Prefetching

Pedro H. Penna, Henrique Freitas,
Márcio Castro and Jean-François Méhaut

Pontifical Catholic University of Minas Gerais
Federal University of Santa Catarina
University of Grenoble Aples

Abstract

The I/O subsystem should provide to the user efficient access to I/O devices. If it does not, the user will certainly become frustrated with his/her virtual experience. In this assignment you should implement a powerful technique that boosts the performance of storage devices: disk block prefetching.

Background

Among the several existing I/O devices, one can point out storage devices as one of the most important. They provide a place in which the user can persist some data and the operating system may use to implement virtual memory.

However, these devices have the drawback of presenting slow read/write ratios, thereby yielding to poor performance. For instance, a single read request may take up to 30 ms, in contrast to 1 ms for the same request in memory. The rationale behind this comes from the fact that storage devices are essentially mechanical devices.

For this reason, and to address performance issues, techniques such as request scheduling, disk block caching and disk block prefetching are frequently employed. In the first technique, read/write requests are scheduled so as to minimize the repositioning time of the disk arm/head. In the second technique, most used blocks are kept in a in-core memory, so that requests on these blocks can be served faster. Finally, in the third technique data is

read in advance to the in-core disk block buffer, thereby speeding up linear read requests.

Assignment Description

The I/O subsystem in Nanvix uses the caching technique to speedup disk-use performance. Requests of user processes are directly served from in-memory buffers which the kernels maintains in a LRU (least recently used) in-core cache.

Although simple, this technique yields to a better overall performance for the I/O subsystem in both, read and write requests. More precisely, caching enables delayed write for write requests and data reuse for read requests, thus maximizing the average disk-bandwidth.

Nevertheless, the overall performance of the I/O subsystem may be further improved with prefetching. For instance, suppose a scenario in which a process performs linear access to a file, like a mp3 does. When only the caching technique is employed, whenever a read request is issued, a cache miss will raise and the request will have to be served from the disk. Alternatively, with prefetching, next blocks in a linear read are loaded in advance asynchronously, and thus increase cache hits. Therefore, in this assignment you should implement the prefetching technique in Nanvix.

Start Point

The implementation of the I/O subsystem is in the `kernel/fs` directory, and it is split in several files. However in this assignment, you should focus on the `buffer.c` file, where all the functions and routines for disk-block buffer cache are implemented. More precisely, you should study the `bread()` function, which serves read requests from the cache. A good idea is to use rely on the implementation of this function to implement a `breada()` function which performs normal read followed by prefetched reads. You should hack the `read()` system call to make use of this new feature, and you should rely on the `test` utility to benchmark your solution.