Simcan2Cloud: симулятор на основе дискретных событий для моделирования и моделирования инфраструктур облачных вычислений.

https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w

* [Пабло К. Каньисарес](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#auth-Pablo_C_-Ca_izares-Aff1) ,
* [Альберто Нуньес](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#auth-Alberto-N__ez-Aff2) ,
* [Адриан Бернал](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#auth-Adri_n-Bernal-Aff3) ,
* [М. Эмилия Камбронеро](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#auth-M__Emilia-Cambronero-Aff3) и amp;
* [Адам Баркер](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#auth-Adam-Barker-Aff4)

[*Журнал облачных вычислений*](https://journalofcloudcomputing.springeropen.com/) **объем 12** , Номер статьи:  133 ( 2023 ) [Цитировать эту статью](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#citeas)

* **1065**доступов
* **1**Цитаты
* [Метрикиподробности](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/metrics)

Абстрактный

Облачные вычисления — это развивающаяся парадигма, распространение которой за последние несколько лет возросло. Этот факт привел к росту рынка облачных вычислений вместе с жесткой конкуренцией за ведущую долю рынка, а также к увеличению количества поставщиков облачных услуг. Постоянно предлагаются новые методы повышения прибыльности поставщиков облачных услуг. Однако для производственных облаков рассматриваются только те методы, которые, как доказано, не препятствуют заключению соглашений об обслуживании. Анализ ожидаемого поведения и производительности облачной инфраструктуры является сложной задачей, поскольку повторяемость и воспроизводимость экспериментов на этих системах затрудняются из-за большого количества пользователей, одновременно имеющих доступ к инфраструктуре. К этому следует добавить сложности, связанные с использованием разных политик предоставления ресурсов, управлением несколькими рабочими нагрузками и применением разных конфигураций ресурсов. Поэтому, чтобы решить эти проблемы, мы представляем Simcan2Cloud, симулятор на основе дискретных событий для моделирования и моделирования сред облачных вычислений. Simcan2Cloud фокусируется на моделировании и моделировании поведения поставщика облачных услуг с высоким уровнем детализации, при этом в моделируемые сценарии интегрируются как облачная инфраструктура, так и взаимодействие пользователей с облаком. Для этой цели Simcan2Cloud поддерживает различные политики распределения ресурсов, соглашения об уровне обслуживания (SLA), а также интуитивно понятный и полный API для включения новых политик управления. Наконец, представлено тщательное экспериментальное исследование для оценки пригодности и применимости Simcan2Cloud с использованием как реальных трассировок, так и синтетических рабочих нагрузок.

Введение

За последние несколько лет облачные вычисления стали эталоном вычислений по требованию. Высокий уровень гибкости, безопасности и экономии средств побудили компании использовать эту вычислительную парадигму для предоставления необходимых им услуг. Согласно отчету Right-Scale 2019 State of the Cloud Report от [ [1](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR1) ], 94% предприятий используют хотя бы один облачный сервис, а расходы на такие сервисы достигли $227,8 млрд. Чтобы удовлетворить этот спрос, существует несколько поставщиков облачных услуг, таких как Amazon Web Services (AWS), Azure, Google Cloud, VMWare Cloud и Oracle Cloud Infrastructure и другие.

Рыночная конкуренция заставила поставщиков услуг искать элементы дифференциации, такие как производительность, качество обслуживания и стоимость. Таким образом, одной из основных целей облачных провайдеров является достижение хорошего баланса между производительностью системы и использованием вычислительных ресурсов при сохранении прибыли. Однако достижение сбалансированной архитектуры, которая достигает этой цели, является сложной задачей. Учитывая развивающуюся компанию, предоставляющую облачные услуги, значительный рост числа пользователей, получающих доступ к услугам, может привести к возникновению узких мест в системе, что может привести к падению прибыли из-за потери пользователей. Основная идея состоит в том, чтобы предоставить центру обработки данных адекватные вычислительные ресурсы для обслуживания новых пользователей, избегая чрезмерного или недостаточного измерения размеров системы.

Чтобы получить хорошее соотношение цены и качества, необходимо провести тщательный анализ облака при обработке различных рабочих нагрузок, что позволяет провайдеру правильно настроить различные параметры облака, такие как виртуальные машины (ВМ), политики распределения ресурсов, и стоимость каждой предлагаемой виртуальной машины [ [2](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR2) ]. Неправильно настроенная облачная среда может привести к снижению общей производительности, что окажет существенное влияние на качество обслуживания и, как следствие, поставит под угрозу репутацию компании.

К сожалению, проведение экспериментального анализа в готовых к использованию средах является сложным, дорогостоящим и в некоторых случаях невозможным из-за необходимости иметь выделенный доступ к системе. Более того, на поведение системы могут повлиять изменения конфигурации производственной системы, такие как добавление дополнительных машин, замена вычислительных ресурсов или установка новой топологии сети.

За последние десять лет исследователи решили эти проблемы, используя методы моделирования [ [3](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR3) , [4](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR4) , [5](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR5) , [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR6) ]. Основные особенности этих методов позволяют создавать инструменты моделирования, подходящие для моделирования, анализа и изучения сложных систем. По сути, симулятор использует абстракцию изучаемой системы, а именно модель, для имитации ее поведения путем представления ее наиболее важных функций. Среди наиболее важных преимуществ, обеспечиваемых моделированием, можно выделить следующие, связанные с облачными системами: (i) исследуемая система не требуется для выполнения моделирования. В целом симуляторы можно запускать на обычном компьютере; (ii) Эксперименты можно легко воспроизвести в моделируемой среде. В большинстве случаев существует большое количество взаимосвязанных параметров и переменных, которые невозможно контролировать в реальной производственной системе, например, одновременный доступ пользователей к системе, что делает невозможным повторяемость экспериментов [ [7](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR7) ]. Однако моделирование позволяет нам воспроизвести тот же эксперимент контролируемым образом; (iii) Эксперименты можно проводить параллельно, повышая производительность без необходимости использования специальных аппаратных ресурсов [ [8](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR8) , [9](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR9) ]. Таким образом, моделирование можно запускать на стандартном рабочем столе – используя доступные ядра ЦП – или, чтобы значительно увеличить количество параллельно выполняемых симуляций, на компьютерном кластере; и (iv) моделирование обеспечивает большую гибкость при внесении изменений в настройки конфигурации. Хотя изменение конфигурации облачной системы является трудоемкой и дорогостоящей задачей, моделирование требует от нас лишь изменения конфигурации модели путем настройки правильных параметров, таких как топология сети или политика распределения ресурсов.

В настоящее время существует широкий спектр платформ моделирования для моделирования систем облачных вычислений. Однако большинство облачных симуляторов ориентированы на представление поведения системы с точки зрения пользователей и не учитывают роль поставщика облачных услуг. Например, DISSECT-CF [ [10](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR10) ] считается одним из наиболее актуальных симуляторов облачных вычислений. Однако различные аспекты, связанные с поставщиком облачных услуг, такие как политики распределения, управление пользователями и затраты, не принимаются во внимание. Кроме того, существует несколько предложений, ориентированных на различные аспекты облачных провайдеров, такие как особенности ценообразования [ [11](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR11) , [12](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR12) , [13](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR13) ], развертывание облаков [ [14](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR14) ], ресурсы моделирования [ [15](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR15) ] и услуги, предлагаемые облачным провайдером [ [16](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR16) ]. Тем не менее, эти работы не направлены на рассмотрение аппаратного обеспечения облачной платформы.

Насколько нам известно, существует несколько платформ моделирования, предназначенных для описания поставщика облачных услуг с разумным уровнем детализации с учетом поддержки инфраструктуры. В этом плане CloudSim [ [8](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR8) ] предлагает несколько политик управления доступными облачными ресурсами, поддерживая различные стратегии выбора хоста, развертывания сервисов и предоставления виртуальных машин. Однако ресурсы облачной инфраструктуры, а также управление и поведение пользователей особо не детализированы. Чтобы преодолеть эти проблемы, мы представляем Simcan2Cloud, основанную на дискретных событиях среду для моделирования облачных систем. Simcan2Cloud в основном ориентирован на поставщика облачных услуг, поддерживая моделирование облачных инфраструктур и взаимодействие пользователей с облаком. Кроме того, для анализа того, как Simcan2Cloud соответствует реальному миру, платформа включает в себя модуль представления трассировки, который позволяет выполнять реальные трассировки, собранные из готовых к эксплуатации систем. Таким образом, мы можем сравнить смоделированную систему с реальной целевой системой, чтобы обнаружить потенциальные несоответствия. Ниже мы выделяем наиболее актуальные и новые особенности предлагаемой нами платформы моделирования:

1. 1.

Гибкие соглашения об уровне обслуживания. Simcan2Cloud учитывает различные определения SLA в средах облачных вычислений. Следовательно, запрошенные ресурсы распределяются между пользователями в соответствии с различными параметрами, установленными в соглашении об уровне обслуживания: доступность ресурсов, время аренды и настраиваемая модель стоимости, которая охватывает несколько аспектов, таких как скидки за задержки, дополнительные затраты на дополнительное время. и компенсация за недоступность.

1. 2.

Очередь ожидания облачного провайдера. Что касается управления пользователями, платформа предоставляет систему очередей для обработки пользователей по их прибытии в облако. Этот механизм позволяет пользователям дождаться запрошенной услуги, подписавшись на систему, вместо того, чтобы немедленно покидать систему.

1. 3.

Приоритетные пользователи. Чтобы улучшить поведение системы, платформа поддерживает управление пользователями с разными уровнями приоритета. Следовательно, пользователям с высоким приоритетом не требуется ждать в очереди облачного провайдера, поскольку их запрошенные ресурсы распределяются на зарезервированных машинах, которые предназначены исключительно для этих пользователей.

1. 4.

Предложение о продлении срока аренды. Что касается аренды услуг, платформа поддерживает продление времени аренды виртуальных машин, когда некоторые службы все еще работают, а время аренды запрошенных ресурсов истекло. Эта функция предназначена для описания типичного поведения в облачных средах.

1. 5.

Использование ресурса. Эта платформа включает в себя модуль для мониторинга использования вычислительных ресурсов в центре обработки данных, таких как процессоры, оперативная память и хранилище. Эта функция позволяет пользователям анализировать модели использования и выявлять деструктивное поведение в этих ключевых подсистемах.

1. 6.

API для легкого включения новых политик управления. Simcan2Cloud поддерживает различные политики планирования распределения ресурсов. Поставщик облачных услуг может выбрать наиболее подходящие алгоритмы для максимизации как процента использования ресурсов, так и прибыли поставщика облачных услуг. Кроме того, платформа предоставляет шаблоны для облегчения создания пользовательских политик планирования и поведения пользователей.

Эта статья организована следующим образом. Во-первых, раздел « [Сопутствующие работы](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec2) » знакомит и анализирует современное состояние симуляторов облачных вычислений. В разделе « [Simcan2Cloud](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec5) » представлена ​​архитектура и детали реализации Simcan2Cloud. [Затем в разделе « Эмпирическое исследование](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec11) » мы представляем эмпирическое исследование , в котором анализируется и обсуждается производительность Simcan2Cloud. Наконец, раздел « [Выводы и дальнейшая работа](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec14) » содержит наши выводы и некоторые направления будущей работы.

Связанных с работой

За последние несколько десятилетий методы моделирования были приняты исследовательским сообществом как ценный способ изучения и анализа сред облачных вычислений. В результате в литературе появилось значительное количество симуляторов облачных вычислений [ [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR6) , [17](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR17) , [18](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR18) , [19](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR19) , [20](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR20) , [21](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR21) ]. Заметный рост числа современных опросов – в среднем с 10 в 2012 году [ [21](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR21) ] до 30 в 2020 году [ [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR6) ] – является явным индикатором возрастающего интереса к разработке облачных симуляторов.

Симуляторы облачных вычислений

В текущей литературе мы обнаружили несколько платформ моделирования, ориентированных на поставщика облачных услуг. Симулятор CloudNetSim++ [ [22](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR22) ] — это облачный симулятор, построенный на базе OMNeT++, который использует INET Framework для моделирования всего сетевого уровня. Этот симулятор позволяет пользователям описывать политики SLA, алгоритмы планирования и затраты на выставление счетов, а также предлагает встроенный пользовательский интерфейс OMNeT++. Таким образом, пользователи должны изучить основы среды OMNeT++ для создания облачных сценариев. Еще одно предложение — Data Center Simulator (DCSim), среда моделирования для моделирования и симуляции инфраструктур центров обработки данных [ [23](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR23) ]. В общих чертах, DCSim фокусируется на уровне IaaS, который используется для предоставления услуг множеству клиентов. Также стоит отметить, что DCSim поддерживает моделирование затрат и соглашений об уровне обслуживания. DISSECT-CF — это платформа моделирования, ориентированная на моделирование совместного использования ресурсов и облачной инфраструктуры с высоким уровнем детализации [ [10](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR10) ]. Этот подход представляет собой довольно детальное моделирование стека IaaS и поддерживает методы энергосбережения для облачных инфраструктур, что позволяет включать новые показатели для анализа различных ресурсов. SCORE [ [24](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR24) ] — симулятор, основанный на Google Omega и написанный на Scala. SCORE моделирует параллельное планирование, потребление энергии и синтетические рабочие нагрузки, а также предлагает механизмы выключения и включения вычислительных узлов. Аналогичным образом, SCORE-GAME является расширением SCORE, которое включает в себя политику планирования энергопотребления, основанную на игре Штакельберга [ [25](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR25) ]. Модель этого симулятора включает в себя две роли: *менеджера по планированию* и *менеджера по энергоэффективности* . Первый обрабатывает задачи максимально быстро, а второй нацелен на минимизацию общего энергопотребления. Таким образом, это предложение основано на конкуренции между этими ролями, где главная цель — найти баланс между потреблением энергии и производительностью. iCanCloud — это платформа моделирования, построенная на платформе OMNeT++ [ [26](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR26) ]. По сути, этот симулятор представляет поведение облачных систем путем моделирования физических машин, поддерживающих облако, конфигурации предоставляемых виртуальных машин и различных политик распределения ресурсов. Кроме того, E-MC2Фреймворк [ [9](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR9) ] был разработан для включения поддержки измерения энергопотребления различных аппаратных компонентов системы, таких как память, центральные процессоры дисковых накопителей и т. д. Таким образом, iCanCloud можно использовать для оценки компромиссов между стоимость и производительность в широком диапазоне облачных сценариев.

Обзоры инструментов облачного моделирования, найденные в текущей литературе, утверждают, что CloudSim [ [8](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR8) ] является одним из ведущих облачных симуляторов [ [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR6) ]. CloudSim использует SimJava в качестве ядра моделирования и позволяет моделировать хосты в центрах обработки данных, виртуальные машины, пользовательские задачи и политики предоставления ресурсов. CloudSim фокусируется на алгоритмах планирования сервисного брокера и реализует политики распределения с разделением пространства и времени для управления вычислительными ресурсами. Этот симулятор предоставляет ограниченную модель сети, поскольку он учитывает только задержки передачи и не имеет реалистичной топологии сети. Поскольку несколько исследователей обнаружили ограничения при проведении экспериментов с CloudSim в изучаемых областях, исследовательское сообщество расширило возможности инструмента, внедрив другие симуляторы на базе CloudSim. Среди таких симуляторов, расширяющих функциональные возможности CloudSim, можно выделить NetworkCloudSim, CloudAnalyst, CDOSim, WokflowSim, CloudExp и UCloud.

NetworkCloudSim [ [27](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR27) ] улучшает сетевой уровень CloudSim, реализуя коммутаторы на нескольких уровнях агрегации и предоставляя модели связи с разными уровнями детализации. Эти реализации позволяют разработчикам моделировать параллельные приложения. CloudAnalyst [ [28](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR28) ] предоставляет графический интерфейс для CloudSim, отображающий географические факторы, которые позволяют настраивать расположение пользователей и центров обработки данных. По сути, функция определения местоположения позволяет симулятору рассчитывать время ответа и обработки запросов. CDOSim [ [29](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR29) ] моделирует стоимость и характеристики производительности сценариев развертывания облака и позволяет разработчикам моделировать задержки и нарушения SLA, помогая им выбрать стратегию развертывания. Хотя этот симулятор реализует миграцию виртуальных машин, он по-прежнему наследует ограниченную сетевую модель от CloudSim.

WokflowSim [ [30](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR30) ] представляет моделирование научных рабочих процессов в облачной среде и кластеризацию заданий, что позволяет исследователям изучать влияние сбоев в работе на рабочие процессы. Этот симулятор не подходит для приложений с интенсивным использованием данных, поскольку не моделирует производительность системы хранения. Симулятор CloudExp [ [31](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR31) ] улучшает CloudSim, включая сложные сетевые модели и модель обработки Map-Reduce. CloudExp предлагает определение SLA на основе измеримых показателей, а также поддерживает набор инструментов генератора рабочих нагрузок для моделирования реальных рабочих нагрузок. Одним из главных недостатков CloudExp является статическая модель измерения производительности виртуальных машин. UCloud [ [32](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR32) ] — это симулятор гибридного облака для университетских сред, ориентированный на сценарии, требующие услуг публичных облаков, когда частное облако заполнено. Кроме того, UCloud реализует мониторинг производительности, университетской деятельности и управление безопасностью, а также учитывает стоимость использования публичного облака, а не стоимость связи в дата-центре.

Сравнение решений Simcan2Cloud и SoTA

В этом разделе мы представляем всестороннее сравнение Simcan2Cloud и некоторых известных облачных симуляторов. Важно подчеркнуть усилия и время, вложенные исследовательским сообществом в создание и поддержку широкого спектра инструментов моделирования, и этот факт привел к существованию широкого спектра облачных симуляторов. Чтобы выбрать те симуляторы, которые получили широкое распространение в сообществе, мы тщательно проанализировали статьи, доступные в текущей литературе, опросы, такие как [ [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR6) ] и [ [17](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR17) ], и общедоступные репозитории.

**Таблица 1. Сравнение основных симуляторов облачных вычислений, существующих в литературе**

[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/1)

В таблице  [1](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab1) анализируются основные различия между существующими облачными симуляторами и подходом, представленным в этой работе, а именно Simcan2Cloud, подчеркивая основной вклад нашего предложения. Таблица состоит из пяти разделов, объединяющих несколько аспектов платформ облачного моделирования. В первом разделе (с надписью « *Основные функции»* ) представлена ​​основная информация о каждой платформе, такая как название инструмента, дата создания ( *Год* ), язык программирования ( *Язык* ), доступность инструмента ( *Доступно* ) и платформа, используемая в качестве основа для создания симулятора ( *Платформы* ). Второй блок, помеченный как *Общие аспекты* , показывает общие особенности симуляционной платформы, то есть возможность использования графического пользовательского интерфейса ( *GUI* ), уровень детализации для представления сети связи ( *модель связи* ), а также возможности платформа для проектирования топологии сети ( *Network topology* ). Третий блок, называемый *CloudProvider* , показывает возможности моделирования поведения облачного провайдера. В этом случае мы анализируем соглашение об уровне обслуживания ( *SLA* ), модель стоимости ( *Cost* ) и предложение о продлении времени аренды ( *Rent ext.* ), которое относится к тем сценариям, в которых некоторые услуги все еще работают, когда время аренды запрошенные ресурсы заканчиваются. В данном конкретном случае пользователи имеют возможность продлить время аренды ресурсов, заплатив дополнительную плату сверх первоначальной стоимости. Кроме того, этот блок охватывает политики планирования ( *Sched. policy* ) и управление очередями пользователей ( *Очередь ожидания* ), что относится к механизму, позволяющему пользователям ждать, если запрошенные ресурсы недоступны. Четвертый блок ( *Пользователи/рабочая нагрузка* ) представляет функции моделирования облачной среды с точки зрения пользователя, который обращается к облаку для запроса услуг. В данном случае мы рассматриваем возможности управления новыми рабочими нагрузками ( *API* ), поддержку представления выполнения трассировок, извлеченных из реальных облаков ( *Real Traces* ), возможность использования различных статистических распределений для создания рабочих нагрузок ( *Traffic Dist.* ). Эта функция представляет ресурсы, запрошенные пользователями, путем указания точного времени, когда каждый пользователь отправляет запрос, и конкретной конфигурации необходимых ресурсов. Последняя функция обозначает поддержку включения приоритетных пользователей на платформу моделирования ( *Prior.users* ). Последний блок с надписью *DC*, ориентирован на аспекты центра обработки данных, которые включают процент оборудования, используемого в ЦОД ( *использование аппаратного обеспечения* ), и поддержку моделирования – с высоким уровнем детализации – различных аппаратных компонентов платформы ( *детализация аппаратного обеспечения* ).

Это сравнение охватывает 15 платформ моделирования, включая Simcan2Cloud. Годы создания платформ варьируются от 2009 до 2022 года. Что касается языка программирования, то Java является преобладающим, поскольку он использовался для написания кода 60% симуляторов, изученных в этом сравнении. Напротив, C++ и Scala использовались для написания 26% и 13% инструментов моделирования соответственно. Все инструменты моделирования, за исключением CloudExp, имеют лицензию с открытым исходным кодом. Большинство платформ моделирования построены на базовой платформе, и только DCSim и DISSECT-CF созданы как независимые инструменты. Среди них CloudSim является предпочтительной платформой для создания новых симуляторов, которую используют 40% из них, за ней следует OMNeT++, которую используют 20% инструментов моделирования. Остальные решения основаны на Google Omega, NS2 и SimJava. Анализируя общие аспекты платформ, мы видим, что только 40% предоставляют полный графический интерфейс, с помощью которого можно полностью моделировать и настраивать облачный сценарий, в то время как 13% предлагают ограниченный графический интерфейс, который облегчает настройку моделируемых сценариев со значительными ограничениями. Что касается связи, только 20% реализуют полную модель связи (т.е. протоколы связи, такие как TCP и UDP), а 60% предоставляют ограниченную модель. Наконец, полная поддержка проектирования сетевых топологий включена лишь в 20% предложений.

Что касается деталей облачного провайдера, мы можем найти две функции, которые реализованы большинством исследуемых платформ, а именно модель затрат и политики планирования, которые поддерживаются 80% решений. Тем не менее, некоторые функции полностью покрываются лишь относительно небольшим процентом симуляторов, например соглашения об уровне обслуживания, которые поддерживаются только 40%. При этом другие возможности, такие как продление аренды и управление очередями обработки запросов пользователей, не реализованы ни в одном из предложений, за исключением Simcan2Cloud.

Однако другие характеристики, такие как предоставление гибкого и открытого API для создания рабочих нагрузок и управления различными типами пользователей, анализируемые решения не поддерживают (опять же исключая наше собственное предложение). Стоит отметить, что только CloudSimNet++ позволяет создавать приоритетные виртуальные машины, но не позволяет управлять несколькими пользователями профилей с разными приоритетами. Моделирование базовой облачной инфраструктуры является важным аспектом платформ моделирования, поскольку оно влияет на достоверность получаемых результатов. При этом только 20% симуляторов имеют высокодетализированную инфраструктуру, позволяющую проектировать гетерогенные системы. При этом мониторинг процента использования ресурсов возможен только в 26% систем.

Как показывает сравнение, существует немного платформ моделирования, предназначенных для описания облачного провайдера – с разумным уровнем детализации – при сохранении поддержки инфраструктуры. В общем, платформа моделирования, которая имеет большинство функций с Simcan2Cloud, — это CloudNetSim++. Однако CloudNetSim++ не учитывает некоторые важные аспекты облака, такие как продление времени аренды виртуальных машин, реализация пользовательских очередей для управления различными типами пользователей, обращающихся к облаку, приоритетные ресурсы, поддержка реальных трассировок и высокодетализированное инфраструктура.

Simcan2Cloud

В этом разделе представлено подробное описание симулятора Simcan2Cloud. Метаданные представлены в Таблице  [2](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab2) , в которой текущая версия симулятора, ссылка на репозиторий, содержащий исходный код, юридическая лицензия на код и используемая система управления версиями кода показаны в строках, обозначенных *C1* - *C4* . соответственно. Язык программирования, используемый для написания Simcan2Cloud, зависимости, требования к компиляции и документация показаны в строках, помеченных *C5* — *C7* соответственно. Адрес электронной почты службы поддержки указан в последней строке (с меткой *C8* ).

Этот раздел разделен на пять различных частей. Обзор Simcan2Cloud представлен в разделе « [Описание программного обеспечения](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec6) ». В разделе « [Соглашения об уровне обслуживания](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec7) » показано, как моделируются SLA с использованием различных параметров конфигурации, а основная архитектура и симуляционное ядро ​​симулятора представлены в разделе « [Архитектура](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec8) ». Далее в разделе « [API](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec9) » представлен API, который предоставляет возможности управления облачными средами. Наконец, в разделе  [3.5](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec10) представлен графический интерфейс пользователя — компонент, предназначенный для простого создания моделируемых сценариев.

**Таблица 2 Метаданные Simcan2Cloud**

[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/2)

Описание программного обеспечения

Simcan2Cloud — это модульная и гибкая платформа моделирования дискретных событий, которая позволяет пользователям моделировать и моделировать инфраструктуры облачных вычислений. Simcan2Cloud был написан с использованием C++ поверх OMNeT++, среды моделирования с открытым исходным кодом [ [33](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR33) ]. Стоит отметить, что OMNeT++ считается стандартом де-факто при моделировании распределенных систем. В частности, за последнее десятилетие OMNeT++ получил широкое распространение в исследовательском сообществе и промышленности для моделирования широкого спектра сложных систем [ [34](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR34) , [35](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR35) , [36](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR36) ]. Среди основных особенностей OMNeT++ можно выделить структурированное программирование и событийно-ориентированную модель, которые закладывают основу для высокой степени гибкости при проектировании распределенных систем. Однако симуляторы, созданные с использованием OMNeT++, требуют использования языка NED для настройки моделируемых сред. Этот факт в большинстве случаев является утомительной и подверженной ошибкам задачей из-за большого количества параметров, которые необходимо настроить в обычном текстовом файле. Чтобы облегчить эту задачу, Simcan2Cloud предоставляет простой в использовании графический интерфейс, который позволяет моделировать сложные инфраструктуры без необходимости глубокого знания этих систем. Кроме того, процесс моделирования улучшен за счет включения репозитория компонентов. Основная идея заключается в том, чтобы обеспечить повторное использование компонентов, необходимых для построения облачной среды, что позволит моделировать и настраивать большие сложные системы за считанные минуты.

Поставщик облачных услуг является одной из наиболее важных частей облачной системы, поэтому Simcan2Cloud включает в себя модульный и полностью настраиваемый модуль поставщика облачных услуг. Этот модуль в основном ориентирован на управление пользователями, политику планирования и распределения, финансовые затраты и соглашения об уровне обслуживания. Кроме того, модуль облачного провайдера позволяет включать как настроенные, так и известные виртуальные машины. Текущая версия Simcan2Cloud предоставляет большую коллекцию экземпляров виртуальных машин, вдохновленных Amazon EC2 [ [37](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR37) ]. Что касается физических ресурсов, Simcan2Cloud предоставляет несколько механизмов для моделирования четырех основных подсистем, а именно хранилища, вычислений, памяти и сети. По сути, объединение компонентов этих подсистем (т. е. дисковых накопителей, сетей связи, процессоров и памяти) позволяет пользователям создавать самые разнообразные облачные сценарии. Они могут варьироваться от небольшого количества физических машин до сложных и гетерогенных центров обработки данных.

Чтобы точно представить поведение облачных систем, Simcan2Cloud может генерировать и обрабатывать реалистичные рабочие нагрузки. Они создаются с использованием большого количества пользователей, от нескольких до тысяч. Приход этих пользователей на платформу можно определить с помощью различных функций распределения. Поведение пользователей, взаимодействующих с облаком, можно легко смоделировать, определив, по сути, запрашиваемые ВМ и приложения, выполняемые на этих ВМ. Управление пользователями, запрашивающими ресурсы у поставщика облачных услуг, осуществляется с помощью полностью настраиваемой системы очередей. Текущая версия Simcan2Cloud немедленно обрабатывает запросы пользователей, если доступны необходимые ресурсы. Напротив, если система не в состоянии предоставить запрошенные ресурсы, у пользователя есть две возможности. Первый вариант состоит в ожидании заранее определенного периода времени с ожиданием того, что запрошенные ресурсы со временем станут доступными. Второй – выйти из системы, не воспользовавшись никакими услугами. Отметим, что эта система очередей была специально разработана для обеспечения гибкости при включении новых политик планирования. Следовательно, эта функция позволяет расширить функциональность симулятора Simcan2Cloud. Стоит отметить, что были также учтены различные приоритеты для пользователей таким образом, чтобы можно было воспроизвести критерии приоритета, используемые наиболее распространенными поставщиками услуг. Таким образом, для распределения ресурсов, запрошенных пользователями, могут применяться различные политики. С этой целью Simcan2Cloud обеспечивает высокий уровень гибкости, позволяя балансировать запросы пользователей между различными центрами обработки данных. Это возможно благодаря выбору политик на основе занятости физических машин (т. е. избеганию фрагментации) и предоставлению разработчикам возможности реализовывать новые и настраиваемые политики. Аналогично, количество политик планирования в модуле гипервизора можно легко расширить и настроить. Одним из ключевых аспектов модуля облачного провайдера является возможность определять и включать новые соглашения об уровне обслуживания, основанные как на функциональных, так и на нефункциональных аспектах. Среди них стоит упомянуть функции и обеспечение, основанные на стоимости (т. е. уровень отказов и время ожидания). Определение соглашений об уровне обслуживания подробно описано в разделе « [Соглашения об уровне обслуживания](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec7) ».

Соглашения об уровне обслуживания

В этом разделе мы определяем, как Simcan2Cloud реализует соглашения об уровне обслуживания (SLA), которые используются в моделируемых облачных сценариях. Поставщики облачных услуг всегда должны подписывать соглашение об уровне обслуживания с пользователями облачных сервисов, желающими использовать облачные услуги. Следовательно, пользователь не может запросить виртуальную машину (ВМ) в облаке, если она не включена в подписанное соглашение об уровне обслуживания. Simcan2Cloud определяет соглашения об уровне обслуживания на основе того, как пользователи взаимодействуют с облаком и как о них заботятся.

Более конкретно, уровень архитектуры определяет несколько параметров SLA, связанных со стоимостью различных виртуальных машин в зависимости от типов пользователей. Текущая версия Simcan2Cloud позволяет различать два типа пользователей: *обычных* и пользователей *с высоким приоритетом* . Первый запрашивает виртуальные машины у поставщика облачных услуг, но может подождать, пока ресурсы станут доступными, если эти запросы не могут быть выполнены немедленно. Если пользователь решает подождать, он подписывается на характеристики виртуальной машины на определенный период, ожидая уведомления, когда виртуальная машина с этими функциями станет доступной. После получения уведомления пользователь начинает выполнение своих приложений. Если срок подписки истекает, а виртуальная машина не становится доступной, пользователь уходит, не имея возможности запускать свои приложения. Пользователи с высоким приоритетом должны немедленно получить запрашиваемые ими ресурсы, но они платят более высокую цену и должны получать компенсацию, если ресурсы не предоставляются.

Параметры, используемые для определения соглашения об уровне обслуживания в Simcan2Cloud, включают:

1. 1.

Базовая стоимость виртуальной машины ( *Base* ). Это стоимость одного часа обслуживания ВМ при обычных обстоятельствах, когда запрос может быть обработан немедленно. Если ни одна виртуальная машина недоступна (с запрошенными функциями), пользователь будет уведомлен об этом и получит скидку.

1. 2.

Скидки на базовую цену ВМ за задержки ( *Discount* ). Если нет доступных ресурсов для немедленного удовлетворения потребностей *обычных* пользователей, они могут подписаться и дождаться, пока запрошенные ресурсы станут доступными. В этом случае цена будет ниже, поэтому к цене аренды ВМ будет применена скидка к обычной стоимости. Пользователь может решить подождать, пока нужная виртуальная машина станет доступной, или уйти.

1. 3.

Увеличение базовой стоимости ВМ для пользователей с высоким приоритетом ( *Inc\_priority* ). Если пользователь решает иметь приоритетное поведение в облаке, он должен заплатить цену, превышающую базовую цену. Это приращение гарантирует, что поставщик облачных услуг резервирует некоторые компьютеры для использования, когда *обычные* (незарезервированные и постоянно работающие) компьютеры не могут удовлетворить требования пользователя. Следовательно, когда *обычные* машины недоступны, поставщик облачных услуг должен запустить резервную виртуальную машину для обслуживания пользователей с высоким приоритетом.

1. 4.

Стоимость дополнительного времени исполнения ( *Предложение* ). Если выполнение приложений, развернутых пользователем, не завершается в течение расчетного времени аренды, поставщик облачных услуг может предложить пользователю продолжить выполнение. Облачные провайдеры предлагают пользователям продление периода аренды по базовой цене за час плюс дополнительная плата. Таким образом, пользователь может либо заплатить за это дополнительное время выполнения, либо отказаться от него и прекратить взаимодействие с системой.

1. 5.

Стоимость компенсации из-за недоступности ресурса ( *Компенсация* ). Пользователи *с высоким приоритетом* платят дополнительную плату за гарантированное обслуживание. Таким образом, им необходимо возместить ущерб, причиненный в том маловероятном случае, если у облачного провайдера не будет доступных виртуальных машин. *Такая ситуация очень маловероятна, но она может возникнуть только в тех случаях, когда в пуле зарезервированных* машин нет доступной виртуальной машины с этими запрошенными функциями . По сути, этот сценарий возникает из-за неожиданного количества *высокоприоритетных* запросов пользователей или вследствие неправильной конфигурации в облаке. Решение этой проблемы, вероятно, будет сосредоточено на добавлении большего количества стоек. Таким образом, можно развернуть дополнительные виртуальные машины, сохраняя при этом сбалансированность системы.

Некоторые параметры являются эксклюзивными для *обычных* пользователей, например скидка на ожидание ресурсов. Другие параметры определяются только для пользователей *с высоким приоритетом* , например, повышенная стоимость получения немедленного внимания и компенсации в случае отсутствия ресурсов. Эти параметры позволяют нам выполнить анализ затрат в инструменте. Одной из наиболее примечательных характеристик Simcan2Cloud является его гибкость. Таким образом, мы намерены обогатить соглашения об уровне обслуживания новыми параметрами и поведением, например, изучая схемы закупок веб-сервисов Amazon и их комбинаций.

**Таблица 3. Конфигурации соглашений об уровне обслуживания для нескольких облачных сценариев.**

[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/3)

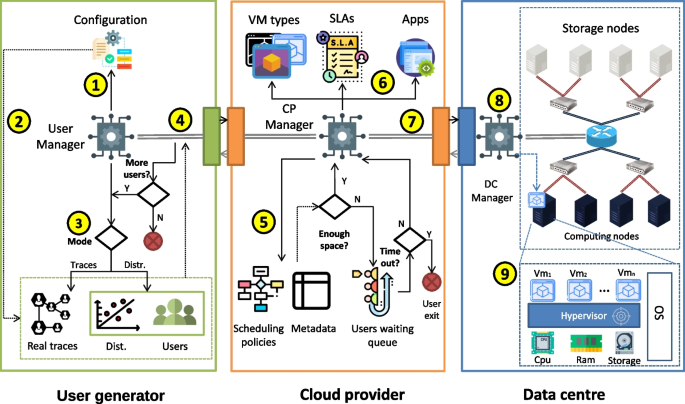
Таблица [3](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab3) иллюстрирует три примера различных возможных сценариев в облаке. Эти сценарии реализуются путем изменения параметров стоимости, определенных в соглашении об уровне обслуживания, подписанном между пользователями и поставщиком облачных услуг. На основе экземпляров ВМ AWS EC2 по требованию [ [37 ] для](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#ref-CR37)*базовой* установлена ​​стоимость 0,012 USD . Три сценария SLA напрямую влияют на доходы пользователей и поставщиков облачных услуг. ВСлА1, пользователи получают отличные предложения, включая скидку 60 % на изначально недоступные ресурсы, надбавку 10 % за приоритетное поведение и увеличение стоимости на 5 % за продление выполнения приложения. Пользователи с высоким приоритетом также получают 80% компенсации за недоступность.СлА2предлагает умеренные преимущества: скидку 20 %, надбавку 50 % за приоритет, увеличение стоимости на 30 % за продолжение подачи заявки и компенсацию 10 %. В отличие,СлА3сильно отдает предпочтение провайдеру с высокими ценами, предлагая только 10% скидку, 70% надбавку за приоритет, 70% увеличение стоимости продолжения приложения и 5% компенсацию. Доход облачного провайдера зависит от этих параметров, обеспечивая баланс между привлекательностью пользователей и максимизацией прибыли.

Архитектура

В облачных вычислениях пользователи имеют доступ к делокализованной вычислительной инфраструктуре через Интернет. Простота доступа к этим платформам и широкий выбор вычислительных конфигураций способствуют массовому использованию систем облачных вычислений. Однако значительный рост количества одновременно обращающихся к системе пользователей без учета соответствующего перемасштабирования инфраструктуры может привести к появлению узких мест. Проектирование облачной инфраструктуры играет ключевую роль в предотвращении этого сценария, но эта задача является сложной, поскольку необходимо учитывать и другие важные факторы. Среди них отметим, например, получение хорошего соотношения затрат и производительности, анализ поведения пользователей и управление ими при доступе к платформе.

На рисунке  [1](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig1) изображена основная архитектура Simcan2Cloud. По сути, облачный сценарий, смоделированный в Simcan2Cloud, состоит из трёх основных модулей: *Генератора пользователей* , *Облачного провайдера* и *Центра обработки данных* . Поведение каждого из этих модулей закодировано в подмодуле *Manager* . Таким образом, чтобы полностью настроить поведение облака, новые менеджеры можно запрограммировать с помощью API, показанного в Таблице  [4](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab4) .

**рисунок 1**

[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/1)

Основная архитектура Simcan2Cloud

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/1)

Модуль *генератора пользователей* (см. крайний левый модуль на рис.  [1](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig1) ) генерирует реалистичные рабочие нагрузки для обработки с помощью смоделированных облачных сценариев. Во-первых, необходимо предоставить параметры конфигурации, позволяющие настроить создание пользователей для представления рабочей нагрузки (метка 1). Эти параметры обрабатываются подмодулем *User Manager* для создания рабочей нагрузки, которая может быть представлена ​​реальной трассировкой или различными статистическими распределениями (метка 2). Далее рабочая нагрузка создается во время выполнения (метка 3). Важно отметить, что Simcan2Cloud представляет собой симулятор, основанный на дискретных событиях, и, следовательно, вычисления, необходимые для создания рабочей нагрузки, не влияют на операции, выполняемые для представления поведения целевой системы.

Модуль в центре рис.  [1](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig1) представляет собой *облачного провайдера* . В общих чертах основные задачи облачного провайдера заключаются в обслуживании запросов пользователей и возврате пользователям ответов, генерируемых дата-центрами. Более конкретно, поставщик облачных услуг отвечает за четыре основные задачи: i) обработка запросов виртуальных машин от пользователей; ii) Перенаправление заданий, запрошенных пользователями, в подходящий центр обработки данных таким образом, чтобы запрошенные виртуальные машины выполнялись на доступных физических ресурсах; iii) Управление подписками пользователей на запрошенные ресурсы; и iv) Определение политики затрат для каждого типа экземпляра виртуальной машины. Обратите внимание, что это поведение можно изменить, включив новый подмодуль *CP Manager* , который можно закодировать с помощью API, предоставляемого Simcan2Cloud. Как только пользователь попадает в облако (метка 4), менеджер облачного провайдера проверяет, подписал ли этот пользователь ранее соглашение об уровне обслуживания. Таким образом, их запросы обрабатываются соответствующим образом с использованием соответствующей политики планирования (метка 5).

Когда к ресурсам может получить доступ пользователь, который их запросил, субмодуль *CP Manager* (метка 6) создает виртуальные машины, содержащие приложения, которые необходимо выполнить. Виртуальные машины развертываются с использованием функций, отраженных в подписанном соглашении об уровне обслуживания. Далее запрошенные ВМ передаются в модуль *Дата-центра* для развертывания на доступных физических ресурсах (метка 7).

Как только ВМ поступает в модуль *Дата-центра* , *Диспетчер ЦОД* находит потенциальные физические машины для развертывания ВМ, которые должны удовлетворять аппаратным требованиям, указанным в настройках ВМ (метка 8). Затем, как только оборудование для развертывания виртуальной машины будет обнаружено, виртуальная машина настраивается на соответствующей физической машине и начинается ее выполнение (метка 9). Наконец, *диспетчер DC* отправляет сообщение менеджеру *CP* , чтобы обновить список доступных ресурсов в центре обработки данных.

На рисунке  [2](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig2) показана диаграмма классов, содержащая основные классы симулятора Simcan2Cloud и то, как эти классы связаны между собой. Более подробную информацию можно найти в Приложении [B.](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec17)Основные классы модуля *Генератор пользователей* показаны зеленым фоном. *UserBase* является основным объектом и обеспечивает базовую функциональность пользовательских модулей с точки зрения управления данными и структурами. Класс *UserGenerator* предоставляет различные методы для создания рабочих нагрузок, определяя точный момент прибытия пользователей в облако либо случайным образом, либо в соответствии с конкретным распределением с помощью класса *UserDistribution* . Сгенерированные рабочие нагрузки могут состоять из пользователей разных типов, например *PriorityUsers* и *RegularUsers* . Однако интеграцию новых генераторов пользователей и экземпляров пользователей можно выполнить путем создания новых классов, наследуемых от *UserGeneratorBase* и *UserBase* соответственно.

Наиболее важные классы, которые реализует модуль *Облачного провайдера* , на диаграмме классов показаны оранжевым фоном (см. рис.  [2](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig2) ). *CloudProviderBase* — это основной объект облачного провайдера. Некоторые из его наиболее важных функций — создание структуры и иерархии объектов, относящихся к центрам обработки данных, а также поиск информации о конкретном компоненте центра обработки данных. Интеграцию новых поставщиков облачных услуг можно выполнить путем создания нового класса, наследуемого от *CloudProviderBase* . Новый класс должен содержать новую политику выбора центров обработки данных, в которых развернуты запрошенные пользователями виртуальные машины. В настоящее время *CloudProviderFirstFit* наследует этот класс и реализует алгоритм, который выбирает первый центр обработки данных, которому соответствует запрос.

**Рис. 2**

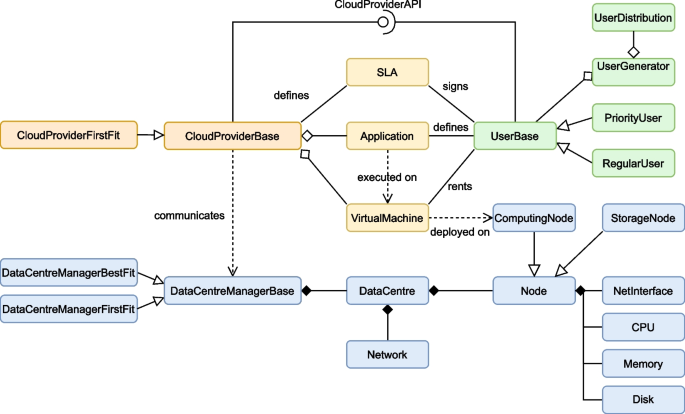
[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/2)

Диаграмма классов, представляющая основные классы ядра моделирования Simcan2Cloud.

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/2)

По сути, модуль *центра обработки данных* содержит физические ресурсы, поддерживающие облако, которые подразделяются на узлы хранения и вычислительные узлы. Физическая машина определяется путем настройки основных подсистем, то есть ЦП, хранилища, памяти и сети. Эти узлы соединены между собой через сеть связи и могут моделироваться независимо друг от друга, что позволяет создавать гетерогенные центры обработки данных. Вся инфраструктура центра обработки данных управляется компонентом, который называется *менеджером центра обработки данных* . Основные классы инфраструктуры Simcan2Cloud показаны синим фоном на диаграмме классов, изображенной на рис.  [2](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig2) . *DataCentreManagerBase* — это основной объект центра обработки данных, и его наиболее важными функциями, среди прочего, являются размещение виртуальных машин на физических машинах и планирование заданий, которые будут выполняться на экземплярах виртуальных машин. Интеграцию новых менеджеров центров обработки данных можно выполнить путем создания нового класса, наследуемого от *DataCentreManagerBase* . В настоящее время Simcan2Cloud включает в себя два разных менеджера центров обработки данных, а именно *DataCentreManagerFirstFit* и *DataCentreManagerBestFit* . Первый размещает виртуальные машины, запрошенные пользователем, в первом доступном слоте списка, содержащего доступные ресурсы. Этот слот может быть узлом или стойкой, в зависимости от вычислительных ресурсов, необходимых для размещения запрошенных виртуальных машин. Последний ориентирован на предотвращение фрагментации, и, следовательно, эта политика касается распределения пользовательских запросов в слоте, который имеет наименьшее количество доступных ресурсов и в который помещается запрос. Кроме того, этот класс предоставляет различные методы для получения информации из ресурсов центра обработки данных, например о количестве доступных ядер ЦП и общем количестве ядер ЦП. Сущность *DataCentre* состоит из набора физических ресурсов, необходимых для правильного функционирования системы, таких как списки вычислительных ресурсов и ресурсов хранения, созданных классом *Nodes* .

API

Для этого он реализует основной функционал платформы моделирования, то есть механизм генерации пользователей и функционал облачного провайдера. Благодаря использованию этого API можно включать в платформу новых пользователей и экземпляры облачных провайдеров с настраиваемым поведением.

В этом разделе описывается API Simcan2Cloud. Основные методы этого API сведены в Таблице  [4](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab4) . Для наглядности показаны только методы, принадлежащие основным модулям — *генерации пользователей* , *облачного провайдера* и *менеджера центра обработки данных .*В первом столбце показан облачный компонент, содержащий метод, а в следующих столбцах указаны идентификатор, имя и описание метода. Важно отметить, что эти методы реализованы в текущей версии Simcan2Cloud и при необходимости могут быть перезаписаны для обеспечения определенных функций. Идея состоит в том, что с помощью этого API в симулятор можно легко включить новые модули, такие как политики распределения ресурсов и поведение пользователей.

Ядро генерации пользователей подробно описано в методах 1–15. Метод 1 инициализирует структуры данных, необходимые для запуска моделирования, а методы 2–3 управляют пользователями, прибывающими в облако. Точнее говоря, *generShuffledUsers* генерирует рабочую нагрузку, случайным образом определяя момент прибытия пользователей в облако. Этот метод особенно полезен для воспроизведения случайности доступа пользователей к облаку в реальных средах. После создания рабочая нагрузка может быть повторена с помощью метода *getNextUser* , который обеспечивает обработку следующего пользователя в облаке.

Создание и управление виртуальными машинами осуществляется методами 4–9. Виртуальные машины, запрошенные пользователями, создаются и отправляются провайдеру облака с помощью методов *createVmRequest* и *sendRequest* соответственно. В зависимости от ресурсов, доступных в облаке, поставщик облака отправляет сообщение с уведомлением, которое обрабатывается и обновляется пользователем с помощью методов *handleResponseVmAccept* , *handleResponseVmReject* и *updateVmUserStatus* . Если провайдер облака отправляет сообщение об отказе, что означает, что запрошенные пользователем виртуальные машины не помещаются в центр обработки данных, пользователь может отправить запрос на подписку в облако, используя метод *подписки* . На данный момент у разработчиков существует несколько вариантов создания нового поведения пользователей, например, выбор между подпиской на поставщика облачных услуг или оставлением облака без внимания их запроса.

Услуги, необходимые пользователям, обрабатываются способами 10-14. Приложения, которые будут выполняться на виртуальных машинах, создаются и отправляются поставщику облака с помощью методов *createAppRequest* и *submitService* . Ответ поставщика облака управляется методами *handleResponseAppAccept* , *handleResponseAppReject* и *handleResponseAppTimeout* . Как и в случае с запросами виртуальных машин, можно создать новое поведение пользователя, учитывая решения пользователя в зависимости от того, отклоняет ли поставщик облачных услуг запрос. Например, пользователь может выбрать другое приложение или уменьшить количество приложений, выполняемых на виртуальной машине. Наконец, метод 15 генерирует отчет, содержащий статистику, полученную в ходе моделирования.

Функциональность облачного провайдера управляется методами 16-21. Создание, инициализация и настройка как инфраструктуры центра обработки данных, так и облачного провайдера выполняются с помощью метода 16, а методы 17–18 управляют виртуальными машинами, запрошенными пользователем. Метод *checkVmUserFit* анализирует, соответствуют ли доступные ресурсы одного из дата-центров требованиям запроса. Затем, если запрос подходит дата-центру, то есть запрошенные ВМ могут быть выполнены на доступных ресурсах, облачный провайдер пересылает запрос в выбранный дата-центр.

Подписка пользователей в облако управляется методами 18 и 20. Метод *updateSubsQueue* анализирует очередь подписки, чтобы найти таймауты, достигающие максимального времени ожидания получения пользователями запрошенных ресурсов, и выбирает следующие запросы для обработки. Стоит отметить, что новые системы очередей для управления пользователями можно запрограммировать, перезаписав метод *updateSubsQueue* . Метод *notifySubscription* уведомляет пользователя о доступности запрошенных ресурсов. Метод *timeoutSubscription* управляет временем ожидания для пользователей, когда истекает время ожидания подписки, а запрошенные ресурсы остаются недоступными. Метод 21, а именно *handleUserAppRequest* , управляет службами, предоставленными пользователями, и перенаправляет запрос в центр обработки данных, где ранее были выделены виртуальные машины.

Функциональность менеджера дата-центра отражена в методах 22-31. Создание, инициализация и настройка инфраструктуры дата-центра выполняются методом 22. Предположим, что в симулятор включена новая политика распределения ресурсов, содержащая новые и сложные структуры данных. В этом случае методы 16 и 22, оба с именем *Initialise* , должны быть перезаписаны, чтобы обрабатывать новые структуры данных. Методы 23–27 управляют виртуальными машинами, запрошенными пользователем. Метод 23, а именно *checkVmUserFit* , анализирует, соответствуют ли доступные ресурсы центра обработки данных требованиям запроса и подходит ли запрос в центр обработки данных. Эта проверка направлена ​​на вычисление того, содержит ли центр обработки данных достаточное количество компьютеров со свободными ресурсами для запуска запрошенных виртуальных машин. В таком случае менеджер центра обработки данных выделяет запрошенные виртуальные машины и отправляет пользователю сообщение о принятии с помощью методов *AcceptVmRequest* и *allocateVM* с идентификаторами 25 и 27 соответственно. В противном случае, если запрос не подходит системе, менеджер центра обработки данных отправляет сообщение об отклонении с помощью метода 26, а именно *ignoreVmRequest* . В текущую версию Simcan2Cloud, как упоминалось в разделе «Архитектура», включены две разные политики распределения, а именно « *наилучшее соответствие»* и *«первое соответствие»* . Однако интеграцию новых политик можно легко выполнить, внедрив новых поставщиков облачных услуг с разными политиками в методах *checkVmUserfit* и *allocateVM* . Методы 28-31 обрабатывают услуги, запрошенные пользователями. Подобно методу 21, *handleUserAppRequest* управляет службами, предоставленными пользователями, и позволяет принять или отклонить запрос с помощью методов 29 и 30 соответственно. Наконец, если выполнение приложения превышает время аренды, пользователю отправляется уведомление о таймауте с помощью метода 31, а именно *timeoutAppRequest* .

**Таблица 4. Выдержка из Simcan2Cloud API**

[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/4)

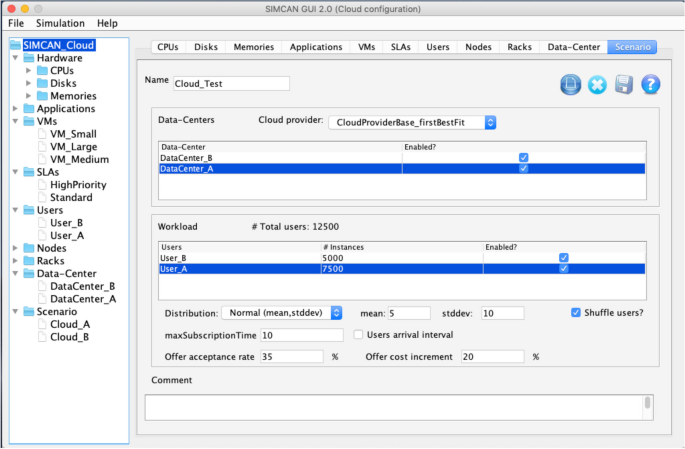
графический интерфейс

Simcan2Cloud позволяет моделировать и проектировать облачные инфраструктуры с высоким уровнем гибкости, настраивая, среди других модулей, политики распределения ресурсов, центры обработки данных и рабочие нагрузки. По сути, конфигурация облака, которое необходимо моделировать в Simcan2Cloud, состоит из двух текстовых файлов, один из которых определяет общую архитектуру облака (центр обработки данных, поставщика облака и поколение пользователей), а другой содержит все необходимые параметры. настроить каждый модуль в облачной среде (особенности каждой физической машины, конфигурация виртуальных машин и распределение пользователей в рабочей нагрузке и т. д.). Следовательно, ручная установка всех необходимых параметров для настройки моделируемого облака — это трудоемкая и подверженная ошибкам задача.

Чтобы облегчить настройку и взаимодействие с механизмом моделирования, Simcan2Cloud предоставляет графический интерфейс пользователя (GUI) (см. рис.  [3](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig3) ) и состоит из трех основных частей: панели с вкладками, древовидной панели и меню.

Панель дерева, расположенная слева от панели с вкладками, показывает репозиторий симулятора Simcan2Cloud, который состоит из всех элементов, которые ранее были смоделированы с помощью панели с вкладками. Эта древовидная панель также позволяет повторно использовать, редактировать и удалять компоненты из репозитория.

**Рис. 3**

[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/3)

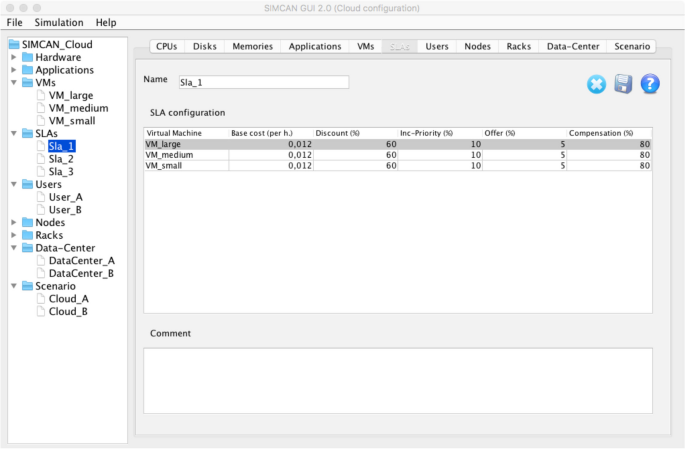
Снимок экрана графического интерфейса Simcan2Cloud

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/3)

Панель с вкладками состоит из 10 различных редакторов, которые позволяют пользователям моделировать и настраивать различные части облака, то есть процессоры, диски, память, приложения, виртуальные машины, соглашения об уровне обслуживания, пользователей, узлы, стойки, центры обработки данных и сценарии. Первые три вкладки, а именно *CPUs* , *Disks* и *Memories* , показывают способы настройки вычислительных ресурсов физической машины. Эти компоненты используются для настройки как вычислительных узлов, так и узлов хранения.

Приложения, отправленные пользователями для выполнения на виртуальной машине, можно смоделировать на панели *«Приложения* » путем настройки общего количества обрабатываемых операций ЦП и ввода-вывода. Таким образом, можно создавать различные типы приложений, например, приложения с интенсивным использованием данных и процессора, которые ориентированы на обработку большого количества операций ввода-вывода и процессора соответственно. Альтернативно, количество итераций также можно настроить. Таким образом, конечный пользователь может моделировать продолжительность выполнения, когда приложение совмещает операции ЦП и ввода-вывода.

**Рис. 4**

[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/4)

Снимок экрана конфигурации SLA в графическом интерфейсе Simcan2Cloud

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/4)

На панели *виртуальных машин* настраиваются количество ядер, емкость хранилища (измеряется в ГБ), стоимость часа, количество ядер и память (измеряется в ГБ) каждой виртуальной машины.

Соглашения об уровне обслуживания настраиваются на панели *SLA* . Таким образом, при создании нового SLA параметры конфигурации существующих ВМ в репозитории отображаются в таблице. На рисунке  [4](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig4) показана конфигурация *Sla\_1* . В этом случае три виртуальные машины, хранящиеся в репозитории (см. левый кадр на рисунке), то есть *VM\_large* , *VM\_medium* и *VM\_small* , можно настроить для текущего соглашения об уровне обслуживания. В таблице можно установить стоимость и другие параметры, такие как предложение и вознаграждение. После назначения этих параметров пользователь может сохранить SLA в репозитории, который будет отображаться в дереве компонентов.

Вкладка « Пользователи *»* позволяет моделировать поведение пользователей, взаимодействующих с облаком. Для этого необходимо настроить количество и тип запрошенных виртуальных машин, а также приложение, которое будет выполняться на этих виртуальных машинах.

Вкладка *«Центр обработки данных»* позволяет настроить центр обработки данных, поддерживающий облако, что моделируется путем настройки вычислительных стоек и стоек хранения данных. По сути, стойка представляет собой совокупность узлов, соединенных между собой посредством сети связи. Чтобы смоделировать стойку, пользователь должен выбрать конфигурацию узлов — на вкладке *«Узлы»* — и количество предоставленных узлов. Отметим, что модульная конструкция Simcan2Cloud позволяет легко моделировать различные облачные инфраструктуры, используя компоненты репозитория (см. левый фрейм графического интерфейса). Наконец, пользователь должен настроить сеть связи для соединения стоек центра обработки данных.

Сценарий моделирования можно смоделировать на панели *«Сценарий»* . Для этого необходимо выбрать базовую инфраструктуру облака (дата-центры), обрабатываемую рабочую нагрузку, распределение пользователей, прибывающих в облако, а также несколько параметров, связанных со временем и стоимостью подписки.

Наконец, меню, расположенное в верхней части графического интерфейса, содержит параметры управления. Это меню создает файлы конфигурации и отображает справку Simcan2Cloud и другие вспомогательные операции.

Эмпирическое исследование

В этом разделе представлено тщательное эмпирическое исследование, в котором моделируются различные облачные конфигурации для проверки применимости и удобства использования Simcan2Cloud. Каждая облачная конфигурация имеет однородный центр обработки данных, то есть все физические машины содержат одинаковые функции: четырехъядерный процессор, 64 ГБ оперативной памяти и запоминающее устройство емкостью 500 ГБ с пропускной способностью чтения/записи 500 Мбит/с. . Эти физические машины, на которых размещаются виртуальные машины, запрошенные пользователями, соединены между собой через сеть связи Gigabit Ethernet, а диспетчер облака подключен к облаку через сеть Ethernet 40 Гбит/с. В этом исследовании мы используем четыре различные конфигурации дата-центра, поддерживающего облако, состоящие из 128, 256, 512 и 1024 физических машин.

Чтобы проанализировать различные особенности платформы, мы разделили эмпирическое исследование на две части. Первый фокусируется на изучении поведения облачных систем с учетом синтетических рабочих нагрузок, а второй заключается в анализе следов, извлеченных из реальной системы.

Эксперимент 1. Синтетические рабочие нагрузки и конфигурации с несколькими процессорами

В этой части мы используем две разные конфигурации ЦП — с разной вычислительной мощностью ядер ЦП — чтобы проанализировать, как вычислительная мощность влияет на общую производительность системы. Кроме того, мы создали синтетическую рабочую нагрузку, основанную на экспоненциальном распределении, для проведения экспериментов. Таким образом, каждое облако обрабатывает рабочую нагрузку, состоящую из 10 000 пользователей, запрашивающих ресурсы. Эта рабочая нагрузка была создана с использованием четырех разных ролей пользователей и трех разных конфигураций виртуальных машин. Все пользователи с одинаковой ролью ведут себя одинаково. В таблице  [5](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab5) показана конфигурация виртуальных машин, где первый столбец относится к имени виртуальной машины, а следующие столбцы представляют соответственно ЦП, память и хранилище, используемые виртуальной машиной. Подробное описание рабочей нагрузки приведено в Таблице  [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab6) , в которой в первом столбце показано имя роли пользователя, во втором столбце указано количество экземпляров, созданных для этой роли пользователя, а в последнем столбце указано количество запрошенных для этой роли ВМ. определенный период времени. В частности, эта рабочая нагрузка содержит, помимо других экземпляров пользователей, 3725 экземпляров пользователей LemmingUser , каждый из которых запрашивает 2 виртуальные машины vmMedium на два часа. Временная *метка* , указывающая, когда пользователи приходят в систему, была рассчитана с использованием экспоненциального распределения со *средним значением = 60,5 секунды* . На каждой запрошенной виртуальной машине пользователи развертывают ресурсоемкое приложение, которое выполняет пять итераций следующих действий: чтение 10 МБ данных с диска.→выполнить 79 200 000 МИ→записать 5 МБ данных на диск. Максимальное время подписки составляет 10 часов, а это значит, что те пользователи, которые не смогли за это время получить доступ к запрошенным ресурсам, оставили систему без присмотра. Наконец, 90% тех пользователей, которым требовалось больше времени для выполнения поданных заявок – после истечения срока аренды – запросили продление, чтобы обеспечить успешное выполнение заявок.

**Таблица 5. Конфигурация виртуальных машин для генерации рабочей нагрузки**

[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/5)

**Таблица 6. Конфигурация ролей пользователей для создания рабочей нагрузки**

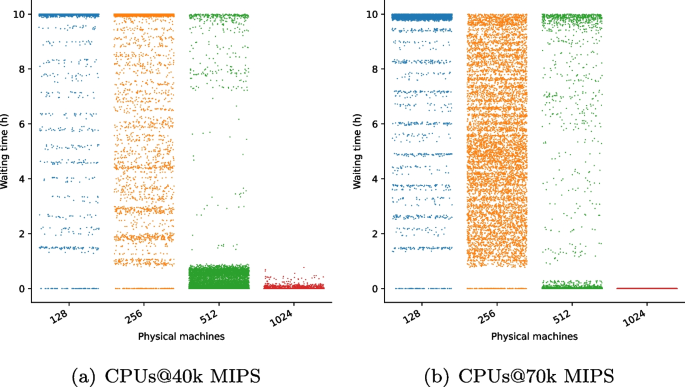
[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/6)

На рис.  [5](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig5) показана общая производительность системы при обработке рабочей нагрузки. По оси Y показано время ожидания каждого пользователя, то есть время, прошедшее с момента запроса пользователем ресурсов у облачного провайдера до момента, когда пользователь получит к ним доступ. Это время ожидания рассчитывается с учетом всех пользователей, ожидающих ресурсов в системе. По оси X показаны четыре конфигурации центра обработки данных, поддерживающего облако, состоящие из 128, 256, 512 и 1024 физических машин, которые показаны синим, оранжевым, зеленым и красным цветом соответственно. Каждая точка представляет пользователя, о котором заботились, то есть облако смогло предоставить запрошенные ресурсы. Обратите внимание, что на этих диаграммах не показаны пользователи без присмотра. Этот эксперимент проводился с использованием двух разных конфигураций процессора. Так, на рис.  [5а](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig5) и б показаны результаты для физических машин, содержащих ЦП с вычислительной мощностью 40 тыс. MIPS и 70 тыс. MIPS соответственно.

На рисунке  [5а](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig5) видно, что при поддержке облака 128 физическими машинами точки на диаграмме разбегаются, что наглядно отражает то, что система достигает точки насыщения и, следовательно, мало пользователей получают доступ к запрошенным ресурсам (синий столбец). Когда количество физических машин увеличивается до 256 и 512 (оранжевый и зеленый столбцы), на диаграмме отображается другой сценарий, в котором точки более сгущены, что означает, что большее количество пользователей получает доступ к ресурсам, что снижает насыщенность система. Однако когда облако содержит 512 узлов, мало кто из пользователей ждал почти десять часов, чтобы получить доступ к запрошенным ресурсам. Наконец, когда облако настроено на 1024 физических машины, нагрузка на систему значительно снижается, что позволяет облаку предоставлять ресурсы всем пользователям в рабочей нагрузке.

На диаграмме Рис.  [5б](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig5) показаны результаты обработки рабочей нагрузки при увеличении мощности ЦП физических машин до 70 тыс. MIPS. В этом случае мы наблюдаем тенденцию, аналогичную той, что показана на предыдущем графике, то есть увеличение количества физических ресурсов улучшает общую производительность системы за счет уменьшения времени ожидания. Однако стоит отметить, что хотя облако также насыщается, когда количество физических машин равно или меньше 512, время ожидания короче. Основное различие между этими сценариями — с использованием CPU@40k MIPS и CPU@70k MIPS — заключается в количестве приложений, которые были успешно выполнены до истечения срока аренды виртуальных машин. Таким образом, в тех случаях, когда все поданные пользователем заявки полностью выполняются, запрошенные ВМ освобождаются, пользователь покидает систему, а запросы новых пользователей обрабатываются. Напротив, когда приложения выполняются не полностью, пользователь может запросить продление и, следовательно, эти ресурсы не предоставляются новым пользователям. В этом случае система может полностью перегрузиться, если доступных ресурсов окажется недостаточно для обработки рабочей нагрузки. Когда облако содержит 1024 физических машины, все пользователи обслуживаются немедленно, что показано в красном столбце.

**Рис. 5**

[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/5)

Оценка производительности с использованием разных процессоров на физических машинах

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/5)

Таблицы  [7](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab7) и  [8](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab8) показывают подробную информацию об этом эксперименте. В первом столбце, помеченном как *«Машины* », показано количество физических компьютеров, поддерживающих облако. Следующие столбцы, помеченные как *U. Attended* , *Average* , *Std* и *Min* , показывают (в часах) количество обслуживаемых пользователей, среднее время ожидания, стандартное отклонение и минимальное время ожидания для всех пользователей в рабочей нагрузке. которые были успешно рассмотрены, соответственно. Следующие три столбца отображают 25тчас, 50тчаси 75тчаспроцентиль, то есть после сортировки данных – от низшего к высшему – время ожидания, ниже которого находятся 25%, 50% и 75% пользователей соответственно.

**Таблица 7. Результаты, полученные при обработке рабочей нагрузки с использованием CPUs@40k MIPS**

[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/7)

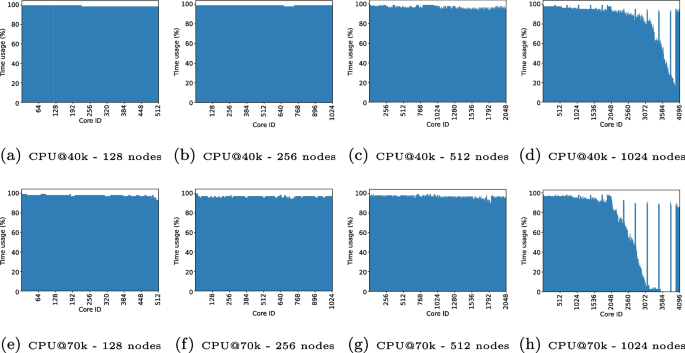
**Таблица 8. Результаты, полученные при обработке рабочей нагрузки с использованием CPUs@70k MIPS**

[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/8)

В этом эксперименте два параметра моделирования оказывают непосредственное влияние на общую производительность системы. Во-первых, мощность процессора, обеспечиваемая физическими машинами. При улучшении процессора мы наблюдаем увеличение количества обслуживаемых пользователей (см. столбец с надписью *U. Attended* ). Когда облако предоставляет 128 физических машин, мы видим, что использование CPU@70k MIPS позволяет облаку успешно обслуживать 4399 пользователей (см. Таблицу  [8](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab8) ). Однако при использовании CPU@40k MIPS обслуживаются только 2163 пользователя (см. Таблицу  [7](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab7) ). Важно отметить, что это улучшение более заметно, когда облако использует небольшое количество физических машин (128 и 256). Использование большего количества физических машин приводит к одинаковому результату для обоих процессоров. Фактически, эти облака, предоставляющие 1024 физических машины, успешно обслуживают всех пользователей в рабочей нагрузке с обоими процессорами. Кроме того, среднее время ожидания сокращается только тогда, когда облако предоставляет 512 физических машин: с 0,771 часа при использовании CPU@70k MIPS до 0,634 часа при использовании CPU@40k MIPS. В остальных случаях использование самого мощного процессора привело к увеличению времени ожидания, поскольку количество обслуживаемых пользователей значительно больше. Второй параметр — это размер облака, представленный количеством физических машин, который напрямую влияет на количество успешно обслуживаемых пользователей. В частности, этот параметр напрямую связан с насыщенностью системы, особенно когда количество физических машин, поддерживающих облако, невелико и, следовательно, не позволяет системе полностью обрабатывать рабочую нагрузку. Следовательно, по мере увеличения количества физических машин в системе развертывается большее количество виртуальных машин и, следовательно, обслуживается большее количество пользователей. Мы видим, что эта разница более заметна при использовании небольшого количества физических машин. Обе таблицы показывают, что насыщенность облака явно снижается, когда система содержит более 512 физических машин.

На рис.  [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig6) показано использование времени каждым ядром ЦП при обработке рабочей нагрузки. По оси X показан идентификатор ядра, а по оси Y — процент использования времени для каждого ядра ЦП, где 100 % представляет собой общее время, необходимое для полной обработки рабочей нагрузки. Эти диаграммы показывают, что когда облако предоставляет до 512 физических машин, использование времени для каждого процессора составляет почти 100%. Эти случаи явно отражают насыщенность системы. Однако когда в облаке используется 1024 физических машины, мы замечаем, что имеется значительное количество ядер ЦП с низким процентом использования времени. В частности, это можно наблюдать в ядрах ЦП в диапазоне от 3300 до 4096 – на рис.  [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig6) г – и в ядрах ЦП в диапазоне от 2500 до 4096 на рис.  [6](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig6) з.

**Рис. 6**

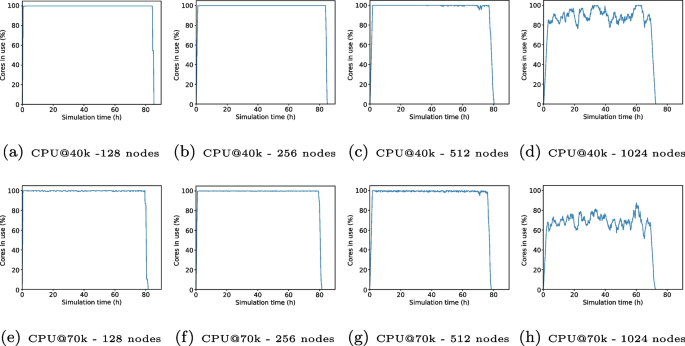
[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/6)

Использование времени (в %) каждого ядра ЦП для обработки рабочей нагрузки.

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/6)

На рис.  [7](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig7) показан процент ядер ЦП, используемых при обработке рабочей нагрузки. Ось X показывает временную шкалу, а ось Y представляет процент используемых ядер ЦП.

**Рис. 7**

[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/7)

Процент ядер ЦП, используемых для обработки рабочей нагрузки

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/7)

Как и в предыдущем эксперименте, в этом случае мы также можем наблюдать насыщенную систему, когда количество физических машин, поддерживающих облако, равно или меньше 512. В частности, за общее время было использовано почти 100% ядер ЦП. необходимые для обработки рабочей нагрузки. Однако увеличение количества физических машин до 1024 приводит к совершенно другому сценарию, который дает разные результаты в зависимости от используемого процессора. Таким образом, при использовании MIP CPUs@40k процент используемых ядер ЦП во время моделирования варьировался от 75% до 100%. Однако использование самого быстрого ЦП значительно повышает общую производительность системы: процент требуемых ЦП колеблется от 57% до 83%, что позволяет облаку немедленно обрабатывать новые запросы пользователей, то есть пользователи обслуживаются по мере запроса ресурсов.

Эксперимент 2: Реальные трассировки и соглашения об уровне обслуживания

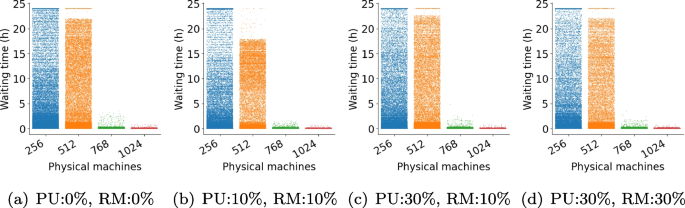
Основная цель этого эксперимента — проанализировать поведение облака с учетомСлА2, описанный в Таблице  [3](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab3) . В частности, мы смоделировали четыре разных сценария, в которых был выбран разный процент приоритетных пользователей (PU) и зарезервированных машин (RM):

* 0% приоритетных пользователей и 0% зарезервированных машин.
* 10% приоритетных пользователей и 10% зарезервированных машин.
* 30% приоритетных пользователей и 10% зарезервированных машин.
* 30% приоритетных пользователей и 30% зарезервированных машин.

В этом эксперименте, в отличие от предыдущего, где для представления пользователей, посещаемых облаком, используется синтетическая рабочая нагрузка, мы используем трассировку, записанную в формате SWF, полученную из реальной системы.[Сноска1](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fn1) . В частности, этот след состоит из 51987 рабочих мест, запущенных с мая 2014 года по август 2014 года и извлеченных из кластера GAIA, который расположен в Университете Люксембурга. Эта трассировка была предварительно обработана для удаления заданий с недопустимыми параметрами, например, временем выполнения = 0. В результате было удалено 130 заданий, в результате чего в симуляторе осталось в общей сложности 51 857 заданий. Что касается центра обработки данных, поддерживающего облако для обработки этой трассировки, мы смоделировали четыре различные конфигурации, состоящие из 256, 512, 768 и 1024 физических машин. Поскольку в трассировке не указаны виртуальные ресурсы, запрошенные для выполнения каждого задания, мы используем*экземпляр vmSmall* (описанный в таблице [5](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab5) ) для всех пользователей в моделируемой среде. Этот эксперимент проводился исходя из того, что каждую виртуальную машину необходимо арендовать минимум на 1 час. Чтобы сравнить результаты, полученные с помощью Simcan2Cloud, и результаты, полученные в эксперименте, выполненном на кластере GAIA, мы расширили эксперименты, убрав это ограничение по времени.

На рис  [. 8](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig8) показана общая производительность системы при обработке ранее описанной трассы. Ось Y представляет время ожидания, а ось X показывает количество физических машин, поддерживающих облако. Кроме того, мы провели эксперимент, варьируя процент приоритетных пользователей и зарезервированных машин. Как мы наблюдали в предыдущих экспериментах, увеличение количества узлов уменьшает время ожидания пользователей. В частности, этот эксперимент показывает, что пропорциональное увеличение количества как приоритетных пользователей, так и зарезервированных машин также уменьшает время ожидания. В частности, это можно увидеть на рис.  [8](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig8) а и б, где процент приоритетных пользователей (PU) и зарезервированных машин (RM) варьируется от 0% до 10% соответственно. Стоит отметить, что приоритетные пользователи не ждут в очереди у облачного провайдера.

**Рис. 8**

[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/8)

Оценка производительности с использованием разного процента приоритетных пользователей и зарезервированных машин.

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/8)

Подробности этих экспериментов показаны в таблице  [9](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab9) , где первый столбец, помеченный как *«Конфигурация»* , представляет процент приоритетных пользователей и зарезервированных машин. Во втором столбце указано количество компьютеров, составляющих облако. Следующие три столбца относятся к количеству посещенных пользователей, то есть количеству посещаемых приоритетных пользователей ( *Pri. Att.* ), количеству приоритетных пользователей, которые были посещены в качестве обычных пользователей ( *Pri. Reg. Att. ). .* ) и количество посещенных постоянных пользователей ( *Reg. Att.* ). Далее, в следующих трех столбцах показано количество пользователей, которые не обслуживались, то есть общее количество необслуживаемых пользователей ( *Total Unatt.* ), обычных пользователей без присмотра ( *Reg. Unatt.* ) и приоритетных пользователей без присмотра ( *Pri. Унатт.* ). Наконец, в последнем столбце показан общий доход, полученный провайдером облака для каждой конфигурации облака ( *Total Income* ).

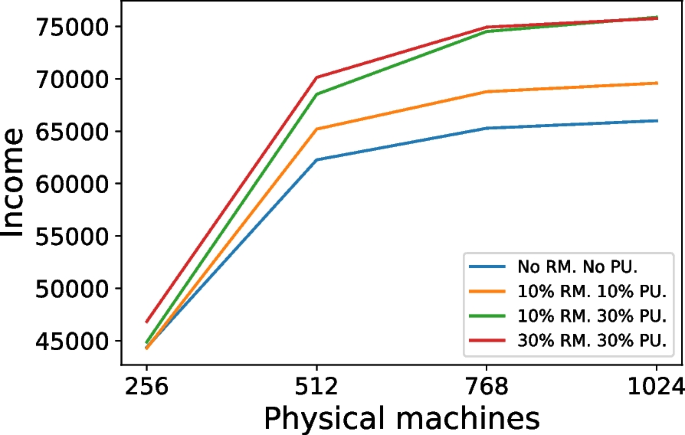
**Таблица 9. Сводка результатов, полученных в *эксперименте 2.***

[**Полноразмерный стол**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/tables/9)

После тщательного анализа этих результатов мы видим, что, учитывая особенности SLA, увеличение процента зарезервированных машин в дата-центре оказывает прямое влияние на общую производительность системы, вызывая уменьшение количества оставленных без присмотра приоритетных пользователей и , одновременно увеличивая количество необслуживаемых постоянных пользователей (см. первую и вторую строку в Таблице  [9](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab9) ). Такая ситуация возникает из-за того, что обычные пользователи не имеют доступа к зарезервированным машинам и, следовательно, количество доступных узлов для этого типа пользователей в этой конфигурации уменьшено. Аналогичным образом, увеличение количества приоритетных пользователей пропорционально количеству зарезервированных машин положительно влияет на общую производительность, что отражается в большом количестве посещаемых приоритетных пользователей и необслуживаемых обычных пользователей (см. первую, вторую и четвертую строки в Таблице 9  [)](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab9) . . В этом конкретном случае приоритетные пользователи могут обслуживаться как обычные пользователи, что сокращает количество доступных узлов для оставшихся обычных пользователей. Однако увеличение количества приоритетных пользователей без учета зарезервированных машин приводит к увеличению количества необслуживаемых приоритетных пользователей (см. третью и четвертую строки в Таблице  [9](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Tab9) ).

Что касается общего дохода, на рис.  [9](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig9) показана сводная информация о прибыли облачного провайдера, которую можно рассчитать с использованием различных конфигураций физических машин, а также процентных долей приоритетных пользователей и зарезервированных машин для обработки реальной трассировки. В этом случае мы наблюдаем, что при использовании меньшего количества физических машин лучший доход достигается за счет отказа от использования приоритетных пользователей и зарезервированных машин. Напротив, когда количество физических машин увеличивается, а количество приоритетных пользователей увеличивается пропорционально количеству зарезервированных машин, результаты показывают иное, что приводит к достижению более высоких доходов.

**Рис. 9**

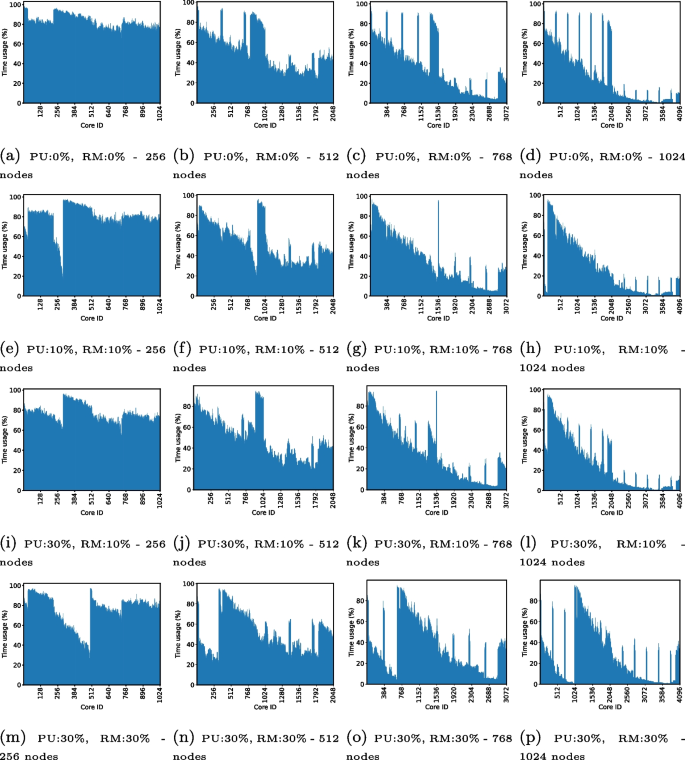
[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/9)

Доход облачного провайдера за обработку реальной рабочей нагрузки

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/9)

Как и в *эксперименте 1* , мы проанализировали использование времени каждым процессором центра обработки данных при обработке реальной трассировки. На рис.  [10](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig10) показано использование ЦП с учетом четырех конфигураций (с разным процентом приоритетных пользователей и зарезервированных машин), разработанных в этом эксперименте. Как и ожидалось, увеличение количества физических машин положительно влияет на общее использование облака. Что касается зарезервированных машин, то чем больше соотношение между зарезервированными машинами и приоритетными пользователями, тем ниже процент использования зарезервированных машин. Этот факт можно увидеть по низкому пику, расположенному на рис.  [10](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig10) д от CPU 200 до 256. Однако при уменьшении этого соотношения (увеличении PU и сохранении того же количества RM) процент использования зарезервированных машин увеличивается. При этом низкий пик, обнаруженный на предыдущем графике, ослабляется, как это видно на рис.  [10и](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig10) .

**Рис. 10**

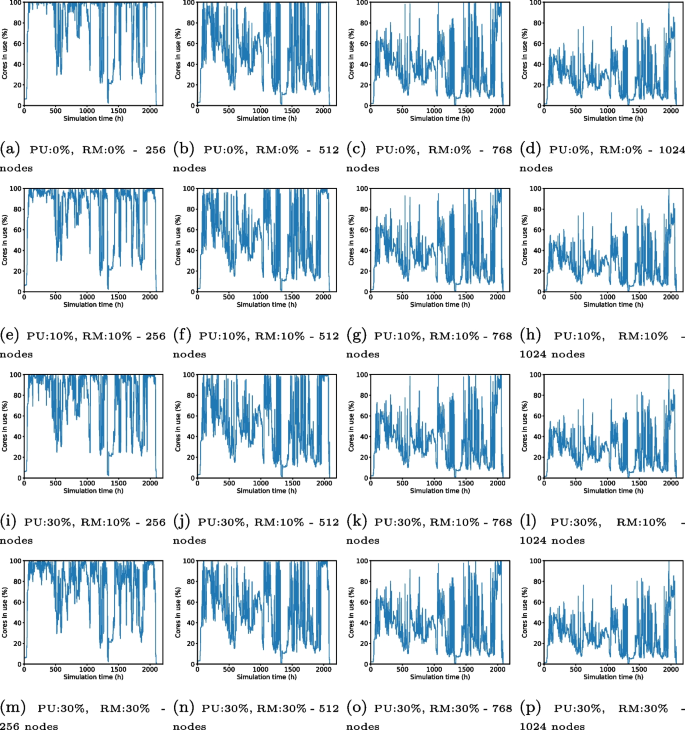
[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/10)

Использование времени (в %) каждого ядра ЦП для обработки реальной рабочей нагрузки.

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/10)

Наконец, процент ядер ЦП, используемых в центре обработки данных, показан на рис.  [11](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig11) . Как и в предыдущих экспериментах, увеличение количества узлов приводит к снижению процента использования кластера. В частности, мы замечаем, что увеличение количества приоритетных пользователей и зарезервированных машин немного снижает процент использования платформы. Этот факт можно наблюдать в первые 500 часов, когда мы можем увидеть более выступающие зубья пилы у приоритетных пользователей. Более того, этот факт проявляется в низких пиках – среди первых 500 часов – на рис.  [11a](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig11) , где достигается 35% использования, тогда как на рис.  [11i](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig11) загрузка ЦП снижается до 25%.

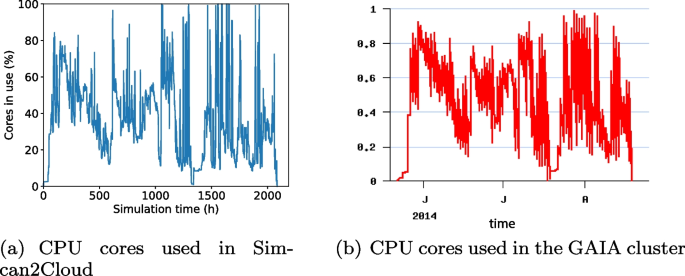
**Рис. 11**

[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/11)

Процент ядер ЦП, используемых для обработки реальной рабочей нагрузки

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/11)

**Рис. 12**

[](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/12)

Сравнение (кластер Simcan2Cloud и GAIA) с упором на процент использования ЦП.

[**Полноразмерное изображение**](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w/figures/12)

На рис.  [12](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig12) показано сравнение использования ЦП Simcan2Cloud (см. рис.  [12](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig12) а) и кластера GAIA (см. рис.  [12](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Fig12) б) для обработки трассировки. В этом эксперименте мы используем центр обработки данных с 640 физическими машинами, каждая из которых оснащена четырехъядерным процессором. Ось X этих диаграмм показывает время, прошедшее с момента начала выполнения системы до полной обработки трассировки. В целом мы отмечаем схожую форму пиков на обоих графиках. Например, Simcan2Cloud представляет те же низкие пики, которые показаны в поведении, полученном для реального кластера (см. низкие пики в часах 535, 1036 и 1470). Аналогичным образом, в моделируемом сценарии также представлены высокие пики (см. высокие пики в часы 307, 1256 и 1850).

Для понимания мы включили в раздел [Приложение](https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00511-w#Sec15) дополнительные графики , на которых показаны результаты анализа поведения дополнительных облачных конфигураций. Тем не менее, мы считаем, что результаты, включенные в раздел «Эмпирическое исследование», достаточно репрезентативны, чтобы сделать обоснованные выводы.

Выводы и будущая работа

В этой статье мы представили Simcan2Cloud, основанную на дискретных событиях среду для моделирования облачных систем. Simcan2Cloud в основном ориентирован на поставщика облачных услуг, поддерживая моделирование облачных инфраструктур и взаимодействие пользователей с облаком. Наш инструмент моделирования учитывает денежные затраты, развертывание в облаке, гибкую конфигурацию предлагаемых виртуальных машин и механизмы анализа моделируемых сред. Кроме того, Simcan2Cloud включает соглашения об уровне обслуживания для моделирования двух типов пользователей, а именно *обычных* пользователей и пользователей *с высоким приоритетом* , в зависимости от того, готовы ли они ждать необходимых им ресурсов или нет.

Было представлено полное тематическое исследование, показывающее моделирование и оценку различных облачных сценариев, а также проанализировано влияние рассматриваемых параметров. Основными рассматриваемыми параметрами были мощность процессора, обеспечиваемая физическими машинами, и размер облака, то есть количество физических машин. В этом исследовании были проанализированы несколько переменных, таких как количество обслуживаемых пользователей, среднее время ожидания, стандартное отклонение и минимальное время ожидания для всех пользователей.

Основная цель эмпирического исследования — оценить пригодность центров обработки данных, поддерживающих облако, для обработки рабочей нагрузки, уделяя особое внимание производительности и масштабируемости системы. После тщательного анализа полученных результатов мы пришли к выводу, что количество машин и процессоров, используемых в дата-центре, напрямую влияют на общую производительность системы. Результаты ясно демонстрируют ситуации, когда центр обработки данных становится перегруженным, что приводит к тому, что значительный процент пользователей оказывается не в состоянии обслуживаться. Еще один интересный параметр — количество зарезервированных машин (RM). Результаты ясно показывают, что для достижения наилучших результатов этот параметр необходимо правильно настроить в соответствии с размером центра обработки данных. Кроме того, мы воспроизвели трассировку, полученную из кластера GAIA, реального готового к эксплуатации кластера, расположенного в Университете Люксембурга. Результаты показывают аналогичную тенденцию производительности по сравнению с результатами, полученными на симуляторе.

Чтобы решить задачу проверки новой платформы моделирования, мы тщательно разработали эксперименты для изучения широкого спектра облачных конфигураций. Изменяя параметры, связанные с центром обработки данных, виртуализированными ресурсами и неоднородностью пользователей, наши эксперименты дают ценную информацию о поведении системы в различных сценариях. Кроме того, мы успешно провели эксперимент, в котором Simcan2Cloud точно воспроизводит трассировку кластера GAIA, демонстрируя аналогичные тенденции производительности. В нашем стремлении к эффективному подходу к проверке мы планируем в качестве будущей работы изучить интеграцию статистических методов и методов метаморфического тестирования (MT). Это включает в себя разработку метаморфических отношений (MR) для таких важных модулей, как генератор пользователей, облачный провайдер и центр обработки данных и других. Эти MR отражают основные свойства тестируемой системы, позволяя нам идентифицировать сценарии, которые отклоняются от ожидаемого поведения из-за невыполнения MR. Такой подход имеет большие перспективы для обеспечения надежности и точности нашей платформы моделирования.

В будущем мы планируем расширить Simcan2Cloud для поддержки одной из последних актуальных тем в области вычислений: туманных вычислений. С этой целью мы планируем включить новые модули для поддержки многоуровневого распределения компонентов, таких как устройства IoT, туманные устройства, мониторинг инфраструктуры и приложения IoT. Что касается устройств Интернета вещей, мы планируем включить датчики и исполнительные механизмы. Эти устройства позволяют создавать самые разные *вещи* , такие как мониторы очага, носимые устройства, датчики окружающей среды и камеры, среди прочего. Туманные устройства ориентированы на устранение разрыва между устройствами Интернета вещей и поставщиком облачных услуг. Таким образом, задержка связи будет значительно сокращена, что является одной из наших тем интересов. Более того, мы планируем весьма детально отслеживать базовую инфраструктуру системы для анализа ее масштабируемости. Кроме того, будут предоставлены модели приложений IoT, позволяющие выполнять приложения на устройствах IoT.

Кроме того, мы планируем расширить спектр возможных облачных конфигураций, увеличив количество и мощность ЦП физических машин, а также диапазон конфигураций оборудования, например дискового пространства хостов. Мы также намерены проанализировать влияние затрат на ресурсы на доходы облачного провайдера.

Доступность данных и материалов

Данные и материалы доступны у соответствующего автора по обоснованному запросу. Исходный код доступен на GitHub: <https://github.com/PabloCCanizares/Simcan2Cloud> .

Примечания

1. https://www.cs.huji.ac.il/labs/parallel/workload/l\_unilu\_gaia/index.html

Сокращения

***API:***

Интерфейс прикладного программирования

***АВС:***

Веб-сервисы Amazon

***ПРОЦЕССОР:***

Центральное процессорное устройство

***ЭК2:***

Эластичное вычислительное облако

***ГБ:***

Гигабайт

***Гбит/с:***

Гигабиты в секунду

***Графический интерфейс:***

Графический интерфейс пользователя

***Ввод/вывод:***

Ввод, вывод

***МБ:***

Мегабайт

***Мбит/с:***

Мегабит в секунду

***МИПС:***

Миллионы инструкций в секунду

***Что:***

Миллионы инструкций

***МОГ:***

Приоритетные пользователи

***БАРАН:***

Оперативная память

***РМ:***

Зарезервированные машины

***Соглашения об уровне обслуживания:***

Соглашение об уровне обслуживания

***Под:***

Уровень развития

***ПТС:***

Протокол управления передачей

***UDP:***

Протокол пользовательских датаграмм

***ВМ:***

Виртуальная машина