УДК 004.75

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

РЕАЛИЗАЦИЯ СЕРВИСНОГО БРОКЕРА В SIMGRID-МОДЕЛИ ОБЛАЧНОЙ СИСТЕМЫ

Облачная система невозможна без сервисного брокера. Реализация сервисного брокера в модели облачной системы, создаваемой с использованием фреймворка моделирования Simgrid, позволяет создавать модели облака с произвольным количеством центров обработки данных. Такие модели используются для оценки влияния политики сервисного брокера и алгоритмов балансировки нагрузки на производительность системы.

Ключевые слова: облачные вычисления, сервисный брокер, центр обработки данных (ЦОД), балансировка нагрузки, сервер, моделирование, Simgrid.

Введение

Облачные системы предоставляют пользователям доступ к динамическим, виртуализированным масштабируемым ресурсам через интернет по запросу. Одно из основных свойств облачных систем – гибкость или эластичность, когда ресурсы предоставляются в количестве, необходимом для удовлетворения требований качества, изложенным в соглашении об уровне обслуживания (ServiceLevelAgreement, SLA). Количество ресурсов, задействованных в предоставлении услуги может как увеличиваться, так и уменьшаться автоматически, в зависимости от разных факторов, но прежде всего от нагрузки.

Ресурсы облака находятся в центрах обработки данных (ЦОД), которые могут быть расположены как угодно, в любой локации земного шара. Важным компонентом инфраструктуры облака является сервисный брокер(ServiceBroker), перенаправляющий клиентские запросы на обслуживание в конкретный ЦОД), основываясь на заданном алгоритме (политике). Так как основная цель сервисных брокеров – достижение оптимальной производительности, политика сервисного брокера должна эффективно выбирать лучший ЦОД с учетом времени обработки, затрат и доступности [1, 2]. Внутри ЦОД распределение нагрузки между серверами осуществляет сервер балансировки, действующий в соответствии с заданным алгоритмом балансировки.

Имитационное моделирование распределенных и облачных систем является одним из способов оценки производительности системы при изменении программного обеспечения или состава оборудования [3], когда прямое измерение дорого, затруднительно или невозможно и часто используется для выбора оптимальной политики сервисного брокера и алгоритма балансировки нагрузки внутри ЦОД [4]. Существует множество фреймворков моделирования, очень распространенной стала платформа CloudSim с графическим интерфейсом CloudAnalyst [1-5], позволяющая создавать ЦОД, зоны пользователей, определять политику сервисного брокера и алгоритмы балансировки нагрузки. Однако CloudAnalyst не позволяет задавать внутреннюю структуру ЦОД и характеристики приложения. В данной работе для создания имитационной модели облака была выбрана платформа (фреймворк) моделирования распределенных и облачных приложений Simgrid [6, 7, 8], позволяющая моделировать физические хосты, устройства связи, линии связи, сети, энергопотребление, виртуальные машины и их перемещение между серверами, различные топологии сетей.

Инфраструктура облака

Обобщенная инфраструктура облачной системы как основа для моделирования представлена на рисунке 1. В облаке несколько ЦОД, объединённых высокоскоростными линиями передачи данных. Местоположение ЦОД задается зоной или регионом. Зона определяет время пиковых нагрузок, характеристики линий связи, достижимость ЦОД,

стоимость соединения. Зона может быть связана с географическим местоположением, тогда она задается граничными координатами, если зона прямоугольная, или линиями границ, то есть замкнутыми кривыми с координатами точек, в этом случае зону удобно отображать на карте. Или это может быть просто условная зона. Группы пользователей также привязаны к зонам. Для группы пользователей, кроме зоны, может задаваться время и интенсивность пиковых и внепиковых нагрузок, количество пользователей, характеристики линий связи.

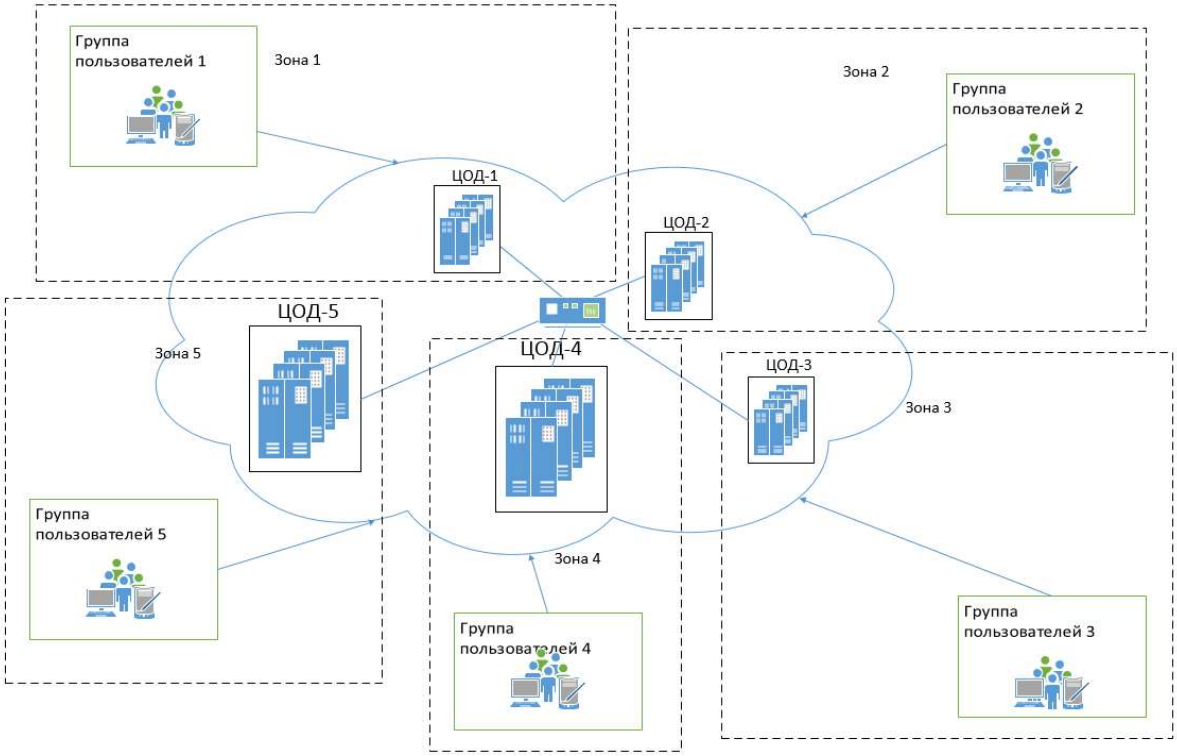

Рисунок 1 – Обобщенная структура облачной системы

Алгоритмы работы (политики) сервисного брокера

Сервисный брокер контролирует маршрутизацию трафика между пользователями и центрами обработки данных и таким образом влияет на время ответа и затраты каждого ЦОД. Рассмотрим некоторые политики маршрутизации.

Маршрутизация, основанная на близости сервиса / Ближайший ЦОД (ServiceProximityBasedRouting/ClosestDataCenterPolicy) [2]. Это простейшая политика сервисного брокера реализована в CloudAnalyst. Сервисный брокер ведет таблицу ЦОД, индексированную по зоне. При получении запроса из базы пользователей производится обращение к сервисному брокеру для определения соответствующего ЦОД (находящегося в той же зоне), запрос будет направлен на контроллер (сервер облака) этого ЦОД.

Брокер получает зону источника запроса и просматривает список «близости» для этой зоны. В этом списке ЦОД сортируются по возрастанию задержки сети, вычисленной для данной зоны. Брокер выбирает первый ЦОД, расположенный в ближайшей зоне. В зоне может быть несколько ЦОД с одинаковой задержкой, тогда ЦОД для обработки запроса выбирается случайным образом.

Маршрутизация, оптимизированная по производительности (Performance Optimized Routing[9]). Эта политика маршрутизации выполняется брокером, ориентированным на лучшее время отклика и является развитием брокера, ориентированного на близость сервиса. Эта политика также реализована в CloudAnalyst.

Брокер ведет список всех доступных ЦОД. При получении запроса брокер, используя маршрутизацию по близости сервиса, находит ближайший ЦОД. После этого брокер определяет ЦОД с наименьшим временем отклика. Это может быть сделано разными способами:

а) брокер активно следит за работой всех дата-центров и ведет журнал времени обслуживания для запросов, запрашивается список времен обслуживания для последних запросов;
б) если время отклика зарегистрировано раньше заданного интервала, время отклика устанавливается в 0. Это означает, что ЦОД не участвовал в обработке (простаивал) в течение, как минимум, этого заданного интервала;
в) сетевая задержка из списка «близости» добавляется к значению, полученному из предыдущих пунктов.

Если последнее время отклика найдено в ближайшем ЦОД, брокер выбирает ближайший ЦОД. В ином случае выбирается или ближайший ЦОД, или ЦОД с лучшим временем отклика, с вероятностью 50:50 (нагрузка балансируется 50:50).

Маршрутизатор с динамической перенастройкой (Dynamically Reconfiguring Router). Брокер рассматривает работу приложения в зависимости от текущей загрузки. Эта политика динамически настраивает количество виртуальных машин в ЦОД в соответствии с текущим временем обработки, которое сравнивается с лучшим достигнутым временем обработки. Используется базовая концепция маршрутизации на основе близости сервиса, но также отслеживается работа приложения, и при необходимости производится его масштабирование, то есть увеличение или уменьшение количества виртуальных машин, выделенных в центре обработки данных [4]. Если время отклика превысит определенный порог, будут созданы дополнительные виртуальные машины. В противном случае, если время ответа уменьшится до определенного порога, количество виртуальных машин будет уменьшено [2,3]. Этот алгоритм реализован в CloudAnalyst не полностью.

Эти три (а фактически две) политики сервисного брокера представлены в CloudAnalyst, но существует множество других алгоритмов/политик, которые тестировались также с помощью CloudAnalyst, который благодаря графическому интерфейсу, относительной простоте настройки и возможности расширения является основным средством разработки и тестирования как политик сервисного брокера, так и алгоритмов балансировки внутри ЦОД. Это, например, усовершенствованный алгоритм ближайшего сервиса (EnhancedProximity-BasedRoutingAlgorithm) [2], маршрутизация, основанная на местоположении ресурсов и оптимизации по стоимости (ResourceLocationandCostbasedPerformanceOptimizedRouting) [9], RoundRobin с весами (WeightedRoundRobin) [10], алгоритм, основанный на вычислении коэффициента эффективности/затрат (CalculatetheratioofEfficiencyoverCostofvirtualmachine) [1] и множество других.

Моделирование облачной системы с использованием Simgrid

Ресурсы моделируемой системы описываются в файле платформы (platformfile) в формате XMLили (в последних версиях) на языке C [8]. Это могут быть физические машины, устройства хранения, сетевых устройств и линий связи. Ресурсы могут создаваться и удаляться программно во время работы модели. Ресурсы находятся в сетевых зонах, которые могут образовывать иерархию. Маршруты между зонами прописываются явно.

Модель SimGrid содержит акторы, выполняющие заданные пользователем функции. Акторы создаются и работают на ресурсах (хосты, линии связи и диски). SimGrid прогнозирует время, затрачиваемое на каждое действие, и соответствующим образом организует работу акторов[6, 8]. Для коммуникации между акторами используются почтовые ящики.

Ход эксперимента по моделированию описывается в файле развертывания (deploymentfile), задается последовательность запуска активностей на хостах и параметры функций для активностей. [6,7,8].

Сценарий моделирования. Рассмотрим систему с тремя зонами. В первой зоне два ЦОД, d11 и d12, во второй зоне ЦОД d21. В третьей зоне нет ЦОД, но в каждой зоне есть база пользователей. Файл платформы:

```
<zone id="world" routing="Floyd">
       <!—Зоныпользователей --Ю
       <zone id="zone1-ubase1" routing="Full">
          <cluster id="ubase11"prefix="ub11-suffix=".uz "radical="0-0»
               speed="1Gf"bw="10MBps"lat="10us"router_id="ub11-ubase11_router.uz"/>
       </zone>
       <zone id="zone2-ubase1" routing="Full">
          <clusterid="ubase21"prefix="ub21-"suffix=".uz"radical="0-
               0"speed="1Gf"bw="10MBps"lat="10us"router_id="ub21-ubase21_router.uz"/>
       </zone>
       <zone id="zone3-ubase1" routing="Full">
          <clusterid="ubase31"prefix="ub31-"suffix=".uz"radical="0-
               0"speed="1Gf"bw="10MBps"lat="10us"router_id="ub31-ubase31_router.uz"/>
       </zone>
       <!—ЗоныЦОД --Ю
       <zone id="zone1-dc1" routing="Floyd">
          <clusterid="dcn11"prefix="dc11-"radical="0-
               10"suffix=".dc"speed="20Gf"core="4"bw="1000MBps"lat="10us"router_id="dc11-
               dcn11_router.dc"/>
       </zone>
       <zone id="zone1-dc2" routing="Floyd">
          <clusterid="dcn12"prefix="dc12-"radical="0-
               10"suffix=".dc"speed="20Gf"core="4"bw="1000MBps"lat="10us"router_id="dc12-
               dcn12_router.dc"/>
       </zone>
       <zone id="zone2-dc1" routing="Floyd">
          <clusterid="dcn21"prefix="dc21-"radical="0-
               10"suffix=".dc"speed="20Gf"core="4"bw="1000MBps"lat="10us"router_id="dc21-
               dcn21_router.dc"/>
       </zone>
</zone>
<!—Линиисвязи --Ю
<link id="link1" bandwidth="30MBps" latency="10ms"/>
…
<link id="link12" bandwidth="1000MBps" latency="10ms"/>
<!—Маршруты между зонами →
<zoneRoutesrc="zone1-dc1"dst="zone1-ubase1"gw_src="dc11-dcn11_router.dc"gw_dst="ub11-
     ubase11_router.uz">
     <link_ctnid="link1"/>
</zoneRoute>
…
<zoneRoutesrc="zone1-dc2"dst="zone2-dc1"gw_src="dc12-dcn12_router.dc"gw_dst="dc21-
      dcn21_router.dc">
      <link_ctnid="link12"/>
</zoneRoute>
</zone>
```

Описание линий связи и маршрутов между зонами опустим. Это как раз наиболее трудоемкая и объемная часть файла платформы, нужно внимательно описать все маршруты между всеми зонами. В данной модели пропускная способность линий связи между ЦОД 1Гб/с, линии связи между ЦОД dc11 и базой пользователей ub11(в той же зоне) – 30Мб/с, между dc11 иub21 – 20Мб/с, база пользователей ub31 соединена со всеми ЦОД линией с пропускной способностью 10 Мб/с.

В каждом ЦОД только кластеры физических машин. Первый экземпляр брокера будет запущен на физической машине в первом ЦОД, актор будет выполнять функцию cloudStart с параметрами, задающими начальное количество серверов каждого типа, количество пользовательских циклов (количество запросов во время одной сессии пользователя), размеры запроса и ответа в байтах, количество операций для имитации вычислений при работе серверов, а также список ЦОД. Остальные серверы (в том числе дополнительные экземпляры брокера) будут созданы позже, во время работы модели. Файл развертывания:

```
      <platformversion="4.1">
```

```
<actorhost="dc11-0.dc"function="cloudStart">
  <argumentvalue="1"/><!--cb   -->
  <argumentvalue="1"/><!--  cs   -->
  <argumentvalue="1"/><!--  load balancers -->
  <argumentvalue="1"/><!--  data bases -->
  <argumentvalue="1"/><!--  app -->
  <argumentvalue="1"/><!--  web   -->
  <argumentvalue="2"/><!--  number of user cycles  -->
  <argumentvalue="1000"/><!--request_size  -->
  <argumentvalue="10000"/><!--response_size  -->
  <argumentvalue="100000000"/><!--calculation_cost  -->
  <argumentvalue="dc11 dc12 dc21"/><!--  dcnames  -->
</actor>
</platform>
```

В функции main() загружаются файл платформы и файл развертывания, и на физической машине dc11-0.dc (первая машина кластера ЦОД dc11) будет запущен актор, выполняющий функцию CloudStart. Также при старте моделирования запускаются активности пользователя на каждой физической машине из зоны пользователей. Эта активность имитирует работу с приложением – пользователь своим действием, например, нажатием кнопки, запускает запрос к веб-серверу, после получения ответа через случайный интервал (в пределах 5–10 с) снова запускает запрос столько раз, сколько указано в файле развертывания (циклы пользователя). Новые запросы будут направляться сразу на веб-сервер, выбранный при выполнении начального запроса.

Функция CloudStart в соответствии со списком аргументов из файла развертывания создает все ЦОД, затем в каждом из них создаются унифицированные базовые серверы (серверы нулевого уровня) [11]. Назначение сервера нулевого уровня – ожидать запроса, при получении его запустить виртуальную машину (сервер первого уровня) для обслуживания запроса на физической машине, указанной балансировщиком нагрузки.

Правила именования. Одной из основных трудностей при моделировании является организация взаимодействия серверов, которая моделируется через почтовые ящики (MailBox). В рассматриваемой модели каждый физический или виртуальный хост имеет почтовый ящик для чтения, его имя совпадает с именем машины – для физических машин, но чаще требуется взаимодействие с виртуальными машинами. Ранее в[11] были введены правила именования серверов (виртуальных машин) для модели с одним ЦОД, теперь их нужно немного изменить, добавим префикс с именем ЦОД перед префиксом типа. Имена серверов нулевого уровня (базовых) будут строиться из префикса ЦОД, префикса типа и порядкового номера в группе серверов данного типа., например "dc11_web_0", "dc11_web_1", "dc12_app_0","dc21_db_0","dc12_lb_0". При создании сервера первого уровня к префиксам добавляется имя клиента – dc12_web_user0 и так далее. По таким правилам имя сервера будет уникальным на уровне всей платформы.

Обработка запроса пользователя

Пользователи подключаются к облаку через интернет, есть несколько «точек входа», адреса которых или заранее известны, или могут быть получены через запрос к DNS серверу. Скорее всего, пользователь получит доступ к веб-сайту, на котором выберет нужный сервис, сгенерировав запрос. Для удовлетворения запроса в ЦОД создается соответствующая инфраструктура, это может быть один или несколько виртуальных серверов. В рассматриваемой модели пользователь подключается к экземпляру брокера, выбираемому по очереди (алгоритм RoundRobin, или карусельный алгоритм, он применяется в DNS серверах), и далее уже брокер преренаправляет запрос в выбранный ЦОД. Это балансировка нагрузки между ЦОД на уровне облака.

Далее запрос поступает к контроллеру облака, который обращается к балансировщику нагрузки для выбора экземпляра сервера обработки и перенаправляет запрос на выбранный сервер. Для обработки запроса может понадобиться больше серверов (например, трехуровневое приложение в [11]), тогда нужно обращаться к балансировщику каждый раз перед обращением к новому серверу. Это обеспечит балансировку нагрузки внутри ЦОД.

Последовательность обработки запроса пользователя для трехуровневого веб-приложения показана на рисунке 2.Веб-серверgw-web является точкой входа в облако, начальным сайтом, с которого выполняется первый запрос, в рассматриваемой модели точкой входа является сервисный брокер.

Алгоритмы работы унифицированных серверов для сервисного брокера и балансировщика нагрузки похожи и включают следующие шаги:

1. Получение запроса от клиента, получение адреса клиента (имя почтового ящика).
2. Обращение к алгоритму выбора, получение имени ЦОД, физической машины или виртуального сервера.
3. Передача результата клиенту.

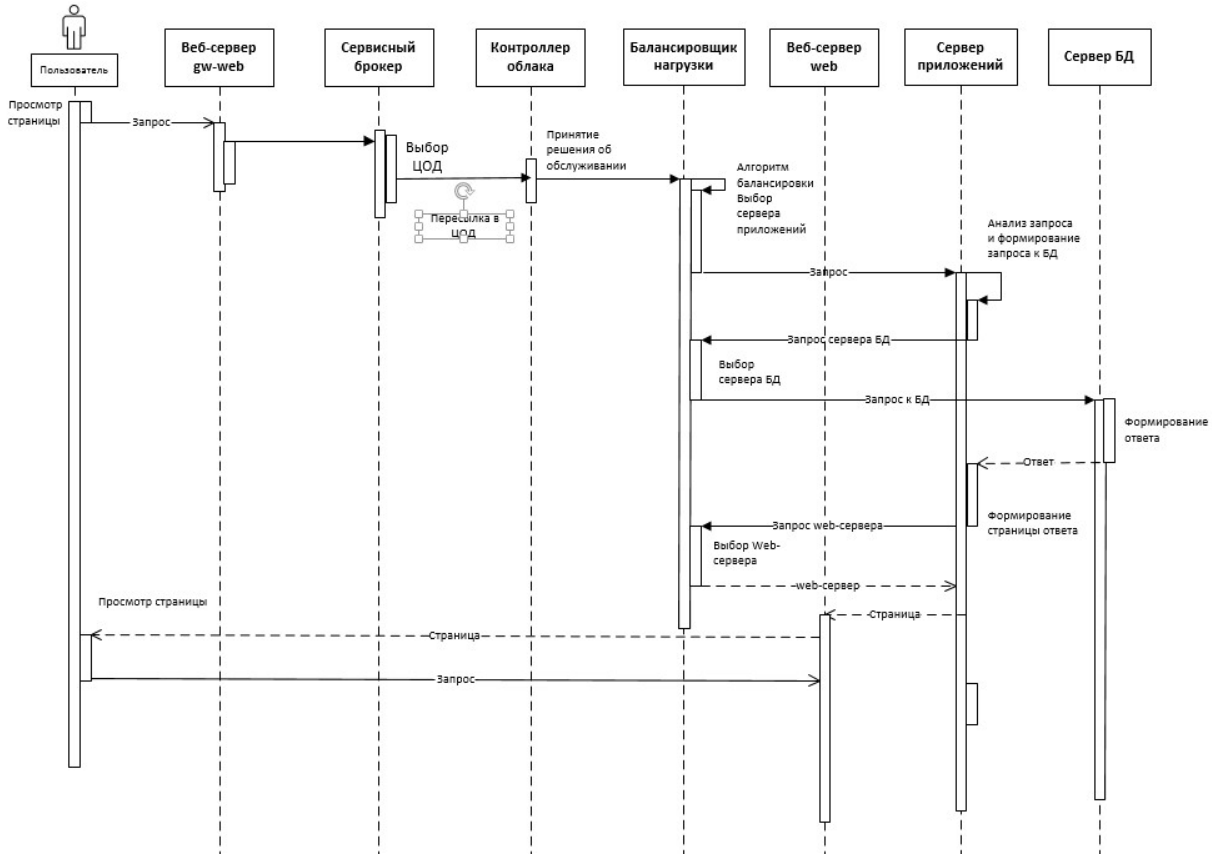

Рисунок 2 – Диаграмма последовательности обработки запроса пользователя для трехуровневого веб-приложения

Таким образом, для реализации разных политик выбора ЦОД и алгоритмов балансировки отдельно создается код, который будет вызываться в зависимости от заданного алгоритма, при этом код сервера остается неизменным.

Проведение эксперимента и анализ результатов моделирования

Целью эксперимента является проверка работоспособности модели облака с несколькими ЦОД. Функция main(), функции активностей, вспомогательные функции, разрабатывались на языке C++.Для отладки использовались средства журналирования SimGrid – APIXBT_LOG [6, 8].Отладочные сообщения позволяют проследить последовательность обработки запроса и особенно важны на этапе отладки взаимодействия серверов, когда их количество сравнительно невелико. Так как проверяется работоспособность самой модели, для выбора ЦОД используется алгоритм RoundRobin – из очереди выбирается первый ЦОД, выбранный ЦОД перемещается в конец очереди). В качестве политики выбора ЦОД этот алгоритм не имеет смысла и может отрицательно повлиять на производительность системы, но он может оказаться полезным для отладки.

Проследим последовательность выполнения запроса от пользовательской машины ub31-0.uz из зоны (в ней нет ЦОД):

```
[ub31-0.uz:ub31-0.uz:(4) 80.000000] [s4u_app_cl/INFO] UHUMAN ub31-0.uz sending  http_get 'ub31-0.uz web_' to CB using mailbox 'dc11_cb_0'
[dc11-0.dc:dc11_cb_0:(5) 80.130478] [s4u_app_cl/INFO] SERVER id = dc11_cb_0 read 'ub31-0.uz web_' from mailbox /dc11_cb_0/
```

Запрос был отправлен на базовый брокер dc11_cb_0, который находится в ЦОД dc11и получен.'

```
[dc11-0.dc:dc11_cb_ub31-0.uz:(23) 80.130742] [s4u_app_cl/INFO] VM CB message 'ub31-0.uz web_' sent to dc11_cs_0
```

Виртуальный сервер dc11_cb_ub31-0.uz пересылает запрос контроллеру облака dc11_cs_0 в том же ЦОД.

```
[ub31-0.uz:ub31-0.uz:(4) 80.261221] [s4u_app_cl/INFO] UHUMAN ub31-0.uz received'dc11_web_0' from CS
```

Пользователь получил от контроллера облака адрес базового сервера, который будет обслуживать его запрос – dc11_web_0 и к которому уже напрямую будут направляться другие запросы в течение сессии. Зная имя ЦОД, пользователь формирует имя виртуального сервера – это dc11_web_ ub31-0.uz

```
[ub31-0.uz:ub31-0.uz:(4) 80.261221] [s4u_app_cl/INFO] UHUMAN 'ub31-0.uz' waiting for response from server 'dc11_web_ub31-0.uz'
```

Пользователь ждет ответа от сервера, который будет его обслуживать.

```
[dc11-1.dc:dc11_cs_ub31-0.uz:(26) 80.261469] [s4u_app_cl/INFO] VM CS message 'ub31-0.uz web_' sent to dc11_web_0
```

Контроллер облака переслал запрос базовому веб-серверу.

```
[dc11-10.dc:dc11_web_ub31-0.uz:(29) 80.261469] [s4u_app_cl/INFO] VM WS id = dc11_web_ub31-0.uz, mailbox is ub31-0.uz sending data is response 1
```

Веб-сервер высылает ответ на первый запрос.

```
[ub31-0.uz:ub31-0.uz:(4) 80.392875] [s4u_app_cl/INFO] UHUMAN 'ub31-0.uz'  received data 'response 0' from server 'dc11_web_ub31-0.uz'
```

Пользователь получил ответ на первый запрос. На это было затрачено 0.39 с.

Второй запрос – 0.2619 с, меньше времени за счет того, что идет прямое обращение к веб-серверу.

```
[ub31-0.uz:ub31-0.uz:(4) 93.392875] [s4u_app_cl/INFO] UHUMAN 'ub31-0.uz' sending data 'request  1' to server 'dc11_web_ub31-0.uz'
[ub31-0.uz:ub31-0.uz:(4) 93.654775] [s4u_app_cl/INFO] UHUMAN 'ub31-0.uz'  received data 'response 1' from server 'dc11_web_ub31-0.uz'
```

Запросы с других машин также можно проследить, они полностью выполнены.

Модель успешно отрабатывает до конца, обслуживает все запросы и для большего количества пользователей, 100, 1000 в одной базе, и для разного начального количества серверов, что говорит о ее работоспособности.

В таблице 1 представлено время отклика для случая, когда в базе пользователей по 1 пользователю и все типы серверов представлены 1 экземпляром, пользователь генерирует 10 запросов за сеанс. Время обслуживания для запросов, следующих за первым, не меняется, так как эти запросы выполняются от начала до конца, не ждут обслуживания, другие запросы не влияют на их обработку. Политика выбора ЦОД.

Таблица 1– Время отклика, в базе пользователей 1 пользователь

ЦОД	Первый запрос	Мин	Среднее	Max
DC11	0.262508	0.131391	0.131391	0.131391
DC12	0.262508	0.131391	0.131391	0.131391
DC21	0.262559	0.131391	0.131391	0.131391

В таблице 2 представлено время отклика для 10 пользователей в каждой из баз. Среднее время для первого запроса одинаковое для всех серверов, минимальное время обслуживания совпадает со временем обслуживания для одной машины, это действительно минимальное время.

Таблица 2 – Время отклика для 10 пользователей в базе

ЦОД	Первый запрос, ср.	Мин	Среднее	Max
DC11	0.262486	0.131391	0.131405	0.131443
DC12	0.262486	0.131391	0.131407	0.131443
DC21	0.262486	0.131391	0.131410	0.131443

В таблице 3 представлены минимальное, среднее и максимальное время обработки первого и последующих запросов в ЦОД,

Таблица 3–Время отклика для 10 пользователей в базе, политика ближайшего сервера

ЦОД	Первый запрос			Остальные запросы сессии		
	Мин	Среднее	Max	Мин	Среднее	Max
DC11	0.262428	0.262431	0.262433	0.131443	0.13147	0.131551
DC12	0.262428	0.262725	0.263176	0.131391	0.131405	0.131443
DC21	0.262376	0.262658	0.263175	0.13139	0.131399	0.131443
	0.262376	0.262661	0.263176	0.13139	0.131409	0.131551

Минимальное время немного уменьшилось и для первого, и для остальных запросов за счет того, что некоторые запросы стали обслуживаться ближайшим сервером, но в системе присутствует база пользователей из зоны, в которой нет ЦОД, и запросы из этой зоны обслуживаются по алгоритму RoundRobin. Такой подход не идеален, так как в случае, когда в зоне пользователя нет ЦОД, запросы будут направляться в ЦОД другой зоны, но возможна ситуация, когда ЦОД занят обслуживанием «чужих» запросов, и запросы от «близких» пользователей перенаправляются в другой ЦОД.

Заключение

Применение унифицированного сервера [11] и предлагаемая структура платформы облегчают создание различных моделей с разным количеством ЦОД, серверов разных типов и баз пользователей. Для создания нового типа сервера требуется добавить только его блок обработки, а при подключении новой политики сервисного брокера или алгоритма балансировки добавляется код соответствующего алгоритма. Представленный подход позволяет создать графический интерфейс для задания параметров модели и автоматической генерации файла платформы и файла развертывания подобно тому, как это сделано в CloudAnalyst, но внутренняя структура ЦОД и характеристики приложения могут быть описаны более подробно. Это позволит тестировать производительность для конкретного приложения в облаке.

Литература

1. Z. Benlalia, A. Beni-hssane, K. Abouelmehdi and A. Ezati, "A new service broker algorithm optimizing the cost and response time for cloud computing", Procedia Computer Science, vol. 151, pp. 992-997, 2019
2. Divya Raj K , Nijeesha G, Santhosh B, 2019, Service Broker Algorithms in Cloud Computing, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) RTESIT – 2019 (VOLUME 7 – ISSUE 08)
3. B. Wickremasinghe, “CloudAnalyst: A CloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments” 433-659 DISTRIBUTED COMPUTING PROJECT, CSSE DEPT., UNIVERSITY OF MELBOURNE.
4. Г.Р. Гарай, А. Черных, А.Ю. Дроздов. Сравнительный анализ методов оценки производительности многоуровневых облачных приложений. Труды ИСП РАН. –Т.27. –Вып. 6, 2015 г. – С. 199–224.
5. Gupta, Kiran &Beri, Rydhm. (2016). Cloud Computing: A Survey on Cloud Simulation Tools. 2.
6. SimGrid: Versatile Simulation of Distributed Systems/ URL:https://simgrid.org/ (дата обращения 25.06.2024)
7. Гайдамако В.В. Моделирование облачной информационно-измерительной системы с помощью библиотеки Simgrid // Проблемы автоматики и управления. – Бишкек:Илим, №1 (36), 2019. – С.90–99.
8. SimGridTutorials. URL: https://simgrid.org/doc/latest/intro_concepts.html (дата обращения 25.06.2024)
9. Santhosh B ,Dr D.H Manjaiah “Resource Location and Cost basedPerformance Optimized Routing algorithm in cloud computing”// Second International Conference on Emerging Research in Computing,Information, Communication and Applications, August 2014
10. Mohammed Radi, “Weighted Round Robin Policy for Service Brokers in a Cloud Environment”, International Arab Conference on Information Technology (ACIT2014).]
11. Гайдамако В.В. Моделирование сервера в инфраструктуре облачного центра обработки данных на базе платформы Simgrid// Проблемы автоматики и управления. –Бишкек: Илим, №3 (48), 2023.