Massachusetts Institute of Technology
Depart of Electrical Engineering and Computer Science
6.826 Principles of Computer Systems

PROBLEM SET 2

Assigned: February 15, 2006 Due: February 22, 2006

Problem 1: Read-ahead disk (40 points)

When data is accessed in a regular pattern, reading ahead can mask read latencies by caching data before it is even requested. Specifically, when data from disk address da is requested, one can cache data from disk addresses $da + 1, da + 2, \ldots, da + N$ in the hope that these data blocks will be read soon afterwards. The implementation of `BufferedDisk` in the notes points out that it could also make a provision for read-ahead, but doesn't.

a) (10 points) Write code for a `ReadAheadDisk`. Like the `BufferedDisk` in the notes, it should handle read requests by looking for as much of the requested data in cache before reading from disk. When doing a disk read, however, the usual behavior is to read $N = 50$ more disk blocks than requested. If the read-ahead runs into data that's already in cache, then the read-ahead is not performed, even if fewer than N disk blocks have been read ahead (in this case, we wouldn't seem to be reading data sequentially anyway). Only reads need be buffered, not writes. The cache should be cleared after each write.

```
CLASS ReadAheadDisk
       EXPORT Byte, Data, DA, E, DBSize, read, write, size,
              check, Crash = ...
```

b) (10 points) Write the abstraction function mapping `ReadAheadDisk` to `Disk`.

c) (20 points) Prove that your code implements the spec of `Disk`.

Problem 2: Striped disk array (60 points)

RAID (Redundant Array of Inexpensive Disks) Level 4 is a technique that enhances the reliability of disks by distributing data across drives (at the sector level), while writing parity information to a separate drive. If one drive in the array fails, the data on that drive can be reconstructed from the data on the other $N - 1$ drives in the array. For this problem, fix a physical sector size p shared by the drives in the array; we will say that the logical sector size (as reflected by `DBSize`) is $(N - 1)p$.

Here is a concrete example. Consider an array of 3 disks using a 1024-byte logical sector size and a 512-byte physical sector size. Logical sector 4, then, would store bytes 4096 through 5119 of the disk data; bytes 4096 through 4607 would be stored on bytes 2048 through 2559 of disk 0 (conveniently, physical sector 4), bytes 4608 through 5119 would be on disk 1 (also physical sector 4), and disk 2 would always store, at byte i, the value $d_2(i) = d_0(i)$ `XOR` $d_1(i)$.

RAID 4 relies on the underlying disks to report errors when appropriate; once an error is reported, it corrects the error using the stored parity information. Thus a read operation, in the normal case, simply needs to collect the information from all relevant drives and return it; in the abnormal case, it needs to read from all drives and return an XOR. Since we only write entire logical sectors at a time, we can always just compute the parity of the whole logical sector as we are writing it out; if we permitted partial-sector writes, then we would have to recalculate the change in parity due to the write.

You may do writes in-order, and assume that no crashes occur (base your code on the implementation of `Disk` on pages 3-4). Also, `Spec` provides an XOR operator on bytes, called `XOR`.

Note that the spec for a RAID array is the same as the spec for an individual drive, except that the error rate is reduced, and the performance is increased; these considerations are not part of our spec at the moment.

a) (15 points) Write code for `RAID4DiskImpl`.

b) (15 points) Write down the abstraction function mapping `RAID4DiskImpl` to `Disk`.

c) (30 points) Prove that your code implements the spec of `Disk`.