently **U**sed page first.

   For buffer sizes of 3 and 4 pages, compute the contents of a FIFO- and a LRU-buffer after you have accessed the following pages in order (pages are loaded and then immediately released, i.e., the `ref_count()` of all pages in the buffer is 0):

   $$p_1, p_2, p_3, p_4, p_1, p_2, p_5, p_1, p_2, p_3, p_4, p_5$$

   Describe your observations in context of *Bélády's Anomaly* briefly.

   (b) In general, is it even possible that *Bélády's Anomaly* occurs for an LRU-buffer? Explain briefly.

2. [20 Points] **LRU-k**

   Here, we will examine LRU-k, a variant of the LRU-k[1] replacement strategy.

   Introduction:
   For each page, LRU-k considers the points in time of the last k references. LRU-k uses this information to estimate how *frequent* a page is referenced.

   Given a set $N = \{p_1, p_2, \ldots, p_n\}$ of *disk pages*. These pages are referenced successively. Below, a sequence of page references $r_1, \ldots, r_t, \ldots$ where each r_t denotes some page p, will be referred to as *reference string*.

   Metric:
   Let us first define a metric that determines the page that LRU-k replaces next.

   Definition 1 *Backward k-Distance*
   Assume a reference string $r_1, r_2, \ldots, r_t$. *The* Backward k-Distance $b_t(p,k)$ *is defined as follows:*

   $$b_t(p,k) = \begin{cases} x, & \text{if } r_{t-x} = p \text{ and there exist exactly } k-1 \text{ other} \\ & \text{indices } i,\ t-x < i \leq t, \text{ with } r_i = p. \\ \infty, & \text{if } p \text{ does not occur at least } k \text{ times in reference} \\ & \text{string } r_1, \ldots, r_t \end{cases}$$

   In other words: Starting from position r_t in the reference string, we are traversing the reference string backwards looking for the k-th occurrence of page p. The distance x is the *Backward k-Distance* of the page. The distance is ∞, if p does not occur at least k times.

[1] http://www.cs.cmu.edu/~christos/courses/721-resources/p297-o_neil.pdf

Example:
Assume the reference string in Figure 1a, a buffer size of 3 pages, and the replacement algorithm LRU-2. Pages with gray backgrounds have been referenced already:

Reference String	r_1	r_2	r_3	r_4	r_5
Page	p_1	p_2	p_3	p_1	p_4

(a) Reference String

Page	*Backward* 2-*Distance*
p_1	$b_5(p_1, 2) = 4$
p_2	$b_5(p_2, 2) = \infty$
p_3	$b_5(p_3, 2) = \infty$

(b) *Backward* 2-*Distance*

Figure 1: Example: LRU-2

The next page we will reference is p_4. The buffer holds p_1, p_2, and p_3 already and thus is full—we have to replace a page in the buffer. Calculation of the *Backward* 2-*Distance* determines the distances shown in Figure 1b. Neither p_2 nor p_3 have two occurrences in the reference string $r_1, \ldots, r_5$. Hence, both pages currently have a *Backward* 2-*Distance* of ∞. Page p_1, on the other hand, occurs twice. We are thus looking for distance x such that $r_{5-x} = p_1$ and $k - 1 = 1$ further references to p_1 occur in the range $r_{5-x}, \ldots, r_5$. For p_1, we find $x = 4$, since $r_{5-4} = r_1 = p_1$ and p_1 is referenced once by r_4.

Replacement Strategy:
If the buffer is full, we replace the page with the greatest *Backward Distance*. In case multiple pages have a *Backward Distance* of ∞, we revert to another replacement strategy. In our case here, we revert back to the classic LRU page replacement algorithm.

Your Tasks:

(a) Is it possible to find a parameter k for LRU-k such that LRU-k becomes equivalent to LRU? Explain briefly.

(b) Scenario:
You are given transactions $\mathcal{T}_1$ and $\mathcal{T}_2$ and a buffer with 11 pages. The transactions reference pages from a table $\mathcal{R}$ with 100 pages.
The transactions $\mathcal{T}_1$ and $\mathcal{T}_2$ have the following properties:

Transaction $\mathcal{T}_1$	$\mathcal{T}_1$ uses only part of $\mathcal{R}$. Specifically, it references pages 1 to 10. Therefore, its reference string is $\underbrace{p_1, p_2, \ldots, p_{10}}_{\times 10}$.
Transaction $\mathcal{T}_2$	$\mathcal{T}_2$ references **all** pages in sequential order. Therefore, the reference string is $p_1, \ldots, p_{100}$.

Starting with transaction $\mathcal{T}_1$, the transactions reference pages in table $\mathcal{R}$ alternatingly. The overall reference string r thus reads as follows:

$$r = p_1, p_1, p_2, p_2, p_3, p_3, p_4, p_4, \ldots, p_{10}, p_{10}, p_1, p_{11}, p_2, p_{12}, p_3, p_{13}, \ldots, p_{10}, p_{100}$$

For this scenario, count the number of buffer misses for both, LRU-2 and the classic LRU algorithm. Which advantages (or disadvantages) do both strategies exhibit?