Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Quim Farrés Masana

Titulació: Grau en Tècniques d'Interacció Digital i de Computació

Títol de Treball Final de Grau: Anàlisi comparativa de PHP i Django per a Desenvolupament Web

Director/a: **Jesus Ojeda Contreras i Jordi Mateo Fornés**

Presentació

Mes: Setembre

Any: 2023

Abstract

This thesis presents a comparative analysis between PHP and Django, two popular frameworks for website development. The aim of the study is to evaluate the strengths and weaknesses of each framework, and provide insights for choosing the most suitable framework for different web development scenarios.

Contents

1 Introduction **3**
- 1.1 Background . . . 3
- 1.2 Objectives . . . 3
- 1.3 Significance . . . 4

2 Literature Review **5**

3 Methodology **7**
- 3.1 Research Design . . . 7
- 3.2 Data Sources . . . 7
- 3.3 Data Extraction . . . 8
- 3.4 Experimental Setup . . . 8
- 3.5 Performance Metrics . . . 8
- 3.6 Load Testing . . . 8
- 3.7 Data Analysis . . . 8

4 Comparison Criteria **9**
- 4.1 Qualitative Criteria . . . 9
- 4.2 Quantitative Criteria . . . 10

5 Comparison Results **11**
- 5.1 Qualitative Criteria Results . . . 11
- 5.2 Quantitative Criteria Results . . . 12
  - 5.2.1 Performance Metrics . . . 12
  - 5.2.2 Observations . . . 12

6 Conclusion **14**
- 6.1 Ease of Development . . . 14
- 6.2 Quantitative Results . . . 14
- 6.3 Framework Adoption . . . 15
- 6.4 Learning and Growth . . . 15

6.5 Conclusion on Django and PHP 15
6.6 Final Thoughts 16

A Code Samples **17**

B Graphs **24**

C References **52**

Chapter 1

Introduction

1.1 Background

In today's digital era, websites have become an essential part of businesses, organizations, and individuals. The efficient and effective development of websites is crucial to meet the growing demands of the online world. To aid in this process, several frameworks have been developed, offering a wide range of features and functionalities. Among these frameworks, PHP and Django have gained significant attention for their capabilities in website development.

1.2 Objectives

The objectives of this thesis are as follows:

1. Compare and analyze the Django framework's features and functionalities with those of php framework in terms of each one's capacity to create and manage high-quality web applications.

2. Analyze the usability and learning curve of both frameworks, examining their ease of use for developers.

3. Examine both frameworks' scalability and capacity for handling large amounts of traffic and data, assessing their performance under high loads.

4. Analyze both frameworks' security features and their capacity to fend off flaws and online threats.

5. Evaluate the integration capabilities of both frameworks with other technologies such as databases and third-party tools.

6. Conduct a comparative analysis of the documentation and online support communities of both frameworks to evaluate their accessibility and quality.

7. Evaluate the customization capabilities of both frameworks and their flexibility to adapt to different business and design requirements.

8. Conduct a comparative analysis of the costs associated with using and maintaining both frameworks.

9. Recommend the most suitable framework for different types of web development projects based on the results of the comparative evaluation.

These objectives provide a clear focus for the thesis and outline the key aspects that will be investigated during the comparative analysis. Each objective will be addressed in the relevant sections of the thesis, supported by evidence and analysis.

1.3 Significance

Understanding the differences and trade-offs between PHP and Django can greatly benefit developers, businesses, and organizations involved in web development. By exploring their features, performance, ease of use, community support, and other relevant aspects, we can identify the areas in which each framework excels and where it may have limitations. This knowledge can guide developers in selecting the appropriate framework, leading to more efficient and successful website development projects.

Chapter 2

Literature Review

This literature review aims to provide insights into the features, advantages, and disadvantages of PHP and Python Django for web development. Understanding the characteristics of each language is crucial for a comprehensive comparative analysis.

PHP, with its syntax resembling C and Perl, offers familiarity to developers from those backgrounds. It supports constructs like the 'switch' statement and 'do...while' loops, making it a flexible language. However, PHP lacks proper namespace support and often relies on prefixes for function names. It provides a casual environment where globals are sometimes used for passing arguments.

On the other hand, Python emphasizes simplicity, clear syntax, and object-oriented programming. It offers namespaces, modules, and a small core, making code more organized and readable. Python provides strong support for keyword arguments, default arguments, and true object-oriented programming. It also offers extensive libraries and frameworks for web development, with Django being a prominent choice.

Documentation plays a significant role in language selection, and both PHP and Python have comprehensive documentation resources. PHP has an extensive community base, contributing to its popularity and providing a wealth of online support. Python's community is also vast, and its web development capabilities are further enhanced by various third-party modules and frameworks.

In summary, PHP offers familiarity and flexibility, while Python emphasizes simplicity, clear syntax, and object-oriented programming. PHP's web development features are built into the core language, while Python's web development capabilities are provided by add-on modules and frameworks like Django, offering more flexibility and a broader ecosystem.

The subsequent chapters will delve into a detailed comparative analysis of these languages and frameworks, examining their usability, scalability, security, integration capabilities, documentation, customization, costs, and suitability for different web development projects.

Chapter 3

Methodology

In this chapter, we outline the methodology employed to conduct a comparative analysis between the PHP and Django frameworks. The research design involved a combination of qualitative and quantitative analysis, with an emphasis on experimental approaches. The objective was to comprehensively evaluate both frameworks, despite having limited prior experience in the field.

3.1 Research Design

The research design encompassed both qualitative and quantitative analysis. The qualitative aspect aimed to explore subjective aspects of the frameworks, such as ease of use and developer-friendliness. It involved an experimental design where the researcher, having limited prior knowledge of the field, sought to compare PHP and Django objectively.

3.2 Data Sources

The primary sources of data for this study included: - **Official Documentation**: Extensive official documentation from PHP and Django was referenced to understand the features, functionalities, and best practices associated with each framework. - **Tutorials**: Tutorials were consulted, particularly for aspects related to load testing, to gain insights and practical knowledge.

3.3 Data Extraction

To evaluate the performance of both frameworks, the Apache JMeter tool was employed. JMeter, an Apache project, is a versatile load testing tool capable of analyzing and measuring the performance of various services. It was used for load testing both the PHP and Django-based websites.

3.4 Experimental Setup

Two websites with identical features were developed, one using Django and the other using PHP. Both websites included login and registration forms, as well as a feed where users could create posts, comment on them, and like posts and comments.

3.5 Performance Metrics

Several performance metrics were considered to evaluate the frameworks: - **Hits Per Second**: This metric measures the number of HTTP requests sent to the web server in one second. - **Response Time**: It calculates the total time taken from a user's request until they receive a response from the server. - **Percentile Over Time**: This statistical measure indicates the value below which a given percentage of observations fall in a group.

3.6 Load Testing

Load testing was conducted using different thread groups for both PHP and Django-based websites. Various combinations of users and thread counts were tested, including users: 5, 10, 20, 50, and thread counts: 2, 5, 20, to assess the performance of the frameworks under different loads.

3.7 Data Analysis

Quantitative data collected during load testing were analyzed using statistical methods. The results were interpreted to draw comparisons between PHP and Django in terms of their performance and scalability.

Chapter 4

Comparison Criteria

In this section, we outline the criteria used to evaluate and compare the PHP and Django frameworks, divided into qualitative and quantitative categories.

4.1 Qualitative Criteria

1. **Documentation and Support**:
   - **Quality of Documentation**: Evaluate comprehensiveness and clarity of official documentation for PHP and Django.
   - **Community Support**: Assess size and responsiveness of the developer community and availability of online resources (e.g., forums, blogs, tutorials).
2. **Customization and Flexibility**:
   - **Flexibility**: Determine adaptability to different business and design requirements.
   - **Customization**: Evaluate the ability to customize and extend the framework's core functionality.
3. **Usability and Learning Curve**:
   - **Ease of Learning**: Assess the learning curve for developers.
   - **Developer-Friendliness**: Consider code readability, ease of debugging, and available development tools.
4. **Community and Ecosystem**:

- **Community Size**: Assess the size and activity of the developer community.
- **Ecosystem**: Consider the availability of third-party packages, extensions, and plugins.

4.2 Quantitative Criteria

1. **Performance**:
    - **Response Time**: Measure page load and response time.
    - **Throughput**: Assess the number of requests handled per unit of time (e.g., requests per second).
    - **Scalability**: Evaluate how well each framework handles increased load and traffic.
2. **Integration Capabilities**:
    - **Database Integration**: Assess integration with various database systems.
    - **Third-Party Integration**: Evaluate ease of integrating external libraries, APIs, and services.
3. **Portability**:
    - Assess the ease of deploying applications built with each framework to different hosting environments and platforms.
4. **Maturity and Stability**:
    - Consider the maturity and stability of each framework in terms of long-term support and backward compatibility.

These criteria provide a comprehensive framework for evaluating both qualitative and quantitative aspects of PHP and Django, forming the basis for the comparative analysis in this study.

Chapter 5

Comparison Results

In this section, we present the results of our comparative analysis of the PHP and Django frameworks. It's important to note that our assessment may carry a bias stemming from our prior experience with other programming languages. However, we found it valuable to share our observations and insights gained during this evaluation.

5.1 Qualitative Criteria Results

In our qualitative assessment, we considered several aspects of both frameworks, including documentation, ease of programming, and depth of functionality.

Documentation Both PHP and Django boast extensive documentation resources. PHP's documentation, being older, offers a vast repository of knowledge. However, it's worth noting that some information may be dated due to the framework's longevity. Django, on the other hand, offers comprehensive and up-to-date documentation.

Ease of Programming During our evaluation, we found that Django allowed us to accomplish tasks more efficiently. With fewer lines of code, we achieved similar results compared to PHP. Importantly, Django abstracts many database complexities, making it accessible to developers with basic database knowledge. PHP, while powerful, seemed slightly more challenging, possibly due to our familiarity bias.

Depth of Functionality We observed that PHP provides greater flexibility for developers well-versed in the language. It allows for deeper customization, but this may come at the cost of increased development time.

5.2 Quantitative Criteria Results

In this section, we present the quantitative results of our performance and response time tests for both the Django and PHP frameworks. We evaluated the performance based on the following parameters:

5.2.1 Performance Metrics

Thread Count: Thread count represents the count of virtual users that we expect to connect to the server.

Loop Count: Loop count allows us to set the number of iterations for every user during testing.

Hits Per Second (HPS): Hits Per Second represents the number of requests sent to the server in one second.

Response Time: Response time is the total time it takes from when a user makes a request until they receive a response.

Percentiles: A percentile is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations fall.

5.2.2 Observations

Here are the conclusions drawn from our performance and response time tests:

Hits Per Second (HPS)

In scenarios with low thread and loop counts, both Django and PHP exhibit similar HPS values. However, as we increase the number of threads and loops, notable differences emerge. PHP demonstrates lower response times, but it is susceptible to peaks, which may impact the stability of the system. In contrast, Django maintains more stable performance but tends to have higher response times in high-load situations.

Response Time

Similar to the HPS results, at lower thread and loop counts, Django and PHP show comparable response times. However, as the number of users and requests increases, distinctions become evident. PHP tends to exhibit higher response times compared to Django. Additionally, PHP displays a greater number of response time peaks, indicating potential variability in user experience.

Percentile Over Time

The trend observed in percentiles mirrors the previous metrics. In scenarios with a low number of requests and simultaneous users, both frameworks perform similarly. However, as the demands on the system increase, Django demonstrates superior performance.

Chapter 6

Conclusion

In this final chapter, we draw upon the research conducted in this thesis to arrive at comprehensive conclusions regarding the comparison of the Django and PHP frameworks for web development.

We also have created the two websites in php and django.
Django
http://quimfarres.pythonanywhere.com/

PHP
https://mysocialwebsitetfg.000webhostapp.com/Social/

6.1 Ease of Development

One of the primary takeaways from this study is the ease of development. Personally, I found it easier to develop websites using Django. The framework's clean and structured design, along with its emphasis on "batteries-included," allowed for faster development and reduced coding effort. Moreover, Django abstracts many complexities, such as database interactions, which made the development process smoother.

6.2 Quantitative Results

Quantitatively, our analysis revealed that the performance differences between PHP and Django, while present, were not significantly pronounced. In scenarios with low thread and loop counts, both frameworks exhibited similar behavior in terms of Hits Per Second (HPS), response time, and percentiles. However, as we scaled up the demands on the systems, nuanced differences emerged. PHP demonstrated lower response times but with noticeable peaks, potentially impacting stability. In contrast, Django maintained more stable performance but sometimes with slightly higher response times.

6.3 Framework Adoption

An interesting observation was the prominence of Django in newer web projects. Being a relatively newer framework, it enjoys increased adoption for contemporary web development endeavors. This reflects the industry's recognition of Django's strengths and its alignment with modern development practices.

6.4 Learning and Growth

This research journey offered valuable insights and learning experiences. Exploring load testing and delving into web development using different languages enriched our understanding of the web development landscape. We not only expanded our knowledge of Django and PHP but also gained proficiency in essential skills such as load testing, which can benefit future projects. Additionally, this research provided an opportunity to enhance our skills in LaTeX, the typesetting system used to create this thesis.

6.5 Conclusion on Django and PHP

In conclusion, the choice between Django and PHP for web development should consider factors such as personal preferences, project requirements, and the specific needs of the development team. While Django emerged as the preferred choice for its ease of use and alignment with modern practices, PHP remains a powerful tool with its own strengths, particularly for developers well-versed in the language.

This research provides a deeper understanding of the capabilities and performance nuances of both frameworks. However, it's important to note that the final decision should be context-dependent. Developers should consider factors beyond performance, such as community support, project scale, and available resources.

6.6 Final Thoughts

As we wrap up this thesis, we reflect on the substantial growth in our knowledge and skills. We now possess a more profound understanding of the complexities of web development and the unique attributes of Django and PHP. This journey has not only broadened our technical expertise but also deepened our awareness for how the world of web development is always changing.

In closing, we extend our gratitude to all who supported us in this endeavor, and we look forward to applying the insights gained from this research in future web development projects.

Appendix A

Code Samples

In this appendix, you will find a selection of code samples that demonstrate key aspects of the web development work conducted during this research. These samples showcase the practical application of both Django and PHP frameworks in building web applications. They serve as illustrative examples of coding techniques and practices discussed in the thesis.

```html
<ul>
    {% for post in posts %}
    <li class="post-box">
        <div class="post-content">
            <strong>{{ post.user }}</strong><br>
            {{ post.text }}<br>
            <small>{{ post.created_at }}</small>
            <form action="{% url 'like_post' post.id %}" method="POST">
                {% csrf_token %}
                <button type="submit">
                    {% if request.user in post.liked_by.all %}Unlike{% else %}Like{% endif %}
                </button>
            </form>
            <small>{{ post.liked_by.all.count }} like(s)</small>
        </div>
        <div>
            <div class="comment-box">
                {% for comment in post.comments.all %}
                <div class="comment-content">
                    <strong>{{ comment.user }}</strong><br>
                    {{ comment.text }}<br>
                    <small>{{ comment.created_at }}</small>
                    <form action="{% url 'like_comment' comment.id %}" method="POST">
                        {% csrf_token %}
                        <button type="submit">
                            {% if request.user in comment.liked_by.all %}Unlike{% else %}Like{% endif %}
                        </button>
                    </form>
                    <small>{{ comment.liked_by.all.count }} like(s)</small>
                </div>
                {% endfor %}
            </div>
        </div>
        <form action="{% url 'add_comment' post.id %}" method="POST">
            {% csrf_token %}
            <textarea name="text" placeholder="Your comment..." required></textarea>
            <button type="submit">Comment</button>
        </form>
    </li>
    {% endfor %}
</ul>
```

Figure A.1: Home page django

```python
@login_required(login_url='login')
def like_post(request, post_id):
    post = Post.objects.get(pk=post_id)
    if request.user in post.liked_by.all():
        post.liked_by.remove(request.user)
    else:
        post.liked_by.add(request.user)
    return redirect('index')
```

Figure A.2: Like post django

```css
/* Body styles */
body {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  background-color: #f6f8fa;
  color: #14171a;
}

.container {
  width: 100%;
  max-width: 600px;
  padding: 20px;
  display: flex;
  flex-direction: column;
  align-items: center;
}

/* Header styles */
header {
  background-color: #ffffff;
  padding: 20px;
  text-align: center;
  width: 100%;
}

.header-container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.logo a {
  text-decoration: none;
  color: #1da1f2;
  font-weight: bold;
}

.logout-button a {
  text-decoration: none;
  color: #1da1f2;
  font-weight: bold;
}
```

Figure A.3: Some css django

```python
def signup(request):
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        password2 = request.POST['password2']

        if password == password2:
            if User.objects.filter(email=email).exists():
                messages.info(request, 'Email Taken')
                return redirect('signup')
            elif User.objects.filter(username=username).exists():
                messages.info(request, 'Username Taken')
                return redirect('signup')
            else:
                user = User.objects.create_user(username=username, email=email, password=password)
                user.save()

                return redirect('signin')
        else:
            messages.info(request, 'Password Not Matching')
            return redirect('signup')
    else:
        return render(request, 'signup.html')
```

Figure A.4: Log in django

```php
    // Validate credentials
    if(empty($username_err) && empty($password_err)){
        // Prepare a select statement
        $sql = "SELECT id, username, password FROM users WHERE username = ?";

        if($stmt = mysqli_prepare($link, $sql)){
            // Bind variables to the prepared statement as parameters
            mysqli_stmt_bind_param($stmt, "s", $param_username);

            // Set parameters
            $param_username = $username;

            // Attempt to execute the prepared statement
            if(mysqli_stmt_execute($stmt)){
                // Store result
                mysqli_stmt_store_result($stmt);

                // Check if username exists, if yes then verify password
                if(mysqli_stmt_num_rows($stmt) == 1){
                    // Bind result variables
                    mysqli_stmt_bind_result($stmt, $id, $username, $hashed_password);
                    if(mysqli_stmt_fetch($stmt)){
                        if(password_verify($password, $hashed_password)){
                            // Password is correct, so start a new session
                            session_start();

                            // Store data in session variables
                            $_SESSION["loggedin"] = true;
                            $_SESSION["id"] = $id;
                            $_SESSION["Username"] = $username;

                            // Redirect user to welcome page
                            header("location: ../");
                        } else{
                            // Password is not valid, display a generic error message
                            $login_err = "Invalid username or password.";
                        }
                    }
                } else{
                    // Username doesn't exist, display a generic error message
                    $login_err = "Invalid username or password.";
                }
            } else{
                echo "Oops! Something went wrong. Please try again later.";
            }
```

Figure A.5: Validate credentials php

```php
                foreach ($posts as $post) {
                    $postId = $post['id'];
                    $userName = $post['user_name'];
                    $postText = $post['post_text'];
                    $postDate = $post['post_date'];
                    $formattedDate = date('F j, Y, g:i a', strtotime($postDate));
                    $likesCount = $post['likes'];

                    echo "<div class='tweet'>";
                    echo "<p class='timestamp'>$formattedDate</p>";
                    echo "<p class='author'><strong>$userName</strong></p>";
                    echo "<p class='content'>$postText</p>";
                    echo "<div class='actions'>";
                    echo "<form action='gestio/like_post.php' method='post'>";
                    echo "<input type='hidden' name='post_id' value='$postId'>";
                    echo "<button type='submit' class='like-button'>Like</button>";
                    echo "</form>";
                    echo "<p class='likes-count'>$likesCount Likes</p>";
                    echo "</div>";

                    // Retrieve comments for the current post
                    $stmt = $pdo->prepare("SELECT * FROM comments WHERE post_id = :post_id ORDER BY comment_date ASC");
                    $stmt->bindParam(":post_id", $postId, PDO::PARAM_INT);
                    $stmt->execute();
                    $comments = $stmt->fetchAll(PDO::FETCH_ASSOC);

                    // Display the comments
                    echo "<div class='comments'>";
                    foreach ($comments as $comment) {
                        $commentId = $comment['id'];
                        $commentUserName = $comment['user_name'];
                        $commentText = $comment['comment_text'];
                        $commentLikesCount = $comment['likes'];
                        $commentDate = $comment['comment_date'];
                        $formattedCommentDate = date('F j, Y, g:i a', strtotime($commentDate));

                        echo "<div class='comment'>";
                        echo "<p class='comment-author'>$commentUserName</p>";
                        echo "<p class='comment-content'>$commentText</p>";
                        echo "<p class='comment-date'>$formattedCommentDate</p>";
                        echo "<div class='comment-actions'>";
                        echo "<form action='gestio/like_comment.php' method='post'>";
                        echo "<input type='hidden' name='comment_id' value='$commentId'>";
                        echo "<button type='submit' class='like-button'>Like</button>";
```

Figure A.6: Show tweets php

```php
// Validate password
$password = (isset($_POST['password'])) ? trim($_POST['password']) : '';
if (empty($password)) {
    $password_err = "Please enter a password.";
} elseif(strlen(trim($_POST["password"])) < 6){
    $password_err = "Password must have atleast 6 characters.";
} else{
    $password = trim($_POST["password"]);
}

// Validate confirm password
$confirm_password = (isset($_POST['confirm_password'])) ? trim($_POST['confirm_password']) : '';
if (empty($confirm_password)) {
    $confirm_password_err = "Please confirm password.";
} else{
    $confirm_password = trim($_POST["confirm_password"]);
    if(empty($password_err) && ($password != $confirm_password)){
        $confirm_password_err = "Password did not match.";
    }
}
```

Figure A.7: validate password php

Appendix B

Graphs

Here are the following graphs, first hits per second next resposne over time and finaly response time precentiles for each combination of Loops and Threads and for each framework.

A summary for the different aspects of the graphs.

Thread count is the count of virtual users that we are expecting to connect to the server.
Loop count is we can set the number of iterations for every user.
Hits Per Second represents the number of requests sent to the server in one second.
Response time is the total time it takes from when a user makes a request until they receive a response.
Percentile is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations fall.

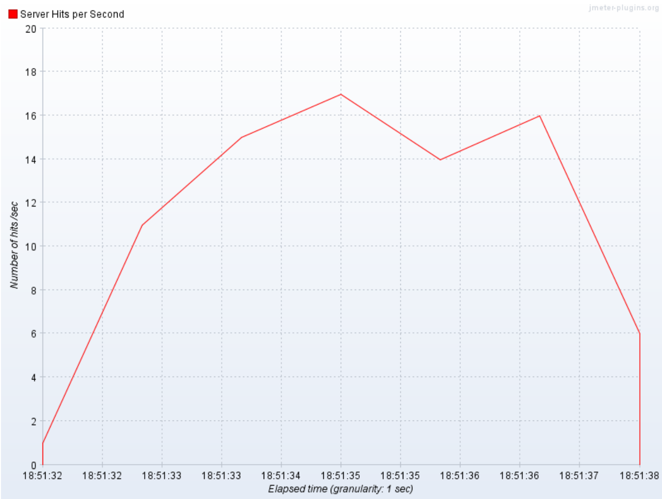

Figure B.1: 10 Threads 2 Loops django

Figure B.2: 10 Threads 2 Loops PHP

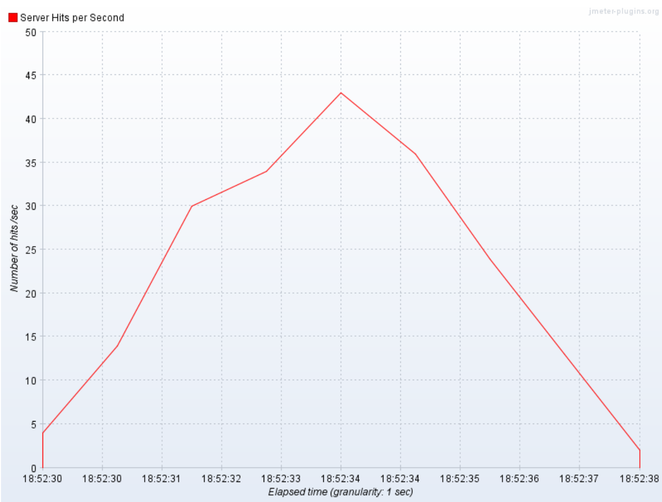

Figure B.3: 10 Threads 5 Loops django

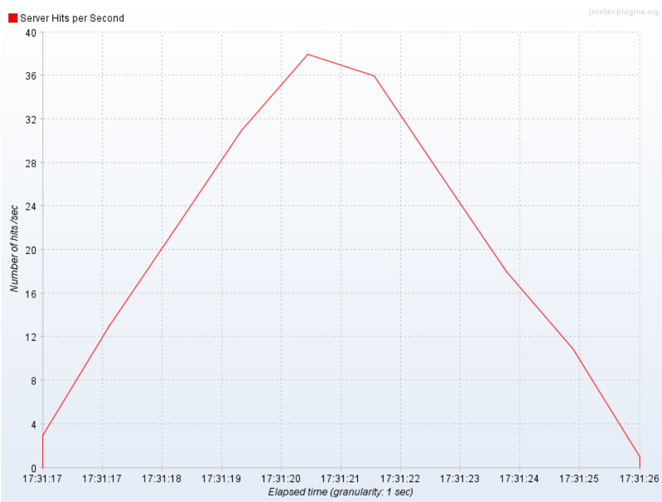

Figure B.4: 10 Threads 5 Loops PHP

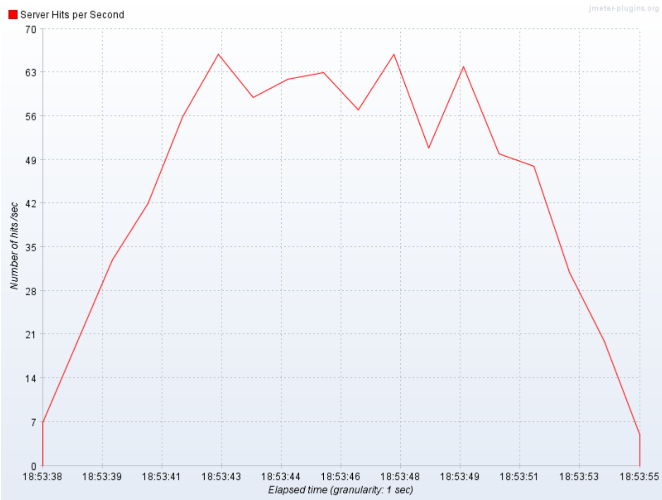

Figure B.5: 10 Threads 20 Loops django

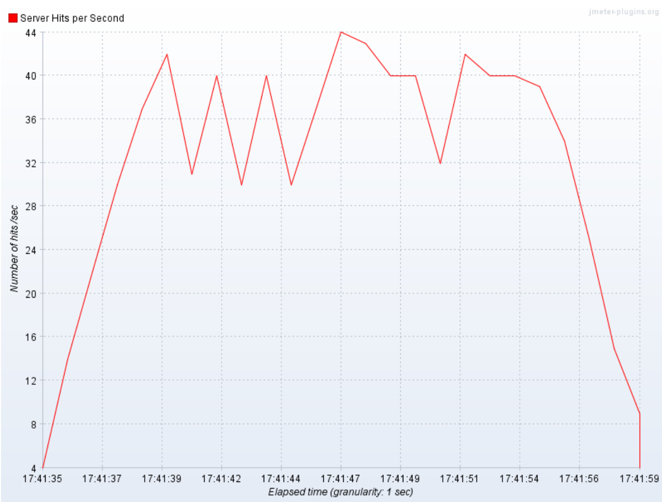

Figure B.6: 10 Threads 20 Loops PHP

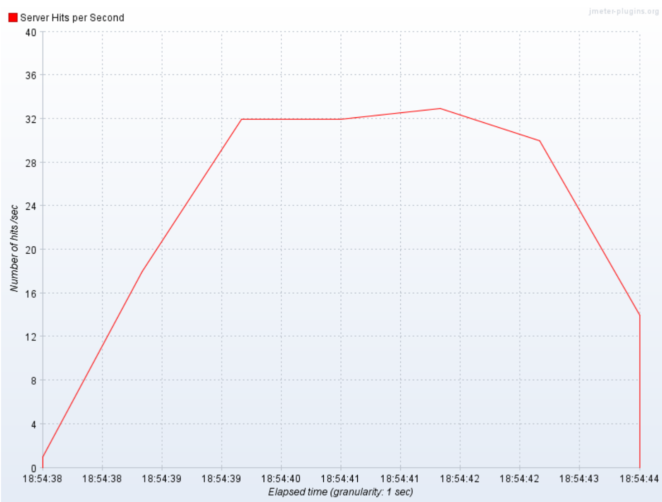

Figure B.7: 20 Threads 2 Loops django

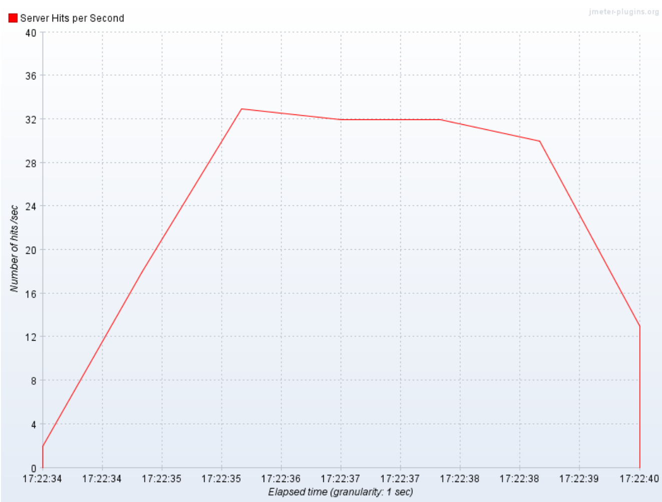

Figure B.8: 20 Threads 2 Loops PHP

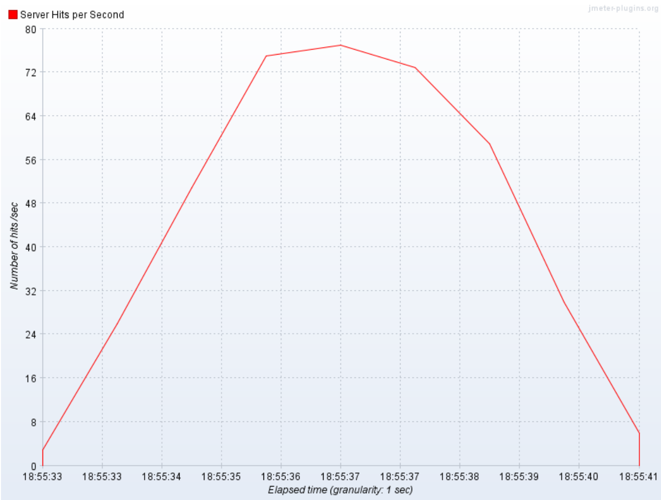

Figure B.9: 20 Threads 5 Loops django

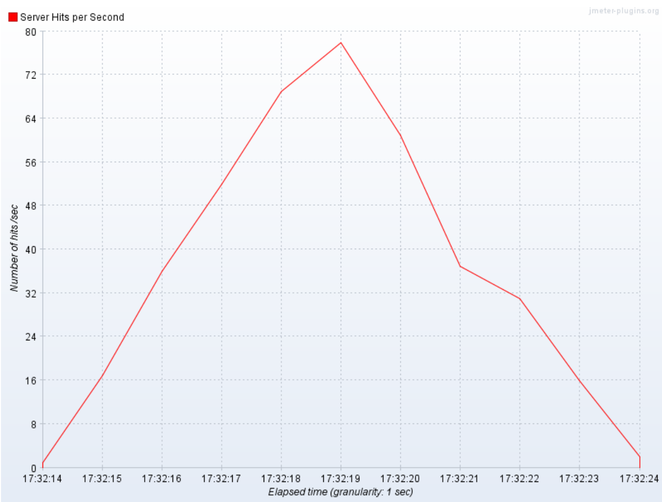

Figure B.10: 20 Threads 5 Loops PHP

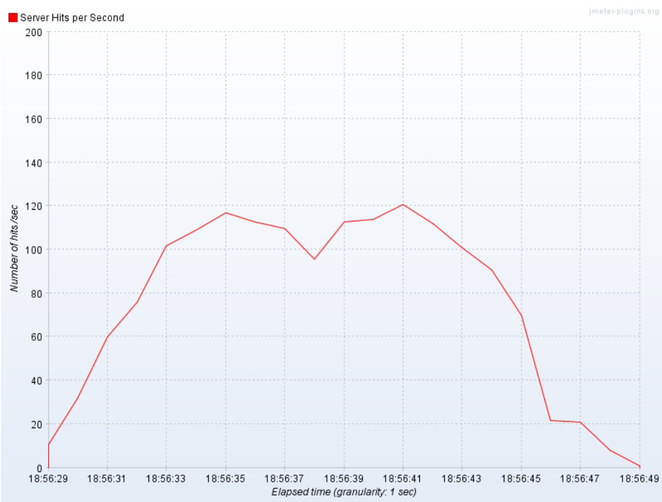

Figure B.11: 20 Threads 20 Loops django

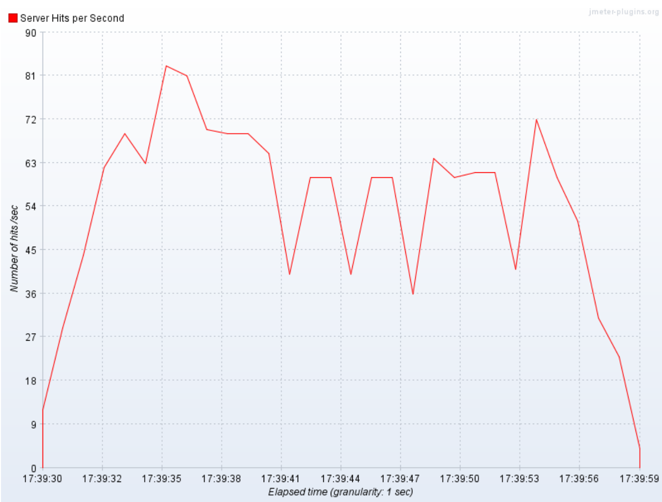

Figure B.12: 20 Threads 20 Loops PHP

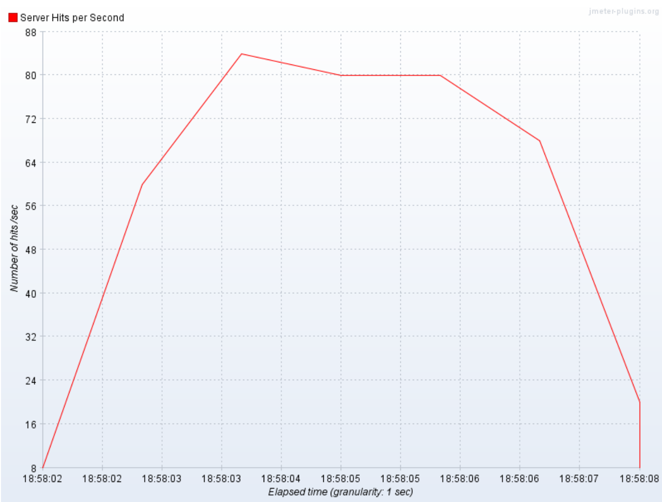

Figure B.13: 50 Threads 2 Loops django

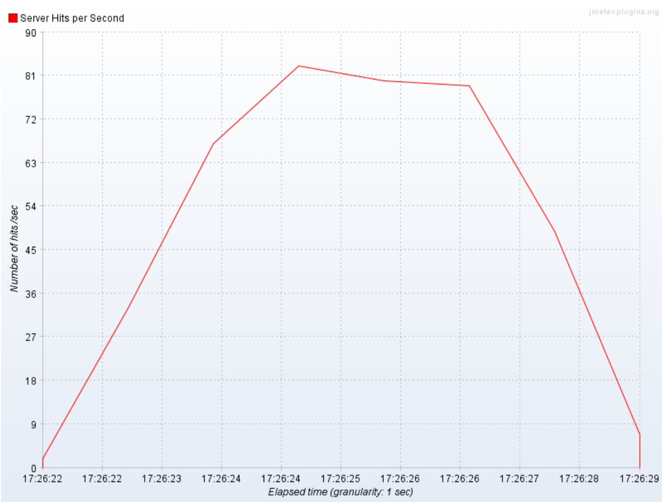

Figure B.14: 50 Threads 2 Loops PHP

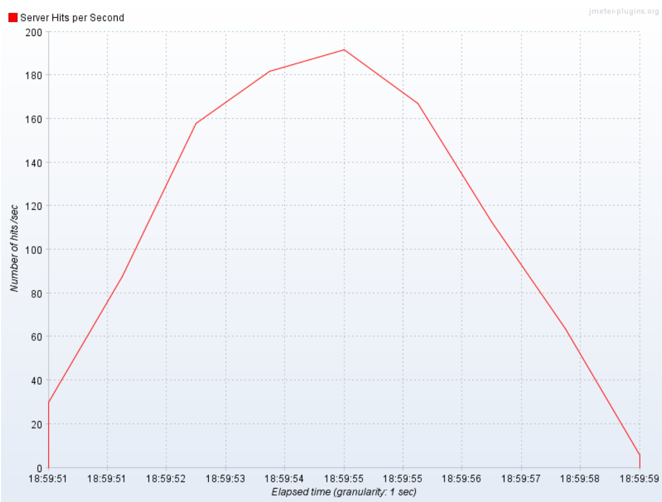

Figure B.15: 50 Threads 5 Loops django

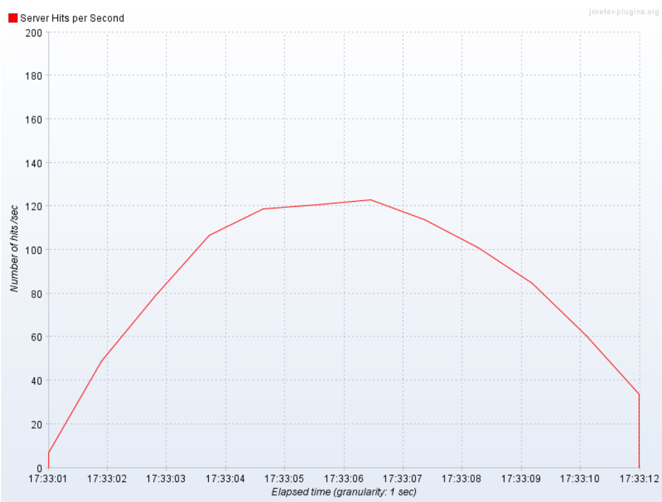

Figure B.16: 50 Threads 5 Loops PHP

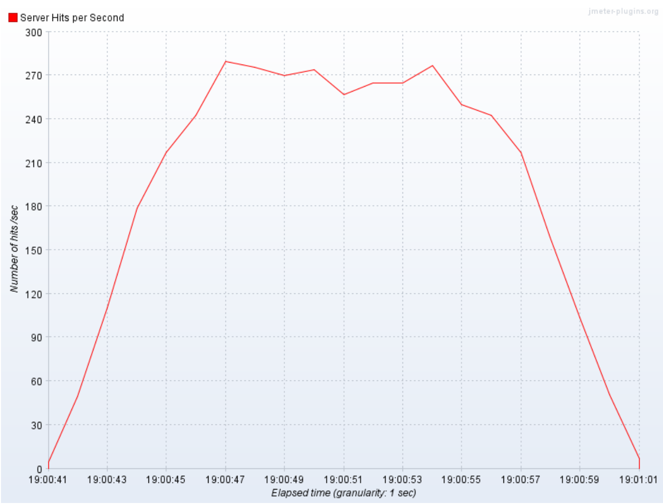

Figure B.17: 50 Threads 20 Loops django

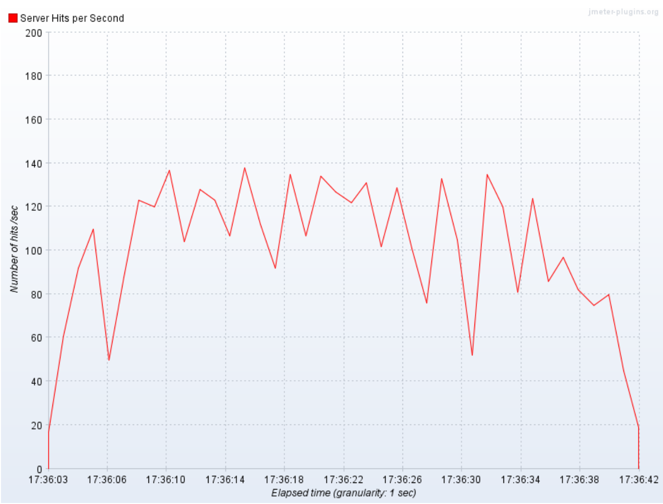

Figure B.18: 50 Threads 20 Loops PHP

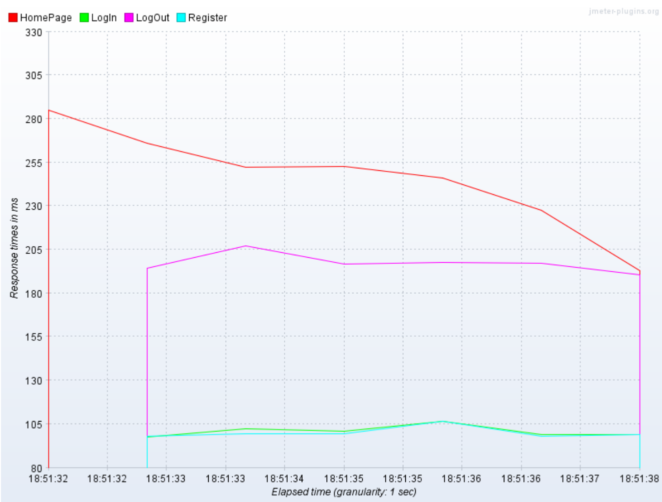

Figure B.19: 10 Threads 2 Loops django

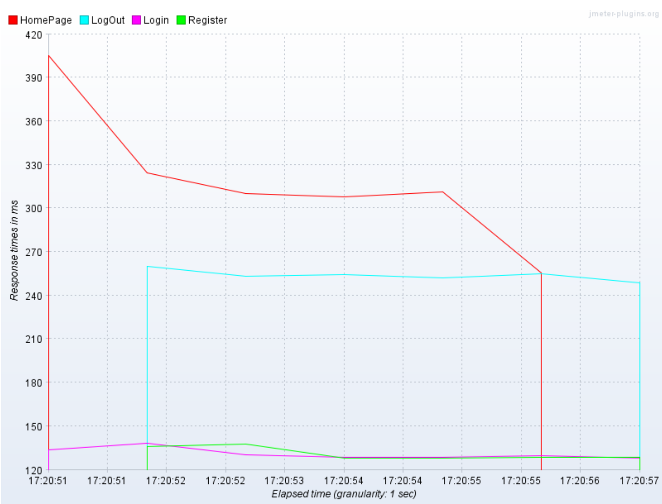

Figure B.20: 10 Threads 2 Loops PHP

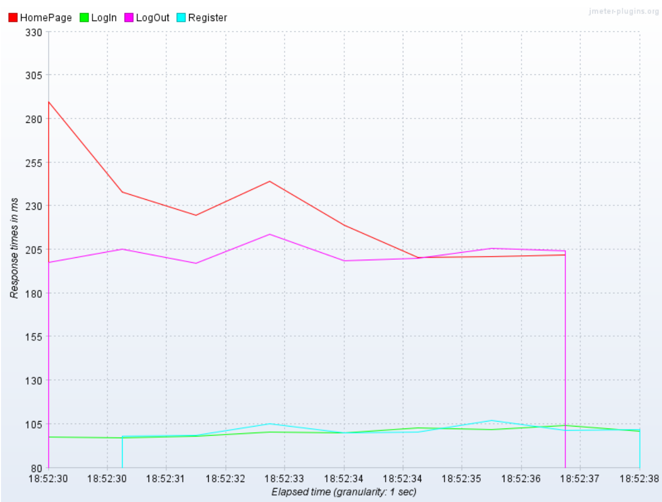

Figure B.21: 10 Threads 5 Loops django

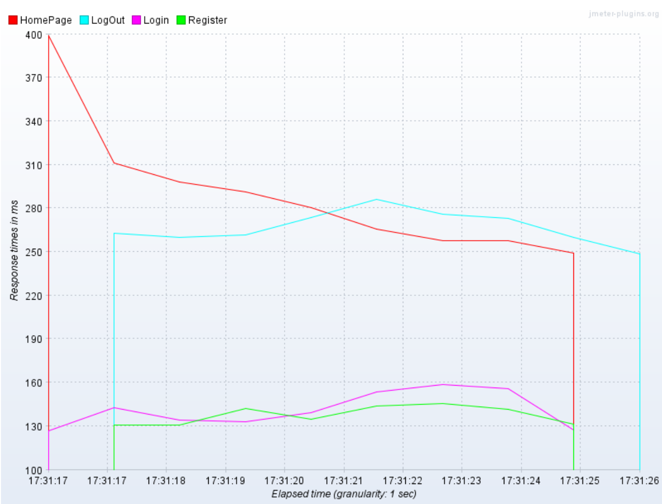

Figure B.22: 10 Threads 5 Loops PHP

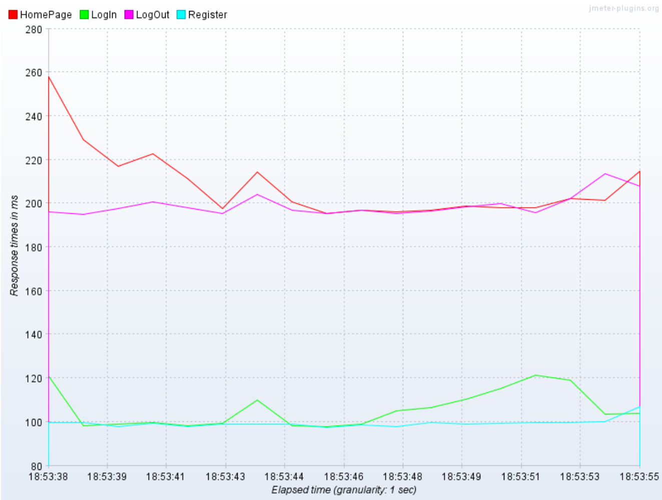

Figure B.23: 10 Threads 20 Loops django

Figure B.24: 10 Threads 20 Loops PHP

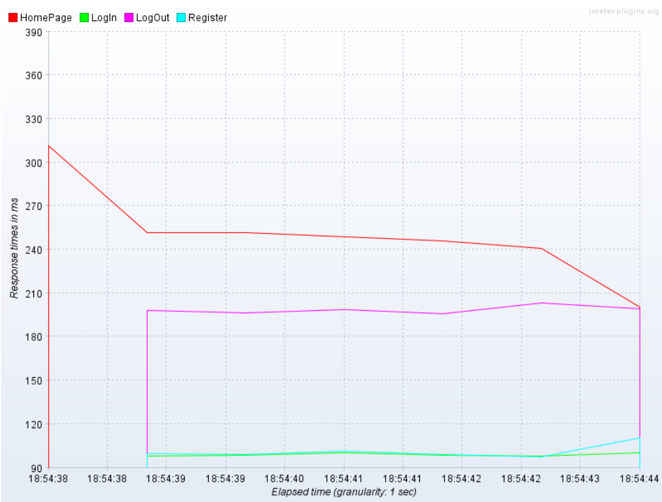

Figure B.25: 20 Threads 2 Loops django

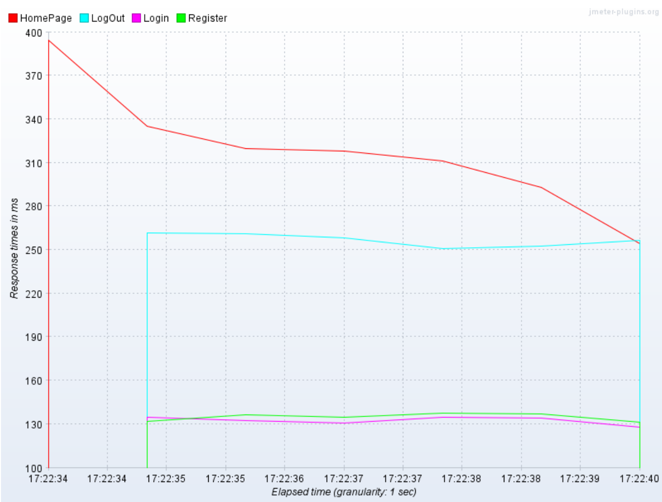

Figure B.26: 20 Threads 2 Loops PHP

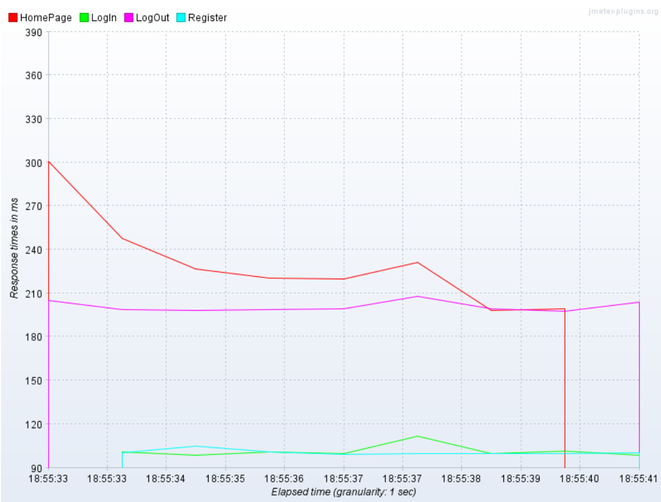

Figure B.27: 20 Threads 5 Loops django

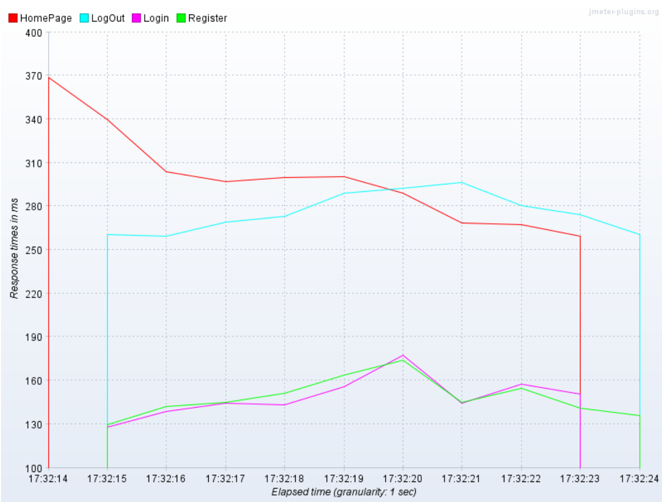

Figure B.28: 20 Threads 5 Loops PHP

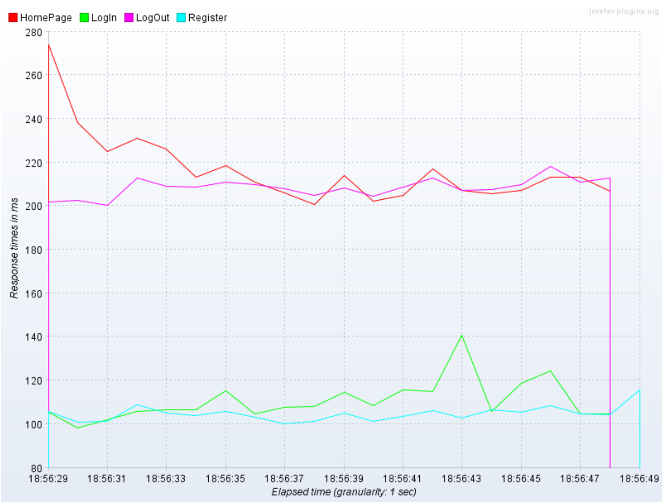

Figure B.29: 20 Threads 20 Loops django

Figure B.30: 20 Threads 20 Loops PHP

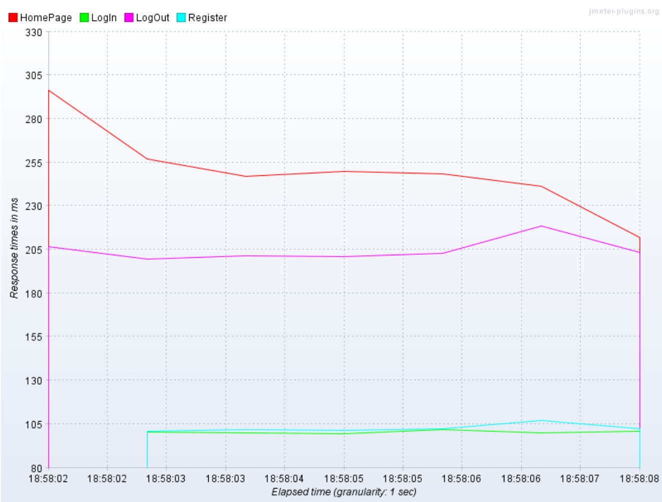

Figure B.31: 50 Threads 2 Loops django

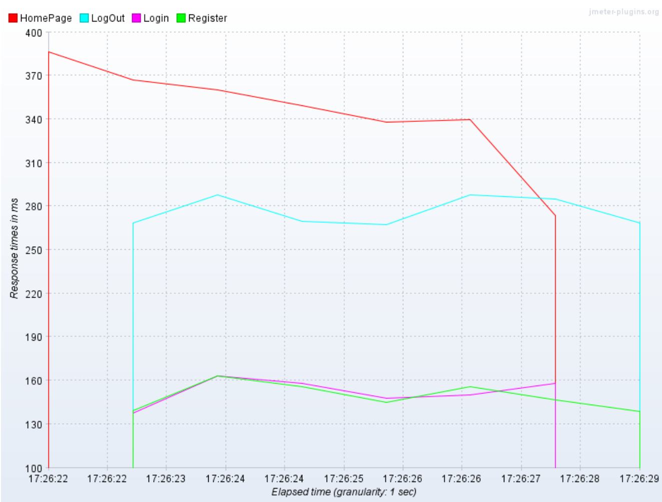

Figure B.32: 50 Threads 2 Loops PHP

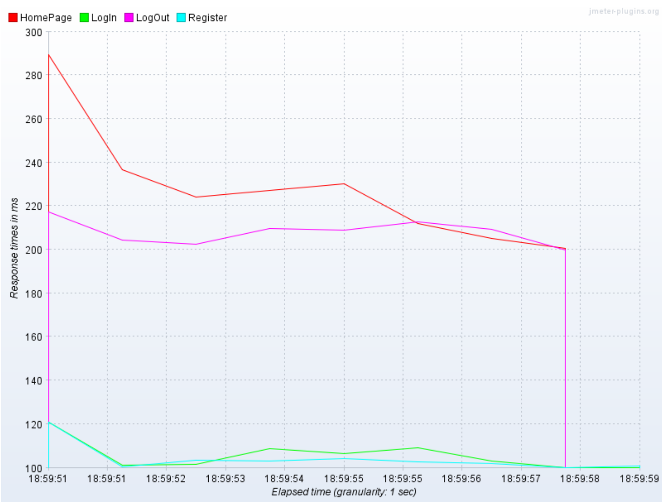

Figure B.33: 50 Threads 5 Loops django

Figure B.34: 50 Threads 5 Loops PHP

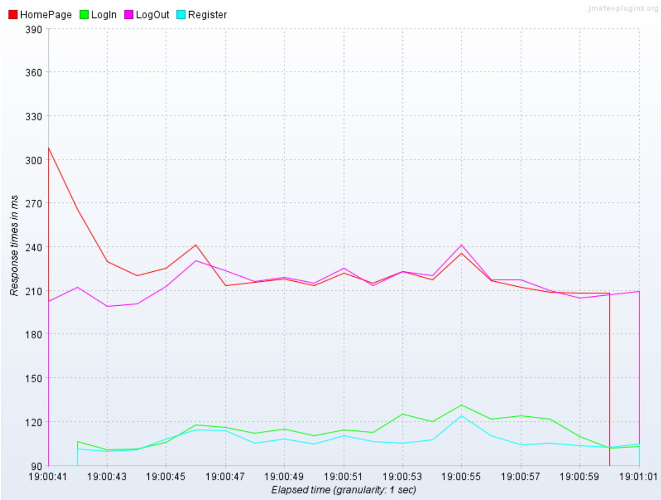

Figure B.35: 50 Threads 20 Loops django

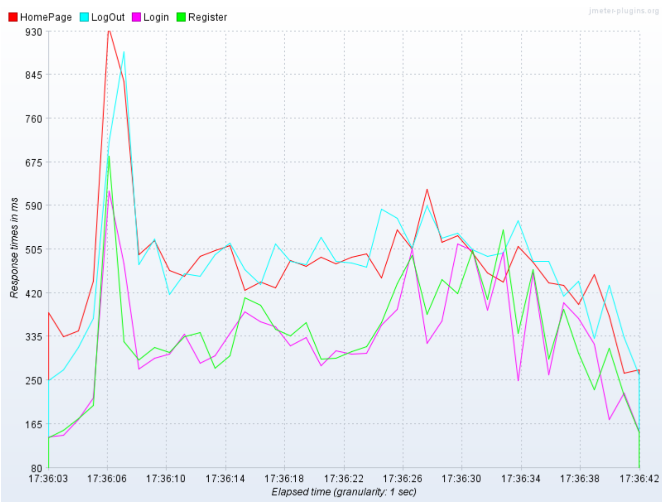

Figure B.36: 50 Threads 20 Loops PHP

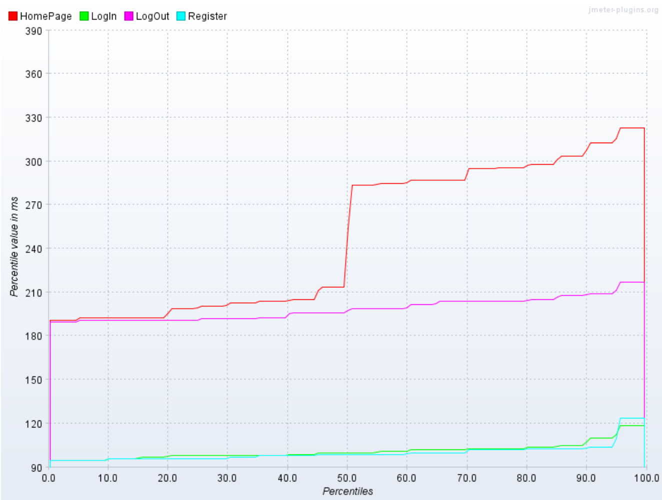

Figure B.37: 10 Threads 2 Loops django

Figure B.38: 10 Threads 2 Loops PHP

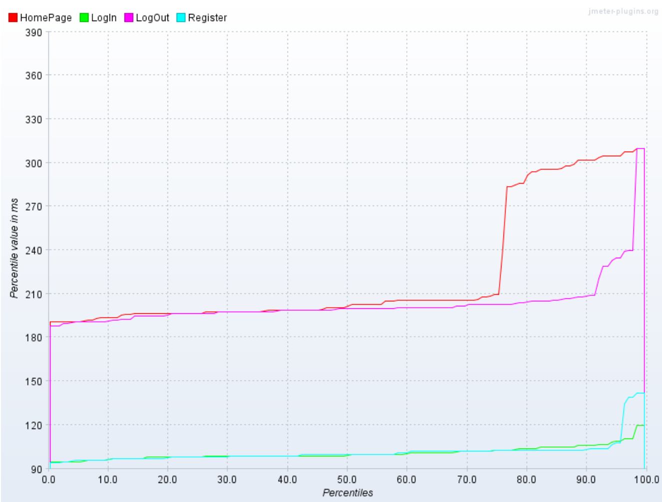

Figure B.39: 10 Threads 5 Loops django

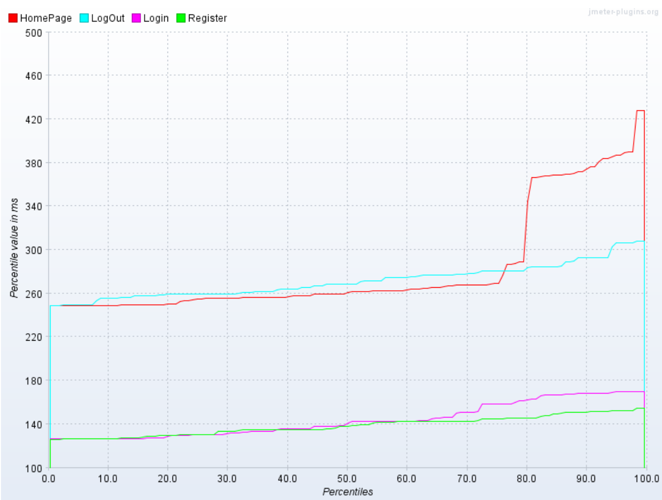

Figure B.40: 10 Threads 5 Loops PHP

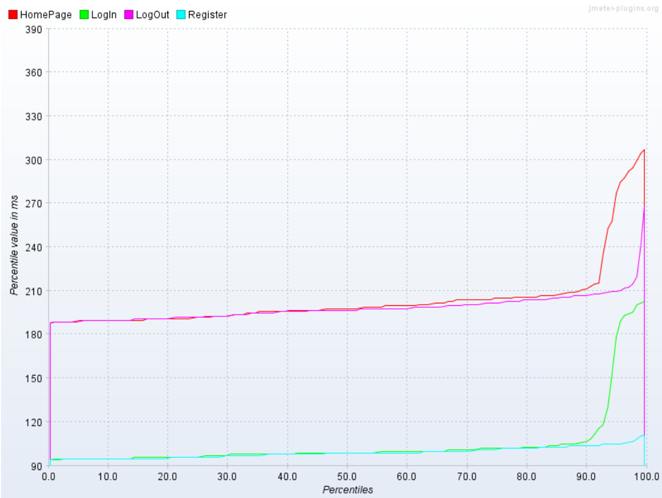

Figure B.41: 10 Threads 20 Loops django

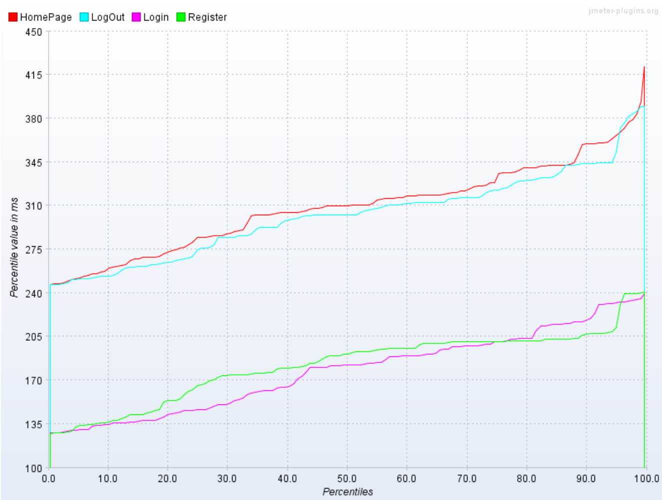

Figure B.42: 10 Threads 20 Loops PHP

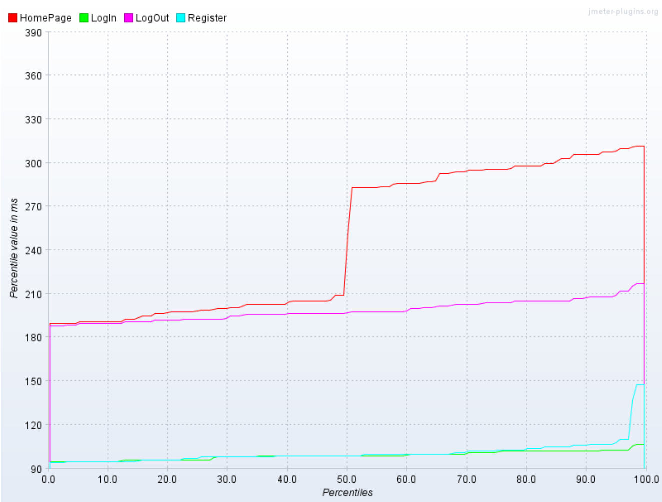

Figure B.43: 20 Threads 2 Loops django

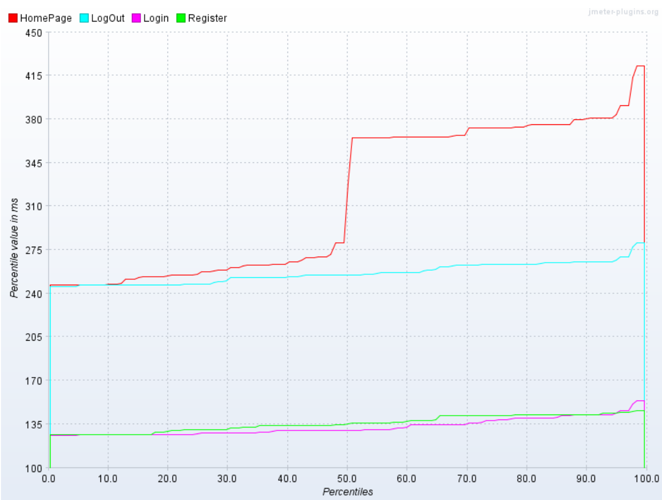

Figure B.44: 20 Threads 2 Loops PHP

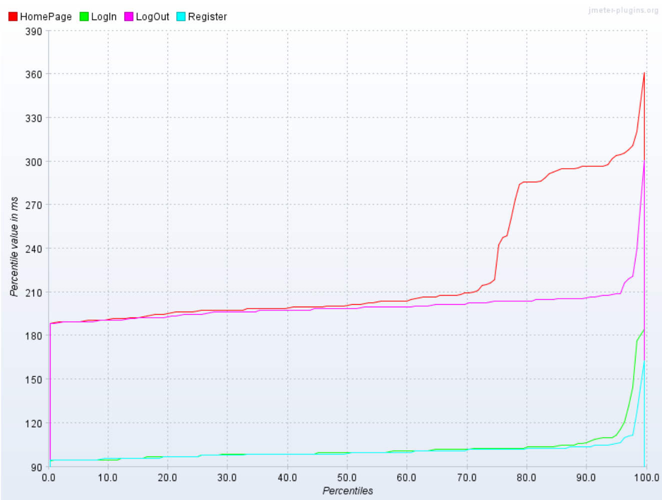

Figure B.45: 20 Threads 5 Loops django

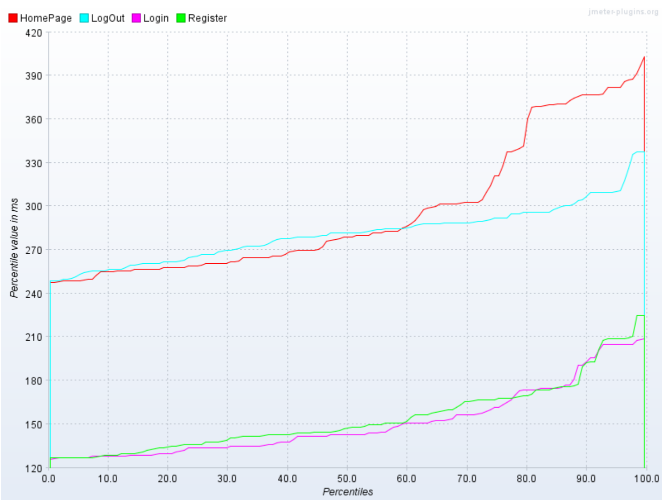

Figure B.46: 20 Threads 5 Loops PHP

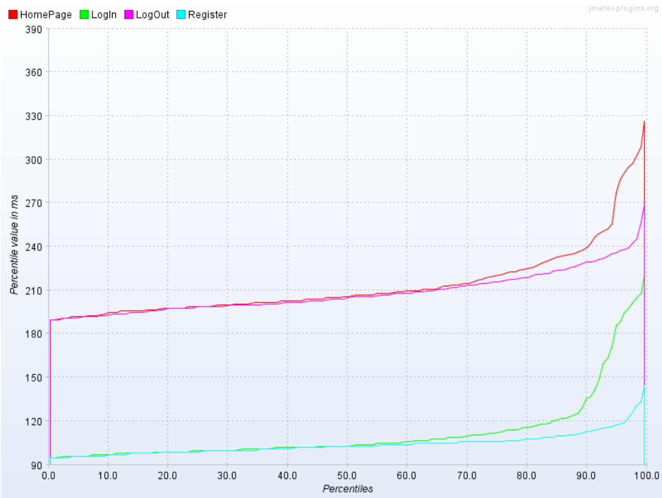

Figure B.47: 20 Threads 20 Loops django

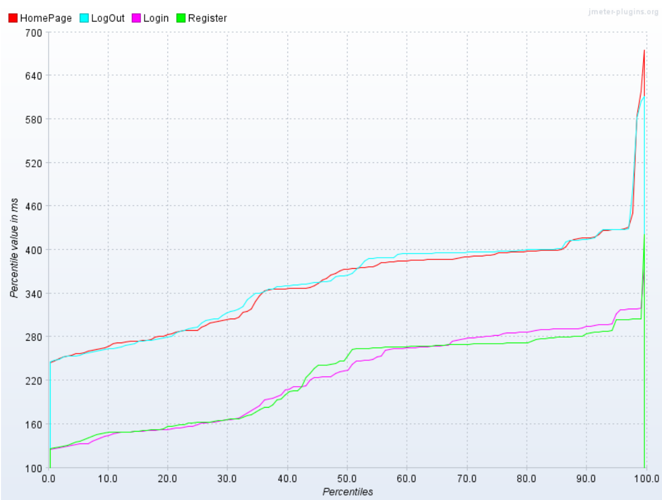

Figure B.48: 20 Threads 20 Loops PHP

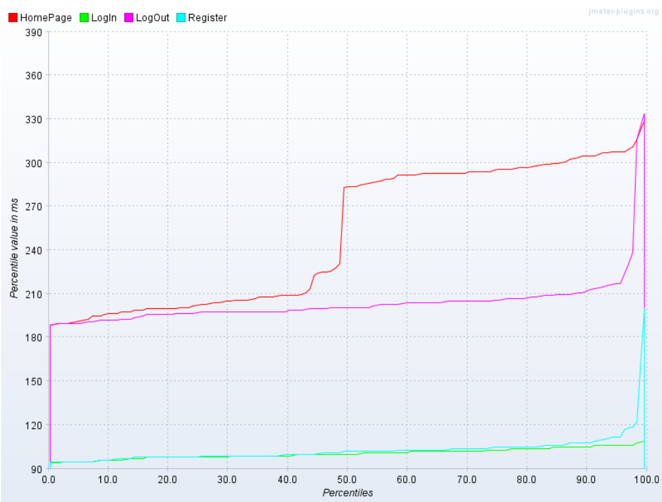

Figure B.49: 50 Threads 2 Loops django

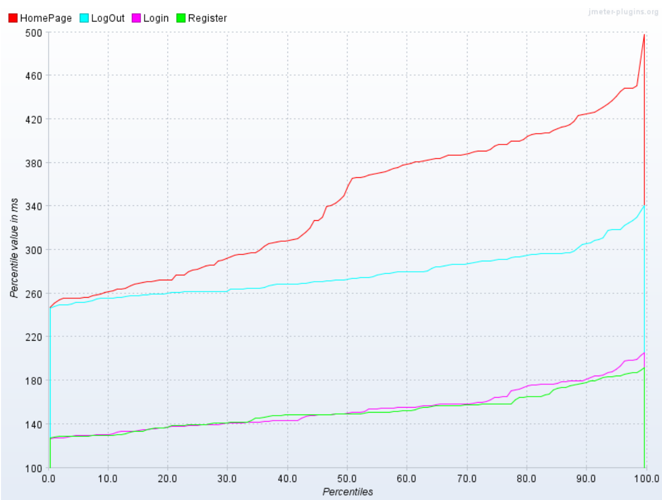

Figure B.50: 50 Threads 2 Loops PHP

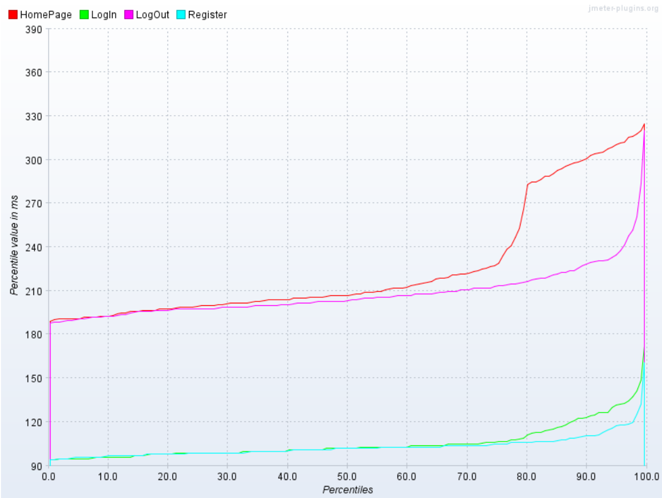

Figure B.51: 50 Threads 5 Loops django

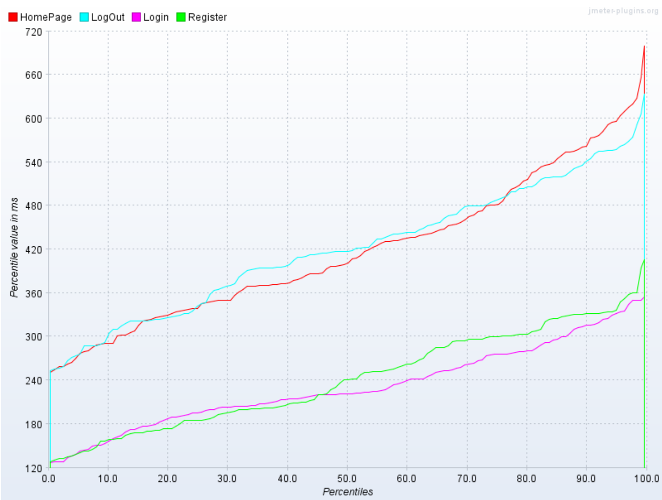

Figure B.52: 50 Threads 5 Loops PHP

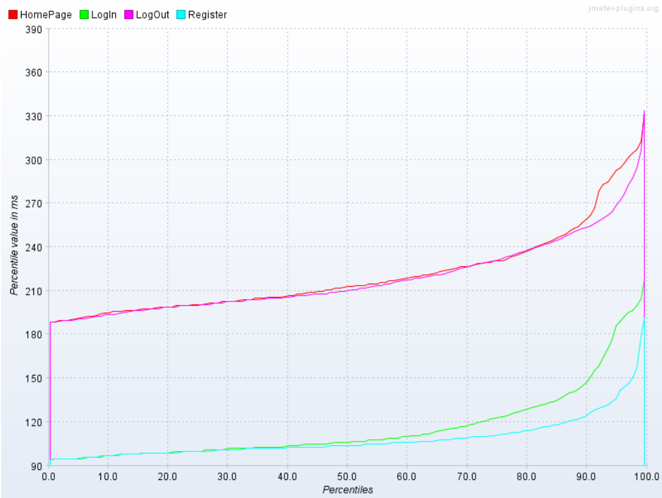

Figure B.53: 50 Threads 20 Loops django

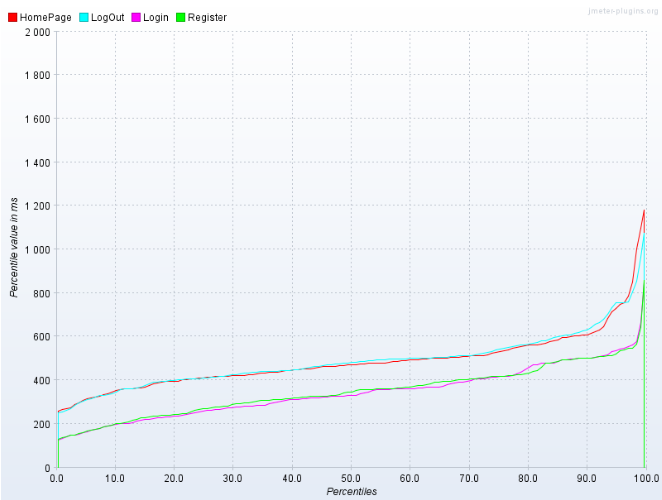

Figure B.54: 50 Threads 20 Loops PHP

Appendix C

References

1. **Official Django Documentation**
https://docs.djangoproject.com/
The official documentation for Django, offering comprehensive information about the framework's features and usage.

2. **Official PHP Documentation**
https://www.php.net/docs.php
The official documentation for PHP, providing detailed guidance on PHP programming.

3. **Stack Overflow**
https://stackoverflow.com/
A popular community-driven QA platform where you can find answers to various programming questions related to Django, PHP, and web development.

4. **GitHub - Django Repository**
https://github.com/django/django
Access the Django source code, report issues, and explore the framework's development on GitHub.

5. **GitHub - PHP Repository**
https://github.com/php/php-src
The GitHub repository for PHP, offering access to PHP's source code and development discussions.

6. **Web Development for Beginners (Tutorials)**
https://developer.mozilla.org/en-US/docs/Learn
Mozilla's Developer Network provides tutorials and guides for web development beginners, including sections on Django and PHP.

7. **Load Testing with Apache JMeter**
https://jmeter.apache.org/usermanual/build-web-test-plan.html
Apache JMeter's official documentation on building web test plans, a valuable

resource for load testing web applications.

8. **LaTeX Project**

https://www.latex-project.org/

The official website of the LaTeX project, providing resources and information about LaTeX typesetting.

9. **Web Development Blogs**

- CSS-Tricks: https://css-tricks.com/ - Smashing Magazine: https://www.smashingmagazine.com/ - SitePoint: https://www.sitepoint.com/

Explore these blogs for articles, tutorials, and insights on various aspects of web development.

10. **Overleaf - Online LaTeX Editor**

https://www.overleaf.com/

Overleaf is a popular online LaTeX editor that simplifies the process of creating and collaborating on LaTeX documents.