Technical Analysis of Advanced Web Applications

A focus on the technology of Reddit

Adam Lloyd, R00117318
BSc. Hons Web Development
Cork Institute of Technology
Co. Cork, Ireland
adam.lloyd@mycit.ie

Abstract - **With increased user connectivity, web applications are becoming the main way people consume and contribute information. Traditional web based content is no longer sufficient to fulfil this need so increasingly complex web based applications are required as is the technology to support and develop them.**

This document will explore the technologies required to enable this advanced functionality and compare the technology choices made by the chosen application, Reddit, to those made by other notable web applications. Analysis of these technologies, the reasons for choosing them and current trends will reveal how modern web applications are developed to meet the speed, reliability and functionality required.

***Index Terms*— Web Application, LAMP, MEAN, WISC, Python, SQL, Apache, JavaScript.**

I. Introduction

Web 2.0 was a term first used in 1999 [1] in an article entitled "Fragmented Future" in which the author states, "The Web will be understood not as screenfuls of text and graphics but as a transport mechanism, the ether through which interactivity happens." This transition, oft referred to in recent times as the Web 2.0 revolution [2], describes the departure from static websites that simply served content to the user to consume to functionality like that of desktop applications. These Web 2.0 applications or simply Web Applications, allow the user to contribute to the sites content (blogs), to store their own data in them (cloud storage) and to use them as a communication tool (social media).

What enables this increased functionality is the underlying technology, no longer simply a physical machine with a web server serving up the content to the user but a collection of technologies supporting that web server. From advanced client side frameworks using JavaScript to update the web application in response to a user's actions without reloading the page to the decentralized cloud storage allowing for access to data all over the world. From the desktop web browsers to the wearable tech devices, we are now more and more connected whoever we go and this is largely due to the advanced web applications of the web 2.0 revolution.

II. Modern Web Applications

The architecture for a modern web application primary consists of 3 layers. The database which persists the data for the application, the client side code which presents the content to the user and the server side code which handles communication between the client and the database. This is commonly referred to in software development as the model view controller (MVC) pattern and is used to separate code into sections to improve maintainability, however it also represents how the underlying technology of modern web applications have come to be structured.

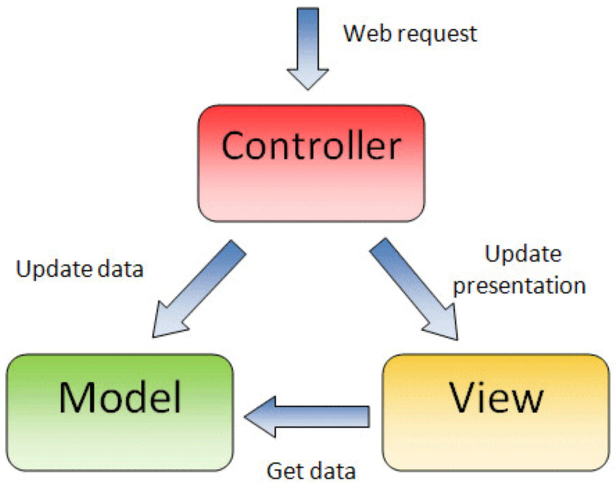

Fig 1: MVC Pattern Diagram
[http://www.beansoftware.com]

The principles of the MVC pattern are such that the functions of each component are separated, in code this simply means different files or directories but in a modern technology stack it is more clearly defined as separate technologies are used for each function.

Common technology stacks are Linux, Apache, MySQL and PHP (LAMP) and Mongo DB, Express.js, Angular.js and Node.js (MEAN) and, along with the operating system that is hosting them, show a clear separation of function.

A. The Model

This is the data storage layer and will commonly be a database management system (DBMS) such as MySQL or Postgres which will store much of the data which is displayed to the users of the application. This system will be hosted on a web server, often the same one as the rest of the application, and communication with the database will be done over network protocols and ultimately be displayed to the user in the view.

B. *The View*

This is what the users of the application will see, a seemingly conventional web page rendered in a web browser, displaying all the content that the user needs to see to use the application. All the user's interactions with the application will be completed through the view often not even requiring page reloads with modern client side technologies using JavaScript. What is displayed in the view is determined by the controller.

C. *The Controller*

The controller is the glue that binds the technology stack together, it is the server side code that controls everything. When the view is updated with a user's action such as leaving a comment, the controller processed this event and communicates with the model to save the data and then updates the view to reflect that change in the browser. The control is where the largest variety of technology choices lie as this is the main technology that will define the application. In the past PHP was the main choice for this language but in recent times C#, Python, JavaScript and Ruby to name but a few have overtaken in popularity.

D. *The Server Environment*

These technologies would be useless without a way for them to communicate with the world and, as such, require a web server to host them. The web server runs the server side code and the DBMS and serves up the view to the user. Traditionally a UNIX based systems running a flavour of Apache is used due to its reliability and security but Windows running Microsoft's IIS is also a viable option.

III. Reddit

The application this paper will focus on is Reddit [3], a web application which is entirely generated from the content its users create. Reddit allows users to post links and other content that other users can view and comment on, it has become the primary way that many internet users get their news and go to have fun. The user generated content is separated into user created sub sections called "Subreddits" which keep the numerous topics separate and allow users to discuss topics they are interested in.

With over 234 million users and over 800k Subreddits [4], Reddit's technology infrastructure needs to be able to handle all this traffic and with its massive popularity it must integrate with many other web and native applications. It is with this in mind that the underlying technology behind this simple but effective web application will be analysed and the reasons behind their choices theorised.

A. *The Tech Stack*

Initially created using LISP, Python now fills the role of the controller for Reddit as it is the primary server side language which handles the communication between the Backbone.js powered client (view) side code and the Postgres database (model). These technologies are housed in the cloud on an AWS instance running UNIX which serves up the application with a Gunicorn, Python based web server [5].

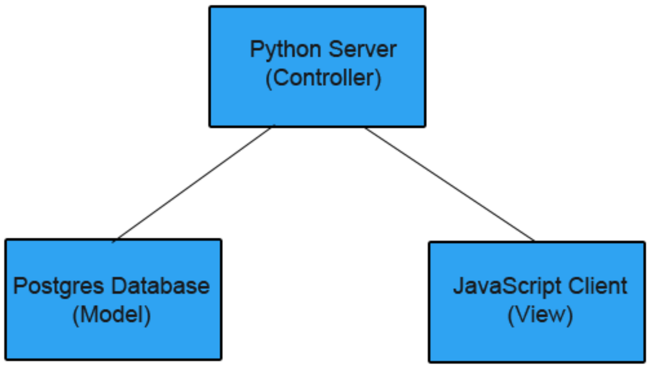

Fig 2 Basic Reddit Technology Stack

Usually the technology stack of an application is easily communicated with an acronym containing the letters of the technologies used, the most common being the LAMP stack and the increasingly popular pure JavaScript MEAN stack. This is not the case for Reddit but the components do resemble the LAMP and MEAN stacks in that they all follow the pattern of a model, controller, view and a hosting environment so there is little variation in the overall architecture at a very high level. Delving a little deeper however, into the specific technologies used, the different choices applications make display a large degree of variety.

IV. Server Side - Python

Traditionally server side code is written in PHP with it still being the most popular technology for web applications as it is the oldest and most well-known language for this purpose used, most notably, by Facebook.

Python was primarily chosen for Reddit as it was a language that the development team were familiar with [6] but its benefits far outweigh that of circumstance. Python embodies the principles that code should be easy to read and quick to develop in [7] making it a solid choice for web development.

The reasons Facebook adopted PHP are much the same as Reddit's were for choosing Python, that their developers were experienced with the language and I believe this to be a common these for many web applications. While Reddit was initially created with LISP, when the demands of the application exceeded what LISP could efficiently provide they rewrote it in Python as it better suited their needs. This was not the case for Facebook however, as the needs of the largest web application in the world exceeded the capabilities of PHP and maintaining it became problematic, instead of rewriting the application in another language, they opted to rewrite PHP itself. They created HACK [8], a new PHP driven language to make clunky PHP task simpler and make the code more maintainable going forward.

This is a stark contract of approaches in development, one adopting a new language which suited their needs, the other building on top of the old language to create the functionality they needed. It is quite certain that Facebooks developers are more than capable of adapting to new advances and trends with web technology but I feel the move to adopt a new language entirely is the most prudent as, in Reddit's case, it allows them access to the wealth of libraries and support for the language where Facebook, having created HACK would be the only source of information for it.

With both PHP and Python being strictly server side languages, they are designed to send information back and forth from the server to be viewed in the client by rendering out web pages for the user to view. With current web trends leaning to that of single page web applications that carry out complex tasks, sometimes in parallel, simply making a request to the server side code to generate a page from the stored content is insufficient. These tasks, for example sending a chat message to another user, are performed without refreshing the page and conventional server side languages do not support this functionality which is now essential for any web application. This is where the client side code takes over.

V. Client Side - JavaScript

Of all the technologies involved in a web application, the client side code is the most consistent as no other language is as widely supported by web browsers or as suitable for the task. JavaScript, which runs inside the client web browser, can update sections of the application without the need for the server to refresh the entire page by performing asynchronous tasks and simply updating the relevant data in the browser, providing near instant interactions for the users using Asynchronous JavaScript and XML (AJAX) [9].

JavaScript alone, while able to perform these asynchronous tasks, is just not practical as the quantity of code quickly becomes unreadable and difficult to maintain. It is for this reason that that majority, if not all, modern web applications opt to use a JavaScript framework to modularise and simplify the code base. Reddit uses Backbone.js for this purpose and with its focus on modular code in the MVC pattern separates the script files into different directories each with a specific purpose, one script might handle calls to an API while another would display the returned data to the client [10].

While a framework such as Backbone provides a large amount of functionality as do many others such as Angular and React, it is not without its overheads. Much of the code generated by these frameworks, as discussed earlier, is to compensate for the shortcomings of the server side language so, much like Facebook developing HACK to make PHP work, the popularity of JavaScript frameworks is somewhat a symptom of inadequate server side languages.

These inadequacies, some might say should be fixed, and Microsoft's ASP.NET framework, is trying to solve this issue by enabling users to build out an application based entirely on Microsoft technologies which integrate seamlessly. The Windows, Internet Information Services, SQL Server and C# (WISC) stack is their offering and takes advantage of the ASP.NET framework which can perform asynchronous tasks [11] to remove the need for advanced JavaScript frameworks leaving merely the need for simple jQuery scripts to add some flair to the interface if desired.

This solution may sound perfect but it is not without overhead in the form of cost. The widely available and cheap, sometimes free, UNIX based servers, open source software and database management systems are replaced by proprietary, licenced products from a large software corporation so the decision is not a simple one to make. One web application that takes advantage of this architecture is Stack Overflow [12], one very similar in scope to Reddit but behind the scenes there is likely to be a higher degree of integration with the ASP .Net implementation than the conventional LAMP, or similar, stacks which would imply a greater degree of maintainability. To the user however, there is very little difference in their performance as all the considerations being discussed will never once enter the mind of the average user. This is especially true of the most hidden technology, the database.

VI. The Database – Postgres

To store the millions of posts that makes up the content of Reddit, they use Postgres. This is the relational database management system(DBMS) which stores all the critical data while Cassandra, a distributed fast caching database, is used to store user comments and chat as it is more suited to rapid high volume queries [5].

The choice to use Postgres is a departure from the conventional MySQL and is likely due to its focus on extensibility and automation with advanced stored procedures which can be saved and used to simplify complex and regularly repeated database operations. With this, it would be well suited to manage the massive number of tasks that the DBMS runs daily to maintain order in the chaos of a user generated message board application and keep the data in line for the various ranking and sorting algorithms used by Reddit [13].

Relational databases are not, however, the only option for modern data storage. Along with the rise of Object Relational Mapping (ORM) frameworks which enable the data to be created as a programming object and mapped to a database easily and intuitively, a new type of data store is increasing in popularity. This is the Not Only SQL (NoSQL) database. Within these data stores, the information is not stored in conventional tables, but rather in a vast associative array in which data is matched by key value pairs. This allows for faster data retrieval as the query does not have to traverse multiple tables, but rather, just the one. It is with this that the main overhead with NoSQL comes into play, you are getting everything. When data is organised in relational tables, the fields themselves can be reduced to ID integers with which to reference other related tables. When the actual fine details of that related table are not needed, only the ID integer is retrieved thus resulting in less data being transferred from the query. While the argument can be made that data transfer should be kept to a minimum, in modern times the average internet speed of users is growing and growing and frankly, a few extra characters is not going to break the bandwidth bank [14].

Of course, there are other considerations, development with a NoSQL database is notably faster as multiple tables do not have to be created but rather the object schema within the program code. This makes it a great choice for quickly building out and scaling applications and with some internet giants like EBay and the SAP business CMS [15] using it along with its popularity in the big data market it is becoming a more common choice for web applications big and small.

The conventional MySQL is still an extremely efficient product and continues to be the mainstay of web applications due to its ease of use and availability, being supported by default on nearly all UNIX based web servers.

VII. Server architecture – UNIX/Gunicorn

This is a topic that has been touched on already in this document and regarding the operating system running the server, is most certainly, outside of varying UNIX distributions, the area in which there is least disparity between applications. This is primarily due to UNIX based systems being world renowned for their stability and security and these being the two major concerns for web based applications. As already mentioned Windows is beginning to gain popularity but its restrictive pricing and lack of compatibility with conventional web development technologies keeps it on the side-lines for now.

While server software varies somewhat more, Apache is by far the most popular web server software serving as the backbone of the LAMP stack. The reasons for its popularity are like that of the UNIX based systems that it conventionally runs on, it is cheap, secure stable and supports MySQL and PHP out of the box. Apache would have been a solid choice for Reddit's infrastructure but they chose to go with Nginx [16] a slightly more advanced web server featuring advanced load balancing and the ability to handle vast amounts of simultaneous connections making it an excellent choice for such a large-scale web application coming in at around 2.5 times faster than apache in general [17].

Nginx however does not natively support Python based applications so another component is required on the server to process requests intended for Python. This, in Reddit's case, is the Gunicorn web server, a native Python server which is part of the applications server side code. It may seem counter intuitive to have two web servers to do one job but, as has been seen in this document already, this is not uncommon in web applications. That is not to say that bridging software compatibility is the only reason Gunicorn is used however. It also plays a role in further increasing the load balancing capabilities of Nginx but handling the http requests for it [18]. Gunicorn listens for requests on, usually port 8000, with Nginx listening on the conventional port 80 so when a http result comes in on port 80, Nginx doesn't need to worry about what to do with it bar simply forwarding it onto port 8000 for Gunicorn to handle.

In the same way Gunicorn supports Nginx by taking over some of its responsibilities, there will always be a selection of supporting software making all the proverbial magic happen.

VIII. Supporting software

Load balancers such as HAProxy, distributed databases like Cassandra, content delivery networks(CDN) like CloudFlare all add small improvements and edge out those last bits of performance from a web application.

IX. Conclusion

With advancements in mobile technology making us more and more connected wherever we are, the demand for fast, reliable applications is greater than ever. With increased demand for applications there is also an increased demand for the technology to create them giving developers a large amount of choice for each section of their application.

With the increasing popularity of WISC and MEAN stacks, current web trends are indicating that the increased integration and simplicity gained from using one language throughout is very desirable. That is not say that multi language solutions will be a thing of the past as common factors for development choices will remain for some time to come and people are slow to adopt new technologies and companies are slow to purchase the software to support them.

From basic PHP to modular JavaScript frameworks, from relational databases on a web server in an office to object oriented NoSQL databases hosted in the cloud, finding the right balance of maintainability, speed and cost is key. Whatever the choice, at the end of the day the application is designed to fulfil a user's needs and they will never know what lies behind the wizard's curtain.

References

[1] Wikipedia, "Web 2.0," Wikipedia, 15 10 2016. [Online]. Available: https://en.wikipedia.org/wiki/Web_2.0. [Accessed 15 10 2016].

[2] R. Lentejas, "The Web 2.0 Reolvution," Intechnic.com, 15 10 2016. [Online]. Available: https://www.intechnic.com/blog/the-web-2-0-revolution. [Accessed 15 10 2016].

[3] Reddit, "Reddit," Reddit, 15 10 2016. [Online]. Available: https://www.reddit.com/. [Accessed 15 10 2016].

[4] C. Smith, "By The Numbers: 60+ Amazing Reddit Statistics," Expanded Ramblings, 15 10 2016. [Online]. Available: http://expandedramblings.com/index.php/reddit-stats. [Accessed 15 10 2016].

[5] "Siftery," Siftery, 16 10 2016. [Online]. Available: https://siftery.com/company/reddit. [Accessed 16 10 2016].

[6] A. Swartz, "Rewriting Reddit," 16 10 2016. [Online]. Available: http://www.aaronsw.com/weblog/rewritingreddit. [Accessed 16 10 2016].

[7] P. Krill, "A developer's guide to the pros and cons of Python," Info World, 24 02 2015. [Online]. Available: http://www.infoworld.com/article/2887974/application-development/a-developer-s-guide-to-the-pro-s-and-con-s-of-python.html. [Accessed 16 10 2016].

[8] S. Melendez, "Why Facebook Invented A New PHP-Derived Language Called "Hack"," Fastcompany.com, 04 07 2014. [Online]. Available: https://www.fastcompany.com/3028778/why-facebook-invented-a-new-php-derived-language-called-hack?show_rev_content. [Accessed 16 10 2016].

[9] “Getting Started,” MDN, 16 10 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started. [Accessed 16 10 2016].

[10] Backbone.js, 17 10 2016. [Online]. Available: http://backbonejs.org. [Accessed 17 10 2016].

[11] S. Hanselman, “The Magic of using Asynchronous Methods in ASP.NET 4.5 plus an important gotcha,” Hanselman.com, 29 08 2013. [Online]. Available: http://www.hanselman.com/blog/TheMagicOfUsingAsynchronousMethodsInASPNET45PlusAnImportantGotcha.aspx. [Accessed 17 10 2016].

[12] N. Craver, “Stack Overflow: The Architecture - 2016 Edition,” Nickcraver.com, 17 02 2016. [Online]. Available: http://nickcraver.com/blog/2016/02/17/stack-overflow-the-architecture-2016-edition/. [Accessed 17 10 2016].

[13] A. Salihefendic, “How Reddit ranking algorithms work,” Medium.com, 08 12 2015. [Online]. Available: https://medium.com/hacking-and-gonzo/how-reddit-ranking-algorithms-work-ef111e33d0d9#.unpnt12xt. [Accessed 17 10 2016].

[14] D. Ocean, “SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems,” Digital Ocean, 21 02 2014. [Online]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Accessed 18 10 2016].

[15] “Flexible enough to fit any industry,” MongoDb, 17 10 2016. [Online]. Available: https://www.mongodb.com/who-uses-mongodb. [Accessed 17 10 2016].

[16] "NGinx,” 17 10 2016. [Online]. Available: https://www.nginx.com/. [Accessed 17 10 2016].

[17] R. Frankel, “NGINX vs. Apache (Pro/Con Review, Uses, & Hosting for Each),” 09 05 2016. [Online]. Available: http://www.hostingadvice.com/how-to/nginx-vs-apache. [Accessed 18 10 2016].

[18] P. Lakshmanan, “Quora,” 01 03 2016. [Online]. Available: https://www.quora.com/What-are-the-differences-between-nginx-and-gunicorn. [Accessed 18 10 2016].