Questions for FFS:

1. *What are the pros and cons of using large blocks.*

   + .More data in each file can be represented by direct data pointers in the i-node;

   + More sequential read, less seek/rotational delay, higher throughput;

   - waste disk space, especially when most files are small; .

2. *Why does LFS have better writer performance than FFS?*

   LFS does not update a block in its original location in disk; instead, all updates are organized and buffered in a segment, and then written to disk one segment at a time. Since LFS has turned random writes to sequential writes, its write-access performance is much better than FFS.

Questions for AFS/NFS:

1. *Briefly explain how caching is used to improve the performance of NFS (or AFS).*

   NFS caches data blocks to improve performance. Before using a data block, it will check whether this data block is still up to date. NFS uses time-out mechanism and attribute-block checking to decide whether to use the cached data blocks.

   AFS caches the whole file. After the first open, later file accesses will get local-disk access throughput. In later versions of AFS, call-back mechanism is used to determine whether a file needs to be discarded and fetched again from the server.

Questions for GFS:

1. *What did GFS do to make sure the master node does not become a performance bottle-neck?*

   1) Data transfer during read and write requests does not go through the master node. Instead, the client only contacts the master node to fetch the chunk-server information. After that, the client directly contacts the chunkservers to conduct read and write.

   2) The client will cache some chunkserver information, and does not always ask master for the chunk-server location.

Questions for RAID:

2. *Compare the performance and fault tolerance capability between RAID 3 and RAID 5.*

   RAID3 uses bit/byte level parity (each data block spreads across all data disks)

   RAID5 uses rotational parity

   Performance:

   RAID3 and RAID5 have the same amount of sequential read/write throughput.

   RAID5 has better random read throughput. Reason 1: RAID5 only needs to read one disk to obtain the target data block. In RAID3, all data disks need to be read in order to obtain one data block. Reason 2: RAID5 does not have a dedicated parity disk. Therefore, all disks in the system can serve random read requests.

   RAID5 has better random write throughput, Reason 1: RAID3 uses a dedicated parity disk. Every random write will require the system to read and write this parity disk. As a result, no parallelism can be achieved within the disk array. RAID5 can conduct G/2 random writes at the same time, because there is no dedicated parity disk in the system (G is the total number of disks in RAID5). 2. Since RAID3 spreads the data of each block across disks, there is an extra `synchronization’ overhead for RAID3.

   Reliability: Both RAID3 and RAID5 can survive one disk failure, no more.

Questions for MapReduce:

1. *Discuss how Map-Reduce systems handle node failures..*

   Master periodically pings every worker node to detect worker failures.

   Once a worker fails, all the finished or on-going mapper tasks, and all on-going reduce tasks running on it will be re-executed on another node. Reducer tasks that are supposed to read data from the re-executed mappers are notified. If the reducer task hasn’t finished reading the data, it will instead read data from the new map-worker.

   Finished reduce tasks do not need to be re-executed.

   Finally, if the master fails, the job will be re-executed.