Труды МАИ. 2026. № 147.
Trudy MAI. 2026. No. 147. (In Russ.)

Научная статья / Original Article
УДК 004.415.532.2
URL: https://trudymai.ru/published.php?ID=188124
EDN: https://www.elibrary.ru/XQPOSY

АЛГОРИТМ ТЕСТИРОВАНИЯ РАЗЛИЧНЫХ КЛАССОВ ЭКВИВАЛЕНТНОСТИ МОМЕНТАЛЬНЫХ СНИМКОВ ВИРТУАЛЬНОЙ МАШИНЫ

О.Р. Лисовский

Акционерное общество «БФГ»,

г. Москва, Россия

✉ Lisoleg555@yandex.ru

Цитирование: Лисовский О.Р. Алгоритм тестирования различных классов эквивалентности моментальных снимков виртуальной машины // Труды МАИ: электрон. журнал. № 147.
URL: https://trudymai.ru/published.php?ID=188124

Аннотация. Работа посвящена проблеме тестирования моментальных снимков виртуальной машины. Из-за активно проходящего импортозамещения, в том числе в сфере авиационной инфраструктуры, появились новые российские продукты виртуализации, которые не имеют отлаженного процесса тестирования. В связи с этим был разработан алгоритм для тестирования одного из модулей гипервизора. Алгоритм основан на применении классов эквивалентности. Сценарии работы с моментальными снимками виртуальной машины были разбиты на три непересекающихся класса: единичный снимок, три последовательных снимка и два параллельных снимка из единого корня. Для каждого из этих классов на основе всех параметров и функций, а также эвристике, учитывающей повторение действий, было составлено конечное число тестов. В результате были получены 182 теста, из которых 170 являются независимыми и могут выполняться параллельно.

Ключевые слова: гипервизор; моментальный снимок; автотесты; параллельные тесты; классы эквивалентности; smoke-тестирование.

ALGORITHM FOR TESTING DIFFERENT EQUIVALENCE CLASSES OF VIRTUAL MACHINE SNAPSHOTS

O.R. Lisovskiy✉

Joint-stock company «BFG»,

Moscow, Russia

✉ Lisoleg555@yandex.ru

Citation: Lisovskiy O.R. Algorithm for testing different equivalence classes of virtual machine snapshots // Trudy MAI. 2026. No. 147. (In Russ.).
URL: https://trudymai.ru/published.php?ID=188124

Abstract. This paper examines the problem of testing virtual machine snapshots. Due to the ongoing import substitution, including in the aviation infrastructure sector, it has been emerged new domestic virtualization products that have a lack of well-established testing process. Therefore, an algorithm for testing one of the hypervisor modules was developed. The algorithm is based on equivalence classes. Scenarios for working with virtual machine snapshots were divided into three disjoint classes: a single snapshot, three sequential snapshots, and two parallel snapshots from a single root. For each of these classes, a finite number of tests were compiled based on all parameters and functions, as well as heuristics accounting for action repetition. This resulted in 182 tests, 170 of which are independent and can be executed in parallel.

Keywords: hypervisor; snapshot; autotests; parallel tests; equivalence classes; smoke testing.

Введение

В современных облачных технологиях широко применяются гипервизоры – программы, с помощью которых создаётся сотни и тысячи виртуальных машин, хранилищ и сетей на физических машинах [1–3], количество которых многократно меньше. Из них создаются виртуальные среды, которые активно применяются в организации инфраструктуры аэропортов [4-6]. Из-за использования в этих средах виртуальных машин с ценными данными требуется возможность быстрого отката изменений их состояния. Такую функцию берут на себя моментальные снимки (snapshot). Они представляют собой быстрый снимок

памяти виртуального жёсткого диска и, при необходимости, снимок оперативной памяти виртуальной машины [7–9]. В ряде случаев с их помощью удалось предотвратить потерю ценных данных в аэропорту [10, 11]. В крупных виртуальных средах может потребоваться несколько последовательных моментальных снимков: первый моментальный снимок порождает второй моментальный снимок, из которого, в свою очередь, следует третий моментальный снимок. Несколько параллельных снимков: из первого снимка идут моментальные снимки два и три, которые не имеют прямой связи между собой. Подобные структуры требуют тщательного тестирования, а из-за больших временных затрат требуют оптимизации количества тестов и надёжного алгоритма, по которому можно выполнять ручное тестирование и составлять автотесты с возможностью их параллельного выполнения. Проблема отсутствия такого алгоритма стоит достаточно остро, так как в последние годы идёт активная разработка различных решений в области виртуализации [12, 13]. На тему работы виртуальных машин написаны различные статьи [7–9], есть литература, описывающая процесс тестирования программного обеспечения [14–17]. Также есть статьи на тему использования средств виртуализации для совершенствования программ и их тестирования [18–20]. Но работ, посвящённых тестированию виртуализации, нет. В России активно идёт процесс импортозамещения, и, в том числе, аэропорты переходят на недавно разработанные российские системы виртуализации [21-22]. Поскольку тестирование виртуализации очень объёмная область, предлагается сконцентрировать внимание на моментальных снимках и представить алгоритм их тестирования. В работе не затрагиваются проблемы взаимосвязи миграции и резервного копирования виртуальных машин с моментальными снимками, рассматривается непосредственно раздел «моментальные снимки».

Основные положения тестирования моментальных снимков и простейший класс эквивалентности

Автором статьи предложен алгоритм тестирования моментальных снимков. Перед их непосредственным тестированием требуется убедиться, что были успешно протестированы установщик гипервизора, разделы «хранилища» и

«виртуальные машины», так как без стабильной работы этих компонентов гипервизора тестирование моментальных снимков будет необъективным [23, 24]. На проверку алгоритму выделяется несколько классов эквивалентности [25] для более быстрой реализации тестовых сценариев и автотестов к разделу «моментальные снимки»:

- единичные снимки;
- три последовательных моментальных снимка;
- несколько параллельных снимков с одним корневым;

Для обеспечения независимости последующих тестов следует проводить каждый тест на отдельной виртуальной машине. Это обусловлено тем, что при возникновении серьёзных ошибок дальнейшее проведение тестов моментальных снимков на этой виртуальной машине будет давать некорректные результаты, в первую очередь в случае с проваленными тестами. Некорректные результаты могут быть получены как на конкретных тестовых сценариях, так и быть следствием накопления незначительных ошибок при многочисленных операциях с моментальными снимками на одной виртуальной машине. Подобный подход также обеспечивает высокую степень параллельности алгоритма, что стоит учесть при автоматизации тестирования моментальных снимков и изначально проектировать автотесты для параллельного выполнения. Однако проведение каждого нового теста на отдельной виртуальной машине имеет следующий недостаток: из-за создания виртуальной машины увеличивается время каждого теста и возникает необходимость очищать хранилище от виртуальных машин, на которых уже закончился тест.

Из-за того, что тестировщики обычно не имеют доступа ко всему исходному коду и того, что для тестирования нескольких структур моментальных снимков нет необходимости в глубоких знаниях работы кода гипервизоров, тестирование можно проводить по методу «чёрного ящика»: разбивать тесты на классы эквивалентности, применять деревья и графы и применять паросочетания совместно с ортогональными массивами [26]. Сценарии со значительным количеством операций со снимками следует рассматривать отдельно, например в нагрузочном тестировании моментальных снимков [27, 28].

В начале предложенного алгоритма проверяется самый простой класс эквивалентности – один моментальный снимок виртуальной машины. Такая структура дерева снимков предполагается к наиболее частому использованию. В её рамках необходимо провести ряд тестов. Первыми должны быть два теста на создание моментального снимка:

- с оперативной памятью виртуальной машины;
- без оперативной памятью виртуальной машины;

Эти действия будут необходимы для всех тестов, приведённых в данной статье, так как для любых действий с моментальными снимками их требуется предварительно создать. Если эти два базовых теста проваливаются, то можно сделать вывод о полной неисправности модуля, что позволяет аварийно завершить работу всего алгоритма, так как все последующие тесты нуждаются в предварительном создании снимка. При нормальных обстоятельствах (успешного прохождения этих двух тестов) алгоритм переходит к более сложным тестам, которые могут выполняться параллельно:

- возврат (revert) к моментальному снимку;
- удаление снимка;
- редактирование снимка (в большинстве случаев предполагает простую смену названия и описания);
- удаление всех моментальных снимков виртуальной машины в одно действие (встречается в гипервизорах для крупных виртуальных сред), далее «удалить все снимки»;

Снимки могут как содержать оперативную память, так и не содержать, поэтому количество тестов для класса эквивалентности единичных моментальных снимков – сначала два теста на создание, после чего два теста на возврат, два теста на удаление, два теста на редактирование и два теста на функцию «удалить все снимки» в параллельном режиме.

Однако можно провести следующую оптимизацию: в тестах на создание моментального снимка удалить их либо через функцию «удалить все снимки», либо через удаление единичного снимка. Таким образом мы избавляемся от двух тестов. Эта оптимизация возможна из-за того, что для удаления моментального

снимка необходимо его предварительно создать, а также из-за того, что предположительно моментальные снимки не будет храниться больше нескольких дней, и пользователи будут создавать и удалять снимки с практически идентичной частотой.

Итого для проверки минимальной работоспособности класса эквивалентности «один моментальный снимок» необходимо восемь тестов. Для любых тестов моментальных снимков обязательно после каждой операции (создание, возврат, удаление, редактирование и «удалить все снимки»)

- проверять состояние виртуальной машины: включена она или выключена (при некорректной реализации моментальных снимков виртуальная машина может поменять состояние);
- актуальность информации на дисковых устройствах виртуальной машины и возможность работы с ними;
- наличие файлов снимков в хранилище гипервизора;
- текущее состояние дерева моментальных снимков на рассматриваемой виртуальной машине;

Часть или весь данный набор тестов можно использовать и для smoke-тестирования (тестирования самых основных функций программы для выявления минимальной работоспособности [14]) всего раздела «моментальные снимки».

Класс эквивалентности «три последовательных моментальных снимка»

Следующий класс эквивалентности – три последовательных моментальных снимка. В такой структуре предполагается сделать первый моментальный снимок, после чего от него сделать второй и завершить её созданием третьего моментального снимка, который по окончании создания будет последним активным (в него записываются изменения виртуального жёсткого диска) снимком (рисунок 1). Этот класс эквивалентности позволяет проверить на минимальном количестве моментальных снимков операции с корневым снимком (с которого начинается дерево снимков), промежуточным снимком (создаётся из корня, но не является последним) и снимком-листом (на котором дерево заканчивается). В начале этого фрагмента алгоритма в первую очередь проводится тест создания данной

структуры, после чего проводятся три теста на редактирование моментальных снимков: корневого, промежуточного и листа. Необходимо учитывать, что снимки могут как включать оперативную память виртуальной машины, так и не включать. При создании и редактировании моментальных снимков влияние на результат теста оказывает только наличие оперативной памяти в снимках, с которыми происходит взаимодействие в рамках теста, а не в структуре глобально, поэтому для данных тестов для оптимизации рассматривается два случая: все моментальные снимки без оперативной памяти, все моментальные снимки с оперативной памятью. В результате мы имеем шесть тестов на редактирование.

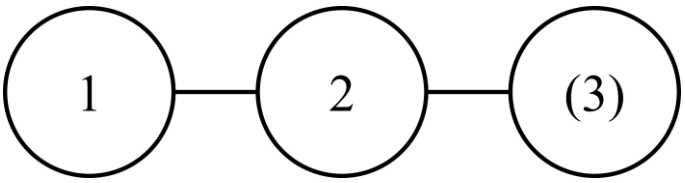

Рисунок 1 – Изображение класса эквивалентности «три последовательных моментальных снимка». Цифрами обозначен порядок создания, а цифрой в скобках – активный моментальный снимок без операций возврата

Тесты на работу функций возврата, удаления и удаления всех снимков более комплексные, так как они все будут включать в себя несколько операций для сокращения количества тестов. Подобная ситуация связана с тем, что необходимо тестирование не только с позиции активного моментального снимка, которым в данных тестах всегда будет снимок-лист, но и с позиции корневого и промежуточного снимков. В тестах на возврат необходимо проверить возвращения с каждого моментального снимка структуры на каждый, в том числе и на себя самого. С тремя последовательными моментальными снимками возможны проверки следующих возвратов: с листа на лист; с листа на промежуточный; с листа на корень; с промежуточного на лист; с промежуточного на промежуточный; с промежуточного на корень; с корня на лист; с корня на промежуточный; с корня на корень. Однако из-за того, что возврат возможен только с активного снимка, которым в конце создания всех трёх всегда будет последний, в проверках с четвёртой по девятую необходимо будет предварительно со снимка-листа вернуться либо на промежуточный снимок, либо на корень, что включает проверки два и три. Это позволяет провести оптимизацию – достаточно будет проверять успешность перехода с листа на корень и промежуточный снимок в прочих тестах. Благодаря этому количество минимально необходимых тестов удалось уменьшить с девяти до семи. В тестах на возврат к моментальному снимку наличие или отсутствие в снимке оперативной памяти

виртуальной машины, как и в тестах на создание снимков, зависит только от интересующих нас моментальных снимков, поэтому можно точно также рассмотреть только два случая: все снимки с оперативной памятью, все снимки без неё. В итоге минимальное количество тестов функции возврата в классе эквивалентности «три последовательных моментальных снимка» будет в два раза больше – 14.

В тестах на удаление моментальных снимков по одному в классе эквивалентности «три последовательных моментальных снимка» необходимо учитывать положение активного снимка и удаляемых моментальных снимков из-за особенностей работы функции удаления снимков. Во многих случаях это не удаление, а консолидация одного моментального снимка с другим, из-за чего данные не удаляются, а переходят следующему снимку. В связи с этим необходимо проверить все возможные комбинации удаления моментальных снимков: удаляется только один снимок, удаляются два снимка, удаляются все три моментальных снимка – итого 15 тестов (рисунок 2).

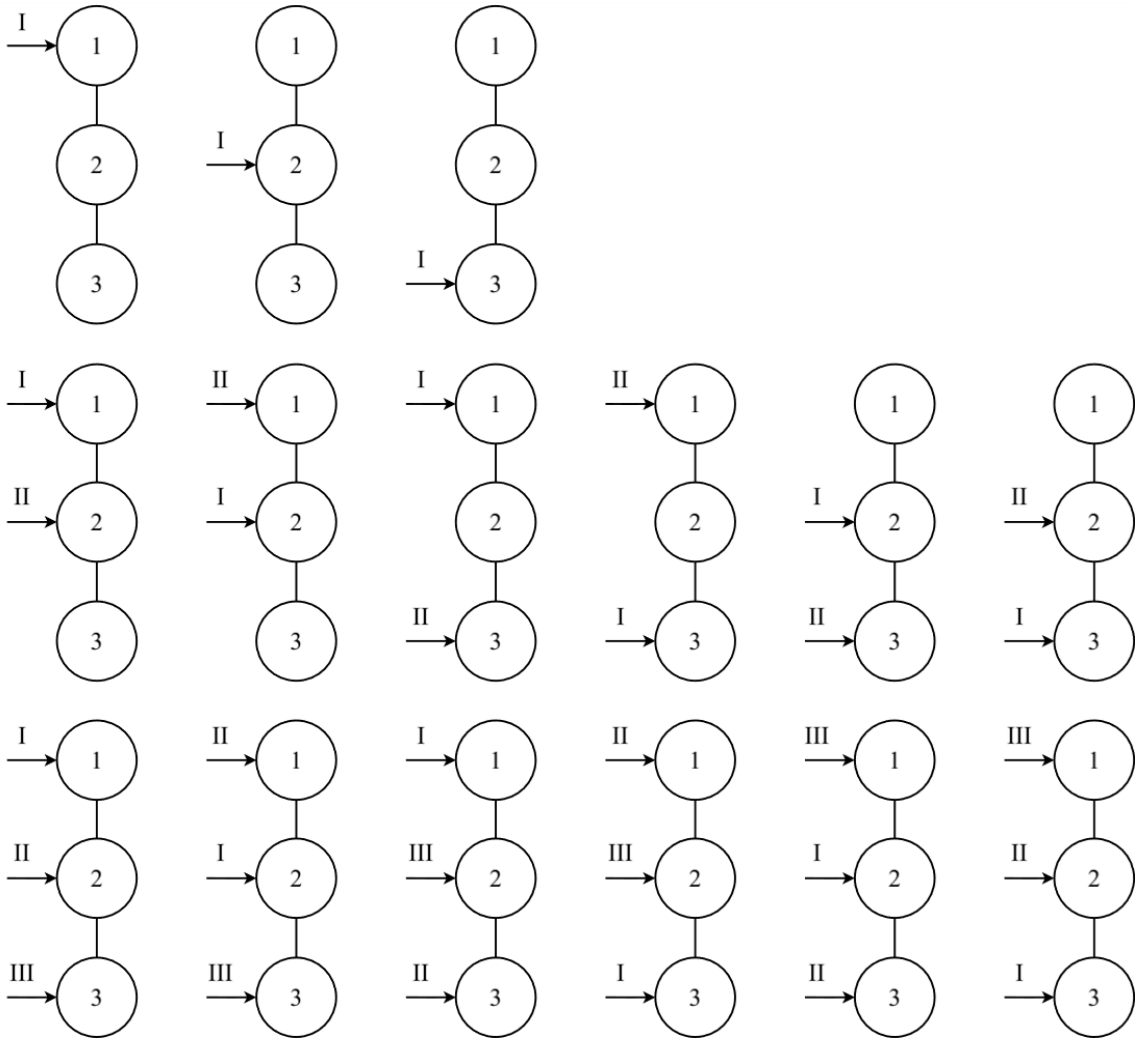

Рисунок 2 – Все возможные комбинации удаления снимков по одному. Арабскими цифрами обозначен порядок создания снимков, стрелками – удаляемые снимки, римскими цифрами – порядок удаления.

Однако подобный набор можно оптимизировать. В тестах на удаление трёх снимков содержатся все состояния структуры дерева этого класса, которые оно принимает в процессе выполнения прочих комбинаций. Благодаря этому при проведении проверок состояния виртуальной машины и её виртуальных дисков после каждой операции с моментальными снимками количество тестов на удаление одиночных снимков можно сократить до шести основных с комбинациями удаления всех трёх моментальных снимков. Однако из-за необходимости проверить удаление в ситуациях, когда активный снимок может быть не только снимком-листом, но и корневым и промежуточным, необходимо увеличить количество тестов в три раза для получения объективной картины удаления и работы консолидации. Это влечёт увеличение тестов на удаление моментальных снимков по одному до 18.

Функция «удалить все снимки» консолидирует все моментальные снимки от корня до активного, а прочие уже полностью удаляет. В связи с этим тесты данной функции будут рассматривать часть комбинаций из описанных выше: удаление двух снимков (шесть комбинаций) и удаление одного снимка (три комбинации). В этих тестах предварительно проводится удаление моментальных снимков согласно этим комбинациям, после чего применяется функция «удалить все снимки», что даёт уже 10 тестов. Также прибавляется дополнительный тест: применение функции «удалить все снимки» при наличии всех трёх моментальных снимков. В данных тестах необходимо учитывать позицию активного снимка (этих позиций может быть три), что делает общее количество тестов функции «удалить все снимки» равным 30.

Наличие или отсутствие оперативной памяти виртуальной машины никак не влияет на результат удаления или консолидации моментального снимка, а только на то, включена виртуальная машина или выключена после операции со снимками. В связи с этим в тестах на класс эквивалентности «три последовательных моментальных снимка» снова можно рассматривать два случая: все снимки с оперативной памятью виртуальной машины, все снимки без неё. Чтобы определить количество тестов, в которых рассматриваются различные вариации удаления моментальных снимков, необходимо взять сумму

тестов на удаление по одному снимку (18) и тестов функции «удалить все снимки» (30), после чего умножить её на два (варианты снимка с оперативной памятью и без неё): итого 96. Количество тестов, в которых рассматриваются различные вариации удаления моментальных снимков в этом классе эквивалентности – это удвоенная сумма тестов на удаление по одному снимку (18) и тестов функции «удалить все снимки» (30): итого 96.

После рассмотрения тестов на удаление возможна небольшая оптимизация: объединить два теста на создание моментальных снимков (с оперативной памятью и без) с двумя тестами на функцию «удалить все снимки» (с соответствующими значениями оперативной памяти виртуально машины), в которых активным снимком будет лист. Это релевантно ввиду непродолжительного срока существования моментальных снимков. Благодаря этой оптимизации в итоговом минимальном количестве тестов класса эквивалентности «три последовательных моментальных снимка» присутствуют шесть тестов на редактирование, 14 тестов на возврат, и 96 тестов на удаление, с которыми совмещены тесты на создание: итого 116. Первоначально в алгоритме будут параллельно запущены два теста на создание и удаление, после чего на остальные можно потратить до 114 потоков. В случае провала первых двух тестов проверка всего класса эквивалентности прекращается.

Класс эквивалентности «параллельные моментальные снимки»

Заключительный класс эквивалентности для алгоритма – параллельные моментальные снимки, которые ссылаются на единый корень исходят из единого корня (рисунок 3).

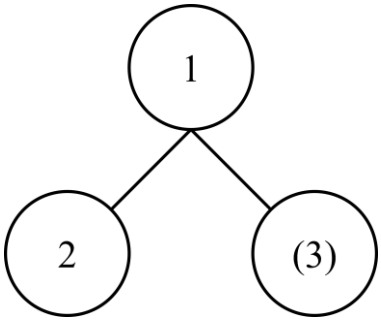

Рисунок 3 – Изображение класса эквивалентности «параллельные моментальные снимки». Цифрами обозначен порядок создания, а цифрой в скобках – активный моментальный снимок без операций возврата

В такой структуре предполагается сделать один корневой моментальный снимок, от него сделать второй снимок последовательно, после чего вернуться на корневой снимок и сделать третий снимок, который будет расположен параллельно второму и не имеет с ним прямой связи. Однако в некоторых тестах на удаление может потребоваться сделать ещё один возврат к корневому моментальному снимку и создать третий параллельный моментальный снимок. Этот класс эквивалентности позволяет проверить на минимальном количестве моментальных снимков поведение дерева снимков в случае, когда требуется одновременное наличие нескольких параллельных вариантов развития виртуальной машины. В первую очередь снова проводятся тесты на создание. Поскольку наличие оперативной памяти в моментальном снимке и в этом классе эквивалентности влияет только на затрагиваемые в рамках тестов снимки, достаточно рассмотреть две ситуации:

- все моментальные снимки с оперативной памятью виртуальной машины,
- все снимки без оперативной памяти.

Вновь два теста на создание снимков. В тестах на редактирование снимков необходимо рассмотреть два случая:

- редактирование корневого моментального снимка,
- редактирование одного параллельного снимка.

И для этих двух случаев необходимо учитывать наличие оперативной памяти, из-за чего получается четыре теста на редактирование моментальных снимков.

Для тестирования функции возврата в классе эквивалентности «параллельные моментальные снимки» можно рассмотреть пять вариантов тестов: возврат из параллельного снимка на него самого; возврат из параллельного снимка на второй параллельный снимок; возврат из параллельного снимка в корень; возврат из корня в корень; возврат из корня в один из параллельных снимков. Но возврат возможен только из активного снимка, которым после создания будет второй параллельный. Следовательно, для проведения двух тестов на возврат из корня необходимо совершить

предварительный возврат на этот корень, то есть выполнить третий тест из списка. Тем самым его можно объединить с любым из двух тестов на возврат из корня и просто проверить состояние виртуальной машины после возврата на корень, оставив четыре теста на возврат. В приведённых тестах снимки могут быть как с оперативной памятью, так и без неё, из-за чего минимальное количество тестов на функцию возврата в классе эквивалентности «параллельные моментальные снимки» равно восьми.

В тестах на удаление моментальных снимков по одному в рассматриваемом классе эквивалентности так же, как и в классе «три последовательных моментальных снимка», необходимо учитывать положение активного снимка и удаляемых моментальных снимков для проверки работы их консолидации. Для удаления одного моментального снимка рассматриваются пять случаев, для удаления двух рассматриваются семь случаев и ещё семь рассматриваются для поочерёдного удаления всех трёх моментальных снимков (рисунок 4). Как можно заметить, после удаления одного снимка все состояния дерева снимков будут встречаться во всех случаях удаления двух и трёх моментальных снимков после первой операции удаления. Это позволяет не включать эти пять тестов в список тестов данной функции. Стоит заметить, что все конечные состояния дерева моментальных снимков с удалением двух снимков встречаются в промежуточных состояниях дерева для удаления трёх снимков – перед удалением последнего снимка. Благодаря подобному вхождению финальных состояний первой и второй группы тестов на поочерёдное удаление моментальных снимков можно рассматривать только семь случаев удаления. Но для этой функции потребуется рассмотреть ещё два случая: создание одного корневого и трёх параллельных снимков с удалением активного или неактивного параллельного снимка. В этих двух тестах проверяется стабильность параллельной структуры дерева. При учёте наличия или отсутствия в снимках оперативной памяти виртуальной машины минимальное количество тестов данной функции увеличивается до 18. При тестировании функции «удалить все снимки» необходимо рассмотреть пять случаев удаления одного моментального снимка и семь случаев удаления двух снимков, которые указаны на рисунке

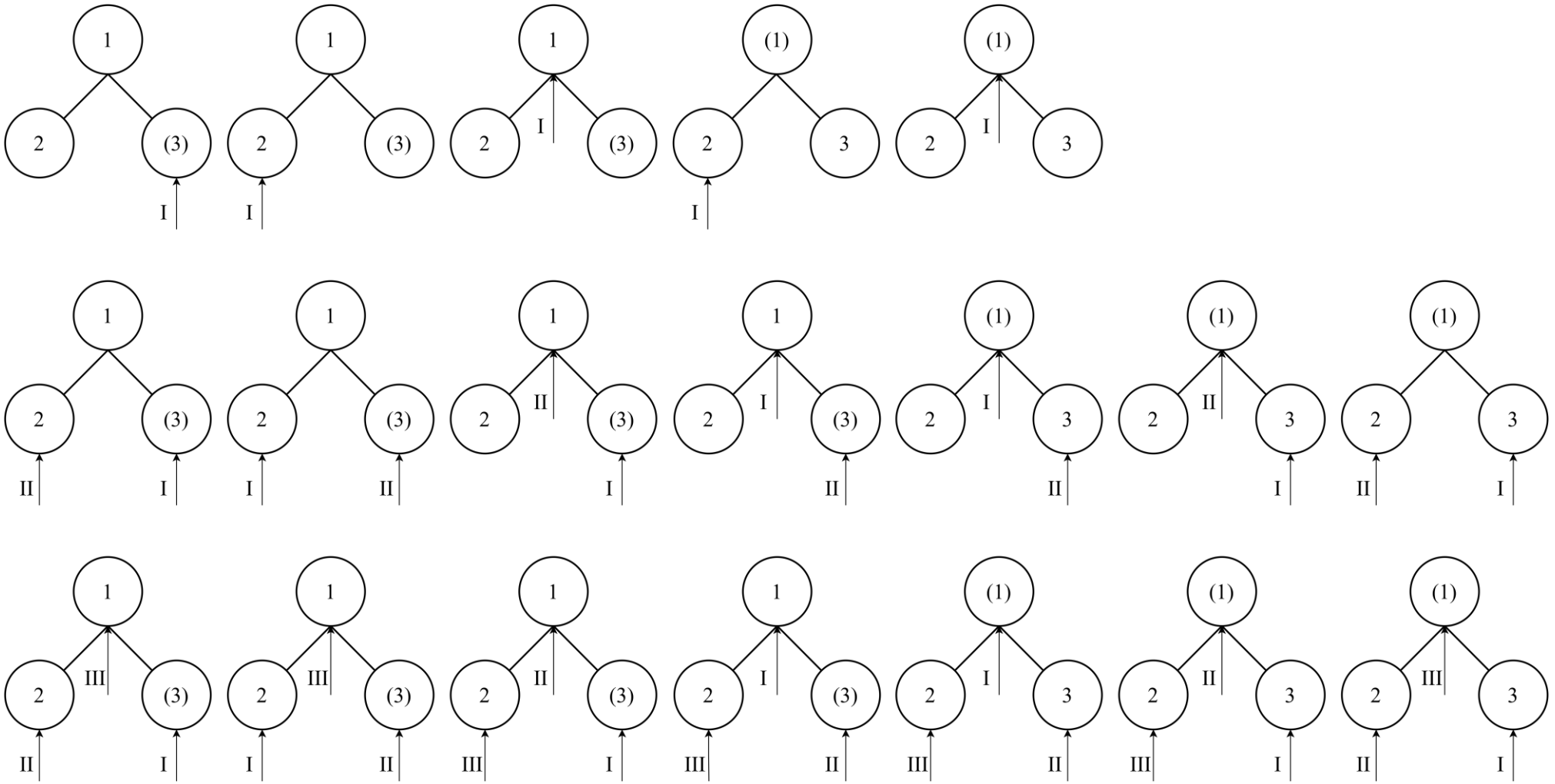

Рисунок 4 – Все возможные комбинации удаления снимков в классе эквивалентности «параллельные моментальные снимки». Арабскими цифрами обозначен порядок создания снимков, стрелками – удаляемые снимки, римскими цифрами – порядок удаления, цифрами в скобках – активный моментальный снимок

В них функция «удалить все снимки» будет активироваться после достижения конечного состояния дерева снимков в каждом указанном случае. Следует рассмотреть ещё два случая: удаление всех снимков при активном параллельном снимке и удаление всех снимков при активном корне. Таким образом при тестировании этой функции рассматриваются 14 ситуаций. С учётом наличия или отсутствия в моментальных снимках оперативной памяти на «удалить все снимки» требуется минимум 28 тестов.

Как и в предыдущем классе эквивалентности, здесь возможна небольшая оптимизация – объединение двух тестов на создание моментальных снимков (с оперативной памятью и без) с двумя тестами на функцию «удалить все снимки» (с соответствующими значениями оперативной памяти виртуально машины), в которых активным снимком будет параллельный. Благодаря этой оптимизации в итоговом минимальном количестве тестов класса эквивалентности «параллельные моментальные снимки» присутствуют четыре теста на редактирование, восемь тестов на возврат, 18 тестов на удаление по одному снимку и 28 тестов функции «удалить все снимки», с которыми совмещены тесты на создание: итого 58. Первоначально в алгоритме будут параллельно запущены два теста на создание и удаление, после чего на остальные можно потратить до 56 потоков.

Результат

В статье предложен стабильный параллельный алгоритм для проведения как ручного, так и автоматического тестирования минимальной работоспособности структур моментальных снимков виртуальной машины на основе анализа основных возможностей снимков [1, 3] и классических методов тестирования программного обеспечения [14–16]. На гипервизоре с работающими хранилищами и виртуальными машинами по предложенному алгоритму необходимо произвести суммарно 182 теста на трёх классах эквивалентности. Сначала в два потока необходимо провести два теста на создание и удаление снимков из класса «один моментальный снимок», после чего провести ещё шесть тестов того же класса, имея возможность применить до шести потоков. Далее алгоритм проводит тесты классов эквивалентности «три

последовательных моментальных снимка» и «параллельные моментальные снимки». Оба могут проходить тестирование параллельно ввиду независимости. В каждом классе изначально параллельно проходят по два теста на создание. На это можно потратить до четырёх потоков. После этого проходят остальные тесты: 114 и 56 соответственно, и на них можно выделить до 170 потоков.

Заключение

Предложен стабильный алгоритм тестирования моментальных снимков виртуальной машины. Он обеспечивает максимальное покрытие при минимальных временных затратах и обеспечивает высокий уровень независимости тестов. Это позволяет применять большое количество потоков, что полностью компенсирует необходимость создавать новую виртуальную машину перед каждым тестом и при этом делает алгоритм частично пригодным для нагрузочного тестирования гипервизора. Всё это упрощает и ускоряет процесс тестирования отечественных гипервизоров для их наискорейшей отладки и внедрения в авиационную инфраструктуру России.

Конфликт интересов

Автор заявляет об отсутствии конфликта интересов.

Conflict of interest

The author declares no conflict of interest.

Благодарности

Автор выражает глубокую признательность доценту кафедры компьютерные науки и прикладная математика Московского авиационного института Владимиру Николаевичу Лукину за помощь в редактировании статьи.

Acknowledgments

The author expresses deep gratitude to Vladimir Nikolaevich Lukin, Associate Professor of the Department of Computer Science and Applied Mathematics at the Moscow Aviation Institute, for his assistance in editing the article.

Список источников

1. Portnoy M. Virtualization Essentials. Indianapolis: John Wiley & Sons, 2012. 286 pp.
2. Chen J., Li D., Mi Z. et al. DuVisor: a User-level Hypervisor Through Delegated Virtualization, 2022, arXiv: 2201.09652. DOI: https://doi.org/10.48550/arXiv.2201.09652
3. McAdams S. Virtualization Components of the Modern Hypervisor // UNF Graduate Theses and Dissertations, 2015. pp. 113.
4. Zurich Airport Modernizes Application Networking and Enhances Customer Services // vmware.com URL: https://www.vmware.com/docs/zurich-airport-case-study (дата обращения 20.11.2025)
5. Scholten U. Virtualization in ANSPs, Airports, and Airlines: Fostering Resilience, Efficiency, and Dynamic Interoperability // skyradar.com URL: https://www.skyradar.com/blog/virtualization-in-ansps-airports-and-airlines-fostering-resilience-efficiency-and-dynamic-interoperability (дата обращения 20.11.2025)
6. Proffitt B., Golan R. Brussels Airport Company Case Study // ovirt.org URL: https://www.ovirt.org/community/user-stories/brussels-airport-case-study.html (дата обращения 22.11.2025)
7. Garg R., Sodha K., Cooperman G. A Generic Checkpoint-Restart Mechanism for Virtual Machines, 2012, arXiv: 1212.1787. DOI: https://doi.org/10.48550/arXiv.1212.1787
8. Blomer J., Berzano D., Buncic P. et al. Micro-CernVM: Slashing the Cost of Building and Deploying Virtual Machines, 2013, arXiv: 1311.2426. DOI: https://doi.org/10.48550/arXiv.1311.2426
9. Nguetchouang K., Dubuc T., Bitchebe S. et al. Virtual Disk Snapshot Management at Scale, 2022, arXiv: 2205.06842 DOI: https://doi.org/10.48550/arXiv.2205.06842
10. West M. U.S. Airport SAN Data Recovery // drivesaversdatarecovery.com URL: https://drivesaversdatarecovery.com/case-studies/case-study-us-airport-san-data-recovery (дата обращения 17.03.2026)

11. Vemula B. Dubai Airports Secures Critical Infrastructure with VMware vDefend // vmware.com URL: https://blogs.vmware.com/security/2025/07/dubai-airports-with-vdefend.html (дата обращения 22.11.2025)

12. Импортозамещение платформ серверной виртуализации // БИТ. Бизнес & Информационные технологии, 2024, № 10 URL: https://bit.samag.ru/uart/more/161 (дата обращения 15.11.2025)

13. Российские системы виртуализации // софтлайн URL: https://softline.ru/about/blog/rossiyskie-sistemy-virtualizatsii (дата обращения 17.11.2025)

14. Chauhan V.K. Smoke Testing // International Journal of Scientific and Research Publications, 2014. vol 4, no. 2, pp. 5

15. Crispin L., Gregory J. Agile Testing: A Practical Guide for Testers and Agile Teams. Crawfordsville: Addison-Wesley Professional, 2008. 576 pp.

16. Kaner C., Falk J., Nguyen H.Q. Testing Computer Software. New York: John Wiley & Sons, 1999. 480 pp.

17. Хабибулин Д. М. Оптимизация производительности автотестов: методы и инструменты // Евразийский союз ученых. Серия: технические и физико-математические науки, 2024, №. 1, с. 62–67. DOI:10.31618/ESU.2413-9335.2024.1.120-121.2055

18. Seddiki D., Galán S.G., Expósito J.E.M. et al. Sustainable expert virtual machine migration in dynamic clouds // Computers and Electrical Engineering., 2022. vol. 102, DOI: https://doi.org/10.1016/j.compeleceng.2022.108257

19. Sentanoe S., Thomas Dangl T., Reiser H.P. KVMIveggur: Flexible, secure, and efficient support for self-service virtual machine introspection // Forensic Science International: Digital Investigation., 2022. vol. 42, DOI: https://doi.org/10.1016/j.fsidi.2022.301397

20. Caviglione L., Gaggero M., Paolucci M. Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters // Soft Computing., 2020. vol. 25, pp. 12569–12588. DOI: https://doi.org/10.1007/s00500-020-05462-x

21. Мушинский А. Аэропорт «Шереметьево» меняет VMware на российское ПО // cnews.ru URL: https://www.cnews.ru/news/top/2025-10-06_aeroport_sheremetevo_zameshchaet (дата обращения 23.11.2025)

22. Королев П. Авиаотрасль массово переходит на отечественный софт // comnews URL: https://www.comnews.ru/content/235377/2024-09-25/2024-w39/1008/aviaotrasl-massovo-perekhodit-otechestvennyy-soft (дата обращения 23.11.2025)

23. Biagiola M., Stocco A., Mesbah A. et al Web Test Dependency Detection, 2019, arXiv:1905.00357. DOI: https://doi.org/10.48550/arXiv.1905.00357

24. Арапбаев Р.Н. Анализ зависимостей по данным: тесты на зависимость и стратегии тестирования. Кандидатская диссертация. Новосибирск, Институт систем информатики имени А. П. Ершова СО РАН, 2008, 116 с.

25. Bhat A., Quadri S. M. K. Equivalence class partitioning and boundary value analysis - A review // IEEE International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 1557 - 1562

26. Irawan Y., Muzid S., Susanti N., Setiawan R.R. System Testing using Black Box Testing Equivalence Partitioning (Case Study at Garbage Bank Management Information System on Karya Sentosa) // The 1st International Conference on Computer Science and Engineering Technology Universitas Muria Kudus, 2018, pp. 7. DOI: https://doi.org/10.4108/eai.24-10-2018.2280526

27. Бевзенко С.А. Исследование эффектов нагрузочного тестирования на производительность и надёжность системы // Universum: технические науки, 2023, № 114, с. 43–49. DOI: 10.32743/UniTech.2023.114.9.16001

28. Meier J.D., Farre C., Bansode P. et al. Performance Testing Guidance for Web Applications patterns & practices. Redmond: Microsoft Press, 2007. 288 pp.

References

1. Portnoy M. Virtualization Essentials. Indianapolis: John Wiley & Sons, 2012. 286 pp.

2. Chen J., Li D., Mi Z. et al. DuVisor: a User-level Hypervisor Through Delegated Virtualization, 2022, arXiv: 2201.09652. DOI: https://doi.org/10.48550/arXiv.2201.09652

3. McAdams S. Virtualization Components of the Modern Hypervisor // UNF Graduate Theses and Dissertations, 2015. pp. 113.

4. Zurich Airport Modernizes Application Networking and Enhances Customer Services // vmware.com URL: https://www.vmware.com/docs/zurich-airport-case-study (accessed 20.11.2025)

5. Scholten U. Virtualization in ANSPs, Airports, and Airlines: Fostering Resilience, Efficiency, and Dynamic Interoperability // skyradar.com URL: https://www.skyradar.com/blog/virtualization-in-ansps-airports-and-airlines-fostering-resilience-efficiency-and-dynamic-interoperability (accessed 20.11.2025)

6. Proffitt B., Golan R. Brussels Airport Company Case Study // ovirt.org URL: https://www.ovirt.org/community/user-stories/brussels-airport-case-study.html (accessed 22.11.2025)

7. Garg R., Sodha K., Cooperman G. A Generic Checkpoint-Restart Mechanism for Virtual Machines, 2012, arXiv: 1212.1787. DOI: https://doi.org/10.48550/arXiv.1212.1787

8. Blomer J., Berzano D., Buncic P. et al. Micro-CernVM: Slashing the Cost of Building and Deploying Virtual Machines, 2013, arXiv: 1311.2426. DOI: https://doi.org/10.48550/arXiv.1311.2426

9. Nguetchouang K., Dubuc T., Bitchebe S. et al. Virtual Disk Snapshot Management at Scale, 2022, arXiv: 2205.06842 DOI: https://doi.org/10.48550/arXiv.2205.06842

10. West M. U.S. Airport SAN Data Recovery // drivesaversdatarecovery.com URL: https://drivesaversdatarecovery.com/case-studies/case-study-us-airport-san-data-recovery (accessed 17.03.2026)

11. Vemula B. Dubai Airports Secures Critical Infrastructure with VMware vDefend // vmware.com URL: https://blogs.vmware.com/security/2025/07/dubai-airports-with-vdefend.html (accessed 22.11.2025)

12. Import substitution of server virtualization platforms // BIT. Business & Information Technology, 2024, vol. 10 URL: https://bit.samag.ru/uart/more/161 (accessed 15.11.2025)

13. Russian virtualization systems // softline URL: https://softline.ru/about/blog/rossiyskie-sistemy-virtualizatsii (accessed 17.11.2025)

14. Chauhan V.K. Smoke Testing // International Journal of Scientific and Research Publications, 2014. vol 4, no. 2, pp. 5

15. Crispin L., Gregory J. Agile Testing: A Practical Guide for Testers and Agile Teams. Crawfordsville: Addison-Wesley Professional, 2008. 576 pp.

16. Kaner C., Falk J., Nguyen H.Q. Testing Computer Software. New York: John Wiley & Sons, 1999. 480 pp.

17. Khabibulin D.M. Optimizing autotest performance: methods and tools // Eurasian Union of Scientists. Series: technical, physical and mathematical sciences, 2024, vol. 1, pp. 62–67. DOI:10.31618/ESU.2413-9335.2024.1.120-121.2055

18. Seddiki D., Galán S.G., Expósito J.E.M. et al. Sustainable expert virtual machine migration in dynamic clouds // Computers and Electrical Engineering., 2022. vol. 102, DOI: https://doi.org/10.1016/j.compeleceng.2022.108257

19. Sentanoe S., Thomas Dangl T., Reiser H.P. KVMIveggur: Flexible, secure, and efficient support for self-service virtual machine introspection // Forensic Science International: Digital Investigation., 2022. vol. 42, DOI: https://doi.org/10.1016/j.fsidi.2022.301397

20. Caviglione L., Gaggero M., Paolucci M. Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters // Soft Computing., 2020. vol. 25, pp. 12569–12588. DOI: https://doi.org/10.1007/s00500-020-05462-x

21. Mushinsky A. Sheremetyevo Airport is switching from VMware to Russian software // cnews.ru URL: https://www.cnews.ru/news/top/2025-10-06_aeroport_sheremetevo_zameshchaet (accessed 23.11.2025)

22. Korolev P. The aviation industry is massively switching to domestic software // comnews URL: https://www.comnews.ru/content/235377/2024-09-25/2024-w39/1008/aviaotrasl-massovo-perekhodit-otechestvennyy-soft (accessed 23.11.2025)

23. Biagiola M., Stocco A., Mesbah A. et al Web Test Dependency Detection, 2019, arXiv:1905.00357. DOI: https://doi.org/10.48550/arXiv.1905.00357

24. Arapbaev R.N. Data dependency analysis: dependency tests and testing strategies. Candidate's thesis. Novosibirsk, A. P. Ershov Institute of Informatics Systems, 2008, 116 pp.

25. Bhat A., Quadri S. M. K. Equivalence class partitioning and boundary value analysis - A review // IEEE International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 1557–1562

26. Bevzenko S.A. Study of the effects of load testing on system performance and reliability // Universum: technical sciences, 2023, vol. 114, pp. 43–49. DOI: 10.32743/UniTech.2023.114.9.16001

27. Meier J.D., Farre C., Bansode P. et al. Performance Testing Guidance for Web Applications patterns & practices. Redmond: Microsoft Press, 2007. 288 pp.

28. Irawan Y., Muzid S., Susanti N., Setiawan R.R. System Testing using Black Box Testing Equivalence Partitioning (Case Study at Garbage Bank Management Information System on Karya Sentosa) // The 1st International Conference on Computer Science and Engineering Technology Universitas Muria Kudus, 2018, pp. 7. DOI: https://doi.org/10.4108/eai.24-10-2018.2280526

Информация об авторах

Олег Романович Лисовский, тестировщик системных решений, АО «БФГ»; аспирант, Московский авиационный институт (национальный исследовательский университет); г. Москва, Российская Федерация; ORCID: https://orcid.org/0009-0000-7768-9182; e-mail: Lisoleg555@yandex.ru

Information about the authors

Oleg R. Lisovskiy, system solutions tester, joint-stock company «BFG»; post graduate student, Moscow Aviation Institute (National Research University); Moscow, Russian Federation; ORCID: https://orcid.org/0009-0000-7768-9182;
e-mail: Lisoleg555@yandex.ru

Получено 18 декабря 2025 ● Принято к публикации 20 марта 2026 ● Опубликовано 30 апреля 2026
Received 18 December 2025 ● Accepted 20 March 2026 ● Published 30 April 2026