When 'more and more' does not help: Sensible Partitioning of Cache

Hamza Bin Sohail
{hsohail}@purdue.edu
Purdue University, West Lafayette IN

1.Introduction

Buffer caching is the caching of disk blocks in memory to avoid costly disk accesses by exploiting spatial and temporal locality. Most buffer cache algorithms [1, 2] are designed to improve the system performance from a holistic perspective thus treating all disk blocks in the same manner. The algorithms are oblivious to the application identity which requested the block. Such a design decision is acceptable in an environment where single jobs are expected to run most of the times but for environments in which a large number of disk-bound processes are expected to be in execution, this design decision should be revisited. Most of the buffer cache algorithms can potentially give a higher hit-ratio, lesser number of disk requests and therefore further reduce execution time of individual jobs if a 'cache occupation limit' is placed on certain processes which do not show improvement in terms of hit-ratio and/or I/O rate even if more cache blocks are allocated to them. Existing buffer cache algorithms can be augmented by an optimization of partitioning the cache and judiciously allocating cache blocks to each application. This optimization can improve system performance significantly under cases where there is a mix of *cache benefiting* and *non-benefiting* applications in concurrent execution. The past and current trend of instantaneous hit-ratio and disk access rate of an application can be used to determine if it is worth allocating more cache blocks to it. This approach can help the operating system identify 'Cache Hogs' - processes that exhibit unfair behavior and affect the I/O performance of other processes – and limit their adverse affects on other processes in the system.

2.Determining the appropriate 'Cache Occupancy' for Applications

Detection schemes of current buffer algorithms focus on identifying sequential and random access patterns in the disk access stream but they do not identify ill-behaving processes which do not exhibit performance improvement in terms of hit-ratio or I/O rate even when more and more blocks are allocated to them. If such a runtime identification mechanism is in place, it helps in answering the question "How much cache does the application *deserve*?". With this information, the appropriate cache occupancy for these applications can be determined. Cache occupancy refers to the maximum cache block quota for that application. It is important to mention that the partitioning of cache does not guarantee performance improvement in all scenarios. For example, two concurrently running identical disk-bound applications which show an identical increase in hit-ratio and decrease in I/O rate as cache size increases may give a lesser hit-ratio and increased I/O rate if each application has half of the cache blocks available to it instead of the entire cache. The detection and partitioning algorithm used to determine cache occupancy of each application can be incorporated in any buffer cache algorithm. The detection scheme consistently monitors the instantaneous hit-ratios and I/O rate of each application/process and is based on the assertion that increased hit-ratio and reduced I/O rate crudely implies a reduced execution time.

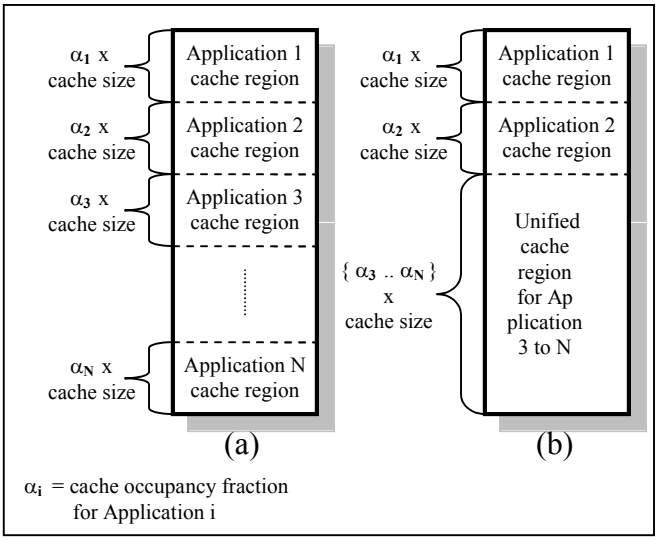

Figure 1: (a) shows a scenario of cache allocation to N disk-bound applications in which all disk-bound applications are bound to a certain fraction of the cache (b) shows a scenario of cache allocation of N disk-bound processes in which application 1 and 2 are bound to α_1 and α_2 fractions of the cache while the other applications get blocks allocated from a unified cache region of $(1 - \alpha_1 - \alpha_2)$ fraction of the entire cache

The scheme categorizes the processes as good, bad and ugly. Good processes show considerable improvement in terms of hit-ratio and I/O rate as more cache blocks are available to that process. Bad processes show an imperceptible improvement in terms of hit-ratio and I/O rate when more blocks are allocated to them while ugly processes are the 'cache hogs'. The cache hogs have a very low hit-ratio and high I/O rate. These applications do not show any improvement whatsoever as more and more cache blocks are allocated to them primarily because of a random access pattern and/or a very large working set. Based on the categorization, the cache occupancy α for that application is determined. The benefit of such partitioning can be manifold as the affects of cache hogging applications are restrained by the cache occupancy metric which does not expose the entire cache to such hogging applications and more cache-friendly applications get a greater share of the cache.

3.References

[1] A. R. Butt, C. Gniady, and Y. C. Hu, "The Performance Impact of Kernel Prefetching on Buffer Cache Replacement Algorithms", *Proc. of SIGMETRICS '05*, June, 2005.

[2] AMP: Program Context Specific Buffer Caching, Feng Zhou, Rob von Behren, and Eric Brewer, Proceedings of the *Usenix Technical Conference 2005*, Anaheim, CA, April 2005.