Research Report

COS30041 Creating Secure and Scalable Software (Swinburne University of Technology)

Abstract

Easy access to internet and increasing number of smart devices have increased the number of people who visits websites tenfold. Ranging from e-commerce websites to online gaming platforms, massive numbers of people being online all the time. More user equals to more request for the web server to handle. During peak hours maintaining sufficient resources just to meet peak requirements can be very costly for any organization. Therefore, preparing for such an event early on is a must, this includes making proper use of all the resources available, making sure nothing is causing serious bottleneck or even optimizing queries to the database.

Introduction

Success of any web application depends on many factors, security and scalability are two of the most important ones. Web applications are by design feed on inputs whose source can be untrusted, perform numerous security-sensitive operations, expose data to potentially malicious observers [1].

No one wants their sensitive data to be exposed to the world. And as we discussed just now, web applications are easily exposed to untrusted sources and unethical personnel. Therefore, keeping a close eye to the security pitfalls of a web application is very important.

Ranking higher in Google's search is one of most effective ways to promote a website/search. Google have their own method to rank websites. Though google is not entirely open about what they are looking for in a website, they prefer HTTPS over HTTP all day long. Google will not rank a website unless it thinks that the website is clear from any malware and can protect the user data.

On the other hand, scalability is important to keep the behavior of an application the same as load (number of users) increases. If a web application is not designed keeping scalability in mind, after a certain number of users, either the bandwidth, CPU, RAM or something else will run out, and requests will take longer to serve. Slower application makes bounce rate (percentage of visitors who enter and then leave) go higher. BBC discovered that they lost 10% of their total users for every additional second it took their pages to load [2].

There many strategies that can be applied to scale a website, implementing them properly is crucial for success of any application.

This paper discusses common scalability and security issues and common strategies and patterns that can be applied for the SwinBlog website.

Literature Review

The literature on Securing and Scaling web applications show variety of approaches (Horizontal scaling, vertical scaling, scaling on the database side, scaling on the server side, frameworks for security and others). The focus of this research paper has been widely researched. Cloud computing tools and architecture has been largely used by the industry to solve common scaling issues. Several authors [1], [3], [4] have discussed thoroughly about their used methodology in this regard.

Scaling in-house

[4] proposed scaling the application and the database separately. Scaling vertically is recommended at the initial stage, where we add more RAM, upgrade to faster processors, etc. When vertical scalability is no longer practical, scaling horizontally is recommended.

The solution that is proposed by [4] is entirely owned by the organization. Meaning, they will be handling the specifications of each computer and the total number of computers. It is a good option for bigger organization who have the means to buy such hardware and have a team to manage the whole system.

Scaling Web Applications with Cloud Platform
In order to meet peek requirements, cloud platform is utilized [3], and in the front-end load-balancer for routing and balancing user requests to web application is discussed. The primary goal is to achieve on-demand scalability by issuing more resources during peak requirements. [9] describes possible scaling solution in cloud platform with Heroku.

Heroku, which is a cloud platform that natively supports Rails applications is suggested as one of the cloud computing platforms. Heroku provides Dyno (instance of a server) to run an application. When there is peak in traffic more Dynos can be added automatically by Heroku to meet the increased demand. Heroku has multiple scaling option,

- Horizontal: adding more dynos when there is increased number of requests per minute.
- Vertical: increase dyno size. When there are heavy web processes, which consume more memory, may require larger dyno type
- Process types: Heroku has the option to scale individual process types independently. Such as we can have a process type to perform search and another process type for image processing.

Heroku is a subscription-based platform and the user can subscribe for more Dynos whenever needed.

Optimizing queries
Another approach in scaling, is optimizing the process, more specifically the queries to the database. [8] has suggested on optimizing the queries to the database. When there is a lot data to be queried through, and a lot of requests on doing similar queries, the database can be a huge bottleneck. Therefore, optimizing queries is essential.

It is common to have multiple SQL tables to be joined together, whenever we join a table it is like having a nested for loop, which can lead to have complexity from O(n) to O(n^m), where ***m*** is the number of nested loops. Therefore, having multiple Joins is never a good thing. But we cannot avoid joining tables, therefore we should use them wisely and try to optimize them as much as possible. Some recommended optimizations:

- If there is any “where” clause then use it before joining table. It would reduce the number of records thus joining table would be on less amount of data
- When joining multiple tables, think about the order of the tables, always keep the table with the least amount of data on the left and keep the table that has more data on the right side and perform a left inner join.
- Create **Views** for common queries and save them for reuse. This would greatly improve search time

Finding Bottleneck!
Without knowing what is causing the bottleneck, trying to optimize an application for performance improvement is the same is shooting in the dark.
[7] suggested to find the bottleneck before trying to improve the performance of the system. The following is suggested:

- Implement sufficient application performance monitoring. (JMeter)
- Gather data (ideally, load test)
- Identify what the bottleneck was
- Fix the bottleneck
- Repeat (until you hit the desired responsiveness)

The author suggests using a profiling tool to see how much time is spent in each part of the application, how much memory is used and how long it took for certain query to be performed.

Partitioning Database

[10] suggests partitioning table when the table is too big! This option can greatly help performance of the database as well. Partitioning divides large tables into multiple smaller tables. Queries on a smaller table is much faster. This is useful when one table gets too big and starts to take too much time for any query. Google's File System (GFS) use similar concept, where it stores data in small chunk and store it over multiple nodes and replicates it multiple times for more availability. A master node keeps track (index) of which chunk resides in which slave node.

Secure Web Application: Best Practice

[5], [6] provides comprehensive guide on common security issues and how to prevent them. [5] suggests, organization should have a web application security plan, which should outline the organization's goals. [6] provides a comprehensive guide on common security threats and how to take measure against them using the Rails framework.

Findings

Creating secure and scalable application is a matter of pre-planning. It covers everything from system architecture to database design. Therefore, proper planning considering security and scalability should be in place for early development stage. For my application SwinBlog here are the scalability aspects that can be taken:

Hosting Web Application

Hosting can be done both in house and online. Currently I am using my own laptop to host SwinBlog locally. If I want to make it public and expect a thousand visitor per minute, then definitely my system would crash. I can follow [4]'s solution direction and build my own server system. But considering this is a blog website and I don't have a team to manage a server system, I prefer [3]'s solution. SwinBlog can be hosted on Heroku, which is a cloud platform that natively supports Rails applications. Instead of creating my own web-server architecture with NGINX, Heroku can provide all the necessary processing power to host my application. Initially it might be easy to scale with in-house vertical scaling, where we can buy faster RAM, SSD and processors, but eventually we have to move on to horizontal scaling. In such scenario we all add more nodes (computers) to our initial architecture and make a cluster. Considering SwinBlog is a simple blogging website, there is no need for such complex solution. Online platform such as Heroku, which provides dyno (instance of a server) to run the application is sufficient to host. When there is peak in traffic more dynos can be added dynamically by Heroku. Heroku has in-built load balancer as well to balance the load between dynos. It is a simple and smart solution for many small to mid-size organizations.

Optimizing Database

SwinBlog does not have very complex SQL algorithms, but there a few moderately complex queries which are filtering, and searching through blogs from different channel. In the future there can be more complex queries where a user can filter through a individual user who wrote all the blogs under a certain section. In such scenarios I can keep in mind the optimizing suggestion and filter out

all the result for particular user id first, and then perform join between other tables. In general, the suggestions are good for optimizing any database.

Partitioning Database
When there will be hundreds of thousands of users in SwinBlog and similar number of blog posts, methods such as partitioning and replication can greatly help scale the system to perform better in terms of raw query time.

Finding Bottleneck!
I can use JMeter to perform performance analysis of SwinBlog and get more accurate representation of how my system will behave when there are is lot of RPMs. Different scenarios such as viewing a blogpost, registering a user, searching for a blogpost etc. can be test plans. Stress testing the system would help us find where we need to improve our process.

Securing Web Application
Following [6] it can be said that; Rails framework provides as many countermeasures for attacks as possible. Following their guide on different attacks I have already implemented countermeasures for major attacks such as, XSS, SSL, reCaptcha, Session Fixation, Cross-Site Request Forgery (CSRF), Privilege Escalation and SQL Injection. The detailed report is included in Design Documentation.

It is important to keep in mind, there is no one single solution that would prevent all the attacks in the world. There is a constant support from the Rails community to further improve security of their framework. But there is always some loophole an attacker can take. Therefore, we should always keep ourselves updated about new attacks, always use the latest version of a framework/plugin and Keep servers up to date.

Conclusion

Different website requires different solution on how to scale them. Scaling a website is an art on itself. It requires proper planning and research (finding bottleneck) to perfect. Through proper planning of peak traffic numbers, most used queries, most complex queries it is possible to optimize them for the absolute best performance. On the hand, securing a web application is a constant battle, there is no one solution to all attacks. Attackers are always trying to come up with new hacks to get into the system, therefore, it is important from our side that we are cautious about what we are allowing for the user to access, and constantly monitoring our system for any suspicious behavior.

References

1. Omer Tripp, Marco Pistoia, Patrick Cousot, Radhia Cousot, and Salvatore Guarnieri.: ANDROMEDA: Accurate and Scalable Security Analysis of Web Applications (2013), viewed on 8th December 2019
2. Why Does Site Speed Matter? | Improve Webpage Speed. Retrieved from https://www.cloudflare.com/learning/performance/why-site-speed-matters/, viewed on 8th December 2019
3. Trieu C. Chieu, Ajay Mohindra, Alexei A. Karve and Alla Segal.: Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment, IBM T. J. Watson Research Center, viewed on 8th December 2019
4. Sviatoslav A., Vlad V.: What if I Tell You That Ruby on Rails Is Scalable. Retrieved from https://rubygarage.org/blog/ruby-on-rails-is-scalable?fbclid=IwAR1NUoqFMI1w2WNUwtOxaxq0-Tf7fOqZWkSWO8pmVFkTNXlcY8s-Cl9rdow, viewed on 8th December 2019

5. Cody Arsenault.: Web Application Security Best Practices. Retrieved from https://www.keycdn.com/blog/web-application-security-best-practices, viewed on 9th December 2019
6. Securing Rails Applications. Retrieve from https://guides.rubyonrails.org/security.html, viewed on 9th December 2019
7. Scott Bartell.: Scaling a Ruby on Rails app on Heroku. Retrieved from https://scottbartell.com/2019/03/26/how-to-scale-ruby-on-rails-app-on-heroku/?fbclid=IwAR0QInxPC4bZpXengO4pWmtx7S4FZPl3Z9m-cBUrqfeScQWrSGrqZTWnj74, viewed on 9th December 2019
8. Apache Software Foundation.: Performance and Efficiency, Performance Enhancers. Retrieved from http://pig.apache.org/docs/r0.10.0/perf.html#performance-enhancers/, viewed on 9th December 2019
9. Adam.: Mastering Heroku, 4 ways to Scale on Heroku. Retrieved from https://masteringheroku.substack.com/p/4-ways-to-scale-on-heroku, viewed on 9th December 2019
10. Kendra Little, How To Decide if You Should Use Table Partitioning. Retrieved from https://www.brentozar.com/archive/2012/03/how-decide-if-should-use-table-partitioning/, viewed on 9th December 2019