DESIGN DOCUMENT

main() create K threads and push empty string in shared queue" dirList" to indicate root.

compareDir()

1)assignID() --> assignID in case of pthreads.

- 1)assignID() --> assignID in case or prineads.
 (a) Scan picked element of the queue and take action on child depending on whether the child is directory or file. Push this child in thread specific/threadprivate deque "dirContent".
 (b) If directory--> map the relative pathname on "dmap" and increment count-> using mutex/critical region--> If a particular thread make it reach threshold, that thread would add this relative name in BFS queue and add it to shared deque "outputD" --> using mutex/critical region--> as a potential candidate for majority subtree.
 (c) If file, map it using as structure as key-> on the basis of relative pathname, size and MD5 Hashvalue and if count reaches threshold, push it is shared use "It size", a using mutex/critical region.

 - push it in shared queue "fList" --> using mutex/critical region.

 (d) One element of BFS is processed and keep a barrier so that all threads reach this point. Clear all maps and not needed variable. Loop again to empty the BFS queue.



compareFiles()

- 1)Pick each element of "fList" queue--> files having majority in terms of relative pathname, size and MD5 Hash value--> Parallel Execution in all
- 1)Pick each element of "fList" queue--> files having majority in terms of relative pathname, size and MD5 Hash value--> Parallel Execution in all file systems.
 2)Perform following actions on each file of "fList" -->

 (a) Use" fread" function to read content of file in small chunks and hash it using shared "fileCMap"--> barrier--> If count is equal to threshold, then set loopvar=1 to make this thread to again go in the loop and read more content. If a particular file reached the end of file, set local flag" Iflag" to zero.
 (b) Clear map after every loop for reading contents.
 (c) After coming out of the loop, check the number of local flags set to 0 which indicate how many files reached end-of-file. If the number of flags set to 0 are greater than or equal to threshold, then it indicate the contents of the files were same and push it to" outputf" deque
 (f) Close the file pointer if not null and wait on the barrier so that all threads would have processed this file flaterate agains with another.

 - (d) Close the file pointer if not null and wait on the barrier so that all threads would have processed this file. Iterate again with another element of the queue.



dirSubtree()

- 1)Create a thread local map" threadMap". Hash the potential candidates list of directories i.e. "outputD" and majority files list i.e. "outputF" and set values as 0 and 1 respectively--> Parallel Execution in all file systems.

 2)Read thread-specific/threadPrivate deque" dirContent" in bottom up approach(from reverse--> originally pushed in level order way) and
- perform following actions—>

 (a) If it is directory, check if its present in the map i.e. is a potential candidate—> Scan its content and check if all children are in majority i.e are present in map and have value as 1. If yes, then set the value for this directory as 1.

 (b) Perform the same check for all the elements of deque—> running concurrently in all the filesystems.
- 3) Iterate over the map, insert the elements into the set of majority directory--> "finalDir"--> using mutex/critical regions. 4)Clear the map and all not needed variables
- 5) For 2 filesystem--> 1 set is maintained for each filesystem--> having the set difference w.r.t to majority i.e. set of elements which are not in



reportResults()

1)For K file system->

- (a) divergent set indicate divergent file systems.
 (b) exact flag set indicate exact file systems.
 (c) "finalDir" have the common subtree.

- 2)For 2 file system-->

 (a) exact flag indicate the exact file systems.
 (b) Set S1 and S2 have contents which are not in majority--> intersections of these sets indicate content which is changed and set difference w.r.t. each other indicate addition/deletion.

Some discussions have been done with Lokesh