УДК 004.4'22

Б.В. Тютин, А.О. Веселов, В.П. Котляров

МАСШТАБИРОВАНИЕ ВЫПОЛНЕНИЯ ТЕСТОВОГО НАБОРА ПРИ АВТОМАТИЗИРОВАННОМ ТЕСТИРОВАНИИ

B.V. Tyutin, A.O. Veselov, V.P. Kotlyarov

TEST SUITE EXECUTION SCALING FOR AUTOMATED TESTING

Рассмотрено понятие масштабирования выполнения тестового набора и подход к решению этой задачи, реализованный в системе тестирования TestCommander. Проведена оценка результатов использования разработанного подхода.

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ. ЮНИТ-ТЕСТИРОВАНИЕ. ВЕРИФИКАЦИЯ. УПРАВЛЕНИЕ ТРЕБОВАНИЯМИ. МАСШТАБИРОВАНИЕ.

This article reviews the test execution scaling problem and the approach for its solution implemented in TestCommander, an automated testing system. Results of the system piloting are analyzed and discussed.

TESTING AUTOMATION. UNIT-TESTING. VERIFICATION. REQUIREMENT MANAGEMENT. SCALING.

В общем виде масштабирование означает способность системы увеличивать свою производительность при добавлении ресурсов пропорционально их объему. Количественно это свойство можно оценить через отношение прироста производительности системы к приросту используемых ресурсов. Чем ближе это отношение к единице, тем лучше.

Применительно к тестированию понятие масштабирования можно определить как способность тестирующей системы ускорять выполнение тестового набора за счет параллельного выполнения его частей. Цель в снижении временных затрат на выполнение тестов, что приводит к возможности более полного покрытия кода тестами, повышению качества тестирования и более эффективному использованию имеющихся аппаратных средств. В существующих системах тестирования и библиотеках для их построения данная задача решена в лучшем случае частично за счет интеграции со сторонними инструментами. Это повышает сложность организации тестирования и, следовательно, связанные с ним затраты.

Основная идея работы состоит в создании метода масштабирования выполнения тестового набора и его реализации в системе автоматического тестирования TestCommander [1], ориентированной на функциональное и модульное тестирование. Для параллельного выполнения тестов осуществлена модификация ядра системы с использованием MPI, что позволило адаптировать TestCommander к работе на многопоточной и кластерной инфраструктуре. Для масштабирования тестового набора применен комбинированный подход, основанный на кластеризации набора тестов и изоляции среды их исполнения.

Возможные подходы к масштабированию. Выделяют два типа масштабирования: вертикальное и горизонтальное [2]. Вертикальное масштабирование – увеличение производительности каждого компонента системы с целью повышения общей производительности. Горизонтальное масштабирование – разбиение системы на более мелкие структурные компоненты и разнесение их по отдельным физическим машинам (или их группам) или увеличение количества серверов, параллельно выполняющих одну и ту же функцию.

В зависимости от типа тестирования и логики тестов можно выделить два подхода

к решению данной задачи. Первый – параллельное выполнение тестов. Он подразумевает разбиение исходного тестового набора на части и их выполнение в различных процессах, расположенных как на одной, так и на разных рабочих станциях. Данный подход может успешно применяться в области модульного тестирования, а также тестирования серверных приложений. Для того чтобы успешно использовать параллельное выполнение тестов необходимо, чтобы выполнение одних тестов не влияло на выполнение других. Второй способ масштабирования тестового набора – дублирование тестового стенда. Это горизонтальное масштабирование процесса тестирования, когда взаимодействующие тестирующая и тестируемая системы развертываются на нескольких платформах. Данный подход требует больших аппаратных ресурсов, однако позволяет выполнять любые виды тестирования.

При проектировании расширения системы автоматизации тестирования TestCommander, поддерживающей масштабирование выполнения тестового набора, были учтены следующие аспекты.

1. Балансировка нагрузки и критерии разделения тестового набора на части. Балансировка может производиться на доступных рабочих узлах в зависимости от нагрузки и их количества, сложности тестов или прогнозируемого времени выполнения.

2. Изоляция выполнения. При выполнении масштабирования требуется обеспечить отсутствие сторонних эффектов при выполнении отдельных тестов, влияющих на весь процесс тестирования.

3. Сбор и обработка результатов. Необходимо обеспечить целостность журналов событий, а также любой другой информации, регистрируемой в ходе тестирования.

4. Прозрачность для пользователя. Процесс масштабирования должен контролироваться системой тестирования, а не тестировщиком, с тем, чтобы выполнение тестового набора без масштабирования было идентично параллельному выполнению с точки зрения пользователя.

Система тестирования. TestCommander представляет собой генератор кода тестового окружения на основе тестовых сценариев, конфигурационного XML файла и целевого кода, отвечающего за сериализацию/десериализацию сообщений, отправляемых/получаемых через внешние интерфейсы окружения. В TestCommander входит набор инструментов, используемых на разных этапах тестирования: макроподстановщик, анализатор трасс, кодогенерирующий шаблон, генератор обертки [1].

В качестве языка тестовых сценариев используется Message Sequence Chart (MSC) [3]. Данный язык позволяет определить взаимодействие различных сущностей, в т. ч. и параллельное, в терминах отправки и приема сигналов. На основании набора тестов в формате MSC и конфигурационного файла автоматически создается код тестов в виде машины состояний, а также прослойка, обрабатывающая сигналы и реализующая взаимодействие с тестируемой системой [4]. Далее полученный код собирается в готовый тестовый набор. Запуск тестов и управление тестированием осуществляется автоматически средствами TestCommander. Результаты тестирования (журнал ошибок, отчеты по каждому пройденному тесту, сводные таблицы) представляются в формате HTML.

Для создания тестов, адекватных к исходным требованиям к продукту, а также автоматизированного создания MSC с контролируемым уровнем покрытия тестами TestCommander интегрирован со средствами разработки Use-case maps (UCM) [1] и верификатором VRS [1].

TestCommander был успешно применен в ряде проектов по автоматизации тестирования телекоммуникационных систем. По результатам его использования можно сказать об удобстве данного инструмента при решении задач функционального и регрессионного тестирования [4].

Кластеризация тестового набора. В контексте задач тестирования под кластеризацией подразумевается разбиение набора тестовых сценариев на отдельные группы тестов с целью их независимого выполнения. Это позволяет эффективно использовать имеющиеся аппаратные ресурсы, распределяя нагрузку более равномерно.

При разбиении тестов на группы необходимо обеспечить выполнение связанных между собой тестов в рамках одного кластера в необходимом порядке. Данное требование напрямую следует из требования воспроизводимости и однозначности процесса тестирования [5].

В системе TestCommander разбиение тестового сценария осуществляется на основании указываемого пользователем числа групп. При этом разделение происходит на основании одного из двух следующих показателей:

- Количество тестов в группе. При использовании данного показателя тестовый набор делится на группы, содержащие указанное пользователем число тестов.
- Сложность тестов. При использовании данного показателя суммарная сложность тестов в различных группах примерно одинакова. Данную величину можно выразить формулой $C = \sum_{i=1}^{N} n_i c_i$, где C – суммарная сложность тестов; n_i – множитель для конкретного события или управляющей конструкции на диаграмме; c_i – весовой коэффициент, соответствующий выражению. Весовые коэффициенты и множители для событий выбираются таким образом, чтобы учитывать требования к времени прохождения теста и интенсивности обмена информацией. При кластеризации тестов TestCommander также учитывает наличие взаимосвязей между тестами.

Изолирование среды исполнения тестов. Изолирование среды исполнения необходимо при параллельном запуске тестов в том случае, если возможно взаимное влияние таких тестов друг на друга, например, при обращении к разделяемому ресурсу или одновременном взаимодействии с одним и тем же экземпляром тестируемой системы [6]. Такое влияние может оказать воздействие на вердикт прохождения теста и привести к нестабильным результатам прогона тестового набора.

Для предотвращения нежелательного взаимодействия параллельно исполняемых тестов система тестирования автоматически создает для каждого теста полностью изолированное тестовое окружение и отдельный экземпляр тестируемой системы (или ее компонент). Изолирование происходит прозрачно для пользователя без необходимости дублирования информации или ручных манипуляций.

Тестирование происходит в рамках каждой из изолированных систем путем взаимодействия компонент тестового окружения с компонентами тестируемой системы, что подразумевает необходимость указания связей между интерфейсами компонент, что является одним из этапов конфигурирования тестового окружения.

В простейшем случае, при отсутствии какой-либо информации о связях, считается, что тестовое окружение состоит всего из одной компоненты, которая напрямую подключается ко всем интерфейсам всех компонент тестируемой системы. В более сложных случаях пользователь имеет возможность при помощи конфигурационного файла описать связи между компонентами. Следует отметить, что такое конфигурирование не влияет на свойство изолированности, т. к. единожды описанное в конфигурационном файле, оно применяется многократно для каждого из тестов в отдельности.

Параллельное выполнение тестов. После успешного завершения этапа кластеризации и конфигурирования возможна генерация кода тестового набора, его сборка и запуск. Данные этапы автоматизированы в системе TestCommander. Генерация и сборка кода осуществляется на основном узле, после чего тестовые наборы распределяются по имеющимся рабочим станциям, либо запускаются на текущей. Запуск тестов координируется системой тестирования. Результаты выполнения тестов собираются вместе, при этом сохраняется структура журналов, что делает данную операция прозрачной для пользователя.

Для возможности параллельного выполнения тестового набора на кластерных вычислительных системах в TestCommander реализован режим кластеризации тестов на этапе выполнения с помощью вызовов MPI. Выполнение отдельных тестов происходит на различных процессорах системы, при этом сохраняются свойства изолиро-

ванности тестов и воспроизводимости тестового набора.

Ограничения используемого подхода. Разработанная в ходе выполнения данной работы методика была испытана в различных проектах тестирования. По результатам ее использования определен ряд ограничений и особенностей реализации. Наиболее важными являются следующие из них:

- сложности в настройке конфигурации, в которой тестирующая система выступает в роли мастера;
- необходимость создания обертки для систем, закрытых от взаимодействия;
- наличие только одного критерия разбиения при выполнении тестов в кластерных системах.

Данные ограничения определяют границы эффективного применения системы тестирования и дальнейшие направления ее развития.

Традиционно тестированию отводится второстепенная роль в цикле создания программного продукта. Некоторые современные подходы к разработке, такие, как test-driven development, привлекают внимание разработчиков к проблеме обеспечения качества программного продукта на всех этапах его создания, что приводит к развитию инструментов тестирования. Новые возможности в данной области позволяют сделать процесс тестирования более удобным, дешевым и полезным [7].

Уменьшение времени выполнения тестового набора и автоматизация запуска тестов ведет к снижению издержек на организацию тестирования и его проведение. Масштабирование выполнения тестов – один из наиболее эффективных способов для достижения этого результата. Реализованный в системе TestCommander подход к решению данной задачи, включающий в себя кластеризацию тестов и их параллельное выполнение на рабочих станциях или узлах кластера, опробован в ряде проектов тестирования компонент телекоммуникационных систем. По результатам данных проектов можно сделать вывод о том, что наиболее эффективно задача масштабирования решалась при организации модульного тестирования за счет возможности параллельного запуска тестов, и при интеграционном тестировании, где главную роль сыграла возможность запуска изолированных многокомпонентных тестов. Прозрачность процесса масштабирования для пользователя системы является фактором, снижающим сложность работы с ней.

СПИСОК ЛИТЕРАТУРЫ

1. **Тютин, Б.В.** Построение системы автоматизации статической и динамической проверки требований к программному продукту [Текст]/ Б.В. Тютин, И.В. Никифоров, В.П. Котляров // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. –СПб.: Изд-во Политехнического ун-та, 2012. – № 4 (152). – С 119–123.

2. **Таненбаум, Эндрю.** Распределенные системы. Принципы и парадигмы [Текст] / Эндрю Таненбаум, Мартин ван Стеен. – СПб.: Питер, 2003.

3. ITU-T Recommendation Z.120: Message sequence chart (MSC). Geneva, Switzerland, October 1996 [Электронный ресурс] /Режим доступа http://eu.sabotage.org/www/ITU/Z/Z0120e.pdf

4. **Веселов, А.О.** Автоматическая настройка тестового окружения телекоммуникационных проектов [Текст] / А.О. Веселов, А.С. Иванов, Б.В. Тютин, В.П. Котляров // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. –СПб.: Изд-во Политехнического ун-та, 2011. – № 4 (128). – С. 149–152.

5. **Osherove, R.** The Art of Unit Testing: With Examples in .Net. [Text] / R. Osherove. – Greenwich: Manning Publications Co, 2010.

6. **Калбертсон, Р.** Быстрое тестирование [Текст] / Р. Калбертсон, К. Браун, Г. Кобб. – М.: ИД «Вильямс», 2002.

7. **Beck, K.** Test-Driven Development: By Example [Text]/ K. Beck. – Addison-Wesley, 2002.

REFERENCES

1. **Tyutin B.V., Nikiforov I.V., Kotlyarov V.P.** Implementation of static and dynamic requirement analisys system [*Elaboration of the toolsuite for automation of the static and dynamic software requirement checking*] / Nauchno-tekhnicheskie vedomosti SPbGPU. Informatika. Telecommunikatsii.

Upravlenie [*St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunications and Control Systems*]. St. Petersburg: Izd-vo Politekhnicheskogo un-ta, 2012. – № 4 (152). – S. 119–123. (rus)

2. **Tanenbaum Andrew S., van Steen Maarten.** Destributed systems. Principles and paradigms. – St. Petersburg: Piter, 2003. (rus)

3. ITU-T Recommendation Z.120: Message sequence chart (MSC). Geneva, Switzerland, October 1996; Availiable http://eu.sabotage.org/www/ITU/Z/Z0120e.pdf

4. **Veselov A.O., Ivanov A.S., Tyutin B.V., Kotlyarov V.P.** Avtomaticheskaya nastroyka testovogo okruzheniya telekommunikatsionnih proektov [*Automation of test environment configuring for telecom projects*] / Nauchno-tekhnicheskie vedomosti SPbGPU. Informatika. Telecommunikatsii. Upravlenie [*St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunications and Control Systems*]. St. Petersburg: Izd-vo Politekhnicheskogo un-ta, 2011. – № 4 (128). – S. 149–152. (rus)

5. **Osherove R.** The Art of Unit Testing: With Examples in .Net. – Greenwich: Manning Publications Co, 2010.

6. **Culbertson Robert, Brown Chris, Cobb Gary.** Byistroe testirovanie [Rapid Testing]. – Moscow: ID «Williams», 2002 (rus)

7. **Beck K.** Test-Driven Development: By Example. – Addison-Wesley, 2002.

ТЮТИН Борис Викторович – *аспирант кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: b.tyutin@gmail.com

TYUTIN, Boris V. – *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: b.tyutin@gmail.com

ВЕСЕЛОВ Алексей Олегович – *аспирант кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: veselov.alexey@gmail.com

VESELOV, Alexey O. – *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: veselov.alexey@gmail.com

КОТЛЯРОВ Всеволод Павлович – *профессор кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета, кандидат технических наук.*
E-mail: vpk@ics2.ecd.spbstu.ru

KOTLYAROV, Vsevolod P. – *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: vpk@ics2.ecd.spbstu.ru