732A54 Big Data Analytics

Help for written exam - Examples of exam question

Arian Barakat

2017-01-11

---

1. Databases for Big Data

1.1 NoSQL data stores and techniques

1. **Question:**
   Explain the main reasons for why NoSQL data stores appeared.

Answer:
There are several reasons why NoSQL appeared (mainly due to 'One Size does not fit all) and these are:

- Frequent schema changes, management of unstructured and semi-structured data
- Huge dataset - data volume increase exponentially
- Different applications have different requirements - new needs

---

2. **Question:**
   List and describe the main characteristics of NoSQL data stores.

Answer:

- Simple and flexible non-relational data models
- High availability and relax data consistency (CAP theorem. **Link to CAP**)
  - BASE vs. ACID
- Easy to distribute - horizontal scalability
- Data are replicated to multiple nodes
  - Down nodes easily replaced
  - No single point of failure
- Cheap and Easy (or not) to implement (opens source)

---

3. **Question:**
   Explain the difference between ACID and BASE properties.

Answer:

BASE:

- Basic Availability: An application works basically all the time (despite partial failures)
- Soft-state: Is in flux and non-deterministic (changes all the time)
- Eventual consistency: Will be in some consistent state (at some time in future)

ACID:

- Atomic: Everything in a transaction succeeds or the entire transaction is rolled back.
- Consistent: A transaction cannot leave the database in an inconsistent state.

- Isolated: Transactions cannot interfere with each other.
- Durable: Completed transactions persist, even when servers restart etc.

In other words, the BASE properties are more relaxed than the ACID properties

Link: Basic differences

---

4. **Question:**
   Discuss the trade-off between consistency and availability in a distribute data store setting.

Answer:
By assuming that in a distribute data store setting there is no single point of failure we can be certain that a customer will be able to view and add to the shopping cart during various failure scenarios. However, due to the availability property, we can also assume some violation regarding the consistency property that there is a small probability that a transaction is not completely executed from beginning to end without interference from other transactions.

---

5. **Question:**
   Discuss different consistency models and why they are needed.

Answer:
There are several consistency models, some examples are:

- Strong Consistency
  - After the update completes, any subsequent access will return the updated value.
- Weak Consistency
  - The system does not guarantee that subsequent accesses will return the updated value. A number of conditions need to be met before the value will be returned. The period between the update and the moment when it is guaranteed that any observer will always see the updated value is dubbed the inconsistency window.
- Eventual Consistency
  - This is a specific form of weak consistency; the storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value. If no failures occur, the maximum size of the inconsistency window can be determined based on factors such as communication delays, the load on the system, and the number of replicas involved in the replication scheme. The most popular system that implements eventual consistency is the domain name system (DNS). Updates to a name are distributed according to a con gured pattern and in combination with time-controlled caches; eventually, all clients will see the update.
- Other variations of the eventual consistency model

These consistency models are needed due to the 'one size does not fit all' term. In other words, for some applications the consistency criteria must be relaxed in order to obtains high availability.

Source:

---

6. **Question:**
   Explain how consistency between replicas is achieved in a distributed data store.

Answer:

---

7. **Question:**
   Explain the CAP theorem.

Answer:
The CAP theorem states that a distributed computer system can only have 2 out of the following three following properties:

- Consistency
- Availability
- Partition tolerance

---

8. **Question:**
   Explain the differences between vertical and horizontal scalability.

Answer:

Overall Definition of Scalability:

- (Improvements of a) System in order to handle the growing amounts of data without losing performance

Vertical Scalability (scale up):

- Adding resources (more CPUs, more memory) to a single node
- Using more threads to handle a local problem

Horizontal Scalability (scale out):

- Adding nodes (more computers, servers) to a distributed system - low costs for commodity hardware
- Often surpasses scalability of vertical approach

In other words, vertical scalability is increasing a system's capacity by increasing the power of one computer, while horizontal scalability is increasing a system's capacity by increasing by increasing the number of computers (and not necessary the power of those computers)

---

9. **Question:**
   Explain how consistent hashing works and what are the problems it addresses.

Answers:
The problem that consistent hashing tries to address is to minimize the number of nodes to be copied after a configuration change. This is done by incorporate hardware characteristics into hashing model and arranging the nodes in a ring and making sure that each node is in charge of the hash values in the range between its neighbor node. If a node is dropped or gets lost, missing data is redistributed to adjacent nodes. However, if a node is added, its hash value is added to the hash table and the hash realm is repartitioned, and hash data will be transferred to new neighbor.

---

10. **Question:**
    Explain how vector clocks work and what are the problems they address.

Answer:
The problem the vector clocks tries to address is to recognize order of distributed events and potential conflicts and this is done by assigning a vector clock to each process. Processes piggyback vector time-stamps on the messages they send to one another and every time a process occur, the process's own vector clock increments with one.

---

11. **Question:**
    List and describe dimensions that can be used to classify NoSQL data stores.

Answer:
There are numerous dimensions you could use to classify a NoSQL data store and a handful are:

- Data model
  - How data is stored - key/value stores, column-oriented, document- oriented, etc.
- Storage model
  - How is it stored, in-memory or persistent?
- Consistency model
  - See question 5 above.
- Physical Model
  - How does the actual physical architecture look like? A single or distributed machine?
- Read/write performance
  -
- Secondary indexes
  - Does the data store support sorting and accessing of tables based on different fields and sorting orders
- Failure handling
  - How does the data store handle failures? Related to the *Partition Tolerance* part of the CAP theorem
- Compression
  - Does the data store have compression methods capabilities?
- Load balancing
  -
- Atomic read-modify-write
  -
- Locking, waits and deadlocks
  -

---

12. **Question:**
    List and describe the main characteristics and applications of NoSQL data stores according to their data models.

Answer:
There are several data models of NoSQL and these are: (1) Key-Value Stores, (2) Document Stores, (3) Column-Family Stores, (4) Graph Databases.

- Key-Value
  - Schema free
    - Keys are unique
    - Values of arbitrary types
  - Efficient in storing distributed data
  - Limited query facilities and indexing
  - Applications
    - Storing web session information
    - User profiles and configuration
    - Shopping cart data
- Document Stores
  - Schema free
    - Keys are unique
    - Values are documents (complex nested data structures)
  - Indexing and querying based on primary key and content
  - Applications
    - Items with similar nature but different structure
    - Blogging platforms
    - Content management systems
    - Event logging
- Column-Family Stores

  - Schema-free
    - Rows have unique keys
    - Values are varying column families and act as keys for the columns they hold
    - Columns consist of key-value pairs
  - Applications
    - Document stores applications
    - Analytic scenarios
      - Web analytics
      - Personalized search
      - Inbox search
- Graph Databases
  - Graph model
    - Nodes/vertices and links/edges
    - Properties consisting of key-value pairs
  - Applications
    - location-based services
    - Recommendation engines
    - Complex network-based applications
      - social-, information-, technological-, and biological network

---

1.2 HDFS

1. **Question:**
   Explain what HDFS is and for what types of applications it is (not) good for.

Answer:
HDFS stands for Hadoop Distributed File-System that handles the data storage on machines/nodes in a cluster and is a component of Apache Hadoop. HDFS is good at storing very large files and streaming accesses, which makes it not suitable for applications that require low-latency data access and the storage of many small files.

---

2. **Question:**
   Explain the organisation of HDFS.

Answer:

The HDFS organisation consist mainly of a master (namenode) and workers (datanodes), in some cases a secondary namenode is also a member of the organisation.

The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. This information is stored persistently on the local disk in the form of two files: the namespace image and the edit log. The namenode also knows the datanodes on which all the blocks for a given file are located, however, it does not store block locations persistently, since this information is reconstructed from datanodes when the system starts.

Datanodes are the workhorses of the filesystem. They store and retrieve blocks when they are told to, and they report back to the namenode periodically with lists of blocks that they are storing.

As mentioned, it is also possible to run a secondary namenode, which despite its name does not act as a namenode. Its main role is to periodically merge the namespace image with the edit log to prevent the edit log from becoming too large. The secondary namenode usually runs on a separate physical machine, since it

requires plenty of CPU and as much memory as the namenode to perform the merge. It keeps a copy of the merged namespace image, which can be used in the event of the namenode failing.

---

3. **Question:**
   Explain the process of reading and writing to HDFS.

Answer:
Source: Chapter 3 in Hadoop, The definitive Guide

File reads:

1. The client opens the the file it wishes to read
2. The DFS calls the namenode to determine the data blocks location on the workers. The workers (datanodes) are sorted with respect to the proximity to the client
3. The DFS returns an FSDataInputStream (an input stream that supports file seeks) to the client for it to read data from and the client calls a read() on the stream
4. The data blocks are read from the datanodes
5. When the data blocks have been retrieved, the client calls close() in the FSDataInputStream.

File writes:

1. The client creates a file by calling create() on the DFS
2. The DFS call the namenode to create a new file in the filesystem's namespace, with no blocks associated with it.
3. The DFS returns a FSDataOutputStream for the client to start writing data to. As the client writes data, DFSOutputStream splits it into packets, which it writes to an internal queue, called the data queue. The data queue is consumed by the Data Streamer, whose responsibility it is to ask the namenode to allocate new blocks by picking a list of suitable datanodes to store the replicas
4. The DataStreamer streams the packets to the first datanode in the pipeline, which stores the packet and forwards it to the second datanode in the pipeline. Similarly, the second datanode stores the packet and forwards it to the third (and last) datanode in the pipeline.
5. DFSOutputStream also maintains an internal queue of packets that are waiting to be acknowledged by datanodes, called the ack queue. A packet is removed from the ack queue only when it has been acknowledged by all the datanodes in the pipeline.
6. When the client has finished writing data, it calls close() on the stream.
7. A closing flushes all the remaining packets to the datanode pipeline and waits for acknowledgments before contacting the namenode to signal that the file is complete.

---

4. **Question:**
   Explain how high availability is achieved in HDFS.

Answer:
We know that the namenode is a single point of failure - if a namenode crashes, the cluster is donw. The secondary namenode is not a backup, it's only a checkpoint and it merges the namespace image with the edit log periodically to prevent the edit log from becoming too large. It lags the state of the primary prevents data loss but does not provide high availability and the time for cold start is approx. 30 minutes.

However, one could arrange pair of namenodes in an active stand-by configuration to achive high availbility by running two redundant NameNodes in the same cluster in an Active/Passive configuration with a hot standby. The configuration is as follows:

- Highly available shared storage for the shared edit log
- Datanodes send block reports to all namenodes
- Clients must provide transparent to the user mechanism to handle failover
- The standby node takes checkpoints of the active namenode namespace instead of the secondary node

---

1.3 Dynamo

1. **Question:**
   Explain the data model and list main applications of Dynamo.

Answer:
The dynamo (also known as Amazon dynamoDB) uses the Key-value data model and hold the Availability and Partition tolerance properties according to the CAP theorem. The main applications of dynamo are:

- Shopping cart Data

---

2. **Question:**
   Explain the Dynamo design considerations and what are the advantages of Dynamo in comparison to RDBMSs.

Answer:
The dynamo database is designed to provide an 'always-on' experiance for the user and in order to achive this high availability, it sacrifices consistency under certain failure scenarios. This mean that dynamo **never rejects writes**.

Other design considerations are:

- Incremental scalability
  - Dynamo should be able to scale out one storage host (henceforth, referred to as "node") at a time, with minimal impact on both operators of the system and the system itself.
- Symmetry
  - Every node in Dynamo should have the same set of responsibilities as its peers; there should be no distinguished node or nodes that take special roles or extra set of responsibilities. In our experience, symmetry simplifies the process of system provisioning and maintenance.
- Decentralization
  - An extension of symmetry, the design should favor decentralized peer-to-peer techniques over centralized control. In the past, centralized control has resulted in outages and the goal is to avoid it as much as possible. This leads to a simpler, more scalable, and more available system.
- Heterogeneity
  - The system needs to be able to exploit heterogeneity in the infrastructure it runs on. e.g. the work distribution must be proportional to the capabilities of the individual servers. This is essential in adding new nodes with higher capacity without having to upgrade all hosts at once.

According to the design considerations above, some advantages of dynamo over RDBMS are:

- No need of complex querying and managing functionalities since we only querying by key
- Scale out more easily than RDBMS

---

3. **Question:**
   Explain how basic NoSQL techniques are applied in Dynamo.

Answer:
Dynamo uses well know techniques to achive scalability and availability. Data is partitioned and replicated using consistent hashing, and consistency is facilitated by object versioning. The consistency among replicas during updates is maintained by a quorum-like technique and a decentralized replica synchronization protocol.

Dynamo employs a gossip based distributed failure detection and membership protocol. Dynamo is a completely decentralized system with minimal need for manual administration. Storage nodes can be added and removed from Dynamo without requiring any manual partitioning or redistribution.

---

4. **Question:**
   Explain versioning and semantic reconciliation in Dynamo.

Answer:
The dynamo uses vector clocks to handle versioning and asynchronous update propagation. Each update is a new immutable version, which can lead to that many versions of an object may exist. Most of the time, new versions subsume the previous version(s), and the system itself can determine the authoritative version (syntactic reconciliation). However, version branching may happen, in the presence of failures combined with concurrent updates, resulting in conflicting versions of an object. In these cases, the system cannot reconcile the multiple versions of the same object and the client must perform the reconciliation in order to collapse multiple branches of data evolution back into one (semantic reconciliation).

In other words, if the versions cannot be syntactically reconciled based on vector clocks alone, they have to be passed to the business logic for semantic reconciliation. Semantic reconciliation introduces additional load on services, so it is desirable to minimize the need for it.

1.4 HBase

1. **Question:**
   Explain the difference between column-oriented and row-oriented storage.

Answer:
The main difference between these storage layouts is that column-oriented databases saves their data grouped by columns. Subsequent column values are stored contiguously on disk. This differs from the usual row-oriented approach of traditional databases, which store entire rows contiguously.

The reason to store values on a per-column basis instead is based on the assumption that, for specific queries, not all of the values are needed. This is often the case in analytical databases in particular, and therefore they are good candidates for this different storage schema.

Reduced I/O is one of the primary reasons for this new layout, but it offers additional advantages playing into the same category: since the values of one column are often very similar in nature or even vary only slightly between logical rows, they are often much better suited for compression than the heterogeneous values of a row-oriented record structure; most compression algorithms only look at a finite window.

---

2. **Question:**
   Explain the data model of HBase.

Answer:
The data model used by HBase is column-oriented. See section 1.1 (question 12) for further explanation of the data model.

---

3. **Question:**
   Give example applications of HBase.

Answer:
Since the layout of HBase is column-oriented, it's suitable for application that require high write throughput, Real-time analytics and fast reads of recent data and table scans. Such applications could for example be:

- Facebook Messaging
- Twitter

1.5 Hive and Shark/SparkSQL

1. **Question:**
   Explain the problem that Hive and Shark/SparkSQL address.

Answer:
The motivation of introducing Hive, Shark and SparkSQL is that MapReduce programming is low level and that end users need to write code even for simplest aggregations. Since hadoop/Spark lacks expressivness, it can be hard to maintain and reuse written code.

The data warehousing solutions mentioned above supports (translates) queries expressed in SQL (or SQL-like), which opens up the possibilites for non-experienced-MapReduce users to use DFS.

---

2. **Question:**
   Explain the data model of Hive and Shark/SparkSQL.

Answer:

- Hive
  - Hive structures data into the well-understood database concepts like tables, columns, rows, and partitions. It supports all the major primitive types – integers, floats, doubles and strings – as well as complex types such as maps, lists and structs. In summary, column-oriented, row-oriented and text file storage formats
- Shark/SparkSQL
  - Resilient Distributed Datasets (RDDs) ?

---

3. **Question:**
   Discuss the trade-off between schema-on-read and schema-on-write approaches.

Answer:
Schema-on-write (RDBMS):

- Create static DB schema
- Transform data into RDBMS
- Query data in RDMBS format
- Good for known unknowns (repition)

Schema-on-read (Hadoop):

- Copy data in its native format
- Creaste schema + parser
- Query data in its native format (ETL on the fly)
- Good for unkknowns unknowns (exploration)

The shema-on-read makes the DB more flexible and allows you to save unstructured, semi-structured, and/or loosely or unorganized data. However, by adopting schema-on-read you sacrifices:

- The data is not self-documenting (i.e., you can't look at a schema to figure out what the data is)
- You have to spend time creating the jobs that create the schema on read
- Can be "expensive" in terms of compute resources (then again, these big data engines were built to handle that)

Source **here**

---

4. **Question:** Explain the difference between OLAP and OLTP.

Answer:
OLAP stands for Online transaction processing and is common in RDBMS (operational systems), while OLTP stands Online analytical processing and is common in data warehouse solutions.

OLTP uses normalized tables to quickly record large amounts of transactions while making sure that these updates of data occur in as few places as possible. Consequently OLTP database are designed for recording the daily operations and transactions of a business.

OLAP uses database tables (fact and dimension tables) to enable multidimensional viewing, analysis and querying of large amounts of data. E.g. OLAP technology could provide management with fast answers to complex queries on their operational data or enable them to analyze their company's historical data for trends and patterns.

---

5. **Question:**
   Explain the main differences between Hive and Shark and what are the advantages they lead to.

Answer:
Shark is built on the Hive codebase, however it swaps hadoop for Spark. This mean that it uses RDDs, which makes it much faster since data is written in memory and more fault tolerant since RDDs are immutable.

---

6. **Question:**
   Explain how fault tolerance is achieved in Shark/SparkSQL.

Answer:
By using Resilient Distributed Datasets (RDDs), Shark is able to perform most computations in memory while offering fine-grained fault tolerance. In case of failure, Shark will recover from the failure by tracking the lineage of each dataset and recompute lost data.

Parallel Computing

1. **Par-Q1 - Question:**
   Define the following technical terms (Be thorough and general. An example is not a definition.):

- Cluster (in high-performance resp. big-data computing)
- Parallel work (of a parallel algorithm)
- Parallel speed-up
- Communication latency (for sending a message from node Pi to node Pj)
- Temporal data locality
- Dynamic task scheduling