Simulation Modelling Practice and Theory 20 (2012) 33–45

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Deconstructing on-board disk cache by using block-level real traces

Yuhui Deng *, Jipeng Zhou, Xiaohua Meng

Department of Computer Science, Jinan University, Guangzhou 510632, PR China

ARTICLE INFO

Article history:
Received 23 December 2010
Received in revised form 11 August 2011
Accepted 12 August 2011
Available online 29 September 2011

Keywords:
On-board cache
Disk drive
Prefetch
Locality

ABSTRACT

On-board disk cache is an effective approach to improve disk performance by reducing the number of physical accesses to the magnetic media. Disk drive manufacturers are increasing the on-board disk cache size to match the capacity growth of the backend magnetic media. Some disk drives nowadays have a cache of 32 MB. Modern computer systems use large amounts of memory to improve performance, any data brought into host memory will be re-accessed there, not in the on-board disk cache. This feature has a significant impact on the behavior of disk cache. This is because computer systems are complex systems consisting of various components. The components are correlated with each other. Therefore, a specific component cannot be isolated from the overall system when we analyze its performance behavior. This paper employs four block-level real traces to explore the performance behavior of the on-board disk cache by considering the impacts of the cache hierarchy contained in computer systems. The analysis gives three major implications: (1) I/O stream at block-level contains negligible temporal locality. Therefore, read/write cache can only achieve marginal benefits. (2) Static write cache does not achieve performance gains since the write stream does not have too much interference with the read stream. Therefore, it is better to leave the on-board disk cache shared by both the write and read streams. (3) Read cache dominates the contribution to the hit ratio besides prefetch. Thus, it is better to focus on improving the read performance rather than write performance of disk cache.

1. Introduction

The storage hierarchy in current computer architectures is designed to take advantage of data access locality to improve overall performance. Each level of the hierarchy has higher speed, lower latency, and smaller size than lower levels [1,2]. The performance gap between processor and memory has been alleviated by fast cache memories. However, the performance gap of RAM to disk drive has been widened to 6 orders of magnitude in 2000 and will continue to widen by about 50% per year [2]. Therefore, the disk I/O subsystem is repeatedly identified as a major bottleneck to system performance in many computing systems.

To alleviate the impact of the widening gap, many research efforts have been invested in improving the performance of disk drives. One of the most effective approaches is employing an on-board disk cache to reduce the number of disk I/Os. Almost all modern disk drives employ a small amount of on-board cache (SRAM) (usually less than 32 MB) as a staging area. This staging area acts both as a speed-matching buffer and a disk cache. As a speed-matching buffer, it allows the disk firmware to overlap the transfers over the interface and to or from the magnetic media, thus bridging the data transfer speed differences between disk interface and slower-speed magnetic media. As a disk cache, it can speed up accesses to data on

* Corresponding author.
E-mail addresses: tyhdeng@jnu.edu.cn, yuhuid@hotmail.com (Y. Deng).

1569-190X/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2011.08.004

the disk drives. Because accessing data from cache is much faster than from magnetic disk, the disk cache can significantly improve performance by avoiding slow mechanical latency, if the data accesses are satisfied from the disk cache (cache hit). However, SRAM is still very expensive. This results in a very small cache size in comparison to the capacity of the magnetic disk. Therefore, it is very important to take full advantage of the precious disk cache space.

In the past decades, how to effectively leverage the limited space of disk cache has been investigated thoroughly by previous work. Smith [3] considered a number of design parameters for such a disk cache, including cache location, cache size, replace algorithms, block sizes, access time, bandwidth, consistency, error recovery, etc. Each of these parameters was discussed and/or evaluated in terms of trace driven simulations. He reported that the addition of a disk cache to a large computer system can significantly improve performance, and a disk cache (in a large IBM-type system) of less than 8 Mbytes can capture 60–95% of I/O requests. Karedla et al. [4] examined some popular strategies and replacement algorithms for disk cache, as well as the advantages and disadvantages of caching at different levels of the computer system hierarchy. They also evaluated the performance of three cache replacement algorithms: random replacement (RR), least recently used (LRU), and a frequency-based variation of LRU known as segmented LRU (SLRU). Soloviev [5] investigated the performance of multi-segment disk caches for prefetch. He concluded that the disk prefetch offers about the same performance as main memory prefetch if request queues are managed in the disk controllers rather than in the host. Otherwise, the memory prefetch makes disk prefetch obsolete. Therefore, he questioned the benefits of using the disk cache for prefetch. Shriver et al. [6] developed performance models of the disk drive components including queues, caches, and the disk mechanism and a workload transformation technique for composing them. They also built a new analytic model for disk drives that do prefetch and request reordering.

Since the volatile disk write cache could result in a data loss when an error or power failure occurs, non-volatile write cache can be employed to solve this problem. DOME [7] is a new destage algorithm for the non-volatile disk write cache. This algorithm leverages an observation that in the write cache, the data, that are overwritten once, have more probability to be overwritten again than the data that are written only once. Traditionally, in order to ensure the correctness of a write, an additional read is immediately issued after the completion of the write to verify the written content. However, the Read After Write (RAW) degrades the overall performance of a disk drive since it doubles the service time of write operations. In order to reduce the performance impacts, Idle Read After Write (IRAW) is proposed to retain the content of a completed and acknowledged write request in the disk cache and verify the content written on the magnetic disk with the cached content during idle times [8]. By using detailed simulations, Zhu and Hu [9] drew conclusions that larger on-board disk cache does not have much impact on system performance. They argued that 512 KB disk cache is enough to reduce the cost without affecting performance. They also reported that when the number of disk cache segments is bigger than the number of concurrent workloads, prefetch offers significant performance improvements for workloads with read sequentiality.

In contrast to the existing work, this paper attempts to explore the performance behavior of on-board disk cache in the hierarchical cache architecture of typical computer systems, deconstruct the disk cache, and analyze the impacts of each function of the disk cache on the disk performance. Four real traces collected at block-level are employed to evaluate the impacts. Experimental results provide useful insights into the behavior of the on-board disk cache.

The remainder of this paper is organized as follows. An overview of hierarchical cache architecture is introduced in Section 2. Section 3 deconstructs the on-board disk cache, and describes each component of them including the cache organization, cache algorithms, and cache functions. The simulation environment and experimental evaluation are depicted in Section 4. Section 5 discusses some related works and future trend. Section 6 concludes the paper with remarks on main contributions and indications.

2. Hierarchical cache architecture

A computer is a complex system composed of several components such as CPU, main memory, disk drive controllers, buses, and disk drives. Fig. 1 shows a basic data flow of a typical computer system [10]. For example, an application issues a file-level I/O which writes its data to the local disk file system (e.g. ext2/ext3) through a write system call. According to write through policy, the data to be written will travel via the volume manager, device driver, PCI bus, disk controller, and finally turned into block-level I/O and be written to the disk drives. If write back policy is chosen, the data in the memory cache will be written back to the disk drives when the data is evicted from the cache. A reply will be finally sent back to the application. When a read request (file-level I/O) issued from an application wishes to access data stored in the disk drive, it first checks the memory cache. If the data can be found in the cache, the read request will be served and the corresponding data will be sent to the application immediately. Otherwise, the required data has to be first retrieved from the disk drives

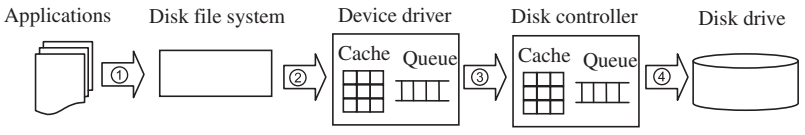

Fig. 1. Data flow and the corresponding computer components.