Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Московский государственный университет путей сообщения» (МИИТ)

На правах рукописи

Удовиченко Антон Олегович

Разработка комплексной методики снижения влияния эффекта «старения» программного обеспечения на работу многомашинной вычислительной системы, построенной на основе технологии виртуальных машин

Специальность 05.13.15 - Вычислительные машины, комплексы и компьютерные сети
Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель
кандидат технических наук, доцент
Соловьев Владимир Павлович

МОСКВА – 2015

Содержание

ВВЕДЕНИЕ 4
Глава 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ОПРЕДЕЛЕНИЕ ЦЕЛИ ИССЛЕДОВАНИЯ 7
1.1. Анализ эффекта «старения» программного обеспечения 7
1.2. Анализ решений по борьбе с эффектом «старения» ПО 10
1.2.1. Общие подходы к борьбе с эффектом «старения» ПО 10
1.2.2. Анализ методов восстановления рабочего состояния программы 13
1.2.3. Анализ методов определения времени начала восстановления программы 16
1.3. Технология виртуальных машин 24
1.3.1. Анализ технологии виртуальных машин 24
1.3.2. Анализ решений по борьбе с эффектом «старения» ПО для виртуальной ИТ-инфраструктуры 28
1.4. Анализ эффективности работы ВС 30
1.5. Выводы 32
Глава 2. РАЗРАБОТКА ЭЛЕМЕНТОВ КОМПЛЕКСНОЙ МЕТОДИКИ БОРЬБЫ С ЭФФЕКТОМ «СТАРЕНИЯ» ПО 35
2.1. Разработка методов восстановления рабочего состояния программы 36
2.1.1. Разработка метода подмены виртуальной машины 36
2.1.2. Разработка метода восстановления платформы виртуализации 39
2.2. Разработка методов определения времени начала восстановления 42
2.2.1. Разработка метода определения времени начала восстановления с учетом условий работы программы 42
2.2.2. Разработка метода определения времени начала восстановления с учетом требования к работы программы 50

2.3. Разработка метода планирования процессов восстановления 55
2.3.1. Анализ задачи планирования процессов восстановления 55
2.3.2. Традиционные подходы к решению многокритериальных задач 59
2.3.3. Разработка алгоритма планирования процессов восстановления 61
2.4. Выводы 70
Глава 3. РАЗРАБОТКА КОМПЛЕКСНОЙ МЕТОДИКИ СНИЖЕНИЯ ВЛИЯНИЯ ЭФФЕКТА «СТАРЕНИЯ» ПО НА РАБОТУ МНОГОМАШИННОЙ ВС 73
3.1. Разработка общей схемы взаимодействия компонентов методики 73
3.2. Разработка политики управления процессами восстановления 78
3.3. Выводы 82
Глава 4. ПОСТАНОВКА И ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ 83
4.1 Реализация разработанной комплексной методики 83
4.2 Подготовка тестового стенда 97
4.2.2 Выбор программ с эффектом «старения» ПО и определение параметров их работы 97
4.2.1 Организация тестового стенда 102
4.3 Проведение экспериментов 105
4.3.1 Сравнительная оценка эффективности разработанной методики 106
4.3.2 Оценка влияния состояния ресурсов ВС на работу разработанной методики 112
4.4 Пример практической реализации 117
4.5 Выводы 117
ЗАКЛЮЧЕНИЕ 119
СПИСОК ЛИТЕРАТУРЫ 121

ВВЕДЕНИЕ

В настоящее время информационные технологии являются практически неотъемлемой составляющей современного общества и во многом определяют успешность ведения бизнеса. Компании и отдельные пользователи все больше зависят в своей работе от эффективности работы вычислительных систем (ВС) и ожидают своевременного и качественного получения доступа к информационным ресурсам. В таких условиях с каждым годом возрастают требования к эффективности функционирования ВС.

Сложность современных ВС и не всегда высокое качество ПО выступают одними из ключевых факторов нарушения работы ВС, которое проявляется в виде постепенного снижения производительности программного обеспечения с последующим сбоем. Рассмотрим пример. Очень распространенной причиной нарушения является не высвобождение оперативной памяти по завершению выполнения программой какой-либо задачи. Допустим, некоторый сервер, выполняющий обслуживание пользователей, выделяет 120 Кб оперативной памяти под каждый запрос, но по окончании его обработки не освобождает 20 Кб ввиду особенностей своей работы. В результате через некоторое время это приведет к исчерпанию свободной оперативной памяти и как следствие к снижению производительности сервера или сбою. О снижении производительности ПО в процессе его непрерывного выполнения с последующем сбоем также говорят как о «старении» процесса выполнения ПО (или эффекте «старения» ПО). В англоязычной научной литературе для данной проблемы используется специальный термин «software aging». Исследованию эффекта «старения» ПО посвящены работы зарубежных и отечественных исследователей, таких как Avritzer A., Castelli. V., Grottke M., Huang Y., Trivedi K., Vaidyanathan K., Ключников К. и др. Среди широко распространенных причин эффекта «старения» ПО, кроме невысвобождения оперативной памяти, выделяют невысвобождение файловых дескрипторов, накопление незавершенных

процессов, дефрагментация оперативной памяти. Практически единственным решением, позволяющим избавиться от данной проблемы, является выявление её источника и его устранение, например, на основе замены ПО или внесение изменений в его исходный код. Сложность заключается в том, что источником эффекта «старения» ПО может выступать любая программа или даже группа программ, а наличие большого количества программ в ВС и сложность их взаимодействия делает решение данной задачи крайне сложной. Кроме того, устранение одного источника не гарантирует возникновение нового и не всегда может быть реализовано в силу экономических или технических причин. Как следствие, борьба с эффектом «старения» заключается в снижении его негативного влияния на работу ВС с использованием специальных методов, известных под термином «software rejuvenation». В научных работах по данной проблематике этот термин определяет технику регулярного восстановления рабочего состояния программы с целью предотвращения серьезных сбоев в будущем, где под рабочим состоянием понимается такое состояние процесса, при котором вероятность возникновения сбоя близка к нулю. Примерами данных методов являются перезапуск программы и перезагрузка ОС.

Одной из главных трудностей борьбы с эффектом «старения» является, тот факт, что существующие решения обладают высокими издержками, связанными, прежде всего, с остановкой выполнения программы в процессе восстановления. Кроме того, за последние несколько лет одним из наиболее значительных и быстро развивающихся направлений в корпоративных вычислениях стала технология виртуальных машин. Данная технология рассматривается специалистами как одна из ключевых технологий построения вычислительных систем. Работа технологии виртуальных машин строится на основе разделения ресурсов физического компьютера между несколькими серверами. Одним из недостатков такого подхода является высокая зависимость эффективности работы серверов от работы платформы виртуализации, нарушение её работы может вести к нарушению работы всех серверов, размещенных на ней.

Проведенный анализ существующих решений по борьбе с эффектом «старения» ПО показал:

- Применение существующих решений, либо связано с нарушением работы пользователей, например, остановкой выполнения программы в процессе восстановления её рабочего состояния, либо они обладают существенными ограничениями, например, область восстановления ограничена компонентами только одной программы.
- Существующие решения обладают различными недостатками, такими как: слабая устойчивость к изменениям работы программы и/или операционной системы (ОС), отсутствие возможности учета негативного воздействия метода восстановления рабочего состояния на работу программы, высокая трудоемкость подготовки решения к работе. Кроме того, решения нацелены на улучшение одного показателя эффективности работы программы.
- Существующие решения не учитывают специфику технологии виртуальных машин при борьбе с эффектом «старения» ПО, а именно размещение нескольких серверов на одной физическом компьютере.

Подводя итог, можно сказать, что проблема снижения негативного влияния эффекта «старения» ПО на работу ВС актуальна и приоритетность данной проблемы возрастает с распространением информационных технологий и ростом требований к эффективности ВС.

Глава 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ОПРЕДЕЛЕНИЕ ЦЕЛИ ИССЛЕДОВАНИЯ

1.1. Анализ эффекта «старения» программного обеспечения

Эффект «старения» программного обеспечения представляет собой общее название явления наблюдаемого во многих программах, общей характеристикой которого является снижение производительности ПО в процессе его непрерывного выполнения с последующем сбоем. Далее приводятся результаты анализа причин возникновения эффекта «старения ПО», особенностей данного явления и его негативного воздействия на работу программного обеспечения.

Широкое распространение данной проблемы связано с возрастанием сложности ВС и увеличением темпов разработки программного обеспечения в конце 20 века и начале 21. Одним из негативных проявлений данных тенденций является недостаточный контроль качества программных продуктов со стороны разработчиков, в результате чего пользователь получает программное обеспечение с ошибками, которые проявляются уже в процессе эксплуатации.

Эффекту «старения» ПО подвержено самое разнообразное программное обеспечения, в качестве примеров можно привести следующие документированные случаи: телекоммуникационные системы [7,8], билинговые системы [9], серверные системы [10,11,12], военные [13] и космические системы [14].

Причинами эффекта «старения» ПО, главным образом, выступают ошибки программного обеспечения. В работе [15] была предложена классификация ошибок по характеру их проявления: Bohrbugs и Heisenbug. Ошибки категории Bohrbug достаточно легко обнаружить, так как проявляются при одних и тех же условиях, например, если нарушение работы ПО произошло на некотором наборе входных данных, то на этом же наборе оно произойдет в следующий раз. Ошибки данной категории могут быть легко выявлены на этапе тестирования или отладки программы. Ошибки категории Heisenbug, в противоположность категории

Bohrbug, представляет собой ошибки, которые потенциально трудно обнаружить и воспроизвести. При повторном воспроизведении условия, например, параметров работы программы и входных данных, при которых произошло нарушение программы, его повторное появление не произойдет. Причина такого поведения заключается во взаимодействии целевого ПО с другим программным и(или) аппаратным обеспечением, которое может иметь случайный характер. В работе [16] приведенная классификация была дополнена категорией Aging-related ошибок, определяющей ошибки, связанные с эффектом «старения» ПО. На рисунке 1.1 представлена взаимосвязь Aging-related ошибок с рассмотренными категориями ошибок ПО. Aging-related ошибке может быть свойственно поведение, как Bohrbug ошибки, так и Heisenbug ошибки. Например, Aging-related Bohrbug ошибка вызывает детерминированный процесс невысвобождения ресурсов ВС, т.е. при одних и тех же входных данных и параметрах работы ПО происходит не- высвобождение одного и того же объема ресурсов. Aging-related Heisenbug ошибка, в свою очередь, является причиной трудно воспроизводимого процесса исчерпания ресурсов ВС, на который оказывают влияние различные случайные факторы, например, очередность получения пакетов по сети.

Рисунок 1.1. Категории ошибок программного обеспечения

Особенность ошибок, определяющих эффект «старения» ПО, заключается в том, что каждая из них в отдельности, по сути, не являться критической, однако их накопление создает условия для возникновения сбоя. В работе [17] эффект «старения» ПО рассматривается как последовательность реализации ошибок программном обеспечение в процессе его выполнения. Большинство ошибок, определяющих эффект «старения», только изменяют внутренне состояние

системы и не вызывают её сбой, однако такая аккумуляция ошибок постепенно формирует внутреннее состояние, в котором эти ошибки уже вызывают сбой.

Широко распространенными вариантами эффекта «старения» ПО являются следующие:

- Утечка ресурсов. Суть данного эффекта «старения» обычно заключается в не высвобождение ресурсов целевой программой или компонентом ОС. Примерами являются не высвобождение памяти программой или ОС, не высвобождение файловых дескрипторов и сокетов ОС, накопление незавершенных процессов или потоков.
- Накопление ошибок. Суть данного эффекта «старения» заключается в аккумуляции ошибок, которые не связаны с конкретным ресурсом ОС, например, накопление ошибок округления в процессе работы программы.

Источником возникновения эффекта «старения» ПО может выступать как целевая программа, например, один из её процессов не производит освобождение памяти после её использования, так и внешний по отношению к программе процесс, например, один из процессов ОС.

Длительность процесса «старения» от момента запуск целевой программы до момента её выхода из строя представляет собой случайную величину. Процесс «старения» может занимать длительный промежуток времени и развиваться практически незаметно. Распределение времени от момента запуска программы до её отказа определяется интенсивностью использования ответственных за «старение» компонентов, а также их вкладом в развитие данного процесса. Например, некоторый сервер, выполняющий обслуживание пользователей, выделяет 120 Кб оперативной памяти под каждый запрос, но по окончании его обработки не освобождает 20 Кб ввиду особенностей своей работы. Если частота запросов к серверу невысокая и объем удерживаемой оперативной памяти незначителен, процесс «старения» может быть достаточно длительным при относительно большом объеме свободной оперативной памяти в ОС и большую часть времени не оказывать существенного влияния на работу сервера. В случае

большого объема удерживаемой оперативной памяти и малого объема свободной оперативной памяти в ОС время между сбоями сервера окажется небольшим.

1.2. Анализ решений по борьбе с эффектом «старения» ПО

1.2.1. Общие подходы к борьбе с эффектом «старения» ПО

Для снижения негативного воздействия эффекта «старения» ПО на работу программ в настоящее время применятся два подхода [9, 18]: реактивный и проактивный. Реактивный подход строится на основе быстрого восстановления работы программы после обнаружения сбоя, например, быстрый перезапуск программы. Решения на основе данного подхода являются универсальными и применятся для восстановления работы программ после сбоев, вызванных различными причинами, например, сбоя питания, отказа аппаратного компонента системы, ошибки оператора. Данный подход оказывается малоэффективным в случае эффекта «старения» ПО, так как он не учитывает его особенности, а именно возможное снижение качества работы (например, производительности) программы под воздействием эффекта «старения» ПО. Также реактивный подход не позволяет контролировать величину потерь (например, количество потерянных запросов), связанную со сбоем программы, так как время наступления сбоя обычно представляет собой случайную величину. Кроме того восстановление программы после сбоя может сопровождаться дополнительными издержками, например, связанные с восстановлением данных, искаженных в результате сбоя.

Проактивный подход позволяет добиться большей эффективности при борьбе с эффектом «старения» ПО, он реализуется на основе специальных методов, известных под термином «software rejuvenation». Данный термин определяет метод регулярного восстановления рабочего состояния программы (далее – метод восстановления) с целью предотвращения серьезных сбоев в будущем. Под рабочим состоянием программы понимается такое состояние её выполнения, при котором вероятность возникновения сбоя близка к нулю. Примерами таких методов являются перезапуск целевой программы и

перезагрузка ОС. Проактивный подход предполагает решение двух задач: восстановление рабочего состояния программы и определение времени начала восстановления. Целью второй задачи является определение момента запуска процесса восстановления рабочего состояния программы, при котором издержки, связанные с воздействием на целевую программу эффекта «старения» ПО и процесса восстановления будет приемлемым.

Решения по борьбе с эффектом «старения» ПО на основе проактивного подхода можно разбить на две группы, в зависимости от их интеграции с восстанавливаемой программой:

- Решения, интегрированные в восстанавливаемую программу.

  Решения данной группы предполагают встраивание механизмов восстановления непосредственно в целевую программу. Практически всегда подобные решения реализуется непосредственно разработчиками программ и учитывают особенности их функционирования.

  В работе [19] для восстановления Apache httpd сервера предлагается решение на основе обработки сигналов, реализованных в программе с её первых версий. В решении используется сигнал SIGUSR1, который сообщает главному процессу сервера, что он должен инициализировать каждый из своих подчиненных процессов, однако только тогда, когда подчиненный процесс завершит обработку активного на данный момент запроса. В данном решении время начала восстановления определяется на основе мониторинга объема оперативной памяти, занимаемой процессами сервера. Таким образом, при достижении заданного значения объема оперативной памяти главному процессу сервера посылается сигнал SIGUSR1, который перезапускает подчиненные процессы по мере завершения ими обработки активных запросов.

  Другое решение данной группы реализовано в сервере JAGR [20], который представляет собой модифицированную версию сервера приложений JBoss. В основе решения авторами заложен принцип пожертвования малым числом пользовательских соединений ради большинства других. Данное решение

основано не на маскировке сбоя для конечных пользователей, а на информировании их об этом и выполнении быстрого восстановления рабочего состояния севера для снижения негативного воздействия сбоя на остальных пользователей. При обнаружении, что какие-то компоненты сервера в ближайшее время выйдут из строя, внутри сервера активируется специальный агент, который выполняет восстановление их рабочего состояния.

Интеграция решений в восстанавливаемую программу позволяет учесть индивидуальные особенности программы и снизить издержки. Основным недостатком решений данной группы является ограниченная область восстановления - решения являются эффективными только в случае, если источником эффекта «старения» ПО выступает целевая программа.

- Автономные решения.

Решения данной группы не требуют внесения изменений в исходный код целевой программы и обычно строятся на основе универсальных методов восстановления, например, перезапуск программы или перезагрузка ОС.

В самом простом варианте автономное решение может быть построено, например, на основе планировщика задач ОС - как перезапуск программы или перезагрузка ОС через фиксированные интервалы времени. Важным является то, что время между восстановлениями должно быть меньше времени между сбоями программы. IBM Director Software Rejuvenation (DSR) [21] предоставляет выбор одного из двух методов восстановления: перезагрузка ОС или перезапуск программы. DSR поддерживает два варианта определения времени начала восстановления: 1) фиксированное время начала восстановления, 2) на основе мониторинга работы целевой программы и/или ОС, например, объема свободной оперативной памяти.

Преимуществом решений данной группы является их универсальность и применимость к уже разработанным программам. Основные же недостатки подобных решений определяются заложенными в них методами восстановления рабочего состояния программы и методами определения

времени начала восстановления.

Таким образом, решения на основе проактивного подхода включает два основных компонента – метод определения времени начала восстановления и метод восстановления рабочего состояния программы. Область применения конкретного решения и его эффективность, главным образом, определяются методами, входящими в его состав.

1.2.2. Анализ методов восстановления рабочего состояния программы

Как было отмечено раннее борьба с эффектом «старения» ПО строится на основе специальных методов, известных под термином «software rejuvenation» [9,11,16,17]. Данный термин определяет метод восстановления рабочего состояния программы с целью предотвращения серьезных сбоев в будущем. При этом под рабочим состоянием программы понимается такое состояние её выполнения, при котором вероятность возникновения сбоя близка к нулю. В данном разделе рассмотрены основные методы восстановления рабочего состояния программ, выделены их ключевые преимущества и недостатки.

Наиболее широкое распространение получили два метода восстановления: перезагрузка ОС и перезапуск целевой программы. Основным их недостатком являются временная остановка программы в процессе восстановления. Второй метод часто обеспечивает меньшую длительность остановки, однако применим только в случае если источником «старения» является целевая программа.

Снижение длительности остановки программы в процессе восстановления обычно достигается за счет согласования процесса восстановления и работы программы. Метод восстановления Micro-reboot [22] является одним из вариантов такого решения и основан на встраивании механизмов восстановления непосредственно в программу. Метод Micro-reboot основан на выборочной остановке и перезапуске компонентов программы, дальнейшая работа которых может привести к сбою. Такой подход считается эффективным по причине того,

что часто малое число компонентов программы ответственно за сбой. Реализация метода Micro-reboot имеет ряд требований к целевой программе:

1. Дробление компонентов. Время перезагрузка на уровне компонентов зависит, главным образом, от времени остановки и инициализации отдельного компонента. Поэтому микро-перезагрузка нацелена на компоненты с, как можно, меньшей длительностью перезапуска.
2. Отделимость состояний. Для обеспечения корректного восстановления, данный метод требует, чтобы целевая программа сохраняла свое текущее состояние в специальном хранилище, за пределами программы.
3. Отделимость компонентов. Для применения метода микро-перезагрузки необходимо, чтобы компоненты целевой программы не были жестко скреплены, например, не допускается применение прямых ссылок в коде программы.
4. Повторные запросы. Для безболезненного перезапуска компонентов целевой программы, взаимодействие между ними должно учитывать временные задержки, так как если компонент связан с компонентом, который сейчас восстанавливается, то он сможет получить ответ от него только после завершения его восстановления.

Плюсом данного метода является то, что он обеспечивает снижение негативного влияния процесса восстановления на работу пользователей с программой. Основными его недостатками являются, во-первых, необходимость модификации исходного кода программы, во-вторых, высокие требования к реализации программы. Кроме того, метод оказывается не эффективным, в случае, если источник эффекта «старения» ПО расположен за пределами целевой программы, например, в одном из процессов ОС.

В работе [19] также предлагается метод восстановления рабочего состояния программы на основе согласования процесса восстановлении и работы целевой программы. Суть данного метода заключается в восстановлении процессов программы, только после завершения ими задач, начатых до момента

восстановления. Работа метода строится на основе специальной обработки сигнала, реализованной в программе. При получении заданного сигнала главный процесс программы выполняет перезапуск каждого из подчиненных процессов, но только после того каждый из них завершит выполнение текущей задачи.

Данный метод обладает теми же недостатками, что и предшествующий, а именно ограниченной областью восстановления, которая включает только подчиненные процессы целевой программы, и специфическими требованиям к реализации программы, связанные с обработкой сигналов.

Широко распространенным подходом к снижению негативного воздействия процессов восстановления рабочего состояния на обслуживание пользователей программой является организация высокопроизводительного кластера серверов [23, 11, 24]. Суть данного метода заключается в исключении сервера из процесса обслуживания пользователей на время его восстановления, за счет перераспределения его нагрузки между остальными серверами кластера. С этой целью планировщик нагрузки кластера в заданное время прекращает направлять запросы на сервер, который предполагается восстановить. По истечению некоторого времени, когда все активные соединения с данным сервером будут завершены, выполняется его восстановление, например, на основе перезагрузки ОС. После завершения восстановления, планировщик нагрузки включает восстановленный сервер в обслуживание пользователей.

Данный метод позволяет избежать недоступности сервиса для пользователей и нарушения их обслуживания в процессе восстановления сервера. Однако возможность использования данного метода предполагает наличие группы серверов, объединенных в высокопроизводительный кластер. Кроме того, применение данного метода сопровождается потерей производительности, за счет дополнительной обработки запросов на стороне планировщика нагрузки [24].

Проведенный анализ методов восстановления показал, что основной проблемой восстановления рабочего состояния программы является её остановка в процессе восстановления. Снижение длительности остановки достигается за

счет согласования процесса восстановления и работы программы. Одним из распространенных вариантов данного решения является встраивание механизмов восстановления в программу, что практически невозможно в случае коммерческих программ и может привести к высоким затратам в случае программ с открытым исходным кодом. Кроме того, область восстановления таких решений ограничена целевой программой. Другой вариант снижения негативного воздействия процессов восстановления на работу пользователей с сервером предполагает развертывание высокопроизводительного кластера серверов.

1.2.3. Анализ методов определения времени начала восстановления программы

При борьбе с эффектом «старения» ПО программа подвергается негативному воздействию также со стороны процессов восстановления, величина которого определяется выбранным методом восстановления, например, метод перезагрузки ОС приводит к временной остановке программы. В результате встает задача определения времени восстановления с целью снижения негативного воздействия, как процессов восстановления, так и эффекта «старения» ПО на работу программы.

Существующие методы определения времени начала восстановления можно условно разделить на две группы:

- методы без мониторинга работы целевой программы и ОС;
- методы с мониторингом работы целевой программы и ОС.

В методах первой группы определение времени начала восстановления, главным образом, выполняется на основе представления функционирования программы через множество её рабочих состояний с оценкой интенсивности переходов между ними. Методы данной группы обычно не требуют большого набора исходных данных и ограничиваются информацией о времени пребывания программы в каждом из состояний и величиной издержек, связанных с нахождением в каждом из них.

Наиболее простым и распространенным подходом к планированию процессов восстановления является назначение некоторой даты или интервала начала восстановления [21]. Назначение времени и интервала начала восстановления соответственно устанавливается исходя из опыта оператора.

Для облегчения работы оператора и повышения точности определения времени начала восстановления применяются специальные аналитические модели, описывающие поведение программ при воздействии на них эффекта «старения» ПО и процессов восстановления. Модели между собой отличаются, главным образом, математическим аппаратом и набором возможных состояний программы. Одним из широко используемых математических аппаратов является Марковская цепь. В одной из первой работ, посвященных разработке аналитических моделей, была применена непрерывная Марковская цепь [3]. Авторами была предложена трехшаговая модель, в которой программа из рабочего состояния может перейти только в состояние, характеризующееся высокой вероятностью возникновения сбоя, после которого возможны переходы, либо в состояние сбоя, либо в состояние восстановления. Оба последних состояния возвращают программу снова в рабочее состояние. Определение времени начала восстановления авторы работы строят на основе оценки финансовых издержек, которые возникают в результате сбоя программы из-за эффекта «старения» и применения выбранного метода восстановления рабочего состояния программы.

Работа [9] послужила основой для разработки новых методов, направленных на совершенствование модели поведения программы, а также комбинированных методов.

В работе [25] представлены две аналитической модели поведения программы на основе полумарковского процесса. Набор состояний и переходов первой модели совпадает с набором, предложенным в работе [9]. Принципиальное отличие второй модели от первой заключается в том, что переход из состояния «Сбой» в «Рабочее состояние» осуществляется только через состояние

«Восстановление рабочего состояния». Таким образом, предполагается, что для перевода программы из состояния сбоя в рабочее состояние выполняется проведение операции восстановления его рабочего состояния. На основе предложенных моделей в работе предлагается непараметрический алгоритм расчета оптимального времени восстановления с точки зрения доступности программы. Одной из особенностей моделей, предложенных в данной работе, является возможность планирования процессов восстановления без определения функции распределения времени между сбоями.

Работа [26] также является развитием работы [9] и направлена на уточнение модели поведения программы под воздействием эффекта «старения» и процессов восстановления. В данной работе модель строится на основе Марковской регенерирующей стохастической сети Петри (Markov regenerative stochastic Petri net), при этом интервал времени между процессами восстановления рассматривается как некоторая фиксированная величина. При построении модели предполагается, что время от запуска/восстановления программы до момента сбоя имеет гипоэкспоненциальное распределение, а время, требующееся на запуск программы, после восстановления или сбоя имеет экспоненциальное распределение.

Основным недостатком рассмотренных методов является то, что их результаты быстро становятся непрактичными в случае наличия определенной свободы в использовании программы и/или ОС, например, установки обновления ОС, изменения объема ресурсов.

Для повышения практичности методов данной группы были разработаны комбинированные методы [27, 28], которые предполагают использование информации о работе программы, например, скорости исчерпания ресурсов.

В работе [27] предложен метод с иерархической структурой, включающей два уровня. На нижнем уровне выполняется построение модели деградации производительности, вызванной утечкой ресурсов, например, оперативной памяти. Цель модели деградации заключается в связывании поведения программы

с величиной утечки ресурсов. Набор состояний программы определяется количеством активных в данный момент независимых процессов. Моделирование поведения программы при наличии утечки ресурсов выполняется на основе непрерывной Марковской цепи. Полученная модель затем используется для оценки интенсивности сбоя программы. Полученный на нижнем уровне результат используется на втором уровне для построения модели управления процессами восстановлением программы на основе полу-Марковского процесса. Данная модель затем используется для формирования оптимального плана восстановления, как с точки зрения минимизации времени простоя программы, так и его стоимости.

В работе [16] предлагается многоуровневый подход к построению модели поведения программы на основе полумарковского процесса. На нижнем уровне выполняется построение аналитической модель, описывающей динамику изменения нагрузки программы. Таким образом, вначале определяется некоторое множество состояний нагрузки на основе кластерного анализа, а также вероятности перехода между состояниями и время нахождения в каждом из них. Для описания нагрузки используются переменные, характеризующие активность процессора (cpuContextSwitch, sysCall) и подсистемы ввода/вывода (pageIn, pageOut). Затем для каждого ресурса, оперативной памяти (свободный объем) и файла подкачки (занятый объем), на основе построенной модели определяется величина его истощения в каждом из состояний. Затем на основе полученных значений и полученной аналитической модели определяется время истощения каждого из ресурсов. Полученный результат используется на более высоком уровне для расчета оптимального интервала начала восстановления с точки зрения минимизации либо времени простоя программы, либо его издержек. Как отмечают сами авторы, разработанное ими решение является достаточно общим и полученные оценки могут не отражать реальное время между сбоями, так как оно может зависеть от множества различных факторов.

Комбинированные методы в некоторых случаях могут дать лучшие

результаты за счет учета дополнительной информации о текущем состоянии программы и/или ОС, однако ценой этого является более трудоемкая процедура их подготовки к работе.

Основными достоинствами методов первой группы является относительно невысокий объем исходных данных и возможность учета негативного воздействия на программу процессов восстановления. Однако фундаментальной проблемой данной группы является то, что их результаты быстро становятся непрактичными при наличии определенной свободы в использовании системы, например, установки обновлений ОС, изменении объема ресурсов.

Вторая группа методов основана на мониторинге одной или нескольких характеристик работы целевой программы и(или) ОС, отражающих процесс «старения», и определении момента достижения ими значения, при котором, как правило, эффективность дальнейшей работы целевой программы становится неприемлемой. Рабочими данными методов данной группы выступают, как правило, количественные характеристики ОС и отдельных процессов, например, свободный объем оперативной памяти, и категорийные данные, например, данные об ошибках из лог-файлов.

Работа [28] является одной из первых работ, посвященных исследованию возможности применения мониторинга характеристик работы ОС к решению проблемы эффекта «старения». В данной работе была предложена методология обнаружения присутствия эффекта «старения» ПО в ОС. Авторами вводится метрика эффекта «старения» ПО – «оцениваемое время исчерпания ресурса», которая показывает предполагаемое время работы ОС до полного исчерпания некоторого ресурса. Данный показатель используется для анализа ресурсов ОС, на предмет их связи с эффектом «старения» ПО, и выявления наиболее критических из них для последующего мониторинга. Оценка времени исчерпания ресурса строится на основе метода линейной регрессии - Sen’s slope estimator. Данная работа имеет практическое значение с точки зрения выявления присутствия эффекта «старения» ПО в ОС, выбора наиболее критических

ресурсов и оценки времени их исчерпания. Однако в данной работе не рассматриваются вопросы определения времени начала восстановления и учета негативного воздействия методов восстановления на работу целевой программы.

В работе [11] предлагается решение на основе построения моделей изменения характеристик ОС и/или целевой программы для последующей оценки их изменения. Построение моделей выполняется на основе следующих методов: простая линейная регрессия, линейная регрессия с h точками прерывания, линейная регрессия на логарифм данных, кусочно-линейная регрессия с k разрывами на логарифм данных. Из множества полученных моделей для каждой характеристики выбирается лучшая с точки зрения точности её описания с помощью метода Mallows' C_p или оценки взвешенной суммы квадратов отклонений. Затем выполняется мониторинг выбранной характеристики и оценка достижения ею некоторого значения, соответствующего установленному значению. Преимуществом данного метода является относительно невысокая трудоемкость его подготовки, возможность выбора между несколькими моделями и возможность использования нескольких характеристик. Основным недостатком является отсутствие возможности планирования процессов восстановления с учетом издержек выбранного метода восстановления и условий работы программы. Кроме того, подготовка моделей, главным образом, строится на основе опыта оператора, в частности определение множества рабочих характеристик.

В работе [12] исследуется применение модели авторегрессии-скользящего среднего (ARMA) к задаче прогнозирования исчерпания ресурсов ОС. На первом шаге, после сбора данных о работе программы и/или ОС, выполняется их анализ на наличие эффекта «старения» ПО. Присутствие эффекта «старения» ПО выявляется на основе анализа изменения производительности программы и/или исчерпания ресурсов в процессе её выполнения. Анализ строится на основе оценки величины тренда каждой их характеристик с помощью метода наименьших квадратов или теста Мэн-Кандела. После анализа выполняется

подготовка ARMA модели, которая включает определение порядка модели с применением автокорреляционной и частной автокорреляционной функции и оценку её точности с использованием критерия Schwarz Information Criteria (SIC). Сравнение, выполненное авторами работы, с моделями из работы [11], показало, что вычислительная сложность использования модели ARMA намного меньше, особенно в случае длительного времени исполнения программы. Основным недостатком данного метода является отсутствие возможности учета издержек используемого метода восстановления рабочего состояния программы.

В работе [29] предлагается практическое руководство для подготовки модели производительности программы. В работе основное внимание уделяют определению набора наиболее важных характеристик с точки зрения точности прогнозирования производительности и анализу зависимости результата модели от набора входных характеристик. На первом шаге подготовки модели набор исходных данных разбивает на три части: первая используется для определения параметров модели, вторая – для подтверждения её достоверности, третья – для оценки её точности. На следующем шаге выполняется определение набора наиболее значимых характеристик на основе одного из методов отбора характеристик: прямого поиска, обратного исключения, экспертной оценки и вероятностной упаковки. Затем выполняется построение моделей производительности программы с применением методов многомерной линейной и нелинейной регрессии. В завершении выполняется оценка достоверности модели и проверку её статистической значимости. Преимуществом предложенного метода является хорошо формализованная процедура подготовки модели производительности программы. Основной недостаток - отсутствие возможности учета издержек используемого метода восстановления рабочего состояния программы.

Наряду с непосредственным мониторингом работы программы и/или ОС применяются также модели изменения производительности программы в процессе её работы. Предложенный в [30] метод основывается на предположении,

что снижение производительности программы, вызванное эффектом «старения» ПО, представляет собой детерминированный процесс. При этом данный процесс зависит от одной из «рабочих» характеристик программы, которая отражает объем выполненной ею работы с момента последнего восстановления или запуска, например, количество обработанных запросов. Построение функции зависимости производительности программы от «рабочей» характеристики выполняется на основе сплайна. На первом шаге подготовки метода выбирается показатель производительности и «рабочая» характеристика. Для этого выполняется набор экспериментов для сбора значений показателей производительности и «рабочих» характеристик. Затем выполняется визуальная оценка стабильности изменения показателей производительности в зависимости «рабочих» характеристик. По результатам оценки выбирается пара показатель производительности-«рабочая» характеристика, демонстрирующая наиболее стабильную зависимость. На следующем шаге выполняется построение сплайна, который затем используется для определения времени начала восстановления.

Работа [31] представляет собой развитие подхода, заложенного в предшествующей работе. Основное внимание направлено на получение данных о работе программы в условиях реальной производственной среды. Основная идея заключается в использовании искусственно созданных групп запросов, посылаемых с высокой скоростью в процессе выполнения программы через заданные интервалы времени для оценки мгновенной производительности. Такой подход позволяет выполнить оценку производительности в условиях реальной производственной среды. Основным недостатком метода является то, что использование реализованного в нем подхода может негативно отражаться на работе пользователей, приводя к потере запросов и увеличении времени отклика. Кроме того, данный метод не гарантирует точность построенной модели, прежде всего, по причине влияния на неё самого процесса подготовки.

Основным преимуществом методов второй группы является более высокая устойчивость к изменениям в работе ОС и программы, в сравнении с методами

первой группы, однако применение большинства из них ограничено рамками исследовательских проектов. Основными недостатками является отсутствие возможности планирования процессов восстановления или трудоемкость подготовки метода к работе. Кроме того, рассмотренные методы не обеспечивают возможности учета условий работы программы.

Проведенный анализ методов определения времени начала восстановления показал наличие ряда общих недостатков. Основным недостатком методов первой группы является слабая устойчивость к изменениям в работе программы и/или ОС. Методы второй группы обладают различными недостатками, основные из которых: отсутствие возможности учета издержек процессов восстановления рабочего состояния программы и сложность подготовки метода к работе. Кроме того, методы обеих групп не учитывают условий работы программы и не обеспечивают согласование процессов восстановления различных программ.

По результатам проведенного анализа методов определения времени восстановления можно сделать вывод, что характерной чертой для обеих групп методов является наличие существенных пробелов в вопросах настройки метода, учета воздействия процесса восстановления на работу программы и взаимного влияния процессов восстановления различных программ.

1.3. Технология виртуальных машин

1.3.1. Анализ технологии виртуальных машин

За последние несколько лет одним из значительных и быстро развивающихся направлений в области построения ИС стала технология виртуальных машин. В соответствии с данными компании Gartner в 2011 году насчитывалось порядка 40% виртуализированных серверов от общего числа, тогда как к 2015 аналитическая компания ожидает повышения данного показателя до 75% [32]. Главной особенностью данной технологии является - возможность функционирования нескольких серверов на одном физическом компьютере с уровнем изоляции с точки зрения совместимости программ и информационной

безопасности близком к уровню изоляции отдельных физических компьютеров. Наиболее распространенным применением данной технологии является консолидация ресурсов большой группы физических серверов с низким коэффициентом использования в среду с изолированными и распределенными ресурсами небольшого числа серверов.

Технология виртуальных машин представляет собой технологию создания на компьютере так называемых виртуальных машин, в которые устанавливается своя собственная операционная система [33, 34]. Таким образом, виртуальная машина представляет собой программный контейнер, в котором установлена ОС со всеми необходимыми службами и программами. Физический компьютер, состоящий из реально существующих компонент аппаратного обеспечения и запускающий виртуальные машины, обычно называют хостом. Виртуальных машин на одном хосте может быть несколько, при этом каждая виртуальная машина имеет свои собственные виртуальные аппаратные компоненты: память, процессор, жесткий диск, сетевые адаптеры. Эти ресурсы предоставляются виртуальной машине за счет физических ресурсов аппаратного обеспечения хоста. В результате, виртуальная инфраструктура (комплекс систем, построенный на основе виртуальных машин, обеспечивающих функционирование всей IT-инфраструктуры) получает новые много возможностей по управлению ресурсами при сохранении существующей схемы деятельности IT-ресурсов.

Работа нескольких виртуальных машин на одном физическом компьютере обеспечивается на основе специального программного обеспечения, отделяющего виртуальные машины от аппаратного обеспечения, - гипервизор. В широком смысле, гипервизор представляет собой программу, обеспечивающую одновременное параллельное выполнение нескольких операционных систем на одном и том же хосте. Гипервизор сам по себе в некотором роде является минимальной операционной системой и предоставляет запущенным под его управлением операционным системам сервис виртуальной машины, а также управляет виртуальными машинами, выделением и освобождением ресурсов для

них.

Выделяют два типа гипервизоров [35]:

- Гипервизор первого типа - программа, исполняемая непосредственно на аппаратном уровне компьютера и выполняющая функции управления аппаратными средствами и виртуальными машинами.
- Гипервизор второго типа - специальный дополнительный программный слой, расположенный поверх ОС хоста, который в основном выполняет функции управления виртуальными машинами, а управление аппаратурой берет на себя ОС хоста.

В настоящее время доминирующе положение на рынке виртуализации серверов занимают системы на основе гипервизоров первого типа. В данных системах функции управления аппаратными средствами и виртуальными машинами разделяются между гипервизором и специальной привилегированной сервисной виртуальной машиной, работающей также под управлением гипервизора. Здесь и далее под платформой виртуализации понимается гипервизор и набор воспитательных программ, обеспечивающих контроль и управление виртуальными машинами. Например, VMware ESX Server представляет собой вариант системы на основе гипервизора первого типа и состоит из двух основных компонентов - собственно гипервизора (vmkernel) и сервисной консоли на базе ядра Linux, которая позволяет осуществлять контроль и мониторинг виртуальных машин. Другой продукт данной компании ESXi представляет компактной версией ESX Server, которая отличается от первого отсутствием сервисной консоли. Citrix XenServer представляет собой вариант системы на основе гипервизора первого типа. XenServer также имеет отдельную виртуальную машину, которая наделена привилегиями прямого доступ к аппаратному обеспечению и управления другими виртуальными машинами.

Главными преимуществами технологии виртуальных машин являются:

- Консолидация серверов и оптимизация инфраструктуры. Технология позволяет достичь более эффективного использования ресурсов, поскольку обеспечивает

объединение ресурсов инфраструктуры в единый пул.

- Сокращение расходов на физическую инфраструктуру. Технология позволяет сократить количество серверов и связанного с ним ИТ-оборудования, а также снизить потребности в обслуживании, электропитании и охлаждении.
- Повышение гибкости и скорости реагирования системы. Данная технология предлагает новый метод управления ИТ-инфраструктурой и снижает время на выполнение повторяющихся заданий, например, инициацию, настройку, отслеживание и техническое обслуживание серверов.
- Повышение доступности приложений и обеспечение непрерывности работы предприятия. Резервное копирование и миграция виртуальных сред целиком без перерывов в обслуживании может сократить периоды планового простоя и обеспечить быстрое восстановление системы в критических ситуациях.

Существующие платформы виртуализации обеспечивают возможность перемещения (миграции) виртуальных машин между хостами. Существуют два типа миграции: «холодная» и «горячая» [36, 37]. В первом случае производится останов виртуальной машины, информация о памяти и процессах передается на хост назначения, где и производится восстановление её работы. Во втором случае, процесс перемещения виртуальной машины намного сложнее в виду постоянного изменения состояния виртуальной машины в процессе перемещения. «Горячая» миграция строится на основе итеративного алгоритма, обеспечивающего передачу памяти виртуальной машины в процессе её работы. Основным преимуществом второго типа является сохранение активности виртуальной машины в процессе миграции, прерывание в среднем составляет порядка 100-150 миллисекунд и не приводит к прерыванию работы сервиса, запущенного внутри ВМ.

Однако использование виртуальных машин связано и с рядом негативных моментов. Прежде всего, размещение нескольких серверов на одном физическом оборудовании ведет к снижению их надёжности. Нарушение работы платформы виртуализации или аппаратных компонентов хоста, например, жесткого диска, может вести к нарушению работы всех серверов, размещенных на данном хосте.

Кроме того, работа технологии виртуальных машин строится на основе разделения ресурсов физического компьютера между несколькими операционными системами. В таком случае, захват пропускной способности сети или 100%-я загрузка процессора хоста одной из виртуальных машин приведёт к катастрофическому падению производительности других виртуальных машин.

1.3.2. Анализ решений по борьбе с эффектом «старения» ПО для виртуальной ИТ-инфраструктуры

В разделе 1.2.1 были рассмотрены общие подходы к борьбе с эффектом «старения» ПО, каждый из которых имеет свои особенности применительно к виртуальной ИТ-инфраструктуре. Основная проблема применения решений на основе, как реактивного, так проактивного подхода, в виртуальной ИТ-инфраструктуре заключается в высокой зависимости работы виртуальных машин от работы платформы виртуализации. Например, нарушение работы платформы виртуализации в процессе восстановления ведет к нарушению работы всех виртуальных машин, находящихся под её управлением.

Обеспечение быстрого восстановления программы, размещенной внутри виртуальной машины, после возникновения сбоя выполняется на основе службы обеспечения высокой доступности (High Availability) [38, 39]. Работа данной службы строится на основе быстрого перезапуска виртуальной машины на хосте с объемом свободных ресурсов достаточным для её размещения. Набор настроек, процесс мониторинга и перезапуска виртуальных машин могут отличаться в зависимости от производителя платформы виртуализации. Например, система управления виртуальной ИТ-инфраструктуры vSphere [39] компании VMware поддерживает возможность назначения приоритетов перезапуска виртуальным машинам, а также действия на случай изоляции хоста с виртуальными машинами. Подобные решения рассчитаны на программы, требующие высокого уровня готовности, но не критичные к времени восстановления и потери рабочего состояния, допускающие ручной перезапуск.

Исследования возможности реализации проактивного подхода на основе технологии виртуальных машин проводились ранее, однако не было представлено законченного решения по снижению негативного воздействия эффекта «старения» ПО на эффективность ВС. Предложенные разработки ограничены, главным образом, созданием отдельных методов восстановления применительно к виртуальной ИТ-инфраструктуре.

Для снижения длительности остановки виртуальных машин в процессе восстановления в работе [40] предложен метод восстановления рабочего состояния, на основе сохранения рабочих состояний в оперативной памяти на время восстановления. Восстановление рабочего состояния привилегированной виртуальной машины реализуется как подмена её образа в оперативной памяти, что позволяет снизить длительность восстановления, но требует внесения изменений в исходный код платформы виртуализации. Другим вариантом, которые рассматривается в той же работе, является остановка виртуальных машин с сохранением их рабочих состояний на жестком диске. В таком случае обеспечивается сохранность рабочих состояний виртуальных машин, однако, остановка в процессе восстановления оказывается более продолжительной.

В работе [24] предлагается общая схема решения на основе организации кластера высокой производительности. Данное решение ориентировано на восстановление серверных программ и не включает возможности восстановления платформы виртуализации. Для обеспечения процесса восстановления используются три виртуальные машины: первая виртуальная машина содержит планировщик нагрузки, вторая - сервер, который обслуживает пользователей, и третья содержит дублирующий сервер, бездействующий до момента восстановления. Первая виртуальная машина также включает компоненты сбора данных о работе сервера, оценки времени начала восстановления и координации процессов восстановления. Вторая и третья виртуальные машины включают компоненты мониторинга работы сервера и процесса восстановления. Данные о работе сервера передаются в первую виртуальную машину, где определяется

время начала восстановления. В назначенное время выполняется запуск процесса восстановления, который строится на основе перераспределения запросов планировщиком нагрузки между второй и третьей виртуальными машинами с целью освобождения второй от обслуживания пользователей. После завершения данной операции выполняется процесс восстановления сервера на основе перезапуска второй виртуальной машины. Достоинством данного решения является отсутствие потери запросов пользователей в процессе восстановления. Основными недостатками является затраты ресурсов на виртуальную машину с планировщиком нагрузки. Кроме того, использование в качестве промежуточного узла планировщика нагрузки ведет к снижению производительности сервера.

Подводя итоги, можно сделать вывод, что на сегодняшний день не существует открытого законченного решения для снижения негативного воздействия эффекта «старения» ПО на работу многомашинной ВС, построенной на основе технологии виртуальных машин.

1.4. Анализ эффективности работы ВС

Современные организации, как правило, имеют вычислительную систему, работающую по принципу "клиент-сервер" и включающую совокупность серверов, предоставляющих различные сервисы клиентам, расположенным внутри и за пределами организации. Широко востребованными большинством организаций серверами являются: Web-сервер, сервер приложений, сервер баз данных, файл-сервер, почтовый сервер. Наиболее критическим с точки зрения эффекта "старения" является именно сервер, так как от него требуется длительное время непрерывной работы, часто 24 часа в сутки и 7 дней в неделю.

Под эффективностью системы обычно понимается степень её соответствия своему назначению. Для сложных систем, какими являются ВС, эффективность не удается определить одной величиной, и поэтому ее представляют набором показателей [42,43,44,45]. Ниже приведены основные показатели и дано краткое описание:

Время отклика (мс). Под временем отклика понимается время, прошедшее с момента отправки запроса, до момента получения последнего пакета из ответа на этот запрос. Обязательными составляющими времени отклика являются ***время обработки запроса на сервере*** и ***сетевая задержка***. Время обработки запроса на сервере обычно зависит от типа запроса, мощности сервера и его загруженности на момент обработки этого запроса. Например, обработка запросов со статическим содержанием требует минимального времени, тогда как динамически создаваемые страницы требуют больше времени. Кроме того, если запрашиваемая страница включает сложные транзакции, например, проверку кредитной карты, то это может привести к значительному увеличению времени обработки. Сетевая задержка определяется задержкой распространения сигнала по каналам связи и задержкой, вносимой активным сетевым оборудованием.

Производительность (пропускная способность). Определяется количеством работы, выполненным системой за единицу времени. Обычно измеряется количеством запросов в секунду.

Коэффициент готовности (%). Определяется вероятностью нахождения системы в работоспособном состоянии в некоторый момент времени. Коэффициент готовности представляет собой отношение времени, в течение которого система имеет способность выполнять свои функции к сумме этого времени и времени вынужденных простоев, взятых за один и тот же календарный срок.

Доля обработанных запросов (%). Рассчитывается как отношение числа обработанных запросов к числу принятых запросов. Эта характеристика учитывает тот факт, что существуют моменты времени, в которых эффективная работа системы более востребована, и моменты, в которые она требуется в меньшей степени. Так, гораздо хуже возникновения сбоя в момент, когда принято большое количество запросов, нежели в другой момент, когда загруженность системы относительно невелика.

Стоимость. Стоимость обычно связана с какой-либо характеристикой

производительности, например, временем отклика или пропускной способностью, в форме отношения «цена/производительность». В стоимость включается как оборудование, так и программное обеспечение оцениваемой системы.

Эффективность ВС можно анализировать с различных точек зрения. Например, с точки зрения конечного пользователя, главные критерии – это малое время отклика и нулевое число отказов в доступе. Тогда как, с точки зрения владельца ВС часто оказывается важным высокая пропускная способность и высокая готовность. Стоит отметить, что в первом и во втором случаях первостепенным является получение пользователем доступа к ВС. Таким образом, выбор в качестве основных показателей - времени отклика и доли обработанных запросов - является вполне обоснованным, так как, по сути, они и определяют основные потребности, как со стороны пользователя, так и владельца ВС.

1.5. Выводы

В данной главе был проведен анализ предметной области исследования. На основе проведенного анализа можно сказать, что решению проблемы эффекта «старения» ПО уделяется достаточно много внимания и актуальность её решения определяется, главным образом, возрастающей сложностью ВС и качеством программного обеспечения.

Проведенный анализ существующих решений по борьбе с эффектом «старения» ПО показал, что основное внимание разработчиков сконцентрировано на применении проактивного подхода. В основе таких решений лежат две группы методов, которые и определяют их достоинства и недостатки: методы восстановления рабочего состояния программы и методы определения времени начала восстановления.

Анализ наиболее распространенных методов восстановления рабочего состояния программы показал, что основным их недостатком является остановка программы в процессе восстановления. Снижение длительности остановки

программы достигается, главным образом, за счет модификации исходного кода целевой программы. Кроме того, проведенный анализ технологии виртуальных машин выявил дополнительные трудности при решении задачи восстановления рабочего состояния программы, связанные, прежде всего, с размещением нескольких виртуальных машин на одном компьютере.

При анализе наиболее востребованных методов определения времени начала восстановления были выделены две группы методов с различными свойствами: методы на основе мониторинга целевой программы и/или ОС и без мониторинга. Основными достоинствами методов первой группы является относительно невысокий объем исходных данных и возможность учета негативного воздействия на программу процессов восстановления. Однако фундаментальной проблемой методов данной группы является то, что их результаты быстро становятся непрактичными в случае некоторой свободы в использовании целевой программы и/или ОС, например, установки обновлений ОС. Методы второй группы являются более устойчивыми к изменениям в работе целевой программы и/или ОС. Их основными недостатками являются: отсутствие возможности учета издержек процессов восстановления и сложность подготовки метода. Кроме того, методы обеих групп не учитывают условий работы программы и не обеспечивают согласование процессов восстановления различных программ.

На основе проведенного в данной главе анализа можно сказать, что проблема борьбы с эффектом «старения» ПО является предметом активных исследований, в связи с этим требуется разработка комплексной методики снижения влияния эффекта «старения» ПО на работу многомашинной вычислительной системы, построенной на основе технологии виртуальных машин. **Целью диссертационной работы является снижение доли потерянных запросов и среднего времени отклика серверов многомашинной вычислительной системы, построенной на основе технологии виртуальных машин, посредством применения разработанной комплексной методики.**

Для разработки комплексной методики и достижения поставленной цели требуется решить следующие задачи:

- Разработка методов восстановления рабочего состояния программы, которые обеспечивают восстановление вне зависимости от источника эффекта «старения» ПО, без прерывания обслуживания пользователей в процессе восстановления и не требуют изменения исходного кода программы.
- Разработка методов определения времени начала восстановления, которые учитывают негативное влияние на работу целевой программы, как эффекта «старения» ПО, так и метода её восстановления.
- Разработка метода планирования процессов восстановления рабочего состояния, который обеспечивает эффективное использование ресурсов ВС с точки зрения возможности реализации процессов восстановления и их согласование с целью улучшения двух показателей их эффективности реализации: своевременность и длительность процесса восстановления.
- Разработка общей схемы взаимодействия компонентов методики и политики управления процессами восстановления, обеспечивающие формирование целостного решения и позволяющие улучшить эффективность работы ВС по двум показателям: среднее время отклика и доля потерянных запросов.

Для оценки эффективности разработанной комплексной методики требуется выполнить её программную реализацию и провести эксперименты.

Глава 2. РАЗРАБОТКА ЭЛЕМЕНТОВ КОМПЛЕКСНОЙ МЕТОДИКИ БОРЬБЫ С ЭФФЕКТОМ «СТАРЕНИЯ» ПО

В силу отмеченных в предшествующей главе недостатков существующим решений по борьбе с эффектом «старения» ПО и особенностей технологии виртуальных машин при разработки комплексной методики борьбы с эффектом «старения» ПО для ВС, построенной на основе технологии виртуальных машин, требуется решение следующих задач:

- Восстановление рабочего состояния программы. Основными целями являются восстановление рабочего состояния программы вне зависимости от источника эффекта «старения» ПО, без прерывания обслуживания пользователей в процессе восстановления, а также отсутствие потребности в изменении исходного кода восстанавливаемой программы.
- Определение времени начала восстановления. При решении данной задачи основное внимание должно уделяться снижению негативного воздействия на работу целевой программы, как эффекта «старения» ПО, так и процессов восстановления рабочего состояния программы. Это может быть достигнуто, например, за счет учета характера изменения издержек выбранного метода восстановления рабочего состояния программы в процессе её выполнения. Кроме того, методы определения времени начала восстановления должны обладать гибкой настройкой и простой процедурой подготовки к работе.
- Планирование процессов восстановления. Решение данной задачи должно обеспечить учет взаимного влияния процессов восстановления и текущего состояния ВС, например, доступного объема ресурсов.

Самое важное, что выделенные задачи должны решаться комплексно и подчиняться одной цели – снижение негативного воздействия эффекта «старения» ПО на работу ВС. Для оценки эффективности работы ВС выбраны два показателя - среднее время отклика и доля потерянных запросов.

2.1. Разработка методов восстановления рабочего состояния программы

При решении задачи восстановления рабочего состояния программы, прежде всего, необходимо ориентироваться на наиболее распространенные и уязвимые элементы ВС с точки зрения эффекта «старения» ПО. Для ВС, построенной на основе технологии виртуальных машин, таковыми являются:

- Сервер (объект типа 1). Под сервером понимается программа, выполняющая обслуживание пользователей через сеть. Эффект «старения» ПО особенно критичен для данных программ, так как от них обычно требуется длительное время непрерывной работы, часто, - 24 часа в сутки и 7 дней в неделю.
- Платформа виртуализации (объект типа 2). Особенность платформы виртуализации заключается в том, что от неё зависит работа всех виртуальных машин, находящихся под её управлением. Соответственно нарушение работы платформы может вести к нарушению работы всех виртуальных машин, находящихся под её управлением.

Данные объекты являются достаточно разными по своей природе - объект типа 1 ориентирован на предоставление пользователям различных сервисов, в то время как объект типа 2 ориентирован на управление и контроль виртуальных машин. По этой причине для обеспечения наилучших результатов, было принято решение о разработке различных методов для каждого из них. Для объекта типа 1 предложен метод подмены виртуальной машины (Virtual Machine Substitution, VMS). Для объекта типа 2 - метод перезапуска платформы виртуализации с перемещением виртуальных машин (Virtual Machine Monitor Rejuvenation with virtual machine migration, VMMR).

2.1.1. Разработка метода подмены виртуальной машины

Разрабатываемый метод восстановления рабочего состояния программы ориентирован на объект типа 1 и должен отвечать следующим требованиям:

- Восстановление вне зависимости от источника эффекта «старения» ПО. Источник эффекта «старения» ПО может быть расположен, как в

восстанавливаемой программе, например, в одном из её процессов, так и за её пределами, например, в компоненте ОС. Ограниченная область восстановления может вести к увеличению частоты восстановления и даже к неэффективности восстановления, например, как в случае использования метода перезапуска целевого программы при расположении источника эффекта «старения» в одном из компонентов ОС. Соответственно, область восстановления в идеале должна затрагиваться целевую программу и все компоненты ОС.

- Сохранение обслуживания пользователей в процессе восстановления. Метод восстановления не должен приводить к прерыванию обслуживания пользователей в процессе восстановления.
- Отсутствие потребности в модификации исходного кода целевой программы. Внесение изменений в исходный код коммерческой программы является недопустимым, а в случае программы с открытым исходным кодом данный процесс может быть сопряжён с высокими затратами.
- Отсутствие необходимости изменения конфигурации ВС. Внесение изменений в ВС, например, включение дополнительной виртуальной машины в процесс обслуживания пользователей, может приводить не только к дополнительному расходу ресурсов, но и существенно сказываться на процессах управления виртуальными машинами.

Для выполнения этих задач был разработан метод подмены виртуальной машины. Его основной идеей является подмена виртуальной машины с целевой программой на новую виртуальную машину с постепенным переносом работы с одной на другую.

Для работы метода создается копия виртуальной машины, в которой размещается целевая программа. Копия может быть создана, например, на основе функции копирования консоли управления виртуальными машинами [37, 38]. Данная копия располагается в хранилище виртуальных машин и находится в неактивном состоянии до начала процесса восстановления. Соответственно, в это

время она не потребляет процессорное время, оперативную память и сетевой ресурс. Условно неактивную виртуальную машину будем именовать дублирующей, а виртуальную машину, выполняющую обслуживание пользователей, - основной.

Таким образом, до момента запуска процесса восстановления обслуживание пользователей выполняется основной виртуальной машиной. Процесс восстановления состоит из трёх этапов:

1. ***Начальная конфигурация виртуальных машин.*** На данном этапе выполняется поиск хоста с достаточным объемом ресурсов для запуска дублирующей виртуальной машины. После этого производится её запуск на данном хосте и согласование времени начала переноса обслуживания пользователей с основной виртуальной машины на дублирующую.
2. ***Перенос обслуживания пользователей.*** Процесс переноса обслуживания пользователей выполняется на основе перераспределения запросов между основной и дублирующей виртуальными машинами: запросы в рамках новых соединений к серверу направляются на обработку в дублирующую, а остальные – в основную. Второй этап завершается после того как работа всех пользователей с основной виртуальной машины будет завершена, т.е. все соединения пользователей с ней будут закрыты.
3. ***Заключительная конфигурация виртуальных машин.*** На данном этапе дублирующая виртуальная машина назначается основной и продолжает обслуживание пользователей, в то время как первая виртуальная машина отключается для высвобождения ресурсов ВС.

Процесс переноса обслуживания пользователей выполняется на уровне сети, что обеспечивает его независимость от восстанавливаемой программы и платформы виртуализации. Для переноса обслуживания пользователей целесообразно использование отдельной подсети, предназначенной для обмена служебной информацией [37, 38].

Задачи, выполняемые в рамках разработанного метода, были разбиты на две

группы и представлены в виде отдельных модулей:

- модуль управления процессом восстановления (модуль №1), который отвечает за начальную и заключительную конфигурации виртуальных машин и координацию процесса восстановления;
- модуль управления процессом переноса обслуживания пользователей (модуль №2), который отвечает за перераспределение запросов между виртуальными машинами.

Общая схема взаимодействия модулей разработанного метода, размещенных внутри основной и дублирующей виртуальных машин, с пользователями и целевой программой приведена на рисунке 2.1.

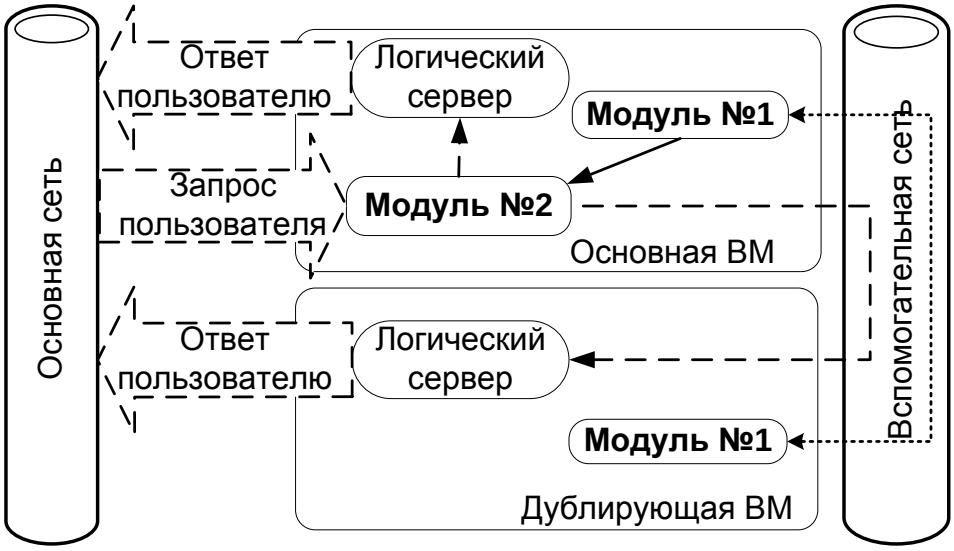

Рисунок 2.1. Общая схема взаимодействия модулей метода восстановления VMS

2.1.2. Разработка метода восстановления платформы виртуализации

Для восстановления объекта типа 2 был разработан метод восстановления, к которому предъявлялся набор требований, схожий с набором требований к ранее разработанному методу VMS:

- Восстановление вне зависимости от источника эффекта «старения» ПО. В данном случае требование приобретает особую актуальность в связи с высокой зависимостью виртуальных машин от платформы виртуализации. Кроме того,

платформа виртуализации представляет собой достаточно сложную систему, включающую как привилегированную виртуальную машину, которая содержит ОС и множество различных программ, а так и гипервизор. В соответствии с этим наиболее целесообразным является обеспечение восстановления платформы виртуализации целиком.

- Сохранение обслуживания пользователей в процессе восстановления. Метод не должен приводить к прерыванию обслуживания пользователей программами, запущенными в виртуальных машинах восстанавливаемой платформы виртуализации, в процессе восстановления.
- Отсутствие потребности внесения изменений в исходный код платформы виртуализации.

Для выполнения этих требований был предложен метод перезапуска платформы виртуализации с перемещением виртуальных машин VMMR. Основная идея метода заключается в освобождении платформы виртуализации от управления виртуальными машинами на время её восстановления.

Освобождение платформы виртуализации выполняется с использованием технологии «горячей» миграции виртуальной машины. Данная технология обеспечивает сохранение активности виртуальной машины на всём протяжении процесса её перемещения и поддерживается всеми популярными платформами виртуализации.

Восстановление методом VMMR включает в себя следующие этапы:

1. ***Определение схемы размещения виртуальных машин.*** Для сохранения активности виртуальных машин выполняется их перенос на соседние хосты с достаточным объемом ресурсов для их размещения. В соответствии с этим, встает задача определения схемы размещения виртуальных машин и параметров их перемещения. Для решения данной задачи был разработан алгоритм планирования процессов восстановления, который будет рассмотрен в разделе 2.3.
2. ***Перемещение виртуальных машин.*** В соответствии со схемой размещения

виртуальных машин выполняется их перемещение на основе технологии «горячей» миграции.

3. ***Восстановление платформы виртуализации.*** После освобождения платформы виртуализации выполняется её восстановление на основе её перезапуска.
4. ***Возвращение виртуальных машин.*** После восстановления платформы виртуализации виртуальные машины могут быть возвращены обратно на основе технологии «горячей» миграции. Данный этап является необязательным и зависит от политики управления ресурсами ВС, принятой в конкретной организации.

Пример схемы освобождения хоста приведен на рисунке 2.2.

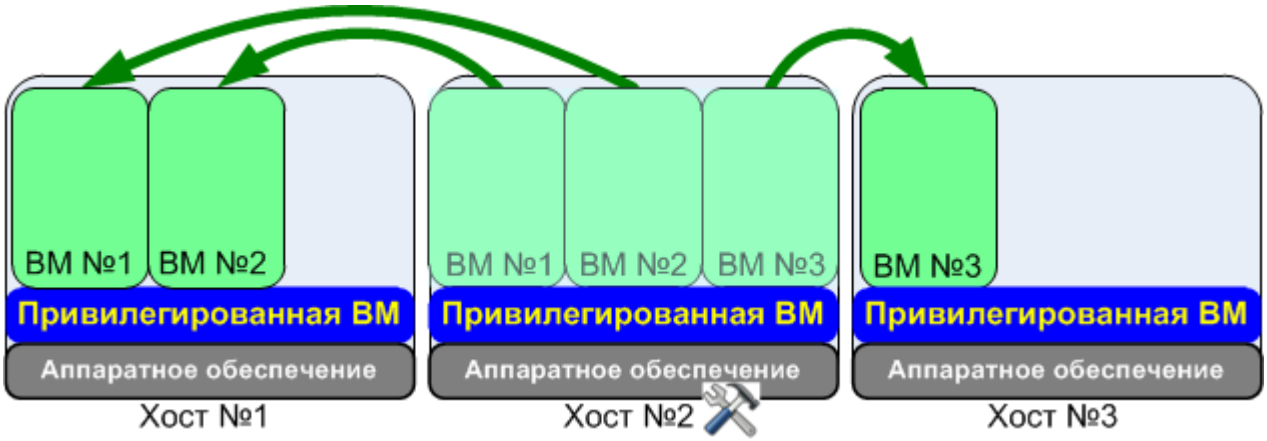

Рисунок 2.2. Пример схемы освобождения платформы виртуализации хоста №2

Одним из основных требований к работе метода является наличие хостов с достаточным объемом свободных ресурсов для перемещения виртуальных машин. В случае дефицита свободных ресурсов не все виртуальные машины будут перемещены. Соответственно, оставшиеся виртуальные машины будут перезапущены вместе с платформой виртуализации. Для большей гибкости при восстановлении платформы виртуализации в случае дефицита свободных ресурсов используются приоритеты перемещения виртуальных машин. Приоритет характеризует важность виртуальных машин для ВС с точки зрения сохранения

их в активном состоянии и задает порядок их перемещения.

2.2. Разработка методов определения времени начала восстановления

В данном разделе представлены два разработанных метода определения времени начала восстановления рабочего состояния программы. Разработка именно двух методов обусловлена принципиальным различием объектов восстановления, отмеченным ранее, а также целями, определенными в диссертационной работе, а именно улучшение двух показателей эффективности ВС, среднее время отклика и доля потерянных запросов. Для объекта типа 2 предложен метод Condition-Based Rejuvenation (CBR), учитывающий условия работы программы. Для объекта типа 1 предложен метод Requirement-Oriented Rejuvenation (ROR), учитывающий требования к эффективности работы целевой программы.

2.2.1. Разработка метода определения времени начала восстановления с учетом условий работы программы

Метод CBR ориентирован на определение времени начала восстановления объекта типа 2. Работа разработанного метода строится на основе выявления характеристики работы целевой программы и/или ОС, наиболее стабильно отражающей процесс «старения», и предусматривает определение времени начала восстановления на основе её мониторинга. Ключевой особенностью разработанного метода CBR является возможность учета изменений условий работы программы в процессе её выполнения.

Описание метода определения времени начала восстановления CBR

Метод CBR включает два этапа:

- Подготовка метода. На данном этапе выполняется определение параметров метода.
- Определение времени начала восстановления. На данном этапе выполняется непосредственно мониторинг характеристик работы целевой программы и/или ОС и расчет времени начала восстановления.

Исходной информацией при подготовке метода выступают исторические данные о работе целевой программы и/или ОС. Этап подготовка метода CBR включает следующие шаги:

1. Выявление индикатора «старения». Под индикатором «старения» понимается характеристика работы ОС или целевой программы (например, объем свободной оперативной памяти, количество файловых дескрипторов), наиболее стабильно отражающая процесс «старения». Выбор режима метода CBR (описание режимов приведено ниже).
2. Расчет нижней границы запуска процесса восстановления. Под нижней границей понимается значение индикатора «старения», начиная с которого допускается запуск процесса восстановления. Время начала восстановления впоследствии уточняется с учетом заданного режима метода.
3. Расчет момента запуска процесса определения времени начала восстановления. Данный параметр является необязательным и может быть использован для снижения вычислительных издержек, связанных с определением времени начала восстановления.

Рассмотрим каждый из перечисленных шагов более подробно:

1. Выявление индикатора «старения». Исследования [12, 28] показывают, что в процессе выполнения программы при воздействии эффекта «старения» ПО, одной или нескольким характеристикам работы восстанавливаемой программы и/или ОС свойственна выраженная тенденция изменения (тренд). В качестве индикатора «старения» целесообразным является выбор характеристики с наиболее стабильной тенденцией изменения. Выбор индикатора «старения» включает следующие шаги:

- ***Сглаживание характеристик.*** Для каждой характеристики выполняется операция сглаживания с использованием медианного фильтра [46]. Значения отсчётов внутри окна фильтра сортируются в порядке возрастания (убывания), затем из них выбирается значение, находящееся в середине упорядоченного списка, которое и поступает на выход фильтра.

- ***Исключение из рассмотрения бесперспективных характеристик.*** Выявление бесперспективных характеристик выполняется на основе анализа их трендов. Все характеристики разбиваются по времени на два равных интервала и для каждой характеристики определяется величина её тренда в целом (ВТЦ – Величина Тренда характеристики в Целом) и величина тренда второго интервала (ВТВ – Величина Тренда Второго интервала данных характеристики). Для определения трендов используется метод наименьших квадратов [47], которому свойственна относительная простая вычислительная процедура и хорошие по статистическим свойствам оценки. Из дальнейшего рассмотрения исключаются такие характеристики, для которых:
  - значение ВТЦ или ВТВ равно нулю;
  - направление тренда характеристики в целом не совпадает с направлением тренда её второй половины;
  - значение ВТВ составляет менее 25% от ВТЦ (отношение величин тренда, ОВТ):

$$OBT = \frac{|BTB|}{|BTЦ|} * 100. \qquad (2.1)$$

- ***Приведение характеристик к общей единице измерения.*** Для этого определяется диапазон изменения каждой из сглаженных характеристик $dR_i = R_i^{\min} - R_i^{\max}$, i=1,…,n – индекс характеристики. Затем для каждой характеристики определяются коэффициенты z_i, для которых выполняется равенство $dR_i * z_i = 1000$ (в абсолютных безразмерных единицах).
- ***Выбор «индикатора старения».*** Для каждой сглаженной характеристики строится её линейная модель с применением метода наименьших квадратов, выбор данного метода обоснован отмеченными ранее его свойствами. Выбор индикатора «старения» выполняется на основе анализа среднеквадратичной ошибки (далее СКО) оценки сглаженной характеристики по её линейной модели:

$$СКО_i = \sqrt{\frac{\sum_{j=1}^{K}(y_i^j - Y_i^j)^2}{K}}, i = 1,..., N, \quad (2.2)$$

где

y_i^j - значение i-ой сглаженной характеристики в момент времени t^j.

Y_i^j - значение i-ой характеристики в момент времени t^j, полученное на основе её линейной модели;

t^j - время, соответствующее j отсчету;

K – количество отсчетов во временном ряде;

N – количество рассматриваемых характеристик.

Расчеты при выборе индикатора выполняются в абсолютных безразмерных единицах. В качестве индикатора «старения» выбирается характеристика с наименьшим СКО.

2. Расчет нижней границы запуска процесса восстановления. Дальнейшее описание метода выполняется на примере индикатора «старения» с возрастающим трендом, в случае нисходящего тренда порядок действий будет тем же с учетом изменения направления тренда. При определении нижней границы принимаются во внимание режим метода (которые будут рассмотрены далее в работе) и допустимый уровень индикатора «старения». В идеале допустимый уровень должен соответствовать моменту, когда воздействие эффекта «старения» ПО на работу целевой программы становится недопустимым с точки зрения её эффективности, например, производительности программы. Расчет нижней границы запуска процесса восстановления выполняется на основе одного из следующих подходов:

- ***На основе статистических характеристик индикатора «старения».*** Допустимый уровень устанавливается равным наименьшему верхнему значению диапазона изменения индикатора «старения», сглаженного медианным фильтром. Нижняя граница запуска процесса восстановления определяется следующим образом:

$$V_b = V^{base} - V^{int} - V`, \quad (2.3)$$

где

V_b - нижняя граница запуска процесса восстановления;

V^{base} - допустимый уровень индикатора «старения»;

V^{int} - изменение сглаженного индикатора в соответствии с выбранным режимом метода;

V` - среднеквадратичная ошибка прогнозирования сглаженного индикатора «старения».

- ***На основе экспертной оценки.*** Применение экспертной оценки при определении нижней границы запуска процесса восстановления выступает как альтернатива предшествующему подходу и может быть целесообразным в отдельных случаях, например, при отсутствии надежных данных о воздействии эффекта «старения» ПО на целевую программу. В таких случаях допустимый уровень индикатора «старения» будет выбираться исходя из опыта работы специалистов с целевой программой. Обзор возможных для использования методов проведения экспертной оценки и обработки результатов приведен в [48]. Для данного подхода нижняя граница запуска процесса восстановления рассчитывается по следующей формуле:

$$V_b = V_{exp}^{base} - V^{int}, \quad (2.4)$$

V_{exp}^{base} - допустимый уровень индикатора «старения», полученный на основе экспертной оценки.

3. Расчет момента запуска процесса определения времени начала восстановления. Очевидно, что определение времени начала восстановления сразу после запуска программы или её восстановления ведет к ненужному расходу вычислительных ресурсов. Чтобы избежать этого, процесс определения времени начала восстановления целесообразно запускать по истечении некоторого времени. При расчете момента запуска процесса определения времени начала восстановления принимается во внимание средняя длительность работы

программы до сбоя (или восстановления):

$$T_a = T_{rej}^{avg} / 2, \quad (2.5)$$

где T_{rej}^{avg} - средняя длительность работы программы до сбоя (или восстановления).

После завершения подготовки метода вступает в действие второй этап. Логика работы метода CBR на втором этапе состоит в следующем:

- До момента начала определения времени начала восстановления выполняется мониторинг индикатора «старения». При достижении времени T_a запускается процесс определения времени начала восстановления.
- После запуска процесса определения времени начала восстановления до запуска процесса восстановления осуществляется мониторинг индикатора «старения», сглаживание его значений на основе медианного фильтра и расчет времени достижения сглаженным индикатором значения V_b :

$$T_b = S(V_b, \mathrm{dt}), \quad (2.6)$$

где

S - функция преобразования значения сглаженного индикатора «старения» V_b ;

dt - интервал прогнозирования сглаженного индикатора «старения».

Преобразование значения сглаженного индикатора «старения» строится на основе линейной экстраполяции его значений за период времени, соответствующий трем интервалам прогнозирования. Выбор большого интервала прогнозирования, с одной стороны, может обеспечить заблаговременное планирование процессов его восстановления, с другой стороны, может вести к снижению точности прогнозирования. В соответствии с этим, интервал прогнозирования наиболее целесообразно выбирать с учетом режима работы метода: для режима №1 – среднее время между двумя последовательными интервалами времени восстановления; для режима №2 - не менее средней длительности процесса восстановления.

При использовании нескольких индикаторов «старения» выбирается наименьшее время запуска процесса восстановления рассчитанное по выбранным

индикаторам.

Режимы метода определения времени начала восстановления CBR

Как показывает практика [42,43,44,45,85] достаточно часто работа пользователей с ВС является неравномерной и сопровождается интервалами низкой активности, вплоть до её отсутствия. Достаточно часто высокая активность пользователей проявляется в дневные часы и практически сходит на нет в ночное время. Как результат учет данного обстоятельства ведет к уменьшению негативного воздействия метода восстановления рабочего состояния программы на обслуживание пользователей, связанное, прежде всего, с потерей запросов в результате остановки программы в процессе восстановления. С целью снижения количества потерянных запросов метод CBR предусматривает учет изменения условий работы программы, связанных с активностью пользователей в процессе выполнения восстанавливаемой программы. Разработанный метод CBR предусматривает два режима:

Режим №1. С учетом заданных интервалов времени восстановления. Применение данного режима предполагает наличие периодов времени с низкой активностью пользователей при работе с ВС, что характеризуется малым количеством запросов к ВС в единицу времени. Например, такой интервал может быть назначен на ночные часы, которые в некоторых организациях характеризуются малым числом запросов к ВС.

Реализация данного режима в методе выполняется на основе введения параметра V^{int}, который характеризует изменение сглаженного индикатора «старения» между двумя последовательными заданными интервалами времени восстановления, от начала первого до завершения второго:

$$V^{int} = V_{avg}^{int} + V``, \qquad (2.7)$$

где

V_{avg}^{int} - математическое ожидание изменения сглаженного индикатора;

V'' - среднеквадратичное отклонение изменения сглаженного индикатора.

Поставляя V^{int} в выражение (2.3) или (2.4) и используя выражение (2.6) выполняется расчет времени T_b. После достижения времени T_b процесс восстановления запускается в ближайший интервал времени восстановления.

Интервалы времени для запуска процесса восстановления определяются на основе статистических данных обслуживания пользователей ВС, а именно количества запросов пользователей к ресурсам ВС в единицу времени, с использованием методов экспертной оценки. Обзор возможных для использования методов проведения экспертной оценки и обработки результатов приведен в [48]. Основное условие при выборе интервалов времени восстановления заключается в следующем - средняя длительность процесса восстановления не должна превышать заданные интервалы времени восстановления:

$$\begin{cases} t_1 \le T_{rej} \\ T_{rej} + t_{ДЛ} \le t_2 \end{cases}, \qquad (2.8)$$

где

$[t_1, t_2]$ - заданный интервал времени восстановления;

$t_{ДЛ}$ - средняя длительность процесса восстановления;

T_{rej} - время начала процесса восстановления.

Режим №2. Без учета заданных интервалов времени восстановления. Характер изменения активности пользователей таков, что не представляется возможным выделить периоды времени с наименьшим количеством запросов к ресурсам ВС сопоставимых со средней длительностью процесса восстановления. В таком случае время начала восстановления рассчитывается без дополнительных условий, т.е. значение параметра V^{int} принимается равным нулю. При использовании данного режима принимается во внимание только средняя длительность процесса восстановления:

$$T_{rej} = T_b - t_{ДЛ}. \qquad (2.10)$$

2.2.2. Разработка метода определения времени начала восстановления с учетом требования к работы программы

Основная идея разработанного метода ROR заключается в определении времени начала восстановления с учетом требований к работе целевой программы по двум показателям эффективности: время обработки запросов и коэффициент готовности. Установленные значения требований характеризуют приемлемый уровень изменения каждого из показателей эффективности. Работа метода ROR строится на основе мониторинга объема работы, выполненной программой с момента её запуска, а именно количества обработанных запросов, и определения времени начала восстановления с точки зрения выполнения требований по выбранным показателям эффективности целевой программы.

Разработанный метод ROR включает два этапа:

- Подготовка метода. На данном этапе выполняется определение параметров работы метода.
- Определение времени восстановления. На данном этапе выполняется непосредственно мониторинг работы целевой программы и расчет времени начала восстановления.

На первом этапе выполняется построение функции изменения времени обработки запросов от объема работы, выполненной программой, и формируется модель ARMA (Autoregressive Moving-Average) для прогнозирования объема работы программы во времени. На втором этапе выполняется мониторинг объема работы, выполненной программы, и расчет времени начала восстановления на основе данных, полученных на этапе подготовки.

Рассмотрим более подробно подготовку разработанного метода ROR. Объем работы, выполненной программой, представляет собой возрастающую функцию от времени и, как показывают исследования, данная характеристика обеспечивает хорошие результаты для отслеживания процесса «старения» программы [30,31]. Объем работы серверной программы удобно измерять количеством обработанных запросов, так как данная характеристика является

легко доступной и может быть получена как на стороне программы, так и за её пределами. Построение функции изменения времени обработки запросов от объема работы выполняется на основе исторической данных с использованием регрессионного сплайна со штрафной функцией [50]. Применение сплайна хорошо подходит для работы с зашумленными данными и предоставляет возможность управления величиной сглаживания, кроме того, данный метод быстр и прост в настройке. Построение функции сплайна $S(t)$ выполняется на основе минимизации следующего выражения:

$$p * \sum_{i=1}^{n} w_i (z_i - S(t_i))^2 + (1-p) \int \frac{d^2 S}{dt^2} dt, \qquad (2.11)$$

где

t_i - значение i-ого отсчета;

z_i - значения экспериментальных данных для i-ого отсчета;

$S(t_i)$ - значение сглаживающий функции сплайн для i-ого отсчета;

w_i - вес i-ого отсчета;

p – коэффициент сглаживания.

Коэффициент сглаживания определяет степень соответствия значений сплайна исходным данным. В случае p=0 будет найдена линейная регрессия, а именно сформирована прямая линия на основе метода наименьших квадратов. Выбор p=1 приведет к полному соответствию между значениями сплайна и исходными данными, а именно будет выполнена интерполяция кубическим сплайном. Наибольший интерес представляет значение коэффициента сглаживания соответствующее $p = 1/(1+h^3/6)$, где h – среднее значение интервала между отсчетами. Данное значение часто обеспечивает наиболее оптимальное соответствие сплайна исходных данных, и позволяет исключить необходимость подбора значения коэффициента сглаживания.

Для прогнозирования количества запросов используется модель ARMA, которая хорошо зарекомендовала себя в широком круге задач при

прогнозировании временных рядов, более подробное её описание и порядок её настройки приведено в [49]. При подготовке модели используется критерий BIC (Bayesian Information Criterion), выбор которого обоснован тем, что, как показывают исследования, он хорошо подходит при большом объеме рабочих данных [51], что соответствует нашему случаю.

Таким образом, на этапе подготовки метода выполняется подготовка функции зависимости времени обработки запросов от количества обработанных запросов (далее «функция времени обработки запросов») и модель ARMA для прогнозирования количества обработанных запросов (далее «модель прогнозирования количества запросов»). На данном этапе также задаются требования к коэффициенту готовности и времени обработки запросов, которые представляют собой наименьшее и наибольшее приемлемое значение для каждого из показателей, соответственно. Кроме того, задается критическое значение времени обработки запросов, определяющее максимальное допустимое время обработки запросов, при достижении которого немедленно запускается процесс восстановления.

На втором этапе выполняется мониторинг количества запросов, обработанных целевой программой, и определение момента достижения программой установленных требований к её эффективности. С этой целью на основе функции времени обработки запросов и модели прогнозирования количества запросов выполняется оценка достижения показателем времени обработки запросов заданному требования к нему. На основе полученной оценки выполняется расчет коэффициента готовности. Логика определения времени начала восстановления на втором этапе состоит в следующем:

- время начала восстановления устанавливается равным времени, когда расчетное время обработки запросов (b_1) достигает установленного требования по данному показателю (a_1), при условии, что расчетный коэффициент готовности (b_2) больше или равен соответствующему ему требованию (a_2): $b_1 = a_1,\ b_2 \geq a_2$.

- в остальных случаях происходит нарушение хотя бы одним из показателей установленных требований. Для таких случаев был разработан специальный алгоритм определения времени начала восстановления, рассмотренный ниже.

Особенность поведения показателей эффективности (время обработки запросов и коэффициент готовности) под воздействием эффекта «старения» ПО заключается в том, что попытка улучшения одного из них ведет к ухудшению другого. В связи с этим требуется поиск компромиссного решения, при котором обеспечивается наименьшее отклонение выбранных показателей эффективности от установленных к ним требований. Задача минимизации нарушений требований является задачей многокритериальной оптимизации. Существует несколько подходов к решению таких задач [52]. В разработанном алгоритме используется метод идеальной точки, суть которого заключается в поиске решения ближайшего к некоторой заданной точке. Часто в качестве координат идеальной точки выбирается сочетание наилучших значений, в разработанном алгоритме идеальная точка задается требованиями, установленными к работе программы по выбранным показателям эффективности. Расстояние между произвольными точками пространства критериев поиска определяется по формуле:

$$r(X,Y) = \min_{i=1,\dots,L} |x_i - y_i|, \qquad (2.12)$$

где $X(x_1,\dots,x_L)$ и $Y(y_1,\dots,y_L)$ – произвольные точки пространства критериев поиска.

Работа алгоритма определения времени начала восстановления строится на основе последовательного приближения решения к оптимальному, при котором расстояние до идеальной точки, рассчитанное по формуле (2.12), является минимальным. На каждой итерации алгоритма выполняется расчет промежуточных значений показателей, которые представляют наилучшее решение на данной итерации. Также на каждой итерации выполняется расчет значений (далее - расчетные значения показателей) каждого из показателей, которые будут иметь место при достижении каждым из показателем найденных

ранее промежуточных значений.

Алгоритм включает следующие шаги:

1. ***Расчет значений показателей на текущей итерации.*** На первой итерации расчетные значения по каждому показателю принимаются равными их значениям на момент достижения показателем «время обработки запросов» установленного для него требования. На последующих итерациях расчетных значений показателей определяются на основе функции времени обработки запросов и модели прогнозирования количества запросов, полученных на этапе подготовки, и промежуточных значений показателей, полученных на предшествующей итерации:

$$A^t = R(B^{tmp}) \text{ и } B^t = P(A^{tmp}), \quad (2.13)$$

где

A^{tmp} и B^{tmp} - промежуточные значения показателей, время обработки запросов и коэффициент готовности, соответственно;

A^t и B^t - расчетные значения показателей итерации t, время обработки запросов и коэффициент готовности, соответственно;

$R(B^{tmp})$ и $P(A^{tmp})$ - преобразования промежуточных значений показателей в расчетные с использованием функции времени обработки запросов.

2. ***Проверка завершения поиска.*** Работа алгоритма завершается в случае, если дальнейший поиск не ведет к улучшению результата, т.е. уменьшению расстояния до точки, заданной требованиями к показателям эффективности:

$$r((A^t, B^t),(A_{rq}, B_{rq})) \geq r((A^{t-1}, B^{t-1}),(A_{rq}, B_{rq})), \quad (2.14)$$

где A_{rq} и B_{rq} - значение требований к показателям, время обработки запросов и коэффициент готовности, соответственно.

В случае завершения выполняется переход на шаг 4. В противном случае поиск продолжается.

3. **Расчет промежуточных значений показателей.** Промежуточные значения вычисляются следующим образом:

$$A^{tmp} = A_{rq} + \frac{(A^t - A_{rq}) + (B_{rq} - B^t)}{2} \quad (2.15)$$

и

$$B^{tmp} = B_{rq} - \frac{(A^t - A_{rq}) + (B_{rq} - B^t)}{2}. \quad (2.16)$$

Выполняется переход на шаг 1.

4. ***Расчет времени начала восстановления.*** В качестве результата принимается время, соответствующее среднему времени по обоим расчетным показателям предшествующей итерации:

$$T^{вс} = (T_1(A^{t-1}) + T_2(B^{t-1}))/2, \quad (2.17)$$

где

$T_1(A^{t-1})$ - время начала восстановления, полученное на основе значения расчетного времени обработки в предшествующей итерации,

$T_2(B^{t-1})$ - время начала восстановления, полученное на основе значения расчетного коэффициента готовности предшествующей итерации.

2.3. Разработка метода планирования процессов восстановления

2.3.1. Анализ задачи планирования процессов восстановления

Целью диссертационного исследования является улучшение эффективности ВС, а именно снижение количества потерянных запросов (Z) и времени отклика сервера (Y).

Основным фактором, определяющим количество потерянных запросов, при борьбе с эффектом «старения» ПО является метод восстановления рабочего состояния программы. Для разрабатываемой комплексной методики были подготовлены два метода восстановления рабочего состояния программы VMS и VMMR, которые обеспечивают возможность восстановления программы без потери запросов. Однако как было отмечено при разработке данных методов, реализация процессов восстановления на основе этих методов определяется возможностью перемещения/размещения виртуальных машин, непосредственно

задействованных в процессе восстановления, например, в случае метода VMMR таковыми являются виртуальные машины, размещенные на восстанавливаемом хосте. В случае не возможности перемещения таких виртуальной машины происходит их остановка, что ведет к потере запросов. Таким образом, основным показателем определяющим потерю запросов для методов восстановления VMS и VMMR выступает количество размещенных/перемещенных виртуальных машин (M^d), непосредственно задействованных в процессе восстановления объекта d - $Z(M^d)$.

Повышение времени отклика сервера, прежде всего, определяется продолжительность негативного воздействия эффекта «старения» ПО. Для разрабатываемой комплексной методики были подготовлены два метода определения времени начала восстановления CMR и ROR, которые учитывают негативного воздействия на целевую программу как эффекта «старения» ПО, так и выбранного метода восстановления её рабочего состояния. Однако начало процесса восстановления в расчетное время может быть невозможно по причине взаимного влияния процессов восстановления различных программ и влияния состояния ресурсов ВС на процесс восстановления. Например, в процессе восстановления одной из платформ виртуализации размещение на ней виртуальных машин, для реализации процесса восстановления другой платформы виртуализации, будет не допустимо. В таком случае возникает задержка запуска процесса восстановления, которая определяется разницей t^d_{calc} (время запуска процесса восстановления объекта d, рассчитанное на основе метода определения времени начала восстановления) и t^d_{pl} (время запуска процесса восстановления объекта d, определенное с учетом взаимного влияния процессов восстановления). Данная задержка увеличивает продолжительность воздействия эффекта «старения» ПО на целевую программу.

Очевидным вариантом снижения взаимного влияния процессов восстановления и повышение количества размещенных виртуальных машин в

каждом процессе восстановления является более эффективное использование свободных ресурсов хостов для размещения виртуальных машин. Здесь важно отметить, что технология виртуальных машин основана на разделение ресурсов отдельного физического хоста между виртуальными машинами. Данная особенность делает невозможным запуск виртуальной машины при наличии достаточного объема свободных ресурсов ВС, но распределенного между несколькими хостами. Например, в случае если свободный объем ресурсов, соответствующий требованиям виртуальной машины, распределен между двумя хостами поровну. Поэтому более эффективное использование свободных ресурсов хостов может быть достигнуто за счет их перераспределением между хостами. Однако остановка ВС или отдельных её компонентов для перераспределения ресурсов недопустима и сопровождается высокими издержками, прежде всего, потерей запросов. Поэтому перемещение виртуальных машин между хостами должно выполняться с сохранением их активности. Для этой цели применима технология «горячей» миграции, которая обеспечивает возможность перемещения виртуальной машины с сохранением её активности и поддерживается наиболее распространенными платформами виртуализации. В таком случае длительность процесса восстановления T_{rj}^{d} объекта d будет возрастать в зависимости от времени, затраченного на перераспределение ресурсов ВС. Более подробно расчет длительности процесса восстановления для каждого из методов восстановления рабочего состояния, предусмотренных в разрабатываемой методике, приведен в разделе 2.3.3. Длительность процесса восстановления так же как и своевременность запуска процесса определяет продолжительность негативного воздействия эффекта «старения» ПО на целевую программу. Таким образом, можно выделить основные показатели, определяющие влияние эффекта «старения» ПО на среднее время отклика - своевременность и длительность процесса восстановления - $Y(t_{calc}^{d}, t_{pl}^{d}, T_{rj}^{d})$.

Таким образом, задача планирования процессов восстановления представляет собой задачу многокритериальной оптимизации:

$$\varphi(Y(t_{calc}^{d}, t_{pl}^{d}, T_{rj}^{d}), Z(M^{d})) \rightarrow min. \quad (2.18)$$

Кроме того, необходимо выделить ряд особенностей задачи, связанных с реализацией процессов восстановления:

- по мере реализации запланированных процессов восстановления выполняется добавление в план новых процессов восстановления;
- по мере работы объектов восстановления параметры их восстановления (например, время начал восстановления) или распределение ресурсов в ВС могут меняться, что приводит к необходимости проверки параметров остальных процессов восстановления (например, время начал восстановления, порядок перемещения виртуальных машин, новое место размещения виртуальной машины) и их корректировки в случае необходимости.

Таким образом, задача планирования процессов восстановления может быть рассмотрена как задача поиска места размещения виртуальных машин с целью размещения наибольшего их количества (M) для каждого объекта восстановления, а также обеспечения своевременности реализации процесса восстановления ($t_{pl} - t_{calc}$) и минимизации длительности данного процесса T_{rj}. Важно отметить, что не все из рассматриваемых критериев поиска являются сравнимыми. Очевидно, что первый критерий, количество размещенных виртуальных машин, т.е. тех чья активность сохраняется в процессе восстановления, является наиболее важным, тогда как два других, своевременность и длительность процесса восстановления, являются относительно сравнимыми.

В результате, требуется разработка алгоритма, который бы обеспечивал не только формирование плана восстановления, но и возможность быстрой его корректировки.

Для решения задач оптимизации широко используются, как точные методы, например, метод ветвей и границ, динамическое программирование [52], так и приближенные, такие как, генетический алгоритм, симуляция отжига [53]. Однако для решения задачи с определенными выше свойствами они могут оказаться неоправданными с точки зрения ресурсных издержек и скорости вычисления. В

соответствии, с чем для решения поставленной задачи был разработан алгоритм на основе разбиения сложной задачи планирования процессов восстановления на более простые подзадачи определения последовательности перемещения виртуальных машин и параметров размещения отдельной виртуальной машины.

2.3.2. Традиционные подходы к решению многокритериальных задач

Основная сложность решения многокритериальной задачи заключается в том, что единственное целесообразное решение определить, как правило, невозможно. Поэтому в процессах решения многокритериальных задач существенную роль играет некоторое лицо, принимающее решение, или специальная решающая процедура, которые осуществляют выбор одного из множества оптимально-компромиссных решений задачи. Существует несколько классических подходов к решению многокритериальных задач [52].

Как было отмечено в предшествующем разделе, особенность задачи планирования процессов восстановления заключается в различной важности критериев. Критерий «количество размещенных виртуальных машин» имеет наибольшее значение перед остальными, в то время как два других являются относительно равновесными. Далее будут рассмотрены методы определения оптимальных решений, которые используются для решения задачи планирование процессов восстановления в разработанном алгоритме.

Одним из таких методов является лексикографическое упорядочение критериев. Данный метод предполагает, что последовательность, в которой критерии перечислены, определяет их значимость в следующем смысле: каждый предшествующий критерий несравненно важнее любого из перечисляемых за ним. Синтез решения, оптимального при лексикографическом упорядочении критериев, реализуется следующим образом.

Допустим, требуется найти решение следующей двухкритериальной задачи:

$$\max_{x \in D} (f_1(x), f_2(x)), \tag{2.19}$$

где D – множество допустимых решений;

$f_1(x)$ и $f_2(x)$ - первый и второй критерий, соответственно.

Сначала решается однокритериальная задача в отношении первого критерия:

$$\max_{x \in D}(f_1(x)) \ . \tag{2.20}$$

Обозначим множество оптимальных решений первой задачи D1. Если D1 одноэлементное множество, то данное единственное решение и является оптимальным для двухкритериальной задачи. В противном случае решается следующая задача:

$$\max_{x \in D1}(f_2(x)) \ . \tag{2.21}$$

Обозначим множество оптимальных решений второй задачи D2. Если D2 одноэлементное множество, то данное единственное решение и является оптимальным для двухкритериальной задачи. В противном случае может оказаться, что оптимальных (при имеющемся лексикографическом упорядочении критериев) решений этой задачи более чем одно и все они эквивалентны.

Вторым методом является линейная свертка критериев, которая преобразует задачу многокритериальной оптимизации в однокритериальную:

$$f(\vec{x}) = \sum_{i=1}^{l} w_i * f_i(\vec{x}), \text{ при условии } \vec{x} \in D, \tag{2.22}$$

где

$f(\vec{x})$ - оценка решения;

$f_i(\vec{x})$ - частная оценка решения;

D – множество допустимых решений;

w_i - некоторые положительные числа, характеризующие относительную важность критериев (коэффициенты относительной важности).

Коэффициенты относительной важности критериев удовлетворяют следующим ограничениям:

$$\begin{cases} w_1 + w_2 + \cdots + w_l = 1 \\ w_i \in (0,1), i = 1, \dots, l \end{cases}, \tag{2.23}$$

В разработанном алгоритме второй метод определения оптимального решения используется для объединения двух критериев: своевременность и длительность процесса восстановления. Метод лексикографического упорядочивания критериев используется для поиска решения по двум критериям: 1) количество размещенных виртуальных машин и 2) критерий, полученный на основе метода линейной свертки.

2.3.3. Разработка алгоритма планирования процессов восстановления

Работа алгоритма планирования процессов восстановления строится на основе последовательного рассмотрения объектов восстановления, для каждого из которых выполняется решение двух подзадач - определение порядка перемещения виртуальных машин и определение параметров размещения виртуальной машины. Последовательность объектов формируется по возрастанию времени запуска процессов их восстановления, рассчитанного с помощью методов определения времени начала восстановления.

После рассмотрения алгоритмом очередного объекта восстановления формируется распределение ресурсов по хостам с учетом реализации процесса его восстановления. Рассмотрение следующего объекта восстановления выполняется уже с учетом изменений распределения ресурсов, вызванных процессом восстановления предшествующего объекта. Так продолжается до тех пор, пока не будут рассмотрены все объекты восстановления, или объекты, у которых время начала находятся в пределах интервала времени планирования. Ограничение на рассмотрение объектов по времени может быть использовано для снижения вычислительной нагрузки, например, изменение распределения ресурсов или параметров процессов восстановления может потребовать обновления плана восстановления. На рисунке 2.3 представлена блок-схема разработанного алгоритма планирования процессов восстановления.

Рисунок 2.3. Блок-схема алгоритма планирования процессов восстановления

Конечно, при использовании выше указанного подхода далеко не всегда будет находиться глобальный оптимум. Тем не менее, мы разбиваем сложную задачу (с множеством объектов восстановления) на две более простых (определение порядка размещения виртуальных машин в отношении одного объекта восстановления и определение параметров размещения отдельной виртуальной машины) и в то же время «не теряем из вида» взаимное влияние процессов восстановления и возможность перераспределения ресурсов ВС.

Рассмотрим теперь, как решается каждая из подзадач.

Задача определения порядка размещения виртуальных машин

Пусть объект восстановления включает группу из T виртуальных машин, для которых необходимо рассчитать порядок перемещения. Целью решения данной подзадачи является поиск наилучшей последовательности перемещения виртуальных машин с точки зрения максимизации количества виртуальных машин, которые могут быть размещены. Сложность поиска решения заключается во взаимном влиянии виртуальных машин, например, размещение одной виртуальной машины с высокими требованиями к ресурсам может привести к невозможности размещения нескольких виртуальных машин с более меньшими

требованиями суммарно соизмеримыми с ней. Для снижения вычислительных издержек и скорости расчета, решение данной подзадачи строится на основе выбора каждой последующей виртуальной машины с наименьшими требованиями к ресурсам. При размещении виртуальных машин могут учитываться их требования по нескольким типам ресурсов, что делает данную задачу задачей многокритериальной оптимизации. Для преобразования многокритериальной задачи в однокритериальную в данном случае хорошо подходит метод линейной свертки критериев [52]. Для каждой виртуальной машины осуществляется преобразование её требований по различным типам ресурсов в одну обобщенную оценку на основе метода линейной свертки:

$$V_{agr}^{i} = \sum_{j=1}^{K} c_j V_j^i, \qquad (2.24)$$

где

V_j^i - требование i-ой виртуальной машины к ресурсу типа j;

V_{agr}^{i} - обобщенная оценка требований i-ой виртуальной машины к ресурсам;

c_j - некоторые положительные числа, характеризующие относительную важность ресурса типа j.

Таким образом, выполняется упорядочивание виртуальных машин по возрастанию значений обобщенной оценки их требований к ресурсами. Блок-схема алгоритма определения порядка размещения виртуальных машин представлена на рисунке 2.4.

Рисунок 2.4. Блок-схема алгоритма определения порядка размещения виртуальных машин

Использование приоритетов в разработанном методе VMMR при восстановлении платформы виртуализации накладывает определенные условия на порядок формировании порядка размещения виртуальных машин. При наличии у виртуальных машин приоритета перемещения последовательность формируется в соответствии с ними, а для групп виртуальных машин с равными приоритетами используется разработанный алгоритм определения порядка размещения виртуальных машин.

Затем по порядку для каждой виртуальной машины выполняется определение параметров размещения.

Задача определения параметров размещения виртуальной машины

Постановка задачи определения параметров размещения виртуальной машины

Первостепенная цель планирования процессов восстановления была обозначена как максимизация количества размещенных виртуальных машин. Достижение данной цели строится за счет перераспределения ресурсов между

хостами. Перераспределение ресурсов выполняется на основе перемещения виртуальных машин между хостами с использованием технологии «горячей» миграции. Соответственно основной целью при решении задачи является: обеспечить размещение заданной виртуальной машины, выполнив при необходимости перераспределение ресурсов ВС. Также ранее были выделены два критерия, своевременность и длительность процесса восстановления, которые объединены в один критерий на основе метода линейной свертки критериев.

Своевременность d-ого процесса восстановления представляет собой величину отклонения времени t_{calc}^{d}, полученного с помощью метода определения времени начала восстановления, и времени t_{pl}^{d}, рассчитываемого при решении задачи планирования процессов восстановления с учетом взаимного влияния процессов восстановления и текущего состояния ВС.

Длительность d-ого процесса восстановления T_{rj}^{d} имеет две составляющих: базовое время восстановления и дополнительное время. Базовое время представляет собой время необходимое для выполнения основного процесса восстановления хоста, например, время перезапуска платформы виртуализации. Дополнительное время определяет длительность подготовительных работ, например, при восстановлении рабочего состояния платформы виртуализации – длительность её освобождения от виртуальных машин и длительность перераспределения ресурсов, которое может потребоваться. В случае метода VMS дополнительное время будет включать длительность запуска дублирующей виртуальной машины и длительность перераспределения ресурсов, которое может потребоваться. В простом случае, при отсутствии потребности в перераспределении ресурсов, длительность процесса восстановления платформы виртуализации представляет собой сумму длительностей «горячей» миграции каждой из виртуальных машин, размещенных на ней, и длительности её перезапуска. Если виртуальная машина не может быть перемещена, например, при недостатке ресурсов, то она перезапускается вместе с платформой

виртуализации. В таком случае, длительность процесса восстановления учитывает продолжительность отключения виртуальной машины и последующего их включения. Кроме того, процесс восстановления может включать дополнительные действия, например, возвращение виртуальной машин на исходный хост после восстановления, в таком случае длительность процесса восстановления возрастает на время, соответствующее обратного перемещения виртуальных машин.

Задача определения параметров размещения виртуальной машины может быть представлена следующей математической моделью (в двух частях):

1) Для любого i-ого хоста (i=1,…,NH) справедлива следующая система ограничений:

$$\begin{cases} \sum_{b=0}^{NV-1} Q_j^b x_i^b \le H_j^i - h_j^i, j = 1, \dots, NR \\ \forall b \in [1, NV]: x_i^b \in \{0,1\} \end{cases}, \quad (2.25)$$

где Q_j^b - требование b-ой виртуальной машины к j-ому ресурсу;

H_j^i - базовый уровень j-ого ресурса i-ом хосте;

h_j^i - требование платформы виртуализации i-ого хоста к j-ому ресурсу;

x_i^b - элемент двоичной матрицы, принимающий значение «1», если b-ая виртуальная машина размещается на i-ой хосте, «0» - в противном случае;

NH – число хостов;

NV – число виртуальных машин;

NR – число типов ресурсов, по которым учитываются при размещении виртуальных машин.

2) Целевая функция:

$$\left|t_{pl}^d - t_{calc}^d\right| + T_{rj}^d \to min, \quad (2.26)$$

где t_{calc}^d - время запуска процесса восстановления объекта d, рассчитанное на основе метода определения времени начала восстановления;

t_{pl}^d - время запуска процесса восстановления объекта d, рассчитываемое при планировании процессов восстановления;

T_{rj}^{d} - длительность процесса восстановления.

Разработанный алгоритм определения параметров размещения виртуальной машины

Для решения задачи определения параметров размещения виртуальной машины предложен итерационный алгоритм, на каждой итерации которого выполняется определение множества решений-кандидатов и проверка их реализации. Решение-кандидат представляет собой хост, перспективный для размещения заданной виртуальной машины, и связанную с ним последовательность вспомогательных перемещений виртуальных машин, обеспечивающую размещения на данном хосте заданной виртуальной машины. Множество $\{S^{q}\}$ определяет множество свободных виртуальных машин на итерации q. Множество $\{S^{tmp}\}$ определяет множество виртуальных машин и их комбинаций, перспективных для формирования множества решений-кандидатов на следующей итерации.

Алгоритм включает следующие шаги:

1. ***Формирование множества решений-кандидатов для текущей итерации.*** На начальной итерации множество решений-кандидатов $\{W^{0}\}$ принимается равным множеству хостов, доступных для размещения заданной виртуальной машины. На последующих итерациях выполняется его формирование на основе множества $\{S^{tmp}\}$ и множества решений-кандидатов предшествующей итерации.

   Каждое решение-кандидат определяется хостом, перспективным с точки зрения возможности размещения заданной виртуальной машины, объем свободных ресурсов которого определяется свободным объема ресурсов хоста до реализации решения-кандидата и требованиями к ресурсам виртуальных машин, перемещаемых с данного хоста. Элемент множества $\{W^{q}\}$ включает элементы множества $\{S^{tmp}\}$ и связанные с ними элементы

множества $\{W^{q-1}\}$. Таким образом, свободный объем ресурсов хоста, определяющего элемент новое множество $\{W^{q}\}$, рассчитывается исходя из требований к ресурсам элемента множества $\{S^{tmp}\}$, найденного на предшествующем шаге:

$$G_j^{q,h} = O_j^i + V_j^h, \text{j} = 1,\ldots, \text{NR}, \quad (2.27)$$

где $G_j^{q,h}$ – объем свободного ресурса j-ого типа для h-ого решения-кандидата итерации q;

V_j^h – требование к ресурсу j-ого типа, определенное для h-ого элемента множества $\{S^{tmp}\}$;

$O_j^{i,h}$ – объема свободного ресурса j-ого типа для i-ого хоста до перемещения виртуальных машин, закрепленных за h-ым элементом множества $\{S^{tmp}\}$.

2. ***Определение множества свободных виртуальных машин.*** На начальной итерации множество виртуальных машин $\{S^0\}$ эквивалентно множеству всех виртуальных машин. На последующих итерациях данное множество формируется из множества $\{S^{q-1}\}$ исключением из него множества $\{S^{tmp}\}$.
3. ***Расчет значения целевой функции для каждого решения-кандидата.*** Для каждого решения-кандидата выполняется расчет значения целевой функции:

$$w_u^q = \left|t_{calc}^{d,u} - t_{pl}^{d,u}\right| + T_{rj}^{d,u}, \quad (2.28)$$

где w_u^q - значение целевой функции u-ого решения-кандидата итерации q;

$t_{calc}^{d,u}$- время начала процесса восстановления объекта d, рассчитанное на основе метода определения времени начала восстановления;

$t_{pl}^{d,u}$- время начала процесса восстановления объекта d, рассчитываемое при планировании процессов восстановления;

$T_{rj}^{d,u}$- длительность процесса восстановления.

4. ***Выбор наилучшего решения-кандидата.*** Определение наилучшего решения-кандидата включает проверку двух условий:

- Проверка реализации решения-кандидата. Реализация решения-кандидата из множества $\{W^{\mathrm{q}}\}$ считается успешной, если связанные с ним процессы перераспределения ресурсов и размещения заданной виртуальной машины, не приводят к превышению базового уровня ресурсов хостов:

$$H_j^i - h_j^i - \sum_{b=0}^{M-1} Q_j^b x_i^b \geq 0,\ \mathrm{j} = 1,\ldots,\mathrm{NR}. \qquad (2.29)$$

- Решение-кандидат является наилучшим, если:

$$w_{min}^q = \min_{u \in U_q}\{w_u^q, w_{min}^{q-1}\}, \qquad (2.30)$$

где

$w_{\min}^q$ - наилучшее значение целевой функции на итерации q;

$w_{\min}^{q-1}$ - наилучшее значение целевой функции до итерации q;

U_q - множество решений-кандидатов итерации q.

5. ***Определение множества перспективных виртуальных машин $\{S^{\mathrm{tmp}}\}$.*** Для каждого элемента множества $\{S^{\mathrm{q}}\}$ выполняется поиск по множеству $\{W^{\mathrm{q}}\}$ минимум одного решения-кандидата, удовлетворяющего её требованиям к ресурсам. Результатом поиска является множество $\{S^{\mathrm{tmp}}\}$. Если множество $\{S^{tmp}\}$ содержит две и более виртуальные машины, размещенные на одном хосте, то для них формируются различные комбинации и выполняется проверка их совместной реализации. В случае успеха найденные комбинации добавляются в множество $\{S^{\mathrm{tmp}}\}$. За каждой найденной виртуальной машиной закрепляется индекс наилучшего решения-кандидатов с точки зрения значения целевой функции. Если множество $\{S^{\mathrm{tmp}}\}$ пусто, то поиск завершается – переход на шаг 6, в противном случае – переход на шаг 1.

6. Завершение алгоритма.

Блок-схема алгоритма определения параметров размещения виртуальной машины представлена на рисунке 2.5.

Рисунок 2.5. Блок-схема алгоритма определения параметров размещения виртуальной машины

Работа алгоритма определения параметров размещения виртуальной машины может быть завершена также при достижении заданного количества итераций. В таком случае в качестве окончательного решения принимается наилучшее решение-кандидат на момент завершения работы алгоритма.

2.4. Выводы

В данной главе были предложены методы, направленные на решение ряда задач, возникающих при борьбе с эффектом «старения» ПО, а именно:

- восстановление рабочего состояния программы;
- определение времени начала восстановления;
- согласование процессов восстановления.

Для решения задачи восстановления рабочего состояния программы разработаны два метода: метод VMS и метод VMMR. Первый метод предназначен для восстановления сервера, второй метод – для платформы виртуализации. Разработанные методы имеют следующие преимущества перед существующими

методами восстановления рабочего состояния программы:

- обеспечивают восстановление вне зависимости от источника эффекта «старения» ПО;
- не приводят к прерыванию обслуживания пользователей в процессе восстановления;
- не требуют модификации исходного кода восстанавливаемой программы.

Для решения задачи определения времени начала восстановления были разработаны два метода:

- метод CBR основан на мониторинге характеристик выполнения целевой программы и/или ОС и отличается от существующих методов тем, что ориентирован на определение времени начала восстановления платформы виртуализации и обеспечивает учет характера изменения условий работы платформы виртуализации;
- метод ROR основан на мониторинге объема работы, выполненной целевой программы, и отличается от существующих методов тем, что ориентирован на определение времени начала восстановления сервера и обеспечивает учет требований к эффективности его работы по двум показателям – время обработки запросов и коэффициент готовности.

Для решения задачи согласования процессов восстановления был разработан метод планирования процессов восстановления RPRR, обладает следующими преимуществами перед существующими методами:

- обеспечивает согласование процессов восстановления различных программ;
- учитывает три показателя, характеризующих эффективность процессов восстановления: количество виртуальных машин, активность которых сохраняется в процессе восстановления, своевременность и длительность процесса восстановления;
- учитывает возможность перераспределения ресурсов ВС при реализации процессов восстановления рабочего состояния.

Таким образом, были решены следующие поставленные в диссертационной работе задачи по разработке комплексной методики снижения влияния эффекта «старения» ПО на работу ВС:

- Разработка методов восстановления рабочего состояния программы, которые обеспечивают восстановление вне зависимости от источника эффекта «старения» ПО, без прерывания обслуживания пользователей в процессе восстановления и не требуют изменения исходного кода восстанавливаемой программы.
- Разработка методов определения времени начала восстановления, которые позволяют учитывать негативное влияние на работу восстанавливаемой программы, как эффекта «старения» ПО, так и метода её восстановления.
- Разработка метода планирования процессов восстановления рабочего состояния, который обеспечивает эффективное использование ресурсов ВС с точки зрения возможности реализации процессов восстановления и их согласование с целью улучшения двух показателей их реализации: своевременность и длительность процесса восстановления.

Следовательно, далее необходимо объединение разработанных методов в целостное решение, направленное на повышения эффективности работы ВС по заданным показателям, среднее время отклика и доля потерянных запросов, что и будет сделано в следующей главе.

Глава 3. РАЗРАБОТКА КОМПЛЕКСНОЙ МЕТОДИКИ СНИЖЕНИЯ ВЛИЯНИЯ ЭФФЕКТА «СТАРЕНИЯ» ПО НА РАБОТУ МНОГОМАШИННОЙ ВС

3.1. Разработка общей схемы взаимодействия компонентов методики

В предшествующей главе были разработаны методы, направленные на решение ряда задач, возникающих при борьбе с эффектом «старения» ПО: восстановление рабочего состояния программы, определения времени начала восстановления и планирование процессов восстановления. Целью данной главы является формирование комплексной методики на основе разработанных методов и направленной на повышение эффективности работы ВС по двум показателям: среднее время отклика и доля потерянных запросов. Достижение данной цели предполагает определение основных компонентов методики, схемы их взаимодействия и политики управления процессами восстановления.

Комплексная методика включает следующие наборы методов, сгруппированные по решаемым ими задачам и типам объекта восстановления:

- ***Восстановление рабочего состояния программы:***
  - Объект типа 1. Для данного типа объекта определены два метода восстановления:
    - Метод подмены виртуальной машины (метод VMS). Основным достоинством данного метода является восстановление рабочего состояния программы вне зависимости от источника эффекта «старения» ПО и работающего без прерывания обслуживания пользователей в процессе восстановления.
    - Метод перезагрузки ОС. Хотя данный метод имеет существенный недостаток, связанный с временной остановкой программы в процессе восстановления, он включен в данную методику, так как нетребователен к ресурсам, универсален и обеспечивает восстановление рабочего состояния программы вне зависимости от источника эффекта «старения»

ПО.

- Объект типа 2. Метод перезапуска платформы виртуализации с перемещением виртуальных машин (метод VMMR). Основным достоинством данного метода является восстановление вне зависимости от источника эффекта «старения» ПО и он работает без прерывания обслуживания пользователей в процессе восстановления.

- ***Определение времени восстановления программы:***
  - Объект типа 1. Время восстановления объекта определяется на основе метода ROR, который обеспечивает учет требований к работе программы по двум показателям эффективности: времени обработки запросов и коэффициенту готовности.
  - Объект типа 2. Время восстановления объекта определяется на основе метода CBR, который обеспечивает возможность учета характера изменения издержек выбранного метода восстановления в процессе выполнения программы.
- ***Согласование процессов восстановления.*** Метод планирования процессов восстановления PRRR.

В разработанной методике выделены два этапа:

- ***Подготовительный этап.*** На данном этапе выполняется расчет параметров работы методов определения времени восстановления и настройка метода восстановления VMS. Определение параметров выполняется для каждого объекта восстановления в соответствии с этапом подготовки, приведенным в описании каждого из методов определения времени начала восстановления. Настройка метода восстановления VMS заключается в установке необходимых для его работы программных компонентов, набор которых определяется конкретной реализацией данного метода. Кроме того, для метода планирования процессов восстановления могут быть заданы приоритеты перемещения виртуальных машин.

- ***Этап эксплуатации.*** На данном этапе выполняется мониторинг объектов восстановления и сбор информации о состоянии ресурсов ВС, подготовка плана восстановления, поддержание его в актуальном состоянии и реализация плана восстановления.

Основные задачи, решаемые в рамках комплексной методики, были разбиты на множество компонентов и определена общая схема их взаимодействия, представленная на рисунке 3.1. В предложенной общей схеме присутствуют следующие компоненты:

- ***Блок мониторинга работы объектов восстановления (Блок №1).*** Данный блок отвечает за сбор данных о работе объектов восстановления, необходимых для определения времени их восстановления. Конкретный набор данных определяется выбранным методом определения времени начала восстановления. Для метода CBR требуется мониторинг выбранного индикатора «старения», а также, в случае режима №2, мониторинг характеристики, определяющей изменение величины издержек метода восстановления. Для метода ROR выполняется мониторинг количества запросов, обработанных объектом с момента запуска или последнего восстановления.
- ***Блок мониторинга состояния ВС (Блок №2).*** Данный блок отвечает за сбор данных о состоянии ресурсов на хостах, распределение виртуальных машин на хостах и требования виртуальных машин к ресурсам. Конкретный перечень ресурсов соответствует перечню ресурсов, который используется системой управления виртуальной ИТ-инфраструктурой.
- ***Блок определения времени начала восстановления объектов типа 2 (Блок №3).*** Основой данного блока является метод CBR, который определяет порядок его работы. На основе данных об объекте восстановления, полученных из блока №1, выполняется оценка времени начала его восстановления.

- ***Блок определения времени начала восстановления объектов типа 1 (Блок №4).*** Данный блок предназначен для определения времени начала восстановления объекта типа 1 при использовании для восстановления его рабочего состояния метода VMS. В основе работы блока лежит метод ROR, который определяет порядок его работы.
- ***Блок согласования процессов восстановления (Блок №5).*** В основе работы блока лежит разработанный метод планирования процессов восстановления RPRR, который определяет порядок его работы. Входными данными данного блока является время начала восстановления объектов и выходные данные блока №2. Результатом работы блока является план восстановления, определяющий параметры процессов восстановления. В случае недостатка ресурсов для размещения виртуальной машины, восстановление объекта типа 1 на основе метода VMS не может быть выполнено. В таком случае для его восстановления используется метод перезагрузки ОС. В соответствии с особенностью данного метода восстановления, а именно временной остановкой объекта восстановления в процессе восстановления, выполняется пересчет времени её восстановления в блоке корректировки плана восстановления.
- ***Блок корректировки плана восстановления (Блок №6).*** Данный блок отвечает за корректировку плана восстановления в отношении объектов типа 1, для которых назначен метод восстановления «перезагрузка ОС». В основе работы данного блока лежит метод определения времени начала восстановления ROR, который определяет порядок его работы. В случае, если корректировка плана не требуется, объекты типа 1 с методом восстановления на основе перезагрузки ОС отсутствуют, то ни какие действия над планом не выполняются в данном блоке. Результатом работы данного блока является план восстановления, который размещается в хранилище актуальных планов восстановления.
- ***Блок контроля процессов восстановления (Блок №7).*** Данный блок отвечает

за реализацию актуального плана восстановления, а именно запуск и контроль выполнения процессов восстановления предусмотренными комплексной методикой методами восстановления рабочего состояния программы, а именно методом VMS, методом VMMR и методом Reboot.

- ***Блок подготовки набора параметров метода ROR (Блок А).*** Данный блок отвечает за подготовку параметров метода ROR. Порядок подготовки данных в блоке соответствует этапу подготовки метода ROR. Выходными данными этого блока для каждого объекта типа 1 являются: функция времени обработки запросов, модель прогнозирования количества запросов, среднее время остановки объекта в процессе восстановления, текущее значение коэффициента готовности, требования к эффективности работы сервера по двум показателям – время обработки запросов и коэффициенту готовности.
- ***Блок подготовки набора параметров метода CBR (Блок Б).*** Данный блок отвечает за подготовку параметров метода CBR. Порядок подготовки данных в блоке соответствует этапу подготовки метода CBR. Выходными данными блока для каждого из объектов восстановления являются: наименование индикатора «старения», время запуска процесса определения времени начала восстановления, условие запуска процесса восстановления, режим метода CBR.

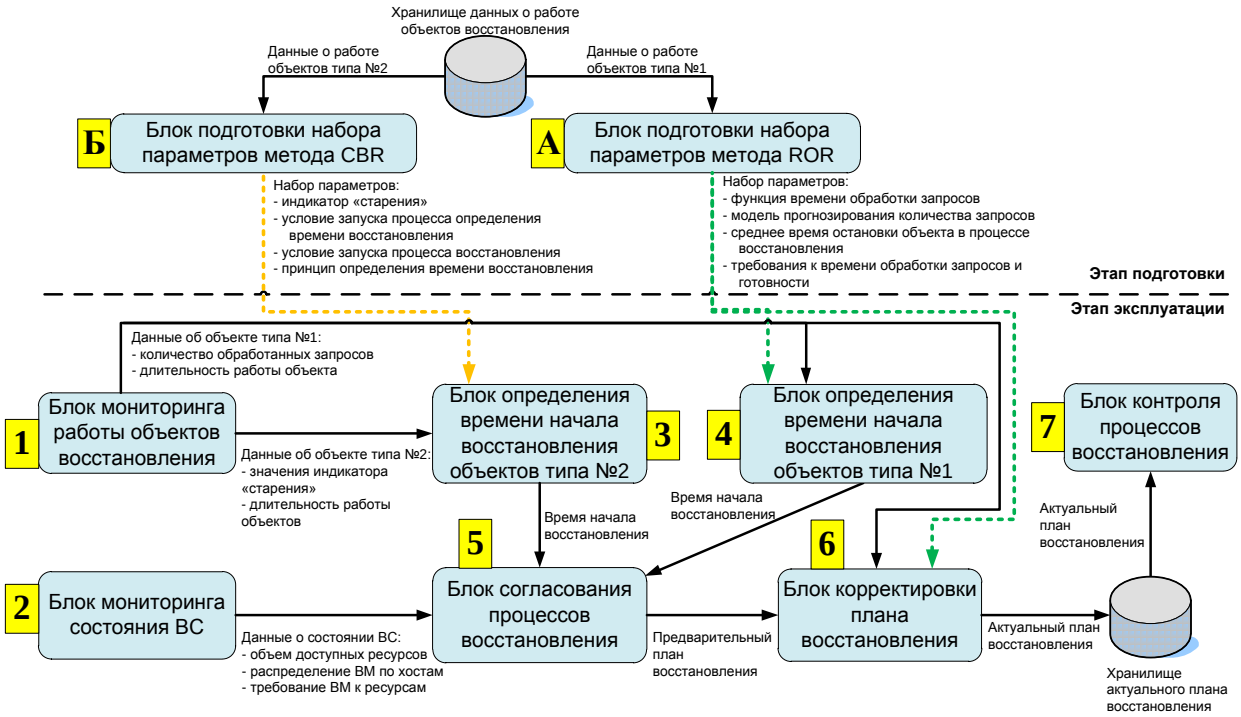

Рисунок 3.1. Общая схема взаимодействия компонентов методики

Таким образом, данный набор компонентов методики обеспечивает решение всех основных задач борьбы с эффектом «старения» ПО с учетом особенностей технологии виртуальных машин, включая сбор данных о работе объектов восстановления и контроль процессов восстановления.

3.2. Разработка политики управления процессами восстановления

В предшествующем разделе был разработан набор компонентов комплексной методики и общая схема их взаимодействия. Управление процессами восстановления предполагает решение следующих основные задач:

- подготовка плана восстановления;
- поддержание плана восстановления в актуальном состоянии;
- реализации плана восстановления.

Процесс подготовки плана восстановления включает следующие шаги:

1. ***Определение времени начала восстановления объектов.*** Данные о текущем состоянии объектов восстановления из блока №1 поступают в блок №3 и №4, для объекта типа 2 и объекта типа 1, соответственно. На основе этих данных

рассчитывается время начала восстановления объектов, которое передается в блок планирования процессов восстановления.

2. ***Формирование плана восстановления.*** В блок №5 поступают данные о времени начала восстановления из блоков № 3 и №4, и данные о текущем состоянии ресурсов ВС из блока №2, на основе которых выполняется формирование плана восстановления.
3. ***Уточнение плана восстановления.*** Из блока №5 план восстановления поступает в блок №6, где выполняется определение времени начала восстановления для объектов типа 1 с методом восстановления «перезагрузка ОС». После обработки план восстановления в блоке №6, он сохраняется в базе данных и используется для управления процессами восстановления.

Схема взаимодействия компонентов методики в процессе подготовки плана восстановления представлена на рисунке 3.2

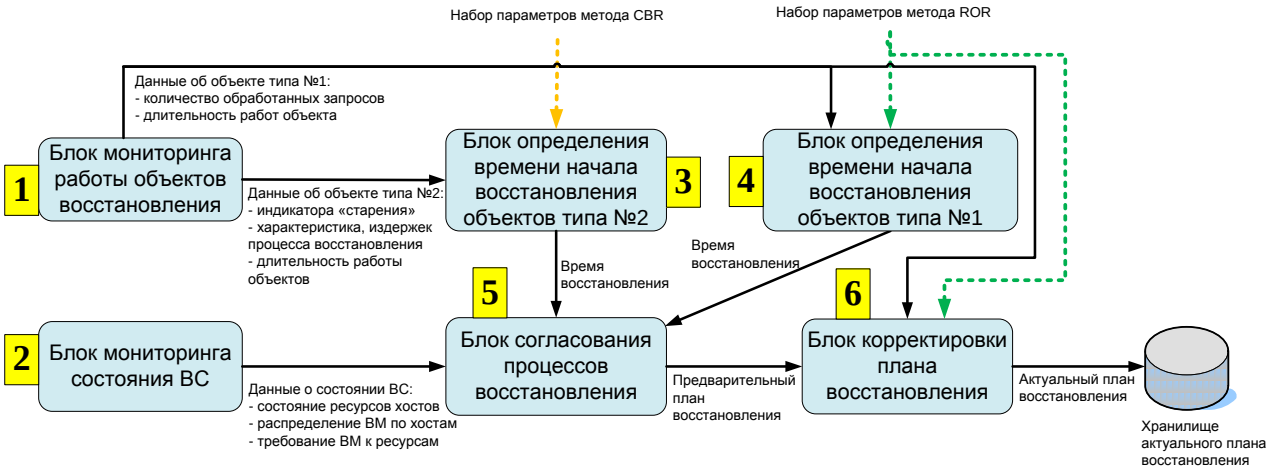

Рисунок 3.2. Схема взаимодействия компонентов методики в процессе подготовки плана восстановления

После завершения формирования плана восстановления управление процессами восстановления выполняется в одном из двух режимов:

1. ***Режим мониторинга.*** В данном режиме выполняется мониторинг работы объектов восстановления и состояние ресурсов ВС. В режиме мониторинга может выполняться обновление плана восстановления при возникновении

одного из следующих событий:

a. Добавлен или удален объект восстановления. Добавление и удаление объекта ведет к изменению условий восстановления следующих за ним объектов по времени начала их процессов восстановления. Соответственно необходимо обновление плана восстановления в отношении объектов, времена начала восстановления которых больше или равны времени начала восстановления добавленного или удаленного объекта.

b. Произошло изменение состояния ресурсов хостов, распределения виртуальных машин по хостам или их требований к ресурсам. Данные изменения ведут к формированию нового распределения ресурсов в ВС, отличного от того, для которого выполнялась разработка текущего плана восстановления. Соответственно для учета возникших изменений также требуется обновление плана восстановления.

c. Произошло изменение времени начала восстановления объекта. Данный случай аналогично первому приводит к изменению условий восстановления следующих по времени начала восстановления объектов. Соответственно в отношении таких объектов выполняется обновление плана восстановления.

Схема взаимодействия компонентов методики в режиме мониторинга представлена на рисунке 3.3

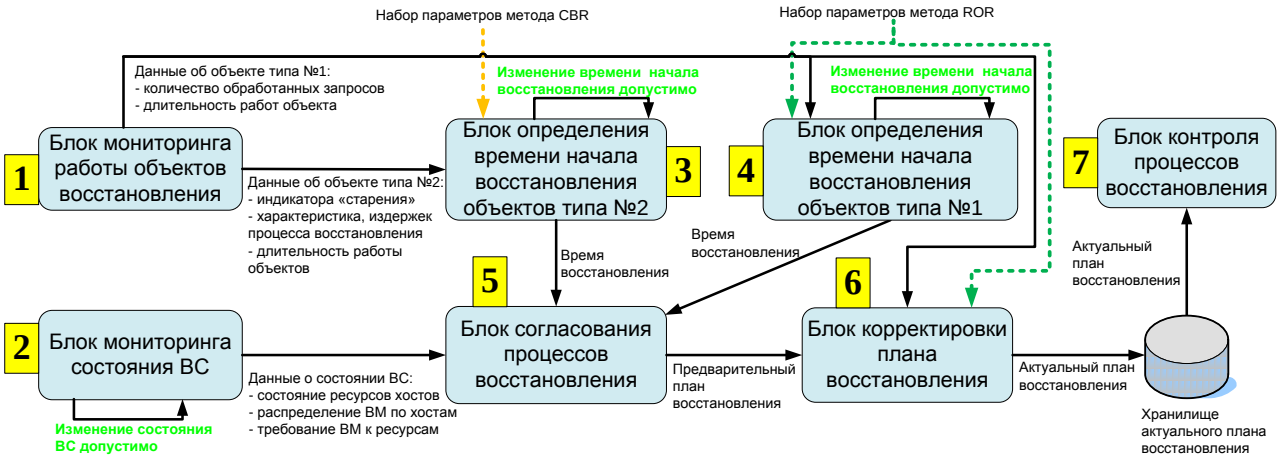

Рисунок 3.3. Схема взаимодействия компонентов методики в режиме мониторинга

Соответственно инициаторами задачи обновления плана восстановления могут выступать блоки №2, №3 и №4. Обновление плана восстановления представляет собой процесс пересчета параметров процессов восстановления, который реализуется блоками согласования процессов восстановления и корректировки плана восстановления. После завершения процесса обновления новый план замещает предшествующий и используется для управления процессами восстановления.

В режиме мониторинга также выполняется активация процессов восстановления в соответствии с планом восстановления. После запуска процесса восстановления управление процессами восстановления выполняется в режиме восстановления.

2. ***Режим восстановления.*** Данный режим активируется после запуска одного из процессов восстановления и действует до тех пор, пока существует хотя бы один активный процесс восстановления. Процессы восстановления приводят к изменению состояния ВС, например, распределению виртуальных машин по хостам, что должно провоцировать запуск процесса обновления плана восстановления в режиме мониторинга. На всем протяжении действия режима восстановления устанавливается запрет на проведение операции обновления плана восстановления. Управление процессами восстановления выполняется в циклическом режиме, а именно, после восстановления объекта выполняется его добавление в план восстановления. Таким образом, при завершении режима восстановления выполняется расчет времени начала следующего восстановления объектов, рабочее состояние которых было восстановлено, и выполняется обновление плана восстановления. После завершения режима восстановления управление процессами восстановления возвращается в режим мониторинга.

3.3. Выводы

В данной главе была разработана комплексная методика снижения влияния эффекта «старения» ПО на работу ВС, которая отличается от существующих решений тем, что учитывает особенности технологии виртуальных машин и позволяет улучшить эффективность ВС по двум показателям – среднее время отклика и доля потерянных запросов.

Для формирования комплексной методики был определен набор компонентов, обеспечивающих решение основных задач борьбы с эффектом «старения» ПО и управления процессами восстановления, и разработана схема их взаимодействия. Для обеспечения функционирования разработанной методики был предложена политика управления процессами восстановления, которая обеспечивает поддержание плана восстановления в актуальном состоянии с учетом изменения параметров восстановления отдельных объектов восстановления и своевременную его реализацию.

Таким образом, была решена следующая поставленная в диссертационной работе задача:

- разработка общей схемы взаимодействия компонентов методики и политики управления процессами восстановления, обеспечивающие формирование целостного решения и позволяющие улучшить эффективность работы ВС по двум показателям: среднее время отклика и доля потерянных запросов.

Соответственно, следующим шагом необходимо реализовать разработанную комплексную методику и выполнить серию экспериментов для оценки её эффективности по заданным критериям, что и будет сделано в следующей главе.

Глава 4. ПОСТАНОВКА И ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ

4.1 Реализация разработанной комплексной методики

В задачу диссертационной работы входит реализация разработанной комплексной методики снижения влияния эффекта «старения» ПО на работу многомашинной ВС, построенной на основе технологии виртуальных машин. Разработка программного обеспечения управления ВС - это сложная работа, состоящая из множества этапов. К программной реализации разработанной методики были сформулированы базовые требования, как с точки зрения конечного пользователя, так и с точки зрения современных подходов программирования:

- Программа должна быть разработана в соответствии с современными подходами модульного и объектно-ориентированного программирования (ООП) [54,55,56]. Прежде всего, необходимо выполнить формирования набора функционально самостоятельных подзадач, которые решаются в программе. Желательно, чтобы математическая обработка была вынесена в отдельный модуль, причем модуль должен быть независимым от других модулей, и соответственно должно обеспечиваться минимальное внесение изменения в программу при добавлении поддержки новых платформ виртуализации и ОС.
- Программа должна максимально использовать встроенные возможности операционных систем [57,58] и платформ виртуализации [33,34].
- Программа должна использовать возможности многозадачности в современных ОС [57,59]. Поскольку в программе имеет место сложная математическая обработка, то, очевидно, требуется использование как минимум двух потоков: главного потока процесса и потока математической обработки. Интерфейс программы должен своевременно обрабатывать системные события и реагировать на действия пользователя.
- Программа должна предоставлять удобный графический интерфейс для пользователя, обеспечивать возможность быстрого ввода и корректировки

параметров, загрузки исходных данных и сохранения результатов, гибкого управления процессом решения задачи.

- Программа должна корректно обрабатывать различные проблемные ситуации, например, некорректные входные данные, нарушение соединения с объектами восстановления. Программа должна разрабатываться с использованием современных подходов тестирования и отладки программного обеспечения [60,61].

С учетом вышеперечисленных требований автором был разработан программный комплекс Virtual Self-Healing (VSHealing) с графическим интерфейсом пользователя, с разбиением исходного кода на модули и с использованием принципов объектно-ориентированного программирования. В качестве средства разработки был использован язык C++ [62] так как данный язык имеет все необходимые для разработки многопоточных и объектно-ориентированных программ, а также широко используется для разработки программ для ОС как семейства Linux, так и MS Windows.

Исходный код программного комплекса VSHealing расположен в 6 модулях:

- ***Главный модуль программы.*** В нем содержатся элементы графического интерфейса, обработчики событий, поступающих от них, основные компоненты, обеспечивающие взаимодействие графического интерфейса, других модулей программы и базы данных.
- ***Модуль сбора данных №1.*** Данный модель отвечает за сбор данных о работе объектов восстановления.
- ***Модуль сбора данных №2.*** Данный модуль отвечает за сбор данных о текущем состоянии ВС: в нем содержатся компоненты подключения к платформе виртуализации и их сохранение в базу данных.
- ***Модуль подготовки объектов восстановления.*** Данный модуль реализует компоненты методики, отвечающие за вычисление параметров методов определения времени начала восстановления, и обеспечивает выполнение файловых операций с исходными данными методов определения времени

начала восстановления.

- ***Модуль математической обработки.*** Данный модуль реализует компоненты методики, отвечающие за определение времени начала восстановления, планирования процессов восстановления и поддержания плана восстановления в актуальном состоянии.
- ***Модуль контроля процессов восстановления.*** Данный модуль отвечает за реализацию плана восстановления и мониторинг процессов восстановления.

Рассмотрим особенности реализации каждого модуля более подробно:

Главный модуль. Одной из важных характеристик программного комплекса является наличие удобного интерфейса взаимодействия с пользователем. Для того чтобы упростить его разработку и сделать исходный код более легким для понимания особенностей функционирования программного комплекса было принято решение выделить элементы взаимодействия с пользователем в отдельный модуль. Главный модуль содержит компоненты графического интерфейса и обработчики событий, поступающих от них, которые обеспечивают:

- графическое представление информации о доступных виртуальных машинах, платформах виртуализации и параметрах их работы;
- графическое представление плана восстановления и параметров восстановления отдельных объектов;
- графический интерфейс для управления процессами восстановления.

Кроме того данный модуль включает компоненты взаимодействия с базой данных, обеспечивающих их загрузку и сохранения. В качестве базы данных программного комплекса была выбрана база данных SQLite [63], которая представляет собой легковесную встраиваемую реляционную базу данных, а также имеет всеобъемлющую документации по работе с ней.

Модуль сбора данных №1. Модуль осуществляет сбор данных о работе объектов восстановления и их сохранение в базе данных. Для использования разработанных методов определения времени начала восстановления CBR и ROR, необходимо определить набор мониторируемых характеристик работы целевой

программы (и ОС) и способы их получения.

В комплексной методике метод CBR используется для определения времени начала восстановления платформы виртуализации. Сбор данных о работе данного объекта может быть выполнен двумя способами:

- С помощью стороннего программного обеспечения. Существует достаточно много средств мониторинга работы компьютерных систем и сетей, например, Nagios[83], Casti[84]. Данные средства обеспечивают возможность мониторинга состояния компьютеров через сеть, и включают достаточно широкий набор функций, таких как графическое представление данных, удаленный мониторинг через шифрованные туннели, отправка оповещений в случае возникновения проблем.
- С помощью средств мониторинга производительностью, встроенных непосредственно в ОС.

Основная сложность применения первого способа заключается в необходимости каким-то образом обеспечить интерфейс взаимодействия выбранного средства мониторинга с разрабатываемым программным комплексом. Кроме того, чтобы иметь возможность получать сведения обо всех необходимых показателях функционирования объекта восстановления может потребоваться не одна, а несколько дополнительных программ.

Поэтому было принято решение о выборе в качестве источника исходных данных средств мониторинга, встроенных в ОС привилегированной виртуальной машины. Для работы с источниками исходных данных на стороне объекта восстановления разработана специальная программа - Агент сбора данных №1. В задачи агента входят: сбор исходных данных от средства контроля, встроенных в ОС, и их передача модулю сбора данных №1.

Анализ научных работ, посвященных решению задачи определения времени начала восстановления на основе мониторинга, позволил сформировать набор широко используемых характеристик мониторинга процесса «старения»:

- свободной объем оперативной памяти (memFree, Кб),

- свободный объем файла подкачки (swpFree, Кб),
- количество файловых дескрипторов (fd),
- размер таблицы процессов (procTbl),
- средняя загрузка ОС за последние 5 минут (avg5).

Стоит отметить, что представленный набор характеристик не является окончательным и может быть изменен.

Применение стороннего программного обеспечения для мониторинга объекта восстановления представляется целесообразным в случае, если оно уже используются для этой задачи. Такой выбор также может обеспечить возможность снижения дополнительных вычислительной нагрузки, связанной с повторным сбором данных о работе объекта восстановления. Однако в таком случае возникает необходимость создании интерфейса взаимодействия выбранного средства мониторинга с разработанным программным комплексом.

Набор данных, используемый в процессе определения времени начала восстановления методом ROR, включает количество запросов, обработанных с момента запуска объекта восстановления. Источник информации о количестве запросов может быть расположен:

- на стороне объекта восстановления. Прежде всего, источниками такой информации являются непосредственно лог-файлы объекта восстановления.
- на промежуточном узле обработки запросов. В данном случае информация о количестве запросов поступает от узла сети, выполняющего промежуточную обработку запросов, например, планировщика нагрузки. Однако данный источник не всегда может быть в наличии.

Поэтому в качестве источника данных были выбраны лог-файлы объекта восстановления. Сбор информации о количестве запросов выполняет специальная программа - Агент сбора данных №2. В задачи агента входит: получение информации о количестве запросов, обработанных объектом восстановления с момента запуска, и её передача модулю сбора данных №1.

Модуль сбора данных №2. Чтобы начать сбор информации о базовом уровне

ресурсов хостов и требованиях виртуальных машин, необходимо сначала выбрать типы ресурсов, которые представляют интерес при размещении виртуальной машины. На сегодняшний день при управлении виртуальными машинами можно выделить 4 основных типа ресурсов: оперативная память, сетевая подсистема, процессор и дисковая подсистема [38,64]. Дисковая подсистема определяет возможности создания новой виртуальной машины и изменения объема её виртуального жесткого диска. Данный ресурс не оказывает влияния на возможность запуска и перемещение виртуальной машины между хостами и поэтому не рассматривается далее. Как ограничивающий фактор оперативная память являются более строгим фактором, нежели чем процессор или сетевая подсистема. Действительно, в силу жестких ограничений физически невозможно поместить количество виртуальных машин больше, чем позволяет емкость оперативной памяти компьютера. Следует отметить, что в большинстве случаев требование к оперативной памяти для виртуальной машины задается перед её запуском и остается неизменным на всём протяжении её функционирования. Однако некоторые платформы виртуализации поддерживают механизм динамического распределения памяти между виртуальными машинами [33,38], что обеспечивает изменение выделяемого виртуальной машины объема оперативной памяти в процессе её работы. Что же касается процессора и сетевой подсистемы, физически при достаточных ресурсах по памяти поместить можно сколь угодно много виртуальных машин, но при нехватке ресурсов по процессору и сетевой подсистеме могут наблюдаться задержки по передаче данных.

Таким образом, оперативная память является ключевым ресурсом, определяющим размещение виртуальных машин. Требования к таким ресурсам как процессор и сетевая подсистема большинством современных платформ виртуализации практически не учитываются при размещении виртуальной машины. Важно отметить, что набор типов ресурсов, которые необходимо учитывать при планировании процессов восстановления, должен соответствовать набору, который используется платформой виртуализации при управлении

виртуальными машинами.

На данный момент существует множество платформ виртуализации, как с открытым исходным кодом, например, KVM [65], так и коммерческих решений, например, Citrix XenServer [66], VMware ESX Server [67], Microsoft Hyper-V [68]. В рамках диссертационного исследования в разработанном программном комплексе реализована поддержка платформы виртуализации Citrix XenServer, которая является одной из широко востребованных. Доступ к платформе виртуализации был реализован на основе специального набора функций, предоставляемых компанией-разработчиком платформы, который обеспечивают возможность подключения к платформе виртуализации, мониторинг виртуальных машин и хостов, и управление ими. Для данной платформы виртуализации при управлении процессами восстановления учитывается тип ресурса - оперативная память [69].

В процессе мониторинга выполняется сбор следующих данных: базовые уровни ресурсов хостов (технические характеристики аппаратных компонент), требованиям виртуальных машин к ресурсам и распределение виртуальных машин по хостам. Под требованиями виртуальных машин к ресурсам понимается некоторая величина, приведенная к тем же единицам измерения, что и базовые уровни ресурсов.

Модуль подготовки объектов восстановления. Набор функциональности, реализованный в данном модуле, отвечает за выполнение задач, определенных для этапа подготовки каждого из методов определения времени начала восстановления.

Каждый из методов определения времени восстановления в модуле подготовки объектов восстановления имеет свои особенности реализации, учитывающие процесс подготовки метода. При подготовке метода определения времени начала восстановления CBR возможно возникновение двух ситуаций:

- исторические данные о работе объекта восстановления есть в наличии на момент подготовки метода. В таком случае определение параметров метода

CBR выполняется точно в соответствии с шагами, определенными для этапа его подготовки;

- исторические данные о работе объекта восстановления отсутствуют на момент подготовки метода. В таком случае процесс определения параметров метода CBR предполагает внесение некоторых уточнений.

При отсутствии исторических данных о работе объекта восстановления определение параметров метода CBR выполняется в процессе его работы на основе использования более одного индикатора «старения». Для этого в качестве индикаторов «старения» устанавливается весь набор характеристик, определенных для мониторинга объекта восстановления. Для каждой из характеристик выполняется определение нижней границы запуска процесса восстановления на основе второго подхода, определенного для метода CBR. При данном подходе допустимый уровень индикатора «старения» выбирается на основе экспертной оценки, а нижняя граница запуска процесса восстановления определяется выражением (2.4).

Величина изменения сглаженного индикатора «старения» для режимов №1 и №2 метода CBR рассчитывается в процессе мониторинга объекта восстановления. При использовании принципа №3 длительность процесса восстановления изначально задается на основе экспертной оценки и уточняется в процессе работы объекта восстановления. По мере работы объекта восстановления и реализации процессов восстановления выполняется расчёт времени запуска процесса определения времени начала восстановления. В дальнейшем по мере накопления информации о работе объекта восстановления может быть выполнена повторная процедура подготовки метода CBR с целью исключения из рассмотрения бесперспективных характеристик и уточнения параметров метода CBR.

Исходными данными для подготовки метода ROR являются количество запросов, обработанных объектом типа 1 с момента его запуска, и время отработки запросов. Возможные источники информации о количестве запросов

приведены при описании модуля сбора данных №2 и сделан выбор в пользу лог-файлов. Источником информации о времени обработки запросов также выступают лог-файлы. В случае отсутствия исходных данных у объекта восстановления по какой-либо причине, например, объект запущен впервые, возможны три варианта последующих действий:

- мониторинг объекта восстановления без его добавления в план восстановления с целью накопления исходных данных. На основе собранных данных затем осуществляется подготовка метода ROR;
- выбор другого метода определения времени восстановления, например, метода CBR;
- подготовка исходных данных на основе моделирования работы объекта типа 1. В данном случае требуется проведение набора тестовых испытаний объекта восстановления в близких к производственным условиях. Для реализации данного варианта может быть использован метод ускорения процесса «старения» [71]. Данный метод направлен не снижение длительности проведения тестовых испытаний при воздействии на программу эффекта «старения» ПО.

Применение первого варианта может оказаться наиболее простым, так как предполагает только отсрочку применения метода ROR. Трудоемкость применения второго варианта определяются особенностями выбранного на замену метода. Величина затрат при использовании третьего варианта связана, прежде всего, с подготовкой и проведением тестовых испытаний.

Кроме функциональности, обеспечивающей подготовку методов определения времени начала восстановления, данный модуль включает дополнительный функционал, отвечающий за файловые операции с исходными данными, сохранение результатов, а также компоненты графического интерфейса и обработчики событий, поступающих от них.

Модуль математической обработки. Данный модуль является ядром разработанного программного комплекса и отвечает за определение времени

начала восстановления, формирование плана восстановления и поддержание его в актуальном состоянии.

В данном модуле реализовано решение следующих задач:

- определение времени начала восстановления методом CBR. Подробное решение данной задачи приведено в разделе 2.2.1.
- определение времени начала восстановления методом ROR. Подробное решение данной задачи приведено в разделе 2.2.2.
- обнаружение потребности в обновлении плана восстановления. Решение данной задачи строится на основе выявления при управлении процессами восстановления условий запуска процесса обновления плана восстановления, которые приведены в разделе 3.3. Для решения данной задачи выполняется мониторинга изменений следующих параметров восстановления: базового уровня ресурсов, требований виртуальных машин к ресурсам, распределение виртуальных машин по хостам и времени начала восстановления. В результате возникновения одного из предопределенных условий выполняется обновление плана восстановления.
- формирование и обновление плана восстановления. Формирование и обновление плана восстановления являются схожими задачами и соответственно используют практически одни и те же компоненты. Основное отличие задачи обновления плана восстановления от задачи формирования заключается в наборе объектов восстановления, для которых выполняется пересчет параметров восстановления. Обновление плана восстановления выполняется только в отношении объектов, на которые потенциально может оказать влияние изменение параметров восстановления. В общем случае это объекты, время начала восстановления которых больше чем время начала восстановления того объекта, параметры восстановления которого изменились.

Кроме реализации решения перечисленных задач модуль осуществляет загрузку исходных данных и сохранение результатов в базе данных.

Модуль контроля процессов восстановления. В главе 3 был определен набор

методов восстановления рабочего состояния программы комплексной методики: метода VMMR, метод перезагрузки ОС и метод VMS. Реализация первых двух осуществляется на основе специального набора функций платформы виртуализации, предоставляемых компанией-разработчиком платформы. Набор функций обеспечивает возможность подключения к платформе виртуализации, осуществления запуска, остановки виртуальных машин и платформ виртуализации, а также перемещения виртуальных машин между хостами на основе технологии «горячей» миграции.

Метод подмены виртуальной машины VMS реализован в виде отдельной программы – Агента восстановления. Для апробации метода VMS разработка агента восстановления была выполнена для ОС семейства Linux, дистрибутив Debain [66]. Реализация метода включала решение трех основных задач: трассировка соединений с объектом восстановления, перераспределение трафика и непосредственно подмена виртуальной машины.

Для решения задач трассировки соединений и перераспределения трафика был использован встроенный в ядро ОС семейства Linux межсетевой экран Netfilter [67], который производит фильтрацию и модификацию пакетов по правилам. В качестве интерфейса управления им используется программа iptables, позволяющая манипулировать набором правил и таблиц Netfilter. Данное средство поддерживает работу с основными протоколами TCP/IP, а также предоставляет возможность создания пользовательских модулей для обработки трафика. Основными преимуществами межсетевого экрана являются его широкое распространение, развитый механизм трассировки соединений, гибкость и простота настройка процесса перераспределения трафика.

Принцип работы Netfilter заключается в том, что все пакеты пропускаются через определенные для них последовательности цепочек. При прохождении пакетом цепочки к нему последовательно применяются все правила этой цепочки в порядке их следования. Под применением правила понимается: во-первых, проверка пакета на соответствие критерию и, во-вторых, если пакет этому

критерию соответствует, применение к нему указанного действия. Под действием может подразумеваться как элементарная операция (встроенное действие, например, ACCEPT, MARK), так и переход в одну из пользовательских цепочек.

Важной особенностью Netfilter является механизм трассировки соединений — специальная подсистема, отслеживающая состояния соединений и позволяющая использовать эту информацию при принятии решений о судьбе отдельных пакетов. В пространстве ядра, в зависимости от типа протокола, пакеты могут иметь несколько различных состояний. Однако вне ядра допустимым является только одно из 4-х базовых состояний: NEW, ESTABLISHED, RELATED и INVALID.

Признак NEW сообщает о том, что пакет является первым для данного соединения. Состояние ESTABLISHED говорит о том, что это не первый пакет в соединении. Соединение получает статус RELATED, если оно связано с другим соединением, имеющим признак ESTABLISHED, т.е. оно инициировано из уже установленного соединения. Признак INVALID говорит о том, что пакет не может быть идентифицирован и поэтому не может иметь определенного статуса. Трассировка соединений производится в цепочке ***PREROUTING***, исключая случаи, когда пакеты создаются локальными процессами на межсетевом экране, в этом случае трассировка производится в цепочке ***OUTPUT***.

В целях апробации была выполнена реализация метода VMS в двух вариантах: с обработкой пакетов в пространстве ядра и с обработкой пакетов в пространстве пользователя. Первый вариант предполагает использование модуля ROUTE из набора расширений для Netfilter - patch-o-matic [74], который служит для перенаправления трафика на заданный интерфейс или по указанному MAC-адресу. Во втором варианте перенаправление трафика выполняется с использованием действия QUEUE [74], которое отвечает за передачу пакета на обработку пользовательскому процессу. Netfilter поддерживается расширение данного действия NFQUEUE, начиная с версии ядра Linux 2.6.14, которое обеспечивает возможность размещения пакетов в очереди, идентифицируемые по

16-разрядному номеру. Для решений нашей задачи достаточно одной очереди. Перераспределение трафика выполняется отдельным процессом агента восстановления, которые отвечает за получение пакета из очереди и его перенаправление на сервер, на котором данный пакет должен быть обработан. Процесс перенаправления пакета строится на основе назначения ему MAC-адреса целевого сервера. Данный вариант реализации не требуется установки дополнительного модуля межсетевого экрана Netfilter, однако может увеличивать время передачи пакетов по сети в процессе восстановления за счет их обработки в пространстве пользователя.

В обеих реализациях процесс перераспределения трафика строится на основе установки правил в цепочку ***PREROUTING*** межсетевого экрана, который реализует следующую логику:

если пакет имеет признак ESTABLISHED или RELATED, то он передается локальному процессу (серверу). В противном случае он перенаправляется на другую виртуальную машину.

Процесс перераспределения трафика завершается после того как все активные соединения с пользователями на основной виртуальной машине будут завершены, т.е. отсутствуют соединения со статусом ESTABLISHED.

Непосредственно сам процесс подмены виртуальной машины включает в себя сокрытие дублирующей виртуальной машины от пользователей в процессе восстановления и смену ролей виртуальной машины. Сокрытие виртуальной машины реализуется на основе изменения режима передачи откликов на запросы ARP интерфейса, через который осуществляется взаимодействие с пользователем. Для данной цели на основе параметра интерфейса arp_ignore [75], устанавливается режим, при котором отклики на запросы ARP не выдаются. После завершения всех активных соединений с основной виртуальной машины выполняется смена ролей виртуальных машин. С этой целью для дублирующей виртуальной машины режим передачи откликов на запросы ARP изменяется на режим, использовавшийся на момент начала процесса восстановления. В тоже

время для основной виртуальной машины устанавливается режим, при котором отклики на запросы ARP не выдаются. Затем выполняется информирование всех узлов сети, о новом соответствии между IP адресом сервера и MAC адресом дублирующей виртуальной машины. Информирование выполняется на основе рассылки пакета ARP оповещения [75], который сообщает узлам сети о необходимости обновления связи между IP и MAC адресами.

Таким образом, основной виртуальной машиной становится вторая виртуальная машина, которая выполняет дальнейшее обслуживание пользователей. Затем выполняется сброс правил для межсетевого экрана Netfilter и отключение виртуальной машины со «старым» сервером с целью освобождения ресурсов ВС.

На рисунке 4.1 представлена схема размещения и взаимодействия компонентов разработанного программного комплекса VSHealing.

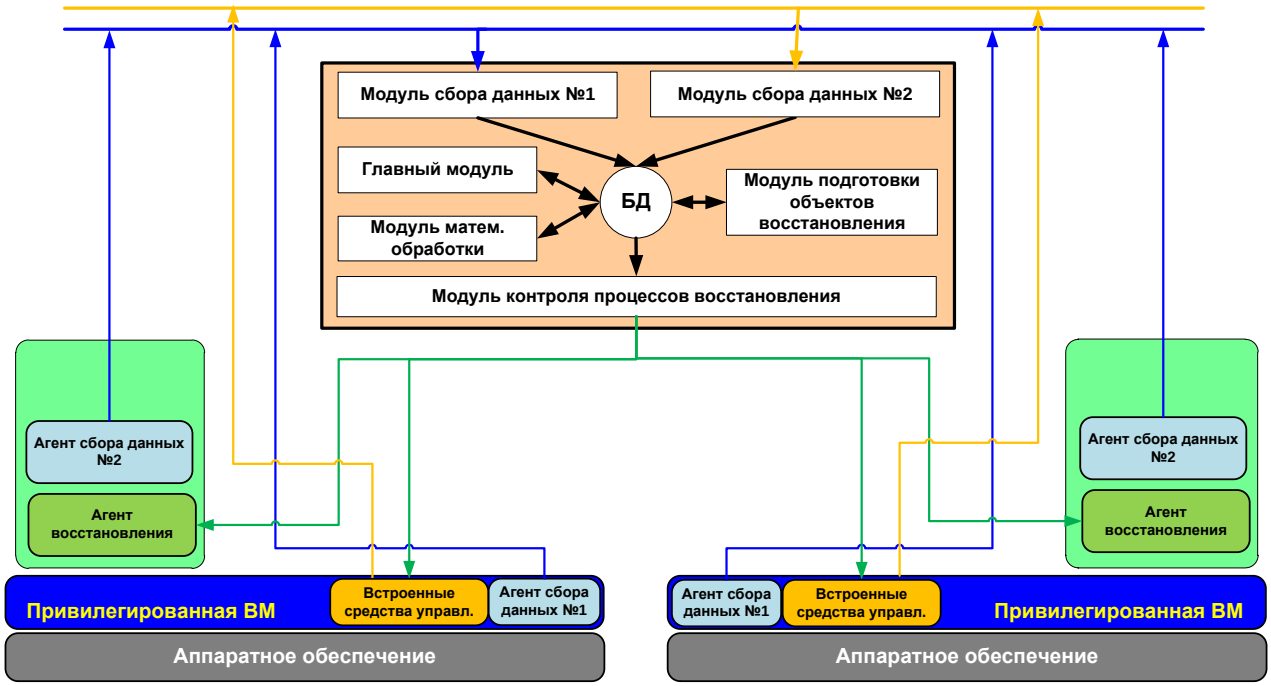

Рисунок 4.1. Схема размещения и взаимодействия компонентов разработанного программного комплекса VSHealing

Важным преимуществом разработанного программного комплекса является

простота добавления поддержки новой платформы виртуализации и ОС, достигаемая за счет внесения изменений только в его отдельные компоненты. Добавление поддержки новой платформы виртуализации затрагивает только модуль сбора данных №2, модуль управления процессами восстановления и агент сбора данных №1. Добавление поддержки новой ОС затрагивает только агент сбора данных №2 и агент восстановления.

4.2 Подготовка тестового стенда

4.2.2 Выбор программ с эффектом «старения» ПО и определение параметров их работы

Выбор с эффектом «старения» ПО

Сложность исследования эффекта «старения» ПО заключается в том, что по своей природе он требует длительного времени непрерывной работы программы. Время до выхода программы из строя по причине воздействия эффекта «старения» может занимать от нескольких часов до нескольких тысяч часов, что определяется множеством причин, например, доступным объёмом ресурсов, интенсивностью использования компонентов, ответственных за «старение», конфигурацией и режимом работы программы. Как результат, в исследованиях используются, либо хорошо известный и воспроизводимый эффект «старения» ПО [12,19,30], либо его эмуляция [31,76,77].

Распространенным подходом к эмуляции процесса «старения» является внесение в тестовую программу изменений, приводящих к возникновению эффекта «старения» ПО, или добавление в ОС программы, которая вызывает его появление. Широко распространенной проблемой, которая реализуется с использованием данного подхода, является утечка оперативной памяти. В работе [31] реализация данной проблемы включает детерминированную и недетерминированную составляющие. Реализация первой составляющей выполняется как невысвобождение памяти в размере 1024 байт на каждый запрос, поступающий к программе. Недетерминированная составляющая формируется

как невысвобождение памяти, объем которой представляет случайную величину и изменяется в интервале от 1 до 100 Кб на запрос.

Проведение исследований на программе с хорошо известным и воспроизводимым эффектом «старения» ПО осуществляется на основе ускорения данного процесса. Широко распространенным подходом к ускорению процесса «старения» ПО является повышение нагрузки на программу. В работе [12,23] для ускорения процесса «старения» при проведении экспериментов используется предельная нагрузка на программу.

В рамках диссертационной работы было проведено исследование существующих решений задачи ускорения тестовых испытаний [71,78,79]. В итоге была выбрана техника ускорения тестирования, предложенная в работе [71], направленная на ускорение процесса «старения». Авторами работы вводится понятие «фактор старения», который представляет собой характеристику работы программы, обеспечивающую управление процессом «старения» при проведении теста. Фактор «старения» выявляется на основе анализа результатов предварительных экспериментов. Примерами факторов «старения»: тип страницы, запрашиваемой на сервере, её размер, интенсивность запросов. Затем выявленные факторы «старения» используются при формировании модели нагрузки или настройки работы программы.

В диссертационной работе проведение экспериментов выполнялось с использованием известных эффектов «старения» ПО. С этой целью были выбраны две программы, «страдающих» эффектом «старения» ПО:

- Контейнер сервлетов Apache Tomcat [80]. Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish. Проведенные автором исследования показали, что для данной программы при определенных режимах работы, например, при нагрузке более 50 зап./с. наблюдается постепенная деградация её производительности с последующим отказом.

- Интерпретатор языка Perl [81]. Система сборки мусора, которой оснащён интерпретатор языка Perl, устроена таким образом, что не всегда обеспечивает автоматическое удаление данных, которые больше не используются [82]. Алгоритм работы системы сборки мусора строится на основе учета ссылок, сделанных на переменную. В результате этого при выполнении программы, написанной на языке Perl, возникают проблемы, связанные с не высвобождением памяти. Например, если переменная вышла из области видимости, но ссылки на неё осталась, то данные не удаляются, хотя имя переменной становится недоступным. Для проведения экспериментов была разработана программы на языке Perl, которая реализует описанный случай не высвобождения памяти интерпретатором.

Для контейнера сервлетов было установлено, что на скорость процесса «старения» наибольшее влияние оказывает интенсивность запросов (рост интенсивности запросов приводит к ускорению процесса «старения»). Данная характеристика была выбрана в качестве фактора «старения» для контейнера сервлетов. Для программы, написанной на языке Perl, в качестве фактора «старения» была выбрана частота вызова функции, которая ответственна за не высвобождение памяти в процессе выполнения программы.

Определение параметров генератора нагрузки

Генерируемый пользователями трафик является одним из параметров при проведении испытаний, поэтому требуется разработать генератор трафика для эмуляции работы пользователей с ВС. При этом разрабатываемый генератор трафика должен отражать поведение как отдельных пользователей, так характер изменения их поведения во времени в целом.

В простейшем случае формирование трафика может быть выполнено на основе записи запросов реальных пользователей. Основным недостатком такого подхода является большая длительность и трудоемкость подготовки необходимого объема данных о работе пользователей. В связи с чем описание процесса генерации трафика пользователем (то есть работа пользователей)

выполняется с использованием различных моделей. В рамках диссертационной работы было проведено исследование существующих моделей работы пользователей с отдельным сервером. В итоге была выбрана модель [85]. Автором данной работы был проведен анализ трафика (полученного от 6692 хостов), собранного из сети университета, по результатам которого была определены основные его характеристики и сформирована модель. Данная модель отражает такой ключевой аспект генерации трафика пользователями в сети как разделение процесса работы пользователей с сервером на соединения, запросы и связанные запросы. Запросы пользователей формируются в рамках соединения, в свою очередь запрос пользователем какого-либо ресурса сопровождается формированием множества новых запросов, связанных с запрашиваемых ресурсов. Схема запроса пользователя представлена на рисунке 4.2

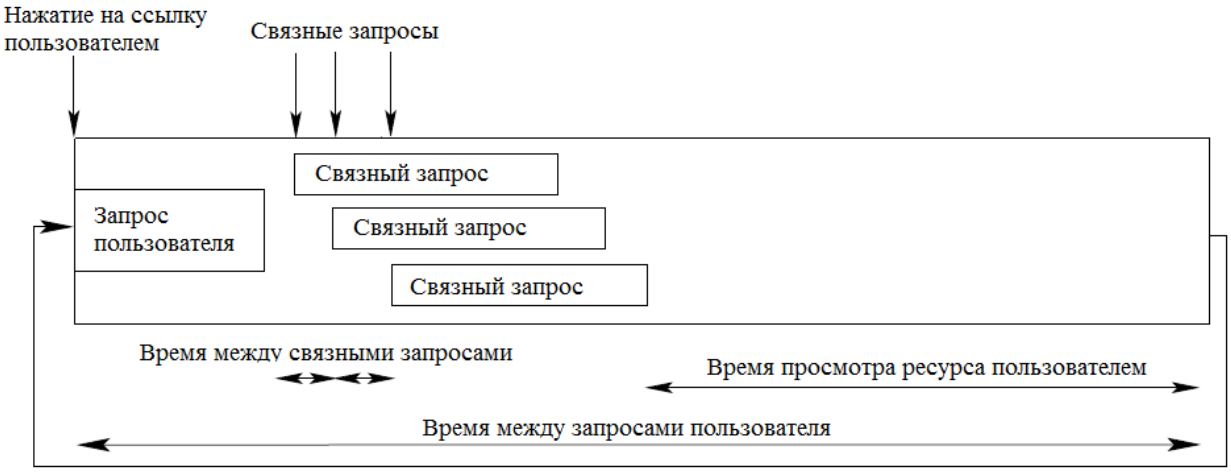

Рисунок 4.2. Схема запроса пользователя

Параметры, определенные в работе [85], отражают средние характеристики работы пользователей и не учитывают изменения характера их поведения во времени в целом. Как показывает практика, интенсивность работы пользователей в течение суток не является постоянной величиной. На рисунке 4.3 приведена диаграмма изменения количества соединений пользователей в течение суток для сайта Сбербанк России (www.sberbank.ru). Низкая активность пользователей,

обычная для ночного времени, сменяется высокой активностью в утренние и дневные часы.

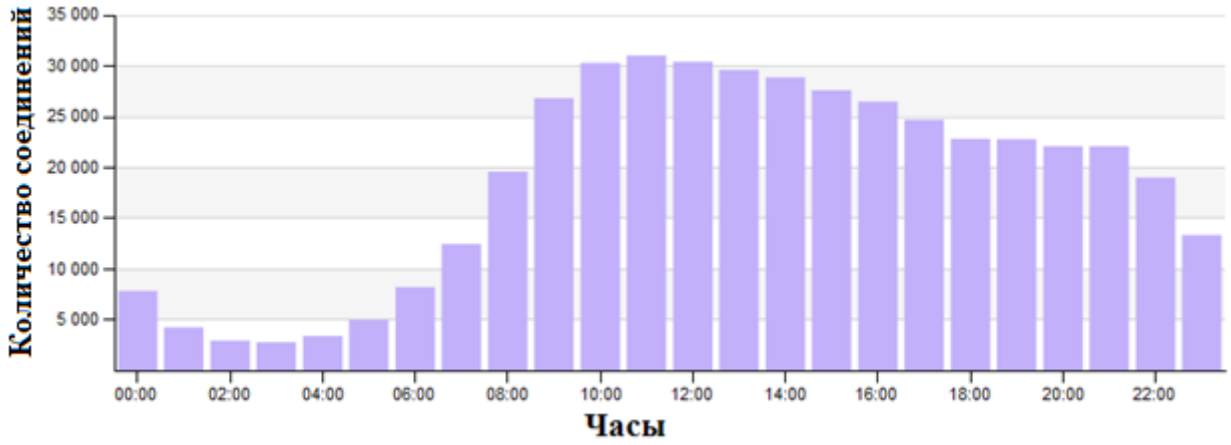

Рисунок 4.3. Количество соединений по часам

На основе данных, представленных на рисунке 4.3, было рассчитано среднее время между соединениями по часам и представлено в таблице 4.1 в виде относительной величины от среднего времени между соединениями за сутки.

Таблица 4.1

Доля времени между соединениями по часам

№	1	2	3	4	5	6	7	8
Доля времени, %	41,5	22,5	15,3	14,4	17,7	26	42,8	65,5
№	9	10	11	12	13	14	15	16
Доля времени, %	104	145	163	167,3	164	160	157,8	151
№	17	18	19	20	21	22	23	24
Доля времени, %	145,1	134,8	124,8	124,1	123,4	118,6	100,3	70,7

На основе рассмотренной выше модели поведения пользователей и приведенной в таблице 4.1 доли времени между соединениями, были определены значения параметров генератора трафика:

- *Время между соединениями.* Характер нагрузки представляет собой периодическую последовательность из 24 интервала продолжительностью 45 с. каждый. Время между соединениями на каждом интервале задается в соответствии со значениями, приведенными в таблице 1, и базовым значением времени между соединениями равным 2401 мс.
- *Количество запросов пользователей в соединении.* Значение данного параметра имеет Вейбулл распределение с параметрами масштаба 10,6819 и формы 0,9124. Минимальное количество запросов составляет 1.
- *Количество связных запросов в запросе пользователя.* Значение данного параметра задается на основе логнормального распределения с математическим ожиданием 2,155 и среднеквадратическим отклонением 1,377.
- *Время между запросами пользователя (мс).* Значение данного параметра задается на основе Вейбулл распределения с параметрами масштаба 53,2432 и формы 0,8569.
- *Время между связными запросами (мс).* Значение данного параметра задается на основе Вейбулл распределения с параметрами масштаба 315778,506 и формы 0,370912.

Значения приведенных параметров соответствуют значениям, указанным в работе [85], за исключением параметров, определяющих временные характеристики трафика. В предшествующем разделе было указано, что фактором «старения» контейнера сервлетов является интенсивность запросов. Для повышения интенсивности запросов значения параметров, определяющих временные характеристики трафика, были уменьшены в несколько тысяч раз по сравнению с исходными.

4.2.1 Организация тестового стенда

Для проведения экспериментов было выполнено построение тестовой виртуальной инфраструктуры, которая включала 5 компьютера, с установленной платформой виртуализации Citrix XenServer, и отдельный компьютер, который

использовался в качестве хранилища дисков виртуальных машин. Доступ к дискам виртуальных машин был организован на основе iSCSI протокола. Протокол iSCSI базируется на сетевом протоколе TCP/IP и разработан для установления взаимодействия и управления системами хранения данных. Данный протокол широко используется при построении виртуальной инфраструктуры [38]. Также при проведении экспериментов были задействованы 5 компьютеров для запуска программ, моделирующих работу пользователей. Кроме того, использовался отдельный компьютер (компьютер администратора), на котором была установлена программа управления виртуальной инфраструктурой Citrix XenCenter и разработанный программный комплекс VSHealing. Характеристики компьютеров, использованных в качестве платформы виртуализации (Компьютер типа А), для хранения дисков виртуальных машин и запуска программы, моделирующей работу пользователей, (Компьютер типа Б) приведены в таблице 4.2. Компьютер администратора имел характеристики, соответствующие компьютеру типа Б. Для проведения экспериментов было задействовано 10 виртуальных машин, характеристики которых приведены в таблице 4.2.

Таблица 4.2

Характеристики компьютеров и виртуальной машины

	Компьютер типа А	Компьютер типа Б	Виртуальная машина
Процессор	Intel Pentium 3 GHz	Intel Pentium 3 GHz	1 процессор
Оперативная память	3 Gb	1 Gb	512 Mb
Файл подкачки	нет	512 Mb	512 Mb
Сетевой интерфейс	Marvell Yukon 88E8001/8003/8010 PCI Gigabit Ethernet	Marvell Yukon 88E8001/8003/8010 PCI Gigabit Ethernet	Виртуальный сетевой интерфейс

Начальное распределение виртуальных машин по хостах варьировалось от 1 до 3 и выбиралось случайным образом в соответствии с равномерным распределением. Схема тестового стенда и начального распределения виртуальных машин по хостам приведены на рисунке 4.4.

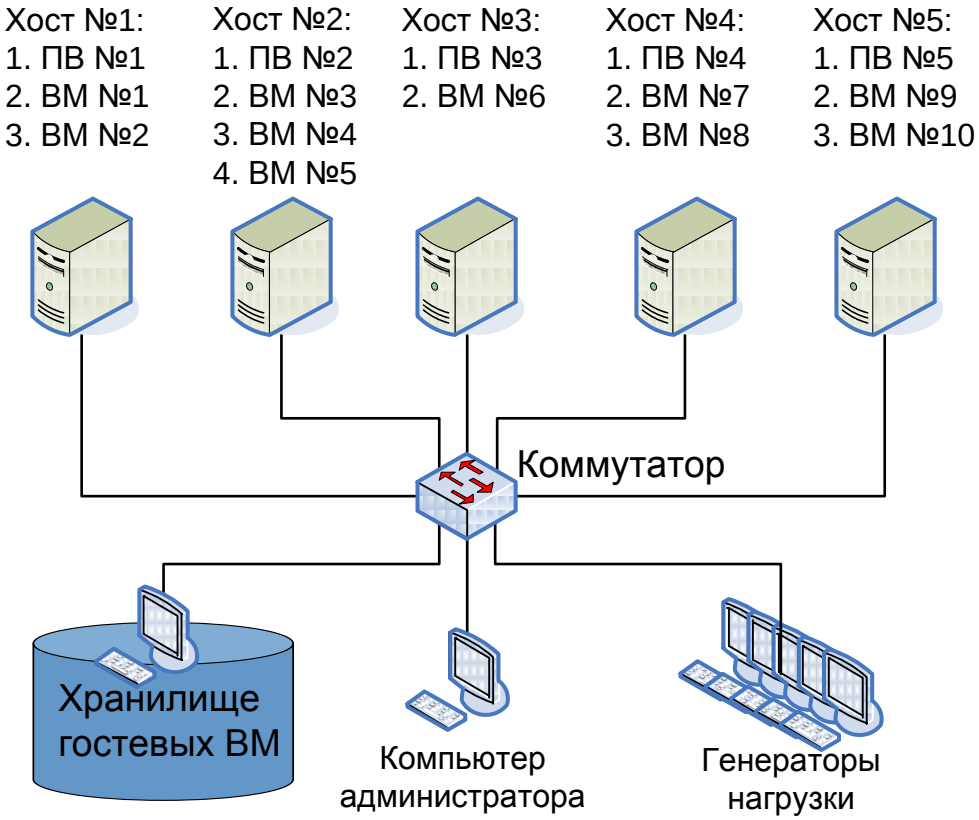

Рисунок 4.4. Схема тестового стенда

Эксперименты проводились для наиболее сложного случая, когда воздействию эффекта «старения» ПО подвержены непосредственно и платформа виртуализации, и сервер. Схема конфигурации хоста и размещение программ, подверженных эффекту «старения» ПО, приведена на примере хоста №5 на рисунке 4.5. Программа №1 представляет собой контейнер сервлетов Apache Tomcat и размещается внутри виртуальной машины, программа №2 представляет собой программу на языке Perl и размещается внутри привилегированной виртуальной машины.

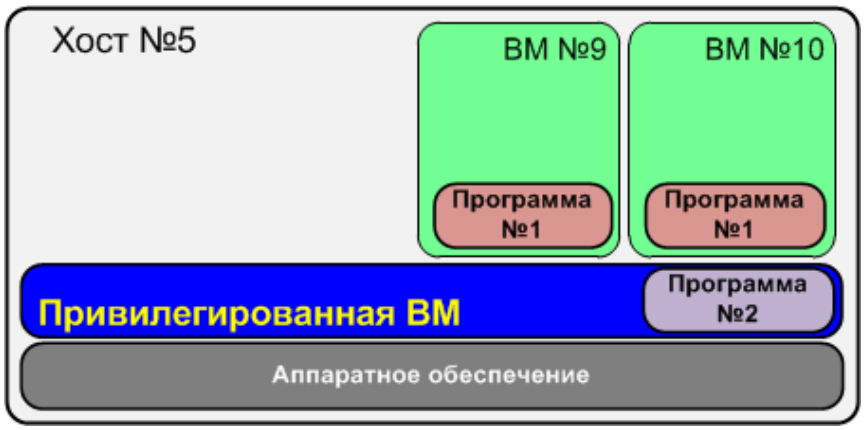

Рисунок 4.5. Пример схемы конфигурации хоста

4.3 Проведение экспериментов

После реализации разработанной комплексной методики была проведена серия экспериментов, целями которых были:

- Сравнить эффективность работы разработанной методики с существующими решениями, не требующими внесения изменений в исходный код восстанавливаемой программы, по двум критериям – среднее время отклика и доля потерянных запросов.
- Исследовать влияние разработанной методики на работу ВС и предложить рекомендации по её использованию. Для этого была проведена серия экспериментов с различными данными, определяющими интенсивность «старения» и объем ресурсов ВС.
- Подтвердить преимущества разработанного метода планирования процессов восстановления по сравнению с использованием для этой цели классического подхода к определению места размещения виртуальных машин. Была проведена серия модельных экспериментов с варьированием объема ресурсов на хостах ВС.

Сравнение проводилось с решениями, которые не требуют внесения изменений в исходный код целевой программ. Как было показано в разделе 1.2 решения по борьбе с эффектом «старения» ПО строятся на основе комбинирования методов определения времени начала восстановления и методов

восстановления рабочего состояния программы. Для проведения экспериментов была выполнена реализация наиболее востребованных методов определения времени начала восстановления и методов восстановления рабочего состояния программы в виде следующих решений:

1. Решение 1. Данное решение представляет собой комбинацию метода перезагрузки ОС для восстановления объектов типа 1 и 2, и метода определения начала времени начала восстановления на основе мониторинга их работы, представленного в работе [12].
2. Решение 2. Данное решение представляет собой комбинацию метода перезагрузки ОС для восстановления объектов типа 1 и 2, и метода определения времени начала восстановления без мониторинга их работы, представленного в работе [25].
3. Решение 3. Данное решение представляет собой комбинацию метода перезапуска программы для восстановления объекта типа 1 и метода перезагрузки ОС для объекта типа 2 с методом определения времени начала восстановления на основе мониторинга их работы [12].
4. Решение 4. Данное решение реализует реактивный подход. Решение основывается на быстром перезапуске виртуальной машины после обнаружения отказа [38].

В следующих разделах подробно описаны поставленные эксперименты и приведены их результаты.

4.3.1 Сравнительная оценка эффективности разработанной методики

Для сравнения эффективности работы подготовленных решений и разработанной методики были сгенерированы два сценария работы ВС при воздействии эффекта «старения». Сценарий здесь и далее в диссертационной работе определяется набором значений параметров работы объектов восстановления, определяющих эффект «старения» ПО:

- параметры генератора трафика;

- интервал времени вызова функции в целевой программе, написанной на языке Perl, ответственной за «старение» объекта типа 2.

Оба сценария отличались скоростью «старения» как объекта типа 1, которая регулировалась временем между соединениями, так и объекта типа 2, которая регулировалась интервалом вызова функции программы, ответственной за процесс «старение». В первом сценарии базовое время между запросами составляло 2401 мс и интервал вызова проблемной функции - 200 мс. Для второго сценария данные параметры были уменьшены на 10%. Остальные параметры генератора трафика для обоих сценариев соответствовали значениям, определенным в разделе 4.2.2.

Далее была проведена серия пробных запусков с целью сравниваемых решений и разработанной комплексной методики. По результатам были определены следующие параметры разработанной методики:

Метод определения времени CBR. В качестве индикатора «старения» был выбран свободный объем файла подкачки. Характер нагрузки обладал четко выраженными интервалами низкой активности, длительностью, сопоставимой с длительность процесса восстановления объекта типа 2, что позволило использовать метод CBR в режиме №1. Нижняя граница запуска процесса восстановления и время запуска процесса определения времени начала восстановления составили 121371 Кб и 3100 сек., соответственно.

Метод ROR. Требование к коэффициенту готовности было выбрано максимальным, а именно 100%. Требование к времени обработки запросов было задано на уровне 10% от значения, соответствующего среднему времени обработки при отсутствии воздействия эффекта «старения» ПО на работу объекта типа 1. Критическое время обработки запросов составляло 1000% от среднего времени обработки при отсутствии эффекта «старения» ПО.

Также по результатам экспериментов были определены параметры сравниваемых решений.

Метод на основе мониторинга объекта типа 1. В качестве характеристики,

используемой для мониторинга, принят свободный объем оперативной памяти. Значение свободного объема оперативной памяти, при котором выполнялся запуск процесса восстановления, составляло 10% от общего объема оперативной памяти.

Метод на основе мониторинга для объекта типа 2. В качестве характеристики, используемой для мониторинга, принят свободный объем файла подкачки. Значение свободного объема файла подкачки, при котором выполнялся запуск процесса восстановления, составляло 10% от общего объема файла подкачки.

Метод без мониторинга для объекта типа 1. Интервал запуска процесса восстановления составлял 2000 с.

Метод без мониторинга для объекта типа 2. Интервал запуска процесса восстановления составлял 6100 с.

Выбор интервала запуска процесса восстановления и значения характеристик, при которых выполнялся запуск процессов восстановления, определялся рекомендациями разработчиков используемых методов и характером воздействия эффекта «старения» ПО на целевую программу. Далее было выполнено тестирование разработанной методики и сравниваемых решений на каждом из сценариев. Длительность каждого эксперимента составляла 8 часов. После завершения каждого эксперимента выполнялась обработка результатов и возврат элементов тестового стенда в исходное состояние для проведения следующего эксперимента. Результаты проведения экспериментов приведены на рисунках 4.6 и 4.7 (разработанная комплексная методика обозначена «Методика»).

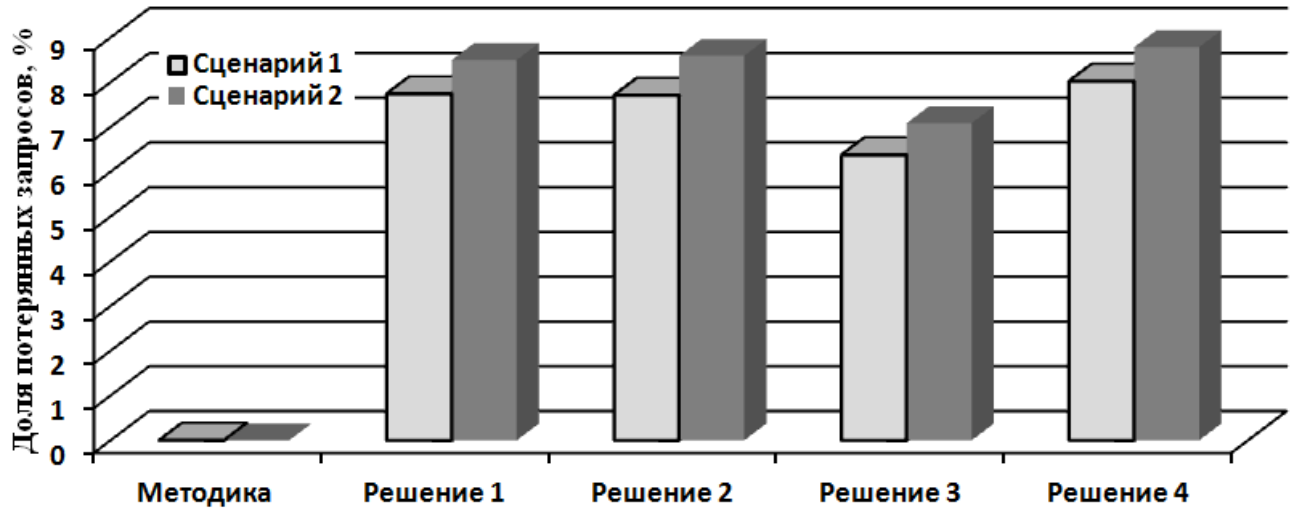

Рисунок 4.6. Доля потерянных запросов при различных сценариях для сравниваемых решений

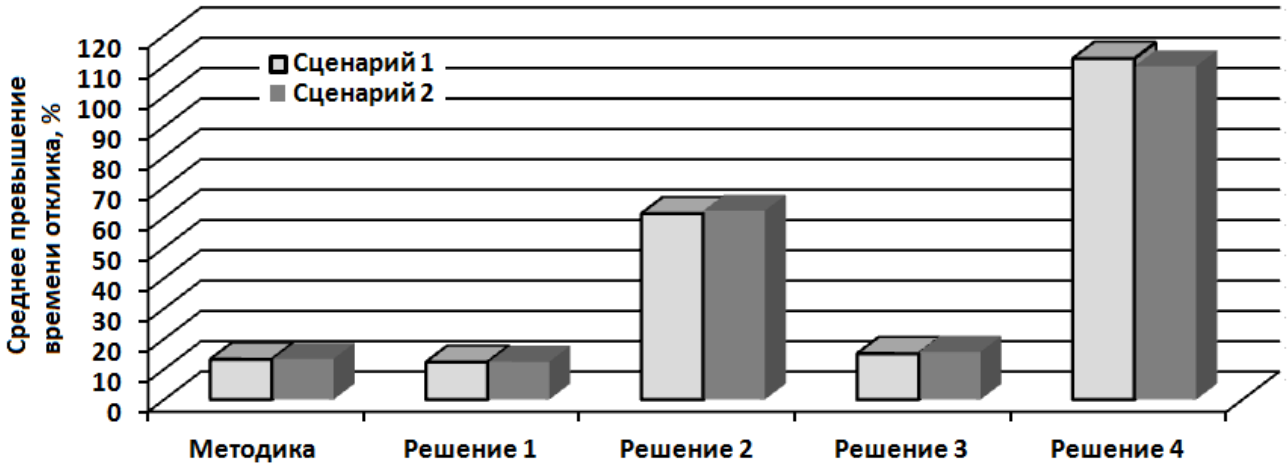

Рисунок 4.7. Среднее превышение времени отклика при различных сценариях для сравниваемых решений

Из приведенных результатов экспериментов можно сделать вывод, что разработанная комплексная методика лучше справляется с задачей борьбы с эффектом «старения» ПО и позволяет получить существенно лучше результаты, а именно полностью исключить потерю запросов и обеспечить среднее время отклика на уровне лучших результатов среди сравниваемых решений.

При разработке методики было отмечено, что используемые в ней методы

восстановления рабочего состояния программы предполагают наличие на время восстановления достаточного объема свободных ресурсов ВС.

Для оценки эффективности работы разработанной методики при ограниченности ресурсов была проведена серия экспериментов, в которой был рассмотрен крайний случай, когда реализация методов VMS и VMMR не возможна. Для этого базовый уровень оперативной памяти на хостах был ограничен, так чтобы исключить возможность перемещения виртуальных машин между хостами. Технология виртуальных машин обеспечивает уровень изоляции программ, размещенных внутри виртуальных машин, соизмеримый с физическими машинами. В связи с чем введение ограничения на оперативную память не оказывает влияния на работу программ, размещенных внутри виртуальных машин. При новых условиях было выполнено тестирование разработанной методики на каждом из сценариев, аналогично предшествующему эксперименту. Результаты эксперимента приведены на рисунках 4.8 и 4.9. Результаты работы методики при ограниченном объеме ресурсов ВС обозначены как «Методика (огр.)».

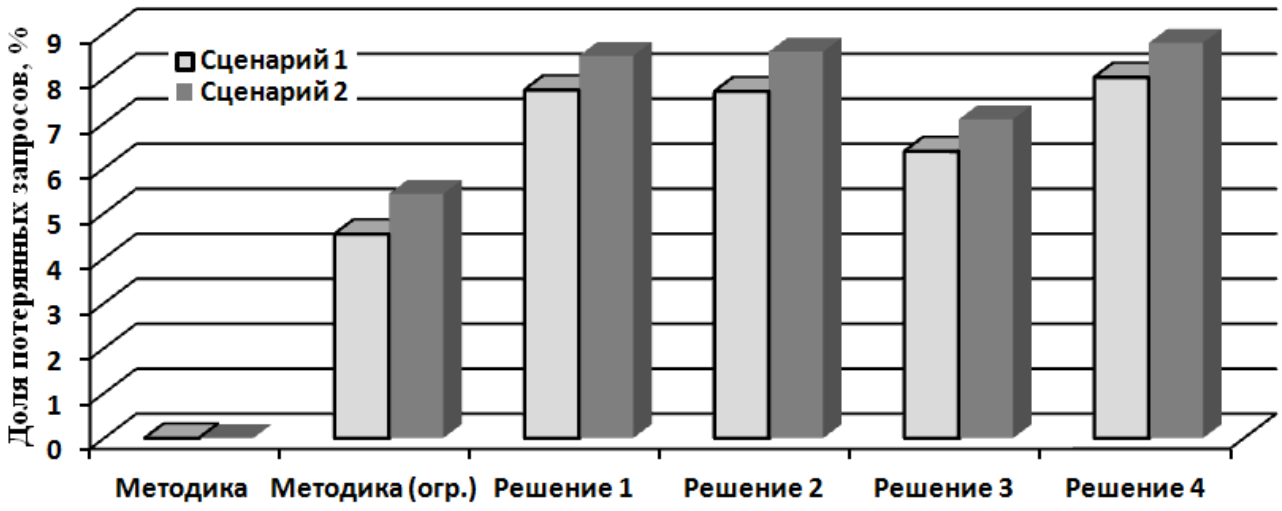

Рисунок 4.8. Доля потерянных запросов при различных сценариях для сравниваемых решений

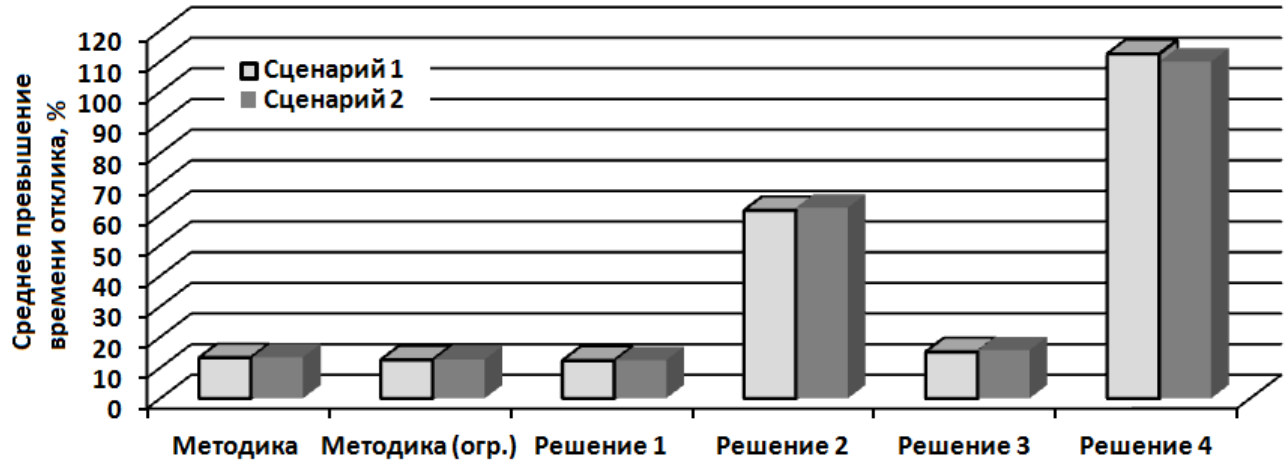

Рисунок 4.9. Среднее превышение времени отклика при различных сценариях для сравниваемых решений

Результаты показывают, что разработанная методика в случае отсутствия достаточного объема ресурсов ВС для реализации процессов восстановления предусмотренными в ней методами VMS и VMMR демонстрирует более слабые результаты в сравнении с её результатами в первой серии экспериментов. Однако обеспечивает снижение доли потерянных запросов более чем на 30% и среднее время отклика на уровне лучших результатов сравниваемых решений.

Как видно, результат работы разработанной методики зависит от наличия достаточного объема ресурсов для реализации процессов восстановления методами VMS и VMMR. В случае его отсутствия вместо метода VMS используется метод перезагрузки ОС, а перезапуск платформы виртуализации выполняется без перемещения виртуальных машин. Это обстоятельство, главным образом, и обуславливает снижение эффективности методики. Решение 1 и разработанная методика в случае ограниченного объема ресурсов (Методика (огр.)) показали близкие результаты, за счет использования схожих методов восстановления. Однако в отличие от решения 1 разработанная методика включает методы определения времени начала восстановления, которые учитывают издержки выбранного метода восстановления рабочего состояния

программы. В результате этого разработанная методика обеспечивает меньшую долю потерянных запросов в сравнении с решением 1.

Таким образом, проведенная серия экспериментов показала, что разработанная комплексная методика лучше справляется с задачей снижения влияния эффекта «старения» ПО на работу ВС и позволяется улучшить выбранные показатели её эффективности, среднее время отклика и доля потерянных запросов. Однако наличие достаточного объема свободных ресурсов для реализации процессов восстановления является той ценой, которую приходится платить за высокие результаты.

4.3.2 Оценка влияния состояния ресурсов ВС на работу разработанной методики

Серия экспериментов, проведенная в предшествующем разделе, показала, что на результаты работы разработанной комплексной методики оказывает влияние возможность реализации процессов восстановления на основе методов восстановления рабочего состояния VMS и VMMR. При разработке данных методов восстановления было отмечено, что для их работы требуется наличие достаточного объема свободных ресурсов. Как видно из предшествующей серии экспериментов отсутствие свободных ресурсов ведет к существенному снижению эффективности применения методики по показателю «доля потерянных запросов». Очевидно, что возможность реализации большего количества процессов восстановления методами VMS и VMMR повысит эффективность методики. Для эффективного использования ресурсов ВС с целью реализации процессов восстановления методами VMS и VMMR, в главе 2 был предложен метод планирования процессов восстановления RPRR.

Проведение экспериментов для оценки влияния состояния ресурсов ВС на эффективность применения методики на реальном оборудовании невозможна из-за того, что потребуется длительное время (более недели при условии использования метода ускорения эффекта «старения» ПО), большое количество

компьютеров (более десятка) и виртуальных машин (несколько десятков). Поэтому для оценки метода планирования процессов восстановления RPRR был поставлен модельный эксперимент.

Моделирование выполнялось для 15 хостов. Для усложнения задачи управления процессами восстановления при размещении виртуальных машин учитывались два типа ресурсов: оперативная память и процессор. Базовый уровень ресурсов на всех хостах был одинаковым. Требования виртуальных машин по каждому типу ресурсов задавались случайным образом в соответствии с равномерным распределением в интервале от 5% до 30% от базового уровня ресурсов хостов. Распределение виртуальных машин по хостам выполнялось в соответствии с законом равномерного распределения и с условием не превышения базовых уровней ресурсов хостов. Объем свободных ресурсов каждого хоста, оставшийся после распределения виртуальных машин, был равномерно распределен между размещенными на нем виртуальными машинами, таким образом, чтобы весь базовый уровень ресурсов хоста был занят. Количество виртуальных машин на каждом хосте составило от 5 до 8, а их общее количество - 80.

Далее была проведена серия запусков объектов восстановления на реальном оборудовании с параметрами работы, заданными в сценарии 1, с целью подготовки шаблонов их работы при воздействии эффекта «старения» ПО. Здесь и далее под шаблоном работы объекта восстановления подразумевается набор характеристик работы объекта восстановления, достаточный для определения времени начала его восстановления с помощью разработанной методики. Также была выполнена оценка длительности перемещения виртуальных машин между хостами, длительность восстановления методом VMS, длительность перезапуска виртуальной машины и платформы виртуализации. Исходя из практических соображений, полученные временные оценки были масштабированы с учетом соотношения длительности интервала времени, определенного при подготовке генератора трафика в разделе 4.2.2, и его реальной длительности. Использование

шаблонов в модельном эксперименте выполнялось следующим образом: для каждого объекта восстановления при его запуске и после очередного процесса восстановления случайным образом выбирался один из шаблонов, который и характеризовал дальнейшую работу данного объекта восстановления.

Модельный эксперимент был выполнен для двух вариантов комплексной методики, которые отличалась методами планирования процессов восстановления:

- Первый вариант представлял собой комплексную методику как есть, а именно планирование процессов восстановления выполнялось на основе разработанного метода RPRR (далее - «Методика 1»);
- Во втором варианте методики планирование процессов восстановления выполнялось на основе классического подхода к размещению виртуальных машин, без учета возможности перераспределения ресурсов. Определение места размещения виртуальной машины в данном варианте методики выполнялось на основе метода «первый подходящий» (First Fit) [86]. Данный метод обеспечивает выбор первого хоста, который удовлетворяет требованиям виртуальной машины к ресурсам (далее - «Методика 2»).

Затем были проведены несколько серий опытов. Все серии опытов отличались максимальным объемом каждого ресурса, добавляемого к базовому уровню ресурса хоста. Опыты в каждой серии отличались добавляемым объемом ресурса, который выбирался в пределах от 0 до заданного максимального добавляемого объема данной серии в соответствии с законом равномерного распределения. Для каждой серии опытов максимальный добавляемый объем ресурса повышался на 5% и рассчитывался по формуле:

$$dR_{i,j} = 5iR'_j / 100, \quad (4.1)$$

где $dR_{i,j}$ - максимальный добавляемый объем ресурса j-ого типа в i-ой серии;

R'_j - уровень j-ого типа ресурса.

Результаты моделирования приведены на рисунке 4.10.

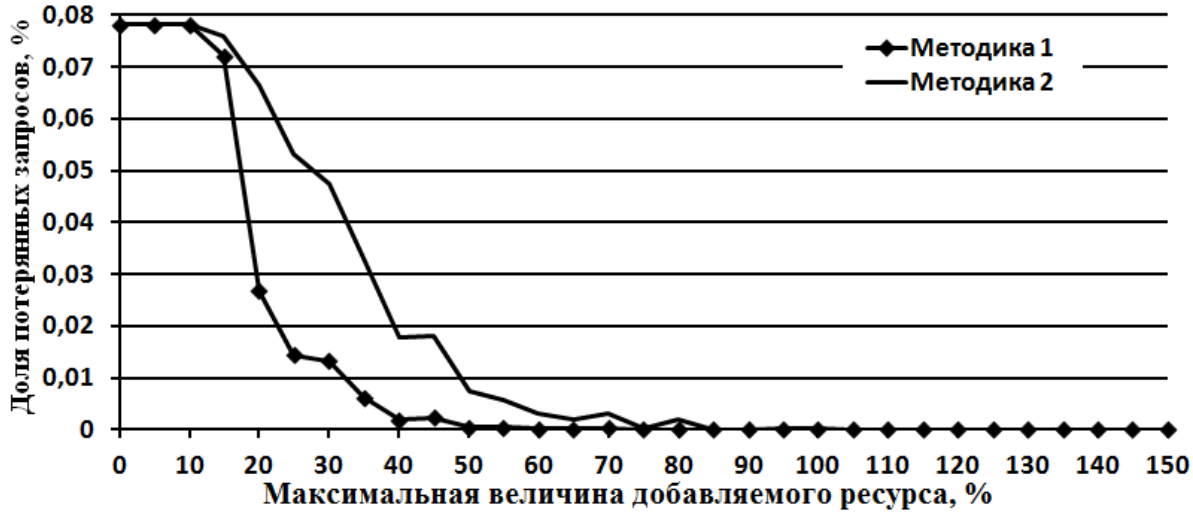

Рисунок 4.10. Доля потерянных запросов для различных вариантов методики

Из результатов моделирования можно сделать вывод, что комплексная методика с разработанным методом планирования процессов восстановления RPRR («Методика 1») лучше справляется с задачей управления процессами восстановления и обеспечивает снижение доли потерянных запросов в среднем более чем на 60% в условиях дефицита ресурсов ВС по сравнению с использованием классического подхода к размещению виртуальных машин («Методика 2»).

На рисунке 4.11 приведены значения минимального, среднего (среднего арифметического) и максимального количества вспомогательных перемещений виртуальных машин в каждой серии опытов при использовании в комплексной методике разработанного метода планирования процессов восстановления RPRR.

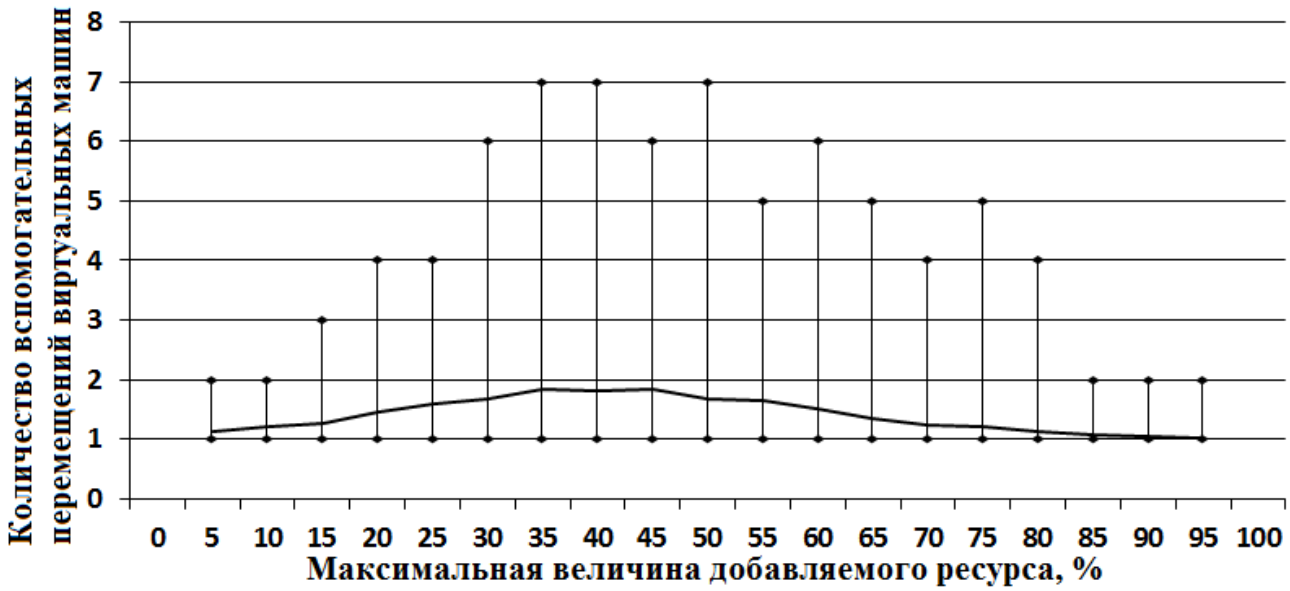

Рисунок 4.11. Количество вспомогательных перемещений виртуальных машин

Как видно из диаграммы на рисунке 4.11 среднее количество вспомогательных перемещений виртуальных машин в каждой серии опытов близко к единице, что позволяет судить о том, что в большинстве случаев достаточным является перемещения одной или двух виртуальных машин для размещения целевой виртуальной машины. Однако в некоторых случаях размещение может потребовать до 7 вспомогательных перемещений. Стоит отметить, что процесс перемещения виртуальной машины между хостами на основе технологии «горячей» миграции может оказывать воздействие на её производительность [36]. Величина данного воздействия определяются, прежде всего, длительностью процесса перемещения виртуальной машины, которая в свою очередь зависит, главным образом, от доступной пропускной способности канала, по которому выполняется перемещение, и интенсивности её работы с оперативной памятью во время перемещения. Длительность данного процесса обычно является несоизмеримо малой по сравнению с общим временем работы виртуальной машины и в большинстве случаев не оказывает существенного влияния на общую производительность ВС. Кроме того, данная технология является стандартным инструментом при балансировке нагрузки между хостами. Однако может оказаться целесообразным ограничить количество

вспомогательных перемещений, например, в условиях ограниченности свободных ресурсов, высоких требованиях к производительности ВС. Однако, окончательное решение о подобных ограничениях должно определяться исходя из условий работы конкретной ВС и требований к её эффективности.

4.4 Пример практической реализации

Комплексная методика снижения влияния эффекта «старения» ПО на эффективность функционирования вычислительной системы, разработанная Удовиченко А.О. в процессе выполнения диссертационной работы, была применена в проекте по модернизации вычислительной системы ЗАО «РНТ», обеспечивающей работу более 450 сотрудников компании. Применение разработанной комплексной методики позволило добиться более высоких показателей эффективности функционирования вычислительной системы по сравнению с ранее использовавшимся решением, а именно снизить долю потерянных запросов пользователей компании при обращении к корпоративным информационным ресурсам и среднее время отклика.

4.5 Выводы

Для оценки эффективности разработанной в диссертационной работе комплексной методики была выполнена её реализация и проведена серия экспериментов. Для реализации методики был сформирован набор компонентов, решающих отдельную задачу в рамках комплексной методике, и определена схема их взаимодействия. Эксперименты проведены с использованием платформы виртуализации Citrix XenServer, для которой была реализована разработанная методика и подготовлен тестовый стенд. Для оценки эффективности методики были проведены эксперименты с использованием реального оборудования и модельный эксперимент.

По результатам проведенных экспериментов сделаны следующие выводы:

- Разработанная методика при достаточном объеме свободных ресурсов ВС для реализации процессов восстановления предусмотренными в методике

методами VMS и VMMR обеспечивает существенно лучшие результаты, а именно полностью исключает потерю запросов и обеспечивает среднее время отклика на уровне лучших результатов среди сравниваемых решений.

- Высокая эффективность применения предложенной методики определяется наличием достаточного объема свободных ресурсов ВС для реализации разработанных методов восстановления VMS и VMMR.
- В случае отсутствия достаточного объема ресурсов ВС для реализации процессов восстановления предусмотренными в методике методами VMS и VMMR она обеспечивает снижение доли потерянных запросов более чем на 30% и среднее время отклика на уровне лучших результатов сравниваемых в диссертации решений.
- Входящий в состав комплексной методики разработанный метод планирования процессов восстановления RPRR учитывает возможность перераспределения ресурсов ВС. Благодаря этому свойству обеспечивается повышение количества реализуемых процессов восстановления предусмотренными в методике методами VMS и VMMR в условиях дефицита ресурсов ВС, что ведет к снижению доли потерянных запросов в среднем более чем на 60% по сравнению с использованием классического подхода к размещению виртуальных машин.

Таким образом, наличие достаточного объема свободных ресурсов ВС для реализации процессов восстановления предусмотренными в разработанной методике методами восстановления рабочего состояния программы VMS и VMMR является условием её высоких результатов.

ЗАКЛЮЧЕНИЕ

В процессе исследований, выполненных в диссертационной работе, получены следующие результаты:

1. Набор методов восстановления рабочего состояния программы для платформы виртуализации (метод VMMR) и сервера (метод VMS) , которые обладают следующими преимуществами перед существующими методами восстановления рабочего состояния программы:
   - обеспечивают восстановление вне зависимости от источника эффекта «старения» ПО;
   - не приводят к прерыванию обслуживания пользователей в процессе восстановления;
   - не требуют модификации исходного кода восстанавливаемой программы.
2. Набор методов определения времени начала восстановления:
   - метод CBR основан на мониторинге характеристик выполнения целевой программы и/или ОС и отличается от существующих методов тем, что ориентирован на определение времени начала восстановления платформы виртуализации и обеспечивает учет характера изменения условий работы платформы виртуализации;
   - метод ROR основан на мониторинге объема работы, выполненной целевой программы, и отличается от существующих методов тем, что ориентирован на определение времени начала восстановления сервера и обеспечивает учет требований к эффективности его работы по двум показателям – время обработки запросов и коэффициент готовности.
3. Разработан метод планирования процессов восстановления RPRR, который обладает следующими преимуществами перед существующими методами:
   - обеспечивает согласование процессов восстановления различных программ;
   - учитывает три показателя, характеризующих эффективность процессов восстановления: количество виртуальных машин, активность которых сохраняется в процессе восстановления, своевременность и длительность

процесса восстановления;

- учитывает возможность перераспределения ресурсов ВС при реализации процессов восстановления рабочего состояния.

4. Разработана общая схема взаимодействия компонентов методики и политика управления процессами восстановления, обеспечивающие формирование целостного решения, которое отличается от существующих тем, что учитывает особенности технологии виртуальных машин и позволяет улучшить работу ВС по двум показателям эффективности: среднее время отклика и доля потерянных запросов.
5. Предложена схема программного комплекса, реализующего разработанную методику, и выполнена её программная реализация.
6. Проведённая серия экспериментов показала, что разработанная комплексная методика превосходит по выбранным показателям (среднее время отклика и доля потерянных запросов) сравниваемые в диссертации решения. При наличии достаточного объема свободных ресурсов ВС для реализации процессов восстановления предусмотренными в разработанной методике методами VMS и VMMR она позволяет полностью исключить потерю запросов и обеспечить среднее время отклика на уровне лучших результатов среди сравниваемых решений. В случае отсутствия достаточного объема ресурсов ВС разработанная методика демонстрирует более чем на 30% лучший результат по показателю доля потерянных запросов, а среднее время отклика на уровне лучших результатов сравниваемых в диссертации решений.

СПИСОК ЛИТЕРАТУРЫ

1. Удовиченко, А.О. Проблема «старения» программного обеспечения и пути её решения / А.О. Удовиченко // Информатизация и связь. - 2012. - №1. - С.17–20.
2. Удовиченко, А.О. "Метод определения времени восстановления рабочего состояния приложения с учетом требований к его эффективности" / В.П. Соловьёв, А.О. Удовиченко // Информатизация и связь. - 2012. - №2. - С.61–66.
3. Удовиченко, А.О. Метод планирования размещения группы виртуальных машин с перераспределением ресурсов / В.П. Соловьёв, А.О. Удовиченко // Программные продукты и системы. - 2012. - №1. - С.134–138.
4. Удовиченко, А.О. Методы восстановления рабочего состояния приложения / А.О. Удовиченко // Программные продукты и системы. - 2012. - №2. - С.113–117.
5. Удовиченко, А.О. Комплексная методика борьбы с эффектом «старения» ПО / Н.Н. Пуцко, А.О. Удовиченко // Программная инженерия. - 2012. - №4. - С.13–18.
6. Удовиченко, А.О. Метод определения времени восстановления приложения, учитывающий условия его работы / А.О. Удовиченко // Информационные системы и технологии. - 2012. - №6. - С.5-15.
7. Avritzer, A. Monitoring smoothly degrading systems for increased dependability / A. Avritzer , E. Weyuker // Empirical Software Engineering - 1997. - V.2(1). - P.59–77.
8. Cisco security advisory: Cisco Catalyst memory leak vulnerability [Электронный ресурс]. - Cisco Systems, 2001. - Режим доступа: http://www.cisco.com/warp/public/707/cisco-sa-20001206-catalyst-memleak.pdf.
9. Huang, Y. Software rejuvenation: Analysis, module and applications / Y. Huang [и др.] // The Proceedings of Fault-Tolerant Computing Symposium. - 1995. - V.25. - P.381-390.
10. Cassidy, K., Gross K., Malekpour A. Advanced pattern recognition for detection of complex software aging in online transaction processing servers / K. Cassidy, K.,

K.Gross, A. Malekpour // International Conference on Dependable Systems and Networks. - 2002. - P.478–482.

11. Castelli, V. Proactive Management of Software Aging / V. Castelli [и др.] // IBM Journal of Research&Development. - 2001. - V.45(2). - P.311-332.
12. Li, L. An Approach for Estimation of Software Aging in a Web Server / L. Li, K. Vaidyanathan, K.Trivedi // International Symposium on Empirical Software Engineering. - 2002. - V.7. - P.91–100.
13. Marshall, E. Fatal Error: How Patriot Overlooked a Scud / E. Marshall // Science. - 1992. - V.255. - P. 1347.
14. Tai, A. On-board preventive maintenance: a design-oriented analytic study for long-life applications / A. Tai, L. Alkalaj, S. Chau // Computer Performance and Dependability Symposium. - 1999. - V.35. - P.215–232.
15. Gray, J. Why do computers stop and what can be done about it? / J. Gray // Symposium on Reliability in Distributed Software and Database Systems. - 1986. - P. 3-12.
16. Vaidyanathan, K. A comprehensive model for software rejuvenation / K. Vaidyanathan, K. Trivedi // IEEE Transactions on dependable and secure computing. - 2005. - V.2(2). - P.124-137.
17. Grottke, M. The Fundamentals of Software Aging / M. Grottke, Jr.Matias, K. Trivedi // 19th International Symposium on Software Reliability Engineering. - 2008. - V.19. - P.1-6.
18. Schmidt, K. High Availability and Disaster Recovery: Concepts, Design, Implementation / K. Schmidt //Springer. - 2006.
19. Matias, R. An Experimental Study on Software Aging and Rejuvenation in Web Servers / R. Matias, P. Filho // 30th Annual International Computer Software and Applications Conference. - 2006. - V.1. - P.189-196.
20. Candea, G. JAGR An Autonomous Self-Recovering Application Server / G. Candea [и др.] // 5th International Workshop on Active Middleware Services. - 2003. - P.168-178.

21. IBM Director Software Rejuvenation: white paper. - IBM Corporation, 2001. – 20p.
22. Candea, G. Microreboot - A Technique for Cheap Recovery / G. Candea [и др.] // 6th Symp. on Operating Systems Design and Implementation. - 2004. - V.6. - P.31-34.
23. Vaidyanathan, K. Analysis and Implementation of Software Rejuvenation in Cluster System / K. Vaidyanathan, R. Harper, S. Hunter, K. Trivedi // ACM SIGMETRICS. - 2001. - V.29(1). - P.62-71.
24. Silva, L. Using Virtualization to Improve Software Rejuvenation / L. Silva, J. Alonso, J. Torres // IEEE Transactions on Computers. - 2009. - V.58(11). - P.1525-1538.
25. Dohi, T. Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule / T. Dohi, K. Goseva-Popstojanova, K. Trivedi // 2000 Pacific Rim International Symposium on Dependable Computing. - 2000. - P.77-84.
26. Garg, S. Analysis of Software Rejuvenation using Markov Regenerative Stochastic Petri Net / S. Garg [и др.] // 60th Int. Symp. on Software Reliability Engineering. - 1995. - P. 24-27.
27. Bao, Y. A Workload-based Analysis of Software Aging and Rejuvenation / Y. Bao, X.Sun, K. Trivedi // IEEE Transactions on Reliability. - 2005. - V.54. - P.541-548.
28. Garg, S. A methodology for detection and estimation of software aging / S. Garg [и др.] // The Ninth International Symposium on Software Reliability Engineering. - 1998. - P.283-292.
29. Hoffman, G. A Best Practice Guide to Resource Forecasting for the Apache Webserver / G. Hoffman, K. Trivedi, M. Malek // IEEE Transactions on Reliability. - 2007. - V.56(4). - P.615-628.
30. Andrzejak, A. Deterministic Models of Software Aging and Optimal Rejuvenation Schedules / A. Andrzejak, L. Silva // 10th IFIP/IEEE Symposium on Integrated Management. - 2007. - V.10. - P.159-168.
31. Andrzejak, A. Using Machine Learning for Non-Intrusive Modeling and Prediction of Software Aging / A. Andrzejak, L. Silva // Network Operations and Management

Symposium. - 2008. - P.25-32.

32. Bittman, T.J. Magic Quadrant for x86 Server Virtualization Infrastructure : Gartner RAS Core Research Note [Электронный ресурс] / T.J. Bittman [etc.]. - Gartner, 2010. - Режим доступа: http://www.gartner.com/technology/media-products/reprints/vmware/article4/article4.html.
33. Marshall, D. Advanced Server Virtualization: VMware and Microsoft Platforms in the Virtual Data Center / D. Marshall, W. Reynolds, D. McCrory. - Auerbach Publications, 2005. - 742 p.
34. Hagen, W. Professional Xen Virtualization / W. Hagen. - Indianapolis: Wrox, 2008. - 405p.
35. Goldberg, R. Architectural Principles for Virtual Computer Systems. Harvard University / R. Goldberg. - Harvard University, 1973. - 229 p.
36. Clark, C. Live Migration of Virtual Machines / C. Clark [и др.] // 2nd Symposium on Networked Systems Design and Implementation. - 2005. - V.2. - P.273-286.
37. vSphere Basic System Administration: User’s Manual EN-000105-08. - VMware, 2011. - 364 p.
38. Citrix XenServer 6.0 Administrator's Guide: User’s Manual 1.1 Edition. - Citrix Systems, 2012. - 189p.
39. vSphere Availability Guide: User’s Manual EN-000316-01. - VMware, 2011. - 60 p.
40. Kourai, K. A fast rejuvenation technique for server consolidation with virtual machines / K. Kourai, S. Chiba // Proceeding of International Conference on Dependable Systems and Networks. - 2007. - V.37. - P.245-255.
41. Martin, F. Patterns of Enterprise Application Architecture / F. Martin. - Boston: Addison-Wesley Professional, 2003. - 560p.
42. Lavenberg, S.S. A Performance Evaluation: Origins and Directions : Lecture Notes in Computer Science / S.S. Lavenberg [и др.]; под ред. G. Haring, C. Lindemann, M. Reiser. - Berlin: Springer Verlag, 2000. - 506 p.
43. Менаске, Д. Производительность web-служб. Анализ, оценка и планирование : Пер. с англ. / Д. Менаске, В. Алмейда. - СПб.: ДиаСофтЮП, 2003. - 480 с.

44. Savoia, A. Web Page Response Time 101 / A. Savoia // STQE. - 2001. - V.3(4). - P.48-53.
45. Киллелиа, П. Тюнинг Web-сервера для профессионалов. 2-е изд. / П. Киллелиа. - СПб.: Питер, 2003. - 528 с.
46. Андреев, А.Л. Автоматизированные видеоинформационные системы : Учебное пособие / А.Л. Андреев. - СПб.: НИУ ИТМО, 2011. - 120 с.
47. Вержбицкий, В.М. Численные методы (математический анализ и обыкновенные дифференциальные уравнения**)** : Учеб. пособие для вузов / В.М. Вержбицкий. – М.: Высш. шк., 2001. - 382 с.
48. Литвак, Б.Г. Экспертная информация. Методы получения и анализа / Б.Г. Литвак. - М.: Радио и связь, 1982. - 184 с.
49. Тихонов, Э.Е. Методы прогнозирования в условиях рынка: Учеб. пособие / Э.Е. Тихонов. - Невинномысск: Изд-во СевКавГТУ, 2006. - 221 с.
50. Eubank, R. Nonparametric regression and spline smoothing : 2nd edition / R. Eubank. - New-York: CRC Press, 1999.
51. Shumway, R. Time series analysis and its applications: With R examples / R. Shumway, D. Stoffer. - New-York: Springer, 2006.
52. Коган, Д.И. Задачи и методы конечномерной оптимизации. Часть 3. Динамическое программирование и дискретная многокритериальная оптимизация : Учебное пособие / Д.И. Коган. - Нижний Новгород: НГУ, 2004. - 157 с.
53. Spall, J. Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control / J. Spall. - San-Francisco: WileyBlackwell, 2003. - 618 p.
54. Booch, G. Object-Oriented Analysis and Design with Applications : Second edition / G. Booch. - Santa Clara: Addison-Wesley, 1997. – 534 p.
55. Parnas, D. On the Criteria To Be Used in Decomposing Systems into Modules / D. Parnas // Communications of the ACM. - 1972. - V.15(№12). - P.1053-1058.
56. Weisfeld, M. The Object-Oriented Thought Process : Third Edition / M. Weisfeld. - Addison-Wesley, 2009.

57. Гордеев, А.В. Операционные системы: Учебник для вузов: 2-е изд. / А.В. Гордеев. - СПб.: Питер, 2007. - 416 с.
58. Таненбаум, Э.С. Современные операционные системы: 2-е изд. / Э.С. Таненбаум. - СПб.: Питер, 2005. - 1038 с.
59. Stallings, W. Operating Systems: Internals and Design : 7th edition / W. Stallings. - NewJersey: Principle Hall, 2012. - 767 p.
60. Синицын, С. В. Верификация программного обеспечения / С. В. Синицын, Н. Ю. Налютин. - М.: БИНОМ, 2008. - 368с.
61. Кем, К. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / К. Кем, Ф. Фолк, Н. Кек. - Киев: ДиаСофт, 2001. - 544 с.
62. Страуструп, Б. Программирование: принципы и практика использования C++, исправленное издание / Б. Страуструп. - М.: Вильямс, 2011. - 1248 с.
63. SQLite [Электронный ресурс]. – Режим доступа: http://www.sqlite.org.
64. vSphere Resource Management Guide : User's Manual EN-000317-02. – VMware, 2011. - 120 p.
65. Kernel Based Virtual Machine (KVM) [Электронный ресурс]. – Режим доступа: http://www.linux-kvm.org.
66. Citrix XenServer [Электронный ресурс]. – Режим доступа: http://www.citrix.com.
67. VMware ESX Server [Электронный ресурс]. – Режим доступа: http://www.vmware.com.
68. Microsoft Hyper-V [Электронный ресурс]. – Режим доступа: http://www.microsoft.com
69. Citrix XenServer 6.0 Virtual Machine Installation Guide : User's Manual 1.0 Edition. - Citrix Systems, 2011. - 67p.
70. Citrix XenServer 6.0 Configuration Limits : v1.18. - Citrix Systems, 2011. - 3p.
71. Matias, R. Accelerated Degradation Tests Applied to Software Aging Experiments / R. Matias, P. Barbetta, K. Trivedi, P. Filho // IEEE Transactions on Reliability. - 2010. - V.59(1). - P.102-114.

72. Debian [Электронный ресурс]. – Режим доступа: http://www.debian.org.
73. NetFilter [Электронный ресурс]. – Режим доступа: http://www.netfilter.org
74. Purdy, G. Linux iptables Pocket Reference / G. Purdy. - Sebastopol: O'Reilly Media, 2004. - 96 p.
75. Benvenuti, C. Understanding Linux Network Internals / C. Benvenuti. – Sebastopol: O'Reilly Media, 2005. - 1035p.
76. Andrzejak, A. Managing Performance of Aging Applications via Synchronized Replica Rejuvenation / A. Andrzejak, M. Moser, L. Silva // 18th IFIP/IEEE DSOM. - 2007. - P.98-109.
77. Gross, K. Proactive Detection of Software Aging Mechanisms in Performance Critical Computers / K. Gross, V. Bhardwaj, R. Bickford // Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop. - 2002. - P.17-23.
78. Mettas, A. Understanding accelerated life testing analysis / A. Mettas // International Reliability Symposium. - 2003. - P.1-16.
79. Ehrlich, W. Software reliability assessment using accelerated testing methods / W. Ehrlich [etc.] // Journal of the Royal Statistical Society. - 1998. - V.47(1). - P.15-30.
80. Apache Tomcat. http://tomcat.apache.org (дата обращения: 20.12.2014).
81. Perl Programming Language [Электронный ресурс]. – Режим доступа: http://www.perl.org.
82. Мичурин, А. Утечки памяти в программах на Perl / А. Мичурин // Системный администратор. - 2004. - №5(18).
83. Nagios [Электронный ресурс]. – Режим доступа: http://www.nagios.org.
84. Casti [Электронный ресурс]. – Режим доступа: http://www.cacti.net.
85. Walters, L.O. A web browsing workload modeling for simulation: master of science thesis / L.O. Walters. - Cape Town: UCT, 2004. - 177 p.
86. Гэри. М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон. - М.:Мир, 1982. - 439 с.