Homework 14

CS307-Operating System (D), Chentao Wu, Spring 2020.

Name: 方泓杰(Hongjie Fang)   Student ID: 518030910150   Email: galaxies@sjtu.edu.cn

- (14.1) Consider a file currently consisting of 100 blocks. Assume that the file-control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one block, the following conditions hold. In the contiguous-allocation case, assume that there is no room to grow at the beginning but there is room to grow at the end. Also assume that the block information to be added is stored in memory.

  a. The block is added at the beginning.
  b. The block is added in the middle.
  c. The block is added at the end.
  d. The block is removed from the beginning.
  e. The block is removed from the middle.
  f. The block is removed from the end.

  Solution. Here are the final solutions of different situations.

Situation	Contiguous-allocation	Linked-allocation	Indexed-allocation
Add a block at the beginning	201	1	1
Add a block in the middle	101	52	1
Add a block at the end	1	3	1
Remove a block from the beginning	198	1	0
Remove a block from the middle	98	52	0
Remove a block from the end	0	100	0

  Here are the specific explanations. In the following explanation, we ignore the operation to the directory since the directory is stored in memory, not in disk.

  – **Method 1: Contiguous-allocation.**
    * **Situation 1: Add a block at the beginning.** Since there is no extra space in the beginning, we have to move the data backward. Therefore, we need to start from the last block, read it and write it in the next block. Repeat the process 100 times, and then we can write the target block at the beginning. Thus, totally we need 100 reading operations and $100 + 1 = 101$ writing operations. Hence, the total disk I/O operation number is $100 + 101 = 201$.
    * **Situation 2: Add a block in the middle.** We assume that we add a block right after the 50th block. Then we need to move the last 50 blocks backward, and we also need to start from the last block, read it and write it in the next block. Repeat the process 50 times, and then we can write the target block in the 51st block in the middle. Thus, totally we need 50 reading operations and $50 + 1 = 51$ writing operations. Hence, the total disk I/O operation number is $50 + 51 = 101$.
    * **Situation 3: Add a block at the end.** Since there are extra spaces in the end, we only need to write the target block in the 101st block at the end. Thus, totally we need 1 writing operation. Hence, the total disk I/O operation number is 1.

    - **Situation 4: Remove a block from the beginning.** Since there should be no extra space in the beginning, we have to move the data forward. Therefore, we need to start from the 2nd block, read it and write it in the previous block. Repeat the process 99 times. Thus, totally we need 99 reading operations and 99 writing operations. Hence, the total disk I/O operation number is $99 + 99 = 198$.
    - **Situation 5: Remove a block from the middle.** We assume that we remove the block right after the 50th block, that is, the 51st block. Since there should be no extra space in the beginning, we need to move the last 49 blocks forward, and we also need to start from the 52nd block, read it and write it in the previous block. Repeat the process 49 times. Thus, totally we need 49 reading operations and 49 writing operations. Hence, the total disk I/O operation number is $49 + 49 = 98$.
    - **Situation 6: Remove a block from the end.** We only need to change the length in the directory in memory, and we do not even need to change the data in the disk. Hence, the total disk I/O operation number is 0.
  - **Method 2: Linked-allocation.**
    - **Situation 1: Add a block at the beginning.** We need to write the new block and update its link to the original first block, which address can be read from the directory in memory. Therefore, totally we need q writing operation. Hence, the total disk I/O operation number is 1.
    - **Situation 2: Add a block in the middle.** We assume that we add a block right after the 50th block. We have to find the 50th block first by traversing through the links, so we need to perform 49 read operations before we get to the 50th block. Then we need to read the data in the 50th block. We write the data in the target block, and update the data in the 50th block and write it back to the 50th block. Thus, totally we need $49 + 1 = 50$ reading operations and $1 + 1 = 2$ writing operations. Hence, the total disk I/O operation number is $50 + 2 = 52$.
    - **Situation 3: Add a block at the end.** We need to write the new block and update the link of the original last block to it, so we need to read the original last block, update the data and write it back to the block. Thus, totally we need 1 reading operation and $1 + 1 = 2$ writing operations. Hence, the total disk I/O operation number is $1 + 2 = 3$.
    - **Situation 4: Remove a block from the beginning.** We need to read the 1st block so we can get the address of the 2nd block, then we only need to update the directory in memory using the address of 2nd block. Thus we only need 1 reading operation. Hence, the total disk I/O operation number is 1.
    - **Situation 5: Remove a block from the middle.** We assume that we remove the block right after the 50th block, that is, the 51st block. We have to find the 51st block first by traversing through the links, so we need to perform 50 read operations before we get to the 51st block. Then we need to read the data in the 51st block to get the address of the 52nd block. We update the link in the 50th block using the address of 52nd block and then we write it back to the 50th block. Thus, totally we need $50 + 1 = 51$ reading operations and 1 writing operation. Hence, the total disk I/O operation number is $51 + 1 = 52$.
    - **Situation 6: Remove a block from the end.** We need to find the block before the last block, that is, the 99th block, by traversing through the links, so we need to perform 98 read operations before we get to the 99th block. Then we need to read the data of the 99th block, clean the link in it and write it back to the 99th block. Then we need to use the address of it to update the directory in memory. Thus,

totally we need $98 + 1 = 99$ reading operations and 1 writing operation. Hence, the total disk I/O operation number is $99 + 1 = 100$.

- **Method 3: Indexed-allocation.**
  - **Situation 1: Add a block at the beginning.** We only need to write the new block and update directory index in the memory. Thus, totally we need 1 writing operation. Hence, the total disk I/O operation number is 1.
  - **Situation 2: Add a block in the middle.** We only need to write the new block and update directory index in the memory. Thus, totally we need 1 writing operation. Hence, the total disk I/O operation number is 1.
  - **Situation 3: Add a block at the end.** We only need to write the new block and update directory index in the memory. Thus, totally we need 1 writing operation. Hence, the total disk I/O operation number is 1.
  - **Situation 4: Remove a block from the beginning.** We only need to delete the target block's number in directory index in the memory, and we do not even need to change the data in the disk. Hence, the total disk I/O operation number is 0.
  - **Situation 5: Remove a block from the middle.** We only need to delete the target block's number in directory index in the memory, and we do not even need to change the data in the disk. Hence, the total disk I/O operation number is 0.
  - **Situation 6: Remove a block from the end.** We only need to delete the target block's number in directory index in the memory, and we do not even need to change the data in the disk. Hence, the total disk I/O operation number is 0.

□