FlexiCache: A Flexible Interface for Customizing Linux File System Buffer Cache Replacement Policies

Pavan Konanki and Ali R. Butt
Virginia Tech., Blacksburg, VA 24061
{kpavan, butta}@cs.vt.edu

Recently, there has been a growing trend towards designing advanced file system buffer cache replacement algorithms such as ARC [1], PCC [2], and LIRS [3]. These algorithms perform much better than standard caching algorithms available in modern operating system kernels. Given the availability of such advanced caching algorithms and the potential for designing better algorithms, it is highly desirable to replace the standard caching policy. Since the performance of many recently developed replacement algorithms is dependent on the application access patterns, a single caching policy cannot be effective for the wide range of applications that are run on modern operating systems. Furthermore, delegating the responsibility of cache management to the application developers immensely complicates the development process for typical applications. It would be useful to have a repertoire of caching policies in the kernel and to allow the operating system to choose the one that best suits the needs of the currently running applications. In addition, as pointed out in [4], the performance of caching algorithms must be studied in the context of all the kernel storage subsystems. Unfortunately, the intertwined unified page and buffer cache in the Linux kernel makes incorporation of new caching policies an excruciating task, and requires in-depth scrutiny of the code as it may threaten the stability of the kernel. To address these issues, we propose FlexiCache, a flexible interface to the existing page and buffer cache management mechanism. FlexiCache allows easy modification of the replacement policy and provides a simple and powerful means for implementing new caching policies.

The key idea of FlexiCache is to utilize a modular design that allows modification of the cache replacement policy by exposing a well-defined interface, while hiding metadata maintenance and the mechanics of cache management such as allocation, fetch, and eviction of blocks. FlexiCache comprises a set of functions that the cache management subsystem would utilize when, for example, it needs to evict or prefetch a set of blocks from the cache or when the cache management daemon needs to be invoked. A new cache replacement algorithm can be easily integrated with the kernel by simply utilizing the FlexiCache interface.

As an illustration, consider incorporating the ARC [1] policy in the kernel. In the current setup this task is error prone as it requires careful and extensive modification to the kernel code. In contrast, by using FlexiCache we would only need to define the ARC-specific data structures and implement the functions that define the eviction policy. Once the functions declared by the FlexiCache interface are implemented (using a small amount of code), the kernel can invoke them to use the ARC policy. Thus, FlexiCache enhances the modularity and flexibility of the cache management subsystem, and allows us to implement various cache replacement policies as easily replaceable components within the kernel. Furthermore, it would also facilitate research in designing advanced caching algorithms.

Our implementation of FlexiCache has three phases. In the first phase, we analyzed the Linux 2.6 kernel code and identified the portions of code that require modification to incorporate FlexiCache in the kernel. In the second phase, we are designing a set of functions through which the cache management subsystem can utilize different caching policies. These functions constitute the FlexiCache interface. Currently, FlexiCache provides an interface through which a caching policy can specify blocks to fetch into or evict from the cache. In addition, FlexiCache supports specification of a set of cache cleanup functions that are invoked periodically. We believe that designing the interface to support a multitude of replacement policies, while still keeping the mechanics of the cache hidden, is a challenging research problem. In the third and final phase, we plan to study how FlexiCache would affect the performance of the kernel.

References

[1] N. Megiddo and D. S. Modha. ARC: A Self-tuning, Low Overhead Replacement Cache. In *Proc. USENIX FAST*, March 2003.

[2] C. Gniady, A. R. Butt, and Y. C. Hu. Program-counter-based pattern classification in buffer caching. In *Proc. USENIX OSDI*, December 2004.

[3] S. Jiang and X. Zhang. LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance. In *Proc. ACM SIGMETRICS*, June 2002.

[4] A. R. Butt, C. Gniady, and Y. C. Hu. The performance impact of kernel prefetching on buffer cache replacement algorithms. In *Proc. ACM SIGMETRICS*, June 2005.