Homework Wan Huzaifah bin Wan Azhar

Answer:

* Running ffs.py -f In.largefile will return allocation that will fill entire block group and the subsequent block group.
* However, adding -L 4 parameter will split the allocation into multiple block group, as in, only 4 block will be allocated to the request on each block group before moving to the next group.



* Running ffs.py -f In.largefile -L 30 will shows that the first block group will be fully allocated before moving to then next block group.
* This is because the default block per group is 30.



* Changing large-file exception parameter from low values to high value will decrease the max filespan because now files are closely to each other and thus the maximum distance between two data blocks is less.



* FFS policy will group the allocation by directory. For example, root node and its data will be placed in the first block group.
* Both /j and /t and its files will be placed in different block group, which only contains their files and inode but not in the first block group.



* span: directories

dir: / dirspan: 28

dir: /j dirspan: 20

dir: /t dirspan: 34

avg dirspan: 27.33

* Because inodes and data block of a directory and file is close to each other, FFS is doing a very good job of minimizing dirspan.
* This is true assuming large file exception parameter is not too low.



* Changing inode table per group to 5 will cause the block group of the root directory to be separated into two block groups.
  + This is done because not many files can be allocated in one block group.
  + A data block can be full simply by having more than one file to be bigger. However, the maximum type of file should still be 5.
* In total, there will be close to 10 or 11 block group, with the first two block group belongs to root directory.
* The rest will be /j and /t directory. However, as there is not enough inode to allocate files to /j and /t, the subsequent group will be allocate to the files, from both directory, which will mix.
* The end result is that the final two block group will not be in the same block group as its directory.
* Dirspan will increase in all directory.



* This policy will reduce dirspan as directory and its files (and inodes) will be close to each other as they are in a set of block group.
* This policy is good because now you can have less inode but still maintain the performance.



group inodes data

0 /ib-d-f-h- ibidifihiiii------------------

1 ---------- ------------------------------

2 ---------- ------------------------------

* The file “/i” are interleaved between other files.
* This means that the file “/i” are not in contiguous block.
* This is problematic because when reading file “/i”, the disk cannot load it in one continuous process.
* So, the disk has to read one data block, seeking the next data block and so on. This will incur performance cost because the file is no longer sequential and much more random access.



* The higher -C n is, the further the filespan of the file as the system will try to allocate into free n blocks, but if its not found, then it will search for another block group.
* -C will increase filespan and dirspan, at the benefit of having contiguous memory.