Prospectus

An Open Source Compare and Contrast Project For Analyzing Performance of Commonly Used Software Architectural Patterns
(Technical Topic)

A Study On The Factors that Contribute To Successful Open Source Software
(STS Topic)

By

Sai Konuri

10/22/2019

Technical Project Team Members: Aman Garg, Vineeth Gaddam

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: Sai Konuri

Approved: _____ Date _____
Ben Laugelli, Department of Engineering and Society

Approved: Nada Basit Date 05/06/2020
Nada Basit, Department of Computer Science

Introduction:

Given the rise of access to the internet for millions of people around the world, the demand for speed, reliability, and scalability has never been higher (Mihaela, & Gorgan, 2016). Web sites are getting larger and more feature rich every year (Wagner). As a result of this, expectations from users have increased exponentially. At the core of these expectations is better performance (Wagner). Having performance issues could potentially make an application inaccessible on any platform. Pinterest found that more people used its search engine and signed up as a result of reduced wait time (Meder, & Antonov, & Chang, 2017). DoubleClick by Google found that more than 50% of mobile site visits were abandoned if a page took longer than 3 seconds to load (Wagner). Given that performance is a recurring challenge that many software architects face, we will design a technical solution that could potentially help them make informed decisions about improving application performance.

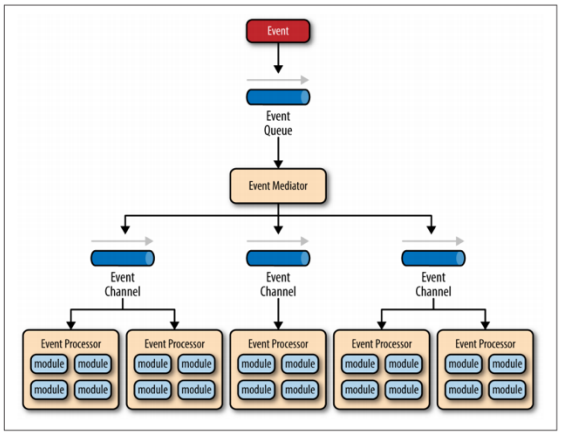
This technical solution that we will be implementing is an open source project that has three components. The first is an informative manual that outlines different performance results and considerations for commonly used architectural patterns. The second is a collection of identical applications implemented in different architectures, designed in such a way that others can extend it to test their own features. The last component is a readily available load testing pipeline. There are several benefits to this solution. The architect can now compare and contrast the performance metrics of the architectures, allowing them to adjust their considerations. A developer can also fork the project, since it is open source, in order to customize for a new architecture, new metric, or testing pipeline. This gives the foundation for planning future application development and weighing performance alternatives.

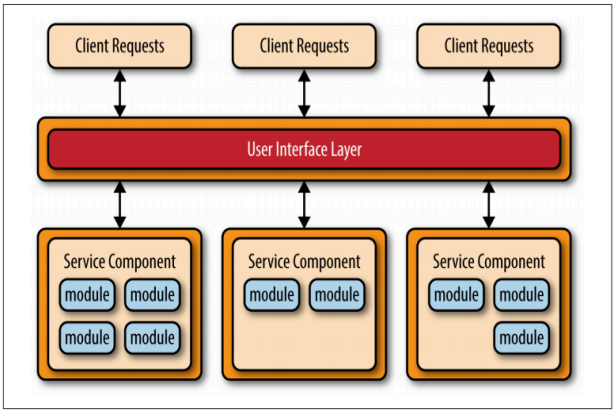
The technical solution alone cannot help improve performance if it cannot reach the architects. The lack of a strong network that harbors open source projects for improving application performance will make it difficult for the solution to be viable. One needs to understand better that there are multiple actors in the ideal open source network. These actors include backing foundations, public administrators, licenses, a user community, developer incentive, and competing projects. By focusing only on the technical solution, all of the necessary social actors will be neglected, thereby causing the project to not reach the desired level of success. The socio-technical problem of meeting consumer demand for performance, speed, and reliability consists of both the open source technical solution and the understanding of these different social factors. The technical solution provides a manual and an interface for developers to explore performance considerations for different architectures. The understanding of the social factors provides a better analysis of the network that is required for the technical solution to reach the target user. Together, they help solve the larger problem of improving application performance in a highly internet dependent world.

Technical Problem Frame:

Software architecture is at the heart of any modern software system's design and development. An architecture is a set of principal design decisions made for the development of some software (Medvidovic, N., & Taylor, R. N., 2010). It is the overarching structure and organization of the software. The figures below show examples of an event driven architecture pattern and a microservice architecture pattern. History of web applications shows that web application architecture has evolved immensely (Shaw, 2001). With the rise of the internet and dynamic web content, the demand for more robust architectures has increased. Although these

feature rich architectures have formed, the analysis of performance differences is limited in practice. Performance is a measure of how "well" the software system does under heavy traffic. Heavy traffic in this case refers to a high volume of requests and execution. In order to measure performance, there are several metrics including but not limited to: average response time, failure rate, CPU usage, and average connection time.





Although not applicable to all software systems, one common modern approach architects employ in improving performance is to horizontally or vertically scale the application. By horizontally scaling an application, the architect introduces additional processors and machines. (Bass & Clements & Kazman, 2015). By vertically scaling, the architect increases CPU power, RAM, and network bandwidth. Therefore, the analysis of performance is immensely limited in practice. Paying for more computing resources provides immediate attention to performance issues, but one simply cannot continually increase them as that becomes very expensive in the long term. By neglecting the performance of the architecture itself, one might overlook the possibility of a flaw in the underlying architectural design for an application.

The study of performance of different architectures is limited in practice. When software architects in the industry make architectural decisions for their business needs, their justification is typically limited to conceptual research, past experience, and other institutional limitations.

There are no concrete compare and contrast methodologies for the architectural options they have. Theorists like Mark Richards have laid out conceptual frameworks that outline how performance could be different for several architectural patterns in order to help architects make informed decisions (Richards, 2015). This, however, still does not provide a hands-on practical method of evaluating the performances of the architectures. By not having this practical method at their disposal, software architects are limited in their analysis of their choices.

Due to the lack of a practical compare and contrast methodology for assessing the performance of different architectures, we will build an open source project that measures the performance of a simple web application built with several commonly used architectural patterns. In addition to this project, we will be writing a manual that outlines the numerical results from load testing the architectures and report any suggestions for different cases. By being open source, the project will allow anyone to fork it and run similar load tests on their own applications and architectures. For the purposes of our research, we will be limiting the software we test to a simple web application. The simple web application is an online textbook catalog for students. This will be implemented with the following architectures: Layered, Event-Driven, Microservice, and 3-Tier. We chose these architectures since they are common in the industry (Richards, 2015). In order to gather metrics (average response time, failure rate, CPU usage, and average connection time), a load testing framework called JMeter will be used. Packaging the testing pipeline, the different architectures, and the performance results will help developers have an easily available resource to expand their exploration into improving performance for their applications.

STS Problem Frame:

One of the most interesting phenomena in the past century is the open source revolution in the software world. Open source software is software that is built for and by the users (Von Hippel, 2001). User communities collaborate with each other in writing the source code without a manufacturing agency overlooking them. We have seen open source become so successful that commercial software companies have adopted the open source software into their own developmental operations (Nagy, D., Yassin, A., 2010). In my research, I will be using a case study on the widely adopted open source web server called the Apache HTTP Server.

One might wonder how and why these user centric communities exist and how they get adopted by commercial giants. Some research scholars attribute a majority of the success of the Apache web server to the developmental practices in open source development (Mockus, 2000). In fact, it is even argued that the software development process in the Apache server project is the reason why commercial web servers were displaced. This is because defects are found and fixed very quickly due to large number of volunteers and "eyeballs." Code is also written more carefully and creatively due to the passion the programmers possess. There are proponents who support the claim that open source software, including the Apache server, is successful due to its quality matching up well with expensive commercial products (Mockus, 2000). Others such as Von Hippel argue that the strong community is the reason for its success. For him, a strong community is formed with three conditions: users have an incentive to innovate, users want to reveal their innovations, and the innovation should compete with commercial products (Von Hippel, 2001). Although these ideas and frameworks are consistent with successful open source software such as the Apache web server, it focuses only on the user community, the software quality, and the price. There are other factors missing in this argument. I will argue that the

Apache web server was widely adopted due to its backing by supporting foundations, public administrators, public licenses, and a strong user community (Gaudeul, 2003). By considering these social factors, one can better understand that the Apache web server was not driven merely by a large community base, but rather by a variety of institutions, people, and technologies.

To understand what it takes for open source to be widely accepted both individually and commercially, I will explore the case of the Apache HTTP Server. The Apache web server is the most deployed web server in the world, powering approximately 60% of the world's web sites (Von Hippel, 2001). Started by Rob McCool, a student from the University of Illinois, the source code was an individual exploration. He shared the source code during the rise of the web, which accrued several other users. Years later, the community has grown to thousands and the server is now widely used commercially. Using the Actor-network theory (ANT) as a framework, this study will attempt to understand how this open source project became an interconnected system of different actors to form a stable network. In an actor-network, both human and non-human actors have agency and contribute to the network's stability. A network builder initially recruits new actors by aligning their interests with the goals of the network. One can also determine the strength of the interconnectedness of different actors. The paper will identify the network builder, the key actors, and the recruitment of actors within the Apache open source project. It will analyze how the network had grown and how the actors expressed agency. ANT provides a way to highlight the human, non-human, societal, and economic factors that contributed to the continued success of the open sourced Apache web server.

Conclusion:

In order to solve the broader problem of helping software architects meet the performance demand that internet users have established, there are two aspects that need to addressed. The first is the technical solution. The solution outlines performance considerations for different architectures. It packages a collection of identical applications implemented with different architectures. It provides a readily available testing pipeline to make it more flexible for newer architectures. This is an open source project in order to allow developers to either use it as a manual or customize it as they wish to meet their needs and make informed decisions. The second is the understanding of different social factors that inhibit such a technical solution from being successful. Using a case study of the open source Apache HTTP Server project, the study will utilize the actor-network theory to identify the network builder, actors, and inhibitors of a successful open source project. It will attempt to understand human, non-human, societal, and economic factors that play a role. Together, by implementing the technical solution and understanding the network of social actors, we can achieve the broader socio-technical goal of helping software developers and architects meet their performance needs.

Word Count: 1902

References

Dinh-Trong, T.T., & Bieman, J.M. (2005). The FreeBSD project: A replication case study of open source development. *IEEE Transactions on Software Engineering, 31*, 481-494.

Gaudeul, A. (2003). The (LA) TEX project: A case study of open source software. *TUGboat*, *24*(1), 132-145.

Len, Bass., & Clements, Paul., & Kazman, Rick. (2013). *Software Architecture in Practice.* Westford, MA: Pearson Education.

Meder, Sam., & Antonov, Vadim., & Chang, Jeff., (2017, March, 3). *Driving user growth with performance improvements.* Retrieved from https://medium.com/pinterest-engineering/driving-user-growth-with-performance-improvements-cfc50dafadd7

Medvidovic, N., & Taylor, R. N. (2010, May). Software architecture: foundations, theory, and practice. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, 471-472.

Mihaela, Ciugudean., & Gorgan, Dorian., (2016), Methodology for Identification and Evaluation of Web Application Performance Oriented Usability Issues. In *Romanian Journal of Human,* 9(2), 155-176

Mockus, A., Fielding, R. T., & Herbsleb, J. (2000, June). A case study of open source software development: the Apache server. *Proceedings of the 22nd international conference on Software engineering*, 263-272.

Nagy, D., Yassin, A. M., & Bhattacherjee, A. (2010). Organizational adoption of open source software: barriers and remedies. *Communications of the ACM*, *53*(3), 148-151.

Richards, Mark. (2015) *Software Architecture Patterns.* Sebastopol, CA: O'Reilly Media.

Shaw, M. (2001, July). The coming-of-age of software architecture research. *Proceedings of the 23rd international conference on Software engineering*, 656.

Varian, H. R., & Shapiro, C. (2003). Linux adoption in the public sector: An economic analysis. *Manuscript. University of California, Berkeley*.

Von Hippel, E. (2001). Learning from open-source software. *MIT Sloan management review*, *42*(4), 82-86.

Wagner, J. (n.d.). Why Performance Matters. *Google Developers*. Retrieved from https://developers.google.com/web/fundamentals/performance/why-performance-matters