COMPSCI 590S Systems for Data Science **Fall 2016**

Lecture 7:

Lecturer: Emery Berger *Scribe(s): Lurdh Pradeep Reddy Ambati, Parth Gandhi*

7.1 Admin

Next project/assignment requires coding web sockets in Java.

7.2 Review of row-based databases v/s column based databases

- Row-based Databases:In a row-based database, rows are stored across the files in a distributed system or in a single file other wise. Advantages of this is adding a new row requires hashing(and searching) and then appending/inserting new row is faster as compared in a column-based database(of similar schema and size).

- Column-based Databases: In a column-based data base, columns of database are stored as different files. So, when a new row needs to be added, (hashing and searching are required here as well) multiple I/O's are required as each field in row is added to the respective column, increasing the disk seek.

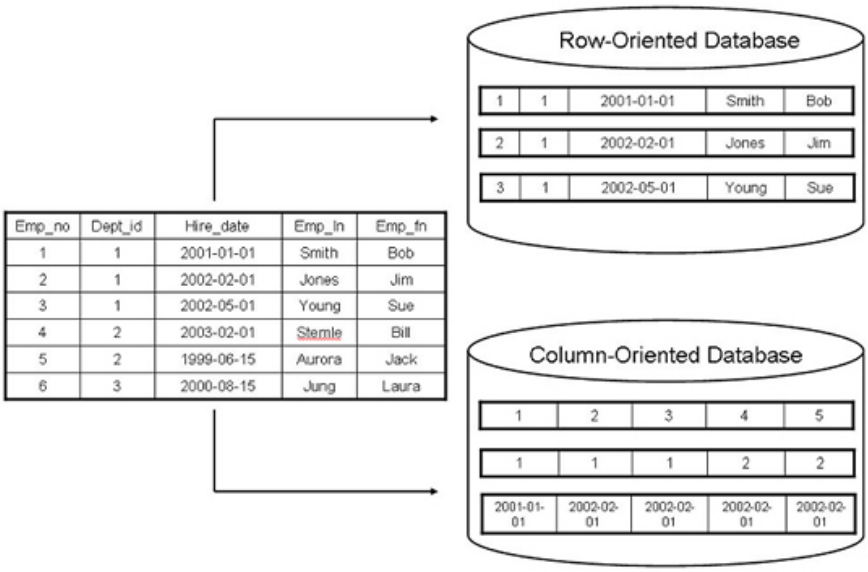

Figure 7.1: Row-based v/s Column-based data bases

7.3 Seek latency v/s Rotational latency

- Seek latency defines the amount of time it takes a hard drives read/write head to find the physical location of a piece of data on the disk.
- Rotational latency is the delay waiting for the rotation of the disk to bring the required disk sector under the read-write head.

  To reduce seek latency, temporal locality should be exploited. In general, seek latency is way higher than rotational latency.

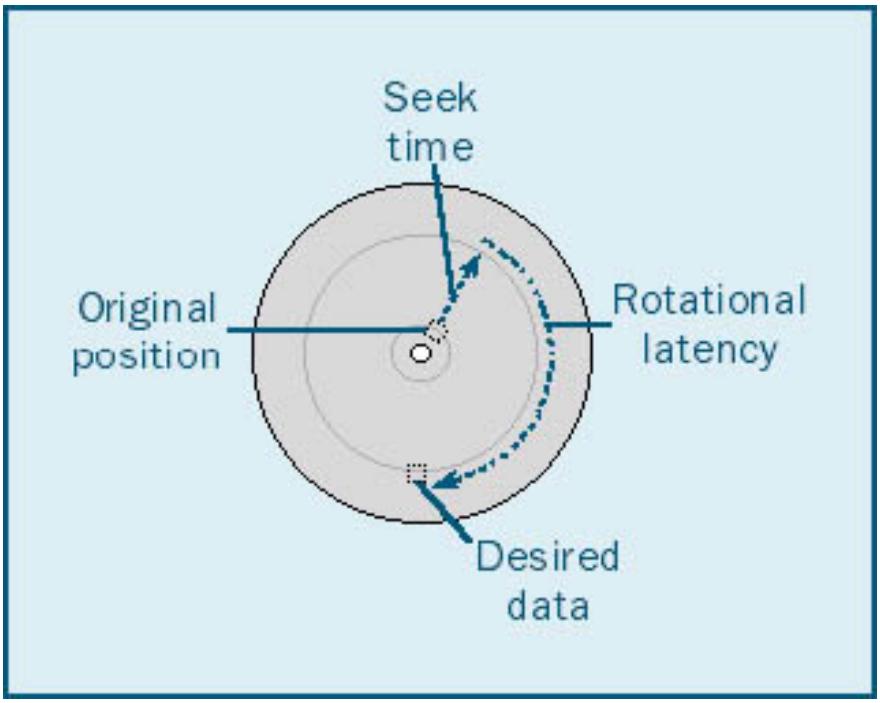

Figure 7.2: Seek latency v/s Rotational latency

7.4 Disk Fragmentation

Part of the files in a disk are distributed/spread across the disk (Non contiguous memory). This mechanism is referred to Disk fragmentation.

7.5 Memory aligned loads and stores

In aligned loads/stores, when a variable/address is loaded or stored the memory address must be a multiple of power of 2 bytes for example 8 or 16 etc. Now, if a variable/address is 13 bytes, and assume that each memory block/chunk is of size - 4 bytes, then the respective variable takes 4 blocks and in the process 3 bytes are wasted and not used for other memory allocations.

7.6 Internal fragmentation and External fragmentation

- Internal fragmentation : In the previous example, (variable of 13 bytes taking 4 blocks, each block of 4 bytes) 3 bytes are wasted, in this scenario, the unusable memory is within an allocated region. This inefficient usage of memory is referred to **internal fragmentation**
- External fragmentation: Total memory space is enough to allocate memory for a requested variable, but the available memory is not contiguous because of this memory allocation cannot be completed. This is referred to **external fragmentation**
- Compaction saves/extract the free spaces between aligned memory allocation. This feature is available in Java(JVM) and is because of its garbage collection, which collects the unused memory periodically.

7.7 Metadata overhead(book keeping) in Linux

In Linux, initial bits of a memory allocation for a pointer stores the size of the pointer/variable(metadata of the pointer). This metadata is the overhead incurred in linux for storing the pointer.

7.8 Memory allocation in Java v/s C/C++

- In Java, memory for a object is allocated in scattered manner. For example, if a object A has 4 int variables, when instance of object A is allocated 4 variables are stored in non contiguous memory. This results in non temporal locality, and inefficient pointer access.

  Note, in Java every object has a lock.

  In C++, variables of a class object are allocated in contiguous fashion, this results in better temporal locality, direct pointer access and control of object layout is possible.

7.9 MapReduce v/s Hadoop

- MapReduce is built on GFS file system and is implemented in C++, which translates to better performance compared to Hadoop. MapReduce is a closed source system and is developed by Google.
- Hadoop is built on HDFS file system and is implemented in Java. It is a open source system and is developed by Yahoo!.

Java is not as efficient as C++, when performance is concerned as object layout can be controlled in C++ for faster object access and garbage collection of java constraints the performance.

7.10 Garbage collection in Java

Young generation: Most of the newly created objects are located here. Since most objects soon become unreachable, many objects are created in the young generation, then disappear.

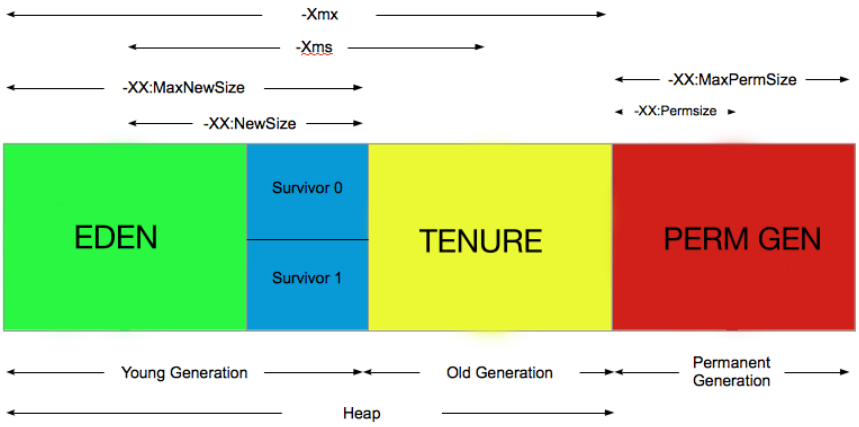

Figure 7.3: JVM Heap

Old generation: The objects that did not become unreachable and survived from the young generation are copied here. It is generally larger than the young generation. As it is bigger in size, the GC occurs less frequently than in the young generation.

Garbage collection Tuning: The purpose of GC tuning is to minimize the number of objects passed to the old area and to decrease Full GC execution time. GC tuning is required for the optimized application performance, as GC can take up high CPU resources which increases the running times of the applications.