***Тема 11 Тестирование производительности аппаратно – программного комплекса***

*Основные показатели (метрики) производительности.*

Одним из результатов, получаемых при нагрузочном тестировании и используемых в дальнейшем для анализа, являются показатели производительности приложения. Основные из них разобраны ниже.

1. Потребление ресурсов центрального процессора (CPU, %).

Метрика, показывающая, сколько времени из заданного определённого интервала было потрачено процессором на вычисления для выбранного процесса. В современных системах важным фактором является способность процесса работать в нескольких потоках, для того, чтобы процессор мог производить вычисления параллельно. Анализ истории потребления ресурсов процессора может объяснять влияние на общую производительность системы потоков обрабатываемых данных, конфигурации приложения и операционной системы, мультипоточности вычислений, и других факторов.

2. Потребление оперативной памяти (Memory usage, Mb).

Метрика, показывающая количество памяти, использованной приложением. Использованная память может делиться на три категории:

* Virtual — объём виртуального адресного пространства, которое использует процессор. Этот объём не обязательно подразумевает использование соответствующего дискового пространства или оперативной памяти. Виртуальное пространство конечно и процесс может быть ограничен в возможности загружать необходимые библиотеки.
* Private — объём адресного пространства, занятого процессом и не разделяемого с другими процессами.
* Working Set — набор страниц памяти, недавно использованных процессом. В случае, когда свободной памяти достаточно, страницы остаются в наборе, даже если они не используются. В случае когда, свободной памяти остается мало, использованные страницы удаляются.

При работе приложения память заполняется ссылками на объекты, которые, в случае неиспользования, могут быть очищены специальным автоматическим процессом, называемым «сборщиком мусора» (англ. Garbage Collector). Время затрачиваемое процессором на очистку памяти таким способом может быть значительным, в случае, когда процесс занял всю доступную память (в Java — так называемый «постоянный Full GC») или когда процессу выделены большие объёмы памяти, нуждающиеся в очистке. На время, требующееся для очистки памяти, доступ процесса к страницам выделенной памяти может быть заблокирован, что может повлиять на конечное время обработки этим процессом данных.

Потребление сетевых ресурсов.

Эта метрика не связана непосредственно с производительностью приложения, однако её показатели могут указывать на пределы производительности системы в целом.

Пример:

Серверное приложение, обрабатывая запрос пользователя, возвращает ему видео-поток, используя сетевой канал в 2 мегабит. Требование гласит, что сервер должен обрабатывать 5 запросов пользователей одновременно.

Нагрузочное тестирование показало, что эффективно сервер может предоставлять данные только 4 пользователям одновременно, так как мультимедиа-поток имеет битрейт в 500 килобит. Очевидно, что предоставление этого потока 5 пользователям одновременно невозможно в силу превышения пропускной способности сетевого канала, а значит, система не удовлетворяет заданным требованиям производительности, хотя при этом потребление ей ресурсов процессора и памяти может быть невысоким

4. Работа с дисковой подсистемой (I/O Wait).

Работа с дисковой подсистемой может значительно влиять на производительность системы, поэтому сбор статистики по работе с диском может помогать выявлять узкие места в этой области. Большое количество чтений или записей может приводить к простаиванию процессора в ожидании обработки данных с диска и в итоге увеличению потребления CPU и увеличению времени отклика.

5. Время выполнения запроса (request response time, ms).

Время выполнения запроса приложением остаётся одним из самых главных показателей производительности системы или приложения. Это время может быть измерено на серверной стороне, как показатель времени, которое требуется серверной части для обработки запроса; так и на клиентской, как показатель полного времени, которое требуется на сериализацию / десериализацию, пересылку и обработку запроса. Надо заметить, что не каждое приложение для тестирования производительности может измерить оба этих времени.

*Требования к производительности АПК.*

Очень важно детализировать требования к производительности и документировать их в каком-либо плане тестирования производительности. В идеальном случае это делается на стадии разработки требований при разработке системы, до проработки деталей её дизайна.

Однако тестирование производительности часто не проводится согласно спецификации, так как нет зафиксированного понимания о максимальном времени ответа для заданного числа пользователей. Тестирование производительности часто используется как часть процесса профайлинга производительности. Его идея заключается в том, чтобы найти «слабое звено» — такую часть системы, оптимизировав время реакции которой, можно улучшить общую производительность системы. Определение конкретной части системы, стоящей на этом критическом пути, иногда очень непростая задача, поэтому некоторые приложения для тестирования включают в себя (или могут быть добавлены с помощью add-on’ов) инструменты, запущенные на сервере (агенты) и наблюдающие за временем выполнения транзакций, временем доступа к базе данных, оверхедами сети и другими показателями серверной части системы, которые могут быть проанализированы вместе с остальной статистикой по производительности.

Тестирование производительности может проводиться с использованием глобальной сети и даже в географически удаленных местах, если учитывать тот факт, что скорость работы сети Интернет зависит от местоположения. Оно также может проводиться и локально, но в этом случае необходимо настроить сетевые маршрутизаторы таким образом, чтобы появилась задержка, присутствующая во всех публичных сетях. Нагрузка, прилагаемая к системе, должна совпадать с реальным положением дел. Так например, если 50 % пользователей системы для доступа к системе используют сетевой канал шириной 56К, а другая половина использует оптический канал, то компьютеры, создающие тестовую нагрузку на систему должны использовать те же соединения (идеальный вариант) или эмулировать задержки вышеуказанных сетевых соединений, следуя заданным профайлам пользователей.

*Определение целей тестирования производительности.*

В общих случаях тестирование производительности может служить разным целям.

* С целью демонстрации того, что система удовлетворяет критериям производительности.
* С целью определения производительность какой из двух или нескольких систем лучше.
* С целью определения, какой элемент нагрузки или часть системы приводит к снижению производительности.

Многие тесты на производительность делаются без попытки осмыслить их реальные цели. Перед началом тестирования всегда должен быть задан бизнес-вопрос: «Какую цель мы преследуем, тестируя производительность?». Ответы на этот вопрос являются частью технико-экономического обоснования(или business case) тестирования. Цели могут различаться в зависимости от технологий, используемых приложением, или его назначения, однако, они всегда включают что-то из нижеследующего:

* параллелизм / Пропускная способность;
* время ответа сервера;
* время отображения;
* требования к производительности;
* типичные вопросы тестирования производительности;
* нагрузочное тестирование, цели и задачи.

Нагрузочное тестирование (load testing) — подвид тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству).

Это простейшая форма тестирования производительности. Нагрузочное тестирование обычно проводится для того, чтобы оценить поведение приложения под заданной ожидаемой нагрузкой. Этой нагрузкой может быть, например, ожидаемое количество одновременно работающих пользователей приложения, совершающих заданное число транзакций за интервал времени. Такой тип тестирования обычно позволяет получить время отклика всех самых важных бизнес-транзакций. В случае наблюдения за базой данных, сервером приложений, сетью и т.д., этот тип тестирования может также идентифицировать некоторые узкие места приложения.

Для исследования времени отклика системы на высоких или пиковых нагрузках производится стресс-тестирование, при котором создаваемая на систему нагрузка превышает нормальные сценарии её использования. Не существует чёткой границы между нагрузочным и стресс-тестированием, однако эти понятия не стоит смешивать, так как эти виды тестирования отвечают на разные бизнес-вопросы и используют различную методологию.

*Основные принципы нагрузочного тестирования.*

Ниже рассмотрены некоторые экспериментальные факты, обобщённые в принципы, используемые при тестировании производительности в целом и применимые к любому типу тестирования производительности (в частности и к нагрузочному тестированию).

1. Уникальность запросов

Даже сформировав реалистичный сценарий работы с системой на основе статистики её использования, необходимо понимать, что всегда найдутся исключения из этого сценария.

2. Время отклика системы

В общем случае время отклика системы подчиняется функции нормального рампределения.

В частности, это означает, что, имея достаточное количество измерений, можно определить вероятность с которой отклик системы на запрос попадёт в тот или иной интервал времени.

3. Зависимость времени отклика системы от степени распределённости этой системы.

Дисперсия нормального распределения времени отклика системы на запрос пропорциональна отношению количества узлов системы, параллельно обрабатывающих такие запросы и количеству запросов, приходящихся на каждый узел.

То есть, на разброс значений времени отклика системы влияет одновременно количество запросов приходящихся на каждый узел системы и само количество узлов, каждый из которых добавляет некоторую случайную величину задержки при обработке запросов.

4. Разброс времени отклика системы

Из утверждений 1, 2 и 3 можно также заключить, что при достаточно большом количестве измерений величины времени обработки запроса в любой системе всегда найдутся запросы, время обработки которых превышает определённые в требованиях максимумы; причем, чем больше суммарное время проведения эксперимента, тем выше окажутся новые максимумы.

Этот факт необходимо учитывать при формировании требований к производительности системы, а также при проведении регулярного нагрузочного тестирования.

5. Точность воспроизведения профилей нагрузки

Необходимая точность воспроизведения профилей нагрузки тем дороже, чем больше компонент содержит система.

Часто невозможно учесть все аспекты профиля нагрузки для сложных систем, так как чем сложнее система, тем больше времени будет затрачено на проектирование, программирование и поддержку адекватного профиля нагрузки для неё, что не всегда является необходимостью. Оптимальный подход в данном случае заключается в балансировании между стоимостью разработки теста и покрытием функциональности системы, в результате которого появляются допущения о влиянии на общую производительность той или иной части тестируемой системы.

*Инструментарий для тестирования производительности.*

Следует отметить, что для большинства видов тестирования производительности используется один и тот же инструментарий, умеющий выполнять типовые задачи.

Существует распространённое ошибочное пониманиетого, что инструменты для нагрузочного тестирования системы — это инструменты такие же по принципу записи и воспроизведения, как и инструменты для автоматизации регрессионного тестирования. Инструменты для нагрузочного тестирования работают на уровне протокола, тогда как инструменты для автоматизации регрессионного тестирования работают на уровне объектов графического пользовательского интерфейса.

Существуют различные инструменты для обнаружения и исследования проблем в различных узлах системы. Все узлы системы могут быть классифицированы следующим образом:

* приложение,
* база данных,
* сеть,
* обработка на клиентской стороне,
* балансировка нагрузки.

Также следует отметить появление сетевых Busines-to-business (B2B) приложений, использующих соглашение об уровне услуг (или SLA, Service Level Agreement). Нарастающая популярность B2B-приложений привела к тому, что всё больше приложений переходят на сервис-ориентированную архитектуру, в случае которой обмен информацией происходит без участия веб-браузеров. Примером такого взаимодействия может служить бюро туристических услуг, запрашивающее информацию об определённом авиарейсе между Санкт-Петербургом и Омском, в то время как авиакомпания обязана предоставить ответ в течение 5 секунд. Часто нарушение договора об SLA грозит крупным штрафом.

*Стрессовое тестирование, цели и задачи.*

Стресс-тести́рование (Stress Testing) — один из видов тестирования ПО, которое оценивает надёжность и устойчивость системы в условиях превышения пределов нормального функционирования. Стресс-тестирование особенно необходимо для «критически важного» ПО, однако также используется и для остального ПО. Обычно стресс-тестирование лучше обнаруживает устойчивость, доступность и обработку исключений системой под большой нагрузкой, чем то, что считается корректным поведением в нормальных условиях.

Термин «стресс-тестирование» часто используется как синоним «нагрузочного тестирования», а также «тестирования производительности», что ошибочно, так как эти виды тестирования отвечают на разные бизнес-вопросы и используют различную методологию.

В общем случае методология стресс-тестирования основана на снятии и анализе показателей производительности приложения при нагрузках, значительно превышающих ожидаемые на стадии сопровождения, и несёт в себе цель определить выносливость или устойчивость приложения на случай всплеска активности по его использованию.

Необходимость стресс-тестирования диктуется следующими факторами:

* большая часть всех систем разрабатываются с допущением о функционировании в нормальном режиме и даже в случае, когда допускается возможность увеличения нагрузки, реальные объёмы её увеличения не принимаются во внимание.
* В случае SLA-контракта стоимость отказа системы в экстремальных условиях может быть очень велика.
* Обнаружение некоторых ошибок или дефектов в функционировании системы не всегда возможно с использованием других типов тестирования.
* Тестирования, проведенного разработчиком, может быть недостаточно для эмуляции условий, при которых происходит отказ системы.
* Предпочтительнее быть готовым к обработке экстремальных условий системы, чем ожидать её отказа.
* Основные направления применения стресс-тестирования:
* Общее исследование поведения системы при пиковых нагрузках.
* Исследование обработки ошибок и исключительных ситуаций системой при пиковых нагрузках.
* Исследование узких мест системы или отдельных компонент при диспропорциональных нагрузках.
* Тестирование ёмкости системы.

Стресс-тестирование, как и нагрузочное тестирование, также может быть использовано для регулярной оценки изменений производительности с целью получения для дальнейшего анализа динамики изменения поведения системы за длительный период.

Пропорциональная нагрузка. Стресс-тестирование может применяться как для обособленных приложений, так и для распределенных систем с клиент-серверной архитектурой. Простейшим примером стресс-тестирования обособленного приложения может являться открытие файла размером в 50 Мб программой Notepad, входящей в комплект ОС Windows. Условия стресс-тестирования приложения обычно формируются исходя из критических бизнес-процессов его функциональности, определенными на стадии разработки требований и анализа рисков группой, ответственной за производительность.

В общем случае в качестве условий для стресс-тестирования может использоваться линейно увеличенная ожидаемая нагрузка.

Диспропорциональная нагрузка. В случае тестирования многозвенных распределенный систем необходимо учитывать уже не только фактический объём нагрузки, состоящей из множества элементов, но и их пропорции в общем объёме. Использование диспропорциональной нагрузки в стресс-тестах может также применяться для выявления узких мест отдельных компонент системы.

*Объемное тестирование, цели и задачи.*

Тестирование емкости (*Capacity Testing*) является одним из самых важных с точки зрения развития бизнеса направлений стресс-тестирования и самых сложных с точки зрения проведения и анализа. Тестирование Ёмкости проводится с целью определить запас прочности системы при полном соответствии требованиям к производительности.

При моделировании нагрузки для тестирования ёмкости системы учитывается как текущая нагрузка в виде количества и пропорций одновременно поступающих в систему запросов, так и ожидаемая в перспективе.

Пример 2:

Веб-сервис предназначен для отображения данных для зарегистрированных пользователей в формате .docx. Разрабатывающая компания собирается опубликовать возможность отображения данных в формате .pdf, при этом ожидается, что текущие пользователи будут продолжать работать со старым форматом, а новые будут использовать формат .pdf. Моделируя нагрузку на системы во время тестирования ёмкости, учитывается не только текущий сценарий её использования (отображение данных в .docx), но и предполагаемый сценарий использования в будущем (часть одновременно работающих с системой пользователей используют .docx, а часть - .pdf).

Результатом тестирования ёмкости приложения или системы является набор максимально допустимых характеристик нагрузки системы, при которых приложение или система отвечает требованиям к производительности, разработанным и документированным на этапе проектирования архитектуры.

Тестирование стабильности или надежности, цели и задачи.

Тестирование стабильности или надежности(Stability / Reliability Testing) — один из видов автоматизированного тестирования ПО, целью которого является проверка работоспособности приложения при длительном тестировании с ожидаемым уровнем нагрузки.

Перед тем как подвергать ПО экстремальным нагрузкам стоит провести проверку стабильности в предполагаемых условиях работы, то есть погрузить продукт в полную рабочую атмосферу. При тестировании, длительность его проведения не имеет первостепенного значения, основная задача — наблюдая за потреблением ресурсов, выявить утечки памяти и проследить чтобы скорость обработки данных и/или время отклика приложения в начале теста и с течением времени не уменьшалась. В противном случае вероятны сбои в работе продукта и перезагрузки системы.

Часто в «домашних» условиях тестирование стабильности совмещают со стресс-тестированием, то есть проверяют не только стабильность, но и способность приложения переносить жесткие условия и сильные нагрузки длительное время.