

Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technology



Diploma Thesis

**Website of African Students Association in Czech
University of Life Sciences**

Wossenyeleh Merid Mekonnen

© 2016 CULS Prague

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Wossenyeleh Merid Mekonnen

Informatics

Thesis title

Website of African students association in CULS

Objectives of thesis

The primary objective of this study is to design an African Students Association website for Czech University of Life Sciences as it is currently non-existent. The secondary objective will be to research what technology will best serve to design and implement a student's association website in a medium to a large scale environment such as Czech University of Life Sciences. The tertiary objective is to test two completed websites designed using two different systems from various aspects such as security, responsiveness, availability, user interface and experience, speed and load test.

Goal

The main goal of the study will be to see which Content Management System will be best suited for building an association website regarding Czech University of Life sciences. The study will be addressing questions like: How user friendly will the website be? How fast will the website be? What technology will be best suited for the job? Is the website up to industry standards after being designed?

Fields

This study will include information and knowledge from the fields of User interface design, Web design, Content management systems and tools for testing websites in general.

Methodology

In the methodology part of this study, both qualitative and quantitative approaches will be utilized.

As Qualitative approach, a questionnaire will be formulated and provided to a targeted population going to Czech University of Life sciences.

For a Quantitative approach, data collected via testing the designed websites will be analyzed using statistical methods together with the data collected via questionnaires.

Based on the statistical analysis made the two different Content Management Systems utilized to design the websites will be compared with one another.

The proposed extent of the thesis

60 – 80 pages

Keywords

web development, web design, content management system, interactive design, students association

Recommended information sources

- Anderson, Stephen P. *Seductive Interaction Design: Creating Playful, Fun, and Effective User Experiences*. Berkeley, CA: New Riders, 2011. Print. ISBN:978-0-321-72552-3
- Budd, Andy, Cameron Moll, and Simon Collison. *CSS Mastery: Advanced Web Standards Solutions*. Berkeley, CA: Friends of Ed, 2006. Print. ISBN: 978-1-4302-2397-9
- Cederholm, Dan, and Jeffrey Zeldman. *CSS3 for Web Designers*. New York: Book Apart, 2010. Print. ISBN 978-0-9844425-2-2
- Duckett, Jon. *HTML and CSS: Design and Build Websites*. Indianapolis, IN: Wiley & Sons, 2011. Print. ISBN: 978-1-118-00818-8
- Kissane, Erin. *The Elements of Content Strategy*. New York: Book Apart, 2011. Print. ISBN: 978-0-9844425-5-3
- Krug, Steve. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. Berkeley: New Riders, 2014. Print. ISBN:978-0-321-96551-6
- Neal, Jen, and Hannah Westlake. *HTML5 & CSS3*. Bournemouth: Imagine Publishing Ltd, 2015. Print. ISBN:9781785461323
- Rahmel, Dan. *Advanced Joomla!* Berkeley, CA: Apress, 2013. Print. ISBN:978-1-4302-1628-5
- Schwartz, B., P. Zaitsev, and V. Tkachenko. *High Performance MySQL*. Third ed. Sebastopol, CA: O'Reilly, 2012. Print. ISBN: 978-1-449-31428-6
- Walter, Aarron, and Jared M. Spool. *Designing for Emotion*. New York, NY: Book Apart/Jeffrey Zeldman, 2011. Print. ISBN: 978-1-937557-01-1
-

Expected date of thesis defence

2016/17 WS – FEM

The Diploma Thesis Supervisor

Ing. Václav Lohr, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 18. 10. 2016

Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 24. 10. 2016

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 22. 11. 2016

Declaration

I declare that I have worked on my Master's thesis titled "Website of African Students Association in Czech University of Life Sciences" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on November 30, 2016

Acknowledgement

I would like to thank Ing. Václav Lohr, Ph.D. for his help and guidance during the length of my studies. I would also like to thank all my teachers, professors and individuals that influenced me for the better. A special thanks to my friends who have been there for me and my family, my beloved mother Fanaye Beyene and my sisters Seble Merid, Hermela Merid and Bezawit Merid who have been a source of inspiration and strength. Finally, I would like to thank the Government of Czech Republic for providing me with the scholarship to pursue my education further.

Webové stránky sdružení afrických studentů na České zemědělské univerzitě

Souhrn

Webové stránky jsou pilířem šíření informací napříč různými platformami a s růstem technologií se neustále mění a rozvíjí způsob, kterým jsou stránky vyvíjeny, což umožňuje vývoj stále stabilnějších a spolehlivějších redakčních systémů. Nicméně většina redakčních systémů je používána jako jediné řešení webového systému, neboť nabízejí odpověď na většinu potřeb internetových stránek. I přes to, že tento přístup umožňuje vyvinout malé životní cykly a podporuje lepší upgrady, aktualizace a zabezpečení, zanedbává ale skutečnost, že různé nové redakční systémy jsou, aby fungovaly dobře, závislé na potřebách webu. Z tohoto důvodu se mnoho redakčních systémů, které jsou volně k dispozici, zaměřuje na různé aspekty, jakými jsou vývoj jazyka nebo optimalizace výkonu. Tato studie zahrnuje srovnání mezi WordPress, běžně známým redakčním systémem, a novým systémem s názvem Grav, který je založený na flat file systému, u malé webové stránky sloužící sdružení afrických studentů na České zemědělské univerzitě. Práce se zabývá vývojem a hostováním webových stránek, přes načítání až k zátěžovému testování pomocí simulovaného provozu virtuálními uživateli. Dále se práce zabývá průzkumem rozsahu uživatelských zkušeností a postojů vybraných studentů na univerzitě. Srovnávání bylo prováděno zaznamenáváním doby odezvy, četnosti chyb a dotazováním. I přes to, že průzkum neukázal významný rozdíl mezi těmito dvěma webovými stránkami v rámci uživatelské zkušenosti, době odezvy a četnosti chyb, poukázal ale, že pro malé webové stránky, jako jsou stránky studentských sdružení, je systém Grav lepší a využívá jen velmi málo zdrojů. Budoucí vývoj tohoto projektu by se měl týkat případného rozšíření a optimalizací systému Grav.

Klíčová slova: redakční systémy, Grav, WordPress, zátěžové testování, vývoj webu, využití webových zdrojů

Website of African Students Association in Czech University of Life Sciences

Summary

Websites have been the pillar of information dissemination across many platforms and with the growth of technology, the way websites are developed is constantly changing and evolving, giving rise to a more stable and reliable content management systems. However, most content management systems are being used as a one solution website system, where they are offered as the answer to most website needs. Although this approach enables development to have a small life cycle and encourages better upgrades, updates and security, it also neglects the fact that, new different content management systems can perform well depending on the need of the website. Due to this, there have been many new content management systems offered freely, that are focusing on different aspects, like development language or performance optimization. For the case of a small website serving an African Student Association with in Czech University of Life sciences, this study covers the comparison between WordPress, a commercially known content management system and, a new content management system called Grav, that is based on a flat file architecture. The project moves from developing and hosting the websites, to load and stress testing them via simulating traffic from virtual users. Finally moves to a survey to scale user experience and attitude of selected students with in the university. The comparison is then made through recorded response times, error rates and questioners. The project concludes that even though the survey didn't show a significant difference between the two websites on user experience, response times and error rates showed that for a small-scale website like a student's association, Grav Performs better and utilizes very little resources. Future development of this project should cover possible extensibility and optimization of Grav.

Keywords: Content Management Systems, Grav, WordPress, Load testing, web development, website resources utilization

Table of content

| | | |
|----------|--|-----------|
| 1 | Introduction | 8 |
| 2 | Objectives and Methodology | 9 |
| 2.1 | Objective | 9 |
| 2.2 | Methodology | 9 |
| 3 | Literature Review | 10 |
| 3.1 | Content Management Systems | 10 |
| 3.2 | The world of open source | 12 |
| 3.3 | Flat file storage | 13 |
| 3.4 | Grav CMS | 13 |
| 3.4.1 | TWIG | 14 |
| 3.4.2 | Markdown | 14 |
| 3.4.3 | YAML | 15 |
| 3.4.4 | Parserdown | 17 |
| 3.4.5 | Doctrine Cache | 18 |
| 3.5 | Folder structure of Grav | 19 |
| 3.6 | WordPress | 20 |
| 3.7 | Drupal | 21 |
| 3.8 | SQL Injection | 22 |
| 3.9 | Performance testing | 23 |
| 3.9.1 | Load Testing | 27 |
| 3.9.2 | Stress Testing | 27 |
| 3.9.3 | Speed Testing | 27 |
| 3.9.4 | Loader.io | 28 |
| 3.9.5 | Load impact | 30 |
| 3.10 | Server Monitoring and Management | 31 |
| 3.10.1 | Terminal tools | 31 |
| 3.10.2 | New Relic | 33 |
| 3.11 | Webhosting | 34 |
| 3.11.1 | Shared Hosting | 34 |
| 3.11.2 | Virtual private servers | 35 |
| 3.11.3 | Cloud hosting | 36 |
| 3.12 | Ubuntu 16.04 | 36 |
| 4 | Practical Part | 38 |
| 4.1 | Selection of Content Management System | 38 |

| | | |
|----------|------------------------------------|-----------|
| 4.2 | Virtual Machine Setup | 38 |
| 4.2.1 | WordPress installation | 41 |
| 4.2.2 | Nginx | 42 |
| 4.3 | Web Design | 43 |
| 4.3.1 | Graphics Theme | 43 |
| 4.4 | Test Cases | 44 |
| 4.4.1 | Test Case A | 44 |
| 4.4.2 | Test Case B | 46 |
| 4.4.3 | Questionnaire | 46 |
| 5 | Results and Discussion..... | 48 |
| 5.1 | Loader.io test..... | 48 |
| 5.1.1 | Loader.io wordpress..... | 48 |
| 5.1.2 | Loader.io Grav | 52 |
| 5.2 | Load Impact test..... | 57 |
| 5.2.1 | Load Impact WordPress..... | 57 |
| 5.2.2 | Load Impact Grav | 58 |
| 5.3 | Questioners..... | 60 |
| 6 | Conclusion..... | 62 |
| | References..... | 63 |
| | Appendix - A..... | 67 |
| | Appendix - B..... | 68 |
| | Appendix - C..... | 69 |
| | Appendix - D..... | 70 |
| | Appendix - E..... | 72 |
| | Appendix - F..... | 75 |
| | Appendix - G | 76 |

List of figures

| | |
|--|----|
| Figure 1 Traditional content management systems (source: own) | 10 |
| Figure 2 Rank of Content Management Usage (source: w3techs, 2016) | 11 |
| Figure 3 Sequence to Scalar mapping (source: own) | 15 |
| Figure 4 Indentation in YAML (source: Evans, 2001)..... | 16 |
| Figure 5 Parsedown vs Markdown comparison (source: parsedown) | 18 |
| Figure 6 Grav folder structure (source: own) | 19 |
| Figure 7 Exploit attempts per day (source: threatpost.com, 2015)..... | 23 |
| Figure 8 IT business value curve (source: Molyneaux, 2014)..... | 25 |
| Figure 9 Key Performance Indicators (source: own)..... | 26 |
| Figure 10 Client per test load testing (source: loader.io, 2016)..... | 28 |
| Figure 11 Clients per second test (source: loader.io, 2016)..... | 29 |
| Figure 12 Maintain client load (source: loader.io, 2016) | 30 |
| Figure 13 Loadimpact testing method (source: loadimpact.com, 2016) | 31 |
| Figure 14 Usage of linux monitoring comands (source: own) | 32 |
| Figure 15 Htop resorce monitoring (source: own)..... | 33 |
| Figure 16 New Relic server monitoring tool (source: Newrelic.com, 2016) | 34 |
| Figure 17 SSH keys creation and impimentation (source: own) | 37 |
| Figure 18 Selecting operating system and processing power (source: own)..... | 39 |
| Figure 19 Selection of datacenter (source: own) | 40 |
| Figure 20 Test virtual machines (source: own) | 40 |
| Figure 21 Configuration of My SQL server for WordPress (source: own) | 41 |
| Figure 22 Configuration of Nginx for WordPress (Source: own) | 42 |
| Figure 23 Color branding selection (source: own) | 44 |
| Figure 24 Load testing case A (source: own) | 45 |
| Figure 25 Client per test results for wordpress (source: own)..... | 48 |
| Figure 26 Wordpress resource utilization results for client per test (source:own) | 49 |
| Figure 27 Maintained client test results for worpress (source: own)..... | 50 |
| Figure 28 Wordpress resource utilization results for maintain client test (source: own) | 50 |
| Figure 29 Client per second test for worpress (source:own) | 51 |
| Figure 30 Client per second test details wordpress (source:own) | 51 |
| Figure 31 Wordpress resource utilization results for client per second test (source: own) . | 52 |

| | |
|--|----|
| Figure 32 Client per-test results for Grav (source: own)..... | 53 |
| Figure 33 Grav resource utilization results for client per test (source: own) | 53 |
| Figure 34 Maintained client test results for Grav (source: own) | 54 |
| Figure 35 Maintained client test failure Grav (source: own)..... | 55 |
| Figure 36 Grav resource utilization results for maintain client test (source: own)..... | 55 |
| Figure 37 Client per second test for Grav (source:own)..... | 56 |
| Figure 38 Grav resource utilization results for client per second test (source: own) | 57 |
| Figure 39 LoadImpact scenario test for WordPress (source:own) | 57 |
| Figure 40 Wordpress resource utilization for loadimpact (source: own) | 58 |
| Figure 41 LoadImpact scenario test for Grav (source:own)..... | 59 |
| Figure 42 Wordpress resource utilization for Grav (source: own)..... | 59 |

List of tables

| | |
|---|----|
| Table 1 Summary of tests for loader.io (source:own) | 45 |
| Table 2 Summary of Questions for survey (source:own)..... | 47 |
| Table 3 Summary of responses for WordPress and Grav (source:own)..... | 60 |

List of Abbreviations

| | |
|------|---------------------------------|
| CMS | Content Management System |
| CSS | Cascading Style Sheet |
| HTML | Hyper Text Mark-up Language |
| HTTP | Hyper Text Transfer Protocol |
| IBM | International Business Machines |
| IP | Internet Protocol |
| KPI | Key Performance Indicator |
| ms | Millisecond |
| PHP | Hypertext Preprocessor |
| QA | Quality Assurance |
| SQL | Structured Query Language |
| sudo | super user do |
| URI | Uniform Resource Identifier |

1 Introduction

Information has been one of the key factors for advancement of technology that has been exhibited within the last century. From the industrial revolution to the recent science discoveries in medicine and transportation to outer space exploration, all have one huge common factor that ensured the success of these ventures, that is information.

From small blogs and knowledge bases to massive information Wikipedias that exist online, content management system has always been the backbone, and fuels advancements in technology and science. Content management systems also play a great role in facilitating the learning process by simplifying the e-learning process.

One aspect of content management system in education apart from dissemination of information is that it also keeps engaging students by providing them with different source of interaction. Apart from education, content management systems also help in creating groups of organizations based on a universities' culture and ethics for advancing knowledge, carrier or dissemination of critical information that will be useful for end users.

Student associations can be an extra engine running alongside an educational institution, assisting in activities and projects while cultivating and preparing the best minds an education system can produce. They also help in bridging cultural, socio-economic and language barriers and make it easy to address problems that arise via the proper setup channels.

Currently, within Czech University of Life Sciences, there is no African Student Association working for the betterment of the university while also facilitating information dissemination. If an association is to be made, there will need to be a small website that will run articles, knowledge base and different types of general and specific information.

2 Objectives and Methodology

2.1 Objective

The main objective of this study is to find out what content management system would be best to design an African Students Association website in Czech University of Life Sciences. The secondary objective of the study is to design and build a website based on a content management system that can allow for growth, both in serving requests and one that is extendable to allow for further development and enhancement of the website. The final objective of the study is to evaluate and compare two websites built by using different tests to observe stability of the systems and whether each of them can handle certain amount of stress and load tests.

2.2 Methodology

The study will begin by a literature review for the main subjects that were encountered in the planning, design and implementation of two different content management systems. Then the study will discuss the tools of developing the website and move to testing via load testing and stress testing methods and observing and recording response times, errors rates, thresholds that might arise. The study will also record and compare resource utilization of the built content management systems during each the test.

As a means of evaluating user experience, a questionnaire will be prepared and distributed to a targeted student body currently attending Czech University of Life Sciences. The responses form the questionnaire will then be analyzed to observe end user experience for both sites that will be designed.

The final step of this study will be the comparison of the two designed websites using the data collected during testing and user experience from the surveys.

3 Literature Review

3.1 Content Management Systems

In today's connected world, content is created in abundance from individuals in elementary school to large international corporations for various reasons. As technology advanced through the years, so did content creation methods along with the sizes of contents created. Youtube.com, a video upload and share website that was launched in May of 2005(YouTube, 2016) as a small platform, currently has one billion users and payed over two billion USD to right holders. (YouTube, 2016).

So, what is a content management system? One author defined it as “A *content management system (CMS)* is a software package that provides some level of automation to the tasks required to effectively manage content “. (Barker, 2015).

A content management system will have a designated place that will house the data or content that will be presented in a structured form whenever needed. This may be in some database form, consolidated with the system or even on a separate standalone system in another geographical location.

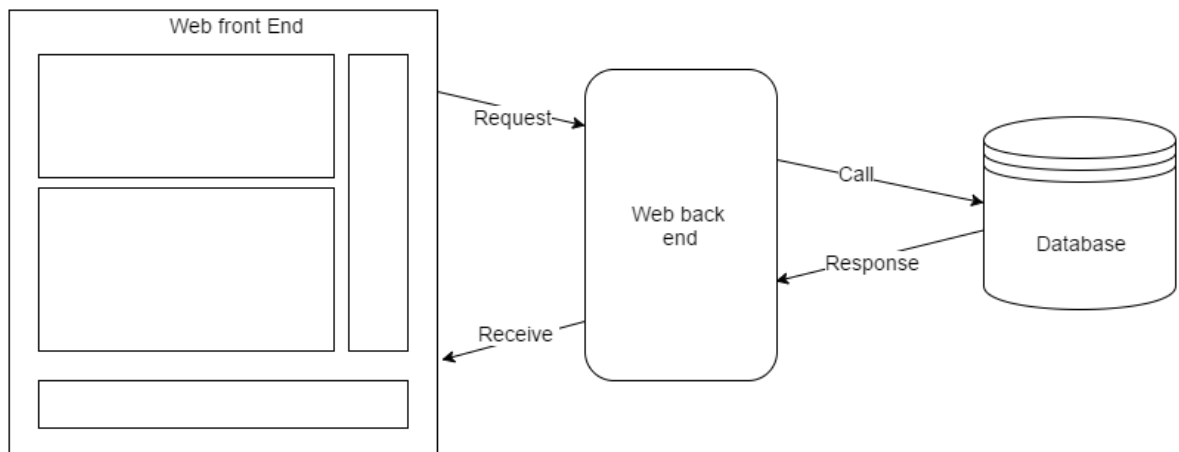


Figure 1 Traditional content management systems (source: own)

From figure 1 above, demonstrates that a content management system works in a way that the back end of the code will store content in databases and when a user requests the data, it will be called back, structured and provided to the frontend of the website for the user.

The history of content management systems is a complicated one as such, there are various arguments that exist on the online and offline community about the first content management system. Content management systems like cafelog and Wikipedia are just some of the names that have existed since the beginning although, the Idea of content management system is one that has always been with mankind since the creation of the first library.

There are various content management systems based on several different types of server side and client side languages. As of November 2016, the number of websites running on some version of a CMS is 46.4 %. Nearly half of the world’s websites currently rely on a CMS to structure and deliver content at the time of this study. From figure 2 below we can observe that, WordPress currently has a bold grip on the CMS world and as much as 58.5 % of all content managements are WordPress based. (w3techs, 2016).

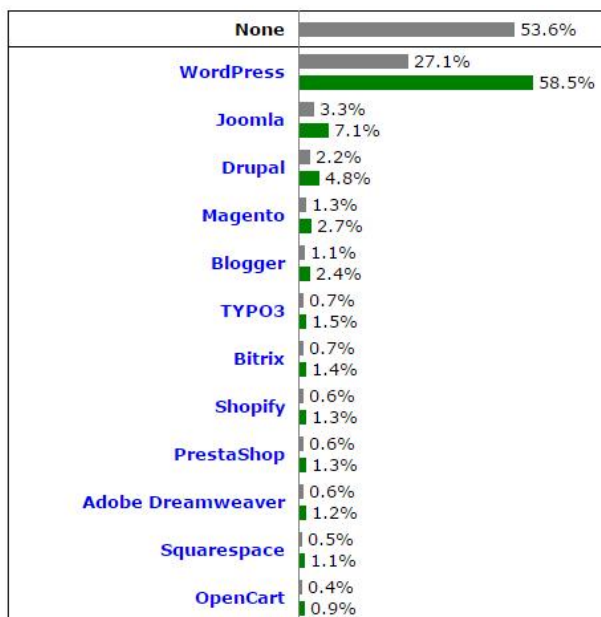


Figure 2 Rank of Content Management Usage (source: w3techs, 2016)

3.2 The world of open source

Throughout the history of publication and artistry in many different aspects in existence, there has always remained a clash between the creator and the distributor of the art itself, over control and distribution of the creation or the art. The technology that propelled the dissipation of information and knowledge, namely the printing press enlarged this problem as authors and printing companies were in constant dispute to get control.

In today's world of information technology, creation also branches out on to software's and applications developed by large multimillion dollar companies, which have hundreds of people working for them at a time. This means developing applications is very costly, requiring a large amount of money from the companies and the license of the product remains with the company that got it made.

Open source licensing is a concept that was conceived by programmers as the copyright-model of large tech companies usually hold the license or the copyright of the software; not the programmers that spent a lot of time working on it. (St.Laurent, 2004)

The popularity of open source software has greatly encouraged the advancement of web development technology and content management systems respectively. Within the past 10 years alone, the world has seen a flourishing community that greatly utilizes new systems while building upon and improving them in a relatively short turnout time.

Most of the content management systems today are completely free and open source, allowing end users, external third party vendors and enthusiast individuals access to the core code and to build upon existing frameworks. Plugins, modules, components and widgets are available to download for free or as a paid service when it comes to Joomla, WordPress and Drupal.

3.3 Flat file storage

The concept of storing information on a flat file is not a new one, in fact it predates the creation of computers represented within the entirety of the 20th century. The philosophy that it is based on, is an interesting one; paper cards with punched holes in them. This concept started back in the days of Herman Hollerith, when he first brought to life the idea of using simple paper cards and then punching holes in to them to represent data, which then are tabulated to create the first forms of structured data. Hollerith patented his idea and implemented it in the United States Census Bureau in the 1890s, alongside the creation of a machine that was used for creating the holes and tabulating them. Thus, information consisted of many boxes with thousands of punched cards in them. In 1911, he consolidated 3 companies to create IBM. (Pugh, 1995).

With the creation of IBM and the punch card system, the company dominated the system widely used until the 1970s. Later in the 1980s, concepts of flat file database systems were very popular and implemented on various systems including DOS and Macintosh operating systems. In today's website design and development community, flat file plays a big role in content creation, manipulation, data storage. This concept in recent years has given rise to various ideas regarding flat file content utilization.

3.4 Grav CMS

Named after a shortened version of nature's own phenomena gravity, the creators of Grav define it as "*file based web-platform*". (getgrav.org, 2016). Grav basically follows a different design philosophy when compared to any other Content Management Systems. To start with, there is no installation required, and it works right out of the box. All it requires is extract the archived file and its up and running in seconds. But one of the major aspects of Grav is that it is a flat-file Content Management System, meaning that there is no database required for the website, making it inherently secure against most commonly known issues of database driven website security issues.

This means that developers and website administrators have a big advantage in that, it helps eliminate one of the biggest security flaw regarding the current, popular and extensively used Content Management Systems. Grav was built with technologies that have already good name in the industry. As the main engine to run, Grav mainly utilizes PHP but it also makes use of a collection of other languages for scripting and coding purposes to build what a user requires with fewer complications as possible.

3.4.1 TWIG

TWIG framework serves as a fast and flexible template engine for PHP programming language. Developed by Sensio Labs own Fabien Potencier, who happens to be the creator of symphony framework as well. (SensioLabs, 2016). The advantage of TWIG is basically that is fast, secure and flexible. Currently Twig requires PHP 5.2.7 to run. Considering that Grav requires PHP 5.5.9 as a minimum requirement Twigs' requirement is superseded by Grav.

As far as security feature of Twig is concerned, Twig has its own built in sandbox mode, which means that any untrusted code or a code from an unknown origin can be opened in the sandbox mode prohibiting the code to perform any operations or allowing user edited codes to be opened in the sandbox mode to be safe.

The third aspect of Twig is that it is very flexible as such developers can create their own custom tags and filters as it comes powered with a flexible lexer and parser. Grav utilizes the power of Twig to have control of user interface.

3.4.2 Markdown

Markdown was created by John Gruber with the help of Aaron Swartz. On his website, John Gruber writes *“Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML).”* (Gruber, 2004).

“The idea was to make writing simple web pages, and especially weblog entries, as easy as writing an email, by allowing you to use much the same syntax and converting it automatically into HTML.” (Swartz, 2004). Markdown was originally written in Perl with the sole purpose of carrying this task. Since its creation in 2004, it has gained some momentum in different communities including publishing and development. Creating header tags and listing tags in HTML might be a tedious work, especially if the user creating them is mainly a content creator rather than a developer. For this reason and because of its fast easy to learn language, grave serves a better purpose in the publishing and content creation world.

3.4.3 YAML

According to the official website, there are hints in its name that YAML gets mistaken for markup language every so often by developers that come across it. *“‘YAML Ain’t Markup Language’ abbreviated YAML is a human friendly data serialization standard for all programming languages.”* (Evans, 2001). It is very easy to see how YAML is considered human friendly considering that it is very readable and understandable and this can be demonstrated in the home page of the YAML website as the developers used YAML to create the contents.

```
1 # this is an example yaml file for asaculs
2 siblings:
3   - first_name: Seble
4   - first_name: Hermela
5   - first_name: Bezawit
6 email:
7   - personal: getwossen@gmail.com
8   - czuculs: xmekm001@czu.cz
```

Figure 3 Sequence to Scalar mapping (source: own)

From figure 3 above, we can see the mapping of sequences to scalars as in the case of first names to siblings demonstrates how; though YAML has been written in its native

form, it remains very readable to humans. The latest version of YAML is 1.2 - 3rd Edition and from its official specification, one can see that the priority of YAML is to be easily readable to humans and uses indentation as way of creating sub or child nodes in writing data. The syntax of YAML is very easy to learn. Other Content management systems like Drupal version 8 utilizes YAML to build native forms.

```
..# Leading comment line spaces are
..# neither content nor indentation.
...
Not indented:
-By one space: |
...-By four
...-spaces
-Flow style: [ # Leading spaces
..-By two, # in flow style
..-Also by two, # are neither
..-Still by two # content nor
...-] # indentation.
```

Legend:

| | |
|---------------------------------|---------|
| s-indent(n) | Content |
| Neither content nor indentation | |

Figure 4 Indentation in YAML (source: Evans, 2001)

As we can see from figure 4 above; all child nodes must be indented one step more than their parents and in addition all the child nodes must be on the same indentation level while their content can be further indented.

Grav utilizes YAML for simple configurations ranging from declaring header blocks on the YAML front matter, to blueprints and page settings. To better elaborate, Grav uses YAML for scalars, sequences and for mappings. In the developer community, it's common to find discussions regarding why the need for YAML when other popular languages like JSON - Java Script Object Notation and XML Extensive Markup Language are extensively and widely used for various similar purposes. In some cases, people argue that JSON and XML serve this purpose better.

XML was created to be backwards compatible with SGML Standard Generalized Markup Language and is designed with supporting structured documentation where as YAML goes further towards data structures and messaging. *"YAML is the result of lessons learned from XML and other technologies."* (Clark C. Evans, 2001). JSON and YAML have different focus in general. JSON is designed primarily with simplicity and

universality in mind and its uses lowest common denominator information model, ensuring that any JSON data can be easily processed by every modern programming environment. On the other hand, YAML is designed to be human readable as a primary goal. In addition to that, according to YAML creators, JSON can be considered as a superset of YAML.

Another reason for the existence of strong discussions regarding the interchangeability of YAML and JSON is that most developers are very comfortable working with JavaScript and thus can utilize JASON natively; which resulted in JASON having a huge number of supporters. Than being said, YAML serves a very good purpose of data serialization.

3.4.4 Parserdown

As markdown is a markup language that need to be parsed to be displayed as an HTML file output; there needs to be a parser that will work in conjunction with markdown.

Parsedown functions by using what the creator calls line-based approach. This approach works by trying to read Markdown like a human; by starting to looks for lines. It uses this method to sniff out and see how the lines start which will allow it to understand different code blocks and then get to the data within the markdown file. For example, if the line starts with a ‘*’, Pasedown will see the sign and figure out that this is going towards text formatting that can range from Italic for one *, to both Italics and Bold for ***. Then it continues to see if there are any more characters after and parse it to Html.

Parsedown is a PHP based is currently the fastest parser of markdown markup language that is currently available. It works by converting markdown markup to HTML. This removes the step of remembering HTML tags and move to writing simple markdown lines if a developer or content creator of a website decides to write and stylize content like tables, bullet points and even working with making fonts Italic or Bold or both. (Rusev, 2013).

Content creators will have the option to go and edit the markdown files directly located on the server to create content, or they can go to the back end and continue to create the content like any other content management system.

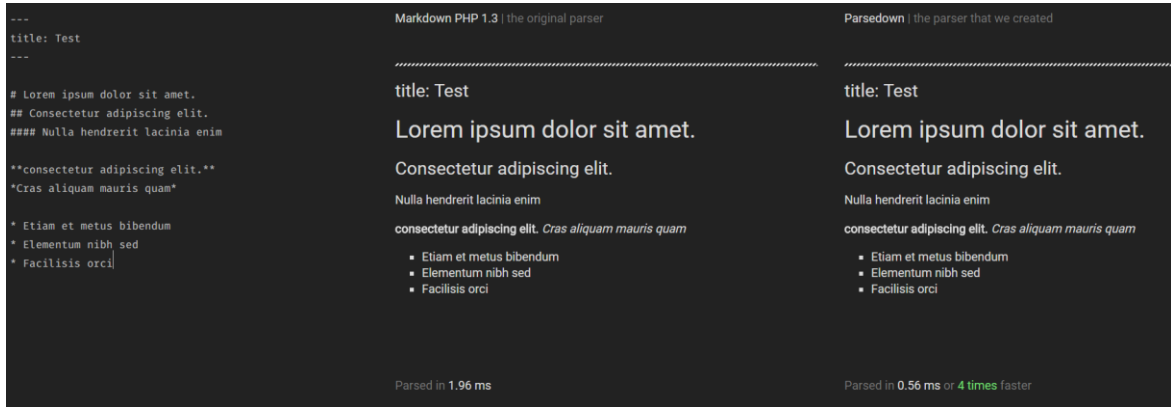


Figure 5 Parsedown vs Markdown comparison (source: parsedown)

To measure how fast parsedown is compared to others, there is a tool on the parsedown website. Parsedown claims to be as much as five times faster than other parsers including PHP’s own native parser; markdown PHP1.3. figure 5 above shows A small 9 line of markdown was fed on to two parsers, namely Markdown PHP 1.3 and parsedown. According to the results we can see that parsedown is 4 times faster in parsing markdown.

3.4.5 Doctrine Cache

Grav uses Doctrine cache for fast caching which will translate to better performance in general. Doctrine basically supplies users with cache drives for commonly used cases such as Memcache and Xcache. Apart from that, it also supplies users with ArrayCache driver which allows users to store data in an array in PHP. (Doctrine, 2010).

3.5 Folder structure of Grav

As Grav is a flat file Content Management System, its folder structure is a very important factor in managing content and system files. Once downloaded and unzipped, it has a total of 11 folders; out of which 8 are the core-top level folders that will enable it to run smoothly.

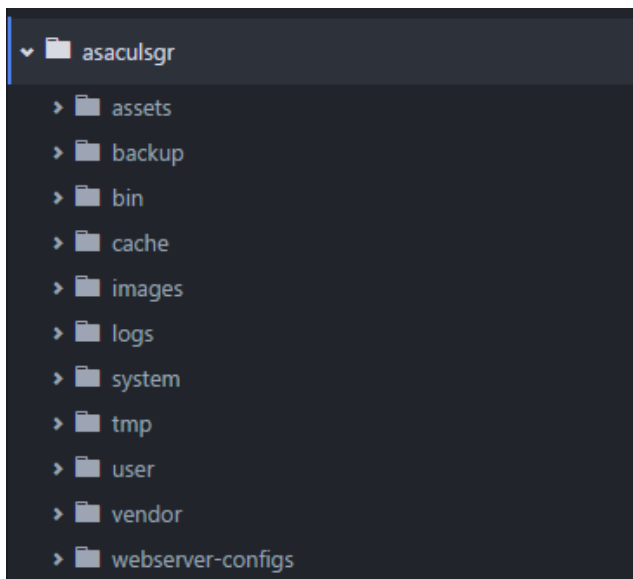


Figure 6 Grav folder structure (source: own)

From figure 6 above, we can see that under the root extraction file of Grav, the following files are listed in a waterfall format. Folder backup is mainly used to run frequent backups of the developed site, as it is handy in cases where user is met with fatal errors. Grav also contains a tmp folder meant to store temporary files during plugin or template installation processes and for keeping temporary files generally.

Folder webservice-configs also extract and come with the download and contain files such as copies of .htaccess file or nginx.conf file to make it convenient for developers setup webserver configuration. If a developer decides to implement Grav using Apache as a webserver; he will have access to .htaccess file, the same will apply for nginx via nginx.conf file.

Majority of the time, a developer will utilize resources found in user folder. Most of Grav's assets are located under this folder including, admin account information, a hashed version of admin password emails, plugins and themes are all located here, making it convenient.

3.6 WordPress

WordPress is one of the most popular open source Content Management System that has a very long history of development and support. The creation of WordPress is attributed to a blog system first coded by Michel Valdrighi called b2 cafelog in 2001. (cafelog.com, 2016). Valdrighi didn't probably realize how much of a 'snowball' effect this will have in the future but, on the 12th of May 2001 at 21:40 in the evening, he published his very first post on b2 cafelog. Today, one can still see his first blog post on the actual website.

Valdrighi worked on b2 cafelog on his spare time, updating it and maintaining it while adding some small improvements through time, but his contributions to developing and maintaining it dwindled over time. In November 5th of 2002, Valdrighi released 0.6.1 version of b2 cafelog and disappeared. By March of 2003, other users can be seen posting messages looking for him on the website. Meanwhile, the lack of progress on b2 cafelog led Matt Mullenweg to post his views which resulted in the proposed collaboration by Mike Little. In collaboration of Matt Mullenweg and Mike Little decided to fork b2 cafelog.

On May 27th of 2003, Matt Mullenweg announced the first release of WordPress with full change logs. (Mullenweg, 2003a). Soon enough in October 2003, WordPress 0.72 final was released by the duo along with b2 cafelog version 0.6.2.2 to help facilitate the change for people who didn't want to completely change to WordPress but also wanted to avoid an SQL injection vulnerability. (Mullenweg, 2003b).

SQL injection vulnerability in WordPress traces its origins back to the first release of WordPress 0.72, where developers can be seen discussing on the b2cafelog site about a fixed SQL injection issue after getting reports by users. Concurrently, they also fixed and re-released b2 cafelog 0.6.2.2; a blogging system that predates WordPress for users that were not ready to upgrade their systems to the first version of WordPress.

What made WordPress so popular amongst developers and content creators? Though the initial idea of WordPress has been to serve as a blogging platform, overtime it evolved to the point where it became very easy to use it to build landing pages and full-fledged websites. Compared to similar CMS which use the same identical technology, WordPress has a very short learning curve and works very well for small to medium scale websites.

Another feature that contributed to the popularity of WordPress is its diverse library for third party plugins; which also in some cases one of its vulnerabilities. As WordPress is an open source platform in its nature, it allows for anyone to be able to develop a plugin and submit it to the online library. Even though there are precautions that are taken by wordpress.org to eliminate the chance of dangerous plugins from being published, they seldom find their way on to the website.

A second issue with plugins is that after being published once, they will need to be constantly updated and maintained to correct flaws and cover security gaps found or reported by users. Although most developers actively maintain the plugins they published, some are not published in a timely manner or are neglected.

3.7 Drupal

Drupal is one of the well-known and used content management system in the world. Like WordPress and most other content management systems, it relies mainly on PHP and needs a database to function as well. The history of Drupal goes back to the year 2000 and is attributed to Dries Buytaert and Hans Snijder. The two students of the

University of Antwerp, frustrated by not getting a good internet connection decide to share an ADSL modem connection and then decide to create a small website to communicate with each other. After much use within their dorms, in 2001 they decided to release it to the world under the name dorp.org as drop meant 'village' in Dutch but only to make a mistake and release it as drop.org (Drupal, 2016).

Since its creation, Drupal has come a long way and contributed a lot in the content management world. Drupal has a variety of features and is one of the most stable content management systems; and is also extensible via the use of plugins. The stability of Drupal has made helped it in gaining popularity by government Content management systems and fortune 500 companies like the Economist, BBC store, the Bermuda government to name a few. The only drawback to Drupal is that it has a very steep learning curve, which turns back most people who set out to find a content management system they can use.

3.8 SQL Injection

One of the biggest commonly known issues of database driven websites is SQL injection. According to w3schools.com, the leading web standardization of web technologies SQL injection is defined as a *“technique where malicious users can inject SQL commands in to an SQL statement, Via webpage input. Injected SQL commands can alter SQL statement and compromise the security of a web application.”* (w3schools, 2016).

Most content management systems that rely on databases actively maintain their releases to counter against SQL injection attacks and in trying to cover loopholes in which attacks might occur. If an attack becomes successful, the attacker can render the entire website unusable, disrupting services and possibly stealing or corrupting data.

For large corporations that utilize database driven content management systems within their intranets and on the web, it can result in a significant cost financially and damage to sensitive data, breach of data or even encrypting attacks for ransom.

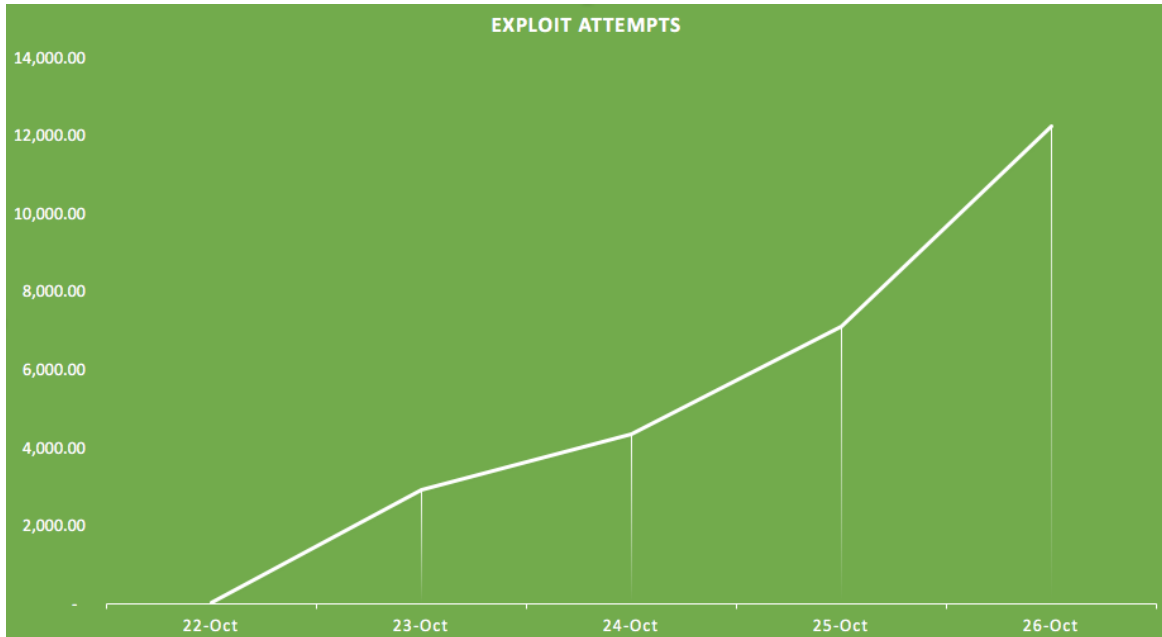


Figure 7 Exploit attempts per day (source: threatpost.com, 2015)

From figure 7 above reported by threatpost.com, we can see that during October of 2015, due to a high SQL injection vulnerability that was disclosed regarding Content Management System Joomla versions 3.2 – 3.4.4 there were 12,000 attacks on a single day. (Brook, 2015). This came after a disclosure regarding a flaw in Joomla release, even though a patch to earlier versions were released as well. Information including the patches were announced on a Thursday evening in Europe as most webmasters were home.

On February of 2015 another CMS, namely WordPress had a very high risk of vulnerability that was discovered in one of its plugins namely WP-Slimstat. Tripwire reported that “*WP-Slimstat, potentially impacting more than one million websites.*” (Santillan, 2015). This opportunity for attack occurred as the plugin had a weak secret key that was easily breakable, giving way to SQL injection attacks.

3.9 Performance testing

The field of performance testing is vast with a lot of variables to consider. Today, web based applications are integral part of corporations and small companies, and help

them achieve their goals daily. Because of this reason, literature found today focus on application testing rather than a simple website testing.

Performance testing has become one of the most common practices in the world of website design, especially in web application development. To test the performance of anything, we first must define a way of measuring performance relative to the work done. From an end user perspective, performance is simply being able to carry out a given task without any delay. (Molyneaux, 2014)

Connectivity to the world-wide web is becoming a standard within our life time and the tremendous speed networking solutions and internet grew has given rise to the ability to stream large amounts of data online on the go. Internet connection has gone from 56 Kbps dial-up to Gigabit internet connections in countries like Japan. As a civilization, having reached the pinnacle of getting information we requested at our figure tips, and the more connected we became, the less patience we have developed on waiting for response from any website or application.

Currently most end users have expectations for websites and web applications to perform at unprecedented speeds and having information at the will of their fingers. In the connected world of today, most companies and organizations rely on online presence and the internet to carry out their daily activities. Adobe reported that, in the 2016 USA black Friday online sales, a new record was seen at 3.34 Billion dollars with a 21.6% growth since last year and 45% of the visits was from mobile phones. (Adobe, 2016).

Another factor for considering performance testing of websites and applications revolves around the fact that most bugs and issues with our completed website tend to surface while a business is running at a late stage of the website and web application life cycle, increasing the cost and effort of resolving the issues. Figure 8 below demonstrates the Information Technology business value curve.

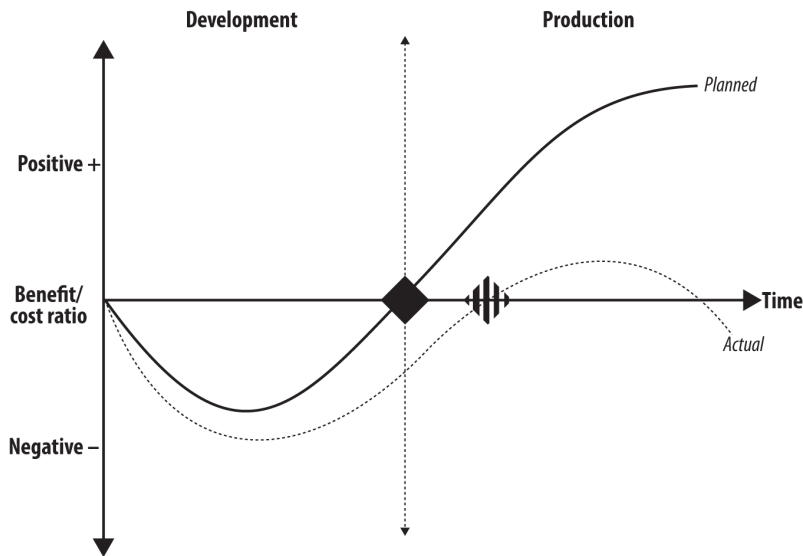


Figure 8 IT business value curve (source: Molyneaux, 2014)

When dialup was thriving in the early 2000's, waiting for half a minute for websites to load or even access was common. Today, the expectation for response time of websites or accessing email clients have increased dramatically to the point that acceptable response times has been reduced to mere 2 - 4 seconds. An article in the Guardian stated that, 32% of internet users in the United Kingdom abandon sites between 2 – 5 seconds for slow sites. (Weatherhead, 2014.).

Another study by the telegraph stated that the attention span of human beings has decreased alarmingly. In fact, telegraph reported that looking at another survey, the attention span of an average Canadian was 12 seconds in the year 2000. The same survey in 2015 revealed that it has dropped to 8 seconds. (Watson, 2015).

Waiting for more than 10 seconds for a website to load can be expected criteria for web portals and web applications dealing with large volume of data, data processing and so on, but considering the attention span of end-users and current technology available for simple websites, acceptable measures of website response time should be well in range of 2-5 seconds with a maximum of 5 seconds.

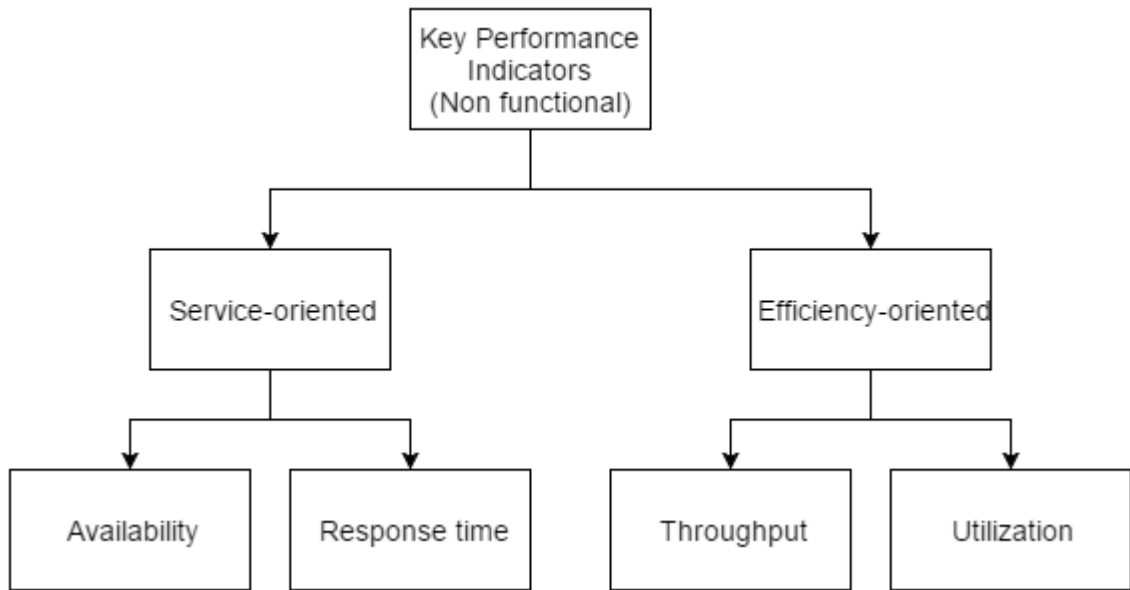


Figure 9 Key Performance Indicators (source: own)

From figure 9 above, we can see that key performance indicators are grouped in to two. The first group, service oriented, focuses on the service giving side of the website; mainly availability and response time. Availability is the amount of time the website is running, functioning and serving users. It is a key factor to consider for websites and systems that have high dependency on being available to customers always. A good example of this is banking websites or web based applications.

Response time of our website gages how long it takes for our website to run tasks or requests provided by our end-user. The quicker the response time the better. With regards to performance testing, it is measured by recording the time between end-user sending a request and the website giving a complete response.

We can see throughput as the how many hits a website will get within a specified amount of time. This helps in identifying if there are any bottlenecks with in the code of the website we have written. As the last efficiency-oriented measure, if performed right, utilization could demonstrate how much infrastructure resources our website is utilizing in the background while performing a variety of tasks. This will enable us to determine what kind of infrastructure demand there will be in the future and whether there are any tasks within the website that will require heavy utilization of resources.

Majority of the time, load testing gets confused with stress testing and in some cases, they are used interchangeably. Though they might be used in conjunction with each other, the aims of the tests are very different as such load testing aims to understand the website from behavioral and stability aspects whereas stress testing aims at finding breaking points and recoverability of a website after its fail threshold has been reached.

3.9.1 Load Testing

Load testing is simulating a given number users accessing the home or different parts of the website in question for a given amount of time, so as to see how many users the website can handle at a given time while still having adequate performance. This gives developers an insight in to the stability and behavior of the website or application in question.

3.9.2 Stress Testing

Stress testing is surpassing the threshold that the website has been set to serve and find out what happens, how and which components fail and it also aims at finding the breaking point of the website and how it recovers.

3.9.3 Speed Testing

Speed testing simply finds out how much time a website takes to load the pages to a given user in general. There are a lot of different tools out there to achieve this goal, the most common one being the developer tools found in Mozilla Firefox and Google Chrome, as they come integrated on to the browser. These tools come out of the box with a simple and intuitive interface that will enable users to see various information; one of them being website load time with a waterfall graph.

The problem with testing a website this way is that if the website is hosted locally, then it would not provide an insight in to how the website behaves in a real-world scenario; that is when its viewed by people around the world from different locations. Thus, it is advisable to look for testing tools that use 3rd party servers that can simulate the various locations that traffic might come from for that specific website.

3.9.4 Loader.io

Loader.io uses common HTTP verbs GET and POST to call landing pages of websites while checking for error and error responses. This feature will enable users to test the core system of their website by discovering the error threshold of a website. The testing methods of loader.io are divided in to 3 major parts. (loader.io, 2016).

Figure 10 below shows how the client per test method sends virtual users across time to targeted host machines. It is a simple test requiring the number of virtual users needed and the duration of the time to simulate them. As an example, if testing for 30,000 users with in a time of 30 seconds. In this case, 1,000 virtual clients will connect with our machine every second. This allows developers to define and set the amount of traffic that they are expecting for their website run the test and see how much of the traffic their infrastructure it can handle.

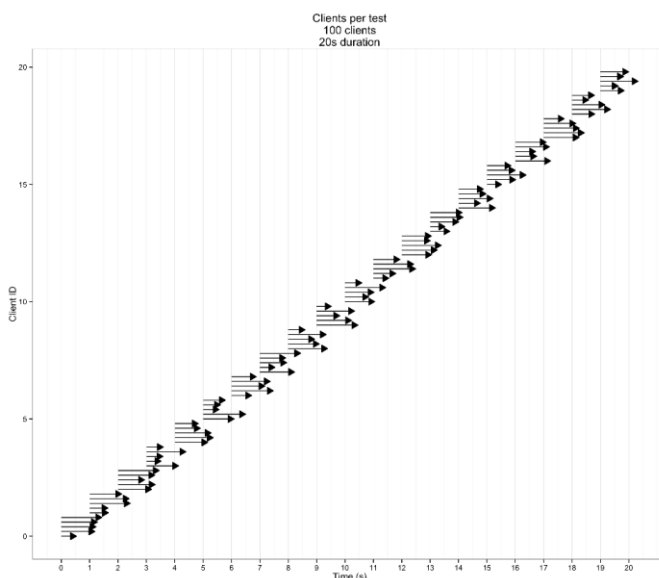


Figure 10 Client per test load testing (source: loader.io, 2016)

The second type of test is called client per second test. This test is similar to clients per test with the difference being that we will be aiming to test virtual clients per second rather than a general fixed number of clients per the whole test. Figure 11 below shows how the client per second is carried out.

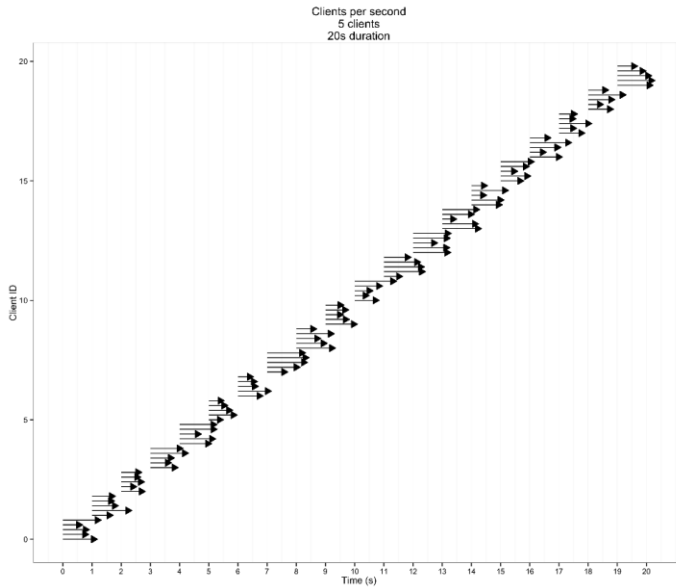


Figure 11 Clients per second test (source: loader.io, 2016)

The last type of test provided by loader.io is called the maintained client load test. This test is a very crucial part of loader.io as it simulates a ramp-up test, where virtual users making calls to the website will start from a small number and will increase to a number preset by the tester. During this test, each instance of virtual user will be making another request as soon as it finishes sending the first request. This test will help us in identifying the threshold of the website and we can see how many clients our website can serve before reaching its threshold. Figure 12 below demonstrates how virtual users flood and make recursive requests on maintain client load test.

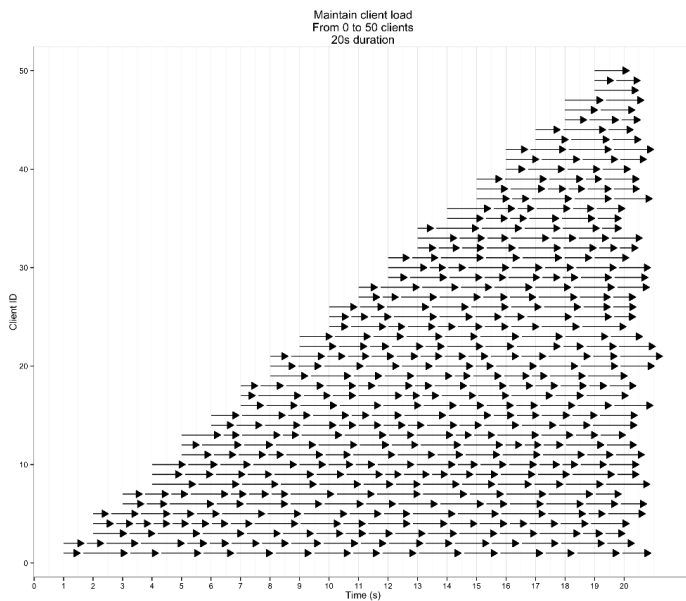


Figure 12 Maintain client load (source: loader.io, 2016)

3.9.5 Load impact

Load impact is an online website and applications testing service with over 20 years of experience in the QA industry. Moving on alongside new technology, an online test mechanism was created by the company so that users can test different scenarios; from small websites and landing pages to heavy demanding applications.

For website testing purposes, load impact runs a test by HTTP verbs like GET, POST, PUT, HEAD, DELETE, OPTIONS, TRACE, PATCH. According to the W3 consortium, the above-mentioned verbs are methods of HTTP/1.1 used in different occasions. (w3.org, 1999). Methods like GET and HEAD are specifically used to retrieve information over the internet by means of using Request -URI, even if the information requested is a data that will need to be processed beforehand. For this reason, GET and HEAD methods are considered safe methods.

Method OPTIONS is a way of requesting for information about communication methods available. Methods, DELETE, PUT, POST and PATCH use different means to request and put or interact with the counter parts.

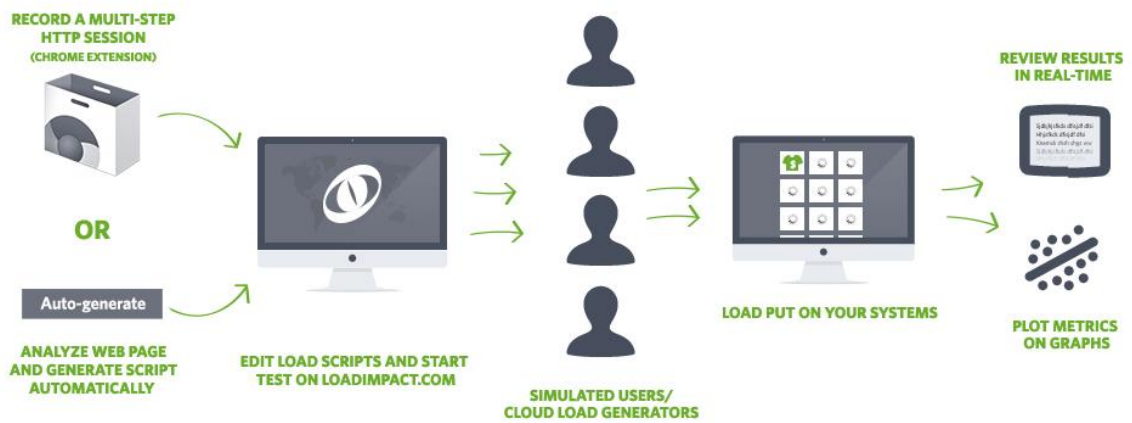


Figure 13 Loadimpact testing method (source: loadimpact.com, 2016)

Figure 13 above demonstrates how loadimpact.com generates HTTP sessions by auto analyzing a destination domain name or IP address or by recording a multistep HTTP session via a chrome browser plugin then enabling the user to view the code generated and edit it. After validating the created session, it simulates the required number of virtual users as a load towards the website. All the information it gathers about the website is then collected and used as needed. The ability of loadimpact.com to allow users to record a multistep HTTP session helps QA to be able to create custom tests that caters to each testing scenario.

3.10 Server Monitoring and Management

3.10.1 Terminal tools

There are a couple of ways that infrastructure or resource monitoring can occur within a server based on a Linux and Unix system whether it is standalone machine or a virtual machine. The first method of monitoring involves using common and available resource monitoring commands to evaluate how much resources a website is utilizing.

```
wordpress@asaculs-wp-2: ~
wordpress@asaculs-wp-2:~$ free
              total        used         free       shared    buff/cache   available
Mem:           500200        163152         64452         11940         272596         295976
Swap:              0              0              0
wordpress@asaculs-wp-2:~$ vmstat -s -S M
 488 M total memory
 161 M used memory
 204 M active memory
 170 M inactive memory
  61 M free memory
  16 M buffer memory
 250 M swap cache
  0 M total swap
  0 M used swap
  0 M free swap
 757 non-nice user cpu ticks
  0 nice user cpu ticks
 524 system cpu ticks
139220 idle cpu ticks
 145 IO-wait cpu ticks
  0 IRQ cpu ticks
  1 softirq cpu ticks
  2 stolen cpu ticks
216327 pages paged in
 52508 pages paged out
  0 pages swapped in
  0 pages swapped out
 44203 interrupts
127644 CPU context switches
1478727619 boot time
 1937 forks
wordpress@asaculs-wp-2:~$
```

Figure 14 Usage of linux monitoring comands (source: own)

Figure 14 shows the usage of *free* and *vmstat -s -S M* in terminal on the virtual machine running Ubuntu 16.04 to display free disk space size and RAM usage in megabytes. Although this method proves to be the simplest one, in most cases repeatedly and constantly monitoring resources that is being consumed is a cumbersome task specially if the method involves of using multiple separate commands that will exit the moment you want to run another. For example, *vmstat* can be run with additional parameters that can enable us to monitor the RAM of our machine every 2 seconds for 5 times as by adding this parameter at the end of the line. What if we want to monitor disk read/write? Then discontinue this command run another? Wait for it to finish or run another terminal in another window?

To solve this problem, we can use third party installable resource monitoring services that will run live, displaying all the current utilization. For this purpose, we can try installing and using a third-party light tool like *Htop*. Figure 15 below demonstrates the interactive nature of *Htop* with better visual aids to monitor resources. It can also monitor resources in real life. It is one step further but still lacks the recoding capabilities that other tools have.

```

wordpress@asaculs-wp-2: ~
CPU[          0.0%]   Tasks: 36, 42 thr; 1 running
Mem[|||||||189M/488M] Load average: 0.00 0.00 0.00
Swp[          0K/0K]   Uptime: 00:35:33

  PID USER   PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---  ---    ---  --  ---    ---    ---  -  ---  ---  ---  ---
1364 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1365 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1366 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.16 mysqld
1367 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1368 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1369 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1370 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1371 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1372 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1373 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1374 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.03 mysqld
1375 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.06 mysqld
1378 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.05 mysqld
1379 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.07 mysqld
1380 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1381 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.13 mysqld
1382 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1383 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1384 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1385 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1386 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1387 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1388 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.01 mysqld
1389 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1390 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1391 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1396 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:00.00 mysqld
1343 mysql  20   0 1077M 126M  9832  S  0.0 25.9 0:01.22 mysqld
  671 root    20   0  122M 28336 28004  S  0.0  5.7 0:00.70 systemd-journald
 1358 root    20   0   354M 23880 19376  S  0.0  4.8 0:00.18 php-fpm: master proc

F1 Help  F2 Setup  F3 Search  F4 Filter  F5 Tree  F6 SortBy  F7 Nice  F8 Nice +  F9 Kill  F10 Quit

```

Figure 15 Htop resorce monitoring (source: own)

Htop is accurate measurement of our resources but it is highly inconvenient, and won't work for us especially if we want to not only display but record data. There are a lot of light weight open source tools for Linux systems than can be installed and help to manage resources but, in the end, they would even add more RAM usage to our machine just to be able to record the results per second.

3.10.2 New Relic

The solution comes in the form of an external server monitoring system, that is setup to constantly gather data and store it in a way that can be retrieved and visualized while also being able to provide real time to help infrastructure management to keep track of resources their machines use

New relic server monitoring tools offer users the chance to install a small script that will capture real live data and steam it back to the virtual machine needed without costing the user RAM or processing power. The real advantage of New Relic is that it helps store

and display data collected from the last 24 hours from multiple to be machines at the same time and to see what triggered certain events and how much resources our server utilized at a given period back in time.

This is generally an ideal situation for website developers, admins and QA engineers as they will be able to find all the machines monitored on one location with the ability to filter through them and see data they need within the last 24 hours, all for free. Figure 16 below showcases the server monitoring services provided by New Relic.

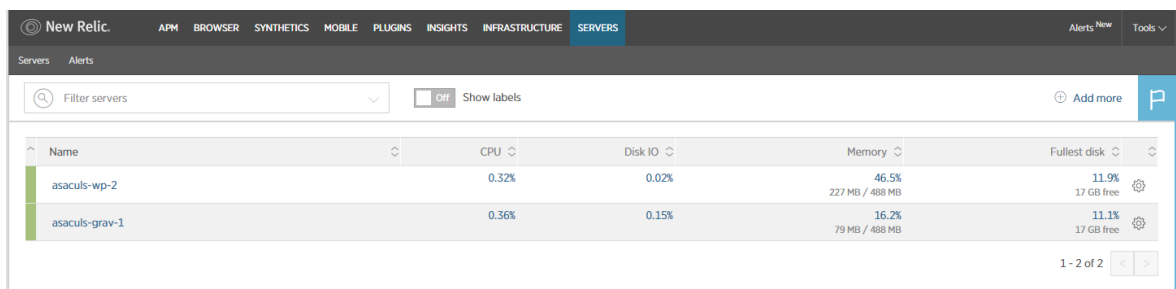


Figure 16 New Relic server monitoring tool (source: Newrelic.com, 2016)

3.11 Webhosting

3.11.1 Shared Hosting

Share hosting is one of the most common ways of hosting a website. This method of hosting a website will utilize all the resources of a single physical machine and supplies them to all the websites or applications that are being hosted within it. Most of the websites that are low traffic and that do not serve a lot of people per day or need a bigger computational power and memory are usually hosting using such plans. As it is the most common one; it is also the cheapest one that can be found. On most cases, shared hosting providers will often update to a certain component late thus, it's hard to find a hosting provider that can fulfill the pickiest of developers.

As an example, ixwebhosting.com services offer very affordable and intriguing packages. Almost all their offers come with unlimited bandwidth, unlimited disk space and unlimited hosted domains with a minimum of 2 dedicated IP's and at least one free domain; all for 3.95 USD per month. That is a good deal for customers that are looking for flexibility but if we decide to go and see what versions of My SQL or PHP they support; then some developers might stray away from such hosting companies as they only have My SQL 5.1, which does not support UTF8mb encoding and up to PHP version 5 only. This means that some content management systems will not be able to run in these environments for example, the latest releases of Grav which requires PHP 5.5.9 or higher.

3.11.2 Virtual private servers

Virtual Private Servers have been a choice of many websites since their inception. To first see virtual private hosting, we will need to see what a virtual machine is. VM ware, a popular virtualization software defines a virtual machine as *“a software computer that, like a physical computer, runs an operating system and applications. The virtual machine is comprised of a set of specification and configuration files and is backed by the physical resources of a host”*. (VM Ware, 2016). This enables web hosts to provide hosting plans that will give end users the option to have a dedicated amount of memory and processing power they will require without the need to go for a dedicated server. This practice insures that the cost of hosting a website is not as expensive as having one dedicated physical machine. In a virtual private server environment, the physical available resources of the machine will be divided in to smaller virtual machines. This will enable different users to run different applications or websites at the same time without interfering one another. Resources will be divided in a predefined way so that when a website is hosted on a virtual private server; the resources allocated for it will remain without being tampered with or utilized by another website or application.

3.11.3 Cloud hosting

During the initial introduction of cloud computing, a mist of confusion was created within the information technology sector. Most companies had questions on whether to move on to the faster and scalable cloud system for their operational needs. In the specific sector of web hosting, the promises of cloud computing resembled the way virtual private servers operated in effect creating a phenomenon called cloud washing. Cloud washing was a term used when companies presented their old technology as new under the banner of cloud computing. Many of today's leading companies were accused by some analysts as cloud washing their services. The distinguishing factors to be able to call a service a cloud is:

- On-demand service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service, or pay-per-use model

Thus, the main differences between old technologies provided by vendors and the new services that exist as cloud can be distinguished from one another.

There are a lot of virtual machine hosting providers and a lot of cloud hosting providers in the market; with the major difference being the resource pooling, measures service and on-demand service. Cloud hosting can provide this services without the client having to call a service center or get in contact with an IT Specialist to setup the virtual machines.

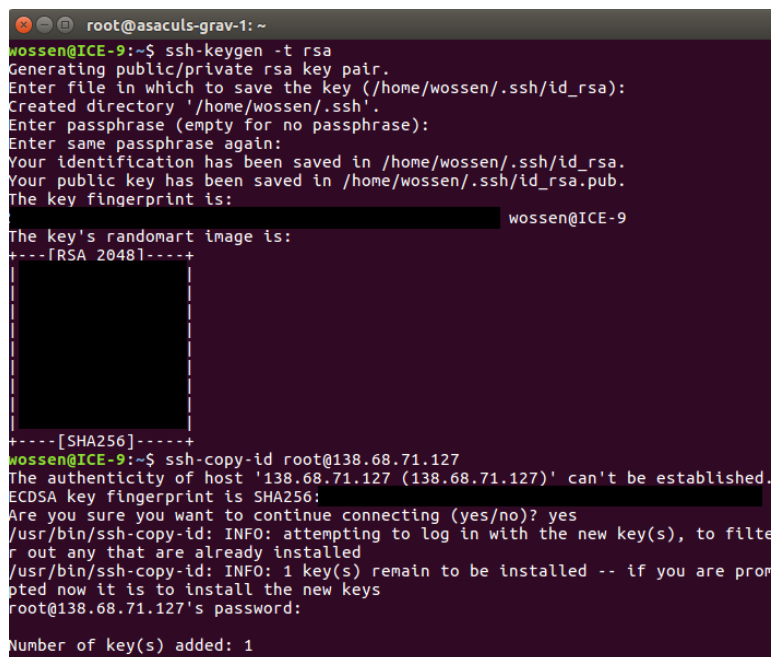
3.12 Ubuntu 16.04

Linux systems were one of the most widely used server based Operating Systems in existence. Because of their versatile nature and well supported and extensive community, a lot of vendors and developers choose to use different Linux distro to facilitate, run and

manage their servers. There are many different distros of Linux systems that are available to the public under GNU license; with the most common one being red hat, Ubuntu, Debian, mint etc.

Ubuntu is a free and popular distro of Linux systems. Because of it is free distribution and highly supported community, Ubuntu is one of the most used Linux distros for developers and web admins to design, develop and manage local and remote servers. The current version of Ubuntu is 16.04 and its well supported by the community.

SSH stands for Secure Shell Script and more often developers and any one with the need to access a Linux server that a website is hosted on will use SSH keys to connect to and run scripts and manage servers. SSH keys provide a safe and convenient way of connecting remotely to Linux servers to conduct daily operations. Figure 17 below shows SSH key generation and transfer to remote virtual server for our Grav website.

A terminal window showing the process of generating an SSH key pair and copying it to a remote host. The user is root@asaculs-grav-1. The terminal output shows the execution of 'ssh-keygen -t rsa', the creation of a directory, and the generation of a key pair. The key fingerprint is displayed as a random image. The user then runs 'ssh-copy-id root@138.68.71.127', which prompts for a password and successfully copies the key to the remote host.

```
root@asaculs-grav-1:~  
wossen@ICE-9:~$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/wossen/.ssh/id_rsa):  
Created directory '/home/wossen/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/wossen/.ssh/id_rsa.  
Your public key has been saved in /home/wossen/.ssh/id_rsa.pub.  
The key fingerprint is:  
wossen@ICE-9  
The key's randomart image is:  
+-----[RSA 2048]-----+  
|  
|  
|  
|  
+-----[SHA256]-----+  
wossen@ICE-9:~$ ssh-copy-id root@138.68.71.127  
The authenticity of host '138.68.71.127 (138.68.71.127)' can't be established.  
ECDSA key fingerprint is SHA256:  
Are you sure you want to continue connecting (yes/no)? yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter  
out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prom  
pted now it is to install the new keys  
root@138.68.71.127's password:  
Number of key(s) added: 1
```

Figure 17 SSH keys creation and impimentation (source: own)

4 Practical Part

4.1 Selection of Content Management System

In this part of the study, two different types of content management systems were chosen and utilized to design a website for African Students Association in Czech University of Life Sciences.

When selecting the content management system to be used, there were three major selection criteria that were implemented. The first criteria is that the system selected must be a fee and open source system. The second selection criteria is the system needed to be utilizing PHP as a server side language, which will enable us to evaluate content management systems in a leveled ground. Finally, the third selection criteria is usage statistics from around the world and new and upcoming technologies for comparison.

As illustrated in the literature review of this study, WordPress is one of the most popular and utilized content management system, and ranks number one with in the community of web developers and users alike due to its simplicity and community infrastructure. This fact alongside WordPress being open source and using PHP made it the first selection for this study. The second selection was Grav as it is fairly a new system relying on less infrastructure and doesn't have usage statistics currently.

4.2 Virtual Machine Setup

Digital Ocean was selected for three major reasons of which the first being that Digital ocean provides a handy tool to allow clients to choose from a wide variety of operating systems with in just a few minutes. The second reason is that Digital Ocean is one of the cheapest cloud Virtual Machine provides in the world, having a plan that starts from 5 USD for a 512 MB RAM, 1 Intel Xeon E5-2650L v3 @ 1.80GHz Processor and 20 GB hard disk space Virtual Machine. The final reason relates to the amount of setup time

regarding having a working operating system on the virtual machine. Both instances of virtual machines were up and running with in a 2 minutes' in total.

The screenshot shows the 'Create Droplets' page on DigitalOcean. At the top, there are navigation links for 'Droplets', 'Images', 'Networking', 'API', and 'Support'. The main heading is 'Create Droplets'. Below it, there's a section 'Choose an image' with a help icon. Underneath, there are two tabs: 'Distributions' (selected) and 'One-click apps'. The 'Distributions' section shows six operating system options, each with a logo and a 'Select version' dropdown. The first option, Ubuntu, is highlighted in blue and shows the version '16.04.1 x64'. Below this is a section 'Choose a size' with two tabs: 'Standard' (selected) and 'High memory'. This section displays six pricing and resource options in a grid. The first option is highlighted in blue.

| Operating System | Price | Resources |
|------------------|--------------------------|--|
| Ubuntu | \$5/mo \$0.007/hour | 512 MB / 1 CPU 20 GB SSD disk 1000 GB transfer |
| FreeBSD | \$10/mo \$0.015/hour | 1 GB / 1 CPU 30 GB SSD disk 2 TB transfer |
| Fedora | \$20/mo \$0.030/hour | 2 GB / 2 CPUs 40 GB SSD disk 3 TB transfer |
| Debian | \$40/mo \$0.060/hour | 4 GB / 2 CPUs 60 GB SSD disk 4 TB transfer |
| CoreOS | \$80/mo \$0.119/hour | 8 GB / 4 CPUs 80 GB SSD disk 5 TB transfer |
| CentOS | \$160/mo \$0.238/hour | 16 GB / 8 CPUs 160 GB SSD disk 6 TB transfer |

Figure 18 Selecting operating system and processing power (source: own)

Figure 18 above shows the setup process, choosing the desired operating system alongside the processing power required. Digital ocean provides the option to select where the datacenter for our virtual machines are going to be placed. Figure 19 below shows the locations of datacenters available. Attached to certain locations, there is also additional services like adding block storage to a virtual machine.

Choose a datacenter region

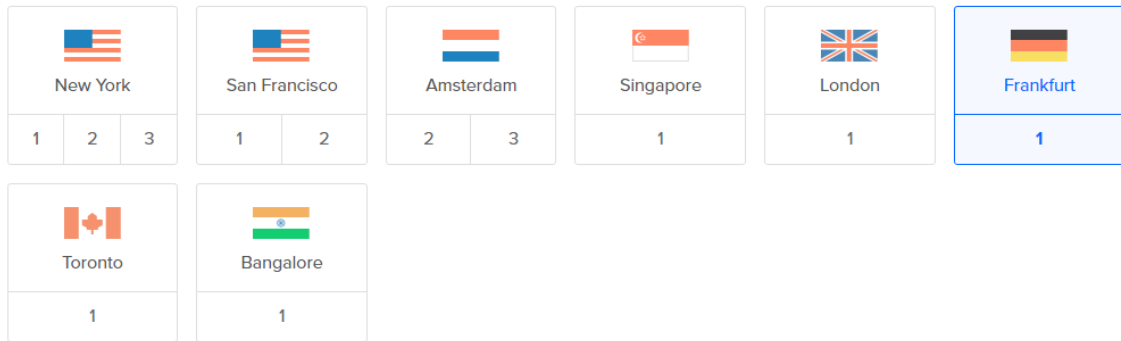


Figure 19 Selection of datacenter (source: own)

The intended location of the website is Prague, but the option of choosing Prague as a datacenter is not possible because digital ocean does not provide it. Thus, to be able to simulate a scenario, that gets close to the real-world as much as possible, and because it is closer to Prague geographically than the other available locations, Frankfurt - Germany was chosen as the location of the servers.

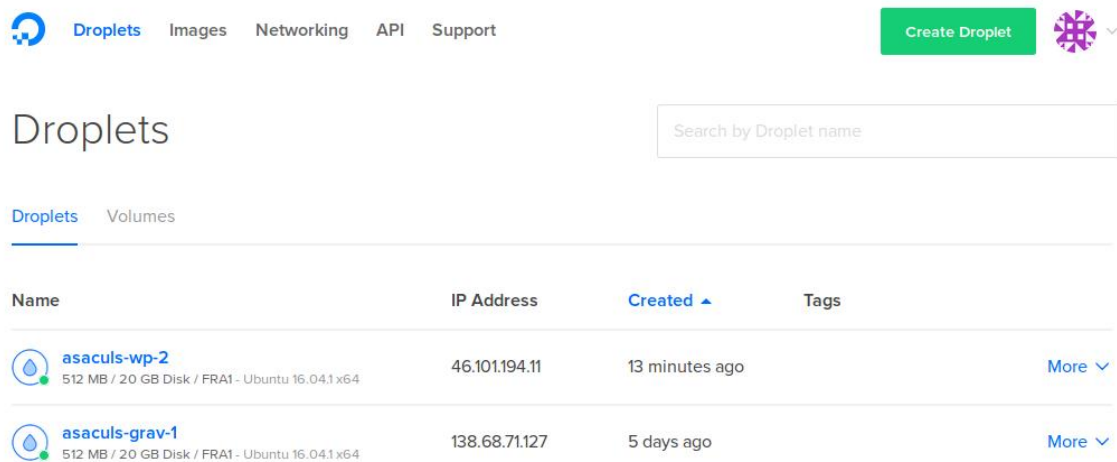


Figure 20 Test virtual machines (source: own)

Figure 20 above shows the virtual machines that were acquisitioned for hosting the two different websites and conducting tests afterwards. Digital Ocean calls the Virtual Machines it offers Droplets. Each Droplets are instances of Virtual Machine that are spine by the user for however long he desires.

For a base operating system, Ubuntu 16.04, X64 was chosen for both droplets for four main reasons: -

- All Ubuntu versions come under a free license
- Version 16.04 is stable, well documented
- Supported via a large online community
- Recommended by a selection of vendors including Digital Ocean

4.2.1 WordPress installation

As it is not advisable to install and configure files as the only root user on a machine, we created a second user called 'wordpress' with sudo abilities. For the ease of development and installation, Ubuntu 16.04 was installed on the local machine of the author to make it easy to use terminal and its commands to configure and install the website. After installation and configuration of Nginx and PHP 7.0, via terminal, on our remote virtual machine, installation of SQL Server 5.7 was installed. Figure 21 below shows the configuration phase of MY SQL 5.7 server on our virtual machine.

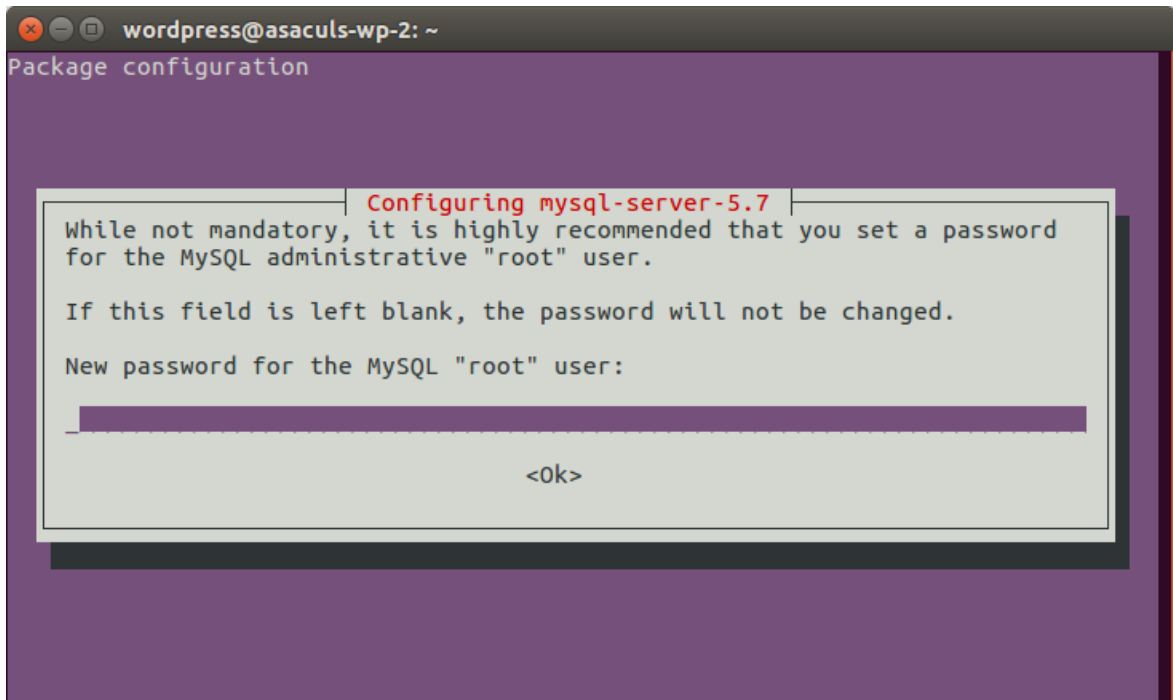


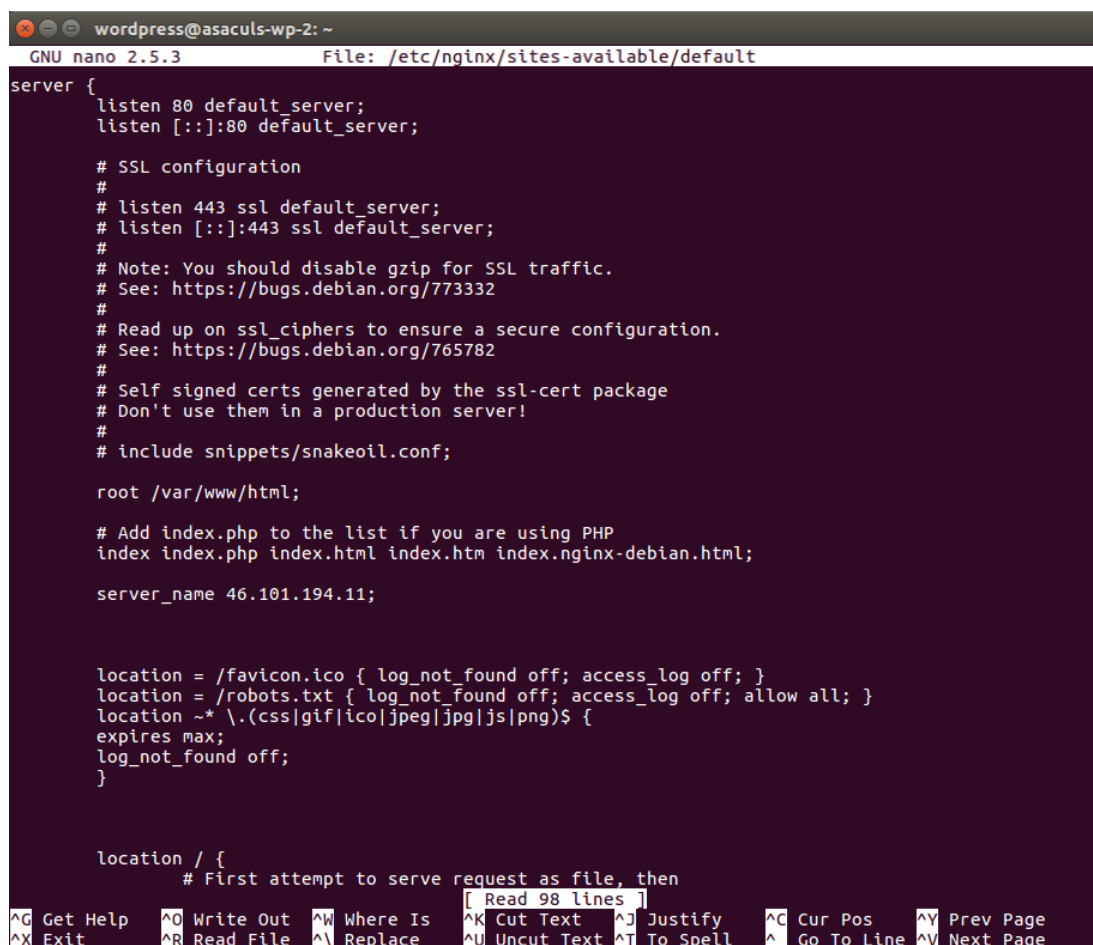
Figure 21 Configuration of My SQL server for WordPress (source: own)

The second phase of WordPress installation consisted of downloading WordPress installation files from wordpress.org and extracting it in to the designated webserver folder. Once extraction is finished, WordPress installation was run and finished in a few minutes.

The installation of Grav runs in a different way than WordPress, as there is no database setup needed. After creating a secondary user with *sudo* capabilities, installing PHP 7.0 Nginx, a terminal command is written to download and extract the Grav package from github.com.

4.2.2 Nginx

As all websites need a version of webserver to be able to run, Nginx was installed on both virtual machines as the primary webserver and configured according to the need of each website. Figure 22 below shows the configuration of Nginx for our WordPress site.



```
wordpress@asaculs-wp-2: ~
GNU nano 2.5.3 File: /etc/nginx/sites-available/default
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.php index.html index.htm index.nginx-debian.html;

    server_name 46.101.194.11;

    location = /favicon.ico { log_not_found off; access_log off; }
    location = /robots.txt { log_not_found off; access_log off; allow all; }
    location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {
        expires max;
        log_not_found off;
    }

    location / {
        # First attempt to serve request as file, then
        [ Read 98 lines ]
    }
}

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   ^Y Prev Page
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line  ^V Next Page
```

Figure 22 Configuration of Nginx for WordPress (Source: own)

4.3 Web Design

The design of the websites was carried out to make both have the same content and having the same number of pages. Both websites will have the following pages.

- Home
- Resources
- News
- About us
- Contact us

To make the websites have the same content not to affect website response times, both were populated with a dummy text with 300 words and 2,224 characters in each page of the websites except the contact us page. A selection of 3 pictures taken by the author and one video streaming from you tube about Czech University of Life Sciences were included to be able to inspect how the websites will behave.

More and more people are becoming mobile while working, leaving the traditional go to work approach to a job, which translates to work being done from different devices. Most people read, research and email on the go using different devices, like smartphones and tablets. This has created a need for websites to be responsive in design to be able to provide service to various types of devices. Both content management systems are designed to be responsive with the default theme they come with, eliminating the need to integrate a separate framework within the websites.

4.3.1 Graphics Theme

As both websites are to represent African student association and to represent it with in Czech University of Life sciences, the author decided to implement a color scheme for both websites that is representative of both aspects. The color theme was designed with the colors of the African union in consideration, alongside with Czech university of life

Sciences as both use shades of green and yellow as main colors. Figure 23 below shows the color plate made on adobe color to represent the design aspects of the website.

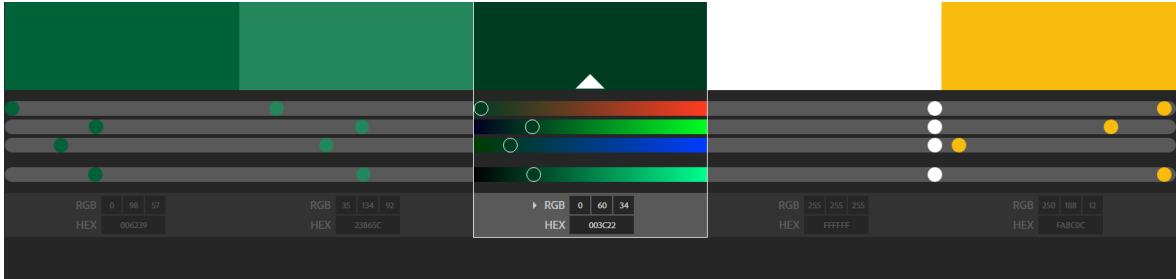


Figure 23 Color branding selection (source: own)

4.4 Test Cases

In an ideal environment, comparing two different Web Content Management Systems would be done on a host machine with dedicated CPU and RAM as it would provide accurate means of measuring the level of stress that is taken by the system. As renting a dedicated private server is very expensive in most cases and as it does not represent the real-world scenario involving this study, the next best way to follow is to obtain virtual machines.

As reflected by the literature review, virtual machines provide the opportunity to own some part of the CPU as well as RAM of the machine which allows proper and accurate measurement of resources utilized. For testing purposes, two Virtual Machines were acquired from cloud hosting provider, Digital Ocean.

4.4.1 Test Case A

For test case A, both virtual machines were linked with new relic server monitoring tool to enable the author to record resources utilized. In addition, both virtual machines were verified by a unique token to be able to proceed with sending virtual traffic to the sites from loader.io. Test case A will focus on sending user per second, ramp up per given time and users per given time tests to observe how the to the websites.

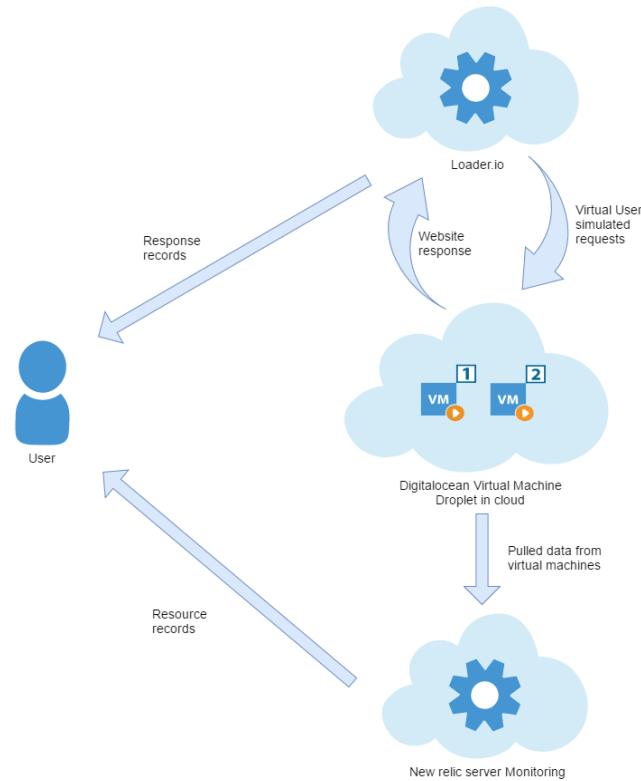


Figure 24 Load testing case A (source: own)

On a client per test, a total of 600 users within 60 seconds will be simulated to both of our websites. This test will help us identify if the predefined use case for the websites can be achieved. The second test is a maintained (ramp up) test of 0 – 150 virtual users simulated over 60 seconds. This test will help us in determining how a website will react with regards to concurrent virtual requests when virtual users are scaled up in time.

The third test that will be used is client per second test, which will simulate the number of clients the website can handle per each second in time. In this part of the test we will send 80 simulated virtual users per second for a duration of 60 seconds. This test will help us to stress test our website and see the breaking point of each of the designed websites.

| No | Test Name | VU's simulated | Time in seconds |
|----|-------------------|----------------|-----------------|
| 1 | Client per test | 600 | 60 |
| 2 | Client per second | 80/second | 60 |
| 3 | Maintained | 0 – 150 | 60 |

Table 1 Summary of tests for loader.io (source:own)

4.4.2 Test Case B

On test case B, we will be testing for real world usage simulations using LoadImpact.com. The simulation will consist of 100 virtual users simulating normal website page navigation and request for a time of 5 minutes. For this purpose, a test case has been recorded and is ready for execution using load impacts scenario recording tool and has been edited to fit the needs of the test.

Furthermore, the test will be carried out to simulate the navigation of the websites using a chrome browser under 3G mobile network to see how the websites respond and utilize resources as they are being called from another geographical location. The reason for this is, as the nature of the websites will be an African Students Association website, it should be considered that some of the traffic the websites will generate will be from the African continent with a 3G connection. Based on this test we will be able to see the response and error rates of the website. Since Africa as a possible location does not exist within loadimapct.com test scenario, the author has opted to test the virtual users from Dublin, Ireland.

4.4.3 Questionnaire

The last part of the test is a questionnaire conducted within the student body of Czech University of Life Sciences to gauge the usability of the websites as students and future students will be the prospective users of the websites designed. The type of the questions used in the study will be scaled questions to better analyze the information gathered. Each website will have a questionnaire designed for this purpose and participants will be asked to navigate the website and answer questions aimed at user experience.

As the collected data is going to reflect on the people's attitude and personal experience towards the websites it will be represented in a Likert scale. A total of 20 questions were designed to gain insight in to the usability of the websites as well as to record the responses of students within the campus. The questionnaires were divided in to two groups to represent the two different websites with each website having 10 questions. Respondents will have the options to select from 5 choices so that they can the one that

best reflects their experiences about the websites. Table 2 below shows the summary of questions alongside their representations for simplified use later.

| Question | Representation |
|--|-----------------------|
| I can clearly see the menu | Q1 |
| I can see the content (text and letters) on the website | Q2 |
| I can see the title of the page | Q3 |
| I know where I am on the website | Q4 |
| Changing pages is fast and easy | Q5 |
| The website loads fast | Q6 |
| I can easily find what I want in this website | Q7 |
| I don't need to scroll left and right to see the contents | Q8 |
| The design of the website is attractive | Q9 |
| The content of this web is organized. | Q10 |

Table 2 Summary of Questions for survey (source:own)

5 Results and Discussion

5.1 Loader.io test

5.1.1 Loader.io wordpress

As shown in figure 25 below, the client per test method of test for handling 600 clients inside of 60 seconds, with 10 clients making a request every second, showed that the test was concluded with 0% error (no errors found) for both internal server error 500 and 400 types. There was no timeout connection or network errors during the test. The website response time for all the requests had an average of 453ms with a maximum response time of 652ms and a minimum response time of 348ms. The test was concluded with 100% success rate with a total of 1200 requests made.



Figure 25 Client per test results for wordpress (source: own)

As shown in figure 26 below, New relic server resources monitoring showed that, during the test, the utilization of RAM by WordPress remained unchanged at 46% of the 521 MB installed (488 available) and 22.6% of the CPU was used during testing. A 1.21 Mb/second rate of data transfer was registered. The effect of the test on Disk utilization was negligible and Disk IO remained at 0.04% for the duration of the test.

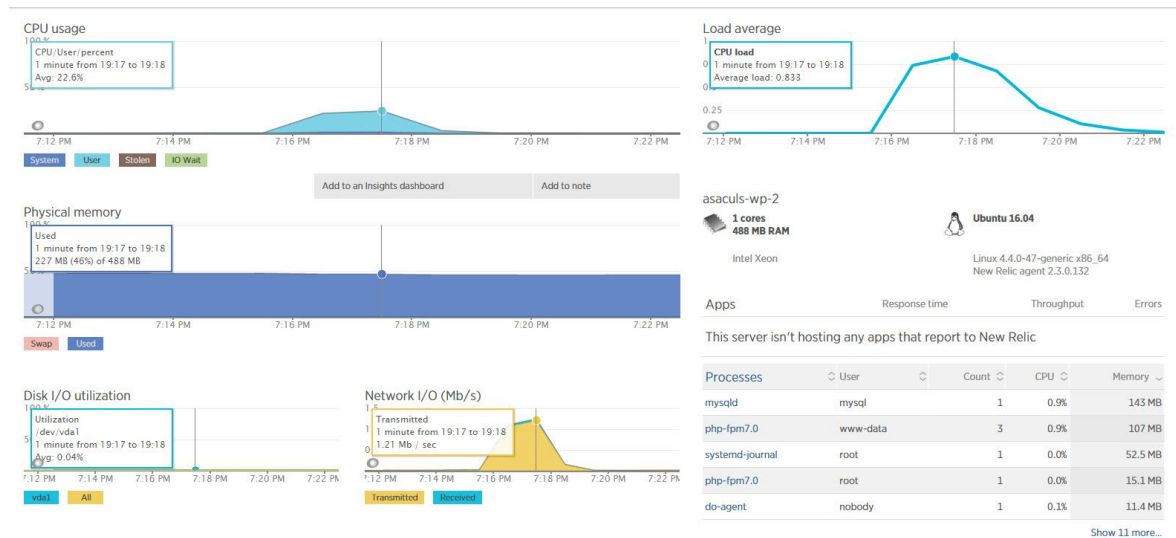


Figure 26 Wordpress resource utilization results for client per test (source:own)

On the maintained (ramp-up) request test from clients starting at 0 – 150 within 60 seconds, It was observed in figure 27 below, that 2544 successful requests got responses. As 140 clients made recursive requests from the server, 237 occurrences of internal server error 500 were recorded. There was no server error 400 observed. An average response time of 2688ms was recorded until 55th second.

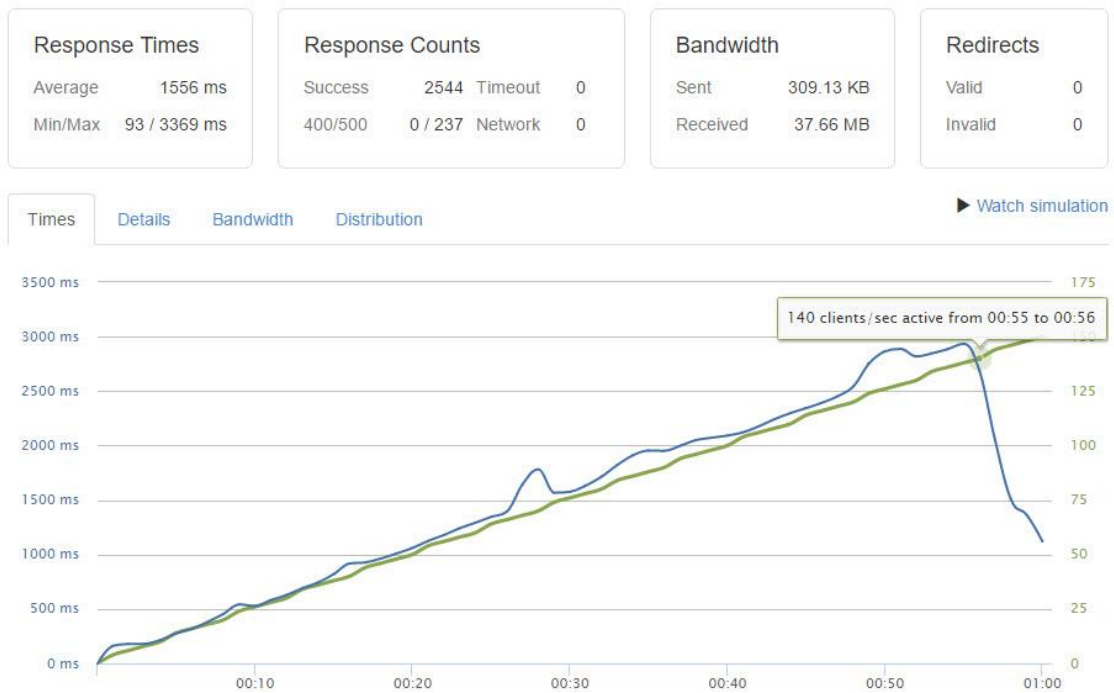


Figure 27 Maintained client test results for wordpress (source: own)

In figure 28 below, it is shown that from new relic server monitor, during the ramp up load test, a maximum of 56.9 % of CPU usage which suggests that CPU intensive tasks were carried out. with 46% of RAM utilized. A network transmission of 3.08 Mb/sec was observed.

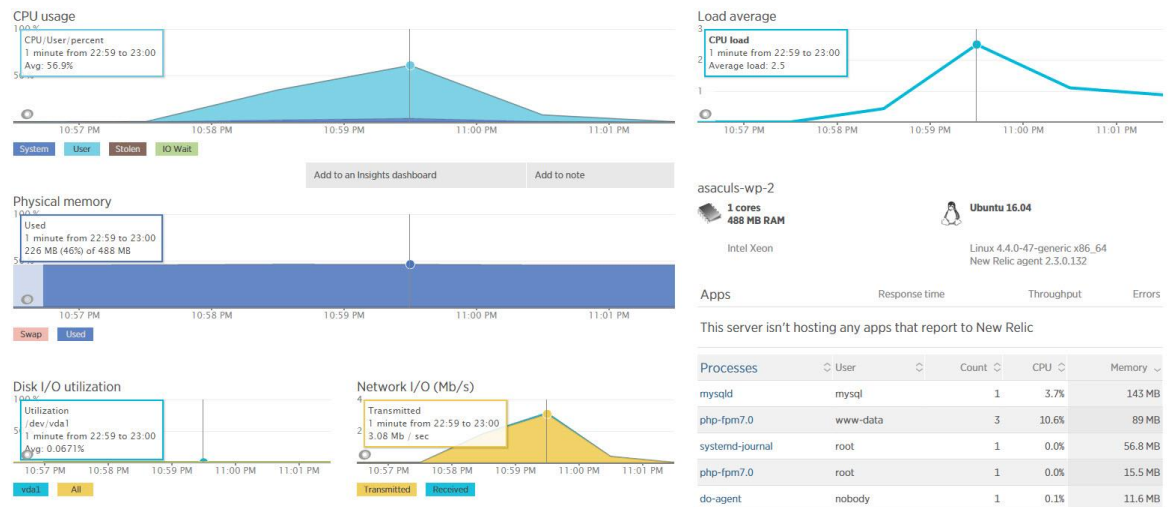


Figure 28 Wordpress resource utilization results for maintain client test (source: own)

On our client per second test, figures 29 and 30 below show that, on 80 clients per second for 60 second test, an error rate of 28.2 was observed with 750 instances of internal error 500 and no internal error 400. As the error responses from the server came starting at the 4th second, the average and minimum values for response times are highly influenced and are not considered as they will report a wrong time. This result suggests that the handling threshold of the WordPress site has been surpassed.

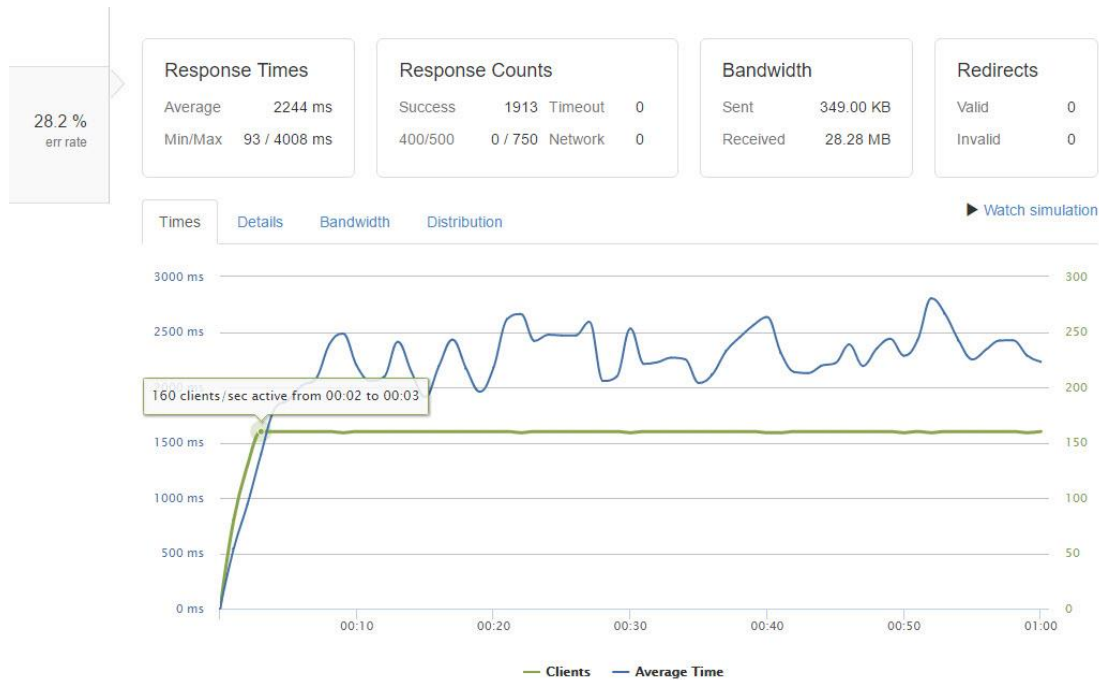


Figure 29 Client per second test for wordpress (source:own)

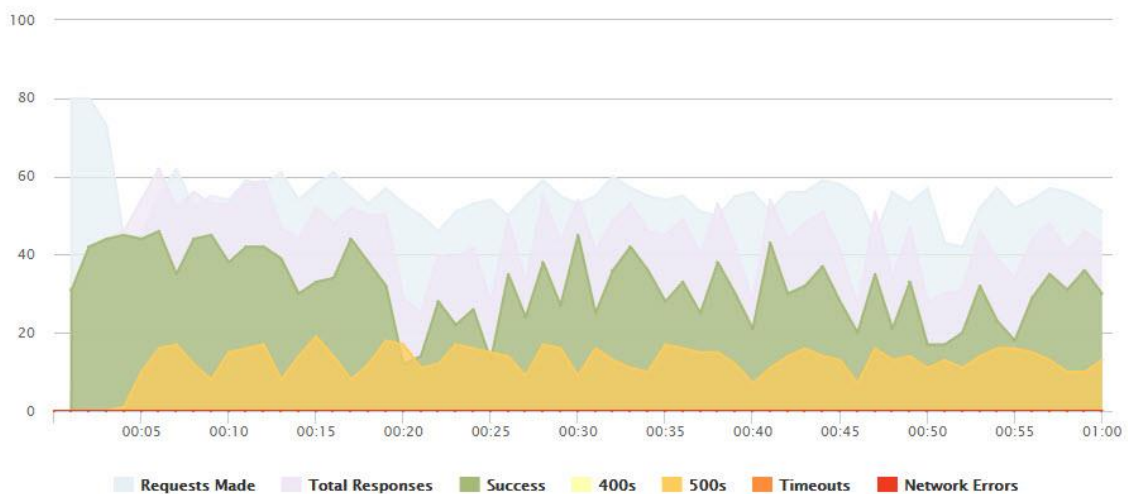


Figure 30 Client per second test details wordpress (source:own)

On figure 31 below shows that the average CPU usage was 74.5% and compared to the other tests, this is the highest usage recorded. RAM usage remained at 47 % of the total while Disk IO remained at negligible figures. It was observed that there was 4.05 Mb/second data transmitted during the test.

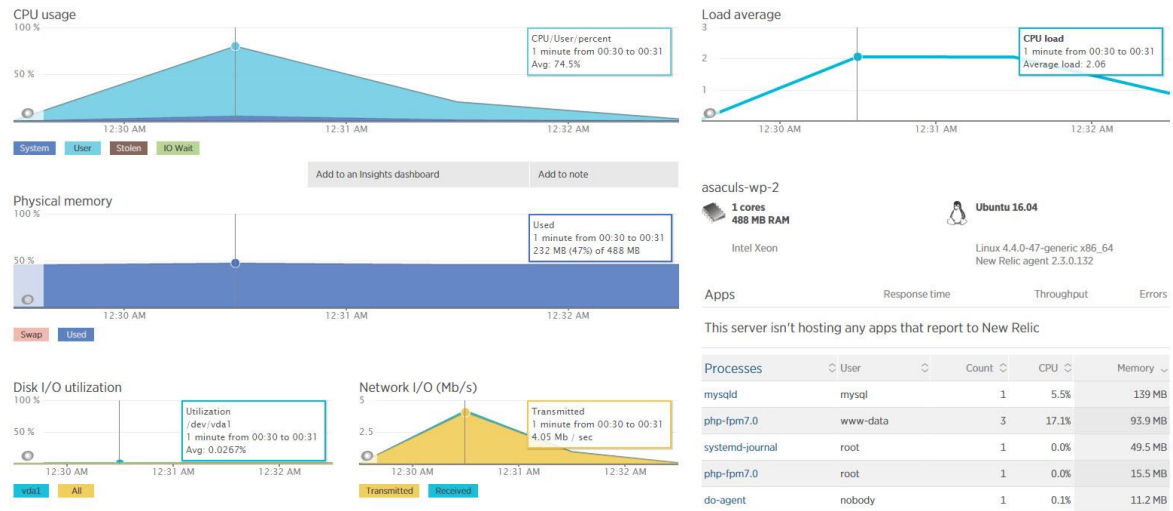


Figure 31 Wordpress resource utilization results for client per second test (source: own)

5.1.2 Loader.io Grav

From figure 32 below, for the client per test method of testing 600 virtual users within 60 seconds, show that all 600 of the virtual users successfully made the request. Furthermore, the average response time for the requests was 134ms with a minimum of 101ms and a maximum of 255ms. There was no internal server error 400 and 500 recorded in the duration of the test. This suggests that clients of such magnitude can be handled very well by the Grav version of the website. Moreover, 0 network errors and 0 timed out connections were also observed with the test.

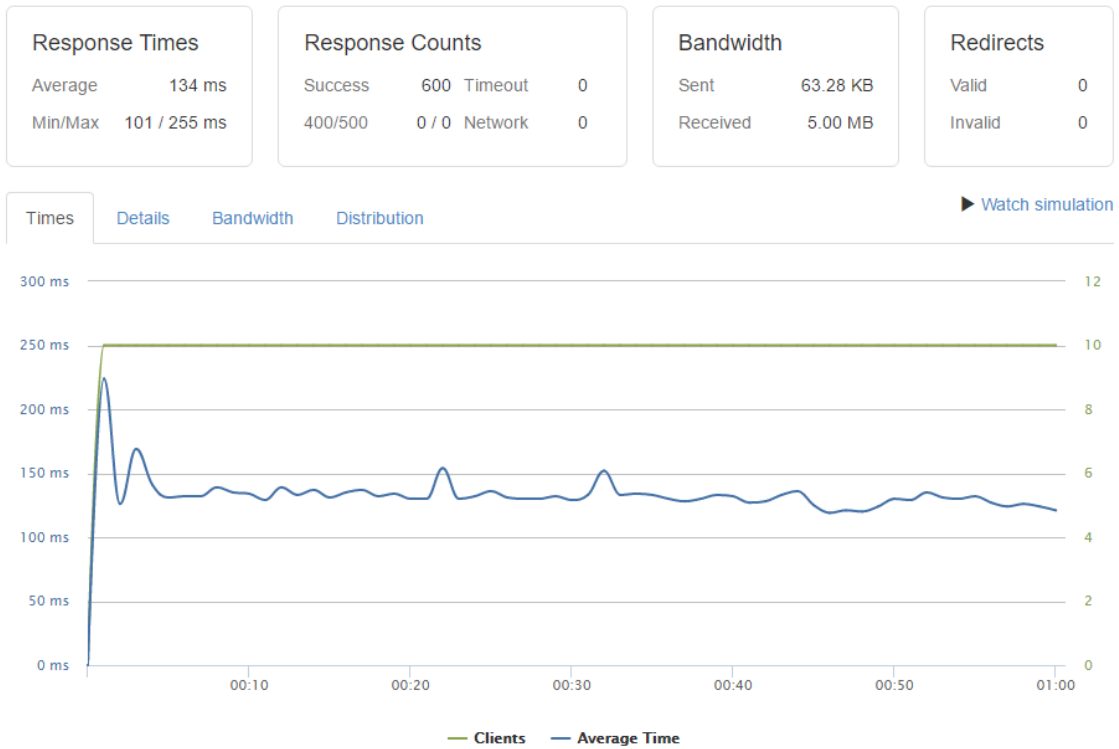


Figure 32 Client per-test results for Grav (source: own)

From the figure 33 below we can see from new relic server monitoring tool that, there was only a 5% usage of CPU while RAM usage stayed flat at 16% (79.4 MB out of 488 MB). Disk utilization and load average remained unchanged. This suggests that Grav site is not stressed in handling the virtual users for this test.

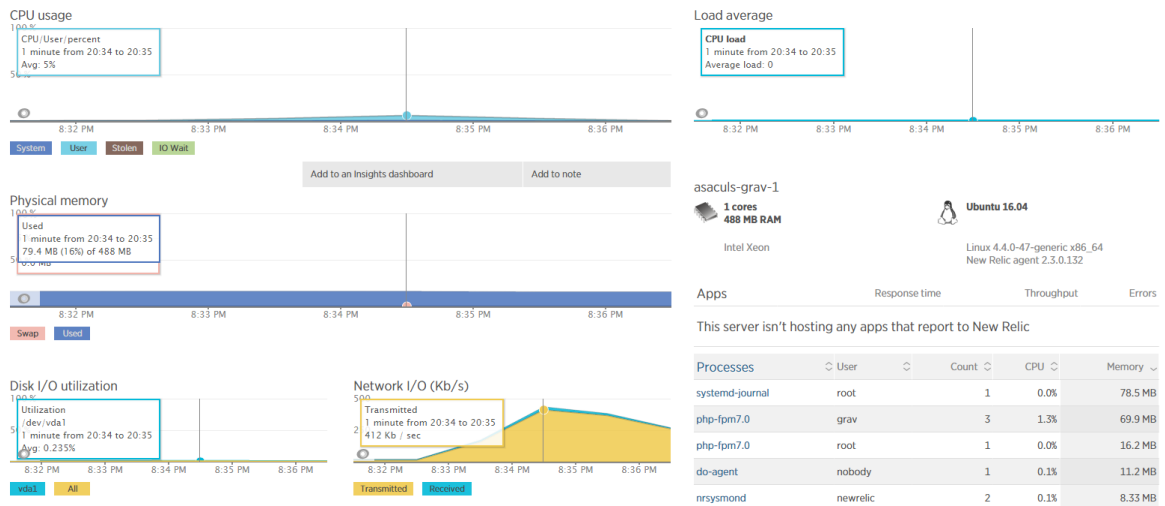


Figure 33 Grav resource utilization results for client per test (source: own)

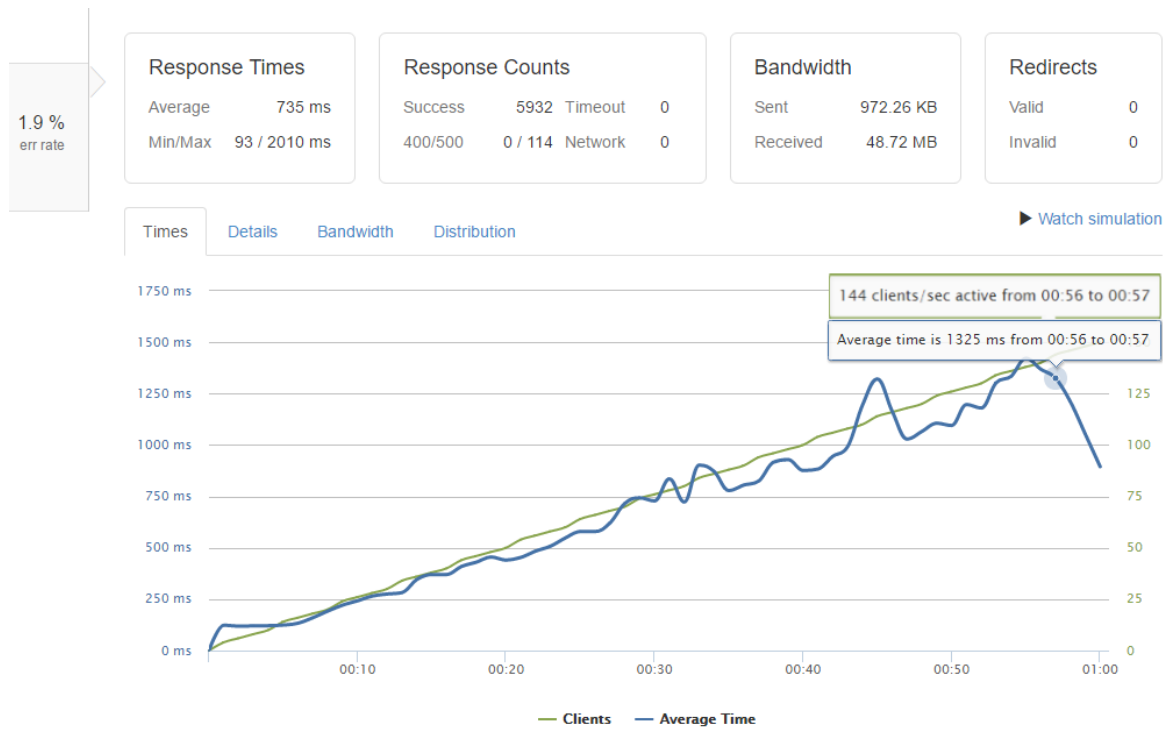


Figure 34 Maintained client test results for Grav (source: own)

Figure 34 above illustrates that on a maintained (ramp-up) of 0 – 150 clients within 60 second test, there was an error rate of 1.9% with 114 Internal server error 500. There were 5932 successful responses counted, however due to the 114 counts of error 500, the minimum and maximum values are affected and not taken in to consideration. We can also see from the figure that on the 57th second, prior to the errors started happening, Grav was registering an average response time of 1325ms.

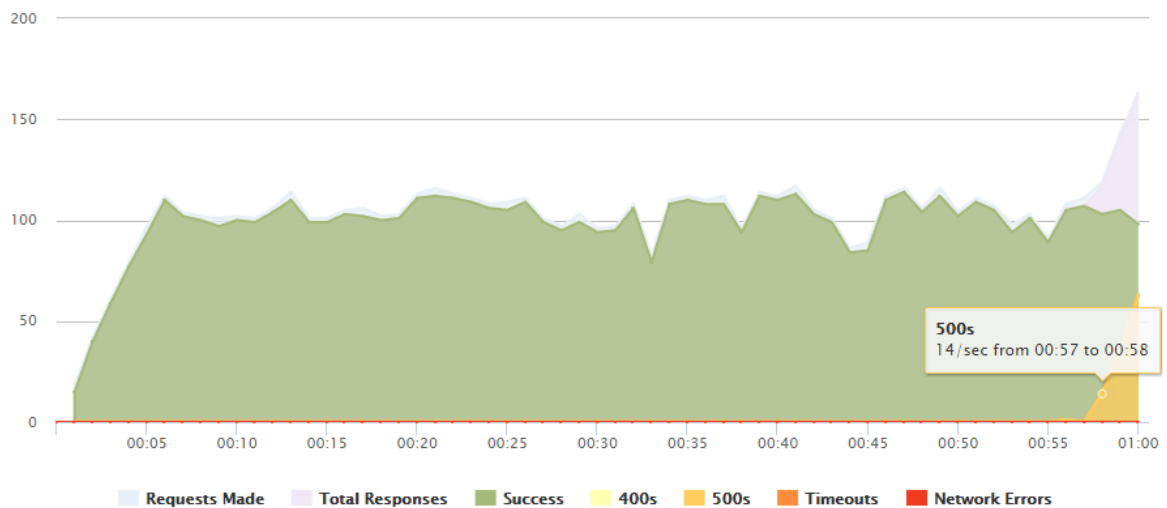


Figure 35 Maintained client test failure Grav (source: own)

From figure 35 above, we can see that during the 57th second of the test, internal server error 500 started to appear at 14 errors per second. There was no internal server error 400 recorded during the test and furthermore there has been no network and timeout connection recorded.

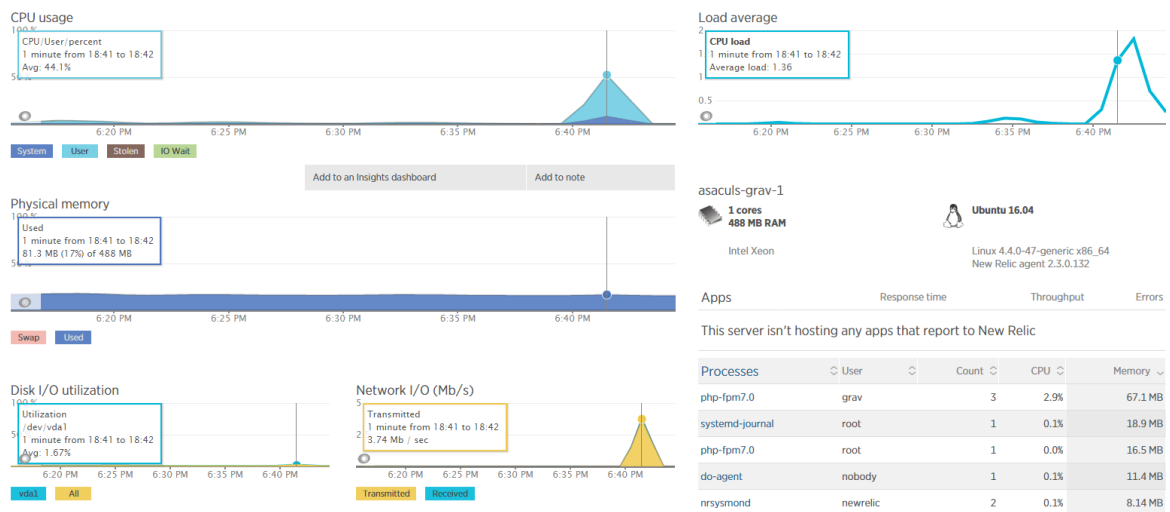


Figure 36 Grav resource utilization results for maintain client test (source: own)

From figure 36 above, we can see that 44.1% of the CPU was used while ram usage stayed at 17% during the test unchanged. Even though Grav registered a spike in processing load, it the figure shows that it still didn't reach its threshold. There was 1.67% usage of disk read/write speeds on average and network transmission of 3.74 Mb/second.

From figure 37 below, we can see that for the client per second test with 80 clients within 60 seconds, it was observed that the test concluded with 0% errors with 4795 successful tests. The average response time was 156ms with a minimum of 102ms and a maximum of 424ms. There was no internal server error 400 and 500 observed in the duration of the test suggesting that Grav site handled the load ease.

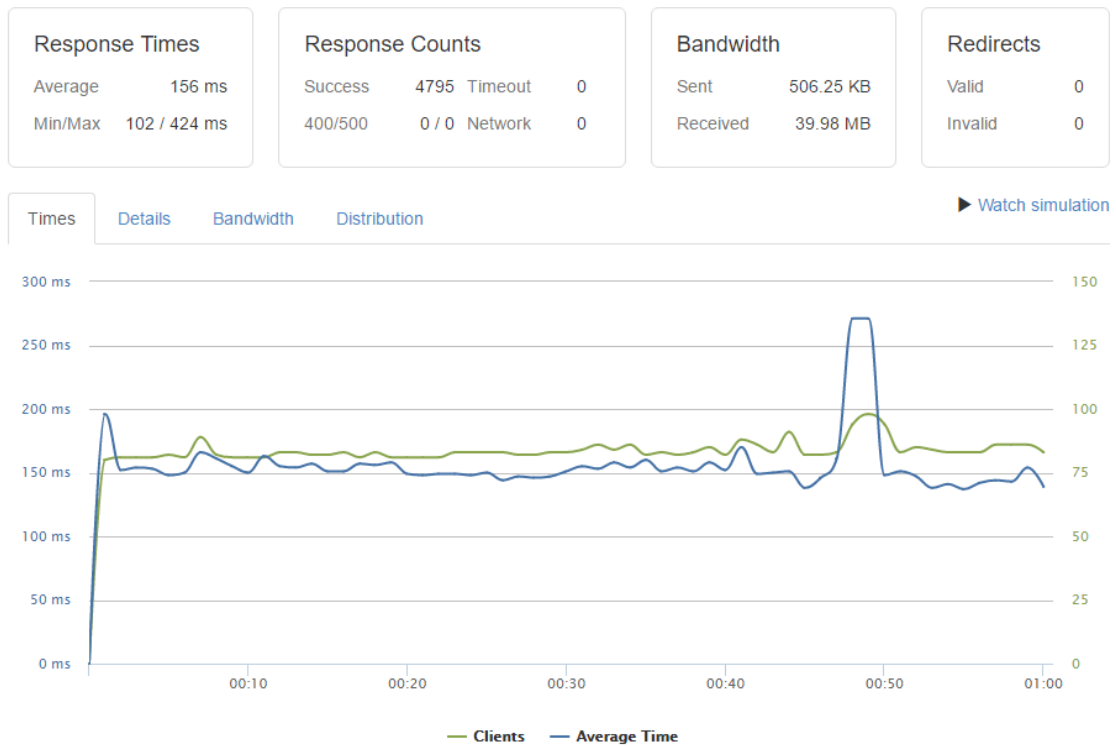


Figure 37 Client per second test for Grav (source:own)

From new relic, we can see from the figure 38 below that only 33.2 % of the CPU was utilized on average with the RAM remaining at 16%. There was a small Disk read/write observed at 0.186% on average and a network data transmission of 0.0139 Mb/second suggesting that although it was a demanding test, the resources and function of Grav remained unchanged.

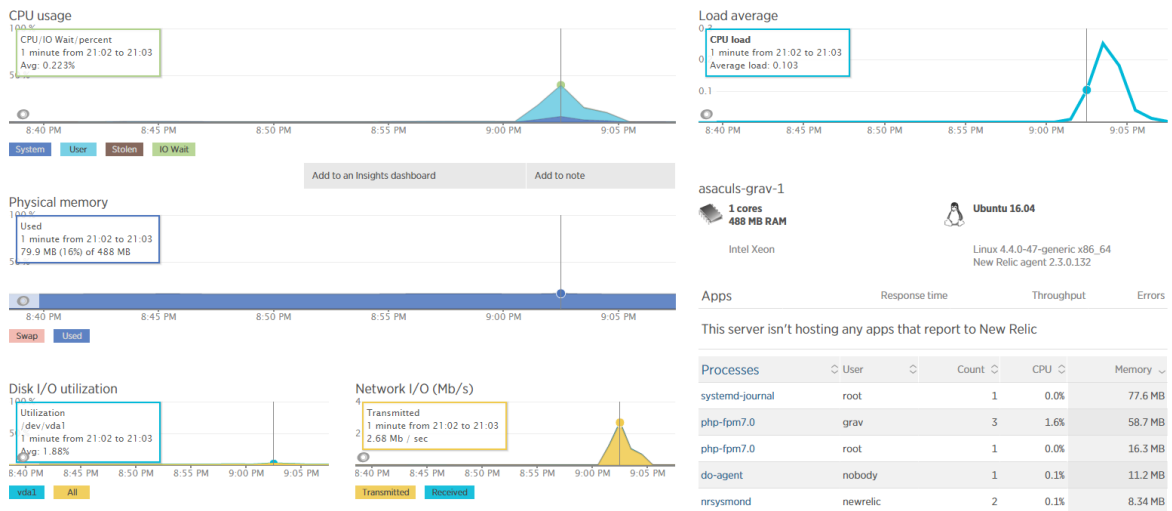


Figure 38 Grav resource utilization results for client per second test (source: own)

5.2 Load Impact test

5.2.1 Load Impact WordPress

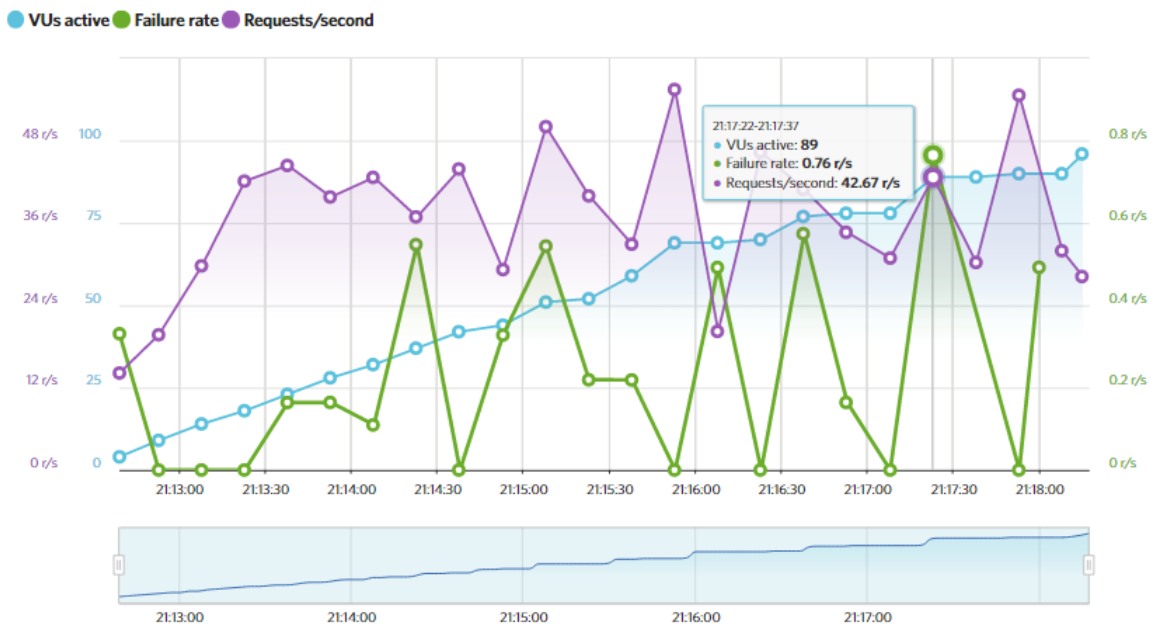


Figure 39 LoadImpact scenario test for WordPress (source:own)

From figure 39 above, it is observed that at 89 Virtual users concurrently requesting information from the website, there existed a failure rate of 0.76/second, with 42.67 rates of requests per second.

As we can see from figure 40 below, new relic server monitoring tools registered no significant change in the utilization of infrastructure during the test. RAM usage was at 46% with 4.77% use of CPU indicating that the virtual machine was not under stress processing the requests.

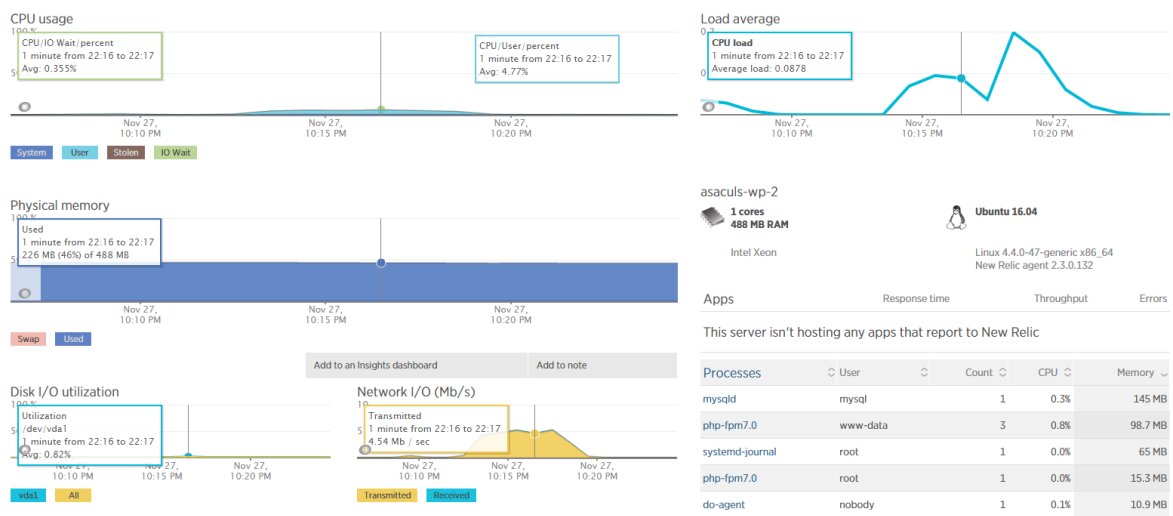


Figure 40 Wordpress resource utilization for loadimpact (source: own)

5.2.2 Load Impact Grav

At a maximum of 100 virtual users performing concurrent tasks on chrome browser with a 3G internet connection for 5 minutes, figure below shows that the highest failure rate observed was less than 0.76 request/second with 89 virtual users active and 64.2 requests/second. This suggests that the website has negligible fail rates with concurrent active users navigating through the website.

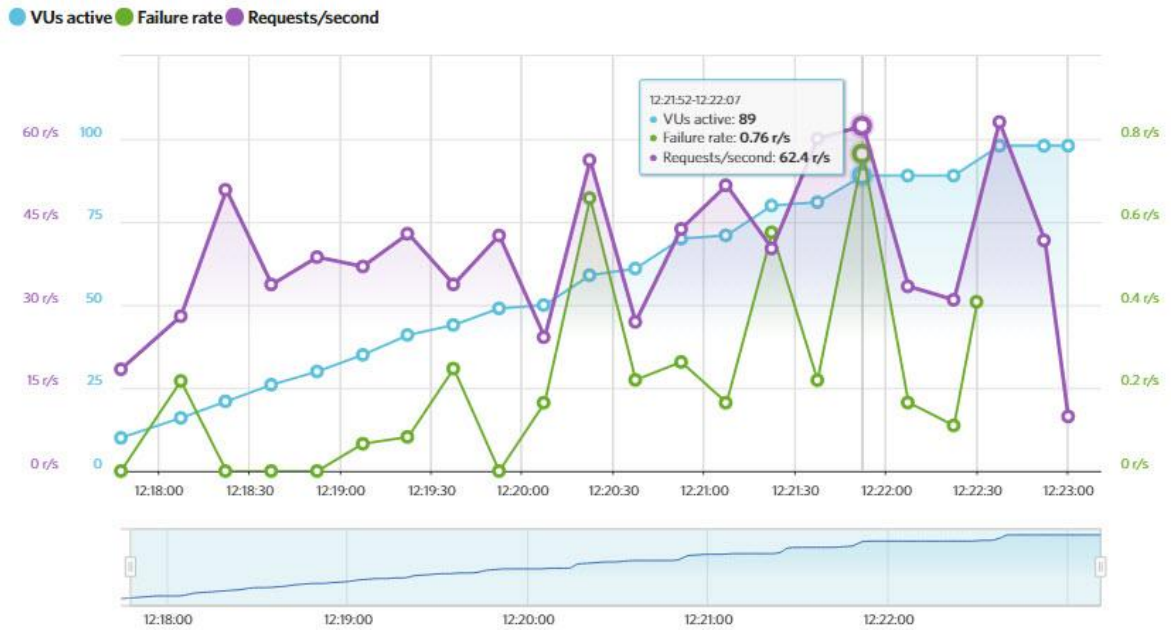


Figure 41 LoadImpact scenario test for Grav (source:own)

Going over to new relic metrics, we can observe from figure 42 below that CPU registered an average of 2.49 % usage during the test and RAM usage remained at 17% (80.8 MB), without having any significant change. This suggests that the website responded well with the test having no significant effect on resource utilization.

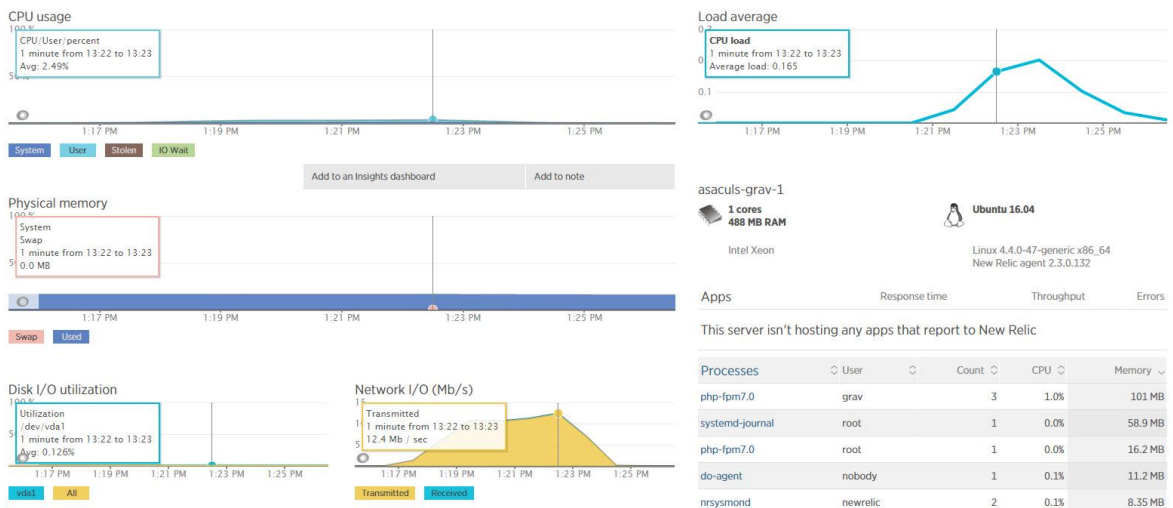


Figure 42 Wordpress resource utilization for Grav (source: own)

5.3 Questioners

There was a total of 48 responders to the two-part Questionnaires, each part focusing on a different website. the following table shows the summary of all the questionnaires. WP shows responses for WordPress site and G represents responses for Grave site.

| Question | I strongly agree | | I am Indifferent | | Its normal | | Disagree | | Strongly Disagree | |
|------------|------------------|-----|------------------|-----|------------|----|----------|----|-------------------|----|
| | WP | G | WP | G | WP | G | WP | G | WP | G |
| Q1 | 79% | 70% | 17% | 30% | 4% | 0% | 0% | 0% | 0% | 0% |
| Q2 | 79% | 59% | 21% | 41% | 0% | 0% | 0% | 0% | 0% | 0% |
| Q3 | 88% | 75% | 4% | 21% | 4% | 0% | 4% | 4% | 0% | 0% |
| Q4 | 58% | 50% | 29% | 50% | 8% | 0% | 4% | 0% | 0% | 0% |
| Q5 | 25% | 59% | 42% | 41% | 33% | 0% | 0% | 0% | 0% | 0% |
| Q6 | 17% | 48% | 38% | 52% | 33% | 0% | 13% | 0% | 0% | 0% |
| Q7 | 54% | 43% | 29% | 57% | 17% | 0% | 0% | 0% | 0% | 0% |
| Q8 | 88% | 79% | 8% | 21% | 4% | 0% | 0% | 0% | 0% | 0% |
| Q9 | 79% | 71% | 4% | 29% | 13% | 0% | 0% | 0% | 4% | 0% |
| Q10 | 71% | 74% | 17% | 26% | 13% | 0% | 0% | 0% | 0% | 0% |

Table 3 Summary of responses for WordPress and Grav (source:own)

From table 3 above, it was observed that responders preferred WordPress for, Question 1(I can clearly see the menu) and clarity of the display of the text, Question 2 (I can see the content text and letters on the website), Indicating navigation was better in wordpress than in grav. It was observed that end users preferred Grav with regards to Questions 5 (Changing pages is fast and easy) and 6 (the website loads fast) which indicates that Grav responds faster for endusers. The responses from the rest of the questions showed no significant favoritism to make a solid conclusion.

The test load.io of both websites indicated that, on the client per test evaluation, the WordPress site averaged 453ms with a minimum of 348ms and maximum of 653ms, while the Grav site managed 134ms response time on average with a minimum of 101ms and 255ms respectively. From this we can see that the Grav site is 319ms faster than the WordPress site. Considering that this test was a smoke screen test i.e. meant to see the performance of the websites under very low amount of stress, to have a difference of 319ms represents a significant difference. We can also observe that Grav utilizes far more less ram than WordPress during the test phase with WordPress RAM on average at 46% while Grav stayed at 16% having a 30% difference.

For the maintained ramp-up test, it was observed that while the WordPress site registered 237 internal server errors with a peak of serving 140 clients while the Grav site had 114 errors while serving at its peak 144 concurrent clients. The WordPress site generated 93 errors more for the same test which also indicates that Grav handles requests better during a high traffic time. It was also observed that WordPress used 56.9% of CPU and 46% of RAM while Disk read/write was negligible where as Grav used 44.1% of the CPU and only 17% of the RAM while disk read/write was negligible.

On the client per second test, WordPress showed a 28.2% error rate having 750 internal network error 500 instances while having 1913 successful responses. WordPress also started to generate error messages starting the 4th second during the 60 second test. Grav concluded the same test with 0% errors and 4795 successful responses and had a response time of 156ms on average with a minimum of 102ms and a maximum of 424ms. During the test, it was also recorded that WordPress used 74.5% of the CPU on average and 47% of the RAM with negligible disk read/write. On the same test, Grav used 33.2 % of the CPU and 16% RAM with 1.88% of disk read/write. The result of the test shows that Grav performs better under load than WordPress.

Scenario simulated tests from load impact showed that both websites handled the simulation successfully.

6 Conclusion

Websites relying on content management systems mostly stick to only one type of framework, disregarding the nature of the website to be designed. Researching a better and faster solution that caters to the specific needs of the website should come first before the decision to use a certain type of content management system. This study has shown that using WordPress because of its versatile nature can be a wrong decision, especially when considering it for a small website like an African Students Association within a Czech university of Life Sciences, meant to handle a small amount of traffic.

The study has shown that for the moderate number of virtual users simulated, Grav has outperformed WordPress in all the load tests while maintaining a very low amount of resource utilization. In addition, this study has also shown that regarding user experience, although WordPress showed to have better response from the sample population, there is no major significant difference between the two websites, underlining the issue that Grav needs to have more developers in its community in developing themes and collaborating as to bring the level of theming and experience to that of WordPress.

References

1. ADOBE, 2016. Media alert: Adobe data shows black Friday breaks online sales record with \$3.34 Billion. [online] 2016. [Accessed: 27 November 2016]. Available at: <http://news.adobe.com/press-release/marketing-cloud/media-alert-adobe-data-shows-black-friday-breaks-online-sales-record-3>
2. BARKER, Dean. 2016. Web Content Management: O'Reilly Media, 2015. ISBN: 978-1-491-90812-9.
3. BROOK, Chris. 2015. Attackers targeting Unpatched Joomla sites through SQL injection vulnerability. [online] 2015 [Accessed: 29 June 2016]. Available at: <https://threatpost.com/attackers-targeting-unpatched-joomla-sites-through-sql-injection-vulnerability/115179/>
4. CAFELOG.COM. 2001. B2 - a classy weblog tool. [online] 2001. [Accessed: 10 October 2016]. Available at: <http://cafelog.com>
5. DOCTRINE DOCUMENTATION. 2010. [online] 2010. [Accessed: 25 October 2016]. Available at: <http://docs.doctrine-project.org/en/latest/reference/caching.html>
6. DRUPAL, 2016. Our History. [online] 2016. [Accessed: 14 June, 2016]. Available at: <https://www.drupal.org/about/history>
7. GRAV DOCUMENTATION. 2016. Grav Documentation. [online] 2016. [Accessed: September 30, 2016]. Available at: <https://learn.getgrav.org/basics/what-is-grav>
8. GRUBER, John. 2004. Daring fireball: Markdown. [online] 2004. [Accessed: 26 July 2016]. Available at: <https://web.archive.org/web/20040402182332/http://daringfireball.net/projects/markdown/>

9. LOAD IMPACT. 2016. Website load testing. [online] 2016. [Accessed: 11 June 2016]. Available at: <https://loadimpact.com/website-testing>
10. LOADER.IO. 2014. Test types. [online] 2014. [Accessed: 16 February 2016]. Available at: <http://support.loader.io/article/16-test-types>
11. Molyneaux, Ian. 2014. The art of application performance testing: Help for programmers and quality assurance. second edition edn. United States: O'Reilly Media, Inc, USA.
12. MULLENWEG, Matt. 2003a. WordPress Now Available. [online] 2003. [Accessed: 14 March, 2016]. Available at: <https://wordpress.org/news/2003/05/wordpress-now-available/>
13. MULLENWEG, Matt. 2003b. 0.72 final version available. [online] 2003. [Accessed: 14 March, 2016]. Available at: <https://wordpress.org/news/2003/05/wordpress-now-available/>
14. PUGH, Emerson.W. 1995. Building IBM: Shaping an industry and its technology. Cambridge, MA: MIT Press.
15. RUSEV, Emanuil. 2013. Parsedown. [online] 2013. [Accessed: 17 July 2016]. Available at: <https://github.com/erusev/parsedown>
16. SANTILLAN, Maritza. 2015. One Million WordPress Websites vulnerable to SQL injection attack. [online] 2015. [Accessed: 11 June 2016]. Available at: <https://www.tripwire.com/state-of-security/latest-security-news/one-million-wordpress-websites-vulnerable-to-sql-injection-attack/>
17. SENSIOLABS. 2016. The flexible, fast, and secure template engine for PHP. [online] 2016. [Accessed: October 10, 2016]. Available at: <http://twig.sensiolabs.org>

18. ST.LAURENT, Andrew M. 2004. Understanding open source and free software licensing. United States: O'Reilly Media, Inc, USA.
19. SWARTZ, Aaron. 2004. Markdown (Aaron Swartz: The Weblog).[ONLINE] 2004. [Accessed: 17 July 2016]. Available at: <http://www.aaronsw.com/weblog/001189>
20. VMWARE. 2016. VSphere documentation center. [online] 2016. [Accessed: 08 August 2016]. Available at: https://pubs.vmware.com/vsphere-50/index.jsp?topic=%2Fcom.vmware.vsphere.vm_admin.doc_50%2FGUID-CEFF6D89-8C19-4143-8C26-4B6D6734D2CB.html
21. W3.ORG. 1999. HTTP/1.1: Method definitions. [online] 1999. [Accessed: 29 November 2016]. Available at: <https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>
22. W3SCHOOLS, 2016. SQL Injection. [online] 2016. [Accessed: 21 January, 2016]. Available at: http://www.w3schools.com/Sql/sql_injection.asp
23. W3TECHS 2016. Usage of content management systems for Websites. [online] 2016. [Accessed: 13 November 2016]. Available at: https://w3techs.com/technologies/overview/content_management/all
24. WATSON, Leon. 2015. Humans have shorter attention span than goldfish, thanks to smartphones. [online] 2015. [Accessed: 26 January 2016]. Available at: <http://www.telegraph.co.uk/science/2016/03/12/humans-have-shorter-attention-span-than-goldfish-thanks-to-smart/>
25. WEATHERHEAD, Rob. 2014. Say it quick, say it well – the attention span of a modern internet consumer. [online] 2014. [Accessed: 17 May 2016]. Available at: <https://www.theguardian.com/media-network/media-network-blog/2012/mar/19/attention-span-internet-consumer>

26. YOUTUBE. 2016. Statistics. Youtube. [online] 2016. [Accessed: September 08,2016.]. Available at: <https://www.youtube.com/yt/about>
27. YOUTUBE. 2016. Statistics. Youtube. [online] 2016. [Accessed: September 08,2016.]. Available at: <https://www.youtube.com/yt/press/statistics.html>

Appendix - A

Grav life cycle as taken from getgrav.org

index.php

1. Check PHP version to ensure we're running at least version **5.5.9**
2. Class loader initialization
3. Obtain Grav instances

Grav.php

1. No instance exists, so call `load()`
2. Add `grav`
3. Initialize the debugger and add it to `debugger`
4. Register the `log` handler
5. Register the error handler
6. Add `url`
7. Add `task`
8. Add `events`
9. Add `cache`
10. Add `session`
11. Add `plugins`
12. Add `themes`
13. Add `twig`
14. Add `taxonomy`
15. Add `language`
16. Add `assets`
17. Add `base_url_absolute`
18. Add `base_url_relative`
19. Add `base_url`
20. Register the `stream` handler
21. Register the `config` handler

Grav.php

4. Call `$grav->process()`
1. Initialize the configuration
2. Initialize the Session
3. Initialize the URI object
4. Initialize the error handler
5. Initialize the debugger
6. Start output buffering
7. Initialize the timezone
8. Initialize the `plugins`
9. Fire `onPluginsInitialized` event
10. Initialize the theme
11. Fire `onThemeInitialized` event
12. Fire `onTaskLoaded` event
13. Initialize `assets`
14. Fire `onAssetsInitialized` event
15. Initialize `twig`

Twig.php

1. Set Twig template paths based on configuration
2. Handle language templates if available
3. Fire `onTwigTemplatePaths` event
4. Load Twig configuration and loader chain
5. Fire `onTwigInitialized` event
6. Load Twig extensions
7. Fire `onTwigExtensions` event
8. Set standard Twig variables (`config`, `url`, `taxonomy`, `assets`, `browser`, etc)

Pages.php

16. Initialize pages
1. Call `buildPages()`
2. Check if cache is good
3. If `cache is good` load pages data from cache
4. If `cache is not good` call `recurse()`
5. Fire `onBuildPagesInitialized` event in `recurse()`
6. If a `.md` file is found:

Page.php

1. Call `init()` to load the file details
2. Set the `filemeta` and `meta` fields
3. Call `header()` to initialize the header variables
4. Call `slug()` to set the URL slug
5. Call `visible()` to set visible state
6. Set `renderTwig()` status based on if folder starts with `.`
7. Fire `onPageProcessed` event
8. If a `folder` is found `recurse()` the children
9. Fire `onFolderProcessed` event
10. Call `buildRoutes()`
11. Initialize `taxonomy` for all pages
12. Build `media` table for fast lookup

17. Fire `onPagesInitialized` event
18. Fire `onPageInitialized` event
19. Add the debugger CSS/JS to the assets
20. Get Output with `Twig's processSite()` method

Twig.php

1. Fire `onTwigSiteVariables` event
2. Get the page output
3. Fire `onTwigPageVariables` also called for each modular subpage
4. If a page is not found or not routable, first fire the `onPageFallbackUrl` event to see if we have a fallback for a media asset and then fire `onPageNotFound` if not
5. Set all Twig variables on the Twig object
6. Set the template name based on file/header/extension information
7. Call `render()` method
8. Return resulting HTML

21. Fire `onOutputGenerated` event
22. Set the HTTP headers
23. Echo the output
24. Flush the output buffers to the page
25. Fire `onOutputRendered` event
26. Connection to client is closed
27. Fire `onShutdown` event

Whenever a page has its `content()` method called, the following lifecycle occurs:

Page.php

1. Fire `onPageContentRaw` event
2. Process the page according to Markdown and Twig settings. Fire `onMarkdownInitialized` event
3. Fire `onPageContentProcessed` event

Appendix - B

WordPress website - 46.101.194.11


ASACULS
African Students Association in CULS

Home Resources Events News About Us Contact Us

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

NULLA HENDRERIT LACINIA ENIM, AT LUCTUS LOREM PLACERAT COMMODO.



Lorem ipsum dolor sit amet, **consectetur adipiscing elit**. Nulla hendrerit lacinia enim, at luctus lorem placerat commodo. Cras tellus leo, egestas id pulvinar eget, aliquam non eros. Proin at lobortis erat. Nunc leo mi, eleifend quis posuere eget, scelerisque mollis elit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nulla in pellentesque ligula. Morbi iaculis tortor sem, vel finibus leo posuere id. Morbi ac sem laoreet, venenatis orci ut, ultrices massa. Donec eget tincidunt sapien. Quisque ac egestas enim, ac vehicula tellus. Aliquam porttitor sed lectus at ullamcorper. Duis at sagittis diam, at accumsan orci. *Cras aliquam mauris quam, a tincidunt sem faucibus sit amet. Donec metus orci, egestas sit amet finibus eu, pharetra eu nisl.*

| *Praesent et condimentum est. Vestibulum ornare venenatis turpis.*

Appendix - C

Grav website - 138.68.71.127



Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nulla hendrerit lacinia enim, at luctus lorem placerat commodo.



Lorem ipsum dolor sit amet, **consectetur adipiscing elit**. Nulla hendrerit lacinia enim, at luctus lorem placerat commodo. Cras tellus leo, egestas id pulvinar eget, aliquam non eros. Proin at lobortis erat. Nunc leo mi, eleifend quis posuere eget, scelerisque mollis elit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nulla in pellentesque ligula. Morbi iaculis tortor sem, vel finibus leo posuere id. Morbi ac sem laoreet, venenatis orci ut, ultrices massa. Donec eget tincidunt sapien. Quisque ac egestas enim, ac vehicula tellus. Aliquam porttitor sed lectus at ullamcorper. Duis at sagittis diam, at accumsan orci. *Cras aliquam mauris quam, a tincidunt sem faucibus sit amet. Donec metus orci, egestas sit amet finibus eu, pharetra eu nisl.*

Prasent et condimentum est. Vestibulum ornare venenatis turpis.

Nullam ut ipsum convallis, consectetur nunc sit amet, vulputate erat. Duis eu sapien at eros tristique molestie eget eu neque. Duis sit amet ullamcorper tellus. Proin finibus, lectus et lobortis consectetur, augue elit condimentum sapien, sed elementum massa lacus ut enim. Phasellus pulvinar auctor ullamcorper. Interdum et malesuada fames ac ante ipsum primis in faucibus. Quisque consectetur nulla eget neque ornare ultrices. Phasellus arcu turpis, euismod vitae dapibus ut, vestibulum nec urna. Morbi commodo ultricies nibh in auctor.

Appendix - D

Custom style guide, custom.css written for Grav

```
/*Header task bar on top stationary*/
#header {
    background-color: rgba(0, 98, 57, 0.9);
}
#header.scrolled {
    background-color: rgba(255, 255, 255, 0.9) !important;
}
/* asaculs logo color*/
#logo h3, #logo a, #navbar span {
    color: rgb(255, 255, 255) !important;
}
/*menu text colors*/
#navbar a {
    color: rgb(255, 255, 255) !important;
}
/*menu line colors on mouse hover*/
#navbar a:before, #navbar a:after {
    background-color: rgb(255, 255, 255) !important;
}
/*Active drop down navigation item*/
#header #navbar ul.navigation li ul li: hover>a {
    background-color: #006239;
}
/* roll down color for asaculs logo*/
#header.scrolled #logo a, #header.scrolled #navbar span {
    color: #006239 !important;
}
/*menu text color on normal*/
#header.scrolled #navbar a {
    color: #006239 !important;
}
/* menu text color on mouse hover*/
#header.scrolled #navbar a: hover {
    color: #006239 !important;
}
/*top two lines color */
#header.scrolled #navbar a:before, #header.scrolled #navbar a:after {
    background-color: #006239 !important;
}
/* Buttons within asaculs*/
.button {
    background: #fff;
    color: #23865C;
```

```

    border: 1px solid #23865C;
    border-radius: 3px;
}
.button:hover {
    background: #23865C;
    color: #fff;
}
.button:active {
    box-shadow: 0 1px 0 #23865C;
}
textarea,
input[type="email"],
input[type="number"],
input[type="password"],
input[type="search"],
input[type="tel"],
input[type="text"],
input[type="url"],
input[type="color"],
input[type="date"],
input[type="datetime"],
input[type="datetime-local"],
input[type="month"],
input[type="time"],
input[type="week"],
select[multiple=multiple]
{
    background-color: white;
    border: 1px solid #006239;
    box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.06);
}
textarea,
input[type="email"]:focus,
input[type="number"]:focus,
input[type="password"]:focus,
input[type="search"]:focus,
input[type="tel"]:focus,
input[type="text"]:focus,
input[type="url"]:focus,
input[type="color"]:focus,
input[type="date"]:focus,
input[type="datetime"]:focus,
input[type="datetime-local"]:focus,
input[type="month"]:focus,
input[type="time"]:focus,
input[type="week"]:focus,
select[multiple=multiple]
{ border-color: #23865C; }

```

Appendix - E

CSS theme sub creation for wordpress site

```
/*
Theme Name: ASACULS-WP
Theme URI:
Description: A Child theme of Twenty Sixteen theme developed for African Students
Association in CULS
Author: Wossenyeleh Merid Mekonnen
Author URI:
Template: twentysixteen
Version: 1.0.0
License: GNU General Public License v2 or later
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Tags: dark green, light, two-columns, responsive-layout, accessibility-ready
Text Domain: twenty-sixteen-asaculs
*/

.entry-title {
    display: none;
}

.menu-toggle {
    border: 1px solid #006239;
    color: #006239;
}

.main-navigation li:hover>a, .main-navigation li.focus>a {
    color: #006239;
}

.menu-toggle.toggled-on, .menu-toggle.toggled-on:hover, .menu-toggle.toggled-on:focus {
    background-color: #006239;
    border-color: #006239;
    color: #fff;
}

.menu-toggle:hover, .menu-toggle:focus {
    border-color: #006239;
    color: #006239;
}

blockquote {
    border: 0 solid #23865C;
    border-left-width: 4px;
    color: #23865C;
}
```

```

    }

    .widget {
    border-top: 4px solid #006239;
    }

    button,
    button[disabled]:hover,
    button[disabled]:focus,
    input[type="button"],
    input[type="button"][disabled]:hover,
    input[type="button"][disabled]:focus,
    input[type="reset"],
    input[type="reset"][disabled]:hover,
    input[type="reset"][disabled]:focus,
    input[type="submit"],
    input[type="submit"][disabled]:hover,
    input[type="submit"][disabled]:focus {
        background: #fff;
        border: 1px solid #23865C;
        border-radius: 3px;
        color:#23865C;
        font-family: Montserrat, "Helvetica Neue", sans-serif;
        font-weight: 700;
        letter-spacing: 0.046875em;
        line-height: 1;
        padding: 0.84375em 0.875em 0.78125em;
        text-transform: uppercase;
    }

    button:hover,
    button:focus,
    input[type="button"]:hover,
    input[type="button"]:focus,
    input[type="reset"]:hover,
    input[type="reset"]:focus,
    input[type="submit"]:hover,
    input[type="submit"]:focus {
        background: #006239;
        color: #fff;
    }

    input[type="date"],
    input[type="time"],
    input[type="datetime-local"],
    input[type="week"],
    input[type="month"],
    input[type="text"],

```

```

input[type="email"],
input[type="url"],
input[type="password"],
input[type="search"],
input[type="tel"],
input[type="number"],
textarea {
    background: #f7f7f7;
    background-image: -webkit-linear-gradient(rgba(255, 255, 255, 0), rgba(255, 255, 255, 0));
    border: 1px solid #006239;
    border-radius: 2px;
    color: #686868;
    padding: 0.625em 0.4375em;
    width: 100%;
}

```

```

input[type="date"]:focus,
input[type="time"]:focus,
input[type="datetime-local"]:focus,
input[type="week"]:focus,
input[type="month"]:focus,
input[type="text"]:focus,
input[type="email"]:focus,
input[type="url"]:focus,
input[type="password"]:focus,
input[type="search"]:focus,
input[type="tel"]:focus,
input[type="number"]:focus,
textarea:focus {
    background-color: #fff;
    border-color: #23865C;
    color: #1a1a1a;
    outline: 0;
}

```

Appendix - F

Security token generation and placement from Loader.io for WordPress

The screenshot shows the Loader.io interface for target verification. At the top, there is a navigation bar with the Loader logo, "Tests", "Target hosts", and "Help" buttons, and a user profile "getwossen@gmail.com". The main heading is "Target Verification: 46.101.194.11". Below this, there are two tabs: "Verify over HTTP" (selected) and "Verify over DNS". A green success message states: "Congrats, target verification passed! Now you can create a test!". Step 1 instructs the user to "Place this verification token in a file:" and provides a text input field. Below this, it says "Or download the file you need." Step 2 instructs the user to "Upload the file to your server so it is accessible at one of the following URLs:" and lists three URLs: "http://46.101.194.11/loaderio-", "http://46.101.194.11/loaderio-", and "http://46.101.194.11/loaderio-". At the bottom, there are two buttons: "Back to targets" and "New Test".

Security Token generation and placement from loader.io for Grav

The screenshot shows the Loader.io interface for target verification. At the top, there is a navigation bar with the Loader logo, "Tests", "Target hosts", and "Help" buttons, and a user profile "getwossen@gmail.com". The main heading is "Target Verification: 138.68.71.127". Below this, there are two tabs: "Verify over HTTP" (selected) and "Verify over DNS". A green success message states: "Congrats, target verification passed! Now you can create a test!". Step 1 instructs the user to "Place this verification token in a file:" and provides a text input field. Below this, it says "Or download the file you need." Step 2 instructs the user to "Upload the file to your server so it is accessible at one of the following URLs:" and lists three URLs: "http://138.68.71.127/loaderio-bd", "http://138.68.71.127/loaderio-bd", and "http://138.68.71.127/loaderio-bd". At the bottom, there are two buttons: "Back to targets" and "New Test".

Appendix - G

Questionnaire for Grav

Questionnaire regarding African Students Association in CULS Part 1 -GRAV

This questionnaire is aimed at students studying in Czech University of Life Sciences

Thank you for taking your time to take this questionnaire.

Go to this web address (138.68.71.127). you can copy paste the numbers on the browser. Answer the questions that follow. For copy paste purposes the address is written below.

138.68.71.127

1. I can clearly see the menu

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

2. I can see the content (text and letters) on the website

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

3. I can see the title of the page

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

4. I know where I am on the website

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

5. Changing pages is fast and easy

- I strongly agree
 - I agree
 - I am indifferent
 - Disagree
 - Strongly disagree
-

6. the website loads fast

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

7. I can easily find what I want in this website

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

8. I don't need to scroll left and right to see the contents

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

9. The design of the website is attractive

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

10. The content of this web is organized.

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

Thank you for taking your time to fill this survey.

Questionnaire for WordPress

Questionnaire regarding African Students Association in CULS Part 2 -WordPress

This questionnaire is aimed at students studying in Czech University of Life Sciences

Thank you for taking your time to take this questionnaire.

Go to this web address (46.101.194.11), you can copy paste the numbers on the browser. Answer the questions that follow. For copy paste purposes the address is written below.

138.68.71.127

1. I can clearly see the menu

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

2. I can see the content (text and letters) on the website

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

3. I can see the title of the page

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

4. I know where I am on the website

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

5. Changing pages is fast and easy

- I strongly agree
 - I agree
 - I am indifferent
 - Disagree
 - Strongly disagree
-

6. the website loads fast

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

7. I can easily find what I want in this website

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

8. I don't need to scroll left and right to see the contents

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

9. The design of the website is attractive

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

10. The content of this web is organized.

- I strongly agree
- I agree
- I am indifferent
- Disagree
- Strongly disagree

Thank you for taking your time to fill this survey.