Q) Interactive SQL

By dispensing with MapReduce and using a distributed query engine that uses

dedicated “always on” daemons (like Impala) or container reuse (like Hive on Tez), it’s

possible to achieve low-latency responses for SQL queries on Hadoop while still scaling

up to large dataset sizes.

Q) Stream processing

Streaming systems like Storm, Spark Streaming, or Samza make it possible to run realtime,

distributed computations on unbounded streams of data and emit results to

Hadoop storage or external systems.

Ans:

Streaming: <http://www.webopedia.com/TERM/S/streaming.html>

Buffer : <http://www.webopedia.com/TERM/B/buffer.html>

Spooling : <http://www.webopedia.com/TERM/S/spooling.html>

Cache: <http://www.webopedia.com/TERM/C/cache.html>

About Real and Virtual Memory : <http://www.pcguide.com/ref/ram/sizeVirtual-c.html>

About Hard Disk, Platters, tracks, sectors, cylinder, read/write heads

About Disk Structures: Track, Track Sector, Geometrical Sector and Cluster/Allocation Unit

Q) Search

The Solr search platform can run on a Hadoop cluster, indexing documents as they are

added to HDFS, and serving search queries from indexes stored in HDFS.

Q) Why can’t we use databases with lots of disks to do large-scale analysis? Why is Hadoop

needed?

The answer to these questions comes from another trend in disk drives: seek time is

improving more slowly than transfer rate. Seeking is the process of moving the disk’s head

to a particular place on the disk to read or write data. It characterizes the latency of a disk

operation, whereas the transfer rate corresponds to a disk’s bandwidth.

If the data access pattern is dominated by seeks, it will take longer to read or write large

portions of the dataset than streaming through it, which operates at the transfer rate. On

the other hand, for updating a small proportion of records in a database, a traditional BTree

(the data structure used in relational databases, which is limited by the rate at which

it can perform seeks) works well. For updating the majority of a database, a B-Tree is less

efficient than MapReduce, which uses Sort/Merge to rebuild the database.

Ans 🡪

<http://www.webopedia.com/TERM/D/data.html> : About Data

<http://www.webopedia.com/TERM/D/database.html> : About DataBase

<http://www.webopedia.com/TERM/H/hypertext.html> : About HyperText

<http://www.webopedia.com/TERM/D/data_structure.html> : About Data Structure

<http://www.webopedia.com/TERM/T/tree_structure.html> : About Tree Data Structure

Here I need more sense from above statements

Q) Normalization poses problems for Hadoop processing because it makes reading a record a

nonlocal operation, and one of the central assumptions that Hadoop makes is that it is

possible to perform (high-speed) streaming reads and writes.

Q) since the output from mappers is fed to the reducers, but this is

under the control of the MapReduce system in this case, it needs to take more care

rerunning a failed reducer than rerunning a failed map, because it has to make sure it can

retrieve the necessary map outputs and, if not, regenerate them by running the relevant

maps again.

Q) The SETI@home problem is very CPU-intensive, which makes it suitable for

running on hundreds of thousands of computers across the world[9] because the time to

transfer the work unit is dwarfed by the time to run the computation on it. Volunteers are

donating CPU cycles, not bandwidth.

Ans:

Data Packets or Network Packet : <https://en.wikipedia.org/wiki/Network_packet>

Throughput : <https://en.wikipedia.org/wiki/Throughput>

Scheduling : <https://en.wikipedia.org/wiki/Scheduling_(computing)>

Thread: <https://en.wikipedia.org/wiki/Thread_(computing)>

Concurrent Computing: <https://en.wikipedia.org/wiki/Concurrent_computing>

User Space/User Land or Kernel Space : <https://en.wikipedia.org/wiki/User_space#cite_note-2>

Popek and Goldberg virtualization requirements : <https://en.wikipedia.org/wiki/Popek_and_Goldberg_virtualization_requirements>

CPU Time: <https://en.wikipedia.org/wiki/CPU_time> and <http://www.webopedia.com/TERM/C/CPU_time.html>

About “Infinite loop”, Thrashing, deadlock and Access Violations/Segmentation Fault

Watchdog timers: <https://evothings.com/watchdog-timers-how-to-reduce-power-usage-in-your-arduino-projects/>

Q) What is parity?

<http://techterms.com/definition/parity>

<http://searchstorage.techtarget.com/definition/parity>

Q) What is Data Access Object Pattern?

<http://www.informit.com/guides/content.aspx?g=java&seqNum=138>

<http://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm>

Q) About Lisp

Q) About Non-Uniform Memory Access i.e. NUMA

Q) About Deterministic and Non Deterministic functions

<https://msdn.microsoft.com/en-us/library/ms178091.aspx>

Q) About “reader” interface that provides implementation about the reading a different input data types in MapReduce.

Q) About Data Structure mapped in memory or Memory mapped IO

<http://lbrandy.com/blog/2010/11/memory-mapped-io-for-fun-and-profit/>

Q) About Atomic Commit and Two-Phase Atomic commit

<http://docs.mongodb.org/master/tutorial/perform-two-phase-commits/>

Q) Segmentation Violation and Bus Errors

Q) About GNU and gdb tool

Q) About Profiling / Data Profiling

Q) Working of Domain Name System (DNS)

Q) About Search Engine Algorithms

Q) What is Hashmap data structure?

Q) What is Inverted Indexing?

<http://www.quora.com/Information-Retrieval/What-is-inverted-index>

Q)