RSTQB
Russian Software Testing Qualifications Board

Сертифицированный тестировщик Тестирование производительности Программа обучения

Версия 2018 RU-2

International Software Testing Qualifications Board

ISTQB®
Certified Tester
Performance Testing

Предоставлено

Квалификационной коллегией США по тестированию ПО (ASTQB)

и

Квалификационной коллегией Германии по тестированию ПО (GTB)

ASTQB
American Software Testing Qualifications Board, Inc.

GTB
German Testing Board
Software. Testing. Excellence.

Уведомление об авторских правах

Рабочая группа по тестированию производительности

Graham Bath
Rex Black
Alex Podelko
Andrew Pollner
Randy Rice

История изменений

Версия	Дата	Примечания
Alpha V04	13 декабря 2016	Версия для конференции в г.Нью-Йорк
Alpha V05	18 декабря 2016	Версия по результатам конференции в г.Нью-Йорк
Alpha V06	23 декабря 2016	Изменение структуры главы 4 Перенумерованы и скорректированы цели обучения по результатам договоренностей, достигнутых на конференции в г. Нью-Йорк
Alpha V07	31 декабря 2016	Добавлены комментарии от Rex Black
Alpha V08	12 февраля 2017	Пре-альфа
Alpha V09	16 апреля 2017	Пре-альфа
Alpha Review V10	28 июня 2017	Версия для альфа-рецензии
V2017v1	27 ноября 2017	Версия для альфа-рецензии
V2017v2	15 декабря 2017	Альфа-обновления
V2017v3	15 января 2018	Техническое редактирование
V2017v4	23 января 2018	Рецензия глоссария
V2018 b1	1 марта 2018	Бета-кандидат для ISTQB
V2018 b2	17 мая 2018	Бета-релиз для ISTQB
V2018 b3	25 августа 2018	Комментарии к бета-рецензии включены в релизную версию
Версия 2018	9 декабря 2018	ISTQB GA релиз
RSTQB 2018	12 мая 2019	Программа обучения Сертифицированный тестировщик Базового уровня Тестирование производительности Перевод на русский язык.
2018 RU-2	12 мая 2024	Приведение к шаблону ISTQB v4.0 Обновление логотипа RSTQB Исправление незначительных ошибок

Содержание

Благодарности

Данный документ был подготовлен Квалификационной коллегией США по тестированию ПО (ASTQB) и Квалификационной коллегией Германии по тестированию ПО (GTB) в составе:

Graham Bath (GTB, сопредседатель рабочей группы)
Rex Black
Alexander Podelko (CMG)
Andrew Pollner (ASTQB, сопредседатель рабочей группы)
Randy Rice

Авторы благодарят команду рецензентов за их вклад и замечания. Коллегия ASTQB хотела бы выразить признательность и поблагодарить Computer Measurement Group (CMG) за их вклад в развитие этой программы.
В рецензировании, обсуждении и предоставлении замечаний при подготовке данной программы обучения принимали участие:

Dani Almog	Marek Majernik	Péter Sótér
Sebastian Chece	Stefan Massonet	Michael Stahl
Todd DeCapua	Judy McKay	Jan Stiller
Wim Decoutere	Gary Mogyorodi	Federico Toledo
Frans Dijkman	Joern Muenzel	Andrea Szabó
Jiangru Fu	Petr Neugebauer	Yaron Tsubery
Matthias Hamburg	Ingvar Nordström	Stephanie Ulrich
Ágota Horváth	Meile Posthuma	Mohit Verma
Mieke Jungebload	Michaël Pilaeten	Armin Wachter
Beata Karpinska	Filip Rechtoris	Huaiwen Yang
Gábor Ladányi	Adam Roman	Ting Yang
Kai Lepler	Dirk Schweier	
Ine Lutterman	Marcus Seyfert	

Данный документ был официально выпущен ISTQB 9 декабря 2018г.

0. Предисловие к программе обучения

0.1 Цель данного документа

Данная программа обучения представляет собой основу для подготовки к сертификации базового уровня в области тестирования производительности. ASTQB® и GTB® предоставляют данную программу:

1. Национальным коллегиям с целью перевода на местные языки и аккредитации организаций, занимающихся обучением. Национальные коллегии могут адаптировать программу под особенности конкретного языка и модифицировать ссылки в соответствии с местными требованиями к материалам, представляемым для публикации.
2. Экзаменационным коллегиям для составления экзаменационных вопросов на местном языке, адаптируя цели обучения для каждой программы.
3. Организациям, занимающимся обучением, для разработки учебных курсов и определения соответствующих методов обучения.
4. Кандидатам на получение сертификатов для подготовки к экзамену (в составе учебного курса или для самостоятельного изучения).
5. Международным обществам системной инженерии и разработки программного обеспечения для повышения квалификации в области тестирования ПО и систем, или как основу для книг и статей.

ASTQB и GTB оставляют за собой право предоставлять эту программу и в других целях по предварительной просьбе и письменному разрешению.

0.2 Базовый уровень сертификации в области тестирования производительности

Сертификация базового уровня предназначена для лиц, занимающихся тестированием ПО, желающих расширить свои знания в области тестирования производительности, а также для тех, кто только начинает карьеру в данной сфере. Также она предназначена для лиц, связанных с инженерией производительности, желающих достичь лучшего понимания тестирования производительности.
Данная программа обучения рассматривает следующие основные аспекты тестирования производительности:

1. Технические
2. Методические
3. Организационные

Данная программа подготовки учитывает и развивает знания, указанные в "ISTQB Программа подготовки продвинутого уровня для технического тест-аналитика" (ISTQB® Advanced Level Technical Test Analyst syllabus) [ISTQB_ALTTA_SYL].

0.3 Бизнес-цели

Данный раздел содержит список бизнес-целей, достигаемых кандидатом, прошедшим сертификацию базового уровня в области тестирования производительности.

PTFL-1	Понимать основные концепции производительности и тестирования производительности
PTFL-2	Определять риски, цели и требования производительности в соответствии с потребностями и ожиданиями заинтересованных лиц
PTFL-3	Знать метрики производительности и способы их сбора
PTFL-4	Разрабатывать планы тестирования производительности для достижения установленных целей и требований

PTFL-5 Концептуально проектировать, реализовывать и выполнять базовые тесты производительности
PTFL-6 Анализировать результаты тестов производительности и формулировать выводы для различных заинтересованных лиц
PTFL-7 Объяснять ход процесса, обоснование, результаты и выводы тестирования производительности различным заинтересованным лицам
PTFL-8 Знать типы и назначение инструментов тестирования производительности, а также критерии их выбора
PTFL-9 Определять, как деятельность по тестированию производительности согласуется с жизненным циклом ПО.

0.4 Цели обучения

Цели обучения обеспечивают достижение бизнес-целей, а также используются для разработки экзаменационных испытаний на получение сертификата базового уровня по тестированию производительности. Каждая цель обучения относится к тому или иному уровню познавательного процесса (К-уровень).
К-уровни, или уровни познавательного процесса, используются для классификации целей обучения в соответствии с уточненной теорией таксономии Блума [Anderson01]. ISTQB® использует теорию таксономии при разработке требований к кандидатам на получение сертификата по усвоению программы курса.
Данная программа подготовки рассматривает четыре К-уровня (К1-К4):

К-уровень	**Ключевое слово**	**Описание**
1	Запомнить	Кандидат должен запомнить или распознать термин или понятие
2	Понять	Кандидат должен дать объяснение для утверждения, относящегося к теме рассматриваемого вопроса
3	Применить	Кандидат должен выбрать правильное применение понятия или метода и применить его в рассматриваемом контексте
4	Проанализировать	Кандидат способен для лучшего понимания разделить информацию, связанную с процедурой или методом, на составные части, а также может различить факты и предположения

В целом, все части данной образовательной программы содержат цели обучения уровня К1. Другими словами, кандидат на получение сертификата должен узнать, запомнить и воспроизвести термин или понятие. Цели обучения уровней К2, К3 и К4 указаны в начале содержащих их глав.

0.5 Рекомендованное время обучения

Для каждой цели обучения в данной программе подготовки было определено минимально необходимое учебное время. Суммарное время для изучения каждой главы указано в ее заголовке. Организаторы обучения должны обратить внимание, что другие программы подготовки ISTQB используют так называемый подход «стандартного времени», который для каждой изучаемой цели отводит фиксированное время в зависимости от К-уровня. Программа обучения по тестированию производительности не следует строго данной схеме. В итоге организаторам обучения предоставлены более тщательно подобранные и реалистичные значения учебного времени для каждой цели обучения.

0.6 Начальные требования

Для сдачи экзамена на получение сертификата базового уровня по тестированию производительности кандидат должен получить сертификат базового уровня по тестированию ПО.

0.7 Источники информации

Определения терминов, использованных в данной программе подготовки, даны в документе ISTQB «Стандартный глоссарий терминов, используемых в тестировании программного обеспечения» (ISTQB's Glossary of Terms used in Software Testing) [ISTQB_GLOSSARY].
Раздел 6 содержит список рекомендованных книг и статей по тестированию производительности.

1. Основные понятия – 60 минут

Ключевые слова

Тестирование ресурсоёмкости, тестирование совместной работы, эффективность, тестирование стабильности, создание нагрузки, нагрузочное тестирование, тестирование производительности, тестирование масштабируемости, тестирование производительности при всплесках нагрузки, стрессовое тестирование

Цели обучения

1.1 Принципы и понятия
PTFL-1.1.1 (K2) Понять принципы тестирования производительности

1.2 Виды тестирования производительности
PTFL-1.2.1 (K2) Понять различные виды тестирования производительности

1.3 Активности тестирования производительности
PTFL-1.3.1 (K1) Узнать активности тестирования производительности

1.4 Понятие создания нагрузки
PTFL-1.4.1 (K2) Понять понятия создания нагрузки

1.5 Распространённые проблемы, которые обнаруживает тестирование производительности, и их причины
PTFL-1.5.1 (K2) Привести примеры проблем, которые обнаруживает тестирование производительности, и их причин

1.1 Принципы тестирования производительности

Производительность – важный аспект опыта потребления для огромного числа пользователей, использующих приложения на различных стационарных и мобильных платформах. Тестирование производительности играет критически важную роль в определении приемлемого уровня качества для конечного пользователя и часто тесно связано с другими дисциплинами, такими как инженерия практичности и инженерия производительности.

Кроме того, оценка функциональности, практичности и других характеристик качества в условиях нагрузки, например, в ходе выполнения нагрузочного теста, может обнаружить проблемы, связанные с этими характеристиками, вызванные нагрузкой.

Тестирование производительности не ограничивается областью веб-приложений, в которой внимание фокусируется на конечном пользователе. Оно также применимо к иным областям, основывающимся на различных архитектурах, таких как классическая клиент-серверная архитектура, распределённая архитектура, встроенные системы.

Технически производительность определена в Модели качества продукта (стандарт ISO 25010 [ISO25000]) как нефункциональная характеристика качества с тремя составляющими, описанными ниже. Приоритизация и основной фокус внимания определяется оценкой рисков и потребностями заинтересованных лиц. Анализ результатов тестирования может выявить иные риски, требующие внимания.

Поведение во времени: как правило, целью тестирования производительности в первую очередь является именно оценка поведения во времени. Эта сторона тестирования производительности

посвящена исследованию способности компонента или системы отвечать на действия пользователей и других систем в течение заданного времени и при заданных условиях. Измерение поведения во времени проводится различным образом, от замеров суммарного времени, за которое система ответила на действия пользователей и других систем, до числа тактов ЦПУ, требующихся программному обеспечению на выполнение определённой задачи.

Использование ресурсов: в случае, если доступность ресурсов системы оценивается как риск, возможно проведение исследования использования таковых ресурсов (например, выделения оперативной памяти) путём проведения специальных тестов производительности.

Ресурсоёмкость: в случае, если ресурсоёмкость системы (например, количество пользователей или объёмы данных) идентифицируется как риск, возможно оценить пригодность выбранной архитектуры системы путём проведения тестов производительности.

Поскольку тестирование производительности должно принимать во внимание эти три различные характеристики качества, оно нередко оказывается экспериментальной деятельностью, что необходимо для проведения замеров и анализа специфических параметров системы. Этот процесс может носить итеративный характер и использоваться для помощи в системном анализе, разработке и реализации, позволяя принимать решения в масштабе архитектуры и формировать ожидания заинтересованных лиц.

Следующие общие принципы тестирования особенно важны для области тестирования производительности:

- Тесты должны соответствовать сформированным ожиданиям различных групп заинтересованных лиц, в особенности пользователей, разработчиков системы и обеспечивающего персонала.
- Тесты должны быть воспроизводимы. Повторное проведение тестов на идентичной системе должно приносить статистически идентичные результаты (в пределах заданного допуска).
- Тесты должны приносить результаты, доступные для понимания заинтересованными лицами и пригодные для сопоставления с их ожиданиями.
- Если позволяют ресурсы, тесты должны проводиться либо на полной системе (или ее части), либо на тестовом окружении, в достаточной мере воспроизводящих производственную среду.
- Бюджет тестирования должен быть разумным, а сроки проведения тестов не должны приводить к увеличению сроков выполнения проекта.

Книги [Molyneaux09] и [Microsoft07] излагают фундаментальные знания принципов и практических аспектов тестирования производительности.

Все три вышеописанных характеристики качества оказывают влияние на масштабируемость тестируемой системы.

1.2 Виды тестирования производительности

Существуют различные виды тестирования производительности. Каждый из них может быть использован в том или ином проекте, в зависимости от целей теста.

Тестирование производительности
Тестирование производительности – собирательное понятие, объединяющее все виды тестирования производительности (отклика) системы или ее части под различными объёмами нагрузки.

Нагрузочное тестирование
Нагрузочное тестирование исследует способность системы обрабатывать растущие объёмы ожидаемой реалистичной нагрузки, порождаемой запросами на совершение транзакций контролируемым числом одновременно работающих пользователей или процессов.

Стрессовое тестирование
Стрессовое тестирование исследует способность системы или компонента обрабатывать пиковые объёмы нагрузки на пределе или за пределами ожидаемой или предусмотренной спецификацией пропускной способности. Стрессовое тестирование также используется для оценки работоспособности системы в условиях уменьшенной доступности ресурсов, таких как вычислительные мощности, полоса пропускания сети, память.

Тестирование масштабируемости
Тестирование масштабируемости исследует способность системы удовлетворять будущим требованиям эффективности, которые могут превосходить настоящие требования. Целью таких тестов является способность системы расти в объёме (например, с увеличением количества пользователей, объёмов хранимых данных), не нарушая текущие требования производительности. После определения пределов масштабируемости системы, в производственной среде возможно установить пороговые значения, мониторинг которых позволит предупредить проблемы, которые могут возникнуть. Кроме того, может быть соответствующим образом изменена аппаратная конфигурация продуктивной среды.

Тестирование производительности при всплесках нагрузки
Тестирование при всплесках нагрузки исследует способность системы правильно реагировать на внезапные всплески нагрузки до пикового уровня и затем вернуться в устойчивое состояние.

Тестирование стабильности
Тестирование стабильности исследует устойчивость работы системы на протяжении временного периода, достаточного для ее эффективного использования. Этот вид тестирования позволяет удостовериться в том, что в системе нет проблем, связанных с доступным объёмом ресурсов (например, утечек памяти, соединений к базе данных, пулов потоков выполнения), которые могли бы с течением времени привести к ухудшению производительности и/или возникновению аварий.

Тестирование совместной работы
Тестирование совместной работы исследует последствия ситуаций, в которых определённые действия выполняются одновременно (например, большое количество пользователей одновременно входят в систему). Проблемы, связанные с конкурентным выполнением кода, особенно трудно обнаружить и воспроизвести, когда они возникают в неконтролируемом (или слабо контролируемом) окружении, например, в производственной среде.

Тестирование ресурсоёмкости
Тестирование ресурсоёмкости определяет максимальное количество пользователей и/или транзакций, которое система может поддерживать, обеспечивая при этом выполнение установленных требований производительности. Эти цели также могут быть сформулированы в отношении объемов данных, полученных в результате операций.

1.3 Активности тестирования производительности

Активности тестирования производительности подразделяются на два основных вида: статические и динамические.

1.3.1. Статические активности тестирования

Для тестирования производительности статические активности зачастую имеют большее значение, чем статические активности для функционального тестирования. Это связано с тем, что значительное количество критических дефектов производительности вносится на этапах проектирования архитектуры и устройства системы. Такие дефекты могут происходить из заблуждений или недостатка знаний проектировщиков и архитекторов. Они также могут возникать вследствие того, что в спецификации изначально не были заложены требования к временам

отклика, пропускной способности или использованию ресурсов, не были надлежащим образом описаны ожидаемая нагрузка, сценарии использования системы или ограничения.

Статические активности, связанные с производительностью, включают:

- Рецензирование требований с акцентом на аспекты и риски производительности;
- Рецензирование схемы базы данных, ER-модели, метаданных, хранимых процедур и запросов;
- Рецензирование архитектуры системы и сетевой инфраструктуры;
- Рецензирование критических участков кода (например, сложных алгоритмов).

1.3.2. Динамические активности тестирования

Тестирование производительности необходимо начать максимально рано в жизненном цикле разработки системы.
Некоторые этапы разработки предоставляют удобную возможность проведения динамического тестирования производительности:

- Этап модульного тестирования, в частности, путём использования результатов профилирования кода - для идентификации потенциальных узких мест производительности, и динамического анализа - для оценки использования ресурсов;
- Этап тестирования интеграции компонентов – для тестирования по ключевым сценариям использования, предусматривающим интеграцию компонентов различными способами;
- Этап системного тестирования – для проверки поведения всех компонентов, задействованных в сценарии использования, под различной нагрузкой;
- Этап системного интеграционного тестирования, с особым вниманием к потокам данных и движению рабочего процесса через ключевые интерфейсы между системами. При системном интеграционном тестировании нередко в качестве пользователя выступает другая система (например, входы датчиков или других систем);
- Этап приёмочного тестирования – чтобы сформировать у пользователей, клиентов и операторов системы уверенность в надлежащей производительности системы и внести корректировки в конфигурацию системы сообразно условиям реального мира (но не для обнаружения дефектов производительности).

На этапах высокоуровневого тестирования, таких как системное тестирование и системное интеграционное тестирование, для получения точных результатов необходимо использование реалистичной тестовой среды, то есть аналогичной производственной среде; реалистичных данных и профиля нагрузки (см. главу 4). При работе команды в гибкой методологии разработки или с иным итеративно-инкрементным подходом необходимо включать статическое и динамическое тестирование производительности в ранние итерации, не дожидаясь поздних итераций, для устранения рисков производительности.

Если часть системы представляет собой специально разработанные или инновационные комплектующие, ранние динамические тесты производительности допустимо проводить с использованием эмуляторов. Однако хорошей практикой является как можно более раннее начало тестирования с использованием реальных комплектующих, поскольку эмуляторы зачастую недостоверно воспроизводят реальные ресурсные ограничения и особенности производительности.

1.4 Понятие создания нагрузки

Для проведения различных видов тестирования производительности, описанных в разделе 1.2, необходимо смоделировать, воссоздать и создать характерную для тестируемой системы нагрузку. Понятие нагрузки сравнимо с понятием входных значений, используемых в сценариях тестирования в функциональном тестировании, но отличается следующими принципиальными особенностями:

- Нагрузка теста производительности представляет ввод значений многими пользователями, а не одним пользователем;

- Нагрузка теста производительности может требовать отдельных аппаратных ресурсов и специальных инструментов её генерации;
- Подход к созданию нагрузки в тестировании производительности зависит от функциональной стабильности тестируемой системы (т.е. отсутствия функциональных дефектов, влияющих на выполнение теста).

Эффективное и надёжное создание требуемой нагрузки является ключевым фактором успеха при проведении тестов производительности. Существуют различные способы создания нагрузки.

Создание нагрузки через пользовательский интерфейс
Этот способ подходит в случае, если необходимо воспроизвести действия лишь небольшого числа пользователей, и в распоряжении имеется достаточное количество программных клиентов для осуществления пользовательского ввода. Такое тестирование можно сочетать с использованием инструментов выполнения функциональных тестов, но подход быстро становится непрактичным по мере роста количества эмулируемых пользователей. Стабильность пользовательского интерфейса (UI) также является критически важным фактором успеха тестирования, поскольку может повлиять на воспроизводимость тестов производительности и существенно увеличить расходы на их поддержку. Тестирование через пользовательский интерфейс может представить наиболее полные результаты в отношении тестирования с точки зрения конечных пользователей.

Создание нагрузки с помощью большого числа тестировщиков
Использование такого подхода возможно в случае доступности большого числа тестировщиков, которые могли бы воспроизводить поведение реальных пользователей. В этом случае работу тестировщиков можно организовать таким образом, чтобы подать желаемый объём нагрузки на систему. Этот метод подходит для тестирования приложений, высокодоступных в географическом плане (например, веб-приложений), и позволяет создать нагрузку посредством широкого круга различных устройств с разнообразными конфигурациями. Хотя данный подход и позволяет использовать очень большое количество пользователей, однако поданную нагрузку нельзя будет воспроизвести с такой точностью, как в случае других методов. Кроме того, вариант создания нагрузки с помощью большого числа тестировщиков более сложен в организации по сравнению с другими методами.

Создание нагрузки через программный интерфейс (API)
Этот подход похож на создание нагрузки через пользовательский интерфейс, но для эмуляции пользовательского взаимодействия с тестируемой системой используется программный интерфейс приложения. Благодаря этому такой подход менее чувствителен к изменениям (например, задержкам) в пользовательском интерфейсе и позволяет выполнять операции независимо от таких изменений. Могут быть разработаны специальные скрипты, многократно совершающие вызовы нужных функций API, что позволяет эмулировать поведение большего числа пользователей по сравнению с тестированием через пользовательский интерфейс.

Создание нагрузки использованием протокола обмена данными
Данный подход предполагает запись взаимодействия пользователя с тестируемой системой на уровне протокола обмена данными и последующее воспроизведение полученного скрипта для эмуляции потенциально очень большого числа пользователей воспроизводимым надёжным образом. Этот подход основан на использовании инструментов, описанных в разделах 4.2.6 и 4.2.7.

1.5 Распространённые виды проблем производительности и их причины

Хотя динамическое тестирование производительности позволяет обнаружить множество проблем различных видов, связанных с производительностью, можно привести примеры самых распространённых проблем (в том числе связанных с аварийными отказами системы) и типичные причины таких проблем.

Медленный отклик под любой нагрузкой
В некоторых случаях время отклика может быть неприемлемо большим вне зависимости от объёма подаваемой нагрузки. Это может быть вызвано такими проблемами производительности как плохое проектирование или реализация базы данных, большое время отклика сети, фоновая нагрузка, и многое другое. Такие проблемы могут быть обнаружены в ходе функционального тестирования и тестирования практичности, а не только тестирования производительности, поэтому тест-аналитики должны всегда иметь их в виду и фиксировать их обнаружение в отчётах.

Медленный отклик под умеренной и значительной нагрузкой
В некоторых случаях время отклика ухудшается до неприемлемого уровня при умеренной нагрузке, даже если такая нагрузка является предусмотренной, ожидаемой и допустимой. Эта ситуация может быть обусловлена исчерпанием одного или нескольких ресурсов либо варьирующимся объёмом фоновой нагрузки.

Ухудшение времени отклика с течением времени
В некоторых случаях время отклика ухудшается постепенно или резко с течением времени. Среди причин, которые могут приводить к таким последствиям, – утечки памяти, фрагментация диска, растущая нагрузка на сеть, исчерпание свободного места файлового хранилища, непредусмотренное разрастание базы данных.

Некорректная или аварийная обработка ошибок под большой нагрузкой
В некоторых случаях при большой или чрезмерной нагрузке время отклика системы остаётся на приемлемом уровне, но ухудшается обработка ошибок. Причиной этого может быть нехватка пулов ресурсов, недостаточный размер очередей и стеков или слишком короткий интервал времени ожидания.

Конкретные примеры перечисленных выше проблем:

- Веб-приложение, предоставляющее информацию об услугах, оказываемых организацией, не отвечает на пользовательские запросы в течение семи секунд (общепринятое требование в отрасли). Требуемый уровень производительности не может быть достигнут при определенной нагрузке.
- Система аварийно завершает работу или не способна ответить на пользовательский ввод при неожиданном получении большого количества пользовательских запросов (например, при выставлении на продажу билетов на важное спортивное мероприятие). Ресурсоёмкость системы недостаточна для обработки такого количества пользователей.
- Время отклика системы существенно ухудшается при получении пользовательских запросов на предоставление большого объёма данных (например, на веб-странице доступен для скачивания большой важный отчёт). Ресурсоёмкость системы недостаточна, чтобы обрабатывать передачу таких объёмов данных.
- Пакетная обработка задач (например, ночная) не может завершиться к началу обработки задач в интерактивном режиме. Системе недостаточно времени для выполнения пакетной обработки.
- В системе реального времени исчерпывается оперативная память, когда одновременно работающие процессы требуют больших объёмов памяти, чем может быть выделено в этот момент. Либо доступный объём памяти недостаточен, либо необходимо иначе определять приоритеты выделения памяти.
- В системе реального времени компонент А подаёт входные значения на обработку компоненту Б, но не успевает делать это с требуемой интенсивностью. Чтобы обеспечить достижение необходимой интенсивности предоставления значений, модули компонента А необходимо исследовать и модифицировать (провести профилирование производительности).

2. Основы измерения производительности - 55 минут

Ключевые слова

измерение, метрика

Цели обучения

2.1 Типичные метрики, собираемые при тестировании производительности
PTFL-2.1.1 (K2) Понять типичные метрики, собираемые при тестировании производительности

2.2 Подготовка результатов тестирования производительности
PTFL-2.2.1 (K2) Объяснить, зачем нужна подготовка результатов тестирования производительности

2.3 Подходы, используемые для сбора метрик
PTFL-2.3.1 (K2) Понять основные источники метрик производительности

2.4 Типичные результаты тестирования производительности
PTFL-2.4.1 (K1) Запомнить типичные результаты тестирования производительности

2.1 Типичные метрики, собираемые при тестировании производительности

2.1.1 Для чего нужны метрики производительности

Точные измерения и метрики, полученные на их основе, важны как для формирования целей, так и для оценки результатов тестирования производительности. Тестирование производительности не должно выполняться без понимания того, какие метрики и измерения требуется получить. Невыполнение этого условия приведет к следующим рискам:

- Неизвестно, насколько приемлемы полученные в результате тестирования уровни производительности
- Требования производительности представляют собой не измеряемые величины
- Не всегда возможно определить тренды ухудшения или улучшения производительности
- Нельзя сравнить результаты теста производительности с набором результатов измерений производительности базовой версии, которая определяет допустимый или недопустимый уровень производительности
- Результаты теста производительности оцениваются субъективным мнением одного или нескольких людей
- Результаты, полученные с помощью инструмента нагрузочного тестирования, не будут понятны

2.1.2 Сбор метрик производительности

Как и при любых формах измерений, метрики следует получать и выражать точными способами. Поэтому любые метрики и измерения, описанные в этом разделе, могут и должны быть определены так, чтобы быть значимыми в конкретном контексте. Для решения этого вопроса выполняются предварительные тесты и выясняется, какие метрики необходимо подвергнуть дальнейшей обработке, а какие добавить.
Например, время отклика, скорее всего, будет находиться в любом наборе метрик производительности. Однако, чтобы быть значимой и действенной, метрика времени отклика

должна быть дополнена информацией о времени суток, количестве одновременно работающих пользователей, объеме обрабатываемых данных и так далее
Набор метрик, собираемых во время конкретного теста производительности, будет изменяться в зависимости от:

- бизнес-контекста задачи (бизнес-процессов, поведения заказчика или пользователя, и ожиданий заинтересованных сторон)
- операционного контекста (технологий и их использования)
- целей тестирования.

Например, набор метрик, выбранный для нагрузочного тестирования веб-сайта международной электронной коммерции, будет отличаться от набора метрик тестирования производительности встроенной системы контроля функционирования медицинских устройств.

Обычный путь выбора показателей и метрик производительности учитывает техническую архитектуру, бизнес-окружение или рабочее окружение, в котором требуется оценить производительность.

Наиболее часто в нагрузочном тестировании используются следующие категории показателей и метрик.

Техническое окружение
Метрики производительности будут зависеть от типа технического окружения, как показано в следующем списке:

- Веб-системы
- Мобильные приложения
- Интернет вещей (IoT)
- Клиентские устройства для настольных приложений
- Серверная обработка
- Мейнфреймы
- Базы данных
- Сети
- Характер программного обеспечения, работающего в конкретном окружении (например, встроенное ПО)

Метрики могут включать в себя следующие показатели:

- Время отклика (например, на транзакцию, на пользователя, времена загрузки страниц)
- Утилизация ресурса (например, загрузка процессора, памяти, пропускная способность сети, сетевая задержка, доступное место на диске, интенсивность подсистемы ввода вывода, свободные и занятые потоки)
- Интенсивность ключевой транзакции (то есть число транзакций, которые могут быть обработаны за заданное время)
- Время обработки пакетных операций (например, времена ожидания, времена обработки, времена полного выполнения операций, время ответа базы данных)
- Количество ошибок, влияющих на производительность
- Времена выполнения операций (например, для вставки, чтения, обновления и удаления данных)
- Фоновая утилизация общего ресурса (особенно в виртуализованных окружениях)
- Метрики программного обеспечения (например, сложность исходного кода)

Бизнес-окружение
Со стороны бизнеса или функционального предназначения, метрики производительности могут включать следующее:

- Эффективность бизнес-процесса (например, скорость выполнения всего бизнес-процесса, включая обычные, альтернативные и ошибочные сценарии)

- Интенсивность обработки данных, транзакций и других выполненных единиц работы (например, число заказов, обработанных за час, число строк, вставленных за минуту)
- Интенсивность событий соблюдения или нарушения соглашения об уровне обслуживания (SLA) (например, число нарушений в единицу времени)
- Область применимости (например, процент глобальных или региональных пользователей, выполняющих действие в заданное время)
- Совместное использование (например, число пользователей, одновременно выполняющих какое-либо действие)
- Временные характеристики (например, число заказов, обработанных в пиковые периоды нагрузки)

Операционное окружение
Задачи операционного окружения — это не то, с чем сталкивается конечный пользователь системы, а то, с чем обычно работают администраторы. С точки зрения тестирования производительности они включают:

- Операционные процессы (например, время, требуемое для старта окружения, выполнения резервного копирования, останова и возобновления работы)
- Восстановление системы (например, например, время, требуемое на восстановление данных из резервного хранилища)
- Оповещения и предупреждения (например, время, требуемое системе, чтобы выдать предупреждение или оповещение)

2.1.3 Выбор метрик производительности

Стоит отметить, что подход «больше метрик» не всегда значит «лучше». Каждая выбранная метрика требует соответствующих способов сбора и предоставления отчетности. Важно определить доступный набор метрик, связанных с целями конкретного теста производительности.

Например, подход Цель-Вопрос-Метрика (Goal-Question-Metric, GQM) – полезный способ, позволяющий сопоставить метрики с целями производительности. Идея состоит в том, чтобы сначала установить цели, а потом задавать вопросы, позволяющие узнать, достигнуты цели или нет. Для каждого вопроса есть связанные метрики, которые обеспечивают измеримость полученного ответа (см. Главу 4.3 Программы обучения Экспертного уровня - Улучшение Процесса Тестирования [ISTQB_ELTM_ITP_SYL] для более полного описания подхода GQM). Стоит отметить, что подход GQM не всегда подходит для процесса нагрузочного тестирования. Например, некоторые метрики нагрузочного тестирования отражают работоспособность системы и не связаны напрямую с целями.

Обычно, после определения первичного списка показателей и сбора по ним измерений оказывается, что для более полного понимания уровня производительности и определения мест, где могут потребоваться корректирующие действия, необходимы дополнительные измерения и метрики.

2.2 Подготовка результатов нагрузочного тестирования

Цель подготовки метрик производительности – обеспечение возможности их понимания и отображения в таком виде, который точно представляет полную картину производительности системы. Бывает сложно сделать правильные выводы, особенно это касается заинтересованных лиц со стороны бизнеса, когда метрики производительности представлены только на детальном уровне.

Для многих заинтересованных лиц основная цель состоит в том, чтобы убедиться, что время отклика системы, веб сайта или другого объекта тестирования находится в допустимых пределах.

Как только получено более глубокое понимание метрик производительности, эти метрики могут быть дополнительно обработаны, преследуя следующие цели:

- Заинтересованные лица со стороны проекта и бизнеса смогут увидеть общий статус производительности системы
- Можно определить тренды производительности
- Метрики производительности представлены более понятно

2.3 Основные источники метрик производительности

Процесс сбора метрик должен оказывать минимальное влияние на производительность системы (так называемый «эффект зондирования»). Кроме того, объем, точность и скорость, с которыми метрики должны собираться, формируют требования к используемым инструментам сбора метрик. Пока практика совместного использования нескольких инструментов достаточно распространена, это может повлечь избыточность и другие проблемы (см. Главу 4.4).

Существует три основных типа инструментов сбора метрик производительности:

Инструменты нагрузочного тестирования
Все инструменты нагрузочного тестирования предоставляют набор измерений и метрик в качестве результата тестирования. Инструменты могут отличаться количеством показываемых метрик, способом отображения метрик и возможностями настройки (см. Главу 5.1).

Некоторые инструменты собирают и отображают метрики производительности в текстовом формате, а более продвинутые инструменты собирают и отображают метрики также и в графическом формате на единой панели. Многие инструменты также имеют возможность экспорта метрик.

Инструменты мониторинга производительности
Инструменты мониторинга производительности часто используются в дополнение к возможностям отчетности используемых инструментов нагрузочного тестирования (см. Главу 5.1). Кроме того, инструменты мониторинга могут использоваться для мониторинга производительности системы на постоянной основе и для информирования системных администраторов о снижении производительности или повышении уровня появления ошибок или предупреждений системы. Такие инструменты также могут использоваться для обнаружения и уведомления в случае подозрительного поведения (такого, как DoS атаки и DDoS атаки).

Инструменты анализа лог-файлов
Существуют инструменты, которые сканируют лог-файлы сервера и собирают на их основе метрики. Некоторые из таких инструментов могут предоставлять полученную информацию в графическом виде.

Информация об ошибках и предупреждениях обычно записывается в лог-файлах серверов. Они обычно включают в себя:

- Высокий уровень использования ресурса, например, высокая утилизация CPU, близкое к исчерпанию свободное место на диске, недостаточная пропускная способность сети
- Ошибки и предупреждения использования памяти, например, нехватка памяти
- Блокировки и проблемы многопоточности, особенно при выполнении операций баз данных
- Ошибки баз данных, такие как исключения и таймауты SQL

2.4 Типичные результаты тестирования производительности

В функциональном тестировании, особенно при проверке конкретных функциональных требований или функциональных элементов пользовательской истории, ожидаемые результаты обычно могут быть четко определены, а полученные результаты однозначно интерпретированы, чтобы понять, прошел тест успешно или нет. Например, отчет месячных продаж содержит верную или неверную общую сумму.

В то время как тесты, проверяющие функциональность, выигрывают от хорошо определенных тестовых предсказателей, тестирование производительности часто не имеет такого источника информации. Не только заинтересованные лица не могут сформулировать требования производительности, но и многие бизнес-аналитики и менеджеры по разработке также не могут сформулировать такие требования. Тестировщики часто получают ограниченные рекомендации для определения ожидаемых результатов тестирования.

Оценивая результаты тестирования производительности, важно досконально изучить результаты. Изначальные результаты могут ввести в заблуждение, скрывая проблемы производительности за остальными результатами, которые оказались в целом хорошими. Например, утилизация ресурсов может быть ниже 75% для всех ключевых ресурсов, потенциально содержащих узкие места, но пропускная способность оказалась меньше, или время отклика ключевых транзакций или сценариев использования получились на порядок ниже ожидаемых.

Конкретные результаты, которые следует оценивать, различаются в зависимости от запущенных тестов и часто включают темы, обсужденные в Главе 2.1.

3. Тестирование производительности в жизненном цикле программного обеспечения – 195 мин

Ключевые слова

метрика, риск, жизненный цикл разработки программного обеспечения, протокол тестирования

Цели обучения

3.1 Основные активности тестирования производительности
PTFL-3.1.1 (K2) Понять основные активности, выполняемые в процессе тестирования производительности

3.2 Риски производительности различных архитектур
PTFL-3.2.1 (K2) Объяснить основные категории рисков, влияющих на производительность для разных аппаратных архитектур

3.3 Риски производительности в течение жизненного цикла разработки программного обеспечения
PTFL-3.3.1 (K4) Проанализировать риски, влияющие на производительность данного продукта в течение жизненного цикла его разработки

3.4 Активности тестирования производительности
PTFL-3.4.1 (K4) Проанализировать проект и определить активности, необходимые для тестирования производительности в каждой фазе жизненного цикла разработки программного обеспечения

3.1 Основные активности тестирования производительности

Тестирование производительности имеет итеративный характер. В процессе тестирования каждая итерация дает ценную информацию о производительности приложения и системы. Данные, собранные во время проведения одной итерации теста, используются для правки или оптимизации параметров приложения и системы. Следующая итерация выявит результаты сделанных ранее изменений и так далее, пока не будут достигнуты цели тестирования.

Активности, выполняемые в процессе тестирования производительности, совпадают с базовым процессом тестирования ISTQB, описанном в [ISTQB_FL_SYL].

Планирование процесса тестирования
Планирование процесса тестирования особенно важно для тестирования производительности, поскольку. необходимо заранее выделить тестовые среды, тестовые данные, инструменты и человеческие ресурсы. Кроме того, на этом этапе задаются ограничения, в пределах которых будет проводиться тестирование производительности.

Во время планирования процесса тестирования составляется список возможных рисков и проводится их анализ, а также актуализируется значимая информация в каждом документе, относящемся к планированию тестирования (например, плане тестирования, уровневом плане тестирования). По мере необходимости намеченные ранее планы тестирования пересматриваются и модифицируются, в результате чего меняются условия возникновения рисков. Для отражения этих изменений соответствующим образом изменяются и риски, их уровни и статусы..

Мониторинг и контроль процесса тестирования

Для своевременной реакции на возможные проблемы, которые могут повлиять на производительность, организуется процесс мониторинга хода тестов и подготавливаются планы действий. Такими проблемами могут быть:

- необходимость увеличения создаваемой нагрузки в случае, когда используемая инфраструктура не обеспечивает заданный ее уровень
- измененное, новое или замененное оборудование
- изменения сетевых компонентов
- изменения в реализации программного обеспечения

Для проверки достижения критериев выхода производится оценка целей теста производительности.

Анализ тестов
Для создания эффективных тестов производительности должны быть проанализированы требования производительности, цели тестирования, соглашения об уровне обслуживания (SLA), архитектура системы, модели процессов и другие факторы, определяющие рамки тестирования. Дополнительно можно провести моделирование и анализ требований к системным ресурсам и/или поведения системы с помощью сводных таблиц или инструментов планирования ресурсов.
Проектирование тестов производительности задаёт конкретные условия, такие как уровни нагрузки, условия синхронизации, и транзакции, подлежащие тестированию. После этого определяются необходимые типы тестирования производительности (например, нагрузочное, стрессовое тестирование, тестирование масштабируемости).

Проектирование тестов
На данном этапе разрабатываются сценарии тестирования производительности. В основном, они имеют модульный характер и могут служить в качестве составных блоков более сложных тестов производительности (см. раздел 4.2).

Реализация теста
На этапе реализации из тестовых сценариев производительности создаются процедуры тестирования производительности. Эти процедуры должны отражать последовательность действий, обычно выполняемых пользователем тестируемой системы, и другие функциональные активности, которые должны быть учтены во время тестирования.

На этом же этапе перед каждым выполнением теста подготавливается или развёртывается заново тестовая среда. Поскольку тестирование производительности чаще всего использует конкретные бизнес-данные, их необходимо заранее подготовить. Для создания корректной рабочей модели подготовленные тестовые данные должны соответствовать фактическим производственным данным по объему и типу.

Выполнение теста Выполнение тестов осуществляется зачастую с использованием инструментов тестирования производительности. Полученные результаты тестирования оцениваются для определения соответствия производительности системы требованиям и другим установленным целям. Все дефекты должны быть зафиксированы.

Завершение тестирования Результаты тестирования производительности предоставляются заинтересованным сторонам (например, архитекторам, менеджерам, владельцам продукта) в виде итогового отчета. В отчёте результаты выражены в метриках, которые часто группируются для упрощения понимания. Для более лёгкого восприятия результатов вместо текстовых метрик обычно используют визуальные средства отчетности, такие как диаграммы и графики.

Тестирование производительности принято считать непрерывным процессом, поскольку оно выполняется на всех стадиях тестирования (компонентного, интеграционного, системного, системного интеграционного и приемочного). По завершении определенной итерации тестирования производительности может быть инициировано завершение тестирования, когда созданные тесты, тестовые артефакты (примеры и процедуры тестирования), тестовые данные и другое тестовое обеспечение архивируются или передаются другим тестировщикам для последующего использования во время работ по сопровождению системы.

Поскольку во время тестирования производительности могут быть созданы или изменены большие объемы данных, на этапе завершения может потребоваться развёртывание тестовой среды заново в исходное состояние. Это делается не только в конце проекта, но и после каждого цикла тестирования.

3.2 Категории рисков производительности для разных архитектур

Как уже говорилось, производительность приложения или системы значительно отличается на разных аппаратных архитектурах, программном обеспечении и системных средах. Хотя составить полный список рисков для всех систем вряд ли возможно, в приведенном ниже списке указаны некоторые из самых типичных:

Системы, построенные на базе единственного компьютера
Это системы или приложения, целиком запущенные на одном физическом (не виртуальном) компьютере.
Производительность может снижаться из-за:

- чрезмерного потребления ресурсов, включая: утечки памяти, фоновую нагрузку (например, работу ПО, обеспечивающего безопасность), медленные подсистемы хранения данных (например, низкоскоростные внешние устройства или фрагментированные жесткие диски), неоптимальную настройку операционной системы;
- использования алгоритмов, которые неэффективно используют доступные ресурсы (например, основную память) и в результате выполняются медленнее, чем требуется.

Многоуровневые системы
Это многокомпонентные системы, которые работают на нескольких серверах, каждый из которых выполняет определенную роль, такую, как сервер базы данных, сервер приложений или сервер презентаций. Безусловно, эти серверы фактически являются компьютерами и подвержены тем же рискам, что перечислены ранее для систем на базе единственного компьютера. Вдобавок, производительность может ухудшиться из-за плохой или немасштабируемой схемы базы данных, узких мест в сети, снижения пропускной способности или мощности на любом из серверов.

Распределённые системы
Эти многокомпонентные системы похожи на многоуровневые, но отличие в том, что рабочие серверы могут быть географически распределены и меняться по ходу работы. Так может работать система электронной торговли, которая обращается к различным серверам баз данных в зависимости от географического местоположения человека, разместившего заказ. В дополнение к рискам многоуровневых архитектур, распределённая архитектура может испытывать проблемы с производительностью критических рабочих процессов или потоков данных, на которые влияют ненадежные или непредсказуемые удаленные серверы, особенно когда такие серверы сталкиваются с проблемами сетевого подключения или периодами интенсивной нагрузки.

Виртуализированные системы
В таких системах на физическом оборудовании запущено несколько виртуальных машин. В них размещают системы и приложения, рассчитанные на работу на одном компьютере, а также серверы, которые являются частями многоуровневой или распределенной архитектур. Среди рисков производительности, специфичных для виртуализации, можно упомянуть чрезмерную нагрузку виртуальных машин на оборудование или неправильную настройку виртуальной машины, что приводит к повышенному потреблению ресурсов.

Эластичные/облачные системы
Такие системы предусматривают возможность масштабирования по необходимости, выделяя дополнительные ресурсы по мере увеличения уровня нагрузки. Обычно архитектурно это распределенные и виртуализированные многоуровневые системы с добавленными механизмами автоматического масштабирования, которые специально разработаны для снижения специфичных

рисков производительности подобных архитектур. Однако в таких системах существуют риски сбоев из-за неверной настройки механизмов автоматического масштабирования во время первоначальной настройки системы или её последующих обновлений.

Клиент-серверные системы
Это системы, работающие на стороне клиента и взаимодействующие по сети с серверной частью, запущенной на единственном, многоуровневом, распределенном или динамическом серверах. К коду, запущенному на клиенте, применимы риски систем на базе единственного компьютера, а к серверной части - упомянутые выше проблемы серверов соответствующих архитектур. На производительность клиент-серверной архитектуры могут ещё повлиять возможные проблемы со скоростью и надежностью сетевого соединения, рисков перегрузок сети в точке подключения клиента (например, через общедоступную сеть Wi-Fi), а также брандмауэры, системы анализа сетевых пакетов и балансировки нагрузки.

Мобильные приложения
Это приложения, запущенное на смартфоне, планшете или другом мобильном устройстве. Оно подвержено тем же рискам, что и клиент-серверные системы и веб-приложения. Помимо этого, проблемы с производительностью на мобильном устройстве могут возникать из-за ограниченных ресурсов, их непостоянной доступности, проблем с доступом к сети (на что влияет местоположение, время работы от батареи, уровень заряда, доступная память устройства и степень его нагрева). Приложения, использующие датчики или модули связи устройства (например, акселерометры или Bluetooth), могут испытывать проблемы из-за медленных потоков данных из этих источников. Наконец, мобильные приложения часто функционально связаны с другими установленными на устройстве приложениями и удаленными веб-сервисами, которые потенциально так же могут стать узким местом.

Встроенные системы работы в реальном времени
Это системы, которые встроены в окружающие нас вещи: автомобили (развлекательные системы и интеллектуальные системы торможения), лифты, светофоры, системы обогрева, вентиляции и кондиционирования воздуха и т.д. Они подвержены большинству рисков мобильных устройств, включая (с повышенной вероятностью) проблемы с подключением к сети, поскольку эти устройства подключены к Интернету. Основное отличие в том, что заторможенность мобильной видеоигры обычно не представляет опасности для пользователя, в то время как подобное замедление в работе тормозной системы автомобиля может привести к катастрофе.

Системы на основе мейнфреймов
В таких системах в Центрах Обработки Данных работают приложения (зачастую разработанные очень давно), обслуживающие важные задачи бизнеса, иногда с использованием пакетной обработки. Большинство из таких приложений довольно предсказуемы и быстры, когда используются так, как было предусмотрено первоначально. Но теперь они часто доступны через API, веб-интерфейс или через прямой доступ к базе данных, что может привести к неожиданным всплескам нагрузки, которые влияют на работу установленных приложений.

Обратите внимание, что любое отдельно взятое приложение или система могут включать в себя две или более из перечисленных выше архитектур. Тогда к этому приложению или системе будут применяться все соответствующие совокупные риски. Фактически, учитывая развитие интернета вещей и взрывное распространение мобильных приложений (две области, где тесная взаимосвязанность и доступ к сети являются правилом) возможно, что в приложении в какой-то форме присутствуют все архитектуры, и, следовательно, могут учитываться все риски.

Хотя архитектура системы и является важным техническим решением с глубоким влиянием на риски производительности, на риски влияют (и создают их) и другие технические факторы. Например, утечки памяти чаще встречаются в языках, позволяющих напрямую управлять памятью кучи, например, в C и C++, а у реляционных и не реляционных баз данных разные проблемы с производительностью. Такие факторы распространяются вплоть до уровня проектирования отдельных функций или методов программы (например, при выборе рекурсивного алгоритма вместо итеративного). Способность тестировщика знать или даже влиять на такие решения будет разной,

в зависимости от ролей и обязанностей тестирования в рамках организации и жизненного цикла системы.

3.3 Риски производительности в течение жизненного цикла разработки программного обеспечения

Процесс анализа рисков качества программного продукта в целом обсуждается в различных программах ISTQB (например, см. [ISTQB_FL_SYL] и [ISTQB_ALTM_SYL]). Можно также найти обсуждения конкретных рисков и соображений, связанных с конкретными характеристиками качества (например, [ISTQB_UT_SYL]) или рассматривающих риски с точки зрения бизнеса или технической точки зрения (например, см. [ISTQB_ALTA_SYL] и [ISTQB_ALTTA_SYL], соответственно). В этом разделе основное внимание уделяется рискам производительности, связанным с качеством продукции, включая способы изменения процесса, участников и соображений.

Для рисков производительности, связанных с качеством продукта, основной процесс в целом одинаковый:

1. Определите риски, влияющие на качество продукта, особенно на такие характеристики, как поведение во времени, использование ресурсов и мощность.
2. Оцените выявленные риски, рассмотрев соответствующие виды архитектур (см. раздел 3.2) и оценив общий уровень для каждого выявленного риска с точки зрения вероятности и степени влияния на систему, используя четко определенные критерии.
3. Примите необходимые меры для снижения каждого выявленного риска на основе его характера и уровня.
4. Регулярно пересматривайте риски для их своевременного устранения до выпуска продукта.

Аналогично анализу рисков качества в целом, в процесс должны быть включены представители как бизнеса, так и технической стороны. В случае анализа рисков, связанных с производительностью, в состав представителей бизнеса должны быть включены участники с чётким пониманием того, как проблемы с производительностью в производственной среде будут влиять на клиентов, внутренних пользователей продукта, бизнес в целом, и на другие заинтересованные стороны. Заинтересованные лица со стороны бизнеса должны осознавать, что предполагаемый способ использования продукта, его влияние на бизнес, общество или безопасность, потенциальный финансовый и / или репутационный ущерб, возможная гражданская или уголовная ответственность, и тому подобные факторы влияют на риски ведения бизнеса, создают новые риски и влияют на масштаб последствий от сбоев.

Кроме того, состав заинтересованных лиц с технической стороны должен включать тех, кто детально понимает влияние выбранных требований, архитектуры, проектирования и реализации на производительность. Эти участники должны осознавать, что решения по архитектуре, проектированию и реализации влияют на риски производительности с технической точки зрения, создавая риски и влияя на вероятность возникновения дефектов.

Выбранный процесс анализа рисков должен иметь необходимый уровень формальности и строгости. Для рисков, связанных с производительностью, особенно важно, чтобы процесс анализа рисков начинался как можно раньше и повторялся на регулярной основе. Другими словами, тестировщик не должен рассчитывать на тестирование производительности, проводимое только после этапов системного и интеграционного тестирования. Многие проекты, особенно крупные и сложные, типа системы систем, столкнулись с неприятными сюрпризами из-за позднего обнаружения дефектов производительности и других подобных дефектов, которые в конечном итоге появились из-за требований, а также проектных, архитектурных и внедренческих решений, принятых на раннем этапе проекта. Таким образом, акцент должен делаться на итеративном подходе к выявлению, оценке, смягчению последствий и управлению рисками на протяжении всего жизненного цикла разработки программного обеспечения.

Например, если большие объемы данных обрабатываются в реляционной БД, медленная производительность операций объединения «многие-ко-многим» (из-за неэффективной структуры базы данных) может проявляться только во время динамического тестирования с использованием больших наборов тестовых данных, подобных тем, которые используются на этапе системного тестирования. Тем не менее, тщательный технический анализ, в который вовлечены опытные инженеры БД, может предсказать проблемы заранее, до внедрения. После такого анализа, повторяемого итеративно, риски идентифицируются и оцениваются снова.

Вдобавок задачи по снижению рисков и управлению ими должны охватывать и влиять на весь процесс разработки программного обеспечения, а не только на динамическое тестирование. Например, когда критические показатели, связанные с производительностью (такие, как ожидаемое количество транзакций или одновременно работающих пользователей), могут быть неизвестны на раннем этапе проекта, важно, чтобы решения проектирования и архитектуры допускали возможность широкого масштабирования (например, используя облачные вычислительные ресурсы, выделяемые по необходимости). Это позволяет заранее принимать решения о смягчении рисков.
Грамотное проектирование систем с учетом аспектов производительности может помочь проектным группам избежать позднего обнаружения критических дефектов производительности на более высоких уровнях тестирования, таких как системное интеграционное или пользовательское приемочное тестирование. Дефекты производительности, обнаруженные на поздней стадии, могут быть чрезвычайно дорогостоящими и даже привести к отмене всего проекта.

Невозможно полностью избавиться от рисков, связанных с производительностью (как, впрочем, и от любого другого типа рисков качества), поскольку в рабочей среде всегда существует определенный риск сбоя, связанный с производительностью. Поэтому процесс управления рисками должен предоставлять реалистичные и конкретные оценки остаточного уровня риска всем заинтересованным лицам, участвующим в процессе, как со стороны бизнеса, так и с технической стороны. Например, недостаточно просто сказать: «Да, все еще остаётся возможность того, что клиенты будут испытывать длительные задержки во время регистрации», поскольку эта фраза не дает представления о том, насколько были снижены риски или о том, какой уровень рисков остался. И наоборот, предоставление четкой информации о проценте клиентов, которые могут испытывать задержки, равные или превышающие конкретные пороговые значения, поможет людям осознать текущее состояние.

3.4 Активности тестирования производительности

Мероприятия по тестированию производительности могут быть организованы и выполняться по-разному, в зависимости от типа жизненного цикла разработки программного обеспечения.

Последовательные модели разработки
Практика тестирования производительности при последовательных моделях разработки, в идеале, заключается в том, чтобы включить критерии производительности в состав критериев приёмки, которые задаются на старте проекта. Тестирование производительности, для усиления тестирования, учитывающего жизненный цикл, должно проводиться на протяжении всего жизненного цикла разработки программного обеспечения. По мере продвижения проекта каждый последующий тест производительности должен основываться на активностях, выполненных на предыдущих шагах, как показано в приведенных ниже активностях.

- Концепция - убедитесь, что цели производительности системы заданы как критерии успеха и приёмки проекта.
- Требования - убедитесь, что требования к производительности определены и соответствуют нуждам заинтересованных лиц.
- Анализ и проектирование - убедитесь, что проектирование проекта отражает требования к производительности.
- Кодирование/Реализация - убедитесь, что код эффективен и соответствует требованиям и проектированию с точки зрения производительности.
- Компонентное тестирование - проведите тестирование производительности на уровне компонентов.

- Тестирование интеграции компонентов - проведите тестирование производительности на уровне интеграции компонентов.
- Системное тестирование - проведите тестирование производительности на уровне системы, включая аппаратное и программное обеспечение, процедуры и данные, которые близки к производственным. Системные интерфейсы могут быть смоделированы при условии, что они дают истинное представление о производительности
- Системное интеграционное тестирование – проведите тестирование всей системы, воспроизводящей продуктивное окружение.
- Приёмочное тестирование - проверьте, что производительность системы соответствует изначально заявленным нуждам пользователя и критериям приёмки.

Итеративные и инкрементальные модели разработки
В подобных моделях разработки, например, в гибких методологиях, тестирование производительности также рассматривается как итеративная и инкрементальная активность (см. [ISTQB_FL_AT]). Тестирование производительности может выполняться как часть первой итерации или как отдельная итерация, полностью посвященная тестированию производительности. Однако, при использовании этих моделей жизненного цикла активности тестирования производительности могут выполняться отдельной, специально выделенной командой.

Непрерывная интеграция (Continuous Integration, CI) обычно выполняется в итеративном и инкрементальном жизненных циклах разработки программного обеспечения, что облегчает выполнение тестов с высокой степенью автоматизации. Чаще всего CI используется для проведения регрессионного тестирования и обеспечения стабильности каждой сборки. Тестирование производительности может быть частью автоматических тестов, выполненных в CI, если тесты подготовлены для выполнения на этапе сборки. Однако, в отличие от автоматизированных функциональных тестов, в тестах производительности встречаются такие задачи как:

- Настройка среды тестирования производительности. Для этого часто требуется тестовая среда, выделяющая ресурсы по требованию, например, на основе облачной платформы.
- Выявление тестов производительности, которые можно автоматизировать в CI. Поскольку время работы тестов CI ограничено, тесты производительности CI могут быть подмножеством более объёмных тестов производительности, которые проводит отдельная команда в другое время цикла разработки.
- Создание тестов производительности для CI. Тесты производительности как часть CI используются в основном для того, чтобы проследить, что изменения в продукте не оказывают негативного влияния на производительность. Изменения, внесенные в новую сборку, могут потребовать создания новых тестов производительности.
- Выполнение тестов производительности частей приложения или системы. Для этого часто требуются инструменты и тестовые среды, способные к проведению ограниченного по времени тестирования производительности, в том числе дающие возможность выбора подмножества запускаемых тестов.

Тестирование производительности в итеративном и инкрементальном жизненных циклах разработки программного обеспечения также может иметь свои собственные жизненные циклы:

- Планирование выпуска. В этой активности тестирование производительности рассматривается с точки зрения всех итераций в выпуске, начиная с первой и заканчивая последней. Выявляются и оцениваются риски производительности, планируются меры по их снижению. Тут же часто планируется окончательное тестирование производительности до выпуска приложения.
- Планирование итерации. Тестирование производительности может выполняться как в процессе итерации, так и после её завершения.
- Создание пользовательской истории. На пользовательских историях часто строятся требования к производительности в гибких методологиях. Подобные истории учитывают конкретные параметры производительности, описанные в соответствующих критериях приёмки. Они часто упоминаются как нефункциональные пользовательские истории. Риски производительности оцениваются более подробно для каждой пользовательской истории.

- Разработка тестов производительности. Для разработки тестов используются требования производительности и критерии, которые описаны в конкретных пользовательских историях (см. раздел 4.2)
- Кодирование/Реализация. Во время кодирования тестирование производительности может выполняться на уровне компонентов. Таким примером может служить настройка алгоритмов для оптимальной производительности.
- Тестирование/оценка. Хотя тестирование, как правило, тесно связано с разработкой, тестирование производительности может выполняться как отдельный процесс, в зависимости от объема и целей тестирования производительности во время итерации. Например, если целью тестирования производительности является проверка производительности итерации как завершенного набора пользовательских историй, потребуется более широкий спектр тестирования производительности, чем при тестировании производительности одной пользовательской истории. Это может быть запланировано в отдельной итерации тестирования производительности.
- Поставка. Поскольку поставка разворачивает приложение в производственной среде, необходимо осуществлять мониторинг, чтобы определить, достигает ли приложение желаемых уровней производительности при фактическом использовании.

Коммерческий коробочный программный продукт (COTS) и другие модели поставщиков / покупателей

Многие организации не разрабатывают сами приложения и системы, а приобретают их или используют готовые проекты с открытым исходным кодом. В таких моделях поставщиков / покупателей производительность является важным фактором, требующим тестирования как со стороны продавца (поставщика / разработчика), так и покупателя (клиента).

Вне зависимости от источника происхождения приложения, клиент часто обязан подтвердить, что производительность соответствует его требованиям. В случае программного обеспечения, разработанного сторонней компанией по заказу, требования к производительности и соответствующие критерии приемки должны быть указаны как часть контракта между сторонами. В случае COTS-приложений клиент сам несет ответственность за тестирование производительности продукта до развертывания в рабочей среде.

4. Цели тестирования производительности – 475 минут

Ключевые слова
Способность системы к одновременной работе, профиль нагрузки, создание нагрузки, операционный профиль, плавное снижение нагрузки (ramp down), плавное повышение нагрузки (ramp up), система систем, пропускная способность системы, план тестирования, имитация паузы на принятие решения и выполнения действия пользователем (think-time), виртуальный пользователь

Цели обучения

4.1 Планирование

PTFL-4.1.1 (K4) Установить цели тестирования производительности на основе значимой информации
PTFL-4.1.2 (K4) Описать план тестирования производительности, учитывающий цели производительности для заданного проекта
PTFL-4.1.3 (K4) Создать презентацию, позволяющую различным группам заинтересованных лиц понять обоснование запланированного тестирования производительности

4.2 Анализ, разработка и реализация

PTFL-4.2.1 (K2) Привести примеры протоколов, типичных для тестирования производительности
PTFL-4.2.2 (K2) Понять понятие транзакций в тестировании производительности
PTFL-4.2.3 (K4) Проанализировать операционные профили, используемые в системе
PTFL-4.2.4 (K4) Создать профили нагрузки на основе операционных профилей для заданных целей тестирования производительности
PTFL-4.2.5 (K4) Проанализировать пропускную способность и готовность системы к одновременной работе множественных пользователей при разработке тестов производительности
PTFL-4.2.6 (K2) Понять базовую структуру скрипта тестирования производительности
PTFL-4.2.7 (K3) Выполнить скрипты тестирования производительности в соответствии с планом тестирования и профилями нагрузки
PTFL-4.2.8 (K2) Понять активности подготовки к выполнению теста производительности

4.3 Выполнение тестирования

PTFL-4.3.1 (K2) Понять основные активности во время выполнения скриптов тестирования производительности

4.4 Анализ результатов и отчетность

PTFL-4.4.1 (K4) Проанализировать и создать отчет о тестировании производительности и полученных результатах и выводах

4.1 Планирование

4.1.1 Постановка целей тестирования производительности

В качестве заинтересованных лиц могут выступать как пользователи, так и представители бизнеса или носители технологий. Различные группы заинтересованных лиц могут иметь различные цели, связанные с тестированием производительности. Заинтересованные лица устанавливают используемую терминологию, цели и критерии их достижения.

Цели тестирования производительности должны отражать взгляды этих различных типов заинтересованных лиц. Хорошей практикой является разграничение пользовательских и технологических целей. Пользовательские цели сосредоточены в первую очередь на удовлетворении целей конечного пользователя и бизнеса. Как правило, пользователей мало заботят типы объектов и то, как продукт поставляется. Они хотят лишь иметь возможность выполнять необходимые им действия.

С другой стороны, технологические цели сосредоточены на эксплуатационных аспектах и способности системы к масштабированию, а также определению условий, при которых проявляется деградация (резкое снижение) производительности системы.

Главные цели тестирования производительности включают определение потенциальных рисков, поиск возможностей для улучшения работы системы, и выявление необходимых изменений.

При сборе информации от заинтересованных лиц должны быть получены ответы на следующие вопросы:

- Какие транзакции должны будут выполняться во время тестирования производительности, и какое время отклика ожидается?
- Какие метрики должны быть собраны (например, использование памяти, пропускная способность сети) и какие значения ожидаются?
- Какие улучшения производительности ожидаются при проведении текущего тестирования по сравнению с предыдущими итерациями?

4.1.2 План тестирования производительности

- План тестирования производительности (ПТП) — это документ, который создается до начала тестирования производительности, и в ходе его проведения может обновляться. Общий план тестирования, который, так же, как и ПТП, содержит информацию о планировании тестирования, должен ссылаться на ПТП ([ISTQB_FL_SYL] ISTQB Foundation Level (Core) Syllabus, Version 2018)

В ПТП должна содержаться следующая информация:

Цели
Цели ПТП описывают задачи, стратегии и методы тестирования производительности. Они определяют количественный ответ на вопрос о пригодности и готовности системы работать под нагрузкой.

Цели тестирования
Здесь для каждого типа заинтересованных лиц приводится полный список целей тестирования, которые должны быть достигнуты тестируемой системой (см. раздел 4.1.1).

Обзор системы
Краткое описание тестируемой системы предоставит контекст измерения параметров производительности. Данный обзор должен включать общее описание функциональности, тестируемой под нагрузкой.

Типы проводимого тестирования производительности
Перечисляются типы тестирования производительности (см. раздел 1.2) вместе с описанием назначения каждого из типов.

Критерии приемки
Тестирование производительности предназначено для определения скорости отклика, пропускной способности, надежности и/или масштабируемости системы при заданной нагрузке. В целом, время отклика представляет важность для пользователей, пропускная способность относится к сфере интересов бизнеса, а использование ресурсов представляет интерес с технической точки зрения. Критерии приемки должны представлять совокупность всех значимых измерений и основываться на:

- Полном перечне целей тестирования производительности
- Соглашениях об уровне обслуживания (SLA)
- Эталонных показателях – наборе значений метрик, полученном в предыдущем тестировании и используемом для сравнения с результатами метрик в ходе текущего тестирования. Они позволяют продемонстрировать конкретные улучшения

производительности и/или подтвердить достижение критериев приемки тестирования производительности. Возможно, потребуется сначала создать набор эталонных показателей на основе маскированных значений базы данных, когда это представляется возможным.

Тестовые данные

Тестовые данные представляют собой широкий круг данных, которые должны быть определены перед тестированием производительности. Они могут включать:

- Учетные данные пользователей (например, учетные записи, позволяющие выполнить одновременную авторизацию в системе)
- Данные, вводимые пользователями (например, данные, которые пользователь должен ввести в приложении для выполнения бизнес-процесса)
- Значения из базы данных (например, базы данных, предварительно наполненной информацией, используемой при тестировании)

Процесс создания тестовых данных должен учитывать следующие аспекты:

- возможность извлечения необходимых данных из производственной рабочей среды
- их импорт в тестируемую систему
- создание новых тестовых данных
- создание резервной копии тестовых данных, которая может быть использована для их восстановления при последующих итерациях тестирования
- маскирование и обезличивание данных. Данная практика используется во время работы с тестовыми данными, полученными из производственной рабочей среды и содержащими персональную информацию, и является обязательной в соответствии с «Общим регламентом по защите данных» (GDPR). Однако, при тестировании производительности маскирование данных несет риск, связанный с тем, что такие данные будут отличаться по своим характеристикам от используемых в производственной рабочей среде.

Конфигурация системы

Раздел ПТП о конфигурации системы включает в себя следующую техническую информацию:

- Описание специфической архитектуры системы, включая серверы (например, web-серверы, серверы баз данных, балансировщики нагрузки)
- Описание каждого из слоев архитектуры (например, слои представления, бизнес-логики, данных)
- Специфичные особенности оборудования (например, ядра ЦПУ, ОЗУ, диски SSD, диски HDD) с указанием версий
- Специфические особенности программного обеспечения (например, приложений, операционных систем, баз данных, сторонних сервисов) с указанием версий
- Внешние системы, взаимодействующие с тестируемой системой, а также их конфигурации (например, система электронной коммерции с интеграцией в NetSuite)
- Номер сборки/версии тестируемой системы

Тестовое окружение

Тестовое окружение часто представляет собой изолированную среду, аналогичную продуктивной, но в меньших масштабах. Данный раздел ПТП должен содержать информацию о том, каким образом результаты тестирования производительности будут экстраполированы на продуктивную среду. Для некоторых систем единственным возможным вариантом тестирования является продуктивная среда. Риски тестирования производительности в подобных случаях должны быть подвергнуты тщательному обсуждению.

Иногда инструменты тестирования расположены вне тестовой среды, из-за чего для их взаимодействия с компонентами системы могут потребоваться особые права доступа. Данный аспект необходимо принять во внимание при конфигурировании тестовой среды.

Тесты производительности также могут проводиться с отдельным компонентом системы, который способен работать без других компонентов. Зачастую это дешевле, чем тестирование всей системы, и такие тесты могут проводиться сразу после разработки данного компонента.

Инструменты тестирования
Данный раздел включает описание инструментов тестирования (в том числе их версий), которые будут использоваться при написании скриптов, выполнении и мониторинге тестирования производительности (см. главу 5). Обычно этот список содержит:

- Инструмент(ы), эмулирующие транзакции пользователей
- Инструменты, обеспечивающие создание нагрузки из нескольких точек архитектуры системы (точек присутствия)
- Инструменты мониторинга производительности системы, включая описанные ранее в пункте «Конфигурация системы»

Профили
Операционные профили представляют собой повторяющиеся последовательности действий при конкретном использовании системы. Совокупность таких операционных профилей представляет собой профиль нагрузки (часто именуемый сценарием). Для большей информации о профилях см. раздел 4.2.3.

Значимые метрики
При тестировании производительности есть возможность проводить значительное количество измерений и собирать большое количество метрик (см. главу 2). Однако, выполнение избыточного количества замеров может затруднить анализ, а также негативно отразиться на фактической производительности приложения. По этим причинам важно определить замеры и метрики, наиболее значимые для достижения целей тестирования производительности.

Следующая таблица, более подробно рассмотренная в разделе 4.4, показывает набор метрик, наиболее характерный для тестирования и мониторинга производительности. В конкретном проекте для этих метрик должны быть заданы целевые значения.

Метрики производительности	
Тип	**Метрика**
Статус виртуального пользователя	Тест пройден, количество виртуальных пользователей Тест не пройден, количество виртуальных пользователей
Время отклика транзакции	Минимальное значение Максимальное значение Среднее значение 90% перцентиль
Транзакций в секунду	Выполнено успешно, транзакций/сек Выполнено неуспешно, транзакций/сек Суммарное количество, транзакций/сек
Запросы (например, к базе данных или на web-сервер)	Запросов в секунду ▪ Минимальное значение ▪ Максимальное значение ▪ Среднее значение ▪ Итого
Пропускная способность	Бит в секунду ▪ Минимальное значение ▪ Максимальное значение ▪ Среднее значение ▪ Итого
HTTP ответов в секунду	Ответов в секунду ▪ Минимальное значение ▪ Максимальное значение ▪ Среднее значение ▪ Итого Коды состояния HTTP-сервера

Мониторинг производительности	
Тип	**Метрика**
Утилизация центрального процессора	% использования центрального процессора
Использование памяти	% использования доступной памяти

Риски
Риски могут включать как ограничения самого тестирования производительности, так и факторы, не относящиеся к нему непосредственно (например, невозможность эмулировать интерфейсы внешних систем, недостаточную нагрузку, невозможность мониторинга серверов). Риски также могут быть вызваны ограничениями тестового окружения (например, недостаточное наполнение данными, уменьшенный масштаб тестового окружения по отношению к производственной рабочей среде). Более подробная информация о видах рисков размещена в разделах 3.2 и 3.3.

4.1.3 Коммуникация при тестировании производительности

Тестировщик должен быть способен доходчиво объяснить всем заинтересованным лицам подход к тестированию производительности и те действия, которые необходимо осуществить (подробно описанные в плане тестирования производительности). Заинтересованные лица, которым может быть адресовано подобное объяснение, могут сильно различаться в зависимости от того, на чем

больше сосредоточено их внимание – от интересов бизнеса и пользователей до интересов технологий и разработки.

Заинтересованные лица со стороны бизнеса
При коммуникации с лицами со стороны бизнеса необходимо учитывать следующие факторы:

- Заинтересованных лиц со стороны бизнеса не сильно интересуют различия между функциональными и нефункциональными характеристиками качества.
- Технические вопросы, касающиеся инструментов тестирования, скриптов и создания нагрузки, имеют второстепенное значение.
- Должно быть четко установлена связь между рисками продукта и целями тестирования производительности.
- Заинтересованные лица должны быть ясно осведомлены о соотношении стоимости планируемого тестирования производительности и тем, насколько применимыми будут его результаты для производственной рабочей среды.
- Должна быть обсуждена повторяемость планируемых тестов. Тяжело ли будет повторить данный тест, или это потребует минимальных усилий?
- Должны быть обсуждены риски проекта. Они включают ограничения и зависимости, касающиеся организации тестов, требований инфраструктуры (например, оборудования, инструментов, данных, тестового окружения, ресурсов) и зависимости от ответственного персонала.
- Необходимо обеспечить, чтобы заинтересованные лица со стороны бизнеса имели общее представление о действиях, предпринимаемых в рамках нагрузочного тестирования, а также об основном плане работ, содержащем стоимость, расписание и контрольные точки (см. разделы 4.2 и 4.3).

Заинтересованные лица со стороны технологий
При общении с заинтересованными лицами со стороны технологий необходимо учитывать следующие факторы:

- Должны быть выяснены планируемый подход к созданию необходимых профилей нагрузки и ожидаемая вовлеченность заинтересованных лиц.
- Должны быть детально объяснены этапы подготовки и выполнения тестов производительности с целью отражения взаимосвязи процесса тестирования с архитектурными рисками.
- Должны быть обсуждены действия, необходимые для повторяемости тестов. Они могут включать как организационные аспекты (например, участие ответственных сотрудников), так и технические.
- В случаях, когда тестовые окружения являются используемыми совместно, должно быть выработано согласованное расписание, чтобы быть уверенным, что результаты теста не подвергались негативному воздействию.
- Должно быть обсуждено и согласовано ослабление потенциального воздействия на работу реальных пользователей в случаях необходимости проведения тестирования производительности на продуктивной среде.
- Заинтересованные лица со стороны технологий должны четко представлять, выполнение каких задач и в какое время от них ожидается.

4.2 Анализ, разработка и реализация

4.2.1 Типичные коммуникационные протоколы

Коммуникационные протоколы определяют набор правил общения между компьютерами и системами. Для разработки надлежащих тестов, нацеленных на конкретные части системы, необходимо понимание протоколов.

Многие коммуникационные протоколы описаны уровнями Базовой эталонной модели взаимодействия открытых систем (Сетевой модели OSI, см. ISO/IEC 7498-1), но некоторые протоколы остаются за рамками этой модели. В тестировании производительности чаще всего

используются протоколы, соответствующие уровням с 5 (сеансовый уровень) по 7 (прикладной уровень). Широко распространены следующие протоколы:

- Для баз данных – ODBC, JDBC, проприетарные протоколы поставщиков баз данных;
- Для веб-приложений – HTTP, HTTPS, HTML;
- Для веб-служб – SOAP, REST

Вообще говоря, уровень модели OSI, на котором больше всего сфокусировано тестирование производительности, относится к уровню тестируемой архитектуры. Например, при тестировании некоторой низкоуровневой встраиваемой архитектуры в центре внимания будут более низкие номера уровней модели OSI.

Дополнительно в тестировании производительности используются:

- Сетевые протоколы – DNS, FTP, IMAP, LDAP, POP3, SMTP, Windows Sockets, CORBA;
- Для мобильных приложений – TruClient, SMP, MMS;
- Для систем с удалённым доступом – Citrix ICA, RTE;
- Для сервис-ориентированной архитектуры – MQSeries, веб-службы.

Понимание архитектуры системы в общих чертах необходимо, поскольку проведение тестов производительности возможно как для отдельных компонентов системы, так и для системы в целом при проведении сквозного тестирования. В традиционных двухуровневых приложениях, построенных по клиент-серверной модели, в качестве «клиента» выступает графический пользовательский интерфейс и основной пользовательский интерфейс, а «сервером» является база данных, работающая на сервере. Для доступа к базе данных подобные приложения используют такие протоколы, как ODBC. С развитием веб-приложений и многоуровневых архитектур, в обработке информации, отображаемой в браузере пользователя, может участвовать большое количество серверов.

Необходимо понимать, какой протокол следует использовать в зависимости от того, какой компонент подвергается тестированию. Например, если проводится сквозное тестирование с эмуляцией действий пользователя в браузере, применяется веб-протокол, например, HTTP/HTTPS. Таким образом можно абстрагироваться от взаимодействия с графическим интерфейсом и сконцентрироваться на взаимодействии и работе серверной части приложения.

4.2.2 Транзакции

Транзакцией называется совокупность действий, выполняемых системой с момента начала одного или нескольких процессов до момента его (их) завершения. Время, за которое транзакция завершилась (также называемое временем отклика), измеряется для оценки производительности системы. В ходе тестирования производительности эти измерения используются для идентификации компонентов системы, которые нуждаются в корректировке или оптимизации.

Эмулируемые транзакции могут включать паузы, чтобы более реалистично отражать время, которое требуется реальному пользователю для выполнения какого-либо действия (например, нажатия кнопки «Отправить»). Общее время транзакции составляют сумма времени отклика транзакции и таких пауз.

Данные о времени отклика транзакции, собранные в ходе теста производительности, демонстрируют изменения времени отклика в зависимости от объёма создаваемой нагрузки. Анализ времен отклика может не обнаружить ухудшения под нагрузкой, в то время как другие показания будут указывать на значительную деградацию. С помощью увеличения нагрузки и измерения времени выполнения транзакций становится возможным выявить причину ухудшения производительности.

Возможны также вложенные транзакции, что позволяет измерять время выполнения как отдельных действий, так и их совокупности. Это может быть использовано, например, при исследовании производительности системы принятия заказов через интернет. Тестировщика в таком случае интересует как длительность каждого отдельного шага процесса совершения заказа (например,

поиска товара, добавления товара в корзину, оплаты товара, подтверждения заказа), так и весь процесс от начала и до конца. Вложенность транзакций позволяет собрать все необходимые данные в одном тесте.

4.2.3 Определение операционных профилей

Операционные профили характеризуют различные шаблоны взаимодействия с приложением как пользователей, так и компонентов иных систем. Для одного приложения может быть определено несколько операционных профилей. Из их сочетания создаётся желаемый профиль нагрузки для достижения конкретных целей тестирования производительности (см. раздел 4.2.4).

В настоящем разделе рассматриваются следующие основные шаги определения операционных профилей:

1. Определение необходимых данных,
2. Сбор данных из одного или нескольких источников,
3. Анализ информации для конструирования профиля использования.

Определение необходимых данных
В случаях, когда с тестируемой системой взаимодействуют пользователи, для моделирования их операционных профилей (того, как они взаимодействуют с системой) необходимо собрать или дать примерную оценку следующих данных:

- Различные типы пользователей и их роли (например, обычный пользователь, зарегистрированный участник, администратор, пользовательская группа с привилегиями).
- Различные типичные задачи, которые выполняют эти пользователи/роли (например, просмотр информации, размещённой на веб-сайте, поиск определённого товара на веб-сайте, выполнение действий, характерных только для конкретной роли). Следует отметить, что такие задачи оптимально моделировать на высоком уровне абстракции (например, на уровне бизнес-процессов или важных пользовательских историй).
- Оценить количество пользователей для каждой роли или задачи в единицу времени в заданный период. Эта информация также будет полезна впоследствии при построении профилей нагрузки.

Сбор данных
Вышеописанные данные можно собрать из различных источников:

- Проведение интервью или семинаров с заинтересованными лицами, например, владельцами продукта, менеджерами по продажам и (потенциальными) конечными пользователями. Такое общение зачастую обнаруживает основные пользовательские операционные профили и даёт ответы на фундаментальный вопрос – «Для кого предназначено это приложение?»
- Функциональная спецификация и требования (при наличии) являются ценным источником информации об ожидаемых типичных случаях использования, что также может помочь идентифицировать типы пользователей и их операционные профили. В случаях, когда функциональная спецификация сформулирована в виде пользовательских историй, стандартный формат позволяет немедленно идентифицировать типы пользователей (то есть «как <тип пользователя>, я хочу <некий функционал>, чтобы <цель>). Аналогичным образом в диаграммах и описаниях случая использования на унифицированном языке моделирования UML определяется «актор».
- Исследование данных об использовании аналогичных приложений и сбор метрик также может помочь в идентификации типов пользователей и формировании первоначальных ожиданий числа пользователей. Рекомендуется использовать автоматически собираемые

данные (например, в инструменте администрирования веб-мастера), в частности, логи мониторинга и собираемые данные об использовании эксплуатируемой в настоящее время системы, на смену которой придёт новая.

- Наблюдение за поведением пользователей при выполнении конкретных задач в приложении может навести на типы операционных профилей, которые необходимо моделировать для тестирования производительности. Рекомендуется координировать эту деятельность с планируемыми тестами практичности (особенно если в распоряжении имеется лаборатория практичности).

Построение операционных профилей
Процесс идентификации и построения операционных профилей состоит из следующих этапов:

- Принимается подход «сверху вниз». Изначально определяются сравнительно простые и широкие операционные профили, которые в дальнейшем уточняются и сужаются только в случае, если это необходимо для достижения целей тестирования производительности (см. раздел 4.1).
- Конкретные пользовательские профили могут быть выбраны в качестве особенно важных для тестирования производительности, если они включают задачи, которые выполняются часто, требуют выполнения критических (высокорисковых) или многочисленных транзакций между различными компонентами системы, или потенциально требуют передачи больших объёмов данных.
- Операционные профили анализируются и уточняются с участием основных заинтересованных лиц перед их использованием в создании профилей нагрузки (см. раздел 4.2.4).

Тестируемая система не всегда подвергается нагрузке от пользователей. Операционные профили могут также потребоваться при тестировании производительности систем следующих типов (обратите внимание, что приводится не исчерпывающий список):

Система пакетной обработки заданий
В данном случае основной интерес представляет пропускная способность такой системы (см. раздел 4.2.5) и её способность выполнить работу за заданное время. Функциональные профили составляются, исходя из типов обработки, которые требуются от процессов пакетной обработки. Например, операционные профили системы торговли ценными бумагами (как правило, такие системы включают обработку транзакций в интерактивном режиме наряду с пакетной обработкой) могут относиться к платёжным транзакциям, проверке учётных данных, проверке соответствия определённых видов биржевых транзакций требованиям законодательства. Каждый из этих операционных профилей предполагает различную обработку той или иной ценной бумаги заданием пакетной обработки. Вышеописанные шаги идентификации операционных профилей для интерактивных пользовательских систем могут быть использованы и в контексте пакетной обработки.

Система систем
Компоненты в многосистемной среде (которая может также являться встроенной) обрабатывают различные типы ввода данных от других систем или компонентов. В зависимости от сущности тестируемой системы, это может привести к необходимости моделирования нескольких различных операционных профилей для полного представления типов ввода от таких систем. Для этого может потребоваться проведение детального анализа (например, содержимого буферов памяти и очередей) совместно с системными архитекторами на основе спецификаций системы и интерфейсов.

4.2.4 Создание профилей нагрузки

Профиль нагрузки описывает активность, которой может подвергнуться тестируемый компонент или система в ходе эксплуатации. Он состоит из определённого числа сущностей, выполняющих действия, входящие в заданный операционный профиль, в течение конкретного времени. Когда

такие сущности олицетворяют пользователей, их обыкновенно называют «виртуальными пользователями».

Основная информация, необходимая для создания реалистичного и воспроизводимого профиля нагрузки:

- Цель тестирования производительности (например, оценить поведение системы при стрессовой нагрузке),
- Операционные профили, которые точно представляют типичные случаи использования системы (см. раздел 4.2.3),
- Известные проблемы, связанные с пропускной способностью или одновременной работой в системе (см. раздел 4.2.5),
- Количество выполнений операций и их распределение во времени, необходимое для создания требуемой нагрузки для тестируемой системы. Типичные примеры:
  - Плавное повышение нагрузки: постепенное увеличение (например, путём добавления по одному виртуальному пользователю раз в минуту)
  - Плавное снижение нагрузки: постепенное уменьшение нагрузки
  - Резкое изменение нагрузки: моментальное изменение нагрузки (например, путём добавления 100 виртуальных пользователей раз в 5 минут).
  - Заданное распределение (например, объём нагрузки эмулирует дневной или сезонный цикл).

Следующий пример демонстрирует построение профиля нагрузки для создания стрессовых условий (на уровне ожидаемой максимальной нагрузки, которую может выдержать система, или свыше него) для тестируемой системы.

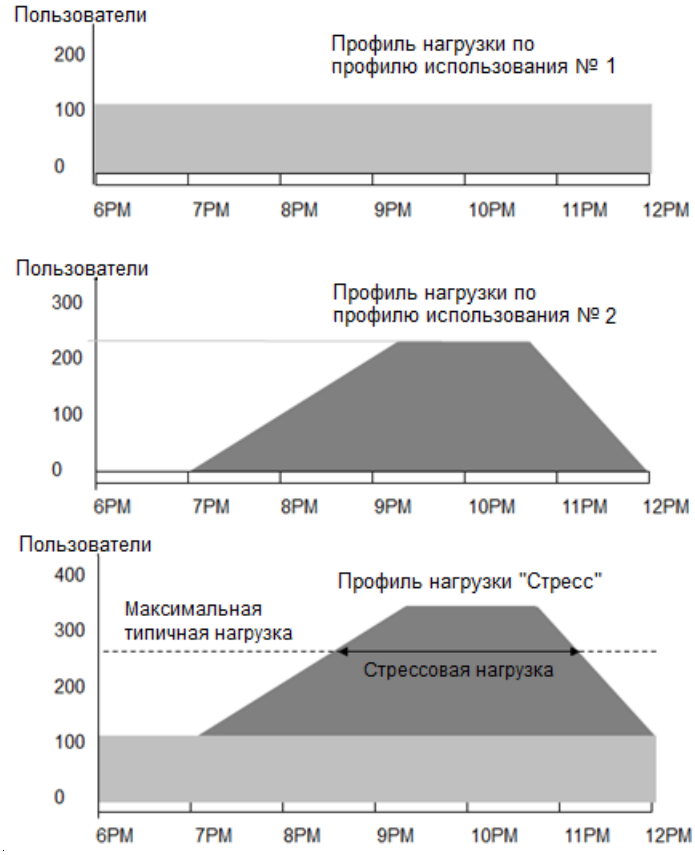

Рисунок 1: Пример построения «стрессового» профиля нагрузки.

В верхней части рисунка изображён профиль нагрузки, состоящий из одной ступени, на которой ввод совершают 100 виртуальных пользователей. Эти виртуальные пользователи выполняют действия, определённые в операционном профиле № 1 на протяжении всего теста. Такая картина типична для профилей нагрузки, представляющих фоновую нагрузку.

В средней части рисунка изображён профиль нагрузки, который состоит из разгона (ramp up) до 220 виртуальных пользователей, которые продолжают работать на протяжении двух часов, после чего постепенно выводятся (ramp down). Каждый из этих виртуальных пользователей совершает действия, определённые в операционном профиле № 2.

В нижней части рисунка изображен профиль нагрузки, представляющий собой сочетание двух предыдущих. Тестируемая система на протяжении трёх часов испытывает стрессовую нагрузку. Другие примеры можно найти в [Bath14].

4.2.5 Анализ пропускной способности и одновременной работы

Важно отличать разные аспекты рабочей нагрузки: пропускную способность и способность к одновременной работе. Для корректного моделирования операционных профилей и профилей нагрузки, следует принимать во внимание оба этих аспекта.

Пропускная способность системы

Пропускная способность измеряется количеством транзакций того или иного типа, которые система обрабатывает за единицу времени. Например, количество заказов в час или число HTTP-запросов в секунду. Пропускную способность системы следует отличать от полосы пропускания сети, которая определяет объём данных, которые могут быть переданы через сетевое соединение за единицу времени (см. раздел 2.1).

Пропускная способность определяет объём нагрузки на систему. К сожалению, зачастую нагрузка систем, работающих в интерактивном режиме, определяется количеством пользователей (обработкой одновременно выполняемых заданий), а не пропускной способностью. Часто это число проще найти, и в некоторых случаях уже это может быть причиной подобного подхода, а в других случаях это обусловлено тем, как инструменты нагрузочного тестирования определяют объём создаваемой нагрузки. Без указания операционного профиля – что именно делает каждый пользователь, и с какой интенсивностью (то есть пропускной способности, собственно, одного пользователя) – количество пользователей не подходит в качестве меры объёма нагрузки. Например, если 500 пользователей каждую минуту выполняют по небольшому запросу каждый, нагрузка составит 30 000 запросов в час. Если те же 500 пользователей выполняют такой же запрос, но лишь раз в час каждый, нагрузка составит 500 запросов в час. Таким образом, при одном и том же количестве пользователей может иметь место шестидесятикратная разница в нагрузке и, соответственно, не менее чем шестидесятикратная разница в требованиях к аппаратной платформе системы.

Моделирование рабочей нагрузки обыкновенно производится с учётом количества используемых виртуальных пользователей (потоков выполнения) и длительности пауз (think-time - времени ожидания между выполнением пользовательских действий). Однако пропускная способность системы также определяется временем обработки, и это время может увеличиться с ростом нагрузки.

Пропускная способность системы = [количество виртуальных пользователей] / ([время обработки] + [длительность пауз])

По мере роста времени обработки может существенно ухудшиться пропускная способность системы, даже если ничего больше не изменится.

Пропускная способность системы особенно важна в тестировании систем пакетной обработки заданий. В данном случае пропускная способность обыкновенно измеряется как количество транзакций, которые могут быть выполнены за конкретный промежуток времени (например, за время ночной пакетной обработки).

Способность к одновременной работе

Способность к одновременной работе представляет собой число одновременных потоков выполнения. Для интерактивных систем это может быть количество одновременных пользователей. Одновременная работа обычно эмулируется инструментами нагрузочного тестирования путём задания количества виртуальных пользователей.

Примечание авторов перевода. Здесь и далее, если не указано иное, под «одновременным» подразумеваются как собственно одновременные (параллельные), так и условно одновременные потоки выполнения (выполняющиеся поочередно на одном и том же процессоре в многопоточном режиме).

Способность к одновременной работе важна. Она означает количество параллельных сессий, каждая из которых может использовать свои ресурсы. Даже если пропускная способность неизменна, количество требуемых ресурсов может варьироваться в зависимости от способности к одновременной работе. Типичная тестируемая система в терминологии теории массового обслуживания представляет собой замкнутую систему, где количество пользователей системы жёстко задано (ограниченная популяция). Пока все пользователи ожидают ответа, в замкнутой системе не могут возникать новые пользователи. Многие публичные системы являются открытыми – новые пользователи появляются все время, даже когда все имеющиеся пользователи ожидают ответа системы.

4.2.6 Основная структура скриптов в тестах производительности

Скрипты тестов производительности предполагают эмуляцию деятельности пользователей или компонентов, которая создаёт нагрузку тестируемой системы (которой может являться как система целиком, так и один из ее компонентов). Они отправляют запросы к серверу в нужном порядке и с нужной интенсивностью.

Выбор способа создания скриптов для теста производительности зависит от используемого подхода к созданию нагрузки (см. раздел 4.1).

- Традиционно для этого записывается обмен данными между клиентом и системой или ее компонентом на уровне протокола и затем, после параметризации и документирования, воспроизводится. С помощью параметризации скрипт становится масштабируемым, параметризованный скрипт легче поддерживать, но сама задача параметризации может потребовать большого труда.
- Запись на уровне графического интерфейса обычно состоит в перехвате с помощью инструмента тестирования действий единичного пользователя в графическом интерфейсе, а затем воспроизведении полученного скрипта посредством инструмента создания нагрузки, имитируя работу множества пользователей.
- Программировать скрипты можно с использованием протокольных запросов (например, HTTP-запросов), действий в графическом пользовательском интерфейсе или вызовов API. При программировании скриптов необходимо определить точную последовательность запросов, отправляемых системе, и получаемых от неё данных, что может оказаться нетривиальной задачей.

Обычно скрипт представляет собой один или несколько модулей кода (написанного на широко используемом языке программирования с некоторыми расширениями или на специализированном языке) или объект, который может быть представлен пользователю как инструмент в графическом интерфейсе. В любом случае скрипт включает запросы к серверу, необходимые для создания нагрузки (например, HTTP-запросы) и дополнительную программную логику, определяющую, как именно эти запросы должны выполняться (в каком порядке, в какой момент времени, с какими параметрами, и что необходимо проверить при этом). Чем сложнее эта логика, тем уместнее использование мощного языка программирования.

Общая структура

Часто в скрипте можно выделить модуль инициализации (где готовится всё необходимое для основной части), основные модули, которые могут быть выполнены в ходе теста неоднократно, и завершающий модуль (в котором предпринимаются необходимые меры для надлежащего завершения теста).

Сбор данных
Для сбора данных о временах отклика в скрипт добавляются таймеры, с помощью которых измеряется длительность выполнения определённого запроса или комбинации запросов. Запросы, время выполнения которых измеряется, должны соответствовать логической единице работы – например, бизнес-транзакции по добавлению предмета в заказ или оформлению заказа.

Важно понимать, какие именно замеры выполняются: в случае скриптов на уровне протокола измеряется только время отклика сервера и сети, в то время как скрипты, работающие с графическим пользовательским интерфейсом, измеряют суммарное время работы (хотя что именно измеряется, зависит от используемой технологии).

Верификация результатов и обработка ошибок
Важной частью скрипта является верификация результатов и обработка ошибок. Даже в лучших инструментах нагрузочного тестирования по умолчанию в лучшем случае имеется лишь минимальная обработка ошибок (например, проверки кода ответа на HTTP-запрос), поэтому рекомендуется добавление дополнительных проверок содержимого получаемых ответов. Хорошая практика – проверять, что скрипт действительно выполняет требуемые действия, по косвенным признакам: например, можно проверить содержимое базы данных, чтобы убедиться в том, что в неё была внесена требуемая информация.

Также в скрипты может быть включена логика, устанавливающая правила того, когда и как именно совершаются запросы к серверу. Пример – точки синхронизации, которые устанавливаются указанием скрипту приостановить выполнение и ожидать наступления определённого события. Точки синхронизации могут использоваться для того, чтобы конкретные действия были выполнены синхронно, или для координации хода выполнения нескольких скриптов.

Скрипты для тестов производительности являются программным обеспечением, поэтому создание таких скриптов – проект по разработке программного обеспечения. В этот проект входит и этап тестирования, на котором проверяется, что скрипт работает ожидаемым образом с любыми надлежащими входными параметрами.

4.2.7 Разработка скриптов для тестов производительности

Скрипты для тестов производительности разрабатываются на основе плана нагрузочного тестирования и профилей нагрузки. Хотя технические детали реализации могут различаться в зависимости от выбранного подхода и инструмента (инструментов), в целом процесс остаётся неизменным. Скрипт разрабатывается с использованием интегрированной среды разработки (IDE) или редактора скриптов для эмуляции поведения пользователя или компонента системы. Обычно один скрипт соответствует конкретному операционному профилю (хотя зачастую возможно комбинирование нескольких операционных профилей в одном скрипте с помощью условных выражений).

После определения последовательности запросов скрипт автоматически записывается или программируется, в зависимости от подхода. Запись скрипта обычно гарантирует точную эмуляцию работы реальной системы, в то время как при программировании скрипта тестировщик полагается на знание корректной последовательности вызовов.

Если используется запись на уровне протокола, в большинстве случаев необходимым шагом после записи является замена всех записанных внутренних идентификаторов, определяющих контекст выполнения. Эти идентификаторы должны быть заменены на переменные, которые от запуска к запуску будут принимать корректные значения, извлекаемые из ответов на запросы к системе (например, идентификатор пользователя, полученный при входе в системе, который необходимо предъявлять во всех последующих транзакциях). Такую параметризацию скрипта также иногда называют «корреляцией». В этом контексте корреляция означает иное, нежели в статистике (где этим понятием обозначают отношение между двумя или несколькими значениями). Продвинутые инструменты нагрузочного тестирования могут частично выполнять корреляцию автоматически, и в некоторых случаях этот процесс может пройти без участия тестировщика, но в более сложных случаях может потребоваться корреляция вручную или определение правил корреляции. Неверная корреляция или её отсутствие – основная причина некорректного выполнения скриптов.

Запуск нескольких виртуальных пользователей, использующих учётные данные одного и того же пользователя и один и тот же набор входных данных (что происходит, например, при выполнении

записанного скрипта без внесения в него изменений помимо необходимой корреляции), вероятнее всего даст результаты, вводящие в заблуждение. Система может скопировать с диска в память для более быстрого доступа (закешировать) все требуемые данные и показать значительно более высокие результаты, чем в реальных условиях эксплуатации (где данные то и дело приходится считывать с диска). Использование одних и тех же учётных и/или входных данных также может вызвать проблемы, связанные с конкурентным выполнением кода (например, если данные блокируются при их изменении пользователем), и тогда полученные результаты будут значительно хуже, чем в производственной среде, поскольку при блокировании данных приложению необходимо ждать освобождения блокировки, прежде чем другой пользователь мог бы изменить те же самые данные.

Таким образом, скрипты и тестовая обвязка нуждаются в параметризации (то есть замене статичных или записанных данных на данные из подготовленного списка возможных вариантов), чтобы каждый виртуальный пользователь использовал надлежащий набор данных. «Надлежащий» означает, что данные разных виртуальных пользователей достаточно сильно различаются, чтобы избежать проблем с кешированием и конкурентным выполнением кода. Какие именно наборы данных являются надлежащими, определяется индивидуально на основе требований к системе, данным и тесту. Эта параметризация зависит от данных в системе и того, как система работает с этими данными, и обычно выполняется вручную, хотя многие инструменты помогают облегчить этот процесс.

В некоторых случаях параметризация необходима, чтобы тест можно было выполнить более одного раза – например, если в тесте создаётся заказ, и его наименование должно быть уникально. Если наименование заказа не параметризовано в скрипте, тест завершится с ошибкой, как только попытается создать заказ с уже существующим (записанным) наименованием.

Для соответствия с операционным профилем в скрипт необходимо добавить паузы, или в случае, если скрипт был записан, их длительность может быть необходимо изменить, для обеспечения необходимого количества запросов (нагрузки) как описано в разделе 4.2.5.

После создания скриптов, соответствующих разным операционным профилям, они комбинируются в тестовом сценарии, который отражает полный профиль нагрузки. Профиль нагрузки определяет, сколько виртуальных пользователей выполняют каждый из скриптов, когда, и с какими параметрами. Конкретные детали реализации зависят от того, какой инструмент нагрузочного тестирования используется, или от тестовой обвязки.

4.2.8 Подготовка к проведению теста производительности

Основные активности подготовки к проведению теста производительности включают:

- Настройку тестируемой системы,
- Развертывание окружения
- Настройку инструментов создания нагрузки и мониторинга, а также проверку сбора всей необходимой информации.

Важно, чтобы тестовая среда была настолько близка по характеристикам к производственной среде, насколько это возможно. Если полное соответствие невозможно, различия должны тщательно учитываться при проецировании результатов тестирования на производственную среду. В идеальном случае используется собственно производственная среда и реальные данные, однако даже тестирование в среде с более ограниченными ресурсами может помочь уменьшить ряд рисков, связанных с производительностью.

Необходимо учитывать, что производительность является нелинейной функцией среды, поэтому чем больше тестовая среда отличается от производственной, тем труднее точно прогнозировать производительность производственной среды. Ошибка прогноза и риски растут по мере увеличения различий между тестовой системой и системой в производственной среде.

Самые важные элементы тестовой среды – данные, аппаратная и программная конфигурация, и конфигурация сети. Объём и структура данных может оказать существенное влияние на результаты тестирования. Использование небольшой выборки или упрощенного набора данных для

тестирования производительности может привести к вводящим в заблуждение результатам, особенно если в производственной среде используется большое количество данных. Предсказать, как изменение объёма данных повлияет на производительность, без проведения сравнения затруднительно. Чем меньше тестовые данные отличаются от данных в производственной среде по объёму и структуре, тем надёжнее результаты тестирования.

Если в ходе тестирования данные создаются или изменяются, может быть необходимо восстановление изначального состояния данных перед следующим циклом тестирования для гарантии того, что система находится в надлежащем состоянии.

В случае, если какие-то части системы или какая-то часть данных недоступна для тестирования производительности по той или иной причине, необходимо реализовать обходное решение. Например, может быть разработана «заглушка» для эмуляции работы стороннего компонента, обрабатывающего транзакции по кредитным картам. Этот процесс часто называется «виртуализацией служб», и существуют специальные средства, призванные помочь с такими задачами. Настоятельно рекомендуется использовать подобные инструменты для изолирования тестируемой системы.

Развертывание среды может осуществляться различными способами. Например, может использоваться один из следующих подходов:

- Традиционные внутренние (и внешние) тестовые лаборатории,
- Облачная среда на условиях «инфраструктура как услуга» (IaaS), когда части системы или система целиком развернута в облаке,
- Облачная среда на условиях «ПО как услуга (SaaS), когда поставщик предоставляет услуги по нагрузочному тестированию.

В зависимости от целей и тестируемых систем предпочтение может быть отдано одному из видов тестового окружения. Например:

- Для тестирования эффекта от улучшения производительности (оптимизации производительности) изолированная среда в лаборатории подходит лучше, поскольку позволяет зафиксировать даже небольшие изменения в работе системы.
- Для проведения полного нагрузочного тестирования всей производственной среды, чтобы убедиться, что система выдержит нагрузку без серьёзных происшествий, тестирование из облака может быть более подходящим (необходимо отметить, что данный способ пригоден только при условии, что тестируемая система доступна из облака).
- Для минимизации расходов, когда время на тестирование производительности сильно ограничено, развертывание тестовой среды в облаке может быть более экономичным решением.

Какой бы подход ни был избран, аппаратная и программная конфигурация системы должны соответствовать целям и плану тестирования. Если среда соответствует производственной, она должна быть сконфигурирована таким же образом. В случае различий допустимы и изменения конфигурации, соответствующие таким различиям. Например, если на тестовых машинах меньше физической памяти, чем на производственных, может быть необходимо изменить программные параметры использования памяти (например, размер кучи виртуальной машины Java) для избегания использования файла подкачки.

Надлежащая конфигурация / эмуляция сети важна для глобальных и мобильных систем. Для глобальных систем (то есть таких, которые используются пользователями или используют для обработки данных центры со всего мира) возможно использовать подход, при котором создание нагрузки осуществляется с машин, размещённых в местах пребывания пользователей. Для мобильных систем наиболее практичным подходом остаётся эмуляция сети ввиду большой вариативности свойств сетевого подключения, которое может использоваться. Некоторые инструменты нагрузочного тестирования оснащены функционалом эмуляции сети, также для этого существуют и самостоятельные решения.

Инструменты создания нагрузки должны быть развернуты надлежащим образом, и мониторинг должен быть сконфигурирован так, чтобы собирать все необходимые для теста метрики. Список метрик зависит от целей теста, но в общем случае в любом тесте производительности рекомендуется собирать, по крайней мере, основные метрики (см. раздел 2.1).

В зависимости от нагрузки, подхода к ее созданию, выбранного инструмента создания нагрузки и конфигурации машин, может потребоваться более одной машины – генератора нагрузки. Для проверки корректности настроек необходимо также вести мониторинг машин, используемых для создания нагрузки. Это поможет избежать ситуацию, когда надлежащая нагрузка не поддерживается из-за медленной работы одного из генераторов нагрузки.

В зависимости от настроек и используемых инструментов, средства создания нагрузки необходимо конфигурировать, чтобы создать необходимую нагрузку. Например, может быть необходимо установить параметры для эмуляции работы браузера или использовать присвоение каждому виртуальному пользователю другого IP-адреса (IP-спуфинг).

Перед запуском тестов необходимо провести валидацию среды и настроек. Обыкновенно это делается путём запуска специально подобранного набора тестов и проверки результатов их работы, а также проверки того, что средства мониторинга собирают важную информацию.

Для проверки того, что тесты работают, как задумано, используются различные методы, включая анализ логов и проверку содержимого базы данных. На этапе подготовки теста необходимо проверить, что требуемая информация фиксируется в логе, система находится в надлежащем состоянии и т. д. Например, если тест существенно меняет состояние системы (добавляет / изменяет информацию в базе данных), может быть необходимо вернуть систему в первоначальное состояние перед повторным тестированием.

4.3 Выполнение тестирования

Выполнение теста производительности включает в себя создание нагрузки на тестируемую систему в соответствии с профилем нагрузки (обычно это осуществляется при помощи скриптов тестирования производительности, запускаемых согласно заданному сценарию), мониторинг всех компонентов тестового окружения, сбор и сохранение результатов и всей информации, касающейся проведения теста. Как правило, продвинутые инструменты тестирования производительности / тестовые обвязки выполняют данные задачи в автоматическом режиме (после, разумеется, соответствующей настройки). Они обычно предоставляют консоль управления для мониторинга тестирования и проведения необходимых корректировок (см. раздел 5.1). Однако, в зависимости от используемого инструмента, тестируемой системы и конкретных тестов, может потребоваться выполнение некоторых действий вручную.

Тестирование производительности обычно предусматривает устойчивое состояние системы, т.е. стабильное поведение. Например, все виртуальные пользователи (потоки) уже проинициализированы и выполняют работу в соответствии с проектом. С изменением нагрузки (например, при добавлении новых пользователей) изменяется также и поведение системы, что затрудняет мониторинг и анализ результатов тестирования. Стадия перехода в устойчивое состояние часто именуется «плавным нарастанием нагрузки» (ramp up), а завершающая стадия выполнения теста часто называется «плавным снижением нагрузки» (ramp down).

Иногда важно провести тестирование системы, находящейся в переходных состояниях, то есть, когда ее поведение изменяется, например, при одновременной авторизации большого количества пользователей или в случае тестирования производительности при всплесках нагрузки. При тестировании в условиях таких переходных состояний важно осознать необходимость тщательного мониторинга и анализа результатов, поскольку некоторые из стандартных подходов (таких, как использование средних значений при мониторинге) могут вводить в заблуждение.

Для отслеживания влияния повышения нагрузки во время ее нарастания (ramp up) на реакцию системы рекомендуется увеличивать нагрузку пошагово. Это позволяет убедиться в том, что для стадии плавного нарастания нагрузки выделено достаточно времени, и система способна обработать нагрузку. Хорошей практикой будет отслеживание, чтобы при достижении устойчивого

состояния как нагрузка, так и реакция системы были стабильными, и чтобы случайные колебания этих значений (которые существуют всегда) не являлись существенными.

Важно определить, как должны обрабатываться ошибки, чтобы исключить системные проблемы. Например, может быть важным, чтобы в случае ошибки пользователь вышел из системы, чтобы освободить все выделенные ему ресурсы.

Если средства мониторинга встроены в инструмент нагрузочного тестирования и настроены надлежащим образом, они обычно запускаются одновременно с началом проведения теста. Однако, в случае использования автономных инструментов мониторинга он должен запускаться отдельно, и собирать необходимую информацию следует таким образом, чтобы можно было проводить последующий анализ совместно с результатами испытаний. То же самое справедливо и в отношении анализа протоколов (логов). Важно синхронизировать во времени работу всех используемых инструментов, чтобы можно было найти всю информацию, относящуюся к конкретной итерации тестирования.

Выполнение теста часто контролируется при помощи консоли инструмента тестирования производительности и анализа протоколов (логов) в режиме реального времени для проверки наличия проблем как в тесте, так и в тестируемой системе. Это помогает избежать бесполезного выполнения крупномасштабных тестов, которые могут оказывать воздействие даже на другие системы в случае незапланированного хода проведения теста (например, при возникновении ошибок, отказе компонентов или создании слишком большой или маленькой нагрузки). Выполнение подобных тестов может быть очень затратным, и тогда, в случае отклонения поведения теста от ожидаемого, может возникнуть необходимость остановки тестирования или внесения на лету некоторых корректировок в тест производительности или конфигурацию системы.

Один из способов проверки во время проведения нагрузочных тестов, взаимодействующих непосредственно на уровне протокола, заключается в запуске скриптов, работающих на уровне графического интерфейса (функциональных), или даже выполнении подобных операционных профилей вручную параллельно с выполнением нагрузочного теста. Данный метод позволяет проверить, что времена отклика, полученные в ходе теста, отличаются от аналогичных значений, измеренных вручную на уровне графического интерфейса пользователя, лишь на величину времени, затрачиваемого на стороне клиента.

В некоторых случаях, когда тестирование производительности проводится в автоматическом режиме (например, как часть процесса непрерывной интеграции, как описано в разделе 3.4), проверки должны проводиться также автоматически, поскольку осуществлять мониторинг и вмешательство вручную может оказаться невозможным. В таких случаях тестовый стенд должен быть способен распознавать любые отклонения или проблемы и выдавать предупреждение (обычно в случае правильного завершения тестирования). Использование такого подхода легче организовать для регрессионного тестирования производительности, когда поведение системы общеизвестно, но сложнее в случае проведения пробных (экспериментальных) или крупномасштабных ресурсоемких тестов производительности, которые требуют корректировок, выполняющихся динамически во время тестирования.

4.4 Анализ результатов и отчетность

В разделе 4.1.2 обсуждались различные метрики из плана тестирования производительности. Перечень этих метрик указывает, что именно необходимо измерять при каждом выполнении теста. По завершении цикла тестирования необходимо собрать данные каждой из указанных метрик.

При анализе эти данные сначала сопоставляют с целью тестирования производительности. После того, как станет понятен режим работы тестируемой системы, можно будет делать выводы, которые представляют важную итоговую часть отчета, включая перечень рекомендуемых действий. Эти действия могут включать внесение изменений в оборудование (например, аппаратных средств, роутеров), программное обеспечение (например, оптимизация приложений и запросов к базе данных) и сетевую инфраструктуру (например, балансировка нагрузки, маршрутизация).

Обычно анализируются следующие данные:

- **Статус моделируемых (виртуальных) пользователей**. Эту метрику необходимо проверить в первую очередь. Обычно ожидается, что все виртуальные пользователи смогли выполнить задачи, указанные в операционном профиле. Желательно, чтобы любые сбои в выполнении этих операций воспроизводили только те ситуации, с которыми могут столкнуться реальные пользователи. Различные ошибки могут оказать влияние на остальные показатели производительности, поэтому так важно сперва убедиться, что все действия пользователя выполнены.
- **Время отклика транзакции**. Данная метрика может быть зафиксирована различными способами, в том числе минимальное, максимальное и среднее значения, перцентиль (например, 90-ый). Минимальное и максимальное показатели отражают точки экстремума производительности системы. Среднее значение производительности отражает лишь математическое среднее, которое может быть искажено за счет «выбросов». В качестве целевого показателя часто используется 90-ый перцентиль, поскольку он характеризует большинство пользователей, достигших определенного порогового значения производительности. Добиваться полного соответствия целям производительности не рекомендуется, поскольку это может потребовать непомерно высоких затрат ресурсов при незначительном эффекте для конечных пользователей.
- **Количество выполненных транзакций в секунду.** Эта метрика предоставляет информацию о том, сколько работы было выполнено системой (какова пропускная способность системы).
- **Количество транзакций, завершившихся отказом.** Эти данные используются при анализе количества выполненных транзакций в секунду. Отказы указывают на то, что рассматриваемый процесс не был завершен, или вообще не выполнялся. Любые обнаруженные отказы являются основанием для разбирательства и исследования их первопричин. Транзакции, при которых произошел сбой, также могут привести к неверным показателям количества выполненных транзакций в секунду, так как такие транзакции завершаются быстрее, чем выполненные полностью.
- **Количество запросов в секунду**. Эта метрика дает представление о количестве обращений виртуальных пользователей к веб-серверу каждую секунду теста.
- **Пропускная способность сети.** Эта метрика обычно измеряется в битах в единицу времени, например, бит в секунду. Она показывает количество данных, получаемых каждую секунду виртуальными пользователями от веб-сервера (см. раздел 4.2.5).
- **Коды ответов HTTP-сервера.** Эти данные замеряются каждую секунду и имеют значения возможных кодов ответа, таких как: 200, 302, 304, 404 (страница не найдена).

Значительная часть этой информации может быть представлена в виде таблиц, однако, графическое представление облегчает просмотр данных и выявление тенденций.

Методы, используемые при анализе данных, могут включать:

- Сравнение результатов с заявленными требованиями
- Наблюдение за тенденциями в результатах
- Статистические методы контроля качества
- Выявление ошибок
- Сравнение ожидаемого и полученного результатов
- Сравнение результатов с моделью производительности
- Сравнение результатов с результатами предыдущих испытаний
- Проверку надлежащего функционирования компонентов (например, серверов, сетей)

Выявление взаимосвязи между метриками может помочь понять, в какой момент производительность системы начинает снижаться. Например, какое количество транзакций в секунду было обработано, когда утилизация процессорных мощностей достигла 90%, и работа системы замедлилась?

Анализ может помочь определить основную причину снижения производительности или сбоя, что, в свою очередь, будет способствовать исправлению. Проведение подтверждающего тестирования поможет определить, устранило ли эту причину внесение корректировок.

Отчетность
Результаты анализа обобщаются и сравниваются с целями, указанными в плане тестирования производительности. Они могут быть представлены в общем отчете о выполнении теста вместе с другими результатами или выделены в специальный отчет о тестировании производительности. Уровень детализации должен соответствовать потребностям заинтересованных лиц. Рекомендации, основанные на этих результатах, обычно касаются критериев выпуска программного обеспечения (включая целевое окружение) или необходимых улучшений производительности.

Стандартный отчет о тестировании производительности может включать:

Основные итоги
Этот раздел заполняется, когда все тесты производительности были проведены, а результаты были проанализированы и осознаны. Его цель состоит в том, чтобы предоставить краткие и понятные выводы, а также рекомендации руководству для достижения целевых показателей.

Результаты тестирования
Результаты тестирования могут включать полностью или частично следующую информацию:

- Резюме с объяснением и подробным изложением результатов.
- Результаты эталонного теста, который служит «снимком» системы в заданный момент времени и составляет основу для сравнения с последующими тестами. Эти результаты должны содержать дату и время начала теста, достигнутое при выполнении теста число одновременно работающих в системе пользователей, установленную производительность и основные выводы, которые могут включать частоту ошибок, время отклика и среднюю пропускную способность.
- Общую архитектурную схему, отражающую все компоненты, которые оказывали (или могли оказывать) влияние на достижение целей тестирования.
- Подробный анализ результатов тестирования (при помощи таблиц и диаграмм), показывающий время отклика, интенсивность транзакций, частоту ошибок и анализ производительности. Также этот анализ содержит описание наблюдаемых событий, таких как момент, когда работа приложения стала нестабильной, и источник ошибок (например, веб-сервер, сервер базы данных).

Протоколы тестирования
При проведении тестирования должно производиться протоколирование. Обычно протокол содержит следующую информацию:

- Дата и время начала тестирования
- Продолжительность тестирования
- Использованные в ходе проведения тестирования скрипты и их группы (если таковые использовались) и соответствующая информация о настройках этих скриптов
- Файл(ы), содержащие тестовые данные
- Имена и расположение файлов данных/протоколов (логов), созданных при проведении тестирования
- Настройки конфигураций программного и аппаратного обеспечения (особенно стоит отметить любые изменения между запусками тестов)
- Средние и пиковые значения утилизации CPU и RAM на веб-серверах и серверах баз данных
- Отметки уровней достигнутой производительности
- Выявленные дефекты

Рекомендации
Рекомендации, основанные на результатах тестирования, могут включать:

- Рекомендуемые технические изменения, такие как изменение конфигурации оборудования, программного обеспечения или сетевой инфраструктуры

- Области, определенные для дальнейшего анализа (например, анализ протоколов (логов) веб-сервера для выявления основных причин проблем и/или ошибок)
- Необходимость дополнительного мониторинга шлюзов, серверов и сетей с целью получения более подробных данных для измерения характеристик и выявления тенденций (например, снижения) производительности

5. Инструменты – 90 минут

Ключевые слова
Генератор нагрузки, управление нагрузкой, инструмент мониторинга, инструмент тестирования производительности

Цели обучения

5.1 Инструментальная поддержка
PTFL-5.1.1 (K2) Понять, каким образом инструменты поддерживают проведение тестирования производительности

5.2 Пригодность инструмента
PTFL-5.2.1 (K4) Оценить пригодность инструментов тестирования производительности при заданном сценарии проекта

5.1 Инструментальная поддержка

Существуют следующие типы инструментов тестирования производительности:

Генераторы нагрузки
Генератор, используя интегрированную среду разработки, редактор скриптов или набор инструментов, способен создавать и запускать множественные экземпляры клиентов, имитирующих поведение пользователя в соответствии с заданным рабочим профилем. Создание нескольких экземпляров подобных клиентов за короткие промежутки времени вызовет нагрузку на тестируемую систему. Генератор как создает нагрузку, так и собирает метрики для последующего создания отчетов.
При выполнении тестов производительности задача заключается в том, чтобы имитировать реальный ход событий настолько, насколько это практически возможно. Зачастую это означает, что потребуются пользовательские запросы, поступающие из разных мест, а не только из одного. Окружения, построенные с несколькими точками создания нагрузки, следует распределить таким образом, чтобы источники нагрузки не находились в одной сети. Это обеспечивает реалистичность тестирования, хотя иногда может искажать результаты, если промежуточные сетевые сегменты создают задержки.

Консоль управления нагрузкой
Консоль управления нагрузкой обеспечивает контроль запуска и остановки генератора(ов) нагрузки. Также она объединяет метрики различных транзакций, заданных в экземплярах объектов создания нагрузки, используемых генератором. Консоль позволяет просматривать отчеты и графики выполнения тестов и способствует проведению анализа результатов.

Инструмент мониторинга
Средства мониторинга запускаются одновременно с тестируемым компонентом или системой и контролируют, записывают и/или анализируют поведение компонента или системы. Наиболее часто отслеживаются очереди веб-сервера, системная память и дисковое пространство. Средства мониторинга могут эффективно помочь найти первопричины снижения производительности тестируемой системы, а также могут использоваться для мониторинга производственной среды при выпуске продукта. Во время выполнения теста производительности средства мониторинга могут также использоваться непосредственно на самом генераторе нагрузки.
Модели лицензирования средств тестирования производительности включают традиционную лицензию на основе рабочих мест/сайтов с полным правом собственности, облачную модель лицензирования с оплатой по мере потребления и лицензии с открытым исходным кодом, которые можно свободно использовать в конкретной среде или посредством облачных решений. Каждая из моделей подразумевает различную схему оплаты и может включать текущую техподдержку. Очевидно, что для любого выбранного инструмента понимание принципов его работы (посредством обучения и/или самообучения), требует времени и финансов.

5.2 Пригодность инструментов

При выборе инструмента нагрузочного тестирования должны учитываться следующие факторы:

Совместимость
Как правило, инструмент выбирается для организации в целом, а не только для отдельного проекта. Это подразумевает учет следующих факторов в организации:

- Протоколы. Как описано в Главе 4.2.1, протоколы являются очень важным аспектом выбора инструмента нагрузочного тестирования. Понимание того, какие протоколы используются системой и списка тех из них, которые будут тестироваться, даст необходимую информацию для оценки пригодности конкретного инструмента.
- Интерфейсы к внешним компонентам. Интерфейсы к компонентам программного обеспечения или другим инструментам в составе полных интеграционных требований могут рассматриваться для соответствия процессу или другим требованиям совместимости (например, соответствие процессу CI).
- Платформы. Совместимость с платформами (и их версиями) в организации имеет существенное значение. Это относится как к платформам, используемым для размещения инструментов, так и платформам, с которыми эти инструменты взаимодействуют для мониторинга и / или генерации нагрузки.

Масштабируемость
Другой фактор, который необходимо учитывать – это общее число виртуальных пользователей, которое может одновременно поддержать инструмент. Это включает в себя несколько факторов:

- Максимальное требуемое число лицензий
- Требования к конфигурации рабочих станций и серверов, генерирующих нагрузку
- Возможность генерации нагрузки из различных местоположений (например, распределенные сервера)

Доступность
Другой фактор, который необходимо рассматривать — это требуемый уровень технических знаний для работы с инструментом. Это часто игнорируется и может привести к неточным результатам из-за некорректной конфигурации тестов неквалифицированными тестировщиками. Для тестирования, требующего сложных сценариев и высокого уровня программируемости и настроек, команды должны удостовериться, что тестировщик имеет необходимые навыки, опыт и подготовку.

Мониторинг
Достаточен ли мониторинг, предоставляемый инструментом? Доступны ли в среде другие инструменты мониторинга, которые могут дополнить мониторинг, предоставляемый инструментом нагрузочного тестирования? Можно ли соотнести показатели мониторинга со списком транзакций? Все эти вопросы требуется обсудить, чтобы понять, сможет ли выбранный инструмент обеспечить уровень мониторинга, требующийся для проекта.
Когда мониторинг представляет собой отдельную программу / инструменты / весь стек, его можно использовать для мониторинга производственной среды при выпуске продукта.

6. Ссылки

6.1 Стандарты

- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)

6.2 Документы ISTQB

- [ISTQB_UT_SYL] ISTQB Foundation Level Usability Testing Syllabus, Version 2018
- [ISTQB_ALTA_SYL] ISTQB Advanced Level Test Analyst Syllabus, Version 2012
- [ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 2012
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2012 (“Программа обучения Продвинутого уровня. Руководитель тестирования”)
- [ISTQB_FL_SYL] ISTQB Foundation Level (Core) Syllabus, Version 2018
- [ISTQB_FL_AT] ISTQB Foundation Level Agile Tester Syllabus, Version 2014 (“Сертифицированный тестировщик в сфере гибких методологий. Программа обучения”)
- [ISTQB_GLOSSARY] ISTQB Glossary of Terms used in Software Testing, http://glossary.istqb.org (ISTQB «Стандартный глоссарий терминов, используемых в тестировании программного обеспечения», https://www.rstqb.org/ru/istqb-downloads.html?file=files/content/rstqb/downloads/ISTQB%20Downloads/ISTQB%20%D0%93%D0%BB%D0%BE%D1%81%D1%81%D0%B0%D1%80%D0%B8%D0%B8%CC%86%20%D0%A2%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2%20%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%202.3.pdf)

6.3 Книги

[Anderson01] Lorin W. Anderson, David R. Krathwohl (eds.) “A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives”, Allyn & Bacon, 2001, ISBN 978-0801319037

[Bath14] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook”, Rocky Nook, 2014, ISBN 978-1-933952-24-6

[Molyneaux09] Ian Molyneaux, “The Art of Application Performance Testing: From Strategy to Tools”, O’Reilly, 2009, ISBN: 9780596520663

[Microsoft07] Microsoft Corporation, “Performance Testing Guidance for Web Applications”, Microsoft, 2007, ISBN: 9780735625709