Question 1 (2p) Give and explain 4 V's (big data properties) and give an example for each
Volume: size of the data e.g. xxx processes 600TB per day
Variety: type and nature of the data e.g. unstructered, semi-structured
Velocity: speed of generation, proeceesing of data e.g. stock market social networks
Veracity: uncertainty of the data e.g. fake form fillouts

Question 2 (2p) (a) Define the notions of read scalability and write scalability. (1p) (b) Describe an example use case / application for which read scalability is important but write scalability is not. (1p)
Read scalability: a system can handle an increasing number of read operations without losing performance

Write scalability a system can handle an increasing number of read operations without losing performance

A streaming server is an example of where read scalability is more important than write scalability. Normally content is written once but accessed by millions.

Question 3 (3p) Consider the following relational database which consists of two relations (Project and Report). Notice that the attribute finalreport in the relation Project is a foreign key that references the primary key (attribute id) in the relation Report. Notice also that multiple projects may have the same final report. Project Report name budget finalreport UsMis 1,000,000 391 AMee3 3,700,000 391 Bee 1,300,000 121 id pages location 121 70 http://acme.com/beerep 391 350 http://acme.com/r391 699 100 http://acme.com/Other Capture all the data in this relational database as (a) a key-value database, (1p) (b) a document database, (1p) (c) a graph database (using the Property Graph model). (1p)

See svar-tdde31-2019-05-29

Question 4 (2p) Describe i) the types of queries and ii) the form of data partitioning implemented in key-value stores (1p), and iii) explain how these things are related to achieving horizontal scalability (1p).
i) Value related queries are not possible since the values are opaque to the system (no secondary index over values)
only CRUD operations (create, retrieve, update, delete) regarding keys are possible
multiple queries are necessary for retrieving multiple entries
ii) horizontal partitioning/sharding -> partitioning based on keys
iii) The limitations of this data model make it very easy to partition the data and distribute the partitions over multiple nodes in a cluster. Horizontal scalability refers to exactly that the adding of resources by just adding more nodes to a distributed system (scale out). Key value pairs are completely independent entities and there is no referencing logic between entities. Therefore partitioning based on keys is very efficient and partitions can be distributed without any problems.

Question 5 (1p) Assume a distributed database system in which every data item is stored on 3 nodes. For such a system to report to an application that a write operation as been finished, what is the number of nodes that are required to complete the write successfully if the system aims to achieve the consistency property as per the CAP theorem.
Stored on 3 nodes -> N=3
Consistency per CAP requires W=N -> W=3

Question 6 (3p) Cluster computing (a) How does a distributed file (in a distributed file system like HDFS) differ from a traditional file, and what is the advantage for the processing of a big-data computations over a distributed file compared to a traditional file? (1p) (b) Describe how modern hybrid clusters used for distributed parallel big-data processing are organized. In particular, specify and explain their control structure and memory structure. (1.5p) (c) Why is it important to consider (operand) data locality when scheduling tasks (e.g., mapper tasks of a MapReduce program) to the nodes a cluster? (0.5p)

a) A distiributed file is split into blocks eg 64 MB and the blocks are distributed over different nodes on a cluster. This allows the parallel processing of the file, since accessing the file can now be done in parallel. A traditional file completely sits on one computer and can only be accessed by multiple computers sequentially.

Question 7 (7p) MapReduce and Spark (a) What (mathematical) properties do functions need to fulfill that are to be used in Combine or Reduce steps of MapReduce, and why? (1p) (b) Which substeps of the MapReduce construct involve disk I/O, and for what purpose? (1p) (c) Why and in what situations can it be beneficial for performance to use a Combiner in a MapReduce instance? (1p) (d) What is a RDD in Spark? Be thorough! (1p) (e) Spark classifies its functions on RDDs into two main categories: ”Transformations” and ”Actions”. Describe the main difference between those, and give one example operation for each category. (1p) (f) How can Spark be used with input data that arrives in a continuous stream (e.g., from external sensors over the network)? In particular, how can such a data stream be structured for processing by Spark? (1p) (g) Describe the execution model of Spark programs. In particular, there exist 2 different kinds of processes, driver and workers. Explain in general which operations of a Spark program are executed by each of them. (1p)

b) record reader, partitioner, shuffle-and-sort, output formatter

d) an RDD(resillient distributed dataset) is a fault-tolerant collection/container of data elements that can be operated on in parallel. It is by default lazy, hence it only materializes on demand. The construction of an RDD and its blocks is described by a data flowgraph. The blocks are not replicated, in case of a faile the block is simply recalculated based on the data flowgraph. RDDs are read-only after construction, partitioned and distributed.

e) Transformation: returns new RDD from RDD
Action: returns a value from an RDD
f) Spark has an extension for processing data streams. It uses a high-level obstraction of a continous data stream called Dstream or discritzied stream. Essentialy it is a continous series of RDDs allowing the operations on RDDs like normal. + windowing
g) Driver creates mapper and reducer tasks, dispatches them to workers -> dynamic load balancing

Workers: executes mapper/reducers tasks for driver, and returns results to driver. The client contacts the driver, the driver splits the file, creates mapper and reducers tasks and dispatches them to workers. The workers return their results or tell the driver where the RDDs reside.

2019-08-22

Describe one of the many new challenges for database systems that NoSQL systems aim to address:
Because we store a lot of information that is being read by lots of people at the same time but it's not critical that the data is 100% correct. More important that it can be accessed by tons of people.

While read scalability can be achieved by scaling horizontally(scale out), it cannot be achieved by scaling vertically(scale up). T/F?
F, my be used to cache more of the read data, which may help to handle more reads. Also, replacing the harddrive of a server with a faster harddrive is another form of scaling up.

Concrete application or use case for which data scalability is important:
A video streaming site, such as youtube., Needs to be able to store a lot of information that is going to be read a lot. Solved with large content distribution networks.

2019-11-01

When is data scalability needed? Bank system for processing many transactions at the same time. e.g black friday, christmass etc

See question in TDDE31_0219-11-01
3) a)
The typically implemented queries uses keys to retrieve value. So we would have to loop through the keys until we found the one we where searching for.
b) This would be done by using Alice as key add a value with IDs that has Alice as name.
“Alice” -> “, , [alice.in.se]”

c) In a document database the information about each of these persons could be its own document. In a key database only the key has to be unique but in a document database the name describing the value all has to be unique within a document.