Apache HBase

Apache HBase is an open-source, non-relational, distributed database modeled after Google's Bigtable and is written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS (Hadoop Distributed File System), providing Bigtable-like capabilities for Hadoop.

1)When Should I Use HBase?

Answer)HBase isn't suitable for every problem.

First, make sure you have enough data. If you have hundreds of millions or billions of rows, then HBase is a good candidate. If you only have a few thousand/million rows, then using a traditional RDBMS might be a better choice due to the fact that all of your data might wind up on a single node (or two) and the rest of the cluster may be sitting idle.

Second, make sure you can live without all the extra features that an RDBMS provides (e.g., typed columns, secondary indexes, transactions, advanced query languages, etc.) An application built against an RDBMS cannot be "ported" to HBase by simply changing a JDBC driver, for example. Consider moving from an RDBMS to HBase as a complete redesign as opposed to a port.

Third, make sure you have enough hardware. Even HDFS doesn't do well with anything less than 5 DataNodes (due to things such as HDFS block replication which has a default of 3), plus a NameNode.

HBase can run quite well stand-alone on a laptop - but this should be considered a development configuration only.

2)Can you create HBase table without assigning column family?

Answer)No, Column family also impact how the data should be stored physically in the HDFS file system, hence there is a mandate that you should always have at least one column family. We can also alter the column families once the table is created.

3)Does HBase support SQL?

Answer)Not really. SQL-ish support for HBase via Hive is in development, however Hive is based on MapReduce which is not generally suitable for low-latency requests.

4)Why are the cells above 10MB not recommended for HBase?

Answer)Large cells don't fit well into HBase's approach to buffering data. First, the large cells bypass the MemStoreLAB when they are written. Then, they cannot be cached in the L2 block cache during read operations. Instead, HBase has to allocate on-heap memory for them each time. This can have a significant impact on the garbage collector within the RegionServer process.

5)How should I design my schema in HBase?

Answer)A good introduction on the strength and weaknesses modelling on the various non-rdbms datastores is to be found in Ian Varley's Master thesis, No Relation: The Mixed Blessings of Non-Relational Databases. It is a little dated now but a good background read if you have a moment on how HBase schema modeling differs from how it is done in an RDBMS. Also, read keyvalue for how HBase stores data internally, and the section on schema.casestudies.

The documentation on the Cloud Bigtable website, Designing Your Schema, is pertinent and nicely done and lessons learned there equally apply here in HBase land; just divide any quoted values by ~10 to get what works for HBase: e.g. where it says individual values can be ~10MBs in size, HBase can do similar perhaps best to go smaller if you can and where it says a maximum of 100 column families in Cloud Bigtable, think ~10 when modeling on HBase.

6)Can you please provide an example of "good de-normalization" in HBase and how its held consistent (in your friends example in a relational db, there would be a cascadingDelete)? As I think of the users table: if I delete an user with the userid='123', do I have to walk through all of the other users column-family "friends" to guaranty consistency?! Is de-normalization in HBase only used to avoid joins? Our webapp doesn't use joins at the moment anyway.

Answer)You lose any concept of foreign keys. You have a primary key, that's it. No secondary keys/indexes, no foreign keys.

It's the responsibility of your application to handle something like deleting a friend and cascading to the friendships. Again, typical small web apps are far simpler to write using SQL, you become responsible for some of the things that were once handled for you.

Another example of "good denormalization" would be something like storing a users "favorite pages". If we want to query this data in two ways: for a given user, all of his favorites. Or, for a given favorite, all of the users who have it as a favorite. Relational database would probably have tables for users, favorites, and userfavorites. Each link would be stored in one row in the userfavorites table. We would have indexes on both 'userid' and 'favoriteid' and could thus query it in both ways described above. In HBase we'd probably put a column in both the users table and the favorites table, there would be no link table.

That would be a very efficient query in both architectures, with relational performing better much better with small datasets but less so with a large dataset.

Now asking for the favorites of these 10 users. That starts to get tricky in HBase and will undoubtedly suffer worse from random reading. The flexibility of SQL allows us to just ask the database for the answer to that question. In a small dataset it will come up with a decent solution, and return the results to you in a matter of milliseconds. Now let's make that userfavorites table a few billion rows, and the number of users you're asking for a couple thousand. The query planner will come up with something but things will fall down and it will end up taking forever. The worst problem will be in the index bloat. Insertions to this link table will start to take a very long time. HBase will perform virtually the same as it did on the small table, if not better because of superior region distribution.

7)How would you design an Hbase table for many-to-many association between two entities, for example Student and Course?
I would define two tables:
Student: student id student data (name, address, ...) courses (use course ids as column qualifiers here) Course: course id course data (name, syllabus, ...) students (use student ids as column qualifiers here)
Does it make sense?

Answer)Your design does make sense.

As you said, you'd probably have two column-families in each of the Student and Course tables. One for the data, another with a column per student or course. For example, a student row might look like: Student : id/row/key = 1001 data:name = Student Name data:address = 123 ABC St courses:2001 = (If you need more information about this association, for example, if they are on the waiting list) courses:2002 = .

This schema gives you fast access to the queries, show all classes for a student (student table, courses family), or all students for a class (courses table, students family).

8)What is the maximum recommended cell size?

Answer)A rough rule of thumb, with little empirical validation, is to keep the data in HDFS and store pointers to the data in HBase if you expect the cell size to be consistently above 10 MB. If you do expect large cell values and you still plan to use HBase for the storage of cell contents, you'll want to increase the block size and the maximum region size for the table to keep the index size reasonable and the split frequency acceptable.

9)Why can't I iterate through the rows of a table in reverse order?

Answer)Because of the way HFile works: for efficiency, column values are put on disk with the length of the value written first and then the bytes of the actual value written second. To navigate through these values in reverse order, these length values would need to be stored twice (at the end as well) or in a side file. A robust secondary index implementation is the likely solution here to ensure the primary use case remains fast.

10)Can I fix OutOfMemoryExceptions in hbase?

Answer)Out-of-the-box, hbase uses a default of 1G heap size. Set the HBASE_HEAPSIZE environment variable in ${HBASE_HOME}/conf/hbase-env.sh if your install needs to run with a larger heap. HBASE_HEAPSIZE is like HADOOP_HEAPSIZE in that its value is the desired heap size in MB. The surrounding '-Xmx' and 'm' needed to make up the maximum heap size java option are added by the hbase start script (See how HBASE_HEAPSIZE is used in the ${HBASE_HOME}/bin/hbase script for clarification).

11)How do I enable hbase DEBUG-level logging?

Answer)Either add the following line to your log4j.properties file log4j.logger.org.apache.hadoop.hbase=DEBUG and restart your cluster or, if running a post-0.15.x version, you can set DEBUG via the UI by clicking on the 'Log Level' link (but you need set 'org.apache.hadoop.hbase' to DEBUG without the 'log4j.logger' prefix).

12)What ports does HBase use?

Answer)Not counting the ports used by hadoop - hdfs and mapreduce - by default, hbase runs the master and its informational http server at 60000 and 60010 respectively and regionservers at 60020 and their informational http server at 60030. ${HBASE_HOME}/conf/hbase-default.xml lists the default values of all ports used. Also check ${HBASE_HOME}/conf/hbase-site.xml for site-specific overrides.

13)Why is HBase ignoring HDFS client configuration such as dfs.replication?

Answer)If you have made HDFS client configuration on your hadoop cluster, HBase will not see this configuration unless you do one of the following:
Add a pointer to your HADOOP_CONF_DIR to CLASSPATH in hbase-env.sh or symlink your hadoop-site.xml from the hbase conf directory.
Add a copy of hadoop-site.xml to ${HBASE_HOME}/conf, or
If only a small set of HDFS client configurations, add them to hbase-site.xml
The first option is the better of the three since it avoids duplication.

14)Can I safely move the master from node A to node B?

Answer)Yes. HBase must be shutdown. Edit your hbase-site.xml configuration across the cluster setting hbase.master to point at the new location.

15)Can I safely move the hbase rootdir in hdfs?

Answer)Yes. HBase must be down for the move. After the move, update the hbase-site.xml across the cluster and restart.

16)How do I add/remove a node?

Answer)For removing nodes, see the section on decommissioning nodes in the HBase
Adding and removing nodes works the same way in HBase and Hadoop. To add a new node, do the following steps:
Edit $HBASE_HOME/conf/regionservers on the Master node and add the new address.
Setup the new node with needed software, permissions.
On that node run $HBASE_HOME/bin/hbase-daemon.sh start regionserver
Confirm it worked by looking at the Master's web UI or in that region server's log.
Removing a node is as easy, first issue "stop" instead of start then remove the address from the regionservers file.
For Hadoop, use the same kind of script (starts with hadoop-*), their process names (datanode, tasktracker), and edit the slaves file. Removing datanodes is tricky, please review the dfsadmin command before doing it.

17)Why do servers have start codes?

Answer)If a region server crashes and recovers, it cannot be given work until its lease times out. If the lease is identified only by an IP address and port number, then that server can't do any progress until the lease times out. A start code is added so that the restarted server can begin doing work immediately upon recovery

18)How do I monitor my HBase Cluster?

Answer)HBase emits performance metrics that you can monitor with Ganglia. Alternatively, you could use SPM for HBase

19)In which file the default configuration of HBase is stored.

Answer)hbase-site.xml

20)What is the RowKey.

Answer)Every row in an HBase table has a unique identifier called its rowkey (Which is equivalent to Primary key in RDBMS, which would be distinct throughout the table). Every interaction you are going to do in database will start with the RowKey only

21)Please specify the command (Java API Class) which you will be using to interact with HBase table.

Answer)Get, Put, Delete, Scan, and Increment

22)Which data type is used to store the data in HBase table column.

Answer)Byte Array,
Put p = new Put(Bytes.toBytes("John Smith"));
All the data in the HBase is stored as raw byte Array (10101010). Now the put instance is created which can be inserted in the HBase users table. © HadoopExam Leaning Resource

23)To locate the HBase data cell which three co-ordinate is used ?

Answer)HBase uses the coordinates to locate a piece of data within a table. The RowKey is the first coordinate. Following three co-ordinates define the location of the cell.
1.RowKey
2.Column Family (Group of columns)
3.Column Qualifier (Name of the columns or column itself e.g. Name, Email, Address) © HadoopExam Leaning Resource
Co-ordinates for the John Smith Name Cell.
["John Smith userID", "info", "name"]

24)When you persist the data in HBase Row, In which tow places HBase writes the data to make sure the durability.

Answer)HBase receives the command and persists the change, or throws an exception if the write fails.
When a write is made, by default, it goes into two places:
a. the write-ahead log (WAL), also referred to as the HLog
b. and the MemStore
The default behavior of HBase recording the write in both places is in order to maintain data durability. Only after the change is written to and confirmed in both places is the write considered complete.

25)What is MemStore?

Answer)The MemStore is a write buffer where HBase accumulates data in memory before a permanent write.
Its contents are flushed to disk to form an HFile when the MemStore fills up.
It doesn’t write to an existing HFile but instead forms a new file on every flush.
There is one MemStore per column family. (The size of the MemStore is defined by the system-wide property in
hbase-site.xml called hbase.hregion.memstore.flush.size)

26)What is HFile ?

Answer)The HFile is the underlying storage format for HBase.
HFiles belong to a column family and a column family can have multiple HFiles.
But a single HFile can’t have data for multiple column families. © HadoopExam.com Leaning Resource

27)How HBase Handles the write failure?

Answer)Failures are common in large distributed systems, and HBase is no exception.
Imagine that the server hosting a MemStore that has not yet been flushed crashes. You’ll lose the data that was in memory but not yet persisted. HBase safeguards against that by writing to the WAL before the write completes. Every server that’s part of the.
HBase cluster keeps a WAL to record changes as they happen. The WAL is a file on the underlying file system. A write isn’t considered successful until the new WAL entry is successfully written. This guarantee makes HBase as durable as the file system backing it. Most of the time, HBase is backed by the Hadoop Distributed Filesystem (HDFS). If HBase goes down, the data that was not yet flushed from the MemStore to the HFile can be recovered by replaying the WAL

28)Which of the API command you will use to read data from HBase.

Answer)Get
Get g = new Get(Bytes.toBytes("John Smith"));
Result r = usersTable.get(g);

29)What is the BlockCache?

Answer)HBase also use the cache where it keeps the most used data in JVM Heap, along side Memstore.The BlockCache is designed to keep frequently accessed data from the HFiles in memory so as to avoid disk reads. Each column family has its own BlockCache

The Block in BlockCache is the unit of data that HBase reads from disk in a single pass. The HFile is physically laid out as a sequence of blocks plus an index over those blocks.This means reading a block from HBase requires only looking up that blocks location in the index and retrieving it from disk.

The block is the smallest indexed unit of data and is the smallest unit of data that can be read from disk.

30)BlockSize is configured on which level?

Answer)The block size is configured per column family, and the default value is 64 KB. You may want to tweak this value larger or smaller depending on your use case.

31)If your requirement is to read the data randomly from HBase User table. Then what would be your preference to keep blcok size.

Answer)Having smaller blocks creates a larger index and thereby consumes more memory. If you frequently perform sequential scans, reading many blocks at a time, you can afford a larger block size. This allows you to save on memory because larger blocks mean fewer index entries and thus a smaller index.

32)What is a block, in a BlockCache ?

Answer)The Block in BlockCache is the unit of data that HBase reads from disk in a single pass. The HFile is physically laid out as a sequence of blocks plus an index over those blocks.

This means reading a block from HBase requires only looking up that blocks location in the index and retrieving it from disk. The block is the smallest indexed unit of data and is the smallest unit of data that can be read from disk.

The block size is configured per column family, and the default value is 64 KB. You may want to tweak this value larger or smaller depending on your use case.

33)While reading the data from HBase, from which three places data will be reconciled before returning the value ?

Answer)a. Reading a row from HBase requires first checking the MemStore for any pending modifications.
b. Then the BlockCache is examined to see if the block containing this row has been recently accessed.
c. Finally, the relevant HFiles on disk are accessed.
d. Note that HFiles contain a snapshot of the MemStore at the point when it was flushed. Data for a complete row can be stored across multiple HFiles.
e. In order to read a complete row, HBase must read across all HFiles that might contain information for that row in order to compose the complete record.

34)Once you delete the data in HBase, when exactly they are physically removed?

Answer)During Major compaction, Because HFiles are immutable, it's not until a major compaction runs that these tombstone records are reconciled and space is truly recovered from deleted records.

35)Please describe minor compaction

Answer)Minor : A minor compaction folds HFiles together, creating a larger HFile from multiple smaller HFiles.

36)Please describe major compactation?

Answer)When a compaction operates over all HFiles in a column family in a given region, it's called a major compaction. Upon completion of a major compaction, all HFiles in the column family are merged into a single file

37)What is tombstone record?

Answer)The Delete command doesn't delete the value immediately. Instead, it marks the record for deletion. That is, a new tombstone record is written for that value, marking it as deleted. The tombstone is used to indicate that the deleted value should no longer be included in Get or Scan results.

38)Can major compaction manually triggered?

Answer)Major compactions can also be triggered (or a particular region) manually from the shell. This is a relatively expensive operation and isn't done often. Minor compactions, on the other hand, are relatively lightweight and happen more frequently.

39)Which process or component is responsible for managing HBase RegionServer?

Answer)HMaster is the implementation of the Master Server.The Master server is responsible for monitoring all RegionServer instances in the cluster, and is the interface for all metadata changes. In a distributed cluster, the Master typically runs on the NameNode.

40)Which component is responsible for managing and monitoring of Regions?

Answer)HRegionServer is the RegionServer implementation. It is responsible for serving and managing regions. In a distributed cluster, a RegionServer runs on a DataNode.

41)What is the use of HColumnDescriptor?

Answer)An HColumnDescriptor contains information about a column family such as the number of versions, compression settings, etc. It is used as input when creating a table or adding a column. It is used as input when creating a table or adding a column. Once set, the parameters that specify a column cannot be changed without deleting the column and recreating it. If there is data stored in the column, it will be deleted when the column is deleted.

42)What is Field swap/promotion?

Answer)You can move the timestamp field of the row key or prefix it with another field. This approach uses the composite row key concept to move the sequential, monotonously increasing timestamp to a secondary position in the row key. If you already have a row key with more than one field, you can swap them. If you have only the timestamp as the current row key, you need to promote another field from the column keys, or even the value, into the row key. There is also a drawback to moving the time to the right-hand side in the composite key: you can only access data, especially time ranges, for a given swapped or promoted field.

43)Please tell us Operational command in Hbase, we you have used?

Answer)There are five main command in HBase.
1. Get
2. Put
3. Delete
4. Scan
5. Increment

44)Write down the Java Code snippet to open a connection in Hbase?

Answer)If you are going to open connection with the help of Java API.
The following code provide the connection
Configuration myConf = HBaseConfiguration.create();
HTableInterface usersTable = new HTable(myConf, "users");

45)Explain what is the row key?

Answer)Row key is defined by the application. As the combined key is pre-fixed by the rowkey, it enables the application to define the desired sort order. It also allows logical grouping of cells and make sure that all cells with the same rowkey are co-located on the same server.

46)What is the Deferred Log Flush in HBase?

Answer)The default behavior for Puts using the Write Ahead Log (WAL) is that HLog edits will be written immediately. If deferred log flush is used, WAL edits are kept in memory until the flush period. The benefit is aggregated and asynchronous HLog- writes, but the potential downside is that if the RegionServer goes down the yet-to-be-flushed edits are lost. This is safer, however, than not using WAL at all with Puts.

Deferred log flush can be configured on tables via HTableDescriptor. The default value of hbase.regionserver.optionallogflushinterval is 1000ms.

47)Can you describe the HBase Client: AutoFlush ?

Answer)When performing a lot of Puts, make sure that setAutoFlush is set to false on your HTable instance. Otherwise, the Puts will be sent one at a time to the RegionServer. Puts added via htable.add(Put) and htable.add(List Put) wind up in the same write buffer. If autoFlush = false, these messages are not sent until the write-buffer is filled. To explicitly flush the messages, call flushCommits. Calling close on the HTable instance will invoke flushCommits.

Apache ZooKeeper

Apache ZooKeeper is a software project of the Apache Software Foundation. It is essentially a distributed hierarchical key-value store, which is used to provide a distributed configuration service, synchronization service, and naming registry for large distributed systems.

1)What Is Zookeper??

Answer)ZooKeeper is a distributed co-ordination service to manage large set of hosts. Co-ordinating and managing a service in a distributed environment is a complicated process. ZooKeeper solves this issue with its simple architecture and API. ZooKeeper allows developers to focus on core application logic without worrying about the distributed nature of the application.

The ZooKeeper framework was originally built at "Yahoo!" for accessing their applications in an easy and robust manner. Later, Apache ZooKeeper became a standard for organized service used by Hadoop, HBase, and other distributed frameworks

2)What Are The Benefits Of Distributed Applications?

Answer)Reliability:Failure of a single or a few systems does not make the whole system to fail.
Scalability : Performance can be increased as and when needed by adding more machines with minor change in the configuration of the application with no downtime.
Transparency: Hides the complexity of the system and shows itself as a single entity / application.

3)What Are The Benefits Of Zookeeper?

Answer)Here are the benefits of using ZooKeeper:
Simple distributed coordination process
Synchronization:Mutual exclusion and co-operation between server processes. This process helps in Apache HBase for configuration management.
Ordered Messages
Serialization :Encode the data according to specific rules. Ensure your application runs consistently. This approach can be used in MapReduce to coordinate queue to execute running threads.
Reliability
Atomicity:Data transfer either succeed or fail completely, but no transaction is partial.

4)Explain The Types Of Znodes?

Answer)Znodes are categorized as persistence, sequential, and ephemeral.

Persistence znode - Persistence znode is alive even after the client, which created that particular znode, is disconnected. By default, all znodes are persistent unless otherwise specified.

Ephemeral znode - Ephemeral znodes are active until the client is alive. When a client gets disconnected from the ZooKeeper ensemble, then the ephemeral znodes get deleted automatically. For this reason, only ephemeral znodes are not allowed to have a children further. If an ephemeral znode is deleted, then the next suitable node will fill its position. Ephemeral znodes play an important role in Leader election.

Sequential znode - Sequential znodes can be either persistent or ephemeral. When a new znode is created as a sequential znode, then ZooKeeper sets the path of the znode by attaching a 10 digit sequence number to the original name. For example, if a znode with path /myapp is created as a sequential znode, ZooKeeper will change the path to /myapp0000000001 and set the next sequence number as 0000000002. If two sequential znodes are created concurrently, then ZooKeeper never uses the same number for each znode. Sequential znodes play an important role in Locking and Synchronization.

5)Explain The Zookeeper Workflow?

Answer)Once a ZooKeeper ensemble starts, it will wait for the clients to connect. Clients will connect to one of the nodes in the ZooKeeper ensemble. It may be a leader or a follower node. Once a client is connected, the node assigns a session ID to the particular client and sends an acknowledgement to the client. If the client does not get an acknowledgment, it simply tries to connect another node in the ZooKeeper ensemble. Once connected to a node, the client will send heartbeats to the node in a regular interval to make sure that the connection is not lost.

If a client wants to read a particular znode, it sends a read request to the node with the znode path and the node returns the requested znode by getting it from its own database. For this reason, reads are fast in ZooKeeper ensemble.

If a client wants to store data in the ZooKeeper ensemble, it sends the znode path and the data to the server. The connected server will forward the request to the leader and then the leader will reissue the writing request to all the followers. If only a majority of the nodes respond successfully, then the write request will succeed and a successful return code will be sent to the client. Otherwise, the write request will fail. The strict majority of nodes is called as Quorum.

6)Explain The Cli In Zookeeper?

Answer)ZooKeeper Command Line Interface (CLI) is used to interact with the ZooKeeper ensemble for development purpose. It is useful for debugging and working around with different options. To perform ZooKeeper CLI operations, first turn on your ZooKeeper server ("bin/zkServer.sh start") and then, ZooKeeper client (bin/zkCli.sh).

Once the client starts, you can perform the following operation:
Create znodes

Get data
Watch znode for changes
Set data
Create children of a znode
List children of a znode
Check Status
Remove or Delete a znode

7)How Can We Create Znodes?

Answer)Create a znode with the given path. The flag argument specifies whether the created znode will be ephemeral, persistent, or sequential. By default, all znodes are persistent.

Ephemeral znodes (flag: e) will be automatically deleted when a session expires or when the client disconnects.

Sequential znodes guaranty that the znode path will be unique.

ZooKeeper ensemble will add sequence number along with 10 digit padding to the znode path. For example, the znode path /myapp will be converted to /myapp0000000001 and the next sequence number will be /myapp0000000002.
If no flags are specified, then the znode is considered as persistent.
create /path /data
To create a Sequential znode, add -s flag as shown below.
create -s /path /data
To create an Ephemeral Znode, add -e flag as shown below.
create -e /path /data

8)How Can We Create Children / Sub-znode?

Answer)Creating children is similar to creating new znodes. The only difference is that the path of the child znode will have the parent path as well.

create /parent/path/subnode/path /data

9)How Can We Remove A Znode?

Answer)Removes a specified znode and recursively all its children. This would happen only if such a znode is available.

rmr /path

10)What Are The Basics Of Zookeeper Api?

Answer)Application interacting with ZooKeeper ensemble is referred as ZooKeeper Client or simply Client. Znode is the core component of ZooKeeper ensemble and ZooKeeper API provides a small set of methods to manipulate all the details of znode with ZooKeeper ensemble. A client should follow the steps given below to have a clear and clean interaction with ZooKeeper ensemble.

Connect to the ZooKeeper ensemble. ZooKeeper ensemble assign a Session ID for the client.

Send heartbeats to the server periodically. Otherwise, the ZooKeeper ensemble expires the Session ID and the client needs to reconnect.

Get / Set the znodes as long as a session ID is active.

Disconnect from the ZooKeeper ensemble, once all the tasks are completed. If the client is inactive for a prolonged time, then the ZooKeeper ensemble will automatically disconnect the client.

11)Explain The Methods Of Zookeeperclass?

Answer)The central part of the ZooKeeper API is ZooKeeper class. It provides options to connect the ZooKeeper ensemble in its constructor and has the following methods -
connect - connect to the ZooKeeper ensemble
ZooKeeper(String connectionString, int sessionTimeout, Watcher watcher)
create - create a znode
create(String path, byte[] data, List acl, CreateMode createMode)
exists - check whether a znode exists and its information
exists(String path, boolean watcher)
getData - get data from a particular znode
getData(String path, Watcher watcher, Stat stat)
setData - set data in a particular znode
setData(String path, byte[] data, int version)
getChildren - get all sub-nodes available in a particular znode
getChildren(String path, Watcher watcher)
delete - get a particular znode and all its children
delete(String path, int version)
close - close a connection

12)Mention Some Instances Where Zookeeper Is Using?

Answer)Below are some of instances where Apache ZooKeeper is being utilized:

Apache Storm, being a real time stateless processing/computing framework, manages its state in ZooKeeper Service

Apache Kafka uses it for choosing leader node for the topic partitions

Apache YARN relies on it for the automatic failover of resource manager (master node)

Yahoo! utilties it as the coordination and failure recovery service for Yahoo! Message Broker, which is a highly scalable publish-subscribe system managing thousands of topics for replication and data delivery. It is used by the Fetching Service for Yahoo! crawler, where it also manages failure recovery.

13)Can Apache Kafka be used without Zookeeper?

Answer)It is not possible to use Apache Kafka without Zookeeper because if the Zookeeper is down Kafka cannot serve client request.

14)What is the role of Zookeeper in HBase architecture?

Answer)In HBase architecture, ZooKeeper is the monitoring server that provides different services like –tracking server failure and network partitions, maintaining the configuration information, establishing communication between the clients and region servers, usability of ephemeral nodes to identify the available servers in the cluster.

15)Explain about ZooKeeper in Kafka

Answer)Apache Kafka uses ZooKeeper to be a highly distributed and scalable system. Zookeeper is used by Kafka to store various configurations and use them across the hadoop cluster in a distributed manner. To achieve distributed-ness, configurations are distributed and replicated throughout the leader and follower nodes in the ZooKeeper ensemble. We cannot directly connect to Kafka by bye-passing ZooKeeper because if the ZooKeeper is down it will not be able to serve the client request.

16)Explain how Zookeeper works

Answer)ZooKeeper is referred to as the King of Coordination and distributed applications use ZooKeeper to store and facilitate important configuration information updates. ZooKeeper works by coordinating the processes of distributed applications. ZooKeeper is a robust replicated synchronization service with eventual consistency. A set of nodes is known as an ensemble and persisted data is distributed between multiple nodes.

3 or more independent servers collectively form a ZooKeeper cluster and elect a master. One client connects to any of the specific server and migrates if a particular node fails. The ensemble of ZooKeeper nodes is alive till the majority of nods are working. The master node in ZooKeeper is dynamically selected by the consensus within the ensemble so if the master node fails then the role of master node will migrate to another node which is selected dynamically. Writes are linear and reads are concurrent in ZooKeeper.

17)List some examples of Zookeeper use cases.

Answer)Found by Elastic uses Zookeeper comprehensively for resource allocation, leader election, high priority notifications and discovery. The entire service of Found built up of various systems that read and write to Zookeeper.
Apache Kafka that depends on ZooKeeper is used by LinkedIn
Storm that relies on ZooKeeper is used by popular companies like Groupon and Twitter.

18)How to use Apache Zookeeper command line interface?

Answer)ZooKeeper has a command line client support for interactive use. The command line interface of ZooKeeper is similar to the file and shell system of UNIX. Data in ZooKeeper is stored in a hierarchy of Znodes where each znode can contain data just similar to a file. Each znode can also have children just like directories in the UNIX file system.

Zookeeper-client command is used to launch the command line client. If the initial prompt is hidden by the log messages after entering the command, users can just hit ENTER to view the prompt.

19)What are the different types of Znodes?

Answer)There are 2 types of Znodes namely- Ephemeral and Sequential Znodes.
The Znodes that get destroyed as soon as the client that created it disconnects are referred to as Ephemeral Znodes.
Sequential Znode is the one in which sequential number is chosen by the ZooKeeper ensemble and is pre-fixed when the client assigns name to the znode.

20)What are watches?

Answer)Client disconnection might be troublesome problem especially when we need to keep a track on the state of Znodes at regular intervals. ZooKeeper has an event system referred to as watch which can be set on Znode to trigger an event whenever it is removed, altered or any new children are created below it.

21)What problems can be addressed by using Zookeeper?

Answer)In the development of distributed systems, creating own protocols for coordinating the hadoop cluster results in failure and frustration for the developers. The architecture of a distributed system can be prone to deadlocks, inconsistency and race conditions. This leads to various difficulties in making the hadoop cluster fast, reliable and scalable. To address all such problems, Apache ZooKeeper can be used as a coordination service to write correct distributed applications without having to reinvent the wheel from the beginning.

22)How should I handle the CONNECTION_LOSS error?

Answer)CONNECTION_LOSS means the link between the client and server was broken. It doesn't necessarily mean that the request failed. If you are doing a create request and the link was broken after the request reached the server and before the response was returned, the create request will succeed. If the link was broken before the packet went onto the wire, the create request failed. Unfortunately, there is no way for the client library to know, so it returns CONNECTION_LOSS. The programmer must figure out if the request succeeded or needs to be retried. Usually this is done in an application specific way. Examples of success detection include checking for the presence of a file to be created or checking the value of a znode to be modified.

When a client (session) becomes partitioned from the ZK serving cluster it will begin searching the list of servers that were specified during session creation. Eventually, when connectivity between the client and at least one of the servers is re-established, the session will either again transition to the connected state (if reconnected within the session timeout value) or it will transition to the expired state (if reconnected after the session timeout). The ZK client library will handle reconnect for you automatically. In particular we have heuristics built into the client library to handle things like herd effect, etc. Only create a new session when you are notified of session expiration (mandatory).

23)How should I handle SESSION_EXPIRED?

Answer)SESSION_EXPIRED automatically closes the ZooKeeper handle. In a correctly operating cluster, you should never see SESSION_EXPIRED. It means that the client was partitioned off from the ZooKeeper service for more the the session timeout and ZooKeeper decided that the client died. Because the ZooKeeper service is ground truth, the client should consider itself dead and go into recovery. If the client is only reading state from ZooKeeper, recovery means just reconnecting. In more complex applications, recovery means recreating ephemeral nodes, vying for leadership roles, and reconstructing published state.

Library writers should be conscious of the severity of the expired state and not try to recover from it. Instead libraries should return a fatal error. Even if the library is simply reading from ZooKeeper, the user of the library may also be doing other things with ZooKeeper that requires more complex recovery.

Session expiration is managed by the ZooKeeper cluster itself, not by the client. When the ZK client establishes a session with the cluster it provides a timeout value. This value is used by the cluster to determine when the client's session expires. Expirations happens when the cluster does not hear from the client within the specified session timeout period (i.e. no heartbeat). At session expiration the cluster will delete any/all ephemeral nodes owned by that session and immediately notify any/all connected clients of the change (anyone watching those znodes). At this point the client of the expired session is still disconnected from the cluster, it will not be notified of the session expiration until/unless it is able to re-establish a connection to the cluster.

The client will stay in disconnected state until the TCP connection is re-established with the cluster, at which point the watcher of the expired session will receive the session expired notification.

24)Is there an easy way to expire a session for testing?

Answer)Yes, a ZooKeeper handle can take a session id and password. This constructor is used to recover a session after total application failure. For example, an application can connect to ZooKeeper, save the session id and password to a file, terminate, restart, read the session id and password, and reconnect to ZooKeeper without loosing the session and the corresponding ephemeral nodes. It is up to the programmer to ensure that the session id and password isn't passed around to multiple instances of an application, otherwise problems can result.

In the case of testing we want to cause a problem, so to explicitly expire a session an application connects to ZooKeeper, saves the session id and password, creates another ZooKeeper handle with that id and password, and then closes the new handle. Since both handles reference the same session, the close on second handle will invalidate the session causing a SESSION_EXPIRED on the first handle.

25)Why doesn't the NodeChildrenChanged and NodeDataChanged watch events return more information about the change?

Answer)When a ZooKeeper server generates the change events, it knows exactly what the change is. In our initial implementation of ZooKeeper we returned this information with the change event, but it turned out that it was impossible to use correctly. There may be a correct way to use it, but we have never seen a case of correct usage. The problem is that watches are used to find out about the latest change. (Otherwise, you would just do periodic gets.) The thing that most programmers seem to miss, when they ask for this feature, is that watches are one time triggers. Observe the following case of data change: a process does a getData on /a with watch set to true and gets v1, another process changes /a to v2 and shortly there after changes /a to v3. The first process would see that /a was changed to v2, but wouldn't know that /a is now /v3.

26)What are the options-process for upgrading ZooKeeper?

Answer)There are two primary ways of doing this; 1) full restart or 2) rolling restart.

In the full restart case you can stage your updated code/configuration/etc., stop all of the servers in the ensemble, switch code/configuration, and restart the ZooKeeper ensemble. If you do this programmatically (scripts typically, ie not by hand) the restart can be done on order of seconds. As a result the clients will lose connectivity to the ZooKeeper cluster during this time, however it looks to the clients just like a network partition. All existing client sessions are maintained and re-established as soon as the ZooKeeper ensemble comes back up. Obviously one drawback to this approach is that if you encounter any issues (it's always a good idea to test or stage these changes on a test harness) the cluster may be down for longer than expected.

The second option, preferable for many users, is to do a rolling restart. In this case you upgrade one server in the ZooKeeper ensemble at a time; bring down the server, upgrade the code/configuration/etc., then restart the server. The server will automatically rejoin the quorum, update it's internal state with the current ZK leader, and begin serving client sessions. As a result of doing a rolling restart, rather than a full restart, the administrator can monitor the ensemble as the upgrade progresses, perhaps rolling back if any issues are encountered.

27)What happens to ZK sessions while the cluster is down?

Answer)Imagine that a client is connected to ZK with a 5 second session timeout, and the administrator brings the entire ZK cluster down for an upgrade. The cluster is down for several minutes, and then is restarted.

In this scenario, the client is able to reconnect and refresh its session. Because session timeouts are tracked by the leader, the session starts counting down again with a fresh timeout when the cluster is restarted. So, as long as the client connects within the first 5 seconds after a leader is elected, it will reconnect without an expiration, and any ephemeral nodes it had prior to the downtime will be maintained.

The same behavior is exhibited when the leader crashes and a new one is elected. In the limit, if the leader is flip-flopping back and forth quickly, sessions will never expire since their timers are getting constantly reset.