Разработка и оценка реальных отказоустойчивых систем виртуальных машин

Daniel J. Scales (VMware) scales@vmware.com

Mike Nelson (VMware) mnelson@vmware.com

Ganesh Venkitachalam (VMware) ganesh@vmware.com

Перевод: А.Ю. Катков (Самара)

Technical Report VMware-TR-2010-001

May 11, 2010

Аннотация

Мы внедрили коммерческую систему корпоративного уровня для обеспечения отказоустойчивости виртуальной машины, основанную на репликации работы основной виртуальный машина (VM) на резервную (вторичную) виртуальную машину на другом сервере. Мы разработали всю систему на базе VMware VSPHERE 4.0, простой в использовании, работающей на стандартных серверах, и, как правило, снижающей производительность реальных приложений менее чем на 10%. Наш метод репликации работы VM аналогичен описанному в Bressoud [3], но мы сделали ряд существенных конструктивных изменений, которые значительно повысили производительность. Кроме того, простая в использовании, коммерческая система, автоматически восстанавливающая избыточность после отказа, требует много дополнительных компонентов, кроме репликации выполняемых VM. Мы разработали и внедрили эти компоненты, а так же решили много практических проблем поддержки приложений, запущенных на виртуальных машинах. В этом документе мы описываем базовую конфигурацию, обсуждаем выбор дизайна и многие другие детали внедрения, и оценку работы для микротестов и реальных приложений.

Об особенностях перевода:

При переводе термины trap и interrupt переводил как прерывание, т.к. исходя из контекста и своего представления о задействованных механизмах лучшей формулировки не нашел (во встреченных случаях, механизмы trap срабатывают через вызов прерывания тем или иным способом).

В общем и целом я плохой переводчик с отрицательными способностями технического писателя. Буду признателен за предложения по улучшению получившегося документа.

Ключевые слова и фразы: virtual machines, fault tolerance, deterministic replay

Аннотация переводчика

Документ полезен тем, кто начинает интересоваться программной организацией систем FT на уровне гипервизора, но одной странички описания и утверждения, что все это работает уже мало. Наиболее важными для меня лично были два момента (это мои выводы, возможно ошибочные):

1. Ссылка на Cully [5] и краткое изложение результатов оценки производительности Remus, приводящие с мой точки зрения к выводу, что при текущем положении дел многопроцессорные ВМ под FT имеют смысл начиная с 3-4 процессоров и, скорее всего, связи между серверами по 10Гбит/с (в остальных случаях они не превзойдут по производительности текущие однопроцессорные реализации VMware).

2. В однопроцессорных ВМ полноценное отслеживание и запись всех недетерминированных событий (а если есть запись со всеми необходимыми параметрами, то реализуемо и воспроизведение) возможна за счет аппаратных средств современных процессоров (для VMware это AMD и Intel. Детали надо смотреть в техническом описании процессоров). При наличии двух и более процессоров любое обращение к памяти может стать недетерминированным событием (для однопроцессорных передаются только недетерминированные события, а в многопроцессорных при текущем положении дел необходима репликация всех изменений памяти). Реализация двухпроцессорных ВМ под FT с накладны расходами, сравнимыми с текущий однопроцессорной реализации VMware, требует соответствующей аппаратной поддержки. Я лично затрудняюсь представить себе приемлемые механизмы, позволяющие отделить обращения к памяти приводящие к недетерминизму, от не имеющих подобного влияния, не говоря о количестве событий, который будет порождаться при обработке различных блокировок и критических секций (аппаратные средства репликации памяти между серверами на мой взгляд не вписываются в текущую концепцию VMware), в результате чего не рассчитываю столкнуться с подобным в реальной жизни ранее 2020 года.

Copyright c 2010
VMware, Inc.

ТОРГОВЫЕ МАРКИ

VMware, эмблема VMware и дизайн, VMware vSphere и Motion - зарегистрированные торговые марки или торговые марки VMware, Inc. в Соединенных Штатах и/или другой стране. Все другие отметки и имена, упомянутые здесь, могут быть торговыми марками других компаний.

1. Введение

Общим методом реализации отказоустойчивых серверов является использование основного и резервного [1] с выполнением актуального копирования основного сервера на резервный. Т.к. обеспечивается идентичность основного и резервного серверов, при выходе из строя основного сервера, резервный может взять на себя обслуживание запросов клиентов без перерыва или потери статуса сервиса. Один из способов репликации серверов иногда называют подходом состояния машины [13]. Идея состоит в моделировании серверов как детерминированных автоматов, являющихся синхронными, начиная от одинакового начального состояния и обеспечения получения входящих запросов в том же порядке. Поскольку большинство серверов или сервисов содержат не детерминированные операции, должна быть дополнительная координация для синхронизации основной и резервной систем.

Координация, обеспечивающая гарантированную детерминированность выполнения физических серверов [14] сложна. Так при увеличении частоты процессора нарастают проблемы синхронизации часов. Напротив, виртуальная машина (ВМ), выполняющаяся под гипервизором, превосходная платформа для осуществления подхода основного/резервного серверов. ВМ это машиной, все операции которой - виртуализированы (включая все ее устройства). Как и физические серверы, ВМ выполняют некоторые недетерминированные операции (например, получение времени или обработка прерывания), в результате чего, появляется необходимость для обеспечения синхронности передать в резервную копию дополнительную информацию. Так как гипервизор полностью контролирует выполнение ВМ, включая получение всей входящей информации, гипервизор в состоянии захватить всю необходимую информацию о недетерминированных операциях на основном ВМ и корректно передать их на резервную ВМ.

Система репликации, основанная на виртуальных машинах, может синхронизировать только часть ВМ, обеспечивая им отказоустойчивость, оставляя другие ВМ в обычном режиме. Кроме того, технология, основанная на ВМ, не требует модификаций аппаратных средств, позволяя системе работать одновременно на старых и новых микропроцессорах, используя повышенную производительность последних. Система, основанная на репликации работы физических серверов, требует модификаций аппаратных средств, за счет чего часто отстает в производительности. Еще одно преимущество виртуальных машин для решения этой задачи - возможность физического разделения основной и резервной копий: например, копируемыми виртуальными машинами можно управлять на физических машинах, расположенных в разных зданиях, что обеспечивает большую надежность, чем расположение в том же самом здании. (Решение с аналогичными характеристиками, а то и существенно превосходящие по производительности, что правда не всегда актуально, возможно и на физических серверах, но требует больше денег и сложнее в реализации – *прим. переводчика*).

Мы реализовали отказоустойчивые ВМ основные/резервные на платформе VMware vSphere 4.0, высокоэффективно управляющей полностью виртуализированными машинами x86. Так как VMware vSphere реализует полноценную x86 виртуальную машину, способную работать со всеми операционными системами и приложениями, работающими на платформе x86, мы автоматически в состоянии обеспечить отказоустойчивость для любых x86 операционных систем и приложений (Смотрите список совместимости гостевых ОС и приложений. С остальными возможны нюансы и почти гарантирована невозможность технической поддержки вендора – *прим. переводчика*). Основная технология, обеспечивающая возможность делать запись выполнения первичной системы и идентичность исполнения резервной известна как детерминированный повтор [15]. Отказоустойчивость (FT - Fault Tolerance) VMware vSphere основана на детерминированной переигровке, но с расширением протоколов и функциональности, необходимым для построения полной отказоустойчивой системы. В дополнение к отказоустойчивости аппаратных средств наша система восстанавливает избыточность, автоматически создавая новую резервную виртуальную машину на любом доступном сервере в том же кластере. Сейчас производственные версии детерминированного повтора и VMware FT поддерживают только однопроцессорные ВМ. Запись и переигрывание выполнения мультипроцессорных ВМ весьма проблематичны в реализации, потому что почти каждый доступ к совместно используемой памяти может быть недетерминированной операцией.

Bressoud [3] описывает внедрение прототипа отказоустойчивых ВМ на платформе HP PARISC. У нас похожий подход, но мы внесли некоторые фундаментальные изменения для увеличения производительности и исследовали много альтернатив. Кроме того, мы прорабатывали и реализовывали много дополнительных компонентов системы и много практических вопросов, чтобы построить полноценную систему, эффективную и удобную для применима клиентами запускающими приложения класса предприятия. Подобно большинству других рассмотренных систем, основанных на опыте, мы только пытаемся иметь дело с отказами-остановками сервера [12], которые можно обнаружить раньше, чем неработоспособность системы вызовет проблемы у внешних систем.

Остальная часть документа организована следующим образом. Во-первых, мы описываем нашу базовую конструкцию и детализируем наши фундаментальные протоколы, гарантирующие, что в случае принятия на себя основных функций резервной машиной при недоступности основной никакие данные не будут потеряны. Затем мы описываем подробно многие практические проблемы, которые должны быть решены, чтобы спроектированная система была корректной, надежной, полнофункциональной и работала автоматически. Мы также описываем несколько вариантов моделей, которые возникают при реализации отказоустойчивых ВМ и обсуждаем компромиссы при соответствующем выборе. Затем, мы приводим результаты производительности для некоторых тестов и нескольких реальных приложений. И в конце даем ссылки на связанные работы.

2. Основы разработки FT- систем

Рисунок 1 показывает основную схему нашей системы отказоустойчивых ВМ. Для данной ВМ, которой мы желаем обеспечить отказоустойчивость (основная ВМ), на другом физическом сервере мы запускаем резервную ВМ, синхронизируещейся с первичной и поддерживается по

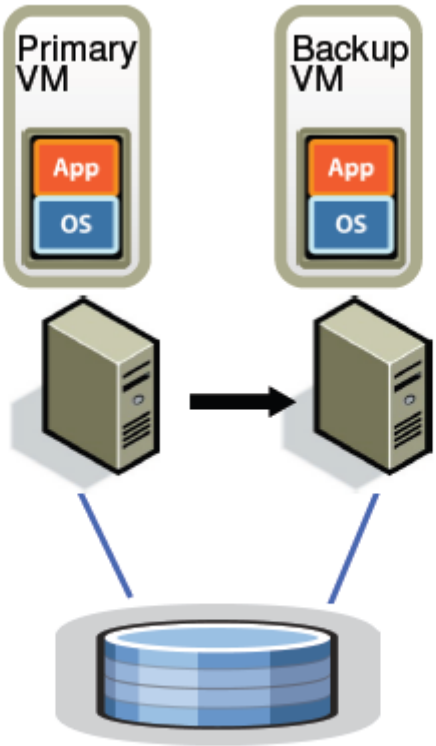

Figure 1: **Basic FT Configuration.**

ходу исполнения в идентичном состоянии с маленькой временной задержкой. Мы говорим, что две ВМ находятся в виртуальном lockstep. Виртуальные диски ВМ находятся на общем хранилище (например Fibre Channel или iSCSI), и поэтому доступны основной и резервной ВМ для ввода и

вывода. (В разделе 4.1 Мы обсудим конфигурацию, в которой первичная и вторичная ВМ расположены на разных не разделяемых дисках.) Т.к. только основная ВМ объявляет о своем присутствии в сети, в следствии чего все входящие сетевые пакеты приходят в основную ВМ. Точно так же и прочие входящие данные (такие как клавиатура и мышь) идут только в основную ВМ.

Все входящие данные, получаемые основной VM, передаются в резервную ВМ через сетевое соединение, называемое каналом регистрации (устанавливается через сеть с VMkernel и галочкой Fault Tolerance logging – *прим. переводчика*). Большая часть нагрузки сервера входящими данными приходится на получаемые по сети и с диска. Как сказано ниже, в разделе 2.1, дополнительная информация передается для гарантии, что резервная ВМ выполнила недетерминированные операции таким же образом как и основная ВМ. Конечный результат состоит в том, что резервная ВМ всегда выполняется тождественно основной ВМ. Однако исходящие данные резервной ВМ гипервизор всегда сбрасывает (отправляет в «черную дыру»), поэтому данные возвращаемые пользователю всегда генерируются только основной ВМ. Как описано в разделе 2.2, первичная и резервная ВМ должны следовать определенному протоколу, включающему явное подтверждение резервной ВМ, чтобы гарантировать, что в случае отказа основной ВМ, никакие данные не будут потеряны.

Процесс быстрого определения отказавшей ВМ (первичная или вторичная) является важнейшей из неописанных проблем. Наша система использует комбинацию heartbeating между соответствующими серверами и мониторинг нагрузки на канал регистрации FT. Кроме того, мы должны гарантировать, что исполняется только одни из серверов (основной или резервный), даже если связь между физическими серверами отсутствует.

В следующих разделах мы предоставляем более подробную информацию о нескольких важных аспектах реализации сервиса. В разделе 2.1 мы освещаем некоторые аспекты технологии детерминированного повтора, гарантирующие, что основная и резервная ВМ остаются синхронными благодаря информации, переданной по каналу регистрации. В разделе 2.2 мы описываем фундаментальные правила нашего протокола FT, гарантирующего, что в случае отказа основной ВМ никакие данные не будут потеряны. В разделе 2.3 мы описываем наши методы обнаружения и корректной реакции на отказ ВМ.

2.1 Реализация записи-повтора

Как мы упомянули, репликация серверов (или ВМ) может быть смоделировано как репликация детерминированных состояний машин. Если два детерминированных состояния машин будут запущены в том же самом начальном состоянии и получат те же самые входящие данные в том же самом порядке, то они пройдут те же самые последовательности состояния и отправят те же самые данные. В самом простом случае одно состояние машины - основное, а другое резервное. Если все входящие данные идут на первичную, то они могут быть переданы резервной копии от первичной через канал регистрации. У полноценного физического компьютера, рассматриваемого как состояние машины, есть широкий набор источников входящих данных от клавиатуры до данных получаемых по сети от клиента. Кроме того, такие недетерминированные события как виртуальные прерывания и чтение количества циклов таймера из процессора, затрагивают состояние машины. Это представляет три проблемы для реального гипервизора, способного управлять любой гостевой операционной системой, которая может запускаться на физической машине: (1) корректный захват всех входящих и недетерминированных данных, необходимых для гарантированно детерминированного выполнения резервной виртуальной машины, (2) правильную передачу входящих и недетерминированных данных резервной виртуальной машине, и (3) сделать это так, чтобы производительность не ухудшилась.

Детерминированный повтор [15] VMware обеспечивает именно эту функциональность для x86 виртуальных машин на платформе VMware vSphere. Детерминированный повтор позволяет захватывать и записывать в файл через поток регистрации входящие данные ВМ и весь возможный недетерминизм, связанный с её работой. Работа ВМ может быть точно воспроизведена позже за счет чтения записей из журнала. Недетерминированные изменения состояния могут или следовать из явных операций, выполненных ВМ, у которых есть недетерминированные результаты (такие как чтение времени суток), или асинхронные события (такие как прерывания), создающие

недетерминизм, потому что точка, в которой они прерывают динамический поток команд, затрагивает выполнение виртуальной машины.

Для недетерминированных операций достаточная информация должна быть зарегистрирована, чтобы стало возможно повторить операцию с тем же самым состоянием и выходными данными. Для недетерминированных событий, таких как прерывания от таймера или завершение ввода/вывода, необходимо также точно зарегистрировать инструкцию, на которой событие имело место. Во время проигрывания событие должно быть повторено в том же самом месте в потоке команд. Детерминированный повтор VMware обеспечивает эффективную запись событий и механизм их воспроизведения, использующий различные методы, включая использование аппаратных счетчиков производительности, разработанных совместно с AMD [2] и Intel [8].

Bressoud [3] упоминает, что делил выполнение ВМ на эпохи, где недетерминированные события, такие как прерывания сохраняются в конце эпохи. Понятие эпохи, кажется, используется в качестве механизма группирования, т.к. заниматься каждым прерыванием отдельно с точным определением инструкции где это произошло слишком накладно. Однако, наш механизм доставки событий достаточно эффективен, чтобы не требовалось использование эпох в детерминированном повторе VMware. Каждое прерывание зарегистрировано, как и зарегистрирован контекст события, что позволяет эффективно подставлять данные в момент выполнения соответствующей инструкции при воспроизведении.

2.2 Протокол FT

Для VMware FT, мы используем детерминированный повтор для создания необходимых записей о работе основной ВМ в журнал, но вместо того, чтобы записывать в журнал на диске, мы передаем их в резервную ВМ через канал регистрации. Резервная ВМ проигрывает записи в режиме реального времени, и следовательно выполняется тождественно к основной ВМ. Однако со строгим протоколом FT мы должны увеличить количество записей о событиях на канале регистрации для уверенности что обеспечиваем отказоустойчивость. У нас следующее фундаментальное требование:

> **Требование к выводу**: если резервная ВМ когда-нибудь после отказа первичной станет действующей, то продолжит выполняться в полной согласованности с данными отправленными первичной ВМ во внешний мир.

Обратите внимание на то, что после того, как отказоустойчивость срабатывает (т.е. резервный ВМ становится действующей после отказа основной ВМ), резервная ВМ, вероятно, из-за множества недетерминированных событий начнет выполняться немного иначе, чем продолжала бы работать основная ВМ. Однако, пока резервная ВМ удовлетворяет «Требованиям вывода», никакое состояние или данные не потеряны во время передачи управления резервной ВМ, клиенты не заметят перерыва или несоответствия в обслуживании.

«Требование вывода» может быть обеспечено, задержкой любого внешнего вывода (как правило, сетевой пакет), пока резервная ВМ не получил всю информацию, позволяющую ей воспроизвести действия, по крайней мере, до операции отправки данных. Первое из необходимых условие состоит в том, что резервная ВМ, должна получить все записи журнала, созданные до операции вывода. Эти записи журнала предоставят возможность работать до последней точки журнала. Предположим, что отказ случился сразу после отправки данных первичной машиной. Резервная ВМ должна знать, что должна продолжить воспроизведение до точки операции вывода и «Пойти вживую» (прекратить повтор и стать основной ВМ, как описано в разделе 2.3) с этой точки. Если резервная ВМ должна начать работать вживую с последней записи в журнале перед операцией вывода, некоторое недетерминированное событие (например, прерывание таймера переданное в ВМ), могут изменить её работу, до того как она выполнит операцию вывода.

Самый легкий способ реализовать вышеупомянутые ограничения «Требования вывода» состоит в том, чтобы при каждой операции вывода создавалась специальная запись в журнале. Затем «Требования вывода» может быть выполнено по этому правилу:

> **Правило вывода**: основная ВМ не может послать данные во внешний мир, пока резервная ВМ не получит и не подтвердит запись в журнале, связанную с операцией, производящей вывод данных.

Если резервная ВМ получила все записи из журнала, включая запись в об операции отправляющей данные, то резервная ВМ будет в способна точно воспроизвести состояние основной ВМ в точке вывода, и поэтому если первичная будет потеряна, резервная корректно придет в состояние совместимое с отправленными данными. С другой стороны, если резервная ВМ станет основной, не получив все необходимые записи из журнала, то её состояние может быстро стать настолько отличным, что оно будет несовместимо с выведенными данными первичной машины. «Правило вывода» до некоторой степени схоже с подходом, описанным в [11], где “внешне синхронный” ввод/вывод на самом деле может быть буферизирован, фактически записан на диск до следующей связи с внешним миром.

Обратите внимание, что в «Правиле вывода» ничего не говорится об остановке работы основной ВМ. Нам нужно только задержать отправку данных, но сама ВМ может продолжить работать. Так как операционные системы не блокируют сеть и запись на диск работая с асинхронными прерываниями, сигнализирующими о завершение операции, ВМ может без проблем продолжить выполнение и не обязательно будет немедленно затронута задержкой вывода. Предыдущая работа [3, 9], напротив указывала, что основная ВМ как правило должна быть полностью остановлена раньше завершения вывода, пока резервная ВМ не признала, что получила всю необходимую информацию от основной ВМ.

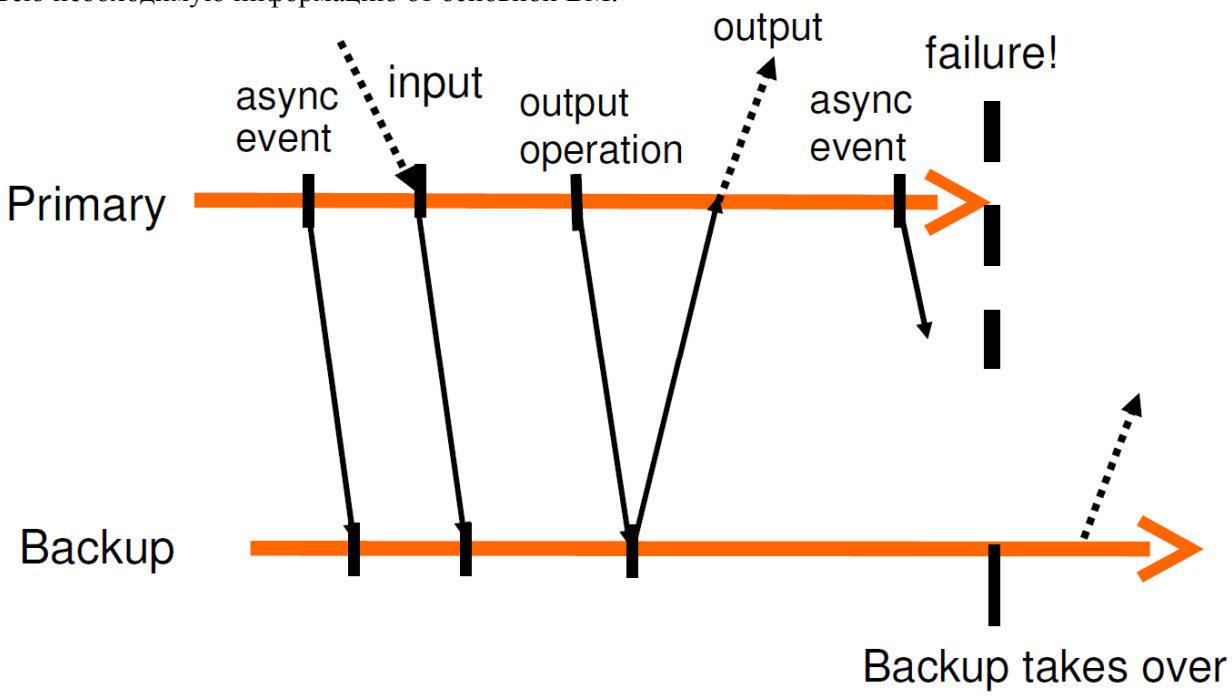

Рисунок 2

В качестве примера, на рисунке 2 мы приводим диаграмму, иллюстрирующую требования протокола FT. Эта диаграмма показывает график времени событий на основной и резервной ВМ. Стрелки, идущие от основной линии до резервной линии поддержки, представляют передачу записей из журнале, а стрелки, идущие от резервной линии к основной, представляют подтверждение получения данных. Информация об асинхронных событиях, операциях ввода и вывода должна быть передана в резервную ВМ как запись журнала и получено подтверждение получения. Как проиллюстрировано на рисунке, вывод во внешний мир отсрочен, пока основная ВМ не получит подтверждение получения данных, связанных с операцией вывода. Из «Правила вывода» следует, резервная ВМ сможет прийти в состояние совместимое с последней операцией вывода основной ВМ. Не будет никакой потери состояния, даже если у первичной будет недетерминированное событие после последнего вывода.

Как сказано в [3, 9], мы не можем гарантировать, что в ситуации сработавшей отказоустойчивости весь вывод произведен только однажды. Без использования транзакций с

двухфазным завершением при отправке данных первичной ВМ, для резервной копии не существует никакого способа узнать, отказ первичной машины произошел непосредственно перед или сразу после отправки последних данных (отказ произошел, но вопрос о моменте отказа – *прим. переводчика*). К счастью, сетевая инфраструктура (включая стандартную работу с TCP) разработана с учетом вероятности потери пакетов и получения дублированных.

Обратите внимание на то, что входящие пакеты для основной ВМ могут быть потеряны также во время ее отказа, и поэтому не будут получены резервной. Однако, входящие пакеты могут быть потеряны из-за самых разных причин, не связанных с отказом сервера. Таким образом, сетевая инфраструктура, операционные системы и все приложения пишутся так, чтобы гарантировано компенсировать потерянные пакеты.

2.3 Диагностирование и реакция на отказ системы.

Как сказано выше, основная и резервная ВМ должны быстро отреагировать, если другая ВМ неработоспособна. Если резервная ВМ отказала, первичная ВМ продолжит работать - то есть, перестанет записывать события (и, следовательно, прекратит отправку записи на канал регистрации) и станет работать в обычном режиме. Если первичная ВМ отказала, и теперь должна начать работать резервная ВМ, процесс немного сложнее. Благодаря отставанию в исполнении, у резервной ВМ, скорее всего, есть некоторое количество записей в журнале, которые она получила и подтвердила, но еще не отработала, так как она еще не достигла соответствующей точки. Резервная ВМ должна продолжить отрабатывать записи журнала, пока не отработает последнюю. В этот момент резервная ВМ выйдет из режима воспроизведения и начнёт работу как обычная ВМ. В сущности, резервная ВМ повышается до основной (и в тот момент без резервной ВМ). Поскольку она уже не резервная ВМ, то теперь она отправляет данные во внешний мир, когда гостевая ОС совершает операцию вывода. В период перехода к нормальному режиму, для обеспечения корректности перехода могут потребоваться некоторые операции конкретного устройства. В частности, для работы сети, VMware FT автоматически извещает устройства (коммутаторы) о новом месте расположения MAC-адреса первичной ВМ (новой первичной ВМ, бывшей резервной). Кроме того, новой первичной ВМ, возможно, потребуется повторить некоторые дисковые операции ввода/вывода (как описано в разделе 3.4).

Обнаружить отказ основной или резервной виртуальных машин можно множеством способов. Для определения работоспособности сервера в связке серверов, на которых запущены отказоустойчивые ВМ, VMware FT использует UDP heartbeating между серверами. Кроме того, VMware FT контролирует трафик журнала, передаваемого от основной в резервную ВМ и подтверждений, отправляемых из резервной ВМ в основную. Пока прерывания таймера регулярны, для функционирования гостевой ОС трафик должен быть регулярным и никогда не прекращаться. Поэтому прекращение потока записей журнала или подтверждений может указывать на отказ ВМ или проблемы сети. Отказ объявляется, если heartbeating или трафик на канале регистрации отсутствует в течении определенного времени (порядка нескольких секунд).

Однако любой подобный метод обнаружения отказа чувствителен к «проблеме разделения мозга» (потери каналов связи между серверами при сохранении работоспособности каждого, компьютерный вариант шизофрении – *прим. переводчика*). Если резервный сервер больше не получает heartbeats от основного, это может говорить о том, что основной отказал, или что между все еще функционирующими серверами потеряны все сетевые соединения. Если резервная ВМ начнет полноценно работать, в то время как основная фактически все еще работает, то вероятно, будет и повреждение данных, и проблемы у клиентов, обращающихся к ВМ. Следовательно, мы должны гарантировать, что при обнаружении отказа будет работать только одна из них. Во избежание «проблемы разделения мозга» для хранения дисков виртуальных машин мы используем общее хранилище. В точке, с которой хочет пойти «в живую» основная или резервная ВМ, она выполняет на общем хранилище атомарную операцию test-and-set (Занято? Если нет, занять - *при. переводчика*). Если операция проходит успешно (было свободно, я занял), ВМ может начинать работать (как единственная, становясь основной с момента создания новой резервной). Если операция не удалась (уже занято), то это значит, что другая ВМ должно быть, уже работает как основная (единственная) и текущая ВМ должна остановить себя («совершить самоубийство»). Если ВМ не может получить доступ к общему хранилищу (пытаясь выполнить атомарную операцию), то

она её повторяет пока не будет успешно выполнена. Обратите внимание, что если общее хранилище не доступно из-за проблем в сети хранилищ, то ВМ скорее всего все равно не могла бы выполнять полезную работу, т.к. виртуальные диски находятся там же. Таким образом использование общего хранилища для решения «проблемы разделения мозга» не добавляет новых проблем.

И заключительный аспект - как только произошёл отказ, и одна из ВМ пошла «в живую», VMware FT автоматически восстанавливает избыточность, создавая новую резервную ВМ на другом хосте. Хотя этот процесс почти не освещен ранее, это фундаментальная основа для создания работающих отказоустойчивых ВМ и требует тщательной проработки. Эта процедура детальнее рассмотрена в разделе 3.1.

2.4 Точки продолжения работы «golive»

Использование детерминированного повторения для обеспечения отказоустойчивости заставило нас добавить интересный механизм к его реализации. Из-за проблем сети или отказа первичной ВМ в любой точке, поток записей из журнала, прочитанных и отработанных резервной ВМ, может также закончиться в любой точке. Возможность завершения отработки журнала регистрации в любом месте должно быть учтено в реализации детерминированного повторения, так как каждый потенциальный модуль проигрывающий записи из журнала (такой как виртуальное устройство) должен проверять и корректно отрабатывать недоступность ожидаемой записи. Например, виртуальное устройство, в следствии проигрывания предыдущей записи в журнале и своего текущего состояния, может ожидать поступление многих дополнительных записей о завершении операций ввода/вывода. Код, который отыгрывает устройство, должен быть написан с проверкой прекращения поступления данных из журнала, чтобы корректно завершить процедуру проигрывания и перевести устройство в состояние позволяющее ВМ работать «вживую».

Для облегчения решения проблемы во многих компонентах системы, мы реализовали точки «golive» (точки продолжения работы «вживую»). Любая одиночная запись в журнале может быть помечена как точка «golive». Смысл в том, что помеченная таким образом запись представляет последнюю запись в серии, необходимых для проигрывания инструкций или особого использования устройства. Если особая операция или инструкция требуют нескольких записей в журнале, то как точка «golive» будет помечена только последняя. На практике, после завершения всех событий для данной инструкции или операции устройства, гипервизор автоматически отмечает последнюю новую запись в журнале как точку «golive».

Точки «golive» используются во время повторения следующим образом. Пока все записи в журнале, прочитанные из канала регистрации буферизованы гипервизором на резервной ВМ, ей разрешено работать только с записями до последней имеющейся точки «golive». Таким образом, ВМ проигрывает до последней имеющейся записи помеченной как точка «golive» и останавливается до получения гипервизором следующей полной серии записей, заканчивающейся помеченной как точка «golive». В результате, виртуальное устройство встретив первую запись из серии связанных с его работой, может предположить, что в журнале есть и все необходимые. Таким образом реализация виртуального устройства не должна содержать все дополнительные проверки и средства восстановления, необходимые в случае прекращения поступления записей из журнала в случайной точке. Точно так же, когда единственная инструкция, выполненная от имени виртуальной машины, генерирует множество записей в журнале, гипервизор воспроизводимой виртуальной машины начинает её эмуляцию только получив все записи необходимые для завершения эмуляции. Схема маркировки незначительно увеличивает задержку повторения ВМ, так как гипервизор генерирующий журнал основной ВМ гарантирует, что последняя запись в журнале при эмуляции каждой единственной инструкции или операции устройства отмечена как точка «golive». Так как значительная задержка резервной ВМ не допустима, основная ВМ также не замедляется за счет точек «golive». (На мой взгляд связь обратная, отсутствие задержек основной момент и определяет возможность исполнение резервной ВМ без таковых. Но у авторов сказано иначе – *прим. переводчика*).

3 Практическая реализация FT

В разделе 2 описаны фундаментальный основы проекта и протоколы FT. Однако, для создания и использования надежной, удобной и автоматической системы, требуется разработать и реализовать много других компонентов.

3.1 Старт и рестарт ВМ защищенных FT

Одним из самых больших дополнительных компонентов, необходимых для эксплуатации, является механизм старта резервной ВМ в состоянии идентичном основной ВМ. Он также используется для перезапуска резервной ВМ после отказа. Следовательно, этот механизм должен быть применимым для работающей основной ВМ, находящейся в произвольном состоянии (случай стартующей ВМ много проще). Кроме того, желательно, чтобы механизм не нарушал ход выполнения основной ВМ, так как это затронет нынешних клиентов.

Для VMware FT, мы адаптировали уже имеющиеся функции VMotion VMware vSphere. VMware VMotion [10] предоставляет возможность миграцию работающих ВМ с одного сервера на другой с минимальным нарушением работы – времени паузы ВМ, как правило – меньше секунды. Мы создали модификацию VMotion, создающую на удаленном сервере точную копию исполняющейся ВМ, но не уничтожая ВМ на текущем сервере. Таким образом, измененный FT VMotion клонирует ВМ на удаленный хост вместо миграции на него. FT VMotion также настраивает канал регистрации, заставляя исходную ВМ перейти в режим регистрации как основную, а клонированную ВМ в режим воспроизведения как резервную. Нормальный VMotion, FT VMotion, как правило, прерывает выполнение основной ВМ меньше, чем на секунду. Следовательно, включение FT на работающей ВМ - легкая, неразрушающая операция.

Другой аспект старта резервной ВМ это выбор сервера, на котором можно ее запустить. Отказоустойчивые ВМ запускаются на группе серверов, у которых есть доступ к общему хранилищу, и таким образом, все ВМ могут выполнятся, как правило, на любых серверах в группе. Это позволяет VMware vSphere восстанавливать избыточность FT, даже при выходе из строя одного или нескольких серверов. VMware vSphere поддерживает объединение в кластеры предоставляя возможность управления и информации о ресурсах. Когда происходит отказ и основной ВМ для обеспечения избыточности требуется новая резервная, она информирует сервис кластеризации.

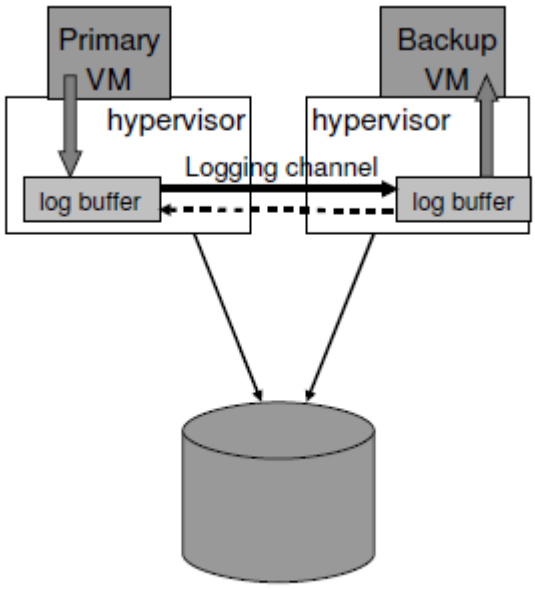

Figure 3: FT Logging Buffers and Channel.

Сервис объединения в кластеры выбирает для резервной ВМ наиболее подходящий сервер, исходя из задействованных ресурсов, нагрузки и прочих ограничений. После этого сервис кластеризации для создания новой резервной ВМ автоматически запускает FT VMotion. Конечно, есть много

дополнительных проблем, таких как повторение попытки, если первая не сумела создать резервную копию и автоматическое обнаружение серверов в кластере после того, как они стали вновь доступны. Результатом служит то, что VMware FT, как правило, может восстановить избыточность ВМ в течение минут после отказа сервера, без значительного перерыва в выполнении отказоустойчивой ВМ.

3.2 Управление каналом регистрации

Есть много интересных деталей реализации управления трафиком в канале регистрации. В нашей реализации гипервизор поддерживает большой буфер под записи регистрации для первичной и вторичной ВМ. Поскольку выполняется основная ВМ, она производит запись в журнала регистрации в буфер, и точно так же резервная ВМ выбирает записи из журнала в своем буфере. Содержание первичного буфера как можно быстрее передают в канал регистрации, откуда они попадают во вторичный буфер сразу по получению. Резервная копия передает подтверждение получения обратно в основную сразу по получению записей из канала регистрации в свой буфер. Это подтверждение позволяет VMware FT определять, когда можно реально отправлять данные, задержанные по «Правилу вывода». Рисунок 3 иллюстрирует этот процесс.

Если у резервной ВМ оказывается пустой буфер журнала регистрации в момент, когда она должна прочитать следующую запись из журнала, то выполнение останавливается пока не будет получена новая запись (или правильнее, новая серия записей по точку «golive» - *прим. переводчика*). Так как резервная ВМ не общается с внешним миром, пауза не затронет клиентов. Точно так же, если у основной ВМ буфером регистрации заполнен, а она должна внести туда запись, останавливается ее выполнение до освобождения достаточного места в буфере (чаще всего за счет передачи в буфер резервной ВМ). Эта остановка выполнения естественный механизм управления потоком, замедляющий основную ВМ, когда интенсивность записи в журнале слишком велика. Однако эта пауза может затронуть клиентов, так как основная ВМ будет полностью остановлена и не реагирует на обращения, пока не сможет сохранить данные в журнале регистрации и продолжить выполнение. Поэтому, наша реализация должно быть создана так, чтобы минимизировать возможность переполнения основного буфера регистрации.

Одной из причина, по которой может переполниться основной буфер регистрации может быть то, что полоса пропускания канала регистрации слишком низкая для существующей скорости генерации записей. В то время как потребляемая полоса пропускания на канале, как правило, не высока (см. разделе 5), мы очень рекомендуем использовать для передачи журнала регистрации сеть на 1 Гбит/с для минимизации риска возникновения узкого места.

Другая причина, переполнения буфера регистрации, состоит в том, что резервная ВМ выполняется слишком медленно и поэтому выбирает записи из журнала с недостаточной скоростью. В целом резервная ВМ должен быть в состоянии повторить выполнение примерно на той же самой скорости, на какой основная ВМ их генерирует. К счастью, накладные расходы записи и воспроизведения в VMware детерминированного повторения примерно одинаковы. Однако, если сервер, на котором выполняется резервная ВМ, загружен сильнее другими ВМ (и следовательно потребляется больше ресурсов), резервная ВМ может не получить достаточно процессорного времени и памяти для выполнения с той же скоростью, что и основная ВМ, несмотря на все усилия планировщика гипервизора.

Есть еще одна причина, почему мы не хотим, чтобы пауза в работе была слишком большой, а заполнение буфера регистрации может привести к неожиданным остановкам. Если основной ВМ отказывает, резервная должна “нагнать”, повторив все подтвержденные записи в журнале, прежде чем пойти работать «в живую» и начать общаться с внешним миром. Время на завершение повторения является задержкой выполнения в основном при отказе. Следовательно, время, через которое резервная копии пойдет «в живую» равно примерно времени обнаружения отказа плюс текущая задержка выполнения. Мы не хотим, чтобы задержка выполнения была большой (больше, чем секунда), так как это добавит значительное время ко времени срабатывания отказоустойчивости (время, через которое резервная ВМ пойдет «в живую»).

Поэтому для избежание чрезмерного отставания резервной ВМ от основной, у нас есть дополнительный механизм, замедляющий при необходимости основную ВМ. В нашем протоколе отправки и подтверждения записей в журнале регистрации, мы посылаем дополнительную

информацию, чтобы определить задержку выполнения в реальном времени между основной и резервной ВМ. Как правило, задержка выполнения - меньше чем 100 миллисекунд. Если резервная ВМ начинает значительно отставать в работе (скажем, больше чем на 1 секунду), VMware FT начинает замедлять основную ВМ, инструктируя планировщик выделять ей немного меньшую долю центрального процессора (первоначально всего на несколько процентов). Используя медленную обратную связь, мы пытаемся постепенно точно определить соответствующую долю центрального процессора для основной ВМ, которая позволит резервной ВМ выполняться с соответствующей скоростью. Если резервная ВМ продолжает отставать, мы продолжаем постепенно уменьшать долю ЦПУ основного ВМ. С другой стороны, если резервная ВМ нагоняет, мы постепенно увеличиваем долю ЦПУ основной ВМ, пока задержка выполнения резервной ВМ не вернется к некоторой небольшой величине.

Обратите внимание на то, что такое замедление основной ВМ случается очень редко, и как правило происходит только, когда система находится в условиях чрезвычайного стресса. Все показатели производительности в раздела 5 включают затраты на любое такое замедление.

3.3 Операции на FT ВМ

Другой практический вопрос касается различных операций контроля (управления) применимых к основной ВМ. Например, если основная ВМ выключена явной командой, резервная ВМ также должна быть остановлена, а не пытаться пойти «в живую». Другой пример: любое управление объемом ресурсов на основной ВМ (таких как увеличение доли центрального процессора) должно быть также применено к резервной. Для выполнения подобных операций по каналу регистрации от основной в резервную передаются специальные управляющие записи, использующиеся для внесения соответствующих изменений.

Большинство операций с ВМ должно быть совершено только на основной ВМ. В этом случае VMware FT посылает все необходимые управляющие записи для внесения соответствующих изменений на резервной ВМ. Единственная операция, которая может выполняться независимо на основной и резервной ВМ, является VMotion. Таким образом, основная и резервная ВМ могут быть перенесены на другой хост независимо друг от друга. Обратите внимание на то, что VMware FT гарантирует, что обе ВМ (основная и резервная) не будут перенесены на один хост, т.к. в этом случае больше не обеспечивалась бы отказоустойчивость.

VMotion основной ВМ создает некоторые проблемы нормальному VMotion, так как резервная ВМ должна быть отключена от источника (основной) и вновь соединена в подходящий момент с основной ВМ, расположенной на новом месте. У VMotion резервной ВМ есть подобная проблема, но имеются и другие сложности. Для нормального VMotion мы требуем, чтобы все незавершенные дисковые операции ввода/вывода были стабилизированы (т.е. закончены) непосредственно перед завершением переключения VMotion. Для основной ВМ эту стабилизацию легко отработать, ожидая завершения физического ввода/вывода и передавая информацию об этом в ВМ. Однако для резервной ВМ, нет никакого легкого способа обеспечить завершение всего ввода/вывода в любой необходимой точке, так как резервная ВМ должна повторить работу основной ВМ и закончить ввод/вывод в той же самой точке выполнения. Основная ВМ может работать под нагрузкой, в которой всегда есть дисковый ввод/вывод во время нормального выполнения. Для решения этой проблемы у VMware FT есть уникальный метод. Когда резервная ВМ в заключительной точке переключения VMotion, она через канал регистрации просит основную ВМ временно стабилизировать все её операции ввода/вывода. Тогда естественно ввод/вывод резервной ВМ будет также стабилизирован в одной точке выполнения, поскольку она повторяет за основной ВМ операцию стабилизации.

3.4 Вопросы реализации дискового ввода/вывода

Есть много тонкостей в реализации решений проблем связанных с дисковым вводом/выводом. Во-первых, надо учесть, что дисковые операции неблокирующие, и с одним и тем же участком диска одновременно могут выполняться несколько операций что создает риск возникновения недетерминизма. Кроме того, наша реализация дискового ввода/вывода использует

DMA для доступа непосредственно к памяти виртуальных машин, таким образом, одновременные дисковые операции, работающие с теми же самым страницам памяти, могут привести к недетерминизму. Наш способ заключается в том, чтобы обнаружить все такие конкурирующие операции дискового ввода/вывода (которые редки) и вынудить их выполниться последовательно одинаковым путем на основной и резервной ВМ. Интересно, что единичная операция дискового чтения, может вызвать конкуренцию, так как массив scatter-gather может сослаться на тот же самый блок памяти многократно, следовательно оставив в итоге содержание блока памяти неопределенным. (scatter/gather DMA - использует список дескрипторов в каждом из которых записаны физические адреса и размеры всех виртуальных страниц памяти выделенной под DMA-буфер, т.е. буфер может быть фрагментирован – *прим. переводчика*) Наше решение состоит в том, чтобы точно также засекать эту конкуренцию ввода/вывода, и в этом случае гарантировать, что итоговое содержание памяти передается в канал регистрации, таким образом, резервная копия получает тоже самое содержание памяти.

Во-вторых, т.к. дисковые операции через DMA получают прямой доступ к памяти ВМ, они могут также конкурировать за него с приложением (или операционной системой) в ВМ. Например, может получиться недетерминированный результат, если приложение/ОС в ВМ читает блок памяти в тоже самое время, когда в него передает данные диск. Эта ситуация также маловероятна, но если она случается, мы должны обнаружить ее и отработать. Одно решение состоит в установке временной защиты памяти на страницах участвующих в дисковых операциях. Защита памяти вызывает прерывание, если ВМ обращается к странице участвующей в незавершенной операции, после чего ВМ останавливается до ее завершения. Поскольку изменение защиты страниц в MMU (Memory Management Unit) дорогостоящая операция, мы предпочли использовать «буферы отскока». «Буфер отскока» - временный буфер, того же самого размера как память, с которой работает дисковая операция. Операция чтения с диска модифицируется для чтения данных в буфер отскока и только после ее завершения данные копируются в память гостевой системы. Точно так же для операций записи на диск, данные, которые пересылаются, сначала копируются в буфер отскока, и операция записи модифицируется, чтобы писать данные из буфера отскока. Использование буфера отскока может замедлить дисковые операции, но мы не заметили, что бы он вызывал существенную разницу в производительности.

В-третьих, есть некоторые проблемы, связанные с отказом при наличии незавершенных дисковых операциях ввода/вывода на основной ВМ и соответственно включении резервной «в живую». Нет никакого способа для новой основной ВМ узнать, была ли дисковая операция ввода/вывода передана диску или успешно закончена. Т.к. дисковый ввод/вывод не был отправлен резервной ВМ во внешний мир, для него не будет никакого явного завершения, которое в конечном счете заставило бы гостевую операционную систему ВМ начинать процедуру аварийного прекращения работы или сброса, в то время как недавно ставшая основной ВМ продолжает работать. Поэтому, мы хотели бы быть уверены, что для каждого ожидающего ввода/вывода в ВМ передается признак его завершения. Мы можем сообщать, что каждый ввод/вывод завершился с ошибкой, так как приемлемо вернуть ошибку, даже при успешном завершении операции. Однако гостевая ОС не может хорошо ответить на ошибки её локального диска. Вместо этого мы повторяем ввод/вывод в процессе выполнения ВМ «в живую». Поскольку мы устранили всю конкуренцию и все операции ввода/вывода непосредственно указывают, с какой памятью и дисковыми блоками они работают, эти операции могут быть повторены, даже если уже закончились успешно (т.е. они - идемпотент).

3.5 Вопросы реализации сетевого ввода/вывода

В VMware vSphere реализовано много способов оптимизации сети для ВМ. Многие из этих способов основаны на асинхронном обновлении гипервизором состояния сетевого устройства виртуальной машины. Например, буферы приема данных могут быть обновлены непосредственно гипервизором во время выполнения ВМ. К сожалению, эти асинхронные обновления состояния ВМ добавляют недетерминизм. Если мы не можем гарантировать, что все обновления происходят в той же самой точке в потоке команд на основной и резервной копиях, выполнение резервной копии может отличаться от первичной.

Самое большое изменение кода сетевой эмуляции для отказоустойчивости - отключение асинхронной сетевой оптимизации. Все обновления состояния сети ВМ, должны быть сделаны, в тот момент когда ВМ не выполняет инструкции, таким образом, мы можем зарегистрировать обновления и повторить их на резервной копии в той же самой точке в потоке команд. Код асинхронно обновляющий кольцевые буферы ВМ с поступающими пакетами, был изменен, чтобы вместо этого заставить гостевую систему послать прерывание гипервизору, чтобы он мог зарегистрировать обновления и затем применить их к ВМ. Точно так же код, который ранее асинхронно брал пакеты из очереди передачи, для FT был отключен, и вместо этого мы требуем, чтобы передача осуществлялась через прерывание передаваемое гипервизору (исключая ниже указанное).

Запрет асинхронных обновлений сетевых устройств вместе с задержкой отправки пакетов, описанной в разделе 2.2, создал некоторые проблемы производительности сети. Мы использовали два подхода для улучшения производительности сети ВМ под управлением FT. Во-первых, мы реализовали группировку пакетов, чтобы уменьшить количество прерываний ВМ (в течении некоторого коротко времени накапливаются пакеты и по переполнению буфера или прошествии времени с момента первого пакета генерируется прерывание, одно на несколько пакетов, что позволяет экономить ресурсы за счет однократного прохождения цепочки обработки прерывания – *прим. переводчика*). Когда данные передаются при достаточно высоком битрейте, мы в состоянии обеспечить одно прерывание на группу пакетов и, в лучшем случае, нулевые прерывания, так как мы можем передавать данные как часть получения новых пакетов (как понял, передача данных может начаться после завершения обработки прерывания по приему, минуя соответственно некоторый объем операций по завершению/началу процедур в драйверах и ядре - *прим. переводчика*). Аналогично, мы можем сократить количество прерываний в ВМ для входящих пакетов, отправив одно прерывание на группу пакетов.

Наша вторая оптимизация производительности сети включает сокращение задержки передачи пакетов. Как сказано выше, мы должны задержать все передаваемые пакеты, пока резервная ВМ не передаст подтверждение получения соответствующих записей из журнала. Ключевым для сокращения задержки передачи является сокращение времени передачи данных резервной ВМ и получения подтверждения. Основа оптимизации в этой области включает обеспечение отправки записей журнала и получения подтверждения без переключения контекста нити (thread). Гипервизор VMware vSphere предоставляет возможность зарегистрировать в стеке TCP функции, которые будут вызываться из контекста отсроченного выполнения (подобно tasklet в Linux) при каждом получении данных по TCP. Это позволяет нам быстро обрабатывать сообщения приходящие по каналу регистрации в резервную ВМ и подтверждения получаемые основной без переключения контекста нити. Кроме того, когда основная ВМ ставит в очередь передачи пакет, планировщик работы в контексте отсроченного выполнения получает указание немедленно выполнить сброс (flush) журнала связанного с исходящей записью (как описано в разделе 2.2).

4 Разработка альтернатив

В нашей реализации VMware FT, мы исследовали много интересных альтернатив. В этом разделе мы рассмотрим некоторые из них.

4.1 Разделяемые диски против неразделяемых

В нашем проекте по умолчанию основные и резервные ВМ разделяют те же самые виртуальные диски. Поэтому, в случае срабатывания отказоустойчивости, содержание общих дисков корректно и доступно. По существу, общий диск считают внешним к основной и резервной ВМ, таким образом, любая запись на общий диск, считается связью с внешним миром. Поэтому, на диск фактически пишет только основная ВМ, причем запись на диск должна быть отложена согласно «Правилу вывода». Модель разделяемого диска - используемая в [3, 9, 7].

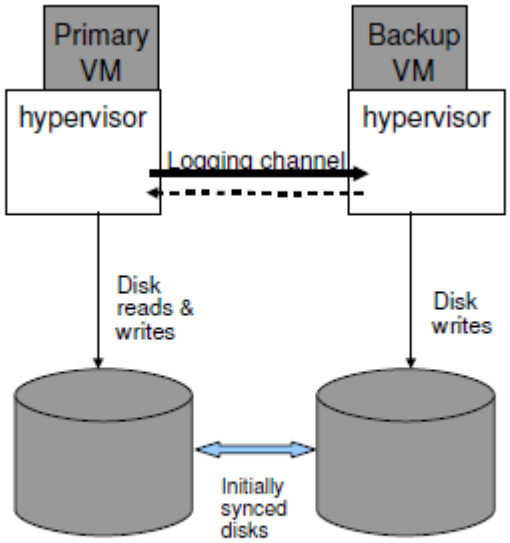

Figure 4: **FT Non-shared Disk Configuration.**

Альтернативная модель работы основной и резервной ВМ, может иметь раздельные (необщие) виртуальные диски. В этом варианте резервная ВМ действительно выполняет все операции записи на свой виртуальный диск, и при этом, что естественно, поддерживает содержание своих виртуальных дисков синхронным с виртуальными дисками основной ВМ. Рисунок 4 иллюстрирует эту конфигурацию. В случае раздельных дисков они по существу считаются частью внутреннего состояния каждой ВМ. Поэтому, запись основной на диск не должна больше откладываться согласно «Правилу вывода». Модель с раздельными дисками полезна когда общее хранилище не доступно для основной и резервной ВМ. Это может быть в случае, если общее хранилище недоступно или слишком дорого, или потому что серверы, на которых расположены основная и резервная ВМ, далеко друг от друга (“дальний FT”). Один из недостатков раздельной модели - две копии виртуальных дисков при включении отказоустойчивости должны явно синхронизироваться каким-то способом. Кроме того, диски могут рассинхронизироваться после отказа, таким образом, они должны повторно синхронизироваться при перезапуске резервной ВМ. Таким образом, FT VMotion должен не только синхронизировать работающее состояние основной и резервной ВМ, но и состояние дисков.

В конфигурации с не разделяемыми дисками, не может быть никакого общего хранилища пригодного для решения проблемы «мозгового разделения». В этом случае система могла бы использовать какую-нибудь внешнюю дополнительную схему, такую как сторонний сервер, с которым могут связаться оба сервера. Если серверы - часть группы больше чем с двумя узлами, система могла бы использовать алгоритм большинства, основанный на членстве в группе. В этом случае ВМ позволили бы начать работать «вживую», если она работает на сервере, являющемся частью общающейся подгруппы, которая содержит большинство оригинальных узлов.

4.2 Выполнение чтения с диска резервной ВМ

В нашем проекте по умолчанию резервная ВМ никогда не читает со своего виртуального диска (или разделенный или необщий). Так как чтение с диска считается входом, естественно передать прочитанные данные через канал регистрации на резервную ВМ.

У альтернативного подхода резервная ВМ должна самостоятельно читать с диска и за счет этого уменьшить объем журнала регистрации. Этот подход может при рабочей нагрузке значительно уменьшить трафик на канале регистрации, возникающий при чтении с диска. Однако у этого подхода есть много тонкостей. Это может замедлить выполнение резервной ВМ, так как резервная ВМ должна выполнять все операции чтения диска и ждать, если они физически не закончены и пока на основной ВМ не будет достигнута точка, где они завершены.

Кроме того, требуются некоторые дополнительные действия для обработки неудачных операций чтения. Если на основной ВМ операция чтения успешна, а на резервной неудачна, на резервной операция должна повторяться пока не удастся, т.к. резервная должна получить в память те же данные, что и основная. С другой стороны, если чтение с диска на основной неудачно, то содержание буфера необходимо передать в резервную через канал регистрации, так как данные в памяти будут неопределенными и не обязательно повторятся чтением с диска резервной ВМ, успешным или неудачным.

Наконец, при чтении с разделяемых дисков в этом альтернативном варианте есть особенность. Если основная ВМ читает данные из конкретного места на диске и через небольшой промежуток времени туда же записывает обновленные, операция чтения должна быть задержана до тех пор, пока резервная не завершит чтение. Эту ситуацию можно обнаружить и корректно отработать, но она усложняет реализацию.

В разделе 5.1 мы приводим некоторые результаты проверки производительности, указывающие, что выполнение чтения диска на резервной ВМ, может вызвать небольшое уменьшение пропускной способности (1-4%) на реальных приложениях, но также может заметно уменьшить полосу пропускания канала регистрации. Следовательно, выполнение дискового чтения на резервной ВМ, может быть полезным в случаях, когда ограничена полоса пропускания канала регистрации.

5 Оценка производительности

В этом разделе мы делаем основную оценку производительности VMware FT под нагрузкой для нескольких приложений и тестирование сети. Для получения этих результатов мы запускаем основную и резервную ВМ на идентичных серверах, каждый с восемью Intel Xeon 2.8 Ghz CPU и 8 Гбайт RAM. Серверы связаны между собой сетью 10 Гбит/с, хотя, как будет показано, во всех случаях реально используется полоса много меньше чем 1 Гбит/с. Оба сервера имеют доступ к общим виртуальным дискам от EMC Clariion подключенным по стандарту 4 Gbit/s Fibre Channel. Для подключения к серверам под рабочей нагрузкой клиент использует сеть на 1 Гбит/с.

В наших тестах производительности мы оцениваем следующие приложения. SPECJbb2005 – промышленный стандарт оценки JAVA-приложений промышленного стандарта, активно использующий память и процессор, но с очень малым количеством операций ввода/вывода. Kernel Compile, рабочая нагрузка, возникающая при компиляцией ядра Linux. Это задание совершает некоторое количество операций чтения/записи на диск и очень интенсивно нагружает процессор и MMU, из-за создания и разрушения многих процессов компиляции. Oracle Swingbench – тестирование нагрузки, на базу данных Oracle 11g в режиме Swingbench OLTP (обработка транзакций онлайн). Тест существенно нагружает диск и сетевой ввод/вывод (в процессе тестирования одновременно нагружает восемьдесят сессий базы данных). DVD MS-SQL Store – тест нагрузка на базу данных Microsoft SQL Server 2005 оценивая DVD Store, с шестнадцать одновременными клиентами.

5.1 Основные результаты оценки производительности

Таблица 1 дает основные результаты тестирования производительности. Для каждого из перечисленных приложений во второй колонке отношение производительности при включенном FT к производительности без FT на той же самой ВМ. Для SPECJbb2005, Kernel Compile, Oracle Swingbench и MS-SQL DVD Store, критерии оценки производительности - соответственно, бизнес операции в секунду, время компиляции в секундах, сделок в секунду и операций в секунду. Отношения подобраны так, чтобы значение меньше 1 указывало, что производительность под FT медленнее. Ясно, что верхним пределом для включения FT является потеря производительности менее 10% (В качестве генеральной линии могу согласиться, но не как абсолютная истина, т.к. при необходимости может оказаться приемлемым и 50% – *прим. переводчика*). SPECJbb2005 полностью computebound (использующий практически только процессор и память) и у него по определению нет никакого времени простоя, но результат теста очень хороший, т.к. у него минимальная недетерминированность, кроме прерываний таймера. Рабочая нагрузка в других тестах содержит дисковые операции ввода/вывода и у ВМ есть некоторое время простоя, таким образом, некоторые

накладные расходы из детерминированного повторения и протокола FT могут быть скрыты фактом, что у ВМ защищенных FT меньше время простоя. В итоге можно сказать, что VMware FT в состоянии поддержать отказоустойчивость ВМ с разумным снижением производительности.

	performance (FT / non-FT)	logging bandwidth
SPECJbb2005	0.98	1.5 Mbits/sec
Kernel Compile	0.95	3.0 Mbits/sec
Oracle Swingbench	0.99	12 Mbits/sec
MS-SQL DVD Store	0.94	18 Mbits/sec

Table 1: Basic Performance Results

В третьей колонке таблицы мы даем среднюю полосу пропускания данных на канале регистрации во время работы приложений. Для этих приложений полоса пропускания регистрации довольно разумна и легко перекрывается сетью на 1 Гбит/с. Фактически, низкие требования к полосе пропускания указывают, что рабочая нагрузка множества ВМ под FT может быть обеспечена той же самой сетью на 1 Гбит/с без снижения производительности.

Мы выяснили, что для ВМ под управлением стандартными операционными системами, такими как Linux и Windows, типичная полоса пропускания регистрации при пассивном состоянии системы, составляет 0.5-1.5 мбит/с (в августе 2014 под ESXi 5.5 большую часть времени это было 0.015 мбит/с – *прим. переводчика*). “Неработающая” полоса пропускания - в основном результат доставки записей о прерываниях таймера. У ВМ с активной рабочей нагрузкой полоса пропускания канала регистрации в основном определяется считанными с диска данными и входящими сетевыми пакетами, пересылаемыми в резервную копию. Мы пришли к выводу, что эвристическое значение для полезной полосы пропускания сети:

> **Полоса пропускания FT** = 1 мегабит/с + 1.2 * (средняя полоса поступления данных с диска [мегабиты/с] + средняя полоса приема данных из сети [мегабиты/с])

Фактор 1.2 является «эмпирическим коэффициентом», учитывающий инкапсуляцию данных принимаемых с диска и по сети в протоколах с соответствующими заголовками и дополнительными записями. Следовательно, полоса пропускания для регистрации может быть намного выше, чем результаты в Таблице 1 для приложений, работающих с большим объемом данных получаемых по сети и с дисков. Для этих видов приложений полоса пропускания канала регистрации может быть узким местом, особенно если канал используется и другими ВМ или прочими способами (например для VMotion – *прим. переводчика)*.

Относительно низкая полоса пропускания, необходимая каналу регистрации для многих реальных приложений, делает отказоустойчивость основанную на повторении довольно привлекательной для «дальней FT», использующей неразделенные диски. Для «дальней FT», где основная и резервная ВМ могут быть отделены на 1-100 километров, оптоволокно легко может обеспечить полосу пропускания 100-1000Mbit/s с временами ожидания меньше чем 10 миллисекунд. Приложениям из Таблице 1 полоса пропускания 100-1000 мегабит/с должна быть достаточной для хорошей работы. Однако обратите внимание, что дополнительное время ожидания передачи в прямом и обратном направлении между основной и резервной ВМ может замедлить передачу по сети и дисковый вывод до 20 миллисекунд. Конфигурация с «дальней FT» будет подходить только для приложений, клиенты которых могут терпеть такое дополнительное время ожидания при каждому запросе.

Для двух приложений мы получили замеры производительности чтения данных с диска на резервной ВМ (как описано в разделе 4.2) в сравнении с отправкой прочитанных данных по каналу регистрации. Для Oracle Swingbench пропускная способность при выполнении чтения на резервной ВМ приблизительно на 4% медленнее; для MS-SQL DVD Store пропускная способность ниже приблизительно на 1%. При этом полоса пропускания канала регистрации уменьшена с 12 мбит/с до 3 мбит/с для Oracle Swingbench, и с 18 мбит/с до 8 мбит/с для MS-SQL DVD Store. Естественно, при большем объеме чтения с диска экономия полосы пропускания может быть намного больше.

Как упомянуто в разделе 4.2, ожидается, что производительность может быть несколько хуже, если чтение диска выполняется на резервном ВМ. Однако в случаях, когда полоса пропускания канала регистрации ограничена (например, конфигурация с «дальней FT»), выполнение чтения диска на резервном VM, может быть полезным.

5.2 Сетевые тесты

Оценка производительности сети для нашей системы по ряду причин может быть довольно сложной. Во-первых, у высокоскоростных сетей может быть очень высокая частота прерываний, требующая высоких показателей регистрации и повторения асинхронных событий. Во-вторых, пакеты получаемые на высокой скорости, вызовут высокий трафик на канале регистрации, так как все подобные пакеты необходимо отправить резервной ВМ по каналу регистрации. В-третьих, оценки, которые посылают пакеты подчиняющиеся «Правилу вывода», которое задерживает отправку сетевых пакетов, пока резервная копия не даст подтверждение получения. Эта задержка увеличит измеренное время ожидания клиента. Эта задержка также может уменьшить полосу пропускания сети с клиенту, так как сетевым протоколам (таким как TCP), вероятно, придется уменьшить скорость передачи по сети, когда увеличится время ожидания передачи в прямом и обратном направлении.

	base bandwidth	FT bandwidth	logging bandwidth
Receive (1Gb)	940	604	730
Transmit (1Gb)	940	855	42
Receive (10Gb)	940	860	990
Transmit (10Gb)	940	935	60

2: Performance of Network Transmit and Receive (all in Mbit/s)

Таблица 2 показывает наши результаты множества тестов, сделанных стандартным тестом netperf. Во всех этих измерениях клиент ВМ и основная ВМ связаны сетью 1 Гбит/с. Первые две строки показывают производительность в случае если основная и резервная ВМ связаны сетью 1 Гбит/с. Третья и четвертая строки - производительность когда серверы связаны сетью 10 Гбит/с, у которой не только более высокая полоса пропускания, но также ниже время ожидания, чем у сети 1 Гбит/с. В качестве грубой оценки, время отклика команды ping между гипервизорами при скорости 1 Гбит/с составляет приблизительно 150 микросекунд, а при 10 Гбит/с приблизительно 90 микросекунд.

Когда FT не разрешен, основная ВМ может получать и передавать данные на скорости близкой (940 мбит/с) к уровню линии 1 Гбит/с. Когда FT разрешен, при входящей рабочей нагрузке полоса пропускания канала регистрации очень велика, так как все входящие сетевые пакеты надо передавать через канал регистрации. Канал регистрации может стать узким местом, как видно из результатов тестов для сети регистрации 1 Гбит/с. Эффект намного меньше для сети регистрации 10 Гбит/с. Когда для FT включен, при рабочей нагрузке, ориентированной на передачу, полоса пропускания канала регистрации значительна, так как все сетевые прерывания все еще должны быть переданы на резервную ВМ. Однако достижимая полоса пропускания сети на передачу, выше, чем полоса пропускания сети на прием. В целом, мы видим, что FT может значительно ограничить полосы пропускания сети при в очень высоких скоростях приема и передачи, но максимально высокие скорости достижимы.

6 Связанные работы

Bressoud и Schneider [3] описали исходную идею программной реализации отказоустойчивости для виртуальных машин реализованную полностью на уровне гипервизора.

Они продемонстрировали возможность хранения резервной виртуальной машины в синхронизации с основной через прототип для серверов с процессорами HP PA-RISC. Однако из-за ограничений архитектуры PA-RISC, они не могли реализовать полностью безопасные, изолированные виртуальные машины. Кроме того, они не реализовали метод обнаружения неудачи или попытки решить любую из практических проблем, описанных в разделе 3. Что еще более важно, они наложили на свой протокол FT много ненужных ограничений. Во-первых, они ввели понятие эпох, где асинхронные события отложены до конца установленного интервала. Понятие эпохи ненужное – они, возможно, наложили его, потому что не могли достаточно эффективно повторить отдельные асинхронные события. Во-вторых, они выдвинули требование, остановки выполнения основной ВМ пока резервная не получит все записи журнала и не подтвердит их получение. Однако должен быть отложен только вывод (такой как сетевой пакет) – сама основная ВМ может продолжить работать.

Bressoud [4] описывает систему, в которой реализована отказоустойчивость в операционной системе (Unixware), и поэтому обеспечивает отказоустойчивость для всех приложений, запущенных на той ОС. Интерфейс системного вызова становится набором операций, которые должны детерминировано реплицироваться. У этой работы есть сходные ограничения и выбор модели как у основанной на гипервизоре.

Napper [9] и Friedman [7] описывают реализацию отказоустойчивых виртуальных машин Java. Они придерживаются модели подобной нашей и Bressoud в отправке информации о входящих данных и недетерминированных операциях на канале регистрации. Как Bressoud, они не сосредотачиваются на обнаружении отказов и восстановлении отказоустойчивости после отказа. Кроме того, их реализация ограничена обеспечением отказоустойчивости для приложений, выполняющихся на виртуальных JAVA-машинах. Эти системы пытаются работать с мультипоточными (multi-threaded) JAVA-приложениями, но требуют или что бы все данные были правильно защищены блокировками, или принудительное использование сериализации доступа к общей памяти.

Dunlap [6] описывает реализацию детерминированного повтора, предназначенного для отладки прикладного программного обеспечения на паравиртуализированной системе. Наша работа поддерживает произвольные операционные системы, выполняющиеся в виртуальных машинах, и осуществляет поддержку отказоустойчивости этих ВМ, которая требует намного более высокого уровня стабильности и производительности.

Cully [5] описывает альтернативный подход поддержки отказоустойчивости ВМ и его реализацию в проекте под названием Remus (http://xgu.ru/wiki/Remus). Согласно этому подходу во время выполнения многократно создаются контрольные точки состояния основной ВМ и передаются на сервер резервного копирования, собирающий контрольные точки. Контрольные точки должны создаваться очень часто (много раз в секунду), так как вывод во внешний мир должен быть отложен до создания очередной точки и подтверждения её получения. Преимущество этого подхода в том, что он одинаково хорош как для однопроцессорных, так и мультипроцессорных ВМ. Основная проблема - то, что при таком методе очень высокие требования к полосе пропускания, чтобы успевать передать возрастающие изменения памяти на каждой контрольной точке. Результаты тестов для Remus, представленные в [5] дают для Kernel Compile и SPECweb от 100% до 225% замедления, при создании 40 контрольных точек в секунду по сети 1 Гбит/с, используемой для передачи изменений памяти. Возможно множество способов оптимизации с целью уменьшении необходимой полосы пропускания сети, но сомнительно что приемлемый результат может быть достигнут на скорости ниже 1 Гбит/с. Наш же рекорд повторения с накладными расходами менее 10%, как правило не требует полосы более 10-50 мбит/с между основным и резервным хостами.

7 Заключение и будущая работа

Мы спроектировали и реализовали в VMware vSphere эффективную и полноценную систему обеспечивающую отказоустойчивость (FT) для виртуальных машин, исполняющихся на кластере серверов. Наш проект основан на репликации выполнения основной ВМ на резервную, расположенную на другом хосте, использующем детерминированное повторение VMware. Если сервер, управляющий основной ВМ отказывает, резервная ВМ немедленно начинает работать без перерыва или потери данных.

В целом, производительность отказоустойчивой ВМ под VMware FT на коммерческой аппаратуре превосходно, и показывает меньше 10% накладных расходов для типичных приложений. Большинство затрат на работу VMware FT порождается детерминированным повторением синхронизирующим основную и резервную ВМ. Минимальный уровень накладных расходов определяется эффективностью детерминированного повторения VMware. Кроме того, полоса пропускания канала регистрации, требуемая для поддержания синхронности основной и резервной ВМ достаточно низкая и часто меньше чем 100 мегабит/с. Она достаточно мала, чтобы было реально обеспечивать существование основной и резервной ВМ на значительном расстоянии (1-100 километров).

Наши тесты VMware FT показали, что эффективная реализация отказоустойчивой ВМ может быть построена на детерминированном повторении. Этот подход позволяет прозрачно обеспечить отказоустойчивость ВМ, управляемых любой ОС и с минимальными накладными расходами. Кроме всего прочего отказоустойчивая система ВМ, действительно полезная клиентам, должна быть надежной, простой в эксплуатации и действующей автоматически. Такая система требует много компонентов кроме поддержки идентичных ВМ. В частности, VMware FT после отказа автоматически восстанавливается избыточность, находя подходящий сервер в кластере и создавая на нем новую резервную ВМ. Решая все необходимые проблемы, мы представляем систему, применимую для реальных приложений в датацентрах клиентов.

В качестве тем будущих разработок нас интересует производительность ранее упомянутых «Дальних FT». Нас также интересует развитие системы в плане работы с частичным отказом аппаратуры. Под частичным отказом аппаратных средств мы имеем в виду потерю функциональности или избыточности сервера, не вызывающей повреждение коррупцию или потерю данных. В качестве примера можно привести потерю всех сетевых соединений ВМ или потерю избыточного блока питания физического сервера. Если частичный отказ аппаратных средств происходит на сервере, управляющем основной ВМ, во многих случаях (но не всех) было бы разумно немедленно передать управление резервной ВМ. Подобная отказоустойчивость могла бы немедленно восстановить полнофункциональный сервис критической ВМ и гарантировать, что ВМ быстро переместиться с потенциально ненадежного сервера.

Благодарности

Мы хотели бы поблагодарить Krishna Raja, протестировавшего многие аспекты производительности. Так же было много людей, привлеченных в реализацию VMware FT. Среди основных конструкторов детерминированного повторения и основной функциональности FT были Lan Huang, Eric Lowe, Slava Malyugin, Alex Mirgorodskiy, Boris Weissman, и Min Xu. Кроме того, есть много других людей, привлеченных к реализации высокоуровневого управления FT в VMware vCenter и проблем реализации, связанных со специфическими виртуальными устройствами, кроме сетевых и дисковых. Karyn Ritter превосходно организовал большую часть работы.

Ссылки

[1] Alsberg, P., and Day, J. A Principle for Resilient Sharing of Distributed Resources. In Proceedings of the Second International Conference on Software Engineering (1976), pp. 627–644.

[2] AMD Corporation. AMD64 Architecture Programmer's Manual. Sunnyvale, CA.

[3] Bressoud, T., and Schneider, F. Hypervisor-based Fault Tolerance. In Proceedings of SOSP 15 (Dec. 1995).

[4] Bressoud, T. C. TFT: A Software System for Application-Transparent Fault Tolerance. In Proceedings of the Twenty-Eighth Annual International Symposium on Fault-Tolerance Computing (June 1998), pp. 128–137.

[5] Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchison, N., and Warfield, A. Remus: High Availability via Asynchronous Virtual Machine Replication. In Proceedings of the Fifth USENIX Symposium on Networked Systems Design and Implementation (Apr. 2008), pp. 161–174.

[6] Dunlap, G. W., King, S. T., Cinar, S., Basrai, M., and Chen, P. M. ReVirt: Enabling Intrusion Analysis through Virtual Machine Logging and Replay. In Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (Dec. 2002).

[7] Friedman, R., and Kama, A. Transparent Fault-Tolerant Java Virtual Machine. In Proceedings of Reliable Distributed System (Oct. 2003), pp. 319–328.

[8] Intel Corporation. Intelˆ A R 64 and IA-32 Architectures Software Developer's Manuals. Santa Clara, CA.

[9] Napper, J., Alvisi, L., and Vin, H. A Fault-Tolerant Java Virtual Machine. In Proceedings of the International Conference on Dependable Systems and Networks (June 2002), pp. 425–434.

[10] Nelson, M., Lim, B.-H., and Hutchins, G. Fast Transparent Migration for Virtual Machines. In Proceedings of the 2005 Annual USENIX Technical Conference (Apr. 2005).

[11] Nightingale, E. B., Veeraraghavan, K., Chen, P. M., and Flinn, J. Rethink the Sync. In Proceedings of the 2006 Symposium on Operating Systems Design and Implementation (Nov. 2002).

[12] Schlicting, R., and Schneider, F. B. Fail-stop Processors: An Approach to Designing Fault-tolerant Computing Systems. ACM Computing Surveys 1, 3 (Aug. 1983), 222–238.

[13] Schneider, F. B. Implementing fault-tolerance services using the state machine approach: A tutorial. ACM Computing Surveys 22, 4 (Dec. 1990), 299–319.

[14] Stratus Technologies. Benefit from Stratus Continuing Processing Technology: Automatic 99.999% Uptime for Microsoft Windows Server Environments. At http://www.stratus.com/pdf/whitepapers/continuous-processing-for-windows.pdf, June 2009.

[15] Xu, M., Malyugin, V., Sheldon, J., Venkitachalam, G., and Weissman, B. ReTrace: Collecting Execution Traces with Virtual Machine Deterministic Replay. In Proceedings of the 2007 Workshop on Modeling, Benchmarking, and Simulation (June 2007).