Problem 1A

Problem Statement :-

Devise a way to store approximately 1mn Images each with Average size of 100KB.

Constraints :-

1mn Images each of size 100KB

- Total Size = 100 * 1000000 = 100GB
- RAM has 32MB space left, so images can't be stored in RAM
- Hard Drive has unlimited space.

Possible Solution :-

Few prerequisites to keep in mind that every solution should satisfy -

- Image reads should be quick and reliable.
- Image writes should be quick(not as quick as read op is acceptable, since number of image writes will be fewer in number than reads) and error free.
- Image security
- Consistency

One of the most efficient way of storing these small file on a hard disk drive is to use a distributed filesystem.
In simpler words, since we have an unlimited size hard disk the best way to use this space is to divide this hard disk into multiple partitions. Why Multiple Partitions ?
Let's take a situation where we have a 400GB storage. Since, we need a 100GB storage for storing 1mn images, wasting rest of the 300GB is not a good practice. So, make four partitions each of 100GB. Now, we can store these images in a distributed way. Use 2 partititons for storing the image and another 2 partitions as a replica . So, in other words each image will have two copies, one is the oiriginal and other is its replica which helps in case if one partitions crashes.
Let's say I have 4 Partitions named P1, P2, P3, P4.
We can store the image on P1 or P2 based on availability of the partitions and its replica in P3 or P4 respectively.
There can be many ways of distributing the image and its replica among 4 partitions. One of the better ways could be to -

Defragment the contents of an image and then store these fragments on P1 and P2 either alternatively or in any other pattern and maintain the mapping of inodes in a DHT for further references. Same could be done for the replica.

So let's check once if all the prerequisites as discussed above is satisfied by this solution

1. **Image reads should be quick and reliable -** Since we have two copies of every image, that means if one of the partition(P1 or P2) is crashed or is busy we can still access the image using(P3 or P4). Using a distributed method of storage will make image reads faster as we can simultaneously read two partitions and get the whole image contents.
2. **Image writes should be quick and reliable -** Writing an image on two partitions one for original image and one for replica, makes write op slower than reads but since the no. of writes are lesser in number than reads, we can consider this approach. Again, using a distributed method will make it much faster as multiple writes can be done on diff. partitions simultaneously.
3. **Image Security -** Image security can be ensured using Filesystem-level encryption. Filesystem-Level encryption includes -
    a. Each file to be encrypted usign a different key. These keys can be mapped into a DB or DHT
    b. Access control using public-key cryptography.
4. **Consistency -** Consistency levels can be kept in check by regular checkup and self-heal of partitions by checking the replica images and the original images.

PROBLEM 1B

Since there are close to 1mn images, therefore we need a naming convention where we don't need to traverse each and every image and check if that image is the desired one. In order to do that we need to somehow code the path in a way that it can be easily decoded from the filename. So, inorder to do that we can divide the filename in two groups of 4. For eg -
If the filename is 23321123.jpg then we can divide its file path like - 2332/1123/23321123.jpg
This method will help us in decoding the file path without searching for it and thus saves a lot of time and computation power. Also, since in the last question I had suggested a solution involving distributing image into two partitions(P1 and P2) and its replica into another partitions(P3 and P4). So if the unit's place digit of the image is even then store the original image in P1 and its replica in P3 . If the unit's place of image name is odd then store it in P2 and its replica in P4.

Another method could be to break the filename into 4 sections each of size 2 instead of two sections of size 4 each. Like- 23321123.jpg will be stored in path 23/32/11/23/23321123.jpg. But this method will make a filesystem(especially if filesystem is ext4) slower in performance as the number of directories will increase in depth by significant amounts.
Instead, using the previous convention as 2332/1123/23321123.jpg will affect the directory width not depth which has less effect on a filesystem.

One of the most efficient improvements to above method is to store the inode numbers corresponding to the filename in a NoSql DB.

Since the platform takes 4byte to store the integers therefore, the maximum size needed to store the file path is 2+2+4 = 8bytes and for 1mn images the total size becomes 8bytes * 1mn = 8MB which is not a huge size to handle.