

TFS4 Buffer Cache

Technical Paper

September-2007, Version 1.0





Copyright Notice

Copyright © 2007, Flash Software Group, Samsung Electronics Co., Ltd

All rights reserved.

Trademarks

TFS4 is a trademark of Memory Division, Samsung Electronics Co., Ltd in Korea and other countries.

Restrictions on Use and Transfer

No part of this guide may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Samsung Electronics Co., Ltd. as governed by Republic of Korea and international copyright laws.

Document confidentiality status: Restricted Distribution

Readers of this document are not granted the right to distribute this document to any third party without prior written agreement from Samsung Electronics Co., Ltd.

Samsung Electronics Co., Ltd may make changes to the contents of this document any time without notice.

THIS DOCUMENT IS INTENDED ONLY TO ASSIST THE READER IN THE USE OF THE PRODUCT. SAMSUNG ELECTRONICS Co., Ltd. SHALL NOT BE LIABLE FOR ANY LOSS OR DAMAGE ARISING FROM THE USE OF ANY INFORMATION IN THIS DOCUMENT. SAMSUNG ELECTRONICS Co., Ltd. ASSUMES NO RESPONSIBILITY FOR ANY ERROR OR OMISSION IN INFORMATION CONTAINED IN THIS DOCUMENT, OR ANY INCORRECT USE OF THE PRODUCT.

Contact Information

Flash Software Group
Memory Division
Samsung Electronics Co., Ltd

Address: San #16 Banwol-Dong, Taeon-Eup
Hwasung-City, Gyeonggi-Do, Korea, 445-701



Introduction

Buffer Cache Management is a solution to reduce direct device read/write operations, and thus improve file system performance. To achieve greater performance with applications that manage large amounts of data, it becomes important to have data brought into cache before it's accessed, to retain such information in cache until it's no longer needed, and possibly to defer writing of modified data to disk to obtain greater efficiency. In order to achieve this purpose, Buffer Cache Management caches device data, and try to let device read/write operations carry out on cache instead of on disk and managing data transfer between a device drive and a cache buffer. The buffer cache manager encapsulates the functionality required to cache device data. In order to perform this task, the Cache Manager interacts with file system drivers.

The functions of Buffer Cache Management include:

- **Read/Write sectors** - Buffer Cache Management asks device driver to do sector read/write operations and cache the sectors' data. If sectors' data are already cached, sector read/write operations can be carried out on cache.
- **Flush cache** - Flush cached data to device.

Buffer Cache Manager caches data at the file system level. When an application accesses data in a file, the file system checks to see whether the data resides in the cache. If the data is in the cache, the file system simply issues disk I/O requests. The file system operates on the data in the cache.

The Cache management has couple of advantages. First, file system can bypass the direct I/O operation to disk and retrieve requested data directly from the cache. Second, the Cache Manager provides delayed-write functionality for modified cached data. By keeping the modified data in cache for some time before actually writing it to disk, the Cache Manager provides greater responsiveness to the user applications that actually perform the write. It can also batch multiple contiguous write operations in cache and write all the modified bytes out in a single I/O operation, which is typically more efficient than performing each smaller write operation individually. Finally, it's possible that a user application may repeatedly modify the same byte range. By deferring the I/O to disk, such modifications are made only in memory, avoiding completely the overhead of repeated write operations to the media.

Cache Read Operation

A typical cache read command operation begins with a conventional read command sequence to input the file start address and length of file to be read and execute the initial read from the memory. In Buffer Cache Management, to read file data, if Buffer Cache Management already cached the file data, Buffer Cache Management can return data to user directly, and no device read operations by device driver are needed. If the specified block is not cached, the Buffer cache management reads the data from the disk and copies it into the cache and finally returns the data to the user.

Cache Write Policy

According to the present system, file data created by the user can be first put to the cache memory (e.g., written in the memory by the application) before they are saved in the disk. When the system writes to a sector that is currently held in cache, it only writes the new information to the appropriate cache unit. The cache unit is single unit of cache memory that stores several contiguous data. These data, however, can be written to the disk for more permanent storage using a pre-selected algorithm, e.g., when there is not enough memory space in the cache memory or when reaching a certain percentage full in the cache memory. The decision for putting these file data to the disk can be made by the Buffer cache module. When there is not enough cache unit to store the file data, the buffer cache management selects one of the cache unit using cache replacement strategy, flushes the dirty data from the cache unit to the disk and puts new data in to the cache unit. This policy is also called **write-back cache policy**.

The cache module keeps track of which locations in the cache have been changed and therefore which memory locations may be stale. This is done by using an extra signature per cache unit, called the "dirty mark". Whenever a write is cached, this mark is set (made to 1) to tell the cache module "when you decide to re-use this cache unit for a new data, you need to write the current contents back to disk". In case of write-back policy, you could have stale data on your disk compared to what is in cache memory. The TFS4 maintains a data structure to log all the dirty cache unit related to a file and implements the 'file flush' so that cache is in sync with the disk and there is no stale data. Also during the file close operation, the data from the cache is flushed back to the disk.