2Q

Ohad Rodeh

Introduction

- This lecture is based on **2Q: a low overhead high performance buffer management replacement algorithm**, Theodore Johnson and Dennis Shasha, Twentieth International Conference on Very Large Databases, 1994.
- 2Q attempts to improve LRU-2 by reducing its CPU overhead from logarithmic to constant

Deficiencies in LRU-2

- Logarithmic computational complexity: priority queue for pages in history
- Tuning parameters:
  - *Correlated Reference Period* : when temporally close accesses occur at a page, they count as a single access
  - *Retained Information Period* : how much history to keep?
- 2Q attempts to create an algorithm that is as good as LRU-2, but has constant-time complexity and is self-tuning

The 2Q algorithm, simplified

- Two lists: A_1 and A_m
- A_1 contains recently accessed pages and is managed in FIFO fashion
- A_m contains pages promoted from A_1 because they were accessed again while they were in A_1. A_m is managed with LRU.
- A limited history of pages evicted from A_1 is maintained in A_{out}

Solving the Correlated Reference Period problem

- We'd like for pages in A_1 that are accessed again during a short period of time to remain in A_1 and not be promoted
- Method: split A_1 into A_{1in} and A_{1out}
- A_{1in} retains pages that were access once recently
- A_{1out} keeps track of pages recently evicted from A_{1in}
- If a page in A_{1in} is accessed again, do nothing
- If a page in A_{1out} is accessed again, read it from disk and promote it to A_m

Keeping history

- How much of a history should we keep?
- This is (still) a tunable parameter
- The authors say that using 50% of the memory slots for history and 25% for A_{1in} is a good choice in practice
- It is possible to, instead of just keeping history, actually keep the pages in memory. This leads to worse performance.

Experimental results

- Zipf distribution with $\alpha = 0.5, 0.86$
- When $\alpha = 0.86$, we get an 80/20 distribution
- When $\alpha = 0.50$, we get an 45/20 distribution
- This distribution is uniform, there is no temporal locality. LFU is ideal here.
- Therefore, set $\|A_{1in}\|$ to 1
- Run simulator with 10^6 references

Zipf with $\alpha = 0.5$

- 2Q and LRU are close
- Both are better than LRU
- Optimal (A_0) is still better

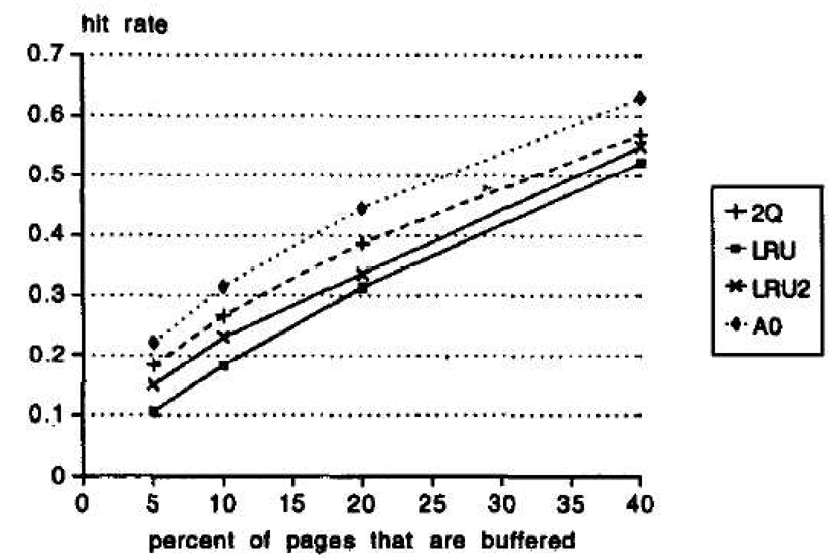


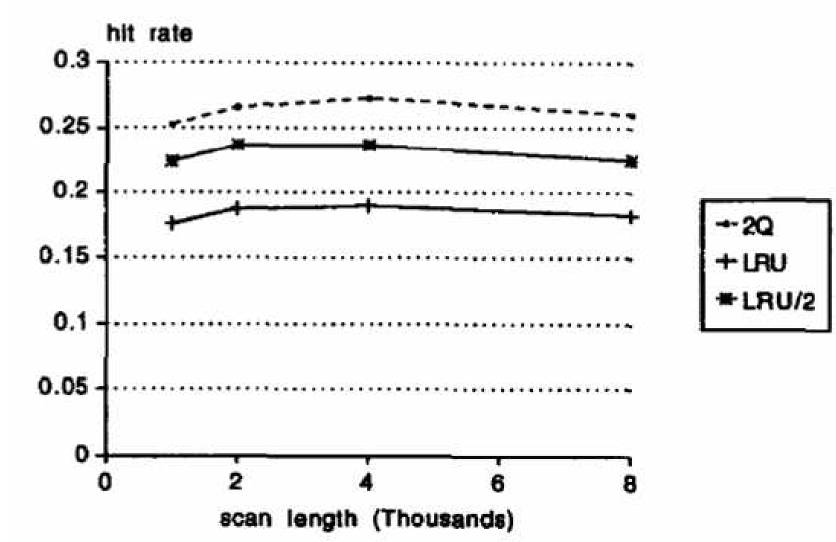
Figure 1: Zipf input comparison with parameter 0.5

Zipf sequential scan

- Added sequential scans to the zipf synthetic trace
- In a sequential scan a page is referenced just once
- A third of the references are from sequential scans

Zipf with $\alpha = 0.5$

- 2Q and LRU-2 are better than LRU
- Hit rates drop compared with no-scans
- The hit rate of 2Q is about 70% compared with no-scan
- The hit rate of LRU-2 is about 60% compared with no-scan



Two pool experiment

- Identical experiment to LRU-2
- LRU-2 and 2Q are very close

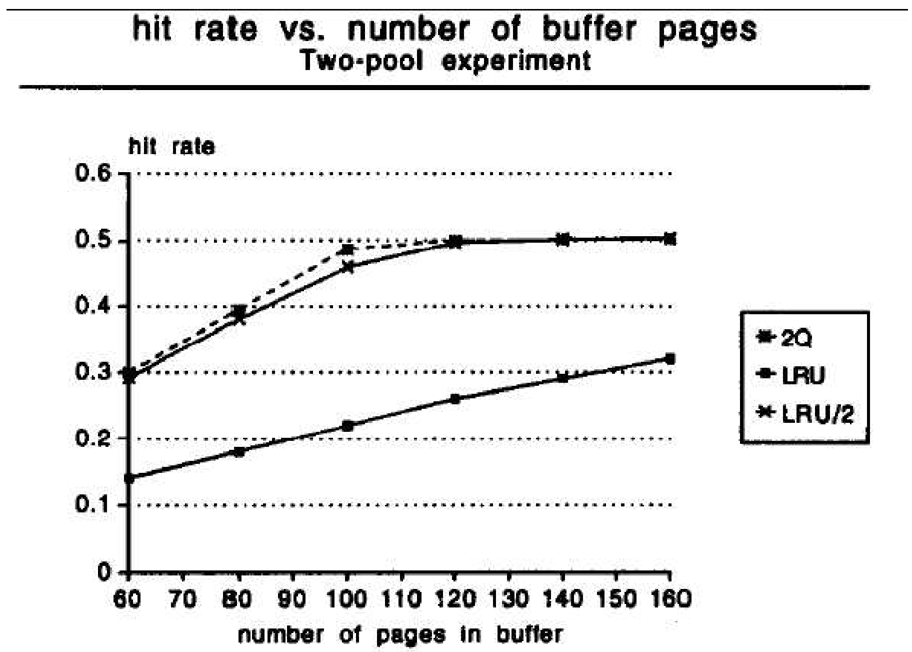


Figure 5: Behavior of Algorithms on Index Accesses to Data Pages

OLTP

- Same experiment as in the LRU-2 paper
- 914145 references to 186880 distinct pages
- LRU-2 and 2Q are close
- They out perform Clock and LRU

Number of page slots	LRU/2	LRU	Gclock	2nd Chance	2Q Kin=30%	2Q Kin=20%
100	.086	.083	.083	.083	.096	.090
200	.164	.144	.144	.141	.196	.181
500	.284	.234	.236	.223	.334	.329
1000	.384	.328	.327	.318	.405	.405
2000	.454	.425	.425	.418	.465	.464
5000	.544	.537	.538	.532	.556	.557
10000	.616	.607	.607	.602	.626	.624
20000	.678	.671	.671	.665	.681	.680

Table 3: OLTP trace hit rate comparison of LRU, LRU/2, and 2Q.

Conclusions

- 2Q is an improvement on LRU-2
- However, it still has a manual parameter: the size of A_{1in}