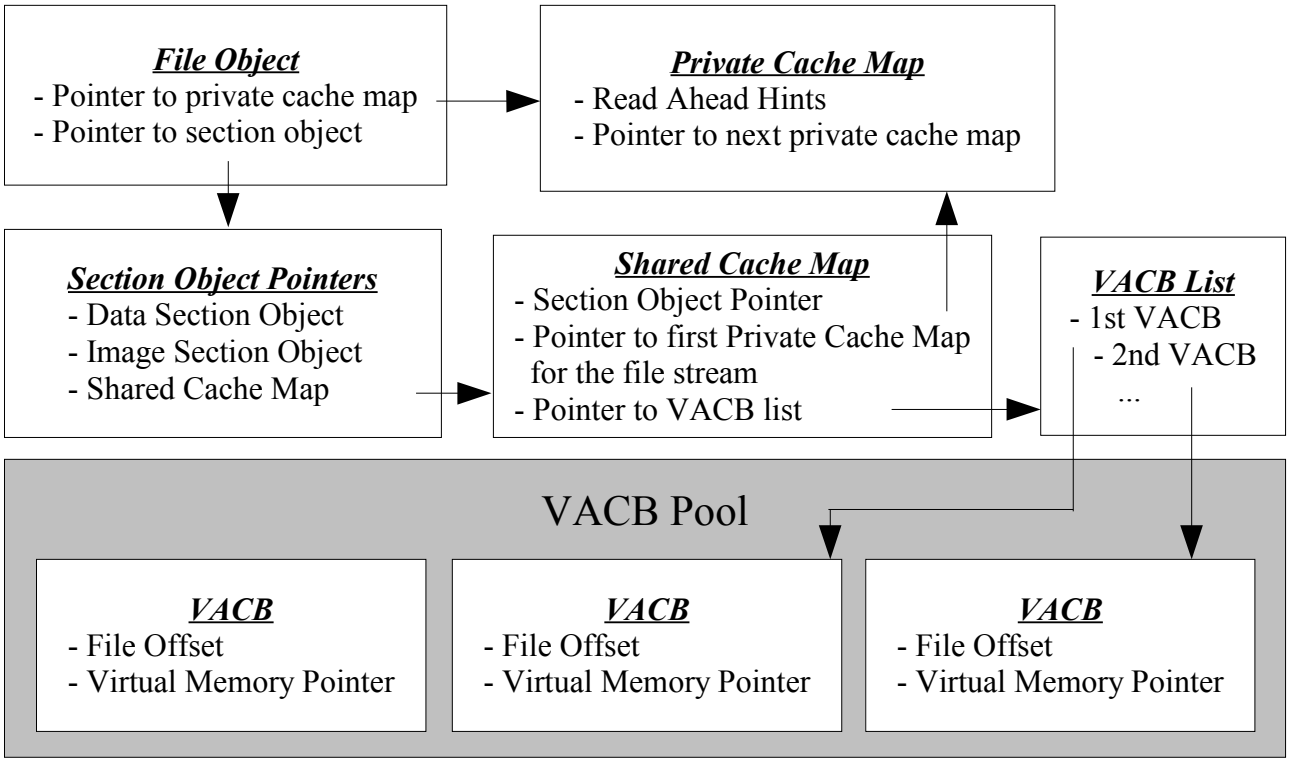
# Cache Manager

[FIXME: overview]

## Data Structures




- ***Private Cache Map***
  Each file object that has been initialized for caching using CcInitializeCacheMap has it's own private cache map structure that is linked from the FILE_OBJECT. For files where caching hasn't been initialized the poiner to private cache map in FILE_OBJECT is set to NULL and so the file system driver can use it for determining if the file stream is cached or not. This structure contains information used for read ahead optimalization.

- ***Shared Cache Map***
  This structure is allocated for each file stream that has caching initialized. It's shared across all file objects representing the file stream and is accessible using the FileObject -> SectionObjectPointer -> SharedCacheMap pointer. This structure consists of pointer to VACB (Virtual Address Control Block) array.

- ***Virtual Address Control Block***
  Each mapped view is represented using instance of this structure. The Cache Manager maintains a fixed size pool of the VACB strcutures and assigns them to specific file streams when needed. All VACB's assigned to specific stream are listed in the Shared Cache Map structure. Each of them contains an virtual address that can be used to access the view and file offset.

- ***Buffer Control Block***
  is a key structure for cache manager interfaces. It's opaque to file system drivers and represents a currently mapped buffer.

## Cache Operations

- ***Read Ahead***
  is operation commonly used when synchronously reading from a file. It's closely tied to the copy interface (described below). The cache manager stores information about two previous read requests in the private cache map and when a next read request is issued it checks if there isn't pattern between the requests. If that is the case, the cache manager wakes up a read ahead thread after satisfying the current request and this thread continues file stream reads with the same pattern. This ensures that, if the pattern will continue, the data will be immediately available on the next request.

  | *NOTE* |
  | --- |
  | Pattern matching is accomplished by checking if the difference between the file offsets of the previous two requests is same as the one between current request and the previous one. Also all the three requests must be for the same amount of data. |

- ***Lazy Write***
  Data stored in the cache that are marked as dirty (meaning they do not correspond to data currently present on the underlying medium) are periodically flushed to the disk (can be other medium for network filesystems, but for simplicity the term “disk” is used in the following text). This operation is called lazy write and is managed by a special thread that performs flushing of dirty data once per second. Not all data are flushed though, but only the amount specified by threshold value in percents. Flushing all the data would result in slightly worse performance because lot of I/O would be performed at once resulting in system slowdown and it's also likely that the most recent dirty blocks will be accessed again.

## Cache Manager Interfaces

## Public Functions

## Glossary