WEEK14 Homework
Tanu Shree

Q1.

In log-structured filesystem, the focus is on write performance. File system changes are buffered in file cache and small random writes are aggregated into large asynchronous sequential writes.
Disk is represented as a stream of blocks that can be written into. When a process creates a new file, data block of new file 'Df' is written to the WRITE buffer. Next new inode 'If' of the file is written to the buffer. The entire inode block is not cached but rather just the inode itself. After that, updated directory block 'Dd' and directory inode 'Id' is written into the WRITE buffer. At some point, the data in the buffer is pushed to the disk in one go sequentially. After that, whenever the file is modified, i.e. data is written into the file until it reaches 20KB size, the updated datafile 'Df'' and inode 'If'' are written into the buffer sequentially. Every time the data is written into the file, the datafile and its inode is updated and written into the buffer sequentially. Sometime later, these are pushed into the disk just after the previous datafiles.
Disk:

Already used	Df	If (B[0]= A0)	Dd	Id (B[X]= A1)	Df'	If' (B[0]= A2)	Df''	If'' (B[0]= A3)	…..	IMAP
	A0		A1		A2		A3			

As the data is pushed into the disk, the log structured file system maintains the pointer to the end of the last write, here at A0. The location of the data in the inode points to the location of data block, here at A0. As the data is written, the end pointer is updated to A1. The location of updated block in directory file say block X is therefore A1 and so on. The inodes are linked together in IMAPs, which makes the access to inodes faster. In LFS, there is no need to maintain free-block list or bitmap, because new space for block is generated through garbage collection process. In this the items which are no more current, can be deleted and live data can be moved to free segment on the disk, therefore increase the block space.

Q2.

NAND flashed storage space is structured into blocks, which in turn are partitioned into pages. Read and write operations are done on pages (block wise). There are basically three operations supported by flash memory:
Read individual pages, Program individual pages (write can only be done on empty page, data cannot be overwritten) and erase data from entire block. Data cannot be erased from individual pages. There is an erase-write sequence followed in NAND flash and since each block can handle limited number of such erase-write cycles, so it is distributed across the block.

The first two systems UNIX Original File system and UNIX Fast File System use journal file system (write-in-place file system). It keeps the on-disk state of the file system consistent by writing a summary of each write operation to the log store somewhere on the disk before writing the changes directly to the long-term place in file system. So, every change in the file

system gets written to the disk twice: once to log and once in permanent location which makes it expensive writing to disk is costly. So, these are not good at performance.

Instead of writing operation to the log and then rewriting the changes in place somewhere else, log structured file systems can directly write the operation to the end of the log. Log structured file system can be treated as one large log and that be used to write the operation directly reducing the cost of write. The log is written in large chunks called “segments”. When the file system needs fresh segment, it first cleans the existing one (garbage collection) by moving the live data to free segment and then do a big stream of write to the empty segment. This system of segments and cleaning is like that done in the NAND flash device, a necessity to erase large contiguous blocks before writing to them. So, log structured file system among the three mentioned here will perform best with the NAND flash SSD.