Operating Systems 211

Peter Pietzuch - prp@doc.ic.ac.uk

Tutorial - File Systems

1. Explain what it means to "defragment a file system"? Are there file systems that do not require defragmentation? How can a file system reduce the amount of fragmentation?

   Answer: Defragmenting a file system means that blocks belonging to the same file are moved around to ensure that they occupy consecutive disk blocks. This improves the read/write performance because it avoids unnecessary disk seeks.

   In general, any file systems can become fragmented to a certain extent. However, some file systems may suffer less from fragmentation when they ensure that blocks for files are allocated in nearby or consecutive locations on the disk. For example, block allocations may be done according to zones.

2. Consider a file system that maintains a unique inode for each file in the system. Each inode includes 8 direct pointers, a single indirect pointer, and a double indirect pointer. The file system block size is 1024 (2^{10}) bytes, and a block pointer occupies 4 bytes.

   How many disk operations will be required if a process reads data from the Nth block of a file? Assume that the file is already open, the buffer cache is empty, and each disk operation reads a single file block. Your answer should be given in terms of N.

   Answer:

   - if $0 \leq N < 8$, then one operation is required
   - if $8 \leq N < 256 + 8$, then two operations are required
   - if $256 + 8 \leq N < 2^3 + 2^8 + 2^{16}$, then three operations are required

3. Consider a file system that uses inodes with single-indirect and double-indirect blocks and a block size of 1024 (2^{10}) bytes. If the block size of the file system is doubled, by approximately what factor does the maximum possible file size increase? Your answer should be an integer.

   Answer: For very large files, almost all data blocks are accessed through the double indirect pointer, so consider only those blocks. For a block size of B and a pointer size of P, the total size of the double-indirect data blocks is

   $$(B/P)^2\, B = B^3 / P^2$$

   So, if B is doubled, the maximum file size will increase by approximately a factor of 2^3, or eight.

4. Consider a hierarchical file system in which free disk space is kept in a free block bitmap.
    a. Suppose that the stored free block bitmap is lost. Can the OS reconstruct the lost information? If so, describe how.

       Answer: Essentially the OS has to run a garbage collection algorithm on the files to recover the free block information. First it would reinitialise the free block bitmap. Then, it would start at the root of the file system (the "/" directory), and mark every block used by every file found through a recursive descent through the file system. When finished, it would update the stored free block bitmap. This is essentially what the UNIX utility fsck does to recover from meta-data corruption.

    b. Suggest a scheme to protect the free block bitmap against such accidental loss.

       Answer: There could be multiple copies of the free block bitmap stored at different locations of the disk. In addition, updates to it would have to be atomic.

5. Consider a file system with an inode organisation. Suppose that, for a given file, the file system has filled up all the blocks stemming from the doubly indirect pointers. Assume that the inode and free block bitmap are both completely in memory, but there is no buffer cache.

   How many disk accesses will it take to write one more byte to the file?

   Answer: This would take four disk accesses. Allocating the four necessary blocks would be free, because the free block bitmap is in main memory. Connecting the triply indirect pointer of the inode to the block that contains doubly indirect pointers does not require any disk accesses, because the inode is already in main memory. Connecting the block that contains the doubly indirect pointer to the block that contains the indirect pointers takes one disk access. Connecting the block that contains the indirect pointers to the block that contains the data block pointers requires the second disk access. Connecting the data block pointers to the actual data block requires the third disk access. Finally, writing the one byte to the data block requires the fourth disk access.