УДК 004.043

Метод выбора контейнера для миграции виртуальной машины в облачном хранилище данных

А. А. Заставенко, А. Ю. Пилипенко, М. А. Скулиш
Национальный технический университет Украины «Киевский политехнический институт», Украина

В данной статье описан механизм балансировки нагрузки для облачного хранилища данных. Описана архитектура системы миграции, простой и сложный алгоритмы многомерной оптимизации системы. Приводятся: метод выбора физического сервера, на который будет осуществляться миграция виртуальной машины на основе оценки потребляемых ресурсов, алгоритм статистического метода выбора контейнера для миграции и метод контроля достаточности ресурсов системы для обработки входной нагрузки, который позволяет исследовать и прогнозировать динамику использования ресурсов. Проведены эксперименты по исследованию свойств предложенного метода.

Ключевые слова: *облачный сервис, умная миграция, масштабирование информационных систем, Platform-as-Infrastructure, кластеризация.*

У даній статті описано механізм балансування навантаження для хмарного сховища даних. Описана архітектура системи міграції, простий і складний алгоритми багатовимірної оптимізації системи. Наводяться: метод вибору фізичного сервера, на який буде здійснюватися міграція віртуальної машини на основі оцінки використовуваних ресурсів, алгоритм статистичного методу вибору контейнера для міграції та метод контролю достатності ресурсів системи для обробки вхідного навантаження, який дозволяє досліджувати і прогнозувати динаміку використання ресурсів. Проведено експерименти з дослідження властивостей запропонованого методу.

Ключові слова: *хмарний сервіс, розумна міграція, масштабування інформаційних систем, Platform-as-Infrastructure, кластеризація.*

This paper presents some mechanism of load balancing for cloud data storage. The architecture of the migration system is described, as well as two algorithms of the system multidimensional optimization: the simple and the complex ones. The article provides a method for selecting physical server the virtual machine will migrate to; a statistical method algorithm for choosing container; and a method of monitoring the adequacy of system resources to process the input load, which allows exploring and predicting the dynamics of resource usage. The experiments aimed to study the proposed method properties have been conducted.

Key words: *cloud service, smart migration, scaling information systems, Platform-as-Infrastructure, clustering.*

1. Общая постановка задачи и её актуальность

Облачное хранилище данных – модель онлайн-хранилища, в котором данные хранятся на многочисленных распределённых в сети серверах, предоставляемых в пользование клиентам. Данная модель используется рядом компаний для размещения программного обеспечения с использованием облачной инфраструктуры с целью масштабирования информационных систем, а также решения проблем пиковых нагрузок и простаивающих ресурсов. Она должна

обеспечивать автоматическую балансировку нагрузки физических серверов, не допуская при этом перегрузку одного сервера и нехватку ресурсов для другого.

Объектом исследования является процесс балансировки нагрузки серверов облачного хранилища данных. Предмет исследования составляют методы выбора серверов для миграции виртуальных машин с целью оптимизации работы системы. Цель работы: повышение эффективности процесса автоматической балансировки нагрузки серверов для решения проблемы пиковых нагрузок и простаивающих ресурсов.

Одним из способов масштабирования информационных систем является использование облачного хостинга. На сегодняшний день существует большое количество компаний, которые предоставляют пользователям услуги для размещения базового программного обеспечения с использованием облачной инфраструктуры и увеличения объема ресурсов системы. При этом обеспечивается автоматическое распределение ресурсов, в зависимости от нагрузки, что позволяет решить проблему пиковых нагрузок и простаивающих ресурсов.

Обеспечение и организация работы кластеров, их размещение на физическом оборудовании ложится на плечи компании предоставляющей облачный сервис. При этом должны быть решены ряд вопросов:

- интерактивное распределение задач клиентов между виртуальными машинами, которые расположены в одном или нескольких кластерах. Эта задача с одной стороны задача балансировки нагрузки, а с другой – задача обеспечения надежности обслуживания клиентов;
- контроль и управление работой кластера. Ресурсов кластера должно всегда быть достаточно всем виртуальным машинам, одновременно работающим на всех серверах кластера.

Таким образом, возникает проблема идентификации момента, когда группа виртуальных машин, которые одновременно обслуживаются на ограниченном физическом ресурсе кластера, создаст нагрузку большую, чем допустимая.

Platform-as-Infrastructure представляет собой изолированный кластер, состоящий из группы серверов и сервисов, которые взаимодействуют как целостная система, предоставляя возможность удобно разворачивать, тестировать, поддерживать и масштабировать систему [2]. Необходимо обеспечивать автоматическую балансировку нагрузки физических серверов, не допуская при этом перегрузку одного сервера и нехватку ресурсов для другого.

Данная система должна обеспечивать:

- непрерывность работы приложений, при этом конечный пользователь не должен знать, каким способом она это делает, а лишь получать качественный сервис без каких-либо перегрузок;
- процесс миграции должен быть целенаправленным и обеспечивать оптимизацию состояния кластера;
- при планировании миграции нужно учитывать, что два контейнера одного окружения не будут располагаться на одном физическом сервере, поэтому она должна поддерживать высокую доступность;

- одним из важных требований к миграции является минимальное время ее выполнения, так как длительная миграция может негативно повлиять на состояние системы;
- система должна обеспечивать защиту от зацикливания, то есть от бесконечной миграции одного и того же пакета;
- должна обеспечиваться защита от сбоев и работа в кластерном режиме, что особенно важно при множественных миграциях.

Таким образом, задачей исследования является отследить загрузку контейнеров и не допустить потерь производительности виртуальных машин связанных с недостатком физических ресурсов принимающего контейнера

2. Обзор существующих методов

Настоящая работа опирается на метод [1-3] живой миграции виртуальных машин, обеспечивающий балансировку нагрузки серверов облачного хранилища данных.

Живая миграция виртуальных машин в основном состоит из передачи его изображения памяти с исходного сервера на сервер назначения. Гипервизор предварительно копирует страницы памяти виртуальной машины к месту назначения, не прерывая ОС или любое из ее приложений. Процесс копирования страницы повторяется в многократных раундах, на которых непрерывно передаются грязные страницы. Обычно, есть ряд страниц, которые модифицируются так часто, что виртуальная машина должна быть остановлена на некоторое время, пока этот ряд полностью не будет передан к месту назначения. Впоследствии виртуальная машина может быть восстановлена на новом сервере.

Живая миграция виртуальных машин позволяет осуществлять движение рабочей нагрузки с почти нулевым прикладным временем простоя. Тем не менее, производительность запущенного приложения будет меньше во время процесса миграции из-за предварительного копирования, вызванного последовательными повторениями памяти. Поскольку продолжительность дополнительных циклов центрального процессора процесса предварительного копирования потребляется и на источнике и на серверах назначения. Дополнительная сумма сетевой полосы пропускания потребляется также, потенциально затрагивая живой отклик интернет-приложений. Кроме того, так как виртуальная машина возобновляется после миграции, ожидается замедление из-за разогрева кэша в месте назначения.

Кроме того, время простоя и производительность приложения, вероятно, будут изменяться по-разному из-за переменных использований памяти и шаблонов доступа. Предыдущие исследования обнаружили, что фактическое время простоя может варьироваться в пределах от 60 мс до 3 с.

3. Описание алгоритма живой миграции

Кластер состоит из физических серверов, на которых размещены виртуальные изолированные контейнеры, выделенные под пользовательские окружения, и управляющее сервера для внутренних системных сервисов [4-7]. Вычислительный контейнер является базовым элементом платформы. Он

представляет собой отдельную виртуальную машину, в которой работает один из компонентов системы (приложение, база данных, балансировщик нагрузки и т.д.). Такое распределение обеспечивает хорошую изоляцию и масштабирование каждого компонента независимо от другого. Все контейнеры являются типизированными и взаимодействуют с ядром по определенному протоколу.

Каждый вид контейнера оптимизируется под доступный объем оперативной памяти, что наиболее актуально для баз данных, так как они изначально настроены на потребление небольшого объема памяти. Из виртуальных контейнеров строится окружение, каждый компонент которого находится в отдельной виртуальной машине. Все компоненты распределяются по разным физическим серверам для обеспечения высокой доступности и надежности. Таким образом, выход из строя одного из серверов не влияет на работу системы, так как имеется его дубликат.

Если система требует большого количества памяти и на текущем физическом сервере наблюдается нехватка ресурсов, наиболее загруженные контейнеры перемещаются на более свободные сервера. Такой процесс называется «умная живая миграция», он позволят избежать нерационального использования ресурсов сервера.

Существует набор сервисов (резервирование, биллинг, сбор статистических данных, безопасность), которые доступны пользователям на верхнем уровне [8]. Все эти услуги подключаются к сервисной шине. Различают три типа веб-клиентов:

- веб-клиенты, работающие с платформой через конкретного хостинг-провайдера;
- веб-клиенты конечного пользователя;
- веб-клиент для администратора платформы.

На нижней ступени архитектуры находятся физические сервера, которые подлежат виртуализации и разделению на виртуальные машины. Также имеется точка входа в платформу, через которую проходят все запросы пользователей к приложениям. Система мониторинга следит за состоянием физических серверов, инфраструктурных контейнеров, веб-приложений и т.д. Если наблюдаются какие-то неполадки в работе системы, она уведомляет об этом администраторов платформы.

Развертывание платформы происходит на наборе физических серверов. Сама платформа кластеризована, каждый структурный компонент работает на отдельной виртуальной машине. Кроме этого, существует второй дублирующий сервер, на котором находятся такие же контейнеры для обеспечения защиты от сбоев.

Недостаток существующих систем живой миграции состоит в том, что при обновлении платформы пользователи, которые работают с ней в этот момент, чувствуют некоторую задержку в работе системы. Такие задержки в основном возникают во время обновления базы данных, так как при изменении структуры базы данных блокируется доступ к ее таблицам.

Решение проблемы состоит в том, чтобы обновление БД производилось итеративно, при этом специальный менеджер обновления должен обеспечивать правильный порядок выполнения обновлений. Все обновления должны быть

обратно совместимыми. Таким образом, для избегания блокировки БД на длительное время используется кластерный режим обновления. Суть этого процесса состоит в следующем: есть два физических сервера, на каждом из которых находится копия платформы, то есть, много контейнеров, работающих одновременно и не мешающих друг другу; когда необходимо обновить платформу, один сервер переходит в состояние “maintenance” и начинает обновляться, при этом экземпляры базы данных реплицируются друг с другом. После обновления запускается синхронизация и обновляется второй сервер, затем оба сервера поднимаются.

Platform-as-Infrastructure должна обеспечивать вертикальное масштабирование (вверх и вниз). Это означает, что виртуальная машина в процессе работы потребляет столько памяти, сколько ей нужно без предварительного резервирования ресурсов. В связи с этим возникает проблема: в случае, если какая-то из виртуальных машин захочет получить максимально необходимое количество ресурсов, возможна ситуация, когда весь ресурс физического устройства память (весь жесткий диск) будет потребляться этой машиной и другие виртуальные машины не смогут расти.

Платформа состоит из большого количества связанных между собой компонент, которые имеют сложную конфигурацию и нуждаются в постоянном управлении. Управлением конфигурацией распределенной системы занимается система автоматизации (рис. 1), которая тиражирует эту конфигурацию на разные площадки.

Рис.1 Структура системы автоматизации

Система автоматизации занимается установкой физических и виртуальных серверов и общих приложений (система аналитики, система мониторинга). Все эти компоненты называются классами. Экземпляры классов называются объектами. Задача системы автоматизации состоит в том, чтобы описать структуру и принцип работы класса и создать несколько его экземпляров с конкретными параметрами. Такая система производит установку и обновление платформы.

Архитектура системы миграции состоит в том, что имеется информация о статистике работы физических серверов и виртуальных машин (контейнеров). Эти данные периодично считываются и формируются в метрики, которые

хранятся в определенном хранилище. Таким образом, обратившись к этому хранилищу, можно в любой момент времени получить информацию о динамике потребления ресурсов по физическому серверу или виртуальному контейнеру. Также имеется информация о количестве ресурсов, которыми располагают физические сервера.

Решение о необходимости миграции, а также о том, что и куда должно мигрировать принимает блок управления. После выбора кандидатов для миграции они помещаются в распределенную очередь, которая обрабатывается специальным процессом. Этот процесс анализирует информацию о количестве элементов в очереди и на основе этой информации выбирает элемент, подлежащий миграции. Процесс физической миграции является синхронным без каких-либо разрывов или возвратов, при этом существует специальный механизм, который следит за тем, чтобы не было ошибок и задержек. Основная задача процесса состоит в том, чтобы обеспечить выбор кандидата для миграции и осуществить его перемещение на определенный физический сервер таким образом, чтобы оптимизировать состояние системы. Таким образом, имеет место задача многомерной оптимизации. Существует несколько алгоритмов решения данной задачи.

Простой алгоритм:

1. Ищется наиболее загруженный сервер (память в приоритете).
2. На сервере выбирается самый оптимальный с точки зрения количества потребляемой памяти (RAM, CPU) и использования жесткого диска контейнер. Оптимальным является контейнер, потребляющий больше всего памяти и меньше всего дискового пространства (для осуществления процесса миграции за минимальное время).
3. Перемещение контейнера.

Данный алгоритм удобный в применении, однако, при большом количестве физических серверов алгоритм является недостаточно оптимальным. Для таких случаев нужно использовать более сложный алгоритм. Его суть состоит в следующем. Определяются специальные правила, которым должно удовлетворять оптимальное решение. Существуют жесткие правила, которые не могут быть нарушены ни при каких обстоятельствах и мягкие правила, которыми можно пренебречь в некоторых случаях. Кроме этого определяются типы решений задачи:

- возможные решения – решения, которые достигаются с нарушением жестких правил (плохие решения);
- осуществимые решения – решения, которые не нарушают жестких условий, но не выполняют часть мягких;
- оптимальные решения – решения, которые выполняю оба типа условий;
- лучшие решения – это оптимальные решения, расчитанные за минимальное время.

Для решения задачи с помощью сложного алгоритма необходимо предварительно определить мягкие и жесткие ограничения.

Жесткие ограничения:

1. Количество ресурсов системы должно быть достаточным для перемещения контейнера. Кроме этого нужно учитывать, что памяти должно быть больше, чем требуется контейнеру.
2. Контейнер не может мигрировать на собственный физический сервер.
3. На одном физическом сервере не должны размещаться контейнеры одного и того же типа. Это условие гарантирует, что в случае сбоя системы будет потеряно минимальное количество данных.

Мягкие ограничения:

1. Мигрировать должны наиболее загруженные с точки зрения оперативной памяти и процессора контейнеры.
2. Мигрировать должны наиболее легкие контейнеры (контейнеры, которые потребляют меньше всего жесткого диска).
3. Целевой физический сервер должен быть наименее загруженным.

Основным недостатком данного алгоритма является отсутствие инструментов учета тенденций изменения скорости потребления ресурсов различными контейнерами. Было проведено исследование эффективности учета статистических данных в процессе выбора контейнера для миграции, а также сервера, на который будет осуществляться миграция.

Для того чтобы правильно подобрать сервер, на который будет осуществляться миграция необходимо оценить тренд изменения динамики использования ресурсов выбранного сервера с учетом нагрузки которую создаст мигрирующий контейнер.

4. Описание метода выбора контейнера для миграции

Для определения момента включения дополнительного технического средства (сервера) необходимо с заданным интервалом времени оценивать текущую статистику загрузки ресурсов, построить тренд, или линейную аппроксимацию, собранных данных о количестве обслуживаемых заявок. На основании тренда, сделать оценку вероятности того, что обслуживание контейнеров расположенных на исследуемом сервере превысит допустимый объем памяти, жесткого диска или загрузку процессора. Миграция необходима в том случае, если хотя бы один ресурс оказывается в дефиците.

На загрузку ресурсов физического сервера одновременно влияет изменение загрузки ресурсов отдельных контейнеров, количество которых более 250. Состояние загрузки ресурсов физического контейнера - это случайная величина, которая согласно Центральной предельной теореме имеет нормальное распределение, так как является суммой большого количества случайных величин.

Таким образом, на основе текущей статистики нагрузки, создаваемой суммой виртуальных контейнеров абонентскими заявкам на тарификацию; оценки верхней границы количества заявок, которые одновременно могут обрабатываться на мощностях доступных серверов будет определена вероятность того, что в течение заданного времени количество поступивших заявок не превысит допустимого значения.

Метод контроля достаточности ресурсов системы для обработки входной нагрузки:

Входные данные:

1) интервал T_1 – интервал времени, в течение которого будет проводиться анализ статистики;

2) интервал dt – интервал дискретизации времени (малый интервал);

3) данные системы мониторинга о количестве потребляемых ресурсов (табл. 1);

Таблица 1. Данные системы мониторинга

t	t_0- dt	t_0-2dt	t_0-3dt	…	T_1
X	x(t)	x(t)	x(t)		

4) интервал времени между проверками на необходимость миграции;

5) Максимально допустимое количество исследуемого ресурса, которое доступно для обслуживания виртуальных машин на сервера.

На рис. 2 x_0 – количество ресурса потребляемое в момент времени t_0, x=at + b – прямая построенная на основе данных статистики на основе метода наименьших квадратов; точки тренда – это возможные значения требуемого ресурса для обслуживания системы на протяжении последующего времени T. Задача оценить вероятность того, что случайный процесс выйдет за пределы допустимого для системы ресурса M.

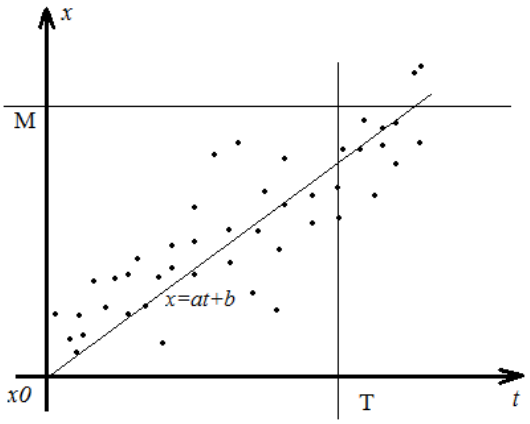

Рис. 2. Прогнозирование динамики использования ресурсов.

Алгоритм метода включает в себя три основных шага:

1. Анализ статистических данных за время (t_0-T_1), где t_0 - текущее время, для которого выполняется расчет. На основе статистических данных системы мониторинга для пары значений (t,x) по методу наименьших квадратов рассчитывается оценка коэффициента a для прямой, аппроксимирующей значения входного нагрузки с табл.1 x = at + b

2. Оценить вероятность P_T того, что в течение заданного времени T количество требуемого ресурса превысит имеющийся ресурс M. Оценка вероятности рассчитывается по формуле (1)

$$P_T = P(\bar{x}_T + 3\sigma > M) \tag{4.1}$$

Среднеквадратическое отклонение определяется по формуле:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4.2}$$

x_i – текущая нагрузка на систему,

$\bar{x}$ – среднее значение занятости ресурса,

На основании полученной оценки принять управляющее решение.

Статистический выбор контейнера для миграции:

ШАГ 1. Методом наименьших квадратов рассчитывается $\hat{a}_i$ – оценка значения коэффициентов для прямых вида $x = \hat{a}_i t + b_i$, которые выражает зависимость количества ресурса (например, оперативной памяти) потребляемого i-м сервером от времени.

ШАГ 2. Из всех доступных для миграции серверов, сервер на котором находится копия мигрирующего контейнера не является доступным, выбрать три физических сервера, для которых значения коэффициентов $\hat{a}_i$, найденных на шаге 1, имеют минимальное значение.

ШАГ 3. Для каждого из трёх выбранных физических серверов сложить поточечно тренды входящей нагрузки сервера и тренды использования ресурсов контейнера, отобранного для миграции. Для каждого нового тренда построить линейную аппроксимацию методом наименьших квадратов и найти соответствующие коэффициенты $\{\hat{a}_1', \hat{a}_2', \hat{a}_3'\}$, где $\hat{a}_i'$ - оценка для параметра определяющего угол наклона прямой отражающей динамику загрузки i-го сервера, исходя из предположения, что в течение оцениваемого периода мигрирующий контейнер обслуживался i-м сервером. Если хотя бы для одного i значение $\hat{a}_i' < 0$, тогда остановка: для миграции выбирается сервер для которого $\hat{a}_i'$ - минимально, иначе переход на шаг 4.

ШАГ 4. Методом контроля достаточности ресурсов оценивается вероятность того, что тренды динамики изменения нагрузки, найденные на шаге 3, не выйдут за установленные пределы допустимой нагрузки каждого i-го физического сервера. В результате выбор останавливается на том физическом сервере, для которого оценка верхней границы вероятности P_T превышения допустимого значения потребляемых ресурсов будет минимальной.

5. Проведение эксперимента для оценки эффективности предложенного метода

С целью подтверждения эффективности предложенного метода, метода выбора контейнера для миграции виртуальной машины, было проведено

моделирование работы пяти контейнеров, которые обслуживали 10 клиентских виртуальных машин. Работа каждой виртуальной машины была продублирована на двух контейнерах. Задача заключалась в том, чтобы, отслеживая загрузку оперативной памяти контейнеров, своевременно выявить возможность перегрузки и принять меры по миграции виртуальной машины.

Моделирование проводилось с помощью пакета GPSS. Для каждого контейнера задавался пул ресурсов, который занимался и освобождался в соответствии с требованиями виртуальных машин, которые были закреплены за контейнерами. Работа виртуальной машины эмитировалась как многоканальное обслуживающее устройство (количество каналов большое, время обслуживания распределено по закону Пуассона), на которое подавался поток заявок, также распределенный по закону Пуассона. На время обслуживания каждой заявки осуществлялось занятие ресурса на заданную величину, то есть из пула соответствующего контейнера вычиталась некоторая заданная величина, ресурс считался полностью занятым, если значение пула равнялось нулю. Если ресурса не хватало заявки отбрасывались. Таким образом обеспечивалась имитация случайного процесса занятия оперативной памяти.

Значения занятости ресурсов контейнеров фиксировались через заданный интервал времени. Для этих статистических данных строился тренд зависимости занятости ресурса от времени, а точнее по методу наименьших квадратов рассчитывался коэффициент наклона аппроксимирующей прямой (3), где (x_k, t_k), $k = \overline{1,n}$ – значение занятости ресурса контейнера и время, когда это значение зафиксировано. А именно, было создано 5 таблиц, в которые каждую секунду вносились данные о занятости ресурсов соответствующих контроллеров, т.е. значение разницы между исходно заданным значением пула и текущим остатком, и соответствующее показание времени моделирования. Расчёт коэффициента аппроксимирующей прямой производился каждые 5 минут на основании статистики трёхсот последних значений загрузки контейнера (n=300), для каждого контейнера отдельно.

$$\hat{a} = \frac{\sum (x_k - \bar{x})(t_k - \bar{t})}{\sum (x_k - \bar{x})^2} \tag{5.1}$$

На основании информации о предельно возможном значении пула ресурсов, статистики трёхсот последних значений загрузки контейнера, а также текущей загрузки ресурсов по формуле (2) рассчитывалось значение σ.

$$\sigma = \sqrt{\frac{1}{300} \sum_{i=1}^{300} (x_i - \bar{x})^2}$$

Тогда исходя из формулы (1)

$$P_T = P(\bar{x}_T > M - 3\sigma)$$

Если при пересчете значение $P_T > 0$, тогда осуществлялось перераспределение виртуальных машин между контейнерами.

Кроме статистики занятия пула ресурсов для каждой виртуальной машины также ведётся статистика занятости ресурса. Исходя из условий модели, каждая из 10-ти виртуальных машин дублируется на двух контейнерах, то есть в процессе работы виртуальной машины ресурс забирается одновременно из пула двух контейнеров. Поэтому после того как расчетное значение $P_T > 0$ для одного из контейнеров, необходимо было выбрать виртуальную машину для миграции, а так же контейнер на который осуществлялась миграция. Для этого из всех виртуальных машин контейнера, где была зафиксирована перегрузка, была выбрана та виртуальная машина, для которой значение оценки коэффициента $\hat{a}$, рассчитанной на основе статистической таблицы потребления ресурсов, было максимально.

После того как виртуальная машина для миграции была выбрана, необходимо выбрать из трёх контейнеров тот, в котором нет дубликата данной виртуальной машины. Для Этого осуществляется суммирование значений статистических таблиц мигрирующей виртуальной машины и контейнера, для которого осуществляется оценка. Для результирующей статистической таблицы рассчитывается значение оценки коэффициента $\hat{a}$. Из всех контейнеров выбирается тот, для которого коэффициент $\hat{a}$ суммарной статистической таблицы будет минимальным.

После миграции в процессе работы виртуальной машины ресурсы занимались из пула того контейнера, куда была осуществлена миграция.

Моделирование показало следующие результаты:

1. Процент потерянных заявок при работе системы без процедуры миграции до 4%.
2. Процент потерянных заявок при работе системы с процедурой миграции до 0,5 %.

6. Выводы по результатам и практическое применение предложенного метода

В данной статье рассматривается способ масштабирования информационных систем с помощью облачного хостинга, что обеспечивает автоматическое распределение ресурсов системы, в зависимости от нагрузки и позволяет решить проблему пиковых нагрузок и простаивающих ресурсов. Были рассмотрены основные задачи, которые должны решать компании, предоставляющие облачный сервис и способы решения этих задач. Был описан метод автоматической балансировки нагрузки физических серверов («умная миграция»), который, не допускает перегрузку одного сервера и нехватку ресурсов для другого.

Для определения момента включения дополнительного сервера необходимо с заданным интервалом времени оценивать текущую статистику загрузки ресурсов. Метод контроля достаточности ресурсов системы для обработки входной нагрузки, предложенный в данной статье позволит принять решение о включении дополнительного сервера на основе данных системы мониторинга о количестве ресурсов, потребляемых системой. Кроме этого, в работе предложен алгоритм статистического метода выбора контейнера для миграции, который

позволяет выбрать наиболее подходящую с точки зрения использования ресурсов виртуальную машину для перемещения на дополнительный сервер.

Предложенный в статье метод выбора контейнера для миграции позволит выбрать наиболее подходящую машину с учетом текущей статистики загрузки ресурсов, и данных о количестве обслуживаемых заявок. Это поможет справиться с проблемой дефицита ресурсов физического сервера. Данная задача может решаться как для системы в целом, когда оценивается статистика количества заявок на тарификацию поступающих в системы онлайн тарификации, так и подсистем, когда оценивается статистика заявок-обращений к подсистемам, системы тарификации. Выбор точки применения метода зависит от выбранной архитектуры привлечения серверов.

ЛИТЕРАТУРА

1. Ye K., Jiang X., Huang D. Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud computing - IEEE International Symposium, 2013. – pp. 267-274.
2. Pahl C., Xiong H. Migration to PaaS clouds – Migration process and architectural concerns – IEEE International Symposium, 2013. – pp.86-91.
3. Jinkyu J. Sung-Hun K., Hwanju K. Analysis of virtual machine live-migration as a method for power-capping. // The Journal of Supercomputing. – 2013. – vol. 66, no 3. – pp. 1629-1655.
4. R. Chu A clustering model for memory resource sharing in large distributed system / Chu R., Xiao N., Lu X. - IEEE, pp. 1-8, 2007.
5. B. Cohen New opportunities for Cloud Application Development / Cohen B. – IEEE International Symposium, 2013. – pp.97-100.
6. I.K.Sawas On Resource Clustering Techniques for Grid Resource Discovery /.Sawas I.K – IEEE International Symposium, 2007. – pp.302-307.
7. Скулиш М. А. Організація роботи групи серверів для забезпечення потреб розподіленої системи тарифікації послуг // Наукові записки Українського науково-дослідного інституту зв'язку. – 2014. – №5(33). – С. 56-64.
8. Скулиш М. А. Метод складання розкладу залучення ресурсів для високонавантажених інформаційних систем // Наукові записки Українського науково-дослідного інституту зв'язку. – 2014. – №6(34). – С. 65-70.
9. Скулиш. М. А., Заставенко А. А. Метод розподілу ресурсів сервера оператора мобільного зв'язку // Вісник НТУУ "КПІ". Серія Радіотехніка, Радіоапаратобудування. – 2015. – № 60 (2015). – С. 96-106.

Надійшла - 28.07.2016.