Locality and the Fast File System

Annalise Tarhan

March 16, 2021

1 Examine the file in in.largefile and then run the simulator with the following flags. What will the resulting allocation look like?

Allocating a 40 block file with a large-file exception of 4 blocks spreads the file evenly among the 10 available groups.

2 Now run with -L 30. What do you expect to see?

Now, the first 30 blocks of the large file will be stored in the same group and the remaining 10 in another. (Except there is already a '/' at the beginning of the first group, which pushes an 11th data block to the second group.)

3 A file's filespan is the max distance between any two data blocks of the file or between the inode and any data block. Calculate the filespan of /a for -L 4 and -L 30. What difference do you expect in filespan as the large-file exception parameter changes from low values to high values?

The filespan when the large-file exception is four equals the distance from the file's inode to the last block, which is 372 if the last data block doesn't wrap around to the first inode. When the large-file exception is 30, it equals the distance from the file's inode to the last block, which is only 59. Larger large-file exception values tend to overwhelm groups but keep filespans small.

4 How do you think the FFS policy will lay the files in in.manyfiles out across groups?

All the files in the root directory will be placed in the first group and the newly created directories will be placed in subsequent groups. Each file created in one of those new directories will be placed in the same group as that directory.

5 Dirspan calculates the spread of files within a particular directory, specifically the max distance between the inodes and data blocks of all files in the directory and the inode and data block of the directory itself. Calculate the dirspan of the files in the three directories in in.manyfiles. How good of a job does FFS do in minimizing dirspan?

Root: 28 J: 20 T: 34
In this case, dirspan was minimal, since each directory was placed in its own group.

6 Now change the size of the inode table per group to 5. How do you think this will change the layout of the files? How does it affect dirspan?

The files are small, (specifically, the average file size is less than the number of blocks in a group divided by the new size of the inode tables,) so reducing the size of the inode table will cause it to fill up before the group's blocks do, spreading files in the same directory across groups and increasing dirspan.

7 Which group should FFS place the inode of a new directory in? The default simulator policy looks for the group with the most free inodes. Try looking at groups in pairs and picking the best pair for the allocation and see how allocation changes with this strategy. How does it affect dirspan? Why might this policy be good?

This strategy starts by skipping every other group, so if a directory fills its group, the one immediately following it is completely empty, which keeps dirspan as low as possible. This is an improvement if directories are likely to fill their original groups, but the drawback is that files start out more spread out, leading to longer seek times for non-consecutive files right from the beginning.

8 Run with in.fragmented and see if you can predict how the files that remain are allocated. What is interesting about the data layout of file /i? Why is it problematic?

This file creates a series of small files, then deletes every other one. Finally, it creates a large file /i, which ends up filling the holes, fragmenting the new file's contents across the entire group.

9 A new policy, contiguous allocation, tries to ensure that each file is allocated contiguously. How does layout change as the parameter passed to -C, which determines how many contiguous blocks are free within a group before allocating a block, increases? Finally, how does -C affect filespan and dirspan?

In this example, increasing -C past two makes no difference since all the holes are only one block wide. In a more varied situation, increasing the parameter passed to -C would avoid larger holes as well, reducing fragmentation and filespan at the cost of increasing dirspan.