Chapter 10
Exploiting Disk Layout and Block Access History for I/O Prefetch

Feng Chen
The Ohio State University, USA

Xiaoning Ding
The Ohio State University, USA

Song Jiang
Wayne State University, USA

ABSTRACT
As the major secondary storage device, the hard disk plays a critical role in modern computer system. In order to improve disk performance, most operating systems conduct data prefetch policies by tracking I/O access pattern, mostly at the level of file abstractions. Though such a solution is useful to exploit application-level access patterns, file-level prefetching has many constraints that limit the capability of fully exploiting disk performance. The reasons are twofold. First, certain prefetch opportunities can only be detected by knowing the data layout on the hard disk, such as metadata blocks. Second, due to the non-uniform access cost on the hard disk, the penalty of mis-prefetching a random block is much more costly than mis-prefetching a sequential block. In order to address the intrinsic limitations of file-level prefetching, we propose to prefetch data blocks directly at the disk level in a portable way. Our proposed scheme, called DiskSeen, is designed to supplement file-level prefetching. DiskSeen observes the workload access pattern by tracking the locations and access times of disk blocks. Based on analysis of the temporal and spatial relationships of disk data blocks, DiskSeen can significantly increase the sequentiality of disk accesses and improve disk performance in turn. We implemented the DiskSeen scheme in the Linux 2.6 kernel and we show that it can significantly improve the effectiveness of file-level prefetching and reduce execution times by 20-53% for various types of applications, including grep, CVS, and TPC-H.

DOI: 10.4018/978-1-60566-850-5.ch010
INTRODUCTION

As the Moore’s law states, over the last three decades the processor speed doubles every 18 months, which brings a steady performance improvement at an exponential rate. In contrast, the access time of the hard disk, an electro-mechanical device, has been improved at a much slower pace, only around 8% per year (Gray & Shenoy, 2000). As a result, the performance gap between processors and hard disks is increasingly widening and this trend will continue in the future. As shown in Figure 1, in 1980 each disk access costs around 87,000 CPU cycles only, while this number grows to 5,000,000 cycles in 2000 (Bryant and O’Hallaron 2003). In other words, relative to the processor speed, the hard disk is becoming 57 times slower during the twenty years. Such an ever-growing performance gap between the processor and the hard disk strongly indicates that, the disk performance is becoming the key bottleneck of overall system performance.

The excessively high access latency of the hard disk essentially stems from its mechanic nature. Hard disk drives store data on the surface of rotating disk platters. These data can be read or written through the disk head attached on the moving disk arms. In general, accessing one data block involves three major operations, each of which causes a delay accordingly. When the hard disk receives a request (read/write) to data on a certain location, the disk arm must first position to the correct disk track where the data is located. This operation results in a seek latency. Then the disk head has to wait until the disk platters rotate to the correct position where the target data block is right beneath the disk head, which causes a rotational latency. Finally, data transfer can be started from or to the disk platter surface, depending on the operation type (read/write), which leads to a transfer latency. These three types of latency together form the aggregate latency of servicing a disk request. Since seek and rotational latencies are essentially determined by the speed of mechanic parts, the first two operations usually account for a large portion of the aggregate service latency to complete a disk access and should be minimized.

The performance of the hard disk is highly dependent on the workload access pattern, i.e.