操作系统作业 5

傅申 PB20000051

2022 年 6 月 4 日

问题 1.

(a) Suppose block A_1 will be updated

RAID-5 For RMW, block A_1 and A_p will be updated, 2 blocks will be accessed.

But for RWM, in addition to A_1 and A_p, block A_2, A_3 and A_4 will be read to calculate the new parity, so 5 blocks will be accessed.

RAID-6 For RMW, block A_1, A_p and A_q will be updated, 3 blocks will be accessed.

But for RWM, in addition to A_1, A_p and A_q, block A_2 and A_3 will be read to calculate the new parity, so 5 blocks will be accessed.

(b)

RAID-5 Suppose A_1, A_2, A_3, A_4, B_1, B_2 and B_3 will be updated.

For RMW, since the update of parity needs the old data, every data block update would cause the parity block to be updated, thus, $2 \times 7 =$ 14 blocks will be accessed.

But for RRW, the update of parity only needs the new data, so the parity block update can be done after all the data block updates. In this case, B_4 will be read to calculate the new parity, thus, 10 blocks (7 updated block + B_4 + 2 parity block) will be accessed.

RAID-6 Suppose A_1, A_2, A_3, B_1, B_2, B_3 and C_1 will be updated.

For RMW, since the update of parity needs the old data, every data block update would cause the 2 parity blocks to be updated, thus, $3 \times 7 =$ 21 blocks will be accessed.

But for RRW, the update of parity only needs the new data, so the parity block update can be done after all the data block updates. In this case, C_1 and C_2 will be read to calculate the new C_p, thus, 15 blocks (7 updated block + C_2 + C_3 + 6 parity block) will be accessed.

问题 2.

FCFS Order: 2150, 2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681;

Total distance: 13011 cylinders;

SSTF Order: 2150, 2069, 2296, 2800, 3681, 4965, 1618, 1523, 1212, 544, 356;

Total distance: 7586 cylinders;

SCAN Order: 2150, 2296, 2800, 3681, 4965, (4999), 2069, 1618, 1523, 1212, 544, 356;
Total distance: 7492 cylinders;

LOOK Order: 2150,2296, 2800, 3681, 4965, 2069, 1618, 1523, 1212, 544, 356;
Total distance: 7424 cylinders;

C-SCAN Order: 2150, 2296, 2800, 3681, 4965, (4999, 0), 356, 544, 1212, 1523, 1618, 2069;
Total distance: 9917 cylinders;

C-LOOK Order: 2150, 2296, 2800, 3681, 4965, 356, 544, 1212, 1523, 1618, 2069;
Total distance: 9137 cylinders;

问题 3.

Most of file operations involve searching the directory for locating the file. To avoid this constant searching, many system require that an `open()` system call be made before a file is first used. The operating system keeps a table, called the **open-file table**, containing information about all open files. When a file operation is requested, the file is specified via an index into this table, so no searching is required. When the file is no longer being used, it is closed by the process, and the operating system removes its entry from the open-file table, potentially releasing locks.

With the open-file table, the open files is indexed in memory, easily accessible by the process. We can get the open file from the open-file table rather than directory traversal for each access. Considering that we usually have a sequential operation to the files, the open-file table can improve performance and reduce I/O.

问题 4.

`755` stands for `rwxr-xr-x`, meaning

- The file's owner can read, write and execute the file.
- The file's group and others can read and execute the file.

问题 5.

Suppose we need to write a new large file to the file system that we have enough space for. But there is no hole big enough to fit the new file. The problem called external fragmentation occurs. To solve this problem, we need the very expensive defragmentation process to reorganize the file system, moving files together to make space for the new file.

Another problem is that the file may not have enough space to grow. To solve this problem, we need to relocate the file to a new location that has enough space.

问题 6.

The advantage:

- External fragmentation problem is solved.
- Files can grow and shrink freely.
- Free block management is easy to implement.
- The random access problem can be eased by keeping a cached version of FAT inside the kernel.

The major problem is that the entire FAT has to be stored in memory so that the performance of looking up of an arbitrary block is satisfactory.

问题 7.

The read order is:

1. Read the root directory.
2. Read the index node of `/a`.
3. Read the disk block of `/a`.
4. Read the index node of `/a/b`.
5. Read the disk block of `/a/b`.
6. Read the index node of `/a/b/c`.
7. Read the disk block of `/a/b/c`.

(a) All read operation will invoke an I/O operation, thus, 7 I/O operation is required.

(b) The inodes are in the memory, so reading them won't cause I/O operation. 4 I/O operation is required.

问题 8.

The block size is 8 KB, i.e., 2^{13} bytes. A pointer requires 2^2 bytes. The maximum file size would be

$$\begin{aligned}&\underbrace{12 \times 2^{13}}_{\text{Direct Block}} &&+\underbrace{1 \times 2^{13-2} \times 2^{13}}_{\text{Indirect Block}} &&+\underbrace{1 \times 2^{13-2} \times 2^{13-2} \times 2^{13}}_{\text{Double Indirect Blocks}} &&+\underbrace{1 \times 2^{13-2} \times 2^{13-2} \times 2^{13-2} \times 2^{12}}_{\text{Triple Indirect Blocks}}\\ =&12 \times 2^{13} &&+2^{24} &&+2^{35} &&+2^{46}\end{aligned}$$

The result is 70,403,120,791,552 bytes, i.e., approximately 64 TB.

问题 9.

A hard link is a directory entry pointing to an existing file. When it is created, there is no new file content created, and the link count of the target file will be incremented. It's like a new pathname of the target file is created.

A symbolic link is a file (shortcut) storing the pathname of the target file. When it is created, there is a new inode created, and the link count of the target file won't change.

问题 10.

The difference between data journaling and metadata journaling is that in data journaling, the data will be written to disk twice, once to the journal and once to the on-disk location, but in metadata journaling, the data will be written to disk only once, to the on-disk location only.

The sequence for data journaling is:

1. Journal write: Write the contents of the transaction (including TxB, metadata, and data)
2. Journal commit: metadata, and data (including TxE)
3. Checkpoint: Write the contents of the update to their on-disk locations
4. Free: free the space of log

The sequence for metadata journaling is:

1. Data write & Journal metadata write: Write the contents of the transaction (including TxB and data) and write the contens of the on-disk locations. Two write can be issued in parallel.
2. Journal commit: Write TxE
3. Checkpoint: Write the contents of the update to their on-disk locations
4. Free: free the space of log

问题 11.

Polling, Interrupt and Direct Memory Access (DMA).

问题 12.

- I/O scheduling
  - Maintain a per-device queue
  - Re-ordering the requests
  - Average waiting time, fairness, etc.
- Buffering - store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch

  - To maintain “copy semantics” (e.g., copy from application’s buffer to kernel buffer)

- Caching - faster device holding copy of data
  - Always just a copy
  - Key to performance
  - Sometimes combined with buffering
- Spooling - hold output for a device

  If device can serve only one request at a time, e.g., Printing
- Error handling and I/O protection
  - OS can recover from disk read error, device unavailable, transient write failures
  - All I/O instructions defined to be privileged
- Power management, etc.