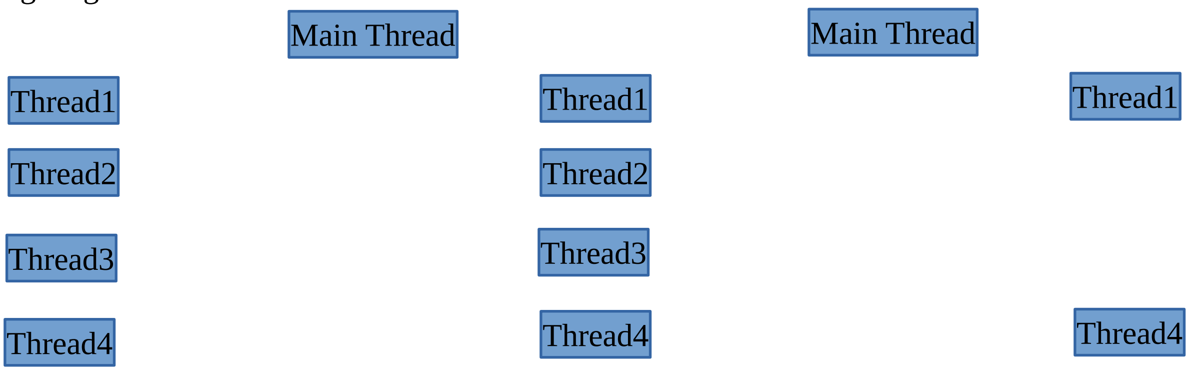
Design Document

2012B5A7589P Rajath S

P-threads -

1. K-Threads are spawned in the main ().
2. Each thread is assigned one of the given filesystems.
3. 4096 B of buffer space is allocated for each thread.
4. Main thread controls the other K-Threads.
5. K-Threads work independently on their own filesystems and dump data on the buffer and wait for an instruction from the main thread.
6. Main thread has two functions -
    a. Compare the data in the buffers and mark if a file/folder is majority / common / non-majority.
    b. Determine the next action to be performed by the children threads.
7. The action performed by threads is indicated by a flag set by the master thread. Some flags set by the master are :
    a. exit_thread – indicates that the thread must tidy up the memory usage and close its operation.
    b. next_child – indicates that the thread must update the buffer with the next path encountered in the current folder.
    c. next_queue – indicates that the next folder on the BFS Queue must be expanded.
    d. current_field – indicates what data related to a file must be copied into the buffer. (0 – pathname, 1 – size, 2 – hash, 3 – 4096 bytes of filecontent).
8. The master thread is activated only after all the K-Threads have finished their action and copied data to the buffer.
9. Master thread selectively activates the threads based on the data requirement for comparison.
10. Concurrency is achieved using semaphores. Each thread waits on its own semaphore. Main thread also waits on its own semaphore.
11. The last thread to complete the action signals the master thread.
12. Master thread signals the threads based on the data requirement.

Timing diagram -



OpenMP -

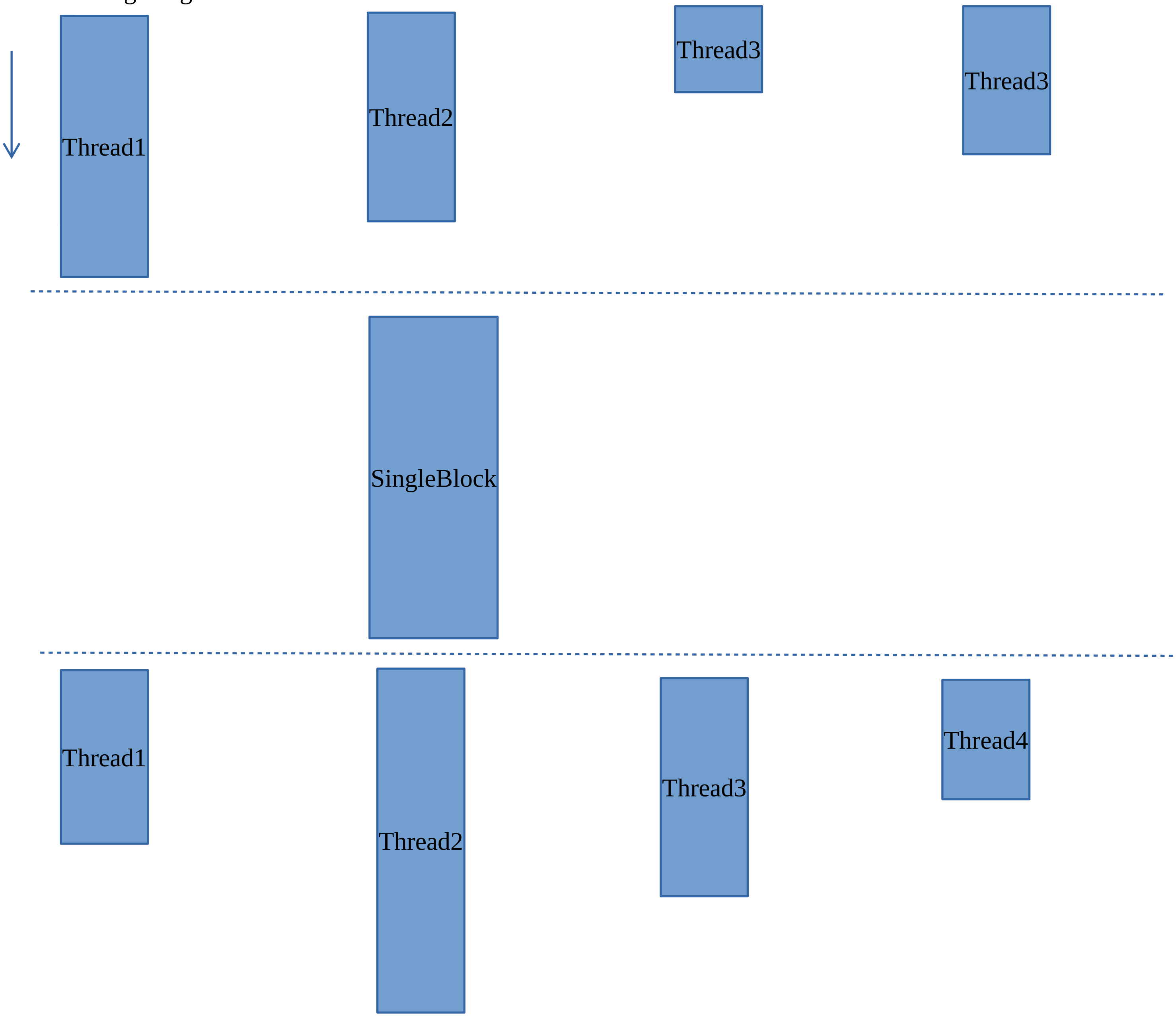
1. A parallel block is executed by K-Threads.
2. A barrier synchronizes all the K-Threads.
3. A single block is executed only by one of the K-Threads.

4. The implicit barrier at the end of single block releases all the K-Threads at once to continue execution.

5. Single block code determines action taken by each individual thread on their own filesystems.

[Everything else is exactly as described in the pthread section].

Timing Diagram -



Note -

1. Iterative BFS is used for Filesystem traversal. A Queue is used for this purpose. (STL Queue).

2. Efficient Comparsion is carried out using QuickUnion data structure (Creating connected components on the fly – for equivalence classes). [QuickUnion implemented as an array].

3. An associative container (balanced tree) is maintained to store the presence/absence data related to folders (in the form of a bitvector) on the higher level. (Standard Template Library implementation of a

“map” is the Red-black trees). The map is indexed by the relative path from the root.
4. Memory usage measurements are carried out using the Massif tool of Valgrind.
5. Timing measurements from VampirTrace.
6. MD5 checksum is used as the hashing function. [OpenSSL library].
7. C++11 standard is followed.
8. Majority Copy >= (floor(num_threads/2)+1)
8. The following interpretation of majority folder is used :
A bit vector is used to indicate the presence/absence of majority copy across different filesystems.
e.g. 1011 means fs1, fs3 and fs4 have the majority copy.
If the intersection of bitvectors of all majority children of a folder also has greater than ceil(k/2) 1s',
then the folder is considered to be majority.
9. Output format :
PATH is a common file/folder.
PATH is a majority file/folder. Filesystems : [list of fs where the majority copy is present]

If a PATH is not printed – means it is neither a common copy nor a majority copy.