УДК 004.75

В. В. Гайдамако,
Кыргызско-Российский (Славянский) университет имени Б.Ельцина, Бишкек
Институт машиноведения, автоматики и геомеханики Национальной академии наук, Бишкек

О МОДЕЛИРОВАНИИ ДИНАМИЧЕСКОЙ МИГРАЦИИ ВИРТУАЛЬНЫХ МАШИН В ПЛАТФОРМЕ SIMGRID

В статье рассматриваются методы динамической миграции виртуальных машин в облачных системах и реализация живой миграции в расширении фреймворка моделирования Simgrid. Рассматриваются методы динамической миграции с пред- и пост-копированием, влияние скорости изменения страниц оперативной памяти на время миграции. Приведены примеры динамической миграции в Simgrid-модели облачной системы.

Ключевые слова: облачные вычисления, центр обработки данных (ЦОД), виртуальная машина (ВМ), гипервизор, миграция ВМ, динамическая миграция, моделирование, Simgrid.

Введение

Облачные вычисления предоставляют пользователю доступ к ресурсам через сеть по запросу в требуемом объеме независимо от местоположения. Ресурсами могут быть не только приложения, но и вычислительные мощности, системы хранения, сети, то есть целая инфраструктура. Основой облачных приложений является технология виртуализации, виртуальные машины (ВМ) с необходимым для обслуживания запроса программным обеспечением размещаются на физических машинах (серверах, хостах). При планировании этого размещения должны учитываться разные факторы – требования соглашения об уровне услуг (Service Level Agreement), показатели производительности, затраты провайдера облака на обслуживание запроса. Потребление электроэнергии и углеродный след [1] должны быть минимальными. В некоторых случаях требуется переместить виртуальные серверы на другую физическую машину – для экономии электроэнергии, балансировки нагрузки, обеспечения доступности сервиса, причем перемещение может производиться как внутри, так и между центрами обработки данных (ЦОД). Так как реальные облачные системы сложны, велики и часто недоступны для натурных экспериментов, в исследовательском сообществе используются фреймворки (платформы) моделирования распределенных и облачных систем. Актуальность, достоверность, гибкость, возможность создания крупномасштабных моделей с использованием открытого фреймворка SimGrid доказана многочисленными публикациями [2], поэтому именно он был выбран для моделирования облачных информационно-измерительных систем [3], включающих, помимо обычных серверов, физические и виртуальные датчики.

Виртуализация и гипервизоры

Виртуализация – процесс, позволяющий нескольким изолированным средам с собственной операционной системой совместно использовать аппаратные ресурсы компьютера. Каждая из таких виртуализированных сред работает с выделенными ей ресурсами – оперативной памятью, вычислительной мощностью, хранилищем, внешними устройствами. Благодаря виртуализации можно переключаться между разными операционными системами на одном сервере без перезагрузки [4]. Виртуализация может быть аппаратной или программной.

Виртуальная машина (ВМ) – это программно-определяемый компьютер, изолированно работающий на физическом компьютере, с установленной отдельно операционной системой и выделенными ресурсами. Физический компьютер, на котором создается ВМ, называется хост-машиной, а виртуальная машина – гостевой. На одной

физической машине может работать несколько виртуальных машин. Виртуальные машины получают доступ к аппаратуре хоста через гипервизор,

Гипервизор, или монитор виртуальных машин – программа, или аппаратная схема, позволяющая одновременно параллельно запускать несколько виртуальных сред (операционных систем) на одном хост-компьютере. Гипервизор обеспечивает изоляцию операционных систем друг от друга, защиту и безопасность, разделение ресурсов между различными запущенными ОС и управление ресурсами [4].

Аппаратная виртуализация – виртуализация с поддержкой специальной процессорной архитектуры. В отличие от программной виртуализации с помощью данной техники возможно использование изолированных гостевых операционных систем, управляемых гипервизором напрямую [5]. Аппаратная виртуализация обеспечивает производительность, сравнимую с производительностью невиртуализованной машины, что дает виртуализации возможность практического использования и влечет ее широкое распространение. Наиболее распространены технологии виртуализации Intel-VT и AMD-V.

Взаимодействия между гостевыми ОС, работающими на одной хост-машине, происходит так, как если бы они выполнялись на разных физических машинах, то есть по сети. Существует два основных типа гипервизора – тип 1 (X), автономный гипервизор, аппаратный гипервизор и тип 2 (V) – гипервизор на основе ОС.

Гипервизоры первого типа устанавливаются непосредственно на аппаратном обеспечении компьютера (bare metal), имеют свои драйверы устройств, свой планировщик и не зависят от хостовой ОС. Обеспечивают высокую производительность, именно они используются в ЦОД и корпоративных приложениях. К этому типу относятся гипервизоры Xen, VMware ESX, Citrix XenServer. KVM использует гипервизор первого типа для хостинга нескольких виртуальных машин в операционной системе Linux.

Гипервизоры второго типа устанавливаются на хостовой ОС. Гостевой код может выполняться прямо на физическом процессоре, но доступ к устройствам ввода-вывода компьютера из гостевой ОС осуществляется через второй компонент, обычный процесс основной ОС – монитор уровня пользователя [4,5]. К таким гипервизорам относятся Microsoft Virtual PC, VMware Workstation, QEMU, Oracle VirtualBox.

Существуют также гибридный (тип 1+). Гибридный гипервизор состоит из двух частей: из тонкого гипервизора, контролирующего процессор и память, а также специальной служебной ОС, работающей под его управлением. Через служебную ОС гостевые ОС получают доступ к физическому оборудованию. Это гипервизоры Microsoft Virtual Server, Sun Logical Domain, OVirt [5].

Контейнерная виртуализация, контейнеризация, виртуализация на уровне операционной системы – метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного. Эти экземпляры с точки зрения выполняемых в них процессов идентичны отдельному экземпляру операционной системы. Ядро обеспечивает полную изолированность контейнеров, поэтому программы из разных контейнеров не могут воздействовать друг на друга [6].

Контейнеры используют ядро хостовой ОС, в контейнере можно запустить ОС только с тем же ядром, что у хостовой ОС, поэтому не нужны дополнительные ресурсы на эмуляцию оборудования и полноценную ОС. Пример – LXD Linux контейнеры, контейнеры Docker [7].

Миграция виртуальных машин в облачной среде

В работе облака может возникнуть необходимость переместить ВМ на другой сервер, иногда в другой ЦОД. Миграция ВМ может понадобиться при балансировке нагрузки, для обеспечения выполнения условий соглашения об уровне обслуживания, в случаях, когда серверу не хватает вычислительной мощности, для экономии

электроэнергии и уменьшения углеродного следа, а также в случае сбоев в работе оборудования. Для экономии электроэнергии ВМ с недогруженных серверов перемещаются на другой сервер, а освободившийся сервер переводится в спящий режим.

Для осуществления миграции ВМ можно или полностью остановить, или приостановить – то есть остановить с сохранением состояния всех работающих приложений, переместить на другой сервер файл ВМ и запустить ее уже на новом сервере. В этом случае ВМ остается недоступной на все время перемещения. Это off-line или холодная миграция.

Динамическая, живая, или онлайн-миграция (Live Virtual Machine Migration) – метод перемещения ВМ, то есть ОС и связанных с ней приложений с одной физической машины на другую без прекращения работы запущенных приложений, то есть приложение остается доступным во время миграции. При осуществлении живой миграции ВМ тоже приостанавливается, но время останова и недоступности значительно меньше.

Для оценки процесса миграции используются следующие показатели:

- общее время миграции – общее время, необходимое для миграции виртуальной машины с ее хост-машины на целевую машину (total migration time);
- время освобождения сервера-источника (eviction time);
- время простоя (downtime). При выполнении динамической миграции виртуальная машина все же будет какое-то время недоступна (время простоя), так как перемещается ВМ в состоянии приостановки (suspended). Это время должно быть минимальным [8].

Производительность миграции (Migration performance) – это общее время миграции и общее время простоя. Факторы, влияющие на производительность миграции, – ширина полосы пропускания канала, объем передаваемых данных, затраты системных ресурсов на миграцию (migration overhead). Также может рассматриваться изменение производительности приложений, выполняющихся на мигрировавшей машине.

Для осуществления живой миграции необходимо переместить: состояние устройств, сетевых подключений, ОС и всех приложений, страницы оперативной памяти (Memory) и, если нужно, данные на внешнем носителе (Storage). Данные на внешнем носителе не переносятся, если они доступны с целевого хоста, на который перемещается ВМ, то есть исходный и целевой хост находятся в кластере, используется сетевое хранилище – Network Access Storage или Storage Area Network (NAS, SAN), к которому оба хоста имеют доступ [9].

Существуют различные подходы к выполнению живой миграции – с предварительным копированием страниц памяти, с пост-копированием и различные их комбинации.

Подход с предварительным копированием (Pre-copy)

Первыми перемещаются страницы памяти, затем все остальное, этапы миграции:

1. Выбор перемещаемой ВМ и целевого хоста.
2. Предварительное копирование состояния памяти ВМ на целевой хост, ВМ не прекращает работы на исходном хосте. Так как ВМ не прекращает работу, во время перемещения некоторые страницы могут быть изменены. Измененные («грязные») страницы также перемещаются, но во время перемещения они также могут быть изменены. Поэтому этот шаг итеративно повторяется до тех пор, пока объем страниц для перемещения не станет достаточно небольшим.
3. Приостановка работы виртуальной машины и пересылка состояния (не связанного с памятью). Гипервизор старается минимизировать это время.

4. Пересылка оставшегося состояния памяти (оставшееся небольшое количество страниц, измененных во время выполнения стадии 2 и удаление зависимостей на исходном хосте) [10, 11].
5. Передача управления ВМ на целевом хосте и возобновление работы.

«Достаточно небольшой» объем для передачи на шаге 2 означает такой объем, который может быть передан за заданное время приостановки ВМ. Этот объем зависит от пропускной способности сети, а также от интенсивности обновления оперативной памяти при работе приложений.

Подход с пост-копированием (Post-copy)

В подходе с пост-копированием сначала переносятся состояния виртуальной машины, не относящиеся к памяти (состояние ЦП), а затем передаются страницы памяти виртуальной машины. Главное преимущество этого подхода заключается в том, что каждая страница памяти передается не более одного раза, что позволяет избежать дублирующих накладных расходов на передачу при предварительном копировании.

Стратегия пост-копирования может обеспечить выигрыш за счет сокращения общего времени миграции при сохранении жизнеспособности виртуальной машины во время миграции, но время простоя при таком подходе будет больше, чем при стратегии с предварительным копированием [10, 11].

Влияние скорости обновления оперативной памяти ВМ на время миграции

Гипервизор пытается минимизировать время останова ВМ, чтобы оно было незаметно для пользователей и приложений. Например, максимальное время простоя для гипервизора Qemu/KVM (де-факто ВМ менеджер Linux) по умолчанию составляет 30 мс [12]. На этапе переноса страниц памяти при предварительном копировании гипервизор Qemu/KVM итеративно копирует обновленные страницы памяти в место назначения, пока размер оставшихся страниц памяти не станет меньше порогового значения, которое позволит достичь времени простоя в 30 мс. Следовательно время миграции в основном зависит от скорости обновления памяти ВМ и сетевой скорости трафика миграции. ВМ, интенсивно обновляющая страницы памяти, потребует более длительного времени миграции, и в худшем случае (т.е. скорость обновления памяти выше пропускной способности сети) миграция не завершится (т. е. не сойдется в техническом плане).

В [11] показаны результаты экспериментов, исследующих скорость обновления памяти в облачных приложениях (веб-сервер и сервер базы данных) в зависимости от рабочих нагрузок. Было показано, что скорость обновления памяти может быть весьма значительной по сравнению с пропускной способностью сети. Предоставление «наивной» модели, которая просто получает стоимость живой миграции путем деления размера памяти виртуальной машины на доступную пропускную способность сети, является неверным, поскольку скорость обновления памяти определяет продолжительность этапа 2 (перенос страниц памяти ВМ) алгоритма предварительного копирования. Критически важно учитывать влияние обновлений памяти, чтобы точно оценить время миграции и генерируемый сетевой трафик.

Кроме того, время выполнения первых двух этапов зависит от активности других виртуальных машин и рабочих нагрузок, запущенных в системе:

- сеть – кроме трафика живой миграции, по сети передаются другие данные. Это может быть рабочая нагрузка перемещаемой ВМ или других ВМ, работающих на том же сервере или в том же сегменте сети, трафик от других живых миграций. Всегда надо учитывать, что доступная для живой миграции пропускная способность сети динамически изменяется;

- когда несколько ВМ размещены на одном хосте, они конкурируют друг с другом за получение ресурсов ЦП. Это соперничество может привести к снижению производительности каждой ВМ. Скорость изменения памяти (то есть появление «грязных» страниц, которые перемещаются несколько раз во время живой миграции) виртуальной машины динамически изменяется из-за активности других ВМ, размещенных на том же сервере.

«Наивная» модель вычисляет время миграции как соотношение объема перемещаемых данных и пропускной способности канала миграции, и это время без учета динамических изменений будет вычислено неправильно. Следовательно такой способ вычисления времени миграции не может использоваться при моделировании процесса живой миграции, так как приведет к ошибочным результатам моделирования, далеким от реального поведения.

Фреймворк моделирования должен учитывать конкуренцию в виртуализированных и невиртуализированных системах: если виртуальные машины размещены на одной физической машине, система должна вычислять правильное время ЦП для каждой виртуальной машины. Если сетевое соединение используется совместно с несколькими передачами данных, включая миграции, системе необходимо назначать правильную полосу пропускания для каждой передачи [11].

Представление виртуальных машин в Simgrid

Рассмотрим интерфейс программирования S4U (SimGrid for you), предпочтительный, а с 2023 года главный интерфейс программирования [13]. Основными языками S4U являются C, C++. Модель Simgrid состоит из Actoров, выполняющих функции, заданные пользователем. Actor выполняет активности (Activity), представляющие вычисления, коммуникации, использование диска так, что они учитываются в модели. Активности используют ресурсы – хосты (физические машины – Host), связи (линия связи и сетевые устройства, Link), диски (Disk). Simgrid вычисляет время выполнения активности и управляет запуском акторов. Коммуникации осуществляются через почтовые ящики (MailBox). Акторы могут выполняться не только на физических, но и на виртуальных машинах (VirtualMachine). Виртуальная машина создается на хосте, при этом можно указать количество процессоров и размер оперативной памяти.

```
VirtualMachine *simgrid::s4u::Host::create_vm(const std::string &name,
                            int core_amount)
VirtualMachine *simgrid::s4u::Host::create_vm(const std::string &name,
                    int core_amount, size_t ramsize)
```

Можно получить информацию о работающей ВМ, определить и изменить состояние, то есть стартовать start() и остановить shutdown(), приостановить suspend() и запустить resume(), так же, как это делается в реальном мире.

Динамическая миграция виртуальных машин в Simgrid

Модель живой миграции, реализованная в SimGrid, основана на подходе предварительного копирования. В [11] описано расширение гипервизора Qemu, созданное для использования фреймворком Simgrid при управлении работой ВМ в модели. Для вычисления времени миграции используются параметры:

- R – размер памяти виртуальной машины;
- α (байт/флопс) – интенсивность обновления памяти виртуальной машины, обозначает, сколько страниц памяти помечаются как «грязные» при выполнении одной операция ЦП (в мире моделирования). Но для конечного пользователя указание этого параметра может быть затруднительно, поэтому его можно указать

через объем изменяемых данных (т. е. Мб/с) при загрузке ЦП на 100% (преобразование затем вычисляется внутренне, связь показана в [11]).

Расширение Qemu позволяет получать интенсивность обновления памяти конкретной виртуальной машины, следовательно можно определить интенсивность обновления памяти для изучаемой рабочей нагрузки. Во время живой миграции модель повторяет передачу данных до конца этапа 5 (передача оставшейся небольшой порции страниц памяти).

Размер данных i-й передачи определяется как X_i байт. На этапе 1 алгоритм предварительного копирования отправляет все страницы памяти, т. е. $X_1 = R$. На этапе 2 алгоритм отправляет страницы памяти, обновленные во время предыдущей передачи данных. В [11] показано, что скорость обновления памяти пропорциональна уровню загрузки ЦП, то есть размер обновленных страниц памяти во время передачи данных пропорционален общему количеству долей ЦП, назначенных виртуальной машине в течение этого периода. Таким образом, размер передачи данных следующей фазы можно определить как $X_{i+1} = \min(\alpha Si; R0)$, где Si (плавающие операции, flops) – это общее количество долей ЦП, назначенных виртуальной машине во время i-й передачи данных, а R_0 – размер памяти рабочего набора (набор страниц памяти, используемых во время выполнения операции), используемого приложениями на виртуальной машине. Размер обновленных страниц памяти никогда не превышает размер рабочего набора [11].

Фреймворк моделирования формулирует набор ограничений для определения продолжительности каждой передачи данных, исходя из скорости сети, задержки и других задач связи, использующих то же сетевое соединение. Если задана максимальная пропускная способность трафика миграции, модель контролирует скорость передачи данных миграции так, чтобы не превышать эту пропускную способность. Этот параметр соответствует migrate-set-speed API libvirt [14].

При передаче данных миграции модель оценивает доступную пропускную способность для текущей живой миграции так же, как это делает гипервизор в реальном мире. Текущая пропускная способность трафика миграции вычисляется путем деления размера отправляемых данных на время, необходимое для отправки данных. Это расчетное значение используется для определения приемлемого размера оставшихся данных при миграции на 2 этапе (передача страниц памяти). Если максимальное время простоя составляет 30 мс (т. е. значение по умолчанию Qemu), а пропускная способность трафика миграции оценивается в 1 Гбит/с, на этапе 3 должно оставаться менее 3750 Кб (измененные страницы). Механизм миграции повторяет итерацию этапа 2 (передачу страниц памяти), пока это условие не будет выполнено. На этапе 4 модель [11] создает задачу связи размером Xn байт для передачи оставшихся страниц памяти. После завершения этой последней задачи система переключает хост выполнения виртуальной машины на пункт назначения.

Расширение (плагин) live_migration

Для моделирования живой миграции в Simgrid используется плагин live_migration, его необходимо инициировать с помощью функции void sg_vm_live_migration_plugin_init(). Плагин позволяет установить и получить интенсивность изменения памяти – функции sg_vm_set_dirty_page_intensity, sg_vm_get_dirty_page_intensity; отслеживать изменение страниц – функции sg_vm_dirty_page_tracking_init, старт отслеживания sg_vm_start_dirty_page_tracking, и останов sg_vm_stop_dirty_page_tracking; устанавливать и получать размер рабочего набора – функции sg_vm_set_working_set_memory, sg_vm_get_working_set_memory; устанавливать и получать скорость миграции – sg_vm_set_migration_speed, sg_vm_get_migration_speed(const_sg_vm_t vm); получать время останова sg_vm_get_max_downtime(const_sg_vm_t vm).

Виртуальная машина для миграции создается с помощью функции `sg_vm_t sg_vm_create_migratable(sg_host_t pm, const char* name, int coreAmount, int ramsize, int mig_netspeed, int dp_intensity)`.

Параметры – физическая машина (хост), на которой создается ВМ, имя ВМ, количество процессорных ядер, размер оперативной памяти (ОП), скорость сети, интенсивность обновления ОП.

Миграция осуществляется функцией `sg_vm_migrate(sg_vm_t vm, sg_host_t dst_pm)`, параметрами являются имя ВМ и хост, на который ее перемещают.

Рассмотрим пример использования плагина на платформе, показанной на рисунке 1.

При запуске модели в ЦОД включается только один сервер – контроллер ЦОД cb1, остальные серверы запускаются в зоне физических машин. Машины внутри зоны физических машин связаны между собой сетью с пропускной способностью 1Гб/с, как и контроллер ЦОД с зоной физических машин. Зона пользователей связывается с ЦОД по медленной линии с пропускной способностью 10Мб/с.

В примере на физической машине создаются ВМ и мигрируют на разные физические машины, сначала из зоны физических машин в том же ЦОД, потом на машину из зоны пользователей, отслеживается время простоя и время миграции.

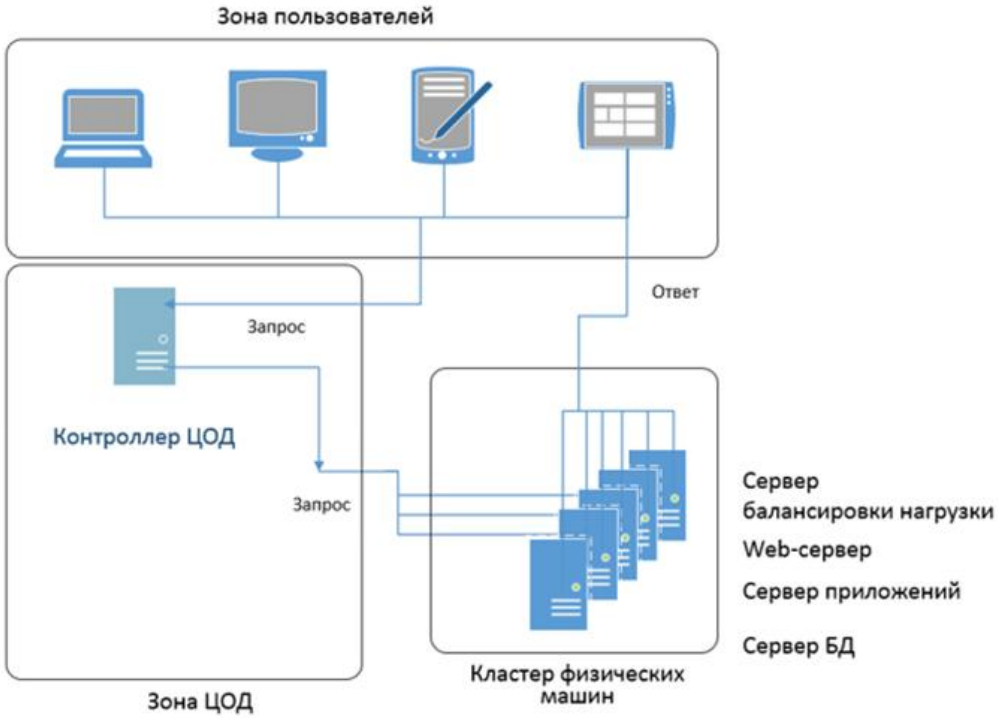

Рисунок 1 – Центр обработки данных с зоной физических машин и зона пользователей [15]

Функция миграции крайне проста [13]:

```
static void vm_migrate(sg4::VirtualMachine* vm, sg4::Host* dst_pm)
{
  const sg4::Host* src_pm = vm->get_pm();
  double mig_sta = sg4::Engine::get_clock();
  sg_vm_migrate(vm, dst_pm);
  double mig_end = sg4::Engine::get_clock();
  XBT_INFO("%s migrated: %s->%s in %g s, speed %g downtime %g",
       vm->get_cname(), src_pm->get_cname(),
       dst_pm->get_cname(), mig_end - mig_sta,
       sg_vm_get_migration_speed(vm), sg_vm_get_max_downtime(vm))
}
```

В таблице 1 представлено время миграции с одной физической машины на другую внутри ЦОД. Мигрируют по очереди: одна ВМ с оперативной памятью 1 Гб (строка 1) и одна с памятью 100 Мб (2), две ВМ с одной машины на другую (3), две ВМ с одной машины на две разные машины (4,5). Увеличение времени миграции в случаях 3-5 обусловлено одновременной передачей страниц памяти по одной линии связи (от одной физической машины).

Таблица 1 – Время миграции ВМ в сети 1 Гб/с

№	Источник			Приемник		Сеть	Время простоя, с	Время миграции, с
	ФМ	Кол-во ВМ	ОП	ФМ	Кол-во ВМ			
1	pm1-0.pm	1	1Гб	pm1-1.pm	1	1Гб/с	0.03	1.08716
2	pm1-0.pm	1	100Мб	pm1-1.pm	1	1Гб/с	0.03	0.112931
3	pm1-0.pm	2	1Гб	pm1-1.pm	2	1Гб/с	0.03	2.16963
4	pm1-0.pm	2	1Гб	pm1-1.pm	1	1Гб/с	0.03	2.16963
5				pm1-2.pm	1	1Гб/с	0.03	2.16963

В таблице 2 представлено время миграции с одной физической машины внутри ЦОД на другую машину из зоны пользователей с медленной линией связи.

Таблица 2 – Время миграции ВМ в сети 10 Мб/с

№	Источник			Приемник		Сеть	Время простоя, с	Время миграции, с
	ФМ	Кол-во ВМ	ОП	ФМ	Кол-во ВМ			
1	pm1-1.pm	1	1 Гб	u-0.uz	1	10 Мб/с	0.03	108.641
2	pm1-1.pm	1	100Мб	u-0.uz	1	10 Мб/с	0.03	11.2183
3	pm1-1.pm	2	1 Гб	u-0.uz	2	10 Мб/с	0.03	216.888
4	pm1-1.pm	2	1 Гб	u-0.uz	1	10 Мб/с	0.03	216.888
5				u-0.uz	1	10 Мб/с	0.03	216.888

Результат очевиден, так как на ВМ нет акторов, они не изменяют страницы, поэтому во время миграции просто один раз копируется вся память. Время простоя (30 мс), или время недоступности ВМ, не меняется, так как нет измененных страниц.

Заключение

В облачных вычислениях миграция виртуальных машин между серверами часто становится необходимой для адаптации к изменяющимся условиям. Так как время простоя и недоступности виртуальной машины должно быть минимальным, осуществляется динамическая (живая) миграция с пред- или пост-копированием страниц оперативной памяти. Для исследования облачных систем из-за дороговизны или недоступности натурных экспериментов используются имитационные модели облачных систем, в которых должны быть представлены и физические, и виртуальные ресурсы, а следовательно, и миграция виртуальных машин. В статье рассмотрено расширение для моделирования живой миграции фреймворка Simgrid, использующее подход с предварительным копированием, описан расчет времени миграции с учетом скорости изменения оперативной памяти и конкурирующего сетевого трафика, приведены примеры использования плагина.

Литература

1. Что такое углеродный след? URL: https://www.microsoft.com/ru-ru/sustainability/learn/reduce-carbon-footprint (дата обращения 25.10.2024)

2. SimGrid Publications. URL: https:/simgrid.org/publications.html (дата обращения 25.10.2024)

3. Гайдамако В.В. Инфраструктура Sensor-Cloud – облачные информационно-измерительные системы. // Проблемы автоматики и управления – 2018. – №2 (35). – С. 109–118.

4. Что такое виртуализация? URL: https://aws.amazon.com/ru/what-is/virtualization/ (дата обращения 25.10.2024

5. Oracle VirtualBox. URL: https://www.virtualbox.org/ (дата обращения 25.10.2024)

6. https://ru.wikipedia.org/wiki/Контейнеризация (дата обращения 25.10.2024)

7. Docker: Accelerated Container Application Development. URL: https://www.docker.com/ (дата обращения 25.10.2024)

8. Soni G, Kalra M (2013) Comparative Study of Live Virtual Machine Migration Techniques in Cloud. Int J Comput Appl 84(14):19–25

9. Костенко В.А., Чупахин А.А. Схемы организации «живой» миграции в центрах обработки данных // Программирование. – 2020. – №5. URL: https://www.elibrary.ru/item.asp?id=43182341

10. Алексанков С. М. Модель процесса динамической миграции с копированием данных после остановки виртуальных машин // Приборостроение. – 2016. – №5. URL: https://cyberleninka.ru/article/n/model-protsessa-dinamicheskoy-migratsii-s-kopirovaniem-dannyh-posle-ostanovki-virtualnyh-mashin (дата обращения: 06.11.2024).

11. Pouilloux, L., Hirofuchi, T., Lebre. SimGrid VM: Virtual Machine Support for a Simulation Framework of Distributed Systems. IEEE Transactions on Cloud Computing, 2018. (1) WWW doi:10.1109/TCC.2015.2481422

12. QEMU. A generic and open source machine emulator and virtualizer. URL: https://www.qemu.org/ (дата обращения 25.10.2024

13. SimGrid Tutorials. URL: https://simgrid.org/doc/latest/intro_concepts.html (дата об-ращения 25.10.2024)

14. libvirt: The virtualization api. URL:http://libvirt.org (дата обращения 25.10.2024)

15. Гайдамако, В.В. 2023. Моделирование сервера в инфраструктуре облачного центра обработки данных на базе платформы SIMGRID // Проблемы автоматики и управления. Вып. 3 (декабрь):81–92. URL: https://pau.imash.kg/index.php/pau/article/view/443