ПРОГРАММИРОВАНИЕ, 2020, № 5, с. 11–15

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК 004.42

СХЕМЫ ОРГАНИЗАЦИИ “ЖИВОЙ” МИГРАЦИИ В ЦЕНТРАХ ОБРАБОТКИ ДАННЫХ

© 2020 г. В. А. Костенко[a],*, А. А. Чупахин[a],**

[a] *Московский государственный университет имени М.В. Ломоносова*
119991 Москва, Ленинские горы, 1, Россия

**E-mail: kost@cs.msu.su*

***E-mail:andrewchup@lvk.cs.msu.ru*

Поступила в редакцию 29.08.2019 г.
После доработки 01.11.2019 г.
Принята к публикации 01.11.2019 г.

В статье описаны схемы организации “живой” миграции в современных гипервизорах и проведен сравнительный анализ потенциальных возможностей схем по критериям: промежуток времени, после которого сервер-источник освободится; общее время миграции; промежуток времени, в течение которого в процессе миграции виртуальная машина недоступна; общее количество переданных данных в процессе миграции и уменьшение производительности мигрирующей виртуальной машины.

DOI: 10.31857/S0132347420050039

1. ВВЕДЕНИЕ

В процессе работы центров обработки данных (ЦОД) создаются новые или удаляются старые виртуальные машины (ВМ). При попытке разместить новую ВМ в ЦОД, может возникнуть следующая ситуация. Суммарного количества свободных физических ресурсов по всем запрошенным требованиям в соглашении о качестве обслуживания (SLA) хватает для размещения ВМ, но не существует физического ресурса, на который можно было бы разместить эту ВМ с выполнением всех ограничений на корректность отображения и запрошенных для него требований в SLA [1, 2]. То есть, в ходе эксплуатации ЦОД возникает фрагментация физических ресурсов, которая приводит к снижению эффективности их использования по критериям: загрузка физических ресурсов ЦОД и процент размещенных запросов из множества поступивших запросов. Фрагментация физических ресурсов может устраняться за счет миграции работающих виртуальных ВМ.

В работе рассмотрены схемы организации “живой” миграции виртуальных машин в рамках одного ЦОД. Указаны гипервизоры, которые поддерживают рассмотренные схемы миграции. Проведено сравнение потенциальных возможностей схем по критериям: время, затрачиваемое на освобождение сервера-источника; общее время миграции; промежуток времени, в течение которого в процессе миграции виртуальная машина недоступна; общее количество переданных данных в процессе миграции и уменьшение производительности мигрирующей виртуальной машины.

2. ЗАДАЧА “ЖИВОЙ” МИГРАЦИИ ВИРТУАЛЬНЫХ МАШИН В ЦОД

Основными факторами, определяющими время миграции, являются:

- пропускная способность выделенного для миграции виртуального канала;
- объем передаваемой информации в байтах во время миграции – зависит от скорости изменения оперативной памяти виртуальной машины.

Для организации “живой” миграции в ЦОД необходимо проложить виртуальный канал между сервером-источником и сервером-приемником таким образом, чтобы он не повлиял на характеристики работающих в ЦОД виртуальных машин. Однако, не всегда возможно проложить канал с такой пропускной способностью, чтобы обеспечить приемлемые затраты времени на миграцию (критерии оценки затрат времени, по которым оценивается эффективность средств миграции приведены в разделе 3). Также возникает проблема поддержания необходимой пропускной способности виртуального канала выделенного для миграции. Так как различные виртуальные каналы могут использовать одни и те же физические каналы связи и одно и то же коммутационное оборудование.

Для ряда виртуальных машин работающих в режиме SaaS возможно получение верхней оцен-

ки скорости изменения памяти, так как для таких виртуальных машин ресурс в основном использует только одно приложение. Для этого потребуется найти историю поведения приложения, для которой количество записей и считываний из памяти максимально.

Для виртуальных машин, работающих в режиме PaaS, может оказаться проблематичным получение верхней оценки скорости изменения памяти. Возможность и метод получения будут зависеть от типа платформы запущенной на виртуальной машине.

Для виртуальных машин, работающих в режиме IaaS получение верхней оценки скорости изменения памяти невозможно. Это связано с тем, что неизвестно какие программы и когда запускаются на виртуальной машине.

3. "ЖИВАЯ" МИГРАЦИЯ В СОВРЕМЕННЫХ ГИПЕРВИЗОРАХ

"Живая" миграция поддерживается во всех современных гипервизорах: KVM [3], Xen [4], VMware [5], Hyper-V [6]. Сравнение эффективности "живой" миграции в различных гипервизорах приведено в статье [7].

"Живая" миграция по сравнению с миграцией остановленной или приостановленной виртуальной машины отличается тем, что во время процесса миграции практически все время виртуальная машина остается доступной. Остановленная виртуальная машина (powered-off): виртуальная машина полностью выключается на сервере-источнике, перемещается, а после включается на сервере-приемнике. Приостановленная виртуальная машина (suspended): состояние всех работающих приложений сохраняется, виртуальная машина переходит в "приостановленное" состояние (аналогично спящему режиму персонального компьютера), далее происходит перемещение виртуальной машины с сервера-источника на сервер-приемник, после этого виртуальная машина вновь запускается с восстановлением состояний всех ранее работающих приложений.

Во всех схемах "живой" миграции используются приостановленные виртуальные машины.

Основные критерии, по которым сравнивают различные схемы миграции:

1. Промежуток времени, после которого сервер-источник освободится (далее eviction time).

2. Общее время миграции (далее total migration time).

3. Промежуток времени, в течение которого в процессе миграции виртуальная машина недоступна (далее downtime).

4. Общее количество переданных данных в процессе миграции виртуальной машины.

5. Уменьшение производительности мигрирующей виртуальной машины (в качестве наиболее объективной оценки уменьшения производительности является увеличение времени выполнения приложений выполняемых на виртуальной машине).

4. СХЕМЫ "ЖИВОЙ" МИГРАЦИИ

"Живая" миграция виртуальных машин включает в себя три этапа [8]:

1. Передача состояния процессора. Передается текущее состояние процессора, информация о размещенных на процессоре в данный момент процессах, используемые в данный момент страницы памяти. Этот этап незначительно влияет на downtime, так как нужно передавать небольшое количество информации.

2. Передача состояния оперативной памяти. Состояние памяти состоит из страниц памяти гостевой операционной системы (т.е. операционной системы виртуальной машины) и страниц памяти всех запущенных процессов внутри виртуальной машины. Некоторые гипервизоры могут отслеживать, какие страницы памяти используются, а какие нет. Это нужно для уменьшения времени миграции, чтобы не передавать лишний раз неиспользованные страницы. Данный этап называется memory migration.

3. Передача содержимого внешнего диска. Это опциональная часть миграции. Диск не нужно мигрировать, если он доступен с сервера-источника и с сервера-получателя, например, в случае Network Attached Storage (NAS). Данная стадия называется storage migration. Если внешний диск не может быть подсоединен к серверу-получателю, то его необходимо перемещать. В этом случае время миграции может значительно увеличиться, так как размер диска может составлять десятки или сотни гигабайт. Так же как и с передачей оперативной памяти, гипервизор может отслеживать, какие блоки данных используются на диске для того, чтобы не передавать неиспользуемые блоки.

Существует две основных схемы организации "живой" миграции: Pre-Copy "живая" миграция [9, 10] и Post-Copy "живая" миграция" [11–13]. В данных схемах рассматриваются первые два этапа миграции. Остальные схемы являются модификацией этих схем.

Pre-Copy "живая" миграция [9, 10]. Оперативная память копируется с сервера-источника на сервер-получатель итерациями, сама виртуальная машина в процессе миграции остается на сервере-источнике. На первой итерации копируется вся память, на последующих итерациях копируются только те страницы, над которыми были произведены изменения. После того как количество итерации превысило некоторый заранее за-

Таблица 1. Сравнение схем организации “живой” миграции

Подход к миграции	Eviction time	Total migration time	Downtime	Transmitted data
Pre-Copy	2	2	1	2
Post-Copy	2	1	3	1
Scatter-Gather	1	2	2	1

данный порог или количество модифицированных страниц на очередной итерации стало меньше заранее заданного порога, виртуальная машина приостанавливается (переводится в состояние suspend) и полностью переносится на сервер-получатель вместе с состоянием процессора и оставшимися модифицированными страницами памяти. Именно этот период неактивности виртуальной машины называется downtime. Для виртуальной машины, которая не интенсивно работает с памятью, downtime будет небольшим, для интенсивно работающих машин во всех итерациях будет передаваться большое количество страниц, когда количество итераций превысит заранее заданный порог, то модифицированных страниц останется все еще много, и поэтому downtime будет большим.

Post-Copy “живая” миграция” [11–13]. Сначала копируется состояние процессора на сервер-получатель. Далее виртуальная машина в процессе миграции уже выполняется на сервере-получателе. Во время миграции сервер-источник посылает страницы памяти на сервер-получатель (эта стадия называется pre-paging) – в надежде, что большинство страниц будет передано на сервер-получатель до того, как виртуальная машина начнет к ним обращаться. Если виртуальная машина обратиться к странице, которой еще нет на сервере-получателе, то будет возбуждено событие remote page fault. В этой ситуации посылается сообщение на сервер-источник с просьбой переслать нужную страницу памяти, далее сервер-источник пересылает ее, сервер-получатель принимает. Чем таких событий будет меньше, тем эффективность этого подхода переноса виртуальной машины будет выше по сравнению с предыдущим подходом. В отличие от подхода, описанного выше, в этом подходе каждая страница памяти передается ровно один раз по сети, что снижает количество использованных сетевых ресурсов. Downtime в этом случае минимальный, так как состояние процессора переносится на первом же шаге на сервер-получатель. Однако деградация производительности сразу после переноса состояния процессора на сервер-получатель может быть больше, чем в Pre-Copy подходе, так как приложения, работающие на виртуальной машине могут быть недоступны до тех пор, пока их рабочие окружения не будут переданы на сервер-получатель.

Гибридная схема. Сначала миграция работает согласно Pre-copy схеме с некоторым порогом на количество итераций. После завершения всех итераций производится передача состояния процессора, далее виртуальная машина уже работает на сервере-получателе. Завершающая стадия миграции работает согласно Post-copy схеме.

Scatter-Gather “живая” миграция [14]. Она основана на Post-Copy схеме. В схеме Scatter-Gather используются посредники. Сначала виртуальная машина переносится с сервера-источника на эти посредники, потом сервер-получатель скачивает с посредников данную виртуальную машину. Посредниками могут выступать другие сервера или middlebox [15]. В описанных выше схемах, значение критерия eviction time равно значению критерия total migration time. Эта схема позволяет снизить значение eviction time во время миграции, если сервер-получатель скачивает виртуальную машину намного медленнее, чем сервер-источник может ее передавать.

Сравнение выше описанных схем организации “живой” миграции приведено в Таблице 1. Данная таблица построена по результатам экспериментов из работы [14]. Для каждой схемы в каждом эксперименте получен ранг от 1 до 3 в соответствии с полученными в работе [14] результатами. 1 – самое малое значение, 3 – самое большое значение. Ранг позволяет понять, какая схема лучше для каждого критерия в среднем по результатам экспериментов.

Однако значения критериев 1–4 характеризующих схему миграции значительно зависят от способа реализации схемы. Влияние на значение этих критериев будет оказывать порядок передачи страниц памяти с сервера-источника на сервер-получатель. Порядок передачи страниц может оказывать сильное влияние на количество передач каждой страницы. Если виртуальная машина делает много обращений к страницам памяти, которые расположены не на том сервере, где выполняется виртуальная машина, то общее количество передач страниц памяти в процессе миграции увеличивается тем сильней, чем больше таких обращений. Вопрос о способе выбора наилучшего порядка передачи страниц памяти является нерешенным.

Также значения критериев 1–4 могут зависеть от способа распределения памяти виртуальной машины и размера страниц. Оптимизация реализации схем миграции по этим характеристикам проблематична. Для такой оптимизации потребу-

ется модификация операционных систем и компиляторов.

При одновременной миграции нескольких виртуальных машин с одного сервера на другой можно уменьшить количество передаваемых данных, используя метод Deduplication migration [14]. Виртуальные машины на одном сервере используют много общих библиотек, так же они могут иметь одну и ту же операционную систему – все это потенциальная возможность уменьшить количество передаваемых страниц. Также существуют способы параллельной миграции, миграции со сжатием данных [14].

5. ПРОБЛЕМА "ЖИВОЙ" МИГРАЦИИ В ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

Вычислительная система реального времени работает в одном из заранее определённых режимов [16]. Например, для комплекса бортового оборудования международной космической станции (МКС) определены следующие режимы работы [17]:

- стандартный режим;
- режим микрогравитации для выполнения научных экспериментов;
- режим сближения и стыковки с транспортными кораблями;
- режим для выхода экипажа в открытый космос;
- режим выживания с отключением наименее важных экспериментов и систем;
- режим аварийного покидания экипажем МКС.

Режим работы характеризуется набором функциональных задач (программ), которые должны выполняться в этом режиме, и набором передаваемых сообщений [16]. Для каждой программы задаются требования к ее выполнению в реальном времени, а для каждого сообщения требования к его передаче в реальном времени.

Требования к выполнению программ в реальном времени могут задаваться одним из способов [18]:

Способ 1: $a_k = (s_k, f_k, t_k)$, где s_k – директивный срок начала выполнения программы k (выполнение программы должно начаться не раньше этого срока); f_k – директивный срок завершения выполнения программы k (выполнение программы должно завершиться до наступления этого срока), t_k – время выполнения программы.

Способ 2: $a_k = (F_k, t_k)$, где F_k – частота ($1/F_k$ – период) выполнения программы; t_k – время выполнения программы. Частота определяет набор отрезков времени выполнения программы, длина которых равна ее периоду.

Второй способ задания требований к выполнению работ в реальном времени может быть сведен к первому. Вычисляется большой цикл как наименьшее общее кратное периодов выполнения работ. Количество запусков работы в большом цикле равно количеству ее периодов в большом цикле. Директивные сроки каждого запуска работы определяются временем начала и завершения соответствующего периода.

Требования к передаче сообщений в реальном времени задаются аналогично требованиям к выполнению программ.

Интерес к подходам организации "живой" миграции в системах реального времени обусловлен требованием "бесшовного" переключения режимов работы системы. При смене режимов работы изменяется набор выполняемых прикладных программ:

– часть программ должна выполняться как в старом, так и новом режиме работы системы,

– часть программ снимается с выполнения,

– добавляются новые программы.

Для каждого режима работы строится свое расписание выполнения программ [16, 18], что бы обеспечить работу системы в реальном времени. В разных режимах работы вычислительной системы реального времени с архитектурой интегрированной модульной авионики [16] программа может выполняться на разных вычислительных модулях. Для программ, которые должны выполняться как в новом, так и в старом режиме требуется обеспечить, что бы во время переключения режима работы системы они не прекращали выполняться или были не доступны не более заданного времени.

6. ЗАКЛЮЧЕНИЕ

Схему для организации "живой" миграции виртуальных машин можно изменять в зависимости от состояний ресурсов в центре обработки данных и от интенсивности работы с памятью мигрирующей виртуальной машины.

На значение критериев эффективности миграции (промежуток времени, после которого сервер-источник освободится; общее время миграции; промежуток времени, в течение которого в процессе миграции виртуальная машина недоступна; общее количество переданных данных в процессе миграции; уменьшение производительности мигрирующей виртуальной машины) кроме выбранной схемы миграции оказывает влияние способ реализации схемы. Сильное влияние может оказать алгоритм, определяющий порядок передачи страниц виртуальной машины с сервера-источника на сервер-получатель. Проблемы построения этого алгоритма связаны с проблемами построения алгоритмов кэширования. "Уровень попаданий" в кэш означает то, насколько велики затраты времени на считывание и запись данных в память. "Уровень попаданий" в страницы памяти, которые размещены на одном сервере

с процессором виртуальной машины, также определяет скорость обращения к памяти и, кроме того, объем данных передаваемых в процессе миграции.

В работе [19] было исследовано влияние “живой” миграции на эффективность работы ЦОД по критериям загрузки физических ресурсов и процента размещенных запросов из исходно поступившего набора запросов на создание виртуальных сетей. Если миграция допустима, то повышается загрузка ресурсов и увеличивается процент размещенных запросов. Однако миграция виртуальных ресурсов без их останова может значительно увеличивать нагрузку на сетевые ресурсы ЦОД. Для миграции виртуальной машины нельзя выделить слишком много сетевых ресурсов без ухудшения характеристик других работающих виртуальных машин, поэтому может потребоваться значительное время для миграции перемещаемых виртуальных машин. Это может приводить к тому, что время размещения новых виртуальных машин будет неприемлемо большим.

7. БЛАГОДАРНОСТИ

Работа выполнена при финансовой поддержке РФФИ, грант № 19-07-00614.

СПИСОК ЛИТЕРАТУРЫ

1. *Костенко В.А., Чупахин А.А.* Подходы к повышению эффективности эксплуатации ЦОД. Программирование. 2019. № 5. С. 36–42.
2. *Зотов И.А., Костенко В.А.* Алгоритм распределения ресурсов в центрах обработки данных с единым планировщиком для различных типов ресурсов. Известия РАН. Теория и системы управления. 2015. № 1. С. 61–71.
3. Kernel Virtual Machine. https://www.linux-kvm.org/page/Main_Page
4. Xen. https://www.xenproject.org/
5. VMware. https://www.vmware.com/
6. Hyper-V. https://docs.microsoft.com/en-us/virtualization/index#pivot=main&panel=containers
7. *Hu W., Hicks A., Zhang L., Dow E.M., Soni V., Jiang H., Bull R., Matthews J.N.* A quantitative study of virtual machine live migration. Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference. ACM, 2013.
8. *Soni G., Kalra M.* Comparative Study of Live Virtual Machine Migration Techniques in Cloud. International Journal of Computer Applications. 2013. V. 84. № 14.
9. *Clark C., Fraser K., Hand S., Hansen J., Jul E., Limpach C., Pratt I., Warfield A.* Live migration of virtual machines. Proceedings Netw. Syst. Des. and Implementation. 2005. P. 273–286.
10. *Nelson M., Lim B.H., Hutchins G.* Fast transparent migration for virtual machines. Proceedings USENIX Ann. Tech. Conf. 2005. P. 25.
11. *Hines M.R., Deshpande U., Gopalan K.* Post-copy live migration of virtual machines. SIGOPS Operating Syst. Rev. 2009. V. 43. № 3. P. 14–26.
12. *Hirofuchi T., Yamahata I.* Yabusame: Postcopy live migration for Qemu/KVM. Presented at KVM Forum, 2011.
13. *Lagar-Cavilla H., Whitney J., Scannell A., Patchin P., Rumble S., De Lara E., Brudno M., Satyanarayanan M.* SnowFlock: Rapid virtual machine cloning for cloud computing. Proceedings EuroSys: Fourth ACM Eur. Conf. Comput. Syst. 2009. P. 1–12.
14. *Deshpande U., Chan D., Chan S., Gopalan K., Bila N.* Scatter-Gather live migration of virtual machines. IEEE Transactions on Cloud Computing, January-March. 2018. V. 6. № 1.
15. Middlebox. http://yuba.stanford.edu/~huangty/sigcomm15_preview/mbpreview.pdf
16. *Костенко В.А.* Архитектура программно-аппаратных комплексов бортового оборудования // Изв. вузов. Приборостроение. 2017. Т. 60. № 3. С. 229–233.
17. *Куминов В., Наумов Б.* Космические компьютеры: открытые стандарты и технологии выходят в открытый космос. Мир компьютерной автоматизации. 2002. № 3. С. 71–79.
18. *Костенко В.А., Смирнов А.С.* Потоковые алгоритмы планирования вычислений в интегрированной модульной авионике. Известия РАН. Теория и системы управления. 2019. № 3. С. 104–114.
19. *Костенко В.А., Чупахин А.А.* Влияние миграции на эффективность использования ресурсов центров обработки данных. Программные системы и инструменты. Тематический сборник. Т. 17. МАКС Пресс Москва, 2017. С. 85–91.