DOI: 10.26565/2304-6201-2020-45-07

УДК 004.415.53

Тестирование производительности программного обеспечения

О.М. Мелкозьорова[1], С.Г. Рассомахін[1]
[1]*Харківський національний університет імені В. Н. Каразіна, майдан Свободи 4, м. Харків, 61022, Україна*
olja.mex@gmail.com

В статье приведен обзор существующих на данный момент типов тестирования производительности. Рассмотрен тест плана с использованием инструмента Jmeter, показана возможность составления тестов и их анализа для получения результата при обеспечении соответствующего качества и удобства использования. Тестирование программного обеспечения – один из наиболее важных этапов при разработке программных продуктов. Универсальный подход к разработке тестов усложняется большими объемами данных, наличием разнообразных методик и инструментов. Особую значимость имеет тестирование вычислительной производительности приложений. По сравнению с другими видами тестирования, то именно этот вид тестирования крайне сложен и требует определенных навыков персонала. Производительность в основном испытывается автоматизированными методами, однако при параметрическом исследовании программ важным становится статистическое тестирование.

Ключевые слова: *тестирование производительности, автоматизация, инструмент, Jmeter, нагрузка.*

У статті наведено огляд існуючих типів тестування продуктивності. Розглянуто тест план з використовуванням інструменту Jmeter, показана можливість складання тестів та їх аналізу для отримання результату при забезпеченні відповідної якості та зручності використання. Тестування програмного забезпечення – один з найбільш важливих етапів при розробці програмних продуктів. Універсальний підхід до розробки тестів ускладнюється великим об’ємом даних, наявністю різноманітних методик та інструментів. Особливу важливість має тестування обчислювальної продуктивності додатків. У порівнянні з іншими типами тестування, саме цей вид вкрай складний та потребує певних навичок персоналу. Продуктивність в основному випробовується автоматизованими методами, однак при параметричному дослідженні програм стає важним статистичне тестування.

Ключові слова: *тестування продуктивності, автоматизація, інструмент, Jmeter, навантаження.*

The article provides an overview of existing performance testing types. A test plan using the Jmeter tool is considered, the ability to compile tests and analyze them to obtain results is provided, while ensuring appropriate quality and ease of use. Software testing is one of the most important steps in software development. The universal approach to test development is complicated by the sheer data volume, the availability of various techniques and tools. Of particular importance is testing the computational performance of applications. Compared to other types of testing, this type is extremely complex and requires some personnel skills. Performance is mostly tested by automated methods, but statistical testing becomes important in parametric program research.

Keywords*: performance testing, automation, tool, Jmeter, load.*

1 Принципы тестирования производительности

Тестирование производительности играет важную роль для достижения необходимого уровня качества программного продукта, а также обеспечения удобства его использования (Usability) потенциальным пользователем. Характеристики функциональной пригодности и удобства использования тестируемых приложений оцениваются при условиях интенсивной нагрузки и длительного использования. Эффективность испытуемой системы по показателю производительности оценивается в соответствии с международным стандартом ISO 25010 [1].

Общепринятыми являются три основных показателя, характеризующих производительность программного продукта [2]:

Оценка времени отклика. Это самая распространенная задача при тестировании производительности. Этот аспект тестирования определяет способность компонентов или системы отвечать пользователю или системе за определенное время при определенных условиях.

Использование ресурсов. Доступность ресурсов (например, распределение оперативной памяти) может быть исследовано при проведении определенных тестов по производительности.

Мощность. Если поведение системы при требуемой мощности ограничивает систему (например, количество пользователей или объем данных), тесты по производительности могут быть необходимы, чтобы оценить пригодность архитектуры системы.

Тестирование производительности происходит экспериментально, при этом должны быть доступны измерения и анализ специфических системных параметров. Тестирование может быть проведено итерационно при системном анализе, дизайне и имплементации архитектурного решения.

При тестировании производительности используются следующие принципы [2]:

тесты должны отвечать ожиданиям пользователей, дизайнеров и персонала;

тесты должны быть воспроизводимыми, то есть повторение тестов на неизменной системе должно приводить к статистически идентичным результатам;

результаты по тестированию производительности должны быть понятны и легко сопоставимы с ожиданием пользователя;

тесты могут быть проведены с использованием доступных ресурсов – на полной или частичной системе;

тесты должны быть доступны и исполняемы в отведенное время.

2 Типы тестирования производительности

Для тестирования производительности могут быть определены различные типы тестов, которые зависят от целей тестирования [2,3,4].

Термин «тестирование производительности» (Performance Testing) включает в себя любое тестирование, которое сфокусировано на производительности (ответной реакции, скоростных показателях) системы или компонентов под различными объемами и характерами нагрузки.

Нагрузочное тестирование (Load Testing) фокусируется на способности системы справляться с возрастающими уровнями ожидаемой нагрузки, возникающей в результате запросов конкурирующих пользователей или процессов. Кроме того, это исследование робастности (запаса прочности) – способности системы сохранять заданные показатели качества при допустимых пределах нагрузки и при некотором превышении этих пределов.

Стрессовое тестирование (Stress Testing) фокусируется на способности системы или компонентов поддерживать пиковые нагрузки, которые находятся на уровне или превышают пределы ожидаемых. Фактически, это исследование поведения приложения при «аномальных» изменениях нагрузки. Данный тип тестирования также используется для того, чтобы оценить способность системы поддерживать показатели, такие как доступная компьютерная мощность, доступная полоса пропускания и память.

Тестирование масштабируемости (Scalability Testing) проверяет способность системы увеличивать свою производительность. Цель этих тестов – определить способность системы расти (например, с большим количеством пользователей, с большим объемом данных) без нарушения определенных требований по производительности. Другими словами, это исследование способности приложения увеличивать показатели производительности в соответствии с увеличением количества доступных приложению ресурсов.

Объемное тестирование (Volume Testing) – исследование производительности системы при обработке различных объемов данных.

Всплесковое тестирование (Spike testing) – основывается на способности системы отвечать корректно при внезапных всплесках пиковых нагрузок и возвращении впоследствии к номинальному состоянию.

Тестирование на выносливость (Endurance Testing) фокусируется на стабильности системы. Этот тип тестирования проверяет наличие проблем с ресурсами (например, нехватка памяти, соединение с базой данных, потоками), которые влияют на производительность, или могут служить причинами сбоев в критических точках.

Тестирование надежности (Reliability Testing) – очень близкое по своему смыслу к предыдущему виду тестирования. Это проверка способности приложения выполнять свои функции в заданных условиях на протяжении заданного времени или заданного количества операций.

Конкурентное тестирование (Concurrency Testing) фокусируется на влиянии ситуации, в которой одновременно происходят множественные специфические действия (например, одновременная работа большого количества пользователей). При этом исследуется поведение системы при обработке большого количества одновременно поступающих запросов, что вызывает конкуренцию между запросами за ресурсы (базу данных, память, канал передачи данных, дисковую систему). Иногда под конкурентным тестированием понимают также исследование работы многопоточного приложения и корректность синхронизации действий.

Тестирование мощности (Capacity Testing) определяет, как много пользователей может обслуживать система.

При тестировании производительности различают статическое и динамическое тестирование.

Статическое тестирование зачастую более важно для тестирования производительности, чем тестирование функциональной пригодности. Это является следствием того, что очень много

критических дефектов вносятся в архитектуру и дизайн системы на этапе проектирования. Дефекты могут возникать из-за недопонимания или недостатка знаний дизайнеров и архитекторов. Функциональные требования, цели использования, ожидаемая нагрузка и существующие ограничения могут быть недостаточно изучены в условиях дефицита времени.

Мероприятия при тестировании производительности могут предусматривать:

- просмотр требований, которые сфокусированы на аспектах производительности и рисков;
- просмотр схем баз данных, диаграмм, процедур и запросов;
- просмотр систем и архитектуры;
- просмотр критических сегментов и системного кода (например, алгоритмов).

Динамическое тестирование должно проводится на всех уровнях тестирования, то есть во время модульного, интеграционного, системного и приемочного.

3 Использование инструмента JMeter для тестирования производительности

Apache JMeter – открытое программного обеспечение. Это приложение, которое используется Java для тестирования функционального поведения и производительности приложений [4].

Рассмотрим пример, в котором для исследования производительности была использована версия Apache JMeter (2.11.20151206) (рисунок 3.1), на виртуальной машине VirtualBox под управлением операционной системы Ubuntu. Тестированию подвергалось веб-приложение. Для тестирования был записан тест-план для трех ссылок (рисунок 3.2).

Перечень элементов, из которых возможно составление тест планов в JMeter:

- Группы потоков (Thread groups);
- Логические контролеры (Logic controller);
- Типовые контроллеры (Sample generating controller);
- Слушатели (Listeners);
- Таймеры (Timers);
- Соответствия (Assertions);
- Конфигурационные элементы (Configuration elements).

Thread groups – начальные точки любого тест плана (рисунок 3.2). Все контроллеры и образцы должны быть в группе потоков. Другие элементы, такие как слушатели, могут располагаться под тест паном, в котором они применяются для всех потоков групп. Элемент группы потоков управляет количеством потоков, который JMeter будет использовать для выполнения теста.

Listener – это компонент, который показывает результаты образцов. Результаты могут быть показаны в дереве, таблице, графах или записаны в лог-файл. К примеру, в тест-плане на рисунке 3.2 выбран Listener view result tree, при помощи которого получаются результаты в виде таблиц и графиков (рисунки 3.3, 3.4, 3.5).

При системном динамическом тестировании систему тестируют в зависимости от характера нагрузки. При этом можно выделить возможные варианты тестов:

Наращивание нагрузки (Ramp up) – позволяет определить точку насыщения (Saturation point), т.е. показатель нагрузки, при которой производительность системы начинает ухудшаться. Проведение нескольких Ramp up тестов при увеличении доступных аппаратных ресурсов позволяет определить степень масштабируемости системы (Scalability).

Низкая, средняя и высокая нагрузка (Low, Mid, High-Load) – позволяет определить время отзыва системы

Тест на выживание (Longevity Test) показывает способность системы работать под достаточно высокими нагрузками.

Тест „час пик” (Rush Hour Test) позволяет оценить реакцию системы на резкое изменение нагрузки. Рассмотрим статистику выполнения тестов на производительность для различного количества пользователей (рисунок 3.3, 3.4, 3.5). При увеличении количества пользователей было бы неплохим результатом линейное увеличение время отклика. В таблицах 3.1 и 3.2 приведены реальное и ожидаемое поведение системы. По этим данным построен график зависимости времени отклика от количества пользователей (рисунок 3.6).

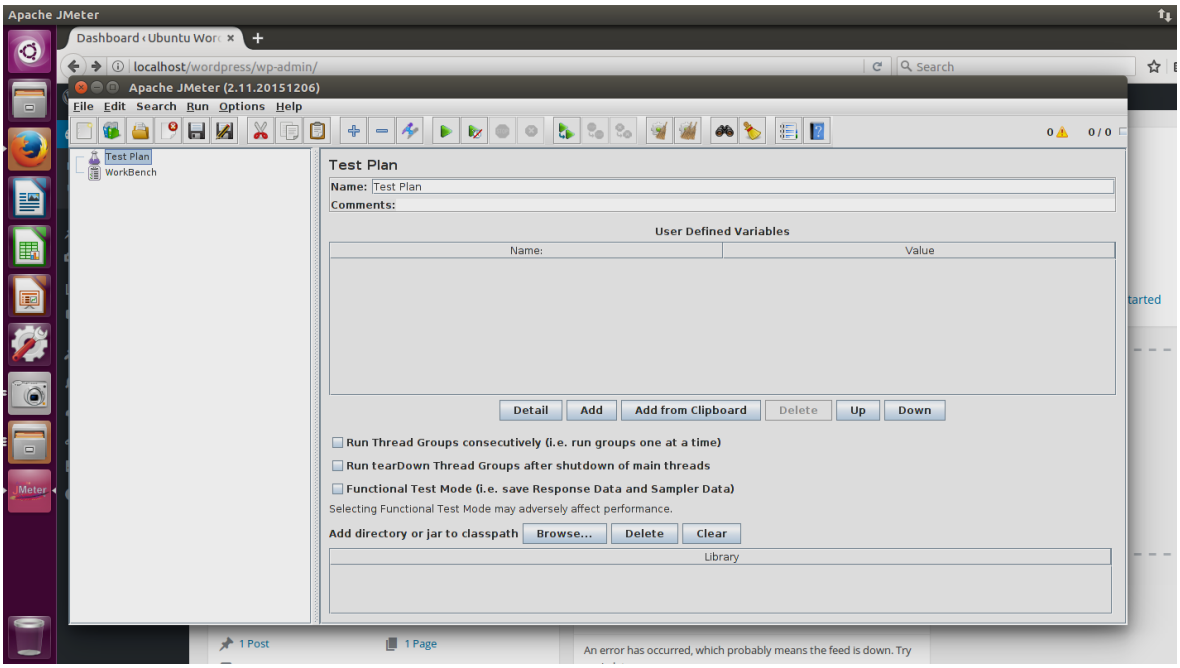


Рисунок 3.1 – Графический интерфейс инструмента для измерения производительности Apache JMeter

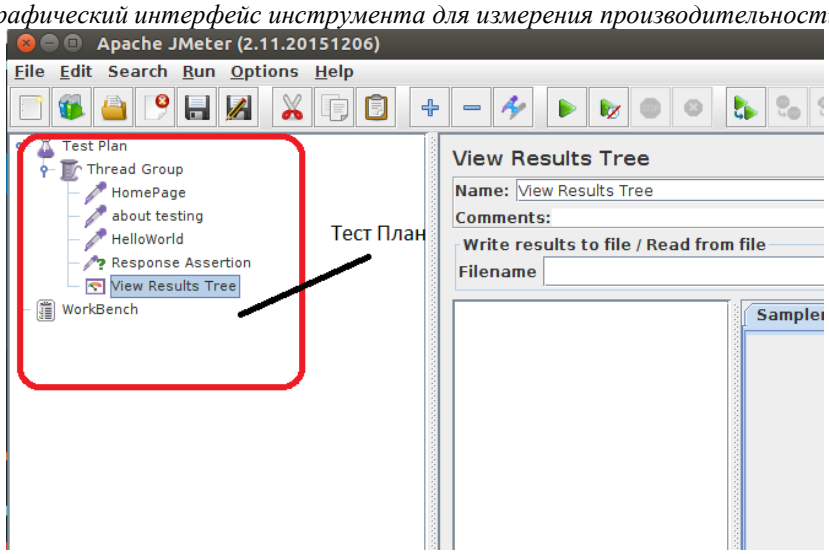


Рисунок 3.2 – Составление тест плана Apache JMeter

Requests	Executions			Response Times (ms)							Networ
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Throughput	Received
Total	**3**	**0**	**0.00%**	**686.00**	**171**	**1691**	**1691.00**	**1691.00**	**1691.00**	**1.39**	**7.46**
HelloWorld	1	0	0.00%	171.00	171	171	171.00	171.00	171.00	5.85	31.49
about testing	1	0	0.00%	196.00	196	196	196.00	196.00	196.00	5.10	27.47
HomePage	1	0	0.00%	1691.00	1691	1691	1691.00	1691.00	1691.00	0.59	3.16

а – один пользователь

Requests	Executions			Response Times (ms)							Netwo
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Throughput	Received
Total	**60**	**0**	**0.00%**	**6712.95**	**1202**	**15346**	**12743.80**	**14691.15**	**15346.00**	**2.61**	**14.04**
about testing	20	0	0.00%	5590.90	2960	11967	8934.90	11817.05	11967.00	1.60	8.61
HelloWorld	20	0	0.00%	2726.60	1202	6194	4079.80	6089.45	6194.00	1.95	10.49
HomePage	20	0	0.00%	11821.35	8851	15346	15002.10	15330.35	15346.00	1.30	6.95

б – двадцать пользователей

Requests	Executions			Response Times (ms)							Netwo
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Throughput	Received
Total	**90**	**0**	**0.00%**	**9888.14**	**762**	**17887**	**16642.40**	**17493.40**	**17887.00**	**2.79**	**14.99**
about testing	30	0	0.00%	9411.87	6048	16852	13421.00	16752.45	16852.00	1.48	7.97
HelloWorld	30	0	0.00%	6208.00	762	12802	10199.10	12660.65	12802.00	2.18	11.72
HomePage	30	0	0.00%	14044.57	10973	17887	17545.20	17792.95	17887.00	1.67	8.94

в – тридцать пользователей

Requests	Executions			Response Times (ms)							Netwo
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Throughput	Received
Total	**150**	**0**	**0.00%**	**16015.81**	**1068**	**29622**	**23844.70**	**25834.25**	**29616.90**	**2.90**	**15.55**
about testing	50	0	0.00%	14687.74	10782	29622	20876.00	25701.00	29622.00	1.47	7.91
HelloWorld	50	0	0.00%	11765.00	1068	20325	16962.20	19746.70	20325.00	2.32	12.52
HomePage	50	0	0.00%	21594.70	16090	29612	25953.50	27546.05	29612.00	1.69	9.01

г– пятьдесят пользователей

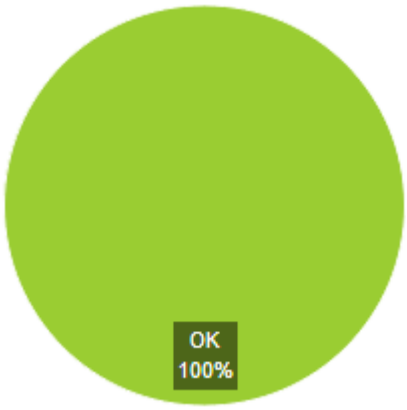
Requests	Executions			Response Times (ms)							Networ
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Throughput	Received
Total	**300**	**0**	**0.00%**	**28937.40**	**263**	**72619**	**38236.30**	**42575.30**	**60841.37**	**2.95**	**15.84**
about testing	100	0	0.00%	27553.36	20936	72619	38329.30	57033.75	72515.62	1.31	7.05
HelloWorld	100	0	0.00%	25833.86	263	46713	29418.70	38150.05	46689.90	1.88	10.13
HomePage	100	0	0.00%	33424.98	24830	44574	40439.40	42043.90	44570.40	2.24	11.96

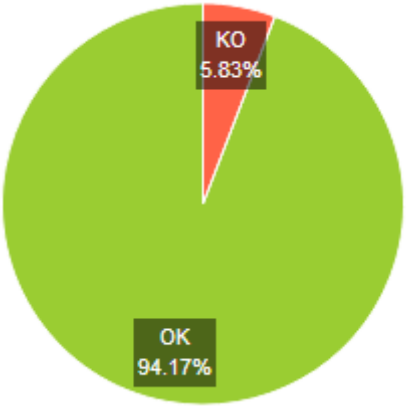
д – сто пользователей

Requests	Executions			Response Times (ms)							Netwo
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Throughput	Received
Total	**600**	**35**	**5.83%**	**83855.28**	**1341**	**298704**	**161982.70**	**224214.60**	**291610.34**	**1.96**	**10.22**
about testing	200	0	0.00%	76435.19	3699	224511	138996.50	194904.25	211290.60	0.80	4.29
HelloWorld	200	0	0.00%	56802.41	1341	169677	88192.50	127971.45	161631.09	1.09	5.89
HomePage	200	35	17.50%	118328.24	51271	298704	261944.20	285946.95	297426.02	0.67	3.24

е– двести пользователей

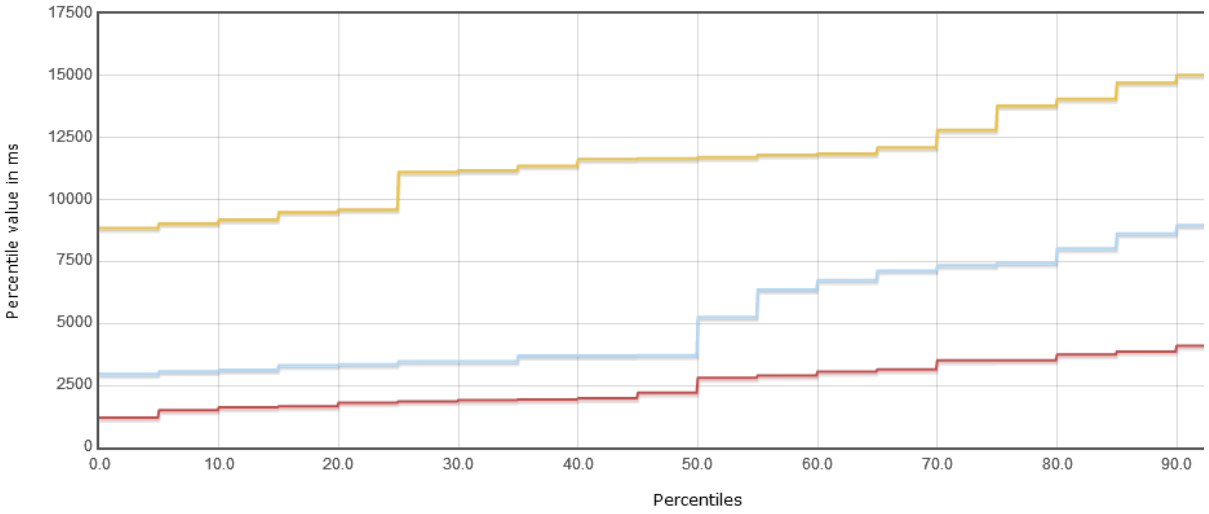
Рисунок 3.3 – Статистика часа отклика для различного количества пользователей



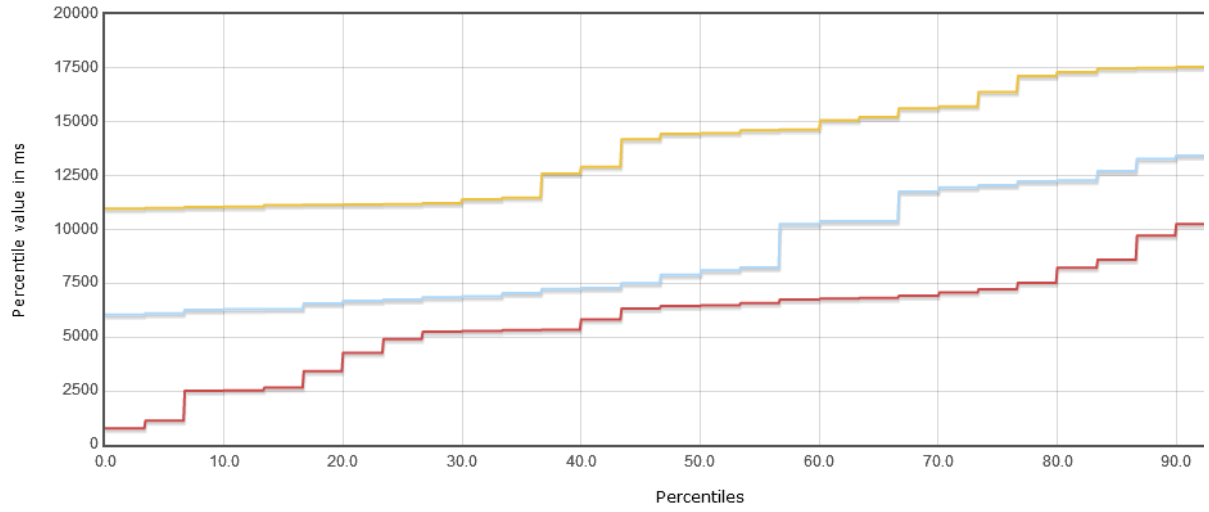


а – один пользователь *б– двести пользователей*

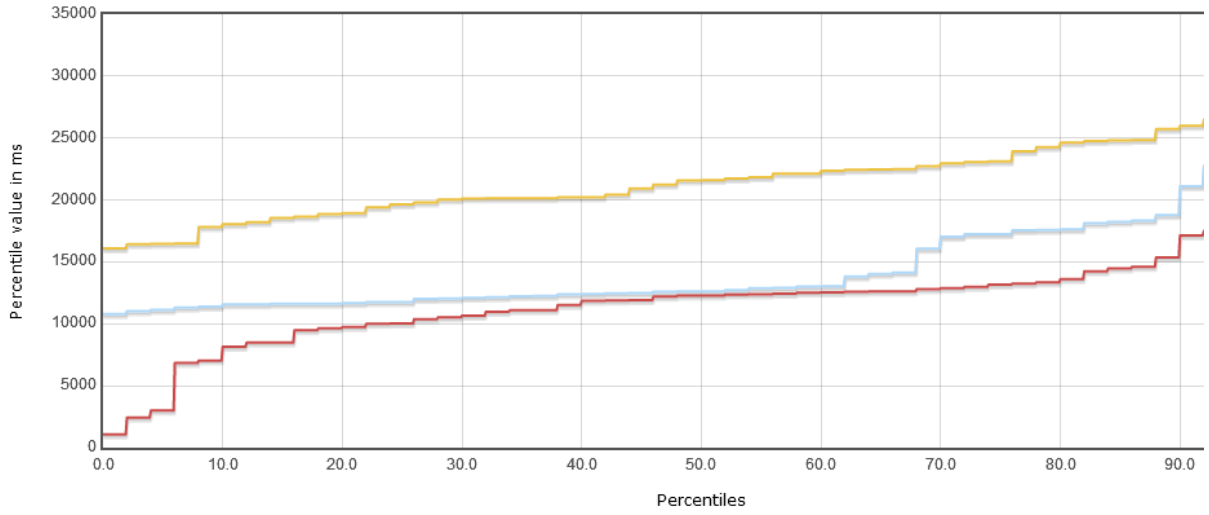
Рисунок 3.4 – Описание запроса (Summary Request)



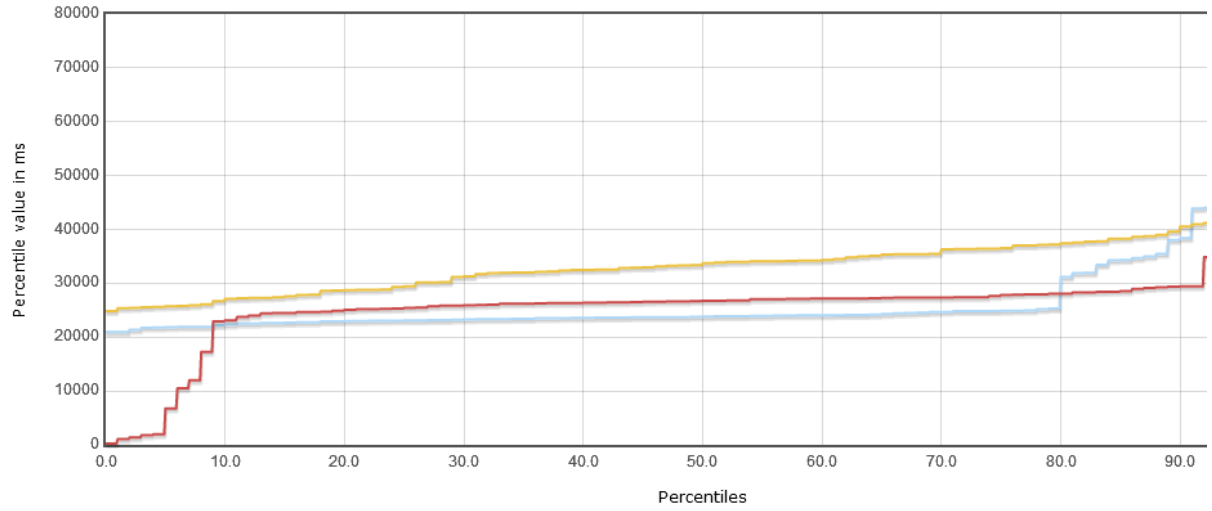
а – двадцать пользователей



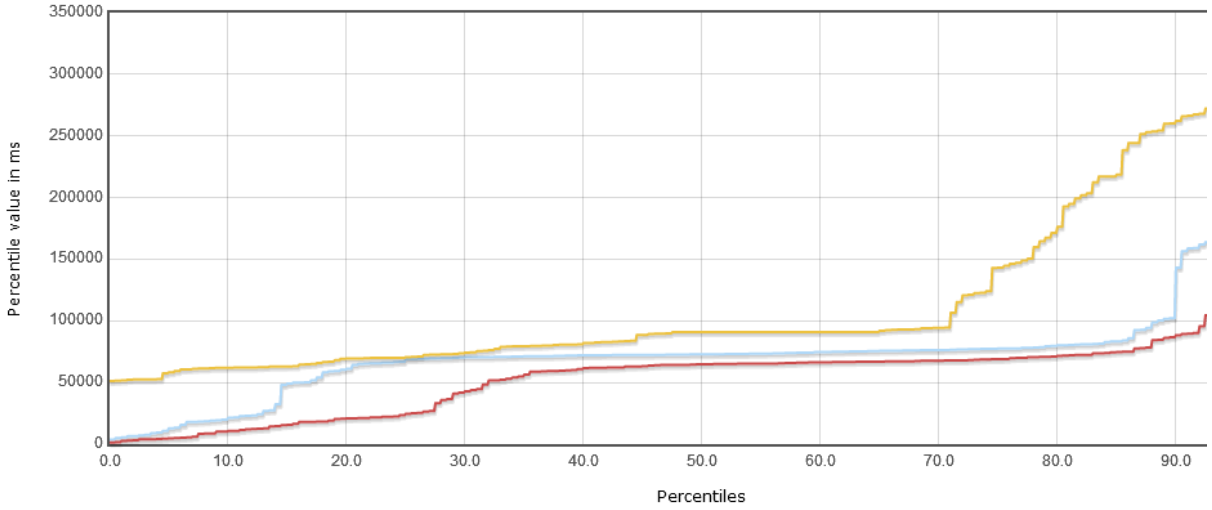
б – тридцать пользователей



в– пятьдесят пользователей



г – сто пользователей



д– двести пользователей

Рисунок 3.5 – Среднее время отклика сервера на запрос пользователя при разной нагрузке сервера

Таблица 3.1 – Реальное поведение системы

Ссылка	Количество пользователей						
	1	20	30	50	100	200	300
1	171	5590	9411	14687	27553	76435	119345
2	196	2726	6208	11765	2533	56802	123664
3	1691	11821	14044	21594	33424	118328	123820

Таблица 3.2– Ожидаемое поведение системы

Ссылка	Количество пользователей						
	1	20	30	50	100	200	300
1	171	3420	5130	8550	17100	34200	51300
2	196	3920	5880	9800	19600	39200	58800
3	1691	33820	50730	84550	169100	338200	507300

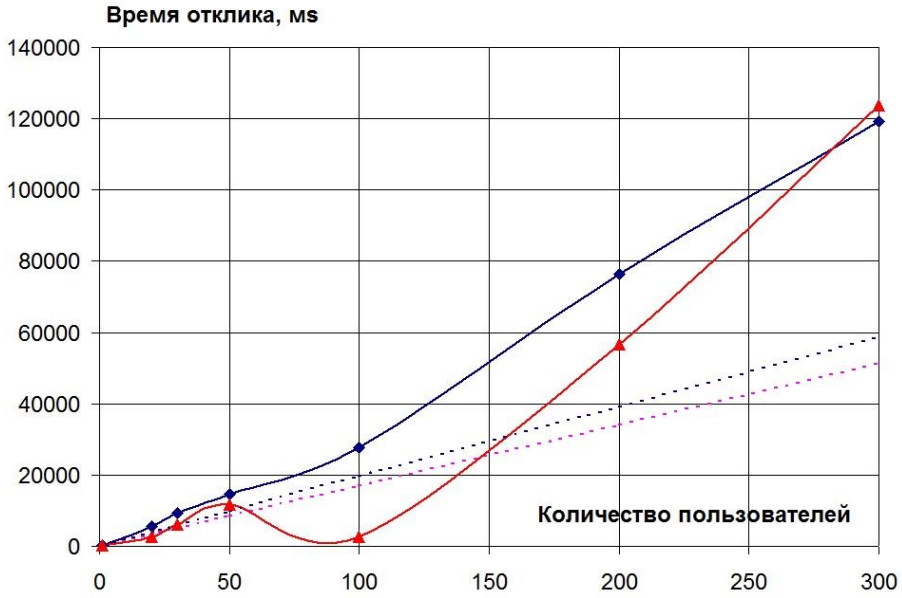


Рисунок 3.6 – Зависимость времени отклика от количества пользователей основная линия – реальное поведение системы, пунктирная линия – ожидаемый результат

Инструмент Jmeter также позволяет тестировать API (Application programming interface). У этого вида тестирования существуют преимущества, в сравнении с GUI (Graphic User interface) тестированием:

1) На этапе раннего тестирования, поскольку вначале разрабатывается API, а потом уже графический интерфейс, существует возможность проверить логику раньше, чем появится GUI.

2) Графического интерфейса вообще может не быть.

3) Скорость – вызвать один запрос занимает доли секунды. А вот через интерфейс повторить процедуру бывает сложно.

4) Автоматизация – даже если нет автоматизированных тестов на уровне API приложения, всегда можно их создать вручную через Jmeter.

Пример составления тест-план показан на рисунке 3.7. В него следует добавить Thread Group – Requesting home page, HTTP запрос, Слушатели (Listeners) View Result.

Tree Graph Results – дерево ответов всех образцов, позволяют увидеть отклик для всех образцов. В добавление к показу отклика индицируется время этого отклика (Sampler Result – рисунок 3.8) и некоторые коды этих откликов (например, в HTML формате – рисунок 3.9).

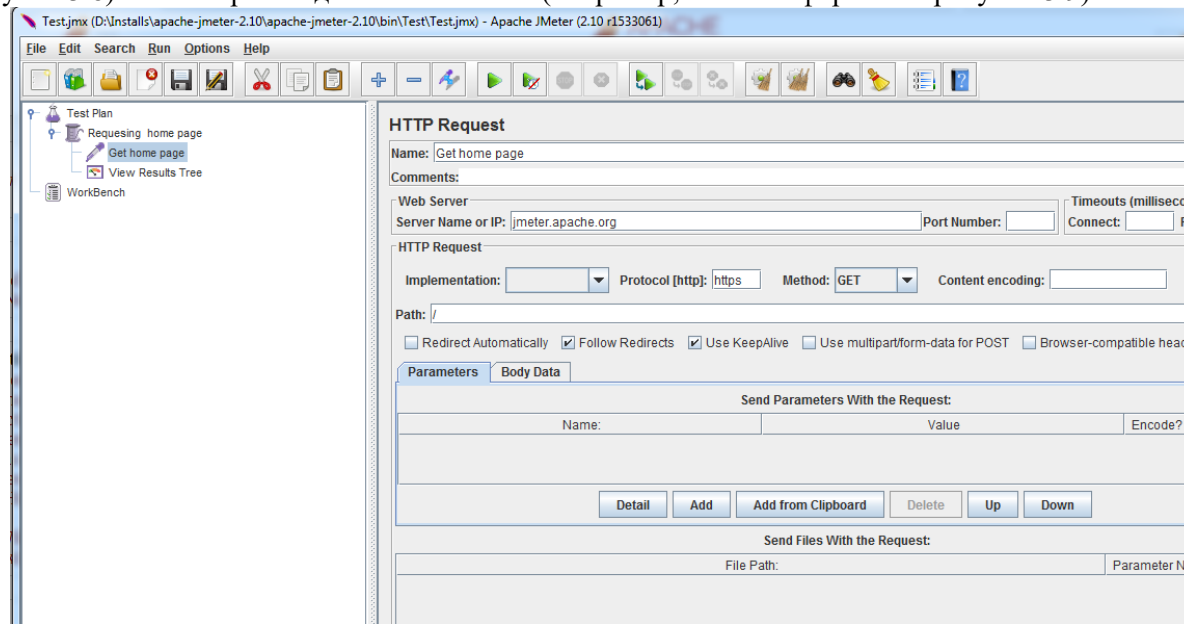


Рисунок 3.7 - Заполнение HTTP запроса

После запуска тест плана, можно увидеть ответ на запрос (рисунок 3.8 и 3.9), сообщение (отклик 200 OK). Если имеются какие-либо проблемы, то сообщение будет выглядеть следующим образом – рисунок 9. Результат запуска тест-плана при выполнении Graph Result показан на рисунке 3.10.

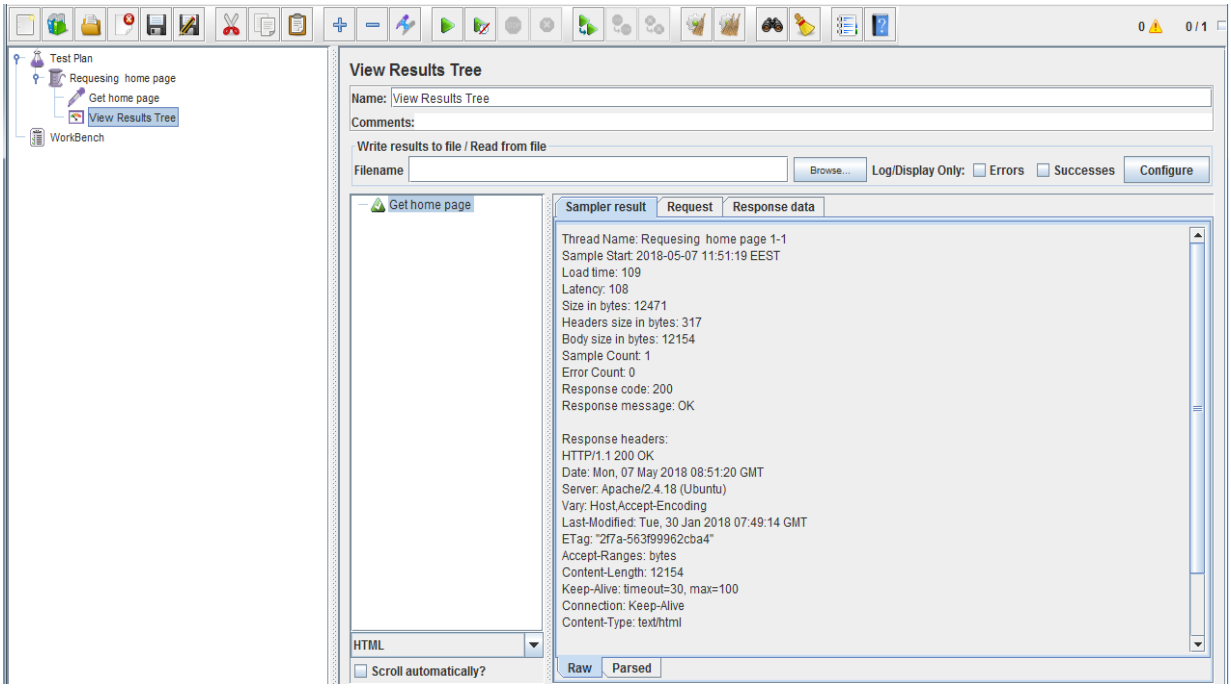


Рисунок 3.8 – Результат запуска тест-плана при выполнении View Result Tree

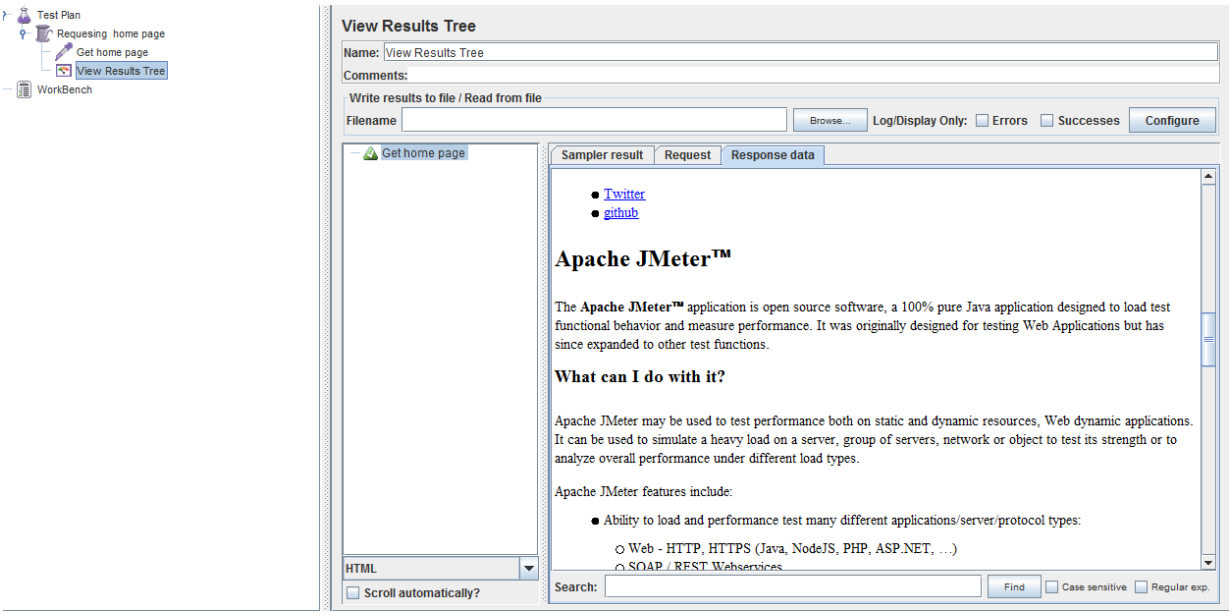


Рисунок 3.9 - Результат запуска тест-плана при выполнении View Result Tree

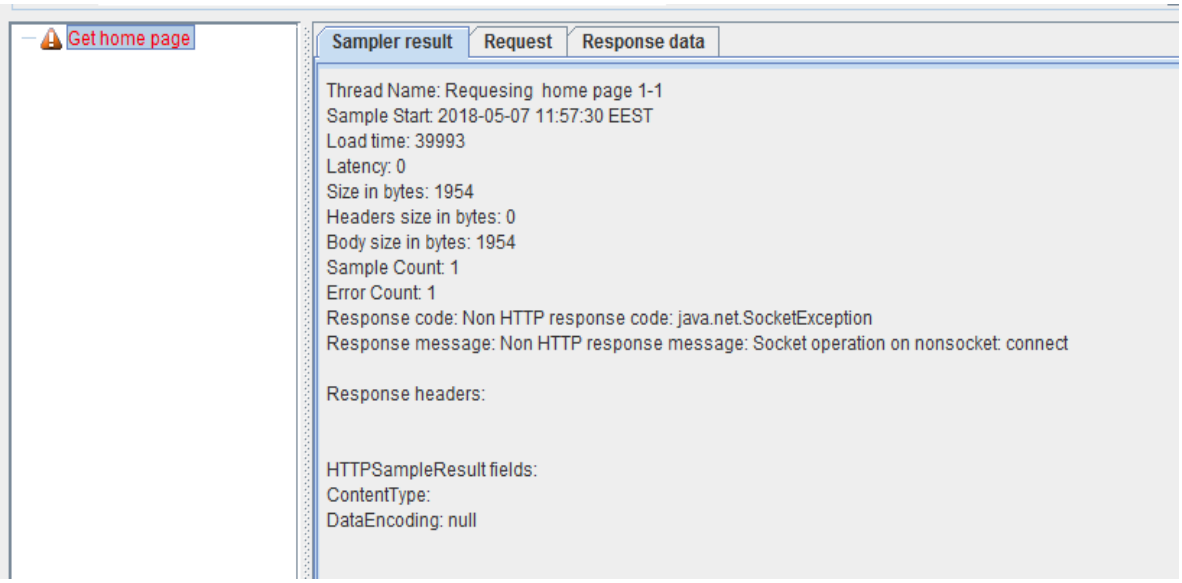


Рисунок 3.10 - Результат запуска тест-плана при выполнении View Result Tree

Выводы

Тестирование производительности – очень сложный и ответственный этап разработки программного обеспечения. Рассмотренная классификация видов тестовых испытаний является, фактически, полнофункциональной замкнутой системой всесторонней оценки качества программных продуктов. При динамическом системном тестировании приложений, тесты разрабатываются на основе функциональных требований исходя из уровня предполагаемой нагрузки на проектируемую систему.

Одним из наиболее эффективных инструментов автоматизированной реализации объективной количественной оценки показателей качества является приложение JMeter. Рассмотренные этапы использования данного инструмента можно рассматривать, как практическое руководство для специалистов-тестировщиков. Особенно ответственным является этап составления тест-плана, которому мы уделили особое внимание в тексте данной работы. Приведенная схема тестирования производительности веб-приложения, иллюстрирует возможности получения количественных данных для анализа производительности в условиях различных уровней нагрузки. Удобство и практическая ценность JMeter заключается в достаточной информативности и иллюстративности результатов испытаний (таблицы и графики) для всестороннего анализа выполнения требований по производительности.

Существенным преимуществом инструмента JMeter является его приспособленность для осуществления API тестирования, позволяющего не только оценить комфортность перспективной эксплуатации программного продукта, но и выявить возможные дефекты логики выполнения вычислительных алгоритмов.

ЛІТЕРАТУРА

1. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 (Last accessed: 23.03.2020).
2. Advanced-Test-Automation-Engineer-Syllabus-GA-2016 URL: https://www.istqb.org/downloads/category/48-advanced-level-test-automation-engineer-documents.html (Last accessed: 23.03.2020).
3. Куликов С. Тестирование программного обеспечения. Базовый курс. 2017. 297 с. URL: http://svyatoslav.biz/software_testing_book/ (Last accessed: 23.03.2020).
4. Jmeter URL: https://jmeter.apache.org/ (Last accessed: 23.03.2020).

REFERENCES

1. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 (Last accessed: 23.03.2020). [in English]
2. Advanced-Test-Automation-Engineer-Syllabus-GA-2016 URL: https://www.istqb.org/downloads/category/48-advanced-level-test-automation-engineer-documents.html (Last accessed: 23.03.2020). [in English]
3. Kulikov S. Software Testing. Basic course. 2017.297 s. URL: http://svyatoslav.biz/software_testing_book/ (Last accessed: 03.23.2020). [In Russian]
4. Jmeter URL: https://jmeter.apache.org/ (Last accessed: 23.03.2020). [in English]

Мелкозьорова Ольга Михайлівна *– кандидат технічних наук; доцент кафедри безпеки інформаційних систем і технологій, Харківський національний університет імені В. Н. Каразіна, майдан Свободи, 4, Харків-22, Україна, 61022;*
e-mail: olja.mex@gmail.com.

Melkozerova Olha M.*, PhD; Associate Professor of Information Systems and Technologies Security Department V. N. Karasin Kharkiv National University, Svobody Sq 4, 61022, Kharkiv, Ukraine,*
e-mail: olja.mex@gmail.com.

Мелкозерова Ольга Михайловна *– кандидат технических наук, доцент кафедры безопасности информационных систем и технологий, Харьковский национальный университет имени В. Н. Каразина, площадь Свободы, 4, Харьков-22, Украина, 61022;*
e-mail: olja.mex@gmail.com.

Рассомахін Сергій Геннадійович *– доктор технічних наук; завідувач кафедри безпеки інформаційних систем і технологій, Харківський національний університет імені В. Н. Каразіна, майдан Свободи, 4, Харків-22, Україна, 61022;*
e-mail: rassomakhin@karazin.ua.

Rassomakhin Sergiy G., *PhD; Doctor of Science, Professor, Head of Department of Information Systems and Technologies Security V. N. Karasin Kharkiv National University, Svobody Sq 4, 61022, Kharkiv, Ukraine,*
e-mail: rassomakhin@karazin.ua.

Рассомахин Сергій Геннадьевич *– доктор технических наук, заведующий кафедры безопасности информационных систем и технологий, Харьковский национальный университет имени В. Н. Каразина, площадь Свободы, 4, Харьков-22, Украина, 61022;*
e-mail: rassomakhin@karazin.ua.

Надійшла у першій редакції 10.03.2020, в останній - 05.05.2020.