УДК 004

ОСНОВНЫЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ СИСТЕМЫ РАСЧЕТА НАГРУЗКИ КРИТИЧЕСКОГО РЕЖИМА

[1]Певнева А.Г., [2]Марьевский В.А., [1]Ананченко И.В., [1]Войтюк Т.Е.

[1]Университет ИТМО, Санкт-Петербург,
e-mail: pevnevaa@inbox.ru, anantchenko@yandex.ru, taire2006@ya.ru;
[2]ФГБВОУ ВО «Военно-космическая академия имени А.Ф. Можайского»,
Санкт-Петербург, e-mail: marevskij@bk.ru

Представленная в настоящей статье проблематика является результатом обобщения опыта тестирования производительности распределенной системы и осмысления этого опыта в теоретическом аспекте. В статье предложены некоторые правила проектирования программного инструментария для проверки комплексной надежности информационной системы, которые полагаются основными. Сформулированы принципы программного конструирования испытательного стенда для оценки производительности и особенности использования инструментов нагрузочного тестирования в ситуациях появления системных и программных ошибок. Даются краткое описание моделей сценариев нагрузочного тестирования, математическое описание процесса тестирования работы информационной системы под критической нагрузкой в терминологии системного анализа и ссылки на обоснование моделей аппаратом теории случайных процессов и систем массового обслуживания. Приведена классификация информационных систем по критерию реакции на повышение нагрузки. Даны развернутое описание ошибок, которые диагностируются в ходе тестирования нагрузки, и классификация этих ошибок по различным критериям. Отмечено, что перспективным представляется использование линейной темпоральной логики с часами (Temporal Logicwith Clock, TLC). Применение инструментов нагрузочного тестирования, обладающих способностью одновременного запуска различных по составу последовательных тестов, позволяет гибко моделировать характерные ситуации, выделяя проблемные места. Высказываются предположения о методологическом подходе к формализации и моделированию процесса тестирования производительности в рамках логического моделирования с использованием специфических модальных алгоритмических логик.

Ключевые слова: нагрузочное тестирование, системный анализ, критическая нагрузка, инструменты стресс-тестирования, разработка тестовых сценариев

MAIN DESIGNING ASPECTS OF THE CALCULATE LOADS SYSTEM IN CRITICAL MODE

[1]Pevneva A.G., [2]Marevskiy V.A., [1]Ananchenko I.V., [1]Voytiyk T.E.

[1]ITMO University, Saint-Petersburg,
e-mail: pevnevaa@inbox.ru, anantchenko@yandex.ru, taire2006@yandex.ru;
[2]Federal state budget military educational institution of higher professional education «Military space Academy named after A.F. Mozhaysky» of the Ministry of defence of the Russian Federation, St. Petersburg, e-mail marevskij@bk.ru

The present article shows issue which is a result of summarizing the experience of testing the performance of adistributed system and comprehending this experience in a theoretical aspect. The article proposes some designing rules of software tools for testing the comprehensive reliability of an information system, as main problem. Article defines principles of programmatic design of the testing stand for evaluating performance. It presents features of using stress testing tools in contest of system and program errors. Article has a brief description of the model's scenarios of stress testing, the mathematical description of the testing process of the information system under critical load by using terminology of system analyses and links for theory of random processes and theory of Queuing systems which are substantiate the model. The information systems classification is given by the criterion of response to increased load. A detailed description of the errors is given that are diagnosed during load testing and the classification of these errors according to various criteria. Assumptions are made about a methodological approach to formalizing and modelling the performance testing process in the framework of logical modelling by using specific modal algorithmic logics.

Keywords: load testing, system analysis, critical load, stress testing tools, development of test scenarios

Поведение информационной системы (ИС) в условиях повышения нагрузки характеризует ее отказоустойчивость и является одной из составляющих комплексной надежности информационной системы [1]. Организация работ по оценке производительности в условиях реальной эксплуатации ИС – это сложный процесс, так как создание разноуровневой нагрузки требует жесткой регламентации поведения большого числа пользователей. В этих условиях возможны информационные потери и нарушение целостности информации [2]. Чтобы оценить поведение системы в критических условиях, можно воспользоваться математическими моделями для их расчета. Методология этих моделей имеет в основе теорию массового обслуживания [3]. Но известны случаи серьезного, до 20%, несовпадения таких расчетов с фактическими [2, 4]. По-видимому, это связано с динамически изменяющейся неоднородностью

всей системы в контексте исследования предельной производительности: критичные элементы программно-аппаратной платформы всегда имеются, но их вклад в комплексную надежность изменяется в зависимости от условий эксплуатации системы. Выявление таких элементов в структуре ИС – это сложное направление [1, 5]. Поэтому и с теоретической, и с практической точки зрения представленная работа является весьма актуальной. Для получения экспериментальных данных о нагрузке может использоваться испытательный стенд. Описание общей структуры такого стенда является целью данной статьи.

Описание процесса нагрузочного тестирования в аспекте системного анализа

Так как в реальных условиях эксплуатации заранее не известен объем нагрузки в каждый момент, для формального описания традиционно используют математический аппарат случайных процессов, с дискретными состояниями и непрерывным временем.

Пусть p_i – вероятности событий, состоящих в том, что система находится в состоянии S_i (рисунок): S_0 – все линии обслуживания свободны, S_1 – одна линия занята обработкой запроса, … S_n – все порты заняты обслуживанием запросов, но в очереди количество запросов пренебрежимо мало, вероятность этого события p_n; S_{n+1} – все порты заняты, в буфере один запрос; … S_{n+m} – все порты заняты, буфер переполнен, т.е. регистрируемое событие – «отказ в обслуживании».

Переход из состояния S_i в S_{i+1} происходит с увеличением количества запросов к определенным портам, интенсивность этого потока λ_{ij}, обратный переход происходит в результате обработки запроса, интенсивности потока обслуживания μ_{ji}. Вероятность $p_i(t)$ того, что система будет находиться в состоянии S_i в момент времени t (вместе с начальными условиями $p_i(0)$), определяется системой дифференциальных уравнений А.М. Колмогорова. Методы решения этой системы известны для марковских процессов с известными характеристиками входных потоков и потоков обслуживания [6].

Далее приведем основные составляющие комплексной производительности [7]:

– количество операций, выполняемых в системе в единицу времени (пропускная способность);

– количество операций, выполняемых одновременно (параллелизм);

– время выполнения операции (время отклика/задержка);

– объем резервной мощности, необходимый системе для обеспечения роста нагрузки (запас ресурсов);

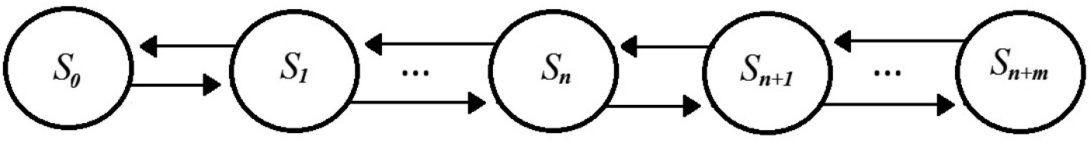
– количество исключений, генерированных в системе при повышенной нагрузке (частота ошибок).

Модельное соответствие этих величин приведенному выше случайному процессу вполне очевидно. Например, под линией обслуживания подразумеваются открытые порты, интенсивность потока обслуживания μ_{ji} – это и есть пропускная способность, однако первая сложность на пути реализации теоретических расчетов – большое число состояний современной ИС, следовательно, большая (до 10^6) размерность задачи.

Кроме этого, модели существенно усложняются в случаях, когда режим работы системы не является стационарным и теряет марковские свойства. Причем при падении интенсивности событий входного потока этим вполне допустимо пренебрегают [8], но при возрастании интенсивности отсутствие марковских свойств процесса серьезно усложняет расчеты, так как для конкретной ИС необходимо выбирать конкретный способ конструирования, упрощения и разрешения математической модели. Улучшение ситуации возможно, если теоретические выкладки проводить совместно с программным проектированием ИС и учитывать их в программных сценариях тестирования.

Наконец, следует указать, что само понятие состояния системы для каждой конкретной ИС является комплексным и события перехода из одного состояния в другое могут не образовывать полную группу событий.

Указанные рассуждения объясняют частые несовпадения расчетных значений с фактическими показателями производительности ИС.



Граф состояний системы обслуживания

Тестирование нагрузки в аспекте проверки надежности

Реакцию ИС системы на повышение нагрузки до критических значений можно классифицировать следующим образом:

– идеальные ИС, где при плавном повышении нагрузки во времени производительность падает до некоторого значения, но в дальнейшем ее можно считать постоянной независимо от увеличения нагрузки;

– устойчивые ИС, где плавному повышению нагрузки после критического уровня соответствует такой же плавный спад производительности;

– неустойчивые ИС, где при незначительном превышении максимальной нагрузки имеет место резкое падение производительности.

Для получения экспериментальных данных, характеризующих поведение ИС, используется тестирование под нагрузкой с применением специального программного инструментария, разработанного с учетом условий эксплуатации. Нагрузочное тестирование – комплексная методика применения тестовых сценариев, которые имитируют возрастание нагрузки на ИС в различных режимах.

Интуитивно ясно, что объем нагрузки связан с количеством запросов различных типов. Известные инструменты стресс-тестирования сети, например утилита slowhttptest, используют следующие параметры: общее количество соединений, i – время ожидания данных, r – количество запросов указанного типа (например, GET запросы), x – размер запроса в байтах, p – время ожидания ответа на запрос.

В ходе нагрузочного тестирования ИС основную роль играет модель взаимосвязи обработки информации. Она является последовательной или конвейерной, так что входные данные на каждом технологическом этапе обработки запроса зависят от выходных данных предыдущего. Чтобы модель была адекватной, разработанные инструменты тестирования должны учитывать взаимозависимость рабочих циклов, преобразуя их в соответствующие этапы выполнения тестовых сценариев. При этом необходимо учесть асинхронную передачу данных в циклах исполнения сценариев.

Особенности использования инструментов нагрузочного тестирования

В зависимости от цели нагрузочное тестирование проводится в следующих направлениях:

– моделирование DOS-атак как моделирование преднамеренных действий со стороны злоумышленника;

– исследование деятельности ИС под повышенной нагрузкой;

– определение предельной величины нагрузки на ИС в конкретном окружении.

Моделирование DOS-атак в этом случае следует понимать как моделирование действий злоумышленника, в распоряжении которого имеются далеко не все ресурсы системы, а только сравнительно небольшая часть каналов для посылки запросов. Благодаря принципу масштабируемости можно организовать массированную тестовую атаку несколькими тысячами запросов в короткий промежуток времени с одной или нескольких нагрузочных станций. Заметим, что уязвимыми к подобным атакам бывают приложения распределенных вычислений, так как в процессе работы каждое соединение ассоциируется с конкретным системным портом, процессом или областью памяти.

Исполнение DOS-атак параллельно с эмуляцией пользовательского поведения в штатном режиме проводится с целью оценки влияния дестабилизирующих воздействий на ход процесса обработки запросов.

Чтобы определить предельную величину нагрузки на ИС в конкретном окружении, для одного и того же тестового сценария выполняется серия испытаний с увеличением нагрузки, т.е. увеличение запросов на обработку к каждой службе. Значение времени ожидания ответа фиксируется в каждом испытании. Нагрузка, при которой время ожидания становится больше некоторого конкретного значения, подобранного специально для данной системы в данном окружении, определяется как критическая.

Принципы конструирования испытательного стенда для оценки производительности

Испытательный стенд предназначен для имитации взаимодействия «клиент – сервер» при работе приложения в режиме одновременной работы пользователей, число которых меняется на несколько порядков в разных сеансах тестирования.

При разработке такого стенда придерживаются следующих принципов.

1. Принцип масштабирования нагрузки при параллельном запуске тестовых сценариев. Число эмулируемых пользователей на виртуальных рабочих станциях должно в пределе отличаться как минимум на три порядка от нормальной средней нагрузки в ходе проведения эксперимента.

2. Принцип централизованной обработки и управления. В состав инструментария должны входить центральная рабочая стан-

ция, которая реализует функции управления экспериментом, средства журналирования эксперимента, статистическая обработка результатов.

3. Принцип адаптивных моделей. Тестовые сценарии для эмуляции пользовательских транзакций должны соответствовать реальным условиям эксплуатации.

4. Принцип независимости (асинхронность исполнения). Любая тестовая транзакция с различными условиями имеет свой план исполнения и количество имитируемых пользователей.

5. Принцип структурированного хранения результатов тестирования. Условия любого эксперимента, в том числе и нагрузочного, должны обеспечивать возможность его повторения. При этом система хранения результатов должна предусматривать удобные инструменты сравнительного анализа. Оптимальным представляется использование реляционных систем управления базами данных для хранения конфигураций и результатов [9].

6. Принцип мониторинга состояния операционной системы. Анализ системной статистики и статистики испытуемой системы выявляет несбалансированность процессов в ИС. Такой анализ, очевидно, проводится на единой временной оси.

Структура стенда, спроектированного по таким принципам, должна включать, кроме самого объекта испытания, рабочую станцию, откуда специалист осуществляет управление процессом тестирования, программное обеспечение статистического анализа результатов тестирования. Отдельно расположены станции, содержащие программные средства эмуляции поведения множества пользователей. Кроме того, в зависимости от аппаратной и программной реализации могут потребоваться дополнительные станции мониторинга системных процессов и процессов обработки запросов в ИС в реальном времени, а также отдельно реализованные генераторы тестов для постоянного обеспечения станций эмуляции нагрузки [1].

Классификация проявляющихся ошибок

Основная цель тестирования – обнаружение ошибок. В источнике [1] ошибки, наиболее часто выявляемые в условиях высокоинтенсивной нагрузки, приведены списком, без классификации. Но, с точки зрения авторов, для проектирования реестра, содержащего данные о проведении нагрузочного тестирования, подобная классификация необходима. При системном подходе все ошибки, влияющие на производительность, можно разделить на три группы: ошибки управления ресурсами, ошибки управления временем, ошибки потери связи между аппаратным и программным обеспечением, которое задействует ИС. Все они возникают вследствие неэффективности алгоритмов диспетчеризации, а также сегментации памяти или иных необратимых процессов. Ошибочные алгоритмы, работающие против общей поставленной задачи перед ПО, накапливая ошибочное значение в каком-либо определенном параметре, занимают лишнее место в памяти.

К ошибкам управления ресурсами отнесем прежде всего утечки памяти (утечки ресурсов). Процесс неконтролируемого уменьшения объема свободной оперативной или виртуальной памяти компьютера, связанный с ошибками в работающих программах, вовремя не освобождающих ненужные участки памяти, или с ошибками системных служб контроля памяти, связан с нагружением памяти, при этом другая информация не имеет возможности поместиться в нее. Это влечет отказы в выделении ресурсов памяти, сетевых соединений, портов. При этом скрипт обходит по очереди каждый узел и выполняет на нем определенные действия. Проблема в том, что, как только скрипт доходит до «лежачего» узла (а они есть практически всегда), его выполнение прерывается. Отказы в выделении ресурсов влекут за собой взаимные блокировки (deadlocks) на таблицах баз данных. При этом процесс 1 блокирует ресурс А, процесс 2 блокирует ресурс Б, процесс 1 пытается получить доступ к ресурсу Б, процесс 2 пытается получить доступ к ресурсу А. В итоге один из процессов должен быть прерван, чтобы другой мог продолжить выполнение.

К ошибкам управления временем отнесем ошибки синхронизации. Если синхронизация работает правильно, число элементов в поле «Содержимое» папки на сервере и в поле «Содержимое» автономной папки совпадает. Если синхронизация не работает или проверки папки сервера и автономной папки не осуществляются, необходимо проверить параметры сверки данных. Ошибки обработки тайм-аутов применяются в основном для реализации таймеров и тайм-аутов, где используются функции, возвращающие текущее время, когда время может быть изменено администратором или злоумышленником. Ошибки управления временем влекут несбалансированность процессов обработки между собой, т.е. перегрузки одних процессов на фоне недогруженности других.

Ошибки двух этих категорий выявляют узкие проблемные места в системе – ресурсы, использование которых имеет наиболь-

шее значение. Балансирование системы/процессов между собой обеспечивает выравнивание загрузок компонентов системы, в результате чего устраняется узкое место и все узлы начинают действовать как ресурсы с одинаковой эффективностью использования.

Заключение

По результатам проведенных обобщений эмпирических наблюдений за тестированием производительности ИС можно сделать вывод о необходимости корректировки расчетных моделей нагрузки в процессе проектирования информационных систем. Обобщение сведений по результатам такого моделирования помогло бы в будущем выявить зависимости между параметрами составляющих системы. Разработка тестовых сценариев должна также проводиться на этапе проектирования компонентов ИС.

Проектирование информационной системы, возможно, требует не только традиционного инфологического и графического подхода, но и формализаций с использованием аппарата алгебры множеств и модальных логик. Перспективным представляется использование линейной темпоральной логики с часами (Temporal Logicwith Clock, TLC). Изначально эта логическая система относится к модальным темпоральным логикам с множественной грануляцией времени. Данная логическая система может использоваться для доказательства корректной работы (верификации и валидации) систем, реагирующих на сторонние запросы. Грануляция, т.е. классификация временных отрезков для каждой предикатной формулы, позволяет выводить в данной логике утверждения, однозначно определяющие синхронизацию процессов, а возрастание нагрузки до критической привело бы к логическому противоречию. Вывод формул, как прямой, так и обратный, в этой логической системе может также использоваться при проектировании тестовых сценариев. Практико-ориентированные разработки в этом направлении представляются достаточно перспективными.

Список литературы

1. Нагрузочное тестирование как элемент формирования безопасных систем [Электронный ресурс]. URL: http://docplayer.ru/80673204-Nagruzochnoe-testirovanie-kak-element-formirovaniya-bezopasnyh-sistem.html (дата обращения: 15.03.2020).

2. Котов А.С., Котов С.Л., Гибин Ю.В. Средства обеспечения надежности функционирования информационных систем // Программные продукты и системы. 2002. № 2. URL: http://www.swsys.ru/index.php?page=article&id=705 (дата обращения: 15.03.2020).

3. Рыжиков Ю.И. Численные методы теории очередей. 1-е изд., стер. СПб.: Издательство «Лань», 2019. 512 с.

4. Masashi Narumoto. Анализ ключевых показателей производительности [Электронный ресурс]. URL: https://habr.com/ru/company/microsoft/blog/271547 (дата обращения: 15.03.2020).

5. Гончаренко В.А. Метод оценивания нагрузоустойчивости критических информационных систем в условиях неопределенности на основе технологии нагрузочного зондирования // Известия ТулГУ. Технические науки. 2018. №1 [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/metod-otsenivaniya-nagruzoustoychivosti-kriticheskih-informatsionnyh-sistem-v-usloviyah-neopredelennosti-na-osnove-tehnologii (дата обращения: 15.03.2020).

6. Рыжиков Ю.И. Алгоритмический подход к задачам массового обслуживания: монография. СПб.: ВКА им. А.Ф. Можайского, 2013. 496 с.

7. Стресс-тестирование: комплексная производительность [Электронный ресурс]. URL: https://habr.com/ru/company/microsoft/blog/271547/ (дата обращения: 15.03.2020).

8. Щеглов А.Ю., Щеглов К.А. Математические модели и методы формального проектирования систем защиты информационных систем: учебное пособие. СПб.: Университет ИТМО, 2015. 93 с.

9. Голиков О.И., Панкратов И.А. Исследование способов повышения эффективности обработки данных в реляционных БД на примере СУБД MySQL // Вестник ВУиТ. 2016. № 2. С. 124–130.