Final Submission Report

for

Database Internals - Disk Arm Simulator

Prepared by:

Prateek Chandan - 120050042
Anurag Shirolkar - 120050003

CSE , 3rd Year
IIT Bombay

Under Prof. N.L. Sarda
Department of Computer Science
IIT Bombay

Date: 23rd November , 2014

Table of content

I. Introduction

AIM

To Create a disk storage simulator with suitable access arm movement strategy, a cache on the disk, and gathering of statistics that will allow us to evaluate performance.

Key Features

1. To add disk storage simulator to the **paged file module**
2. The simulator will simulate pages for 4 disks , where disks will have 3 platter , 100 tracks per platter (100 to keep disk size small) and no of sectors varying from 101 in innermost sector to 200 in outermost sector
3. Also it will simulate a RAID 1 Disk Technology
4. A suitable arm movement strategy
5. A cache on the disk of size 1000 pages
6. Gathering of statistics that will allow to evaluate performance

II. Design

- The Design for disk arm simulator contains data structures for all the parts present in the hard disk and disk controller as classes and the internals of those parts like spindles , caches etc as the members of those classes. Also some helper structs like Statistics and Buffer
- The main program contains only interface through the functions provided by the disc controller which will check from the cache if the data is present in it or it will read/write the data from the hard disks and will also maintain proper statistics
- The disk controller also provides functionality to report and log all the stats of the read/write operations

III. Data structures

Following is the list and description of data structures that is used in the project

1. Disk Controller

The data structure for simulation of complete hard drive. This contains an array of disks (4+4 disks) which can be read or written. The RAID 1 Technology Is implemented in this hard disk class

This class just takes in the input to read and write data into the hard disk and a timer

The disk controller also has a **cache** in it.

2. HardDisk

The Hard disk class has array of platters . To keep the design realistic we have kept 3 platters in each hard disk where read / write can occur on both of the surfaces of the hard disk.

The hard disk will also has rotation and arm movement variable which will simulate the rotation of the disk and arm movement of the disk maintain the seek time and write latency

Along with multiple parameter to store statistics of time taken for all the read write request to the hard disk

3. Platter

Each Platter class has a list of tracks. The no of tracks is 100 to keep the size of disk to be small such that it can be simulated.

4. Track

Each track has a list of sectors. The sector size varies depending the track is on outer cylinder of the disk or the inner cylinder. The sector size varies from 101 on inner track to 200 on outer track

5. Sector

The Sector is the smallest segment which can be read or written. This class stores data in it and provides the read write functionality to the Data. To implement this reading and writing efficiently we have used a character array of 512 bytes. PF Layer was supposed to be used here but is not used as it will involve actual seek time and latency of hard disk which will make the simulation time very high. The size of data which is written will be 512 bytes

6. Cache

This class acts as cache to the hard disk. It will has fixed memory size which will store the address of memory and data referencing to that memory. The no of entries stored in a cache is less say 1000. And it is fully set associative cache to ensure that the miss penalty is least as reading from disk is very slow

7. Statistics

This Class is used to store all the statistics , reset and print them

IV. Algorithms

Disk Controller

In the disk controller we have used mainly demultiplexing from logical address to hard disk no , sector no , platter no ad also the cache. The following briefly explains the algorithm:

Least Frequently Used Replacement Policy in Cache

In cache we have kept an array of 1000*size_of_page and **least frequently used replacement policy** is used in cache

Least Frequently Used (**LFU**) is a type of cache algorithm used to manage memory within a computer. The standard characteristics of this method involve the system keeping track of the number of times a block is referenced in memory. When the cache is full and requires more room the system will purge the item with the lowest reference frequency.

Demultiplexing from Logical Address to CHS Address

User Provides a logical address to the disk controller which can vary from 1 - size for all hard disk. A demultiplexing is used at hard disk controller and then **CHS (Cylinder , Head , Sector)** address is passed to the hard disk. The demultiplexing is done as follows :

- Disk no = Address / Size of Disk
- Platter no = (Address % Size of Disk) / Size of Platter
- Track no = ((Address % Size of Disk) % Size of Platter) / size of track
- Sector no = ((Address % Size of Disk) % Size of Platter) % size of track

Hard Disk

Elevator Algorithm

The **elevator algorithm** (also **SCAN**) is a disk scheduling algorithm to determine the motion of the disk's arm and head in servicing read and write requests

When a new request arrives while the drive is idle, the initial arm/head movement will be in the direction of the cylinder where the data is stored, either *in* or *out*. As additional requests arrive, requests are serviced only in the current direction of arm movement until the arm reaches the edge of the disk. When this happens, the direction of the arm reverses, and the requests that were remaining in the opposite direction are serviced, and so on.
The buffer is implemented using set in C++. when a new request arrives, it is inserted into the buffer in O(log(buffer_size)). The give_next function in the buffer object takes a direction as an argument and returns the next track in the given direction.
This elevator algorithm is used at hard disk to do the read and write works

Buffer and Timers

To Implement the elevator algorithm , each of the hard disk contains a its own buffer and a timer variable is shared with it by the controller. Upon read / write instruction the operation is simply added into the buffer and on each clock cycle one operation from buffer is operated using the elevator technology

Statistics

All the time taken in the the write latency at sector and seek time in the platter is stored efficiently in the statistics class. This also has functionality to reset the data.

V. Test Cases

Elevator Algorithm

(This is carried out by keeping the cache off)

Input :

operation	platter no.	track no.	sector no.
write	0	0	1
write	1	1	1
read	0	0	1
read	1	1	1

Actual execution sequece (when cache is turned off):

operation	platter no.	track no.	sector no.
write	0	0	1
read	0	0	1
write	1	1	1
read	1	1	1

Cache Effect

Input :

Output :

WITHOUT CACHE	**WITH CACHE**
STATS FOR DISK 1 : no of seeks : 2 no of block transfers : 2 no of reads : 0 no of writes : 2 total time : 16 STATS FOR MIRRORED DISK 1 : no of seeks : 2 no of block transfers : 2 **no of reads : 4** no of writes : 2 total time : 16	STATS FOR DISK 1 : no of seeks : 2 no of block transfers : 2 no of reads : 0 no of writes : 2 total time : 16 STATS FOR DISK 1 : no of seeks : 2 no of block transfers : 2 **no of reads : 0** no of writes : 2 total time : 16

RAID1 effect

BEFORE	**AFTER**
49995 ========== STATS FOR DISK 1 : no of seeks : 1232 no of block transfers : 1234 no of reads : 0 no of writes : 1234 total time : 9864 STATS FOR DISK 2 : no of seeks : 1274 no of block transfers : 1275	49995 ========== STATS FOR DISK 1 : no of seeks : 1887 no of block transfers : 1234 no of reads : 657 no of writes : 1234 total time : 12484 STATS FOR DISK 2 : no of seeks : 1912 no of block transfers : 1275

no of reads : 0
no of writes : 1275
total time : 10196

STATS FOR DISK 3 :
no of seeks : 1251
no of block transfers : 1253
no of reads : 0
no of writes : 1253
total time : 10016

STATS FOR DISK 4 :
no of seeks : 1250
no of block transfers : 1251
no of reads : 0
no of writes : 1251
total time : 10004

STATS FOR MIRRORED DISK 1 :
no of seeks : 2468
no of block transfers : 1234
no of reads : 1240
no of writes : 1234
total time : 14808

STATS FOR MIRRORED DISK 2 :
no of seeks : 2565
no of block transfers : 1275
no of reads : 1298
no of writes : 1275
total time : 15360

STATS FOR MIRRORED DISK 3 :
no of seeks : 2445
no of block transfers : 1253
no of reads : 1196
no of writes : 1253
total time : 14792

STATS FOR MIRRORED DISK 4 :
no of seeks : 2474
no of block transfers : 1251

no of reads : 641
no of writes : 1275
total time : 12748

STATS FOR DISK 3 :
no of seeks : 1840
no of block transfers : 1253
no of reads : 590
no of writes : 1253
total time : 12372

STATS FOR DISK 4 :
no of seeks : 1864
no of block transfers : 1251
no of reads : 618
no of writes : 1251
total time : 12460

STATS FOR MIRRORED DISK 1 :
no of seeks : 1813
no of block transfers : 1234
no of reads : 583
no of writes : 1234
total time : 12188

STATS FOR MIRRORED DISK 2 :
no of seeks : 1928
no of block transfers : 1275
no of reads : 657
no of writes : 1275
total time : 12812

STATS FOR MIRRORED DISK 3 :
no of seeks : 1857
no of block transfers : 1253
no of reads : 606
no of writes : 1253
total time : 12440

STATS FOR MIRRORED DISK 4 :
no of seeks : 1860
no of block transfers : 1251

no of reads : 1228 no of writes : 1251 total time : 14900	no of reads : 610 no of writes : 1251 total time : 12444

VI. Conclusion

In the completion of this Project we have learnt following :

- The Implementation of hard disk technology
- RAID1 implementation
- Effect of Cache in improvement of performance
- Effect of Elevator Algorithm to improve performace
- Paging in hard disk