Proposal for Implementing CAR/CART for Eviction Policy

Problem: Currently in Quickstep our block eviction policy implementations all face the issue of needing to coordinate mutexes in order to properly function. This includes the k-LRU implementation that is the most commonly used eviction policy that we are currently utilizing. This usage of mutexes and the need for locks in general leads to contention and is a potential bottleneck when memory is low or a given block size is low. Experimental results confirm that our current eviction policy implementation leads to reduced performance for some TPC-H queries in these cases.

Proposed Solution: The solution that I propose is to adapt and implement a modification of a traditional clock algorithm for memory management, called Clock with Adaptive Replacement (CAR). As created by Bansal and Modha in their 2004 paper [1], these algorithms build on the traditional clock algorithm by implementing features from the Adaptive Replacement Cache (ARC) algorithm to improve it. CAR with Temporal filtering (CART) is an additional adaptation of the CAR algorithm that the authors propose is better suited for database systems. As CART is largely CAR with additional features built in, implementing CART after a CAR implementation should be doable with little hassle, so the two algorithms could be compared in the database world that we work with in Quickstep.

Alternate Solution: The alternative solution to the mutex problem would be attempting to modify and rework the k-LRU algorithm to involve less lock contention.

Benefits Compared to Alternative: Implementing an entirely new algorithm comes with its advantages. As described in the paper introducing these algorithms, the CAR method solves multiple problems that LRU cannot, outside of just lock contention. Namely, we are able to keep frequency in mind just as much as recency. While k-LRU as an extension of LRU does keep this in mind somewhat, the CAR method is more powerful in this regard. As well, LRU can be polluted by a scan of every block in the system, which CAR is resilient to. CART in particular will also allow us to keep time in mind as well as a factor for eviction, which is an added dimension from previous algorithms. Aside from the algorithmic advantages, this is a known method to avoid the problem we face. Attempting to modify k-LRU to have reduced lock contention is a problem with an unknown solution, and some form of lock will always be necessary for the scheme, especially as we move Quickstep to a multiple query environment in the future.

Disadvantages/Difficulties Compared to Alternative: As proposed by the paper, CAR is a page eviction algorithm for normal system memory. Adapting the language and features used to Quickstep's needs might prove to be a challenge. The machinery needed for k-LRU is already built into the system and is known to work, but new processes and data structures will be needed for CAR. As well, there is a currently-unknown amount of overhead to the CAR algorithm. The paper proposes it as on-par with ARC for performance and above LRU, but the Quickstep-specific machinery needed might change this. There is also a potential concern with patents, as IBM-sponsored algorithms can carry restrictive ones, but I was unable to find anything about this one way or the other in my research.

References

1. Bansal, Sorav, and Dharmendra S. Modha. "CAR: Clock with Adaptive Replacement." In *FAST*, vol. 4, pp. 187-200. 2004.