In this article, I am trying to cover all the **scalability interview questions** you may be asked when you are looking for a web application software architect job. This list will also make a foundation for java architect interview questions or any other programming **language agnostic** software architect interview questions.

What Do You Mean By High Availability?

Having better service capacity with high availability and low latency is mission critical for almost all businesses.  
  
Availability means the ability of the application user to access the system, If a user cannot access the application, it is assumed unavailable. High Availability means the application will be available, without interruption.  
  
Achieving high availability for a application is not always a easy task. Using redundant server nodes with clustering is a common way to achieve higher level of availability in web applications.   
  
Availability is commonly expressed as a percentage of uptime in a given year.

What Is Scalability?

Scalability is the ability of a system, network, or process to handle a growing amount of load by adding more resources. The adding of resource can be done in two ways

* **Scaling Up**

This involves adding more resources to the existing nodes. For example, adding more RAM, Storage or processing power.

* **Scaling Out**

This involves adding more nodes to support more users.

Any of the approaches can be used for scaling up/out a application, however the cost of adding resources (per user) may change as the volume increases. If we add resources to the system It should increase the ability of application to take more load in a proportional manner of added resources.  
  
An ideal application should be able to serve high level of load in less resources. However, in practical, linearly scalable system may be the best option achievable.   
  
Poorly designed applications may have really high cost on scaling up/out since it will require more resources/user as the load increases.

What Is A Cluster?

A cluster is group of computer machines that can individually run a software. Clusters are typically utilized to achieve high availability for a server software.   
  
Clustering is used in many types of servers for high availability.

* **App Server Cluster**

An app server cluster is group of machines that can run a application server that can be reliably utilized with a minimum of down-time.

* **Database Server Cluster**

An database server cluster is group of machines that can run a database server that can be reliably utilized with a minimum of down-time.

Why Do You Need Clustering?

Clustering is needed for achieving high availability for a server software. The main purpose of clustering is to achieve 100% availability or a zero down time in service.   
  
A typical server software can be running on one computer machine and it can serve as long as there is no hardware failure or some other failure.  
  
By creating a cluster of more than one machine, we can reduce the chances of our service going un-available in case one of the machine fails.  
  
Doing clustering does not always guarantee that service will be 100% available since there can still be a chance that all the machine in a cluster fail at the same time. However it in not very likely in case you have many machines and they are located at different location or supported by their own resources.

What Is Middle Tier Clustering?

Middle tier clustering is just a cluster that is used for service the middle tier in a application. This is popular since many clients may be using middle tier and a lot of heavy load may also be served by middle tier that requires it be to highly available.  
  
Failure of middle tier can cause multiple clients and systems to fail, therefore its one of the approaches to do clustering at the middle tier of a application.  
  
In java world, it is really common to have EJB server clusters that are used by many clients. In general any application that has a business logic that can be shared across multiple client can use a middle tier cluster for high availability.

What Is Load Balancing?

Load balancing is simple technique for distributing workloads across multiple machines or clusters.   
The most common and simple load balancing algorithm is Round Robin. In this type of load balancing the request is divided in circular order ensuring all machines get equal number of requests and no single machine is overloaded or underloaded.   
  
**The Purpose of load balancing is to**

* Optimize resource usage (Avoid overload and under-load of any machines.)
* Achieve Maximum Throughput
* Minimize response time

**Most common load balancing techniques in web based applications are**

1. Round robin
2. Session affinity or sticky session
3. IP Address affinity

What Is Sticky Session (session Affinity) Load Balancing? What Do You Mean By 'session Affinity'?

Sticky session or a session affinity technique another popular load balancing technique that requires a user session to be always served by a allocated machine. 

Why Sticky Session?

In a load balanced server application where user information is stored in session it will be required to keep the session data available to all machines. This can be avoided by always serving a particular user session request from one machine.

How It Is Done?

The machine is associated with a session as soon as the session is created. All the requests in a particular session are always redirected to the associated machine. This ensures the user data is only at one machine and load is also shared.  
  
In Java world, this is typically done by using jsessionid cookie. The cookie is sent to the client for the first request and every subsequent request by client must be containing that same cookie to identify the session.

What Are The Issues With Sticky Session?

There are few issues that you may face with this approach

* The client browser may not support cookies, and your load balancer will not be able to identify if a request belongs to a session. This may cause strange behavior for the users who use no cookie based browsers.
* In case one of the machine fails or goes down, the user information (served by that machine) will be lost and there will be no way to recover user session.

What Is IP Address Affinity Technique For Load Balancing?

IP address affinity is another popular way to do load balancing. In this approach, the client IP address is associated with a server node. All requests from a client IP address are served by one server node.  
  
This approach can be really easy to implement since IP address is always available in a HTTP request header and no additional settings need to be performed.  
  
This type of load balancing can be useful if you clients are likely to have disabled cookies.  
  
However there is a down side of this approach. If many of your users are behind a NATed IP address then all of them will end up using the same server node. This may cause uneven load on your server nodes.  
  
NATed IP address is really common, in fact anytime you are browsing from a office network its likely that you and all your coworkers are using same NATed IP address.

What Is Fail Over?

Fail over means switching to another machine when one of the machine fails.   
  
Fail over is a important technique in achieving high availability. Typically a load balancer is configured to fail over to another machine when the main machie fails.   
  
To achieve least down time, most load balancer support a feature of heart beat check. This ensures that target machine is responding. As soon as a hear beat signal fails, load balancer stops sending request to that machine and redirects to other machines or cluster.

What Is Session Replication?

Session replication is used in application server clusters to achieve session failover.   
A user session is replicated to other machines of a cluster, every time the session data changes.  
If a machine fails, the load balancer can simply send incoming requests to another server in the cluster.  
The user can be sent to any server in the cluster since all machines in a cluster have copy of the session.  
  
Session replication may allow your application to have session failover but it may require you to have extra cost in terms of memory and network bandwidth.

What Does Distributable Tag Means In Web.xml ?

In Java world, JEE applications use the concept of distributable web applications to provide session-failover and enable load balancing.  
  
You can set a JEE application to support session replication by adding distributable tag in web.xml file.

<distributable />

What Are The Requirements For Making A Java EE Application Session Replication Enabled?

Setting distributable tag in web.xml just enables the application to support session replication, however it does not guarantee that your application will work fine in a session replicated environment.  
  
JEE Application developer needs to make sure following things are taken care during web application development.

* All attributes/objects that are saved in HTTP Session are serializable. This means all your custom objects and child objects of that should be serializable.
* Making changes to any session attribute should be done using session.setAttribute() method. If you have reference to a java object that was previously set in session, you must call session.setAttribute() method every time you make any change to the object.

What Are Different Mechanism Of Session Replication?

Session replication between multiple cluster nodes can be done in many ways. The best approach may depend on the type of application. However there are few common methods used by application server vendors.

* Using session persistence, and saving the session to a shared file system (PersistenceManager + FileStore) . This will allow all machines in a cluster to be able to access the persisted session from the shared file system.
* Using session persistence, and saving the session to a shared database (PersistenceManager + JDBCStore) - This will allow all machines in a cluster to be able to access the persisted session from the shared database system.
* Using in-memory-replication, This will create a in memory copy of session in all the cluster nodes.

What Is CAP Theorem?

The CAP Theorem for distributed computing was published by Eric Brewer, This states that it is not possible for a distributed computer system to simultaneously provide all three of the following guarantees:

1. Consistency (all nodes see the same data even at the same time with concurrent updates )
2. Availability (a guarantee that every request receives a response about whether it was successful or failed)
3. Partition tolerance (the system continues to operate despite arbitrary message loss or failure of part of the system)

The CAP acronym corresponds to these 3 guarantees. This theorem has created the base for modern distributed computing approaches.   
  
Worlds most high volume traffic companies (e.g. Amazon, Google, Facebook) use this as basis for deciding their application architecture.   
  
Its important to understand that only two of these three conditions can be guaranteed to be met by a system.

What Is Sharding?

Sharding is a architectural approach that distributes a single logical database system into a cluster of machines.  
  
Sharding is Horizontal partitioning design scheme. In this database design rows of a database table are stored separately, instead of splitting into columns (like in normalization and vertical partitioning). Each partition is called as a shard, which can be independently located on a separate database server or physical location.  
  
Sharding makes a database system highly scalable. The total number of rows in each table in each database is reduced since the tables are divided and distributed into multiple servers. This reduces the index size, which generally means improved search performance.  
  
The most common approach for creating shards is by the use of consistent hashing of a unique id in application (e.g. user id).   
  
**The downsides of sharding are,**

* It requires application to be aware of the data location.
* Any addition or deletion of nodes from system will require some rebalance to be done in the system.
* If you require lot of cross node join queries then your performance will be really bad. Therefore, knowing how the data will be used for querying becomes really important.
* A wrong sharding logic may result in worse performance. Therefore make sure you shard based on the application need.

What Is ACID Property Of A System?

ACID is a acronym which is commonly used to define the properties of a relational database system, it stand for following terms

* **Atomicity** - This property guarantees that if one part of the transaction fails, the entire transaction will fail, and the database state will be left unchanged.
* **Consistency** - This property ensures that any transaction will bring the database from one valid state to another.
* **Isolation** - This property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially.
* **Durable** - means that once a transaction has been committed, it will remain so, even in the event of power loss.

What Is BASE Property Of A System?

BASE properties are the common properties of recently evolved NOSQL databases. According to CAP theorem, a BASE system does not guarantee consistency. This is a contrived acronym that is mapped to following property of a system in terms of the CAP theorem

* **Basically available** indicates that the system is guaranteed to be available
* **Soft state**indicates that the state of the system may change over time, even without input. This is mainly due to the eventually consistent model.
* **Eventual consistency** indicates that the system will become consistent over time, given that the system doesn't receive input during that time.

What Do You Mean By Eventual Consistency? What Does Eventually Consistent Mean?

Unlike relational database property of Strict consistency, eventual consistency property of a system ensures that any transaction will eventually (not immediately) bring the database from one valid state to another.   
  
This means there can be intermediate states that are not consistent between multiple nodes.   
  
Eventually consistent systems are useful at scenarios where absolute consistency is not critical. For example in case of Twitter status update, if some users of the system do not see the latest status from a particular user its may not be very devastating for system.   
  
Eventually consistent systems can not be used for use cases where absolute/strict consistency is required. For example a banking transactions system can not be using eventual consistency since it must consistently have the state of a transaction at any point of time. Your account balance should not show different amount if accessed from different ATM machines.  
  
**Some reference material for better understanding on eventual consistency**

* Microsoft Research Whitepaper about [Eventual Consistency](http://research.microsoft.com/apps/pubs/default.aspx?id=189249)
* Amazon CTO about [Eventual Consistency](http://www.allthingsdistributed.com/2007/12/eventually_consistent.html" \o "Eventual Consistency)

What Is Shared Nothing Architecture? How Does It Scale?

A shared nothing architecture (SN) is a distributed computing approach in which each node is independent and self-sufficient, and there is no single point of contention required across the system.

* This means no resources are shared between nodes (No shared memory, No shared file storage)
* The nodes are able to work independently without depending on each other for any work.
* Failure on one node affects only the users of that node, however other nodes continue to work without any disruption.

This approach is highly scalable since it avoid the existence of single bottleneck in the system. Shared nothing is recently become popular for web development due to its linear scalability. Google has been using it for long time.  
  
In theory, A shared nothing system can scale almost infinitely simply by adding nodes in the form of inexpensive machines.

How Do You Update A Live Heavy Traffic Site With Minimum Or Zero Down Time?

Deploying a newer version of a live website can be a challenging task specially when a website has high traffic. Any downtime is going to affect the users. There are a few best practices that we can follow  
  
**Before deploying on Production**

* Thoroughly test the new changes and ensure it working in a test environment which is almost identical to production system.
* If possible do automation of test cases as much as possible. We use selenium for a lot of functional testing.
* Create a automated sanity testing script (also called as smoke test) that can be run on production (without affecting real data). These are typically readonly type of test cases. However depending on your application needs you can add more cases to this. Make sure it can be run quickly by keeping it short.
* Create scripts for all manual tasks(if possible), avoiding any hand typing mistakes during day of deployment.
* Test the script to make sure they work on a non-production environment.
* Keep the build artifacts ready. e.g application deployment files, database scripts, config files etc.
* Create a checklist of things to do on day of deployment.
* Rehearse. Deploy in a non-prod environment is almost identical to production. Try this with production data volumes(if possible). Make a note of time required for your tasks so you can plan accordingly.

**When doing deploying on a production environment.**

* Keep backup of current site/data to be able to rollback.
* Use sanity test cases before doing a lot of in depth testing.

his page is for Java/J2EE job seekers and interviewers and for people interested in self assessments on Java/J2EE technologies. There are numerous books and articles on the market covering specific topics like Java, J2EE, EJB, Design Patterns, ANT, Maven, CVS, Multi-Threading, Servlets, JSP, emerging technologies like AOP (Aspect Oriented Programming), Test Driven Development (TDD), Dependency Injection Dl (aka IoC – Inversion of Control) etc. But from an interview perspective it is not possible to brush up on all these books where each book usually has from 300 pages to 600 pages. The basic purpose of this page is to cover question (and only questions)  on all the core concepts and key areas, which all Java/J2EE developers, designers and architects should be conversant with to perform well in their current jobs and to launch a successful career by doing well at interviews. The interviewer can also use this book to make sure that they hire the right candidate depending on their requirements.

**Multi-Threading**

* What language features are available to allow shared access to data in a multi-threading environment? (Hint: Synchronized block,Synchronized method,wait, notify)
* What is the difference between synchronized method and synchronized block? (Hint:Block on subset of data. Smaller code segment).
* What Java language features would you use to implement a producer (one thread) and a consumer (another thread) passing data via a stack? (Hint: wait, notify)

**Java language**

* What Java classes are provided for date manipulation? (Hint:Calendar, Date)
* What is the difference between String and StringBuffer? (Hint: mutable, efficient)
* How do you ensure a class is Serializable? (Hint:Implement Serializable)
* What is the difference between static and instance field of a class? (Hint:Per class vs. Per Object)
* What methods do you need to implement to store a class in Hashtable or HashMap? (Hint: hashCode(), equals()) .
* How do you exclude a field of a class from serialization? (Hint: transient)

**Inheritance**

* What is the difference between an Interface and an abstract base class? (Hint: interface inheritance, implementation inheritance.)
* What does overriding a method mean? (Hint: Inheritance)
* What about overloading? (Hint: different signature)

**Memory**

* What is the Java heap, and what is the stack? (Hint: dynamic, program thread execution.)
* Why does garbage collection occur and when can it occur? (Hint: To recover memory, as heap gets full.)
* If I have a circular reference of objects, but I no longer reference any of them from any executing thread, will these cause garbage collection problems? (Hint: no)

**Exceptions**

* What is the problem or benefits of catching or throwing type “java.lang.Exception”? (Hint: Hides all subsequent exceptions.)
* What is the difference between a runtime exception and a checked exception? (Hint: Must catch or throw checked exceptions.)

**JSP**

* What is the best practice regarding the use of scriptlets in JSP pages? Why? (Hint: Avoid)How can you avoid scriptlet code? (Hint:custom tags, Java beans)
* What do you understand by the term JSP compilation? (Hint: compiles to servlet code)

**Servlets**

* What does Servlet API provide to store user data between requests? (Hint: HttpSession)
* What is the difference between forwarding a request and redirecting? (Hint: redirect return to browser )
* What object do you use to forward a request? (Hint: RequestDispatcher)
* What do you need to be concerned about with storing data in a servlet instance fields? (Hint: Multi-threaded.)
* What’s the requirement on data stored in HttpSession in a clustered (distributable) environment? (Hint: Serializable)
* If I store an object in session, then change its state, is the state replicated to distributed Session? (Hint: No, only on setAttribute() call.)
* How does URL-pattern for servlet work in the web.xml? (Hint: /ddd/\* or \*.jsp)
* What is a filter, and how does it work? (Hint: Before/after request, chain.)

**JDBC**

* What form of statement would you use to include user-supplied values? (Hint: PreparedStatement)
* Why might a preparedStatement be more efficient than a statement? (Hint: Execution plan cache.)
* How would you prevent an SQL injection attack in JDBC? (Hint: PreparsedStatement )
* What is the performance impact of testing against NULL in WHERE clause on Oracle? (Hint: Full table scan. )
* List advantages and disadvantages in using stored procedures? (Hint: Pro: integration with existing dbase, reduced network trafficCon: not portable, mutliple language knowledge required )
* What is the difference between sql.Date, sql.Time, and sql.Timestamp? (Hint: Date only, time only, date and time )
* If you had a missing int value how do you indicate this to PreparedStatement? (Hint: setNull(pos, TYPE))
* How can I perform multiple inserts in one database interaction? (Hint: executeBatch)Given this problem: Program reads 100,000 rows, converts to Java class in list, then converts list to XML file using reflection. Runs out of program memory. How would you fix? (Hint: Read one row at time, limit select, allocate more heap (result set = cursor) )
* How might you model object inheritance in database tables? (Hint: Table per hierarchy, table per class, table per concrete class)

**JNDI**

* What are typical uses for the JNDI API within an enterprise application? (Hint: Resource management, LDAP access)
* Explain the difference between a lookup of these “java:comp/env/ejb/MyBean” and “ejb/MyBean”? (Hint: logical mapping performed for java:comp/env )
* What is the difference between new InitialContext() from servlet or from an EJB? (Hint: Different JNDI environments initialized EJB controller by ejb-jar.xml, servlet by web.xml.)
* What is an LDAP server used for in an enterprise environment? (Hint: authentication, authorization)
* What is authentication, and authorization? (Hint: Confirming identity, confirming access rights )

**EJB**

* What is the difference between Stateless and Stateful session beans (used?) (Hint: Stateful holds per client state )
* What is the difference between Session bean and Entity bean (when used?) (Hint: Entity used for persistence )
* With Stateless Session bean pooling, when would a container typically take a instance from the pool and when would it return it? (Hint: for each business method )
* What is the difference between “Required”, “Supports”, “RequiresNew” “NotSupported”, “Mandatory”, “Never”? (Hint: Needs transaction, existing OK but doesn’t need, must start new one, suspends transaction, must already be started, error if transaction)
* What is “pass-by-reference” and “pass-by-value”, and how does it affect J2EE applications? (Hint: Reference to actual object versus copy of object. RMI pass by value.)
* What EJB patterns, best practices are you aware of? Describe at least two? (Hint: Façade, delegate, value list, DAO, value object).
* Describe some issues/concerns you have with the J2EE specification? (Hint: Get their general opinion of J2EE)
* What do you understand by the term “offline optimistic locking” or long-lived business transaction? How might you implement this using EJB? (Hint: version number, date, field comparisons.)
* Explain performance difference between getting a list of summary information (e.g. customer list) via finder using a BMP entity vs. Session using DAO? (Hint: BMP: n+1 database reads, n RMI calls.)
* What is meant by a coarse-grained and a fine-grained interface? (Hint: Amount of data transferred per method call)

**XML/XSLT**

* What is the difference between a DOM parser and a SAX parser? (Hint: DOM: reads entire model, SAX: event published during parsing.)
* What is the difference between DTD and XML Schema? (Hint: level of detail, Schema is in XML.)
* What does the JAXP API do for you? (Hint: Parser independence. )What is XSLT and how can it be used? (Hint: XML transformation. )
* What would be the XPath to select any element called table with the class attribute of info? (Hint: Table[@class=’info’])

**JMS**

* How can asynchronous events be managed in J2EE? (Hint: JMS)
* How do transactions affect the onMessage() handling of a MDB? (Hint: Taking off queue. )
* If you send a JMS message from an EJB, and transaction rollback, will message be sent? (Hint: yes)
* How do you indicate what topic or queue MDB should react to? (Hint: deployment descriptor )
* What is the difference between a topic and a queue? (Hint: broadcast, single)

**SOAP**

* What is a Web service, and how does it relate to SOAP? (Hint: SOAP is the protocol.)
* What is a common transport for SOAP messages? (Hint: HTTP )
* What is WSDL? How would you use a WSDL file? (Hint: XML description of Web Service: interface and how to bind to it. )
* With new J2EE SOAP support what is: JAXR, JAX-RPC, and SAAJ? (Hint: registry, rpc, attachments)

**Security**

* Where can container level security be applied in J2EE application? (Hint: Web Uri’s, EJB methods)
* How can the current user be obtained in a J2EE application (Web and Enterprise)? (Hint: getUserPrincipal, getCallerPrincipal )
* How can you perform role checks in a J2EE application (Web and enterprise)? (Hint: IsUserInRole(), IsCallerInRole() )

**Design**

* Name some types of UML diagrams? (Hint: class, sequence, activity, use case)
* Describe some types of relationships can you show on class diagrams? (Hint: generalization, aggregation, uses)
* What is the difference between association, aggregation, and generalization? (Hint: Relationship, ownership, inheritance)
* What is a sequence diagram used to display? ( Hint: Object instance interactions via operations/signals)What design patterns do you use. Describe one you have used (not singleton)? (Hint: e.g. Builder, Factory, Visitor, Chain of Command )
* Describe the observer pattern and an example of how it would be used (Hint: e.g. event notification when model changes to view )
* What are Use Cases? (Hint: Define interaction between actors and the system )What is your understanding of encapsulation? (Hint: Encapsulate data and behavior within class )
* What is your understanding of polymorphism? (Hint: Class hierarchy, runtime determine instance )

**Process**

* Have you heard of or used test-driven development? (Hint: e.g. XP process )
* What development processes have you followed in the past? (Hint: Rational, XP, waterfall )
* How do you approach capturing client requirements? (Hint: Numbered requirements, use cases )
* What process steps would you include between the capture of requirements and when coding begins? (Hint: Architecture, Design, UML modeling, etc )
* How would you go about solving performance issue in an application? (Hint: Set goals, establish bench, profile application, make changes one at a time )
* What developer based testing are you familiar with (before system testing?) (Hint: Unit test discussion )
* How might you test a business system exposed via a Web interface? (Hint: Automated script emulating browser)
* What is your experience with iterative development? (Hint: Multiple iteration before release)

**Distributed Application**

* Explain a typical architecture of a business system exposed via Web interface? (Hint: Explain tiers (presentation, enterprise, resource) Java technology used in each tiers, hardware distribution of Web servers, application server, database server )
* Describe what tiers you might use in a typical large scale (> 200 concurrent users) application and the responsibilities of each tier (where would validation, presentation, business logic, persistence occur). (Hint: Another way of asking same question as above if their answer wasn’t specific enough)
* Describe what you understand by being able to “scale” an application? How does a J2EE environment aid scaling? (Hint: Vertical and Horizontal scaling. Thread management, clustering, split tiers )
* What are some security issues in Internet based applications? (Hint: authentication, authorization, data encryption, denial service, xss attacks, SQL injection attacks )

**General**

* What configuration management are you familiar with? (Hint: e.g. CVS, ClearCase )
* What issue/tracking process have you followed? (Hint: Want details on bug recording and resolution process).
* What are some key factors to working well within a team? (Hint: Gets a view on how you would work within interviewer’s environment.)
* What attributes do you assess when considering a new job? (what makes it a good one)? (Hint: Insight into what motivates you.)
* What was the last computing magazine you read? Last computing book?
* What is a regular online magazine/reference you use? (Hint: Understand how up to date you keep yourself.)