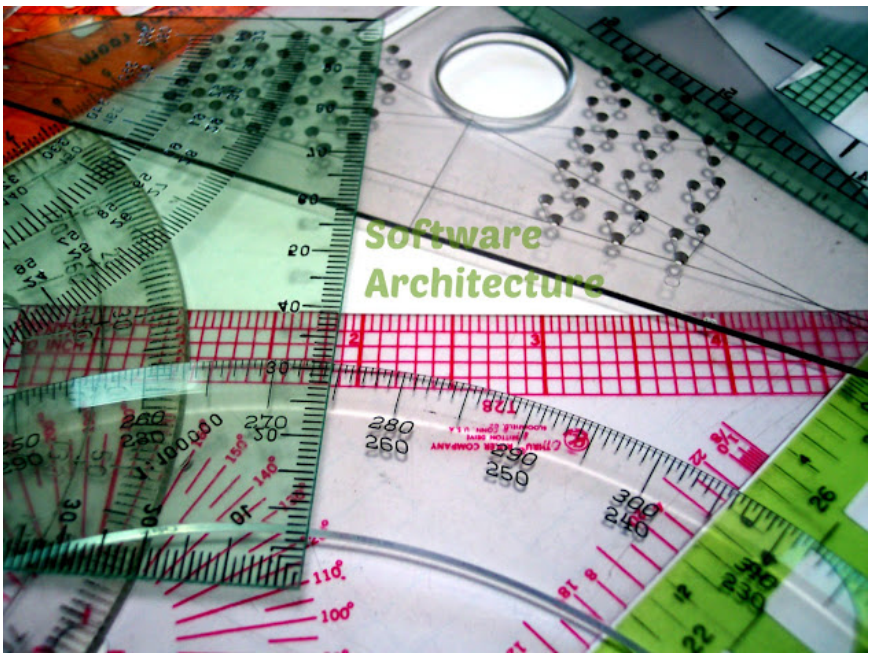
FROMDEV

20 Software Architect Questions & Answers for Scalability

http://www.fromdev.com/2013/07/architect-interview-questions-and-answers.html



Web applications scalability is a common problem most of the web architect face. Any internet facing web application may require to be highly scalable due to heavy load of traffic. Now a days, developing a smart web application is much more than creating dynamic Web pages. Irrespective of programming languages like Java, PHP, .NET, Python, Ruby or others, these challenges are faced by software architects everyday.

Discover Software Scalability Application software Web application

As the web is growing, our **need of building larger and more scalable applications** is also becoming more important.

In this decade, lot of distributed web applications are being developed that can utilize the resources from multiple machines, by separating the application functionality into manageable group of tasks that can be deployed in a distribute systems. There are numerous benefits to dividing applications this way, some of the most important are re-usability, scalability, and manageability.

Discover Java (programming language) Programming language Server (computing)

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.

If you are aspiring to become an architect then you must also check some of the **best books available for software architects**

Java Interview Preparation Tips

- Part 0: Things You Must Know For a Java Interview
- Part 1: Important Tips To Learn Java (For Beginners)
- Part 2: Core Java Interview Questions
- Part 3: JDBC Interview Questions
- Part 4: Collections Framework Interview Questions
- Part 5: Threading Interview Questions
- Part 6: Serialization Interview Questions
- Part 7: Classpath Related Questions
- Part 8: Java Architect Scalability 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
- Amazon CTO about 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.

I hope you find these questions useful in your software architect profession. What is the questions you would ask if you are interviewing an architect?

Posted by ***Sachin FromDev***

POST A COMMENT

DEFAULT COMMENTS FACEBOOK COMMENTS

27 comments

Add a comment as Tejas Joshi

Top comments

Sachin FromDev via Google+ 3 years ago - Shared publicly

Web application scalability techniques have been around for several years however a very few people have clear understanding of each of these concepts. If you are hiring a software architect, dont forget to ask these questions.

I am sure there are many more things you may want to ask before these questions however if your

+3 1 · Reply

Sachin FromDev via Google+ 5 months ago - Shared publicly

http://www.fromdev.com/2013/07/architect-interview-questions-and-answers.html

+1 1 · Reply

Sourced 9 months ago - Shared publicly

Scalability is often a major difficulty for IT professionals. These 20 questions and answers are a must read for Software Architects.

+1 1 · Reply

FromDev via Google+ 1 year ago - Shared publicly

Interview questions and answers for software architects. Mainly focused on web application and scalability.

+1 1 · Reply

Shweta Tandon via Google+ 1 year ago - Shared publicly

FromDev originally shared this
Interview questions and answers for software architects. Mainly focused on web application and scalability.

+1 1 · Reply

Sachin FromDev via Google+ 1 year ago - Shared publicly

FromDev originally shared this
Interview questions and answers for software architects. Mainly focused on web application and scalability.

1 · Reply

Sachin FromDev via Google+ 1 year ago - Shared publicly

FromDev originally shared this
Interview questions and answers for software architects. Mainly focused on web application and scalability.

+2 1 · Reply

Sachin FromDev via Google+ 1 year ago - Shared publicly

FromDev originally shared this
Interview questions and answers for software architects. Mainly focused on web application and scalability.

+2 1 · Reply

kiriti mishra 1 week ago - Shared publicly
Thanks for the notes. Really helpful!

1 · Reply

Dhanya KP (Devu) 1 month ago - Shared publicly
very informative

1 · Reply

Aarif Javadeveloper 5 months ago - Shared publicly
www.javatechnologycenter.com one shop for all java,j2ee interviews

+1 1 · Reply

Katam Chinna 6 months ago - Shared publicly
Really Good article. Keep up the good work. All the best

1 · Reply

sonu agarwal 6 months ago - Shared publicly
Well written and very informative.

1 · Reply

Pawan Gupta 10 months ago - Shared publicly
Nicely explained insight.

1 · Reply

FromDev via Google+ 3 years ago - Shared publicly
I had been thinking about writing this for long time. Web application scalability is a lot of times taken for granted however not many people really understand how to really scale a web application when big surge of traffic starts coming.

A lot of recent frameworks and technologies have evolved around these concepts and many people

+1 1 · Reply

Clement Amarnath 1 year ago - Shared publicly

Good one, Thanks for the collection

1 · Reply

arpit mithal 7 months ago - Shared publicly

Diligently done :)

1 · Reply

Pavel Tonev 1 year ago - Shared publicly

Really valuable and concise article!

1 · Reply

Chaitanya Singh shared this via Google+ 2 years ago - Shared publicly

+1 1 · Reply

Dan Pritchett 3 years ago - Shared publicly

Nice summary of most of the scalability concepts.

+1 1 · Reply

Show more

...