УДК 004.414.23, 519.876.5, 519.876.2

ПАКЕТЫ МОДЕЛИРОВАНИЯ ОБЛАЧНЫХ ИНФРАСТРУКТУР

Кореньков Владимир Васильевич[1], Муравьев Анатолий Николаевич[2], Нечаевский Андрей Васильевич[3]

[1]Доктор технических наук, профессор Института системного анализа и управления;
ГБОУ ВПО «Международный Университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
Директор Лаборатории информационных технологий (ЛИТ, ОИЯИ),
Объединенный институт ядерных исследований, Лаборатория информационных технологий;
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, 6;
e-mail: korenkov@cv.jinr.ru.

[2]Ассистент Института системного анализа и управления;
ГБОУ ВПО «Международный Университет природы, общества и человека «Дубна»;
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: a.n.muravev@gmail.

[3]Инженер-программист;
Объединенный институт ядерных исследований;
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, 6;
e-mail: nechav@mail.ru.

В работе представлено краткое описание пакетов моделирования облачных инфраструктур: CloudSim, iCanCloud, CReST. Описываются их архитектурные особенности, функциональность пакетов, выходные результаты симуляции модели. Рассматриваются достоинства и недостатки систем, представлены результаты сравнительного анализа пакетов по их функциональным возможностям.

Ключевые слова: облачные технологии, моделирование, облачные сервисы, политики предоставления ресурсов, платформа как сервис, инфраструктура как сервис, программное обеспечение как сервис.

CLOUD COMPUTING SIMULATION PACKAGES

Korenkov Vldimir[1], Muravyev Anatoly[2], Nechaevsky Andrey[3]

[1]Doctor of Technical science, professor;
Dubna International University of Nature, Society, and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19.
Deputy Director of the Laboratory;
Joint institute for nuclear researches, Laboratory of Information Technologies;
141980, Moscow reg., Dubna, Joliot-Curie, 6;
e-mail: korenkov@cv.jinr.ru.

[2]Assistant;
Dubna International University of Nature, Society and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: a.n.muravev@gmail.com.

[3]Software Engineer;
Joint institute for nuclear researches,
Laboratory of Information Technologies;
141980, Moscow reg., Dubna, Joliot-Curie, 6;
e-mail: nechav@mail.ru.

The paper introduces an overview of packages for modeling of cloud infrastructures: CloudSim, iCanCloud, CReST. The paper contains the description of architectural features, functionality of packages, and deliverables of mode's simulation. There is also the analysis of systems' advantages and disadvantages; and the results of comparative analysis of packages according to their functional capabilities have been presented.

Keywords: cloud computing, modeling, cloud services, policy provisioning, platform as a service, infrastructure as a service, software as a service.

Введение

Облачные вычисления – это новая парадигма, в которой динамически масштабируемые виртуализированные вычислительные ресурсы предоставляются как сервис через Интернет. Облачные вычисления могут быть определены как «тип параллельных и распределенных систем, состоящих из набора взаимосвязанных и виртуальных компьютеров, которые предоставлены динамически и представлены как один или несколько объединенных вычислительных ресурсов на основе соглашения об уровне обслуживания согласованное через договор между провайдером сервиса и потребителем» [4]. Примером инфраструктур облачных вычислений являются *Microsoft Azure* [1], *Amazon EC*2, *Google App Engine* и *Aneka* [2].

Облачные вычисления реализованы как сервисы, которые представляют инфраструктуру, платформу и программное обеспечение. Такие сервисы получили названия *IaaS* (*Infrastructure as a Service*), *PaaS* (*Platform as a Service*), *SaaS* (*Software as a Service*), соответственно. Доступ к сервисам осуществляется на основе подписки, и оплата происходит только за те ресурсы, которые используются.

Одно из свойств облачных платформ – это способность динамически адаптировать объем предоставляемых ресурсов для приложений в порядке изменения спроса, которые либо предсказуемы и происходят в соответствии с доступным шаблоном, наблюдаемые в течении определенного промежутка времени, либо возникающие из-за острого увеличения популярности приложения. Такая способность облаков особенно полезна для гибких приложений, таких как веб-хостинг, доставка контента, и социальные сети. Эти приложения часто демонстрируют кратковременное поведение (использование шаблона) и имеют разные требования *QoS* в зависимости от времени критического состояния и шаблонов взаимодействий пользователей (onli*ne*/*offline*). Таким образом, требуется развитие методов динамической поставки, чтобы гарантировать этим приложениям достигать *QoS* на переходных процессах.

Даже если Облако всё чаще рассматривается как платформа, которая может поддерживать гибкое управление приложениями, оно стоит перед определенными ограничениями, имеющие отношения к основным проблемам, таким как масштабируемость и местоположение. В тех случаях, когда число запросов превышает количество активных подключений, приложение может пойти на компромисс на поставляемую полноту качества обслуживания пользователям. Одним из решений этой проблемы является объединение нескольких облаков как части федерации. И как следствие необходимо разработать следующее поколение методов динамического предоставления сервисов, которые могут получить преимущества из созданной архитектуры. Такой подход позволяет предоставлять приложения с помощью нескольких облаков, которые являются элементами федерации.

В отличие от других распространенных вычислительных технологий (грид, суперкомпьтеры, *volunteer computing*) в облачных вычислительных системах широко применяются технологии и инструменты виртуализации. Следовательно, по сравнению с грид, облака содержат дополнительный уровень (уровень виртуализации), который действует как среда выполнения, управления и распределения сервисов приложений. На этом уровне создаются виртуальные машины, которые являются программной системой, эмулирующей аппаратное обеспечение. Поэтому принципы построения грид-инфраструктуры не полностью охватывают процесс построения облачной инфраструктуры. Основными проблемами при создании и поддержки облака являются:

- политика распределения виртуальных машин (далее ВМ) между хостами;
- управление выполнением приложения;

- стратегия динамического распределения нагрузок между дата-центрами;
- мониторинг динамического состояния системы.

Примерами систем и сервисов, использующие облачные инфраструктуры, могут быть социальные сети, системы веб-хостинга и системы анализа данных в режиме реального времени. Каждая из этих систем имеет различные структуры, конфигурации и требования к аппаратному и программному обеспечению. Проведение количественной оценки производительности систем и политик предоставления услуг в облачной инфраструктуре является сложной задачей. Использование реальной инфраструктуры для сравнительного анализа производительности и воспроизведения результатов, на которые можно положится, является нецелесообразным. Кроме этого, при использовании сторонних провайдеров облачных инфраструктур, сводит к минимуму контроль над инфраструктурой, и как следствие, к невозможности проведение экспериментов в повторяемой, надежной и масштабируемой среде.

Yahoo [7] и *HP* [8] представили глобальную тестовую базу Облачных вычислений, называемую *Open Cirrus*, которая поддерживает федерацию дата-центров, расположенных в 10 организациях [6]. Создание такого экспериментального окружения является дорогим и трудно реализуемым экспериментом, поскольку условия ресурса постоянно варьируются от времени к времени из-за совместного использования. Кроме того, их доступность ограничена членам их коллаборации.

Решением проблем оценки производительности облачной системы и политик предоставления сервисов облака, является использование инструментов моделирования и симуляции. При использовании таких инструментов, анализ работы инфраструктуры происходит в полностью управляемой среде. Это позволяет воспроизводить тесты с одинаковыми условиями, искать узкие места в системе до ее реализации в реальных облаках и проводить эксперименты с различными сочетаниями рабочих нагрузок и сценариями производительности ресурсов. Результаты такого анализа позволяют находить эффективные решения для построение облачной инфраструктуры, формировать адаптивные методы предоставления ресурсов, уменьшать стоимость и время развертывания инфраструктуры.

На сегодняшний день существует несколько систем моделирования облачных инфраструктур: *CloudSim* [9*], iCanCloud* [11*], CReST* [14]. Эти программные пакеты позволяют создавать модели облачных систем с определенной функциональностью и конфигурацией. Готовая модель запускается на симуляцию, в результате чего, системы моделирования предоставляют статистическую информацию по наиболее важным характеристикам: время выполнения задач, жизненный цикл виртуальных машин, использование ресурсов. Анализируя эту информацию, разработчик может выявить узкие места в модели и предусмотреть их решение, реализовав которое, можно проверить следующей итерацией симуляции.

1. Пакеты моделирования облачных инфраструктур

1.1 CloudSim

Система моделирования *CloudSim* представляет собой расширяемый набор инструментальных средств, который включает моделирование и симуляцию систем Облачных вычислений и предоставляемого окружения. Инструментарий *CloudSim* поддерживает как моделирование поведения компонентов облачной системы, таких как дата-центры, виртуальные машины, так и политику предоставления ресурсов. Он реализует универсальные методики предоставления, которые могут быть расширены с небольшими усилиями. В настоящее время, он поддерживает моделирование и симуляцию среды облачных вычислений входящую в состав одного или нескольких облаков (федерация облаков).

CloudSim реализована на языке *Java*, с использованием системы профайлинга *JProfiler*, что благоприятно сказывается на уменьшении риска утечек памяти и повышении стабильности программы при запуске нескольких потоков одновременно. Проект активно развивается, что хорошо видно по обновлениям системы в публичном репозитории. Для работы необходимо загрузить исходные коды системы *CloudSim* и создать проект в любой удобной интегрированной среде разработки (*IDE*). Создание модели производится через создание соответствующих сущностей в коде и заполнении их параметров. Симуляция созданной модели запускается путем компилирования, сборки и запуска непосредственно главного файла в проекте.

1.1.1 Архитектура CloudSim

На рисунке 1 изображена многоуровневая схема программной платформы *CloudSim* и архитектурные компоненты [9]. Первоначально *CloudSim* был основан на дискретном ядре симуляции событий *SimJava* [3], который успешно применен в платформе моделирования *DataGrid - GridSim*. Однако, ядро *SimJava* было заменено на другое ядро *CloudSim Core Simulation Framework*, для обеспечения таких функциональных возможностей как:

- Сбрасывание симуляции программно в режиме реального времени.
- Деактивация (удержание) объектов.
- Переключение контекста объектов между различными состояниями (например, ожидание активности).
- Поддержка создания новых сущностей в режиме реально времени и включения их в симуляцию.
- Уменьшение издержек производительности, при моделировании и симуляции системы большого размера.

Слой симуляции *CloudSim* предоставляет поддержку моделирования и симуляции виртуальной среды дата-центра, основанного на облаке, который включает специализированные интерфейсы управления виртуальными машинами, памятью, хранилищами, и полосой пропускания. Основные проблемы, такие как обеспечение ВМ хостами, управление выполнением приложений, и мониторинг динамического состояния системы, контролируются на этом слое. Провайдер облака, который хочет изучить эффективность различных политик распределения его хостов для ВМ, должен реализовывать его стратегии на этом уровне. Такое внедрение может быть сделано программно, расширяя ключевую функциональность предоставления ВМ. В этом слое есть четкое различие, связанное с обеспечением ВМ хостами. Облачный хост может одновременно располагать набор ВМ, которые выполняют приложения, основанные на *SaaS*, предоставляя определённый уровень *QoS*. Также, этот слой предоставляет функциональность, которую разработчики облачных приложений могут расширять для выполнения сложных рабочих нагрузок и исследования производительности приложения.

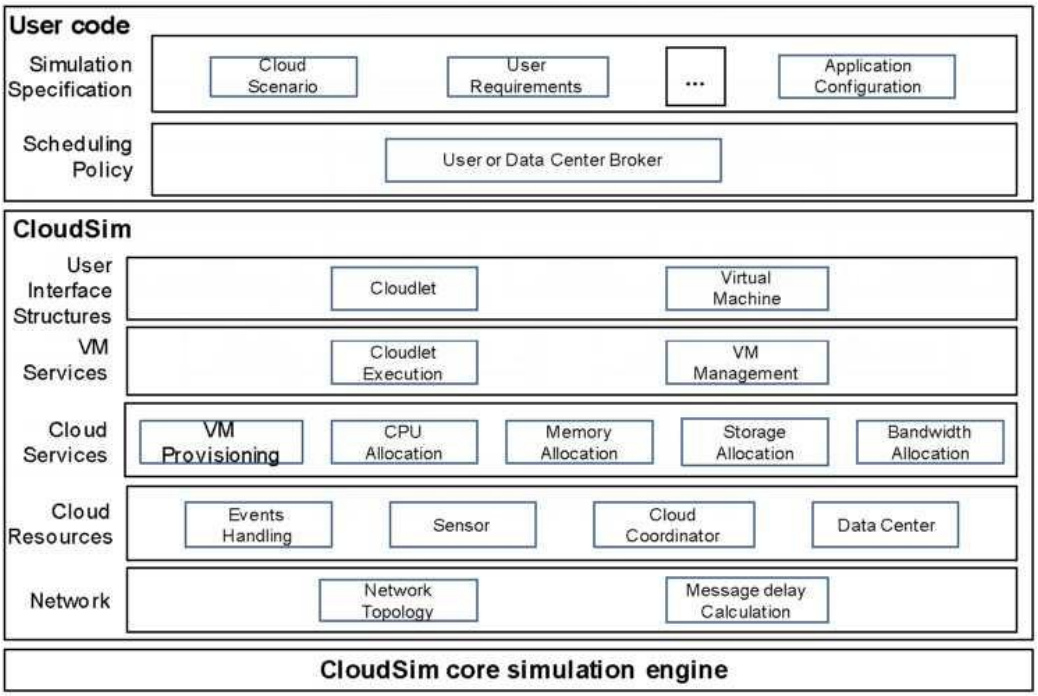

Рис. 1. Архитектура CloudSim

Самый верхний уровень в схеме архитектуры *CloudSim* – это пользовательский код, который предоставляет базовые сущности для хостов (число машин, их спецификация и т.д.), приложения (количество заданий и их требования), ВМ, количество пользователей и типы их приложений, и

брокер политик планирования. Расширяя эти базовые сущности, разработчик приложения облака может производить следующие действия:

- генерация сочетаний, требуемых распределенных рабочих нагрузок, конфигурацию приложений;
- для модели облака доступно сценарии и нагрузочные тесты, основанные на конфигурации пользователя;
- реализация пользовательских методологий предоставления приложений для облаков и их федерации.

1.1.2 Функциональность CloudSim

CloudSim позволяет моделировать [9]:

- основные уровни облака (*IaaS, PaaS, SaaS*);
- распределение виртуальных машин в рамках одного или нескольких вычислительных узлов;
- латентность сообщений;
- федерацию облаков;
- динамическую рабочую нагрузку;
- потребление энергии дата-центра;
- динамическое создание сущностей модели.

Модель облачной инфраструктуры в *CloudSim* приближена к реальной структуре облака. Основные сервисы уровня инфраструктуры (*IaaS*) моделируются на основе дата-центра. Дата-центр может управлять несколькими хостами, которые поочередно управляют виртуальным машинами во время их жизненного цикла. Под хостом здесь понимается физический вычислительный сервер в облаке, которому определены такие свойства как: вычислительная мощность (выражена в *MIPS*), объем оперативной памяти, объем постоянного хранилища. *CloudSim* позволяет моделировать политики распределения виртуальных машин на уровне хоста и задач на уровне виртуальных машин. Здесь, политика ВМ означает политику управления операциями, связанные с жизненным циклом ВМ, такими как: снабжение хостов ВМ'ами, создание ВМ, уничтожение ВМ, и миграции ВМ. Так пользователь может определить будут ли виртуальные машины работать параллельно на одном хосте (распределение виртуальных машин по времени) или последовательно в порядке очереди (распределение виртуальных машин по объему хоста). Аналогичным образом определяется процесс выполнения задач на уровне виртуальных машин. Задачи выполняются параллельно на одной ВМ, с учетом распределения по ядрам процессора виртуальной машины, или последовательно друг за другом.

Моделирование в *CloudSim* основано на событиях, где различные сущности связываются между собой посредством передачи сообщений при возникновении события. Каждое сообщение подвергается задержки при передачи от сущности к сущности. такие задержки в сети моделируются через матрицу задержек. Элементы этой матрицы представляют собой время в миллисекундах, которое должно пройти при передачи сообщения от одной сущности к другой. Описание топологии сохранено в формате *BRITE* [5], который содержит набор сетевых узлов, которые могут быть больше, чем набор моделируемых узлов. Эти узлы представляют различные сущности *CloudSim*, включая хосты, дата-центры, Облачные Брокеры и т.д. Назначение Облачного брокера заключается в создании облаков, мониторинге нагрузки на облачные ресурсы, мониторинге выполнения приложений в облаке, и экспорте облачных сервисов в федерацию облаков.

CloudSim поддерживает моделирование динамической рабочей нагрузки, которая может изменятся с течением времени. Лидирующие облачные поставщики, включая *Amazon и Azure*, представляют контейнеры/шаблоны виртуальных машин чтобы разместить ряд сервисов *SaaS* типа и предоставляют провайдерам SaaS понятие неограниченного пула ресурсов, который может быть арендован на лету с требуемыми конфигурациями. В *CloudSim* подобный механизм предоставляет методы и переменные для определения ресурсов и требований на уровне виртуальных машин приложений SaaS по требованию, что позволяет моделировать не только динамическую рабочую нагрузку, но и нештатные ситуации в работе дата-центра.

1.1.3 Результаты моделирования в CloudSim

В результате симуляции модели CloudSim предоставляет подробную статистическую информацию по жизненному циклу ВМ, загруженности ВМ, очереди отправленных, принятых и выполненных задач по времени. При моделировании федерации облаков, выводится информация о загруженности частного и публичного облаков. Добавляя в модель характеристики потребления энергии каждой единицей облачной инфраструктуры, можно получить данные об эффективности потребления электроэнергии. Дополнительную информацию всегда можно получить, добавив выгрузку данных в лог в интересующие сущности CloudSim. В итоге, полученных данных достаточно для анализа работы спроектированной модели и политик предоставления.

1.2 iCanCloud

iCanCloud – это ещё одна платформа для симуляции и моделирования облачных инфраструктур. Основной целью *iCanCloud* является нахождения компромиссного решения между стоимостью и производительностью для заданного набора приложений, выполняемых на конкретном аппаратном обеспечении, и предоставления пользователям полезную информацию о таких расходах. Кроме этого, *iCanCloud* может быть использован широким кругом пользователей, начиная от основных пользователей, переходящих на облачную инфраструктуру, до разработчиков больших распределенных приложений [11].

iCanCloud обладает графическим интерфейсом пользователя, который позволяет быстро и легко моделировать облачную инфраструктуру с доступными ресурсами, такими как ВМ, облачный гипервизор, приложения и аппаратную часть. Создание пользовательских ресурсов осуществляется добавлением собственных классов в системы *iCanCloud*, написанных на языке программирования *C++*.

1.2.1 Архитектура iCanCloud

iCanCloud разработана как компонент дискретной среды моделирования событий *OMNeT++*. Ее основная область применения является моделирование сетей связи. Сеть здесь обозначена в более широком смысле и представляет собой проводные и беспроводные сети связи, сети на чипе, сети массового обслуживания и т.д. [12]. Для моделирования сетей в *iCanCloud* используется симуляционный пакет *INET*, который также является компонентов *OMNeT++*. Этот пакет содержит в себе готовые модели некоторых проводных и беспроводных сетевых протоколов, таких как: *UDP*, *TCP*, *SCTP*, *IP*, *IPv6*, *Ethernet*, *PPP*, 802.11, *MPLS*, *OSPF* и другие [13].

На рисунке 2 отображена базовая многоуровневая схема архитектуры *iCanCloud*. Архитектура *iCanCloud* была разработана на базовой идеи систем облачных вычислений: предоставление пользователям псевдонастраиваемую аппаратную среду, где они могут запускать свои приложения.

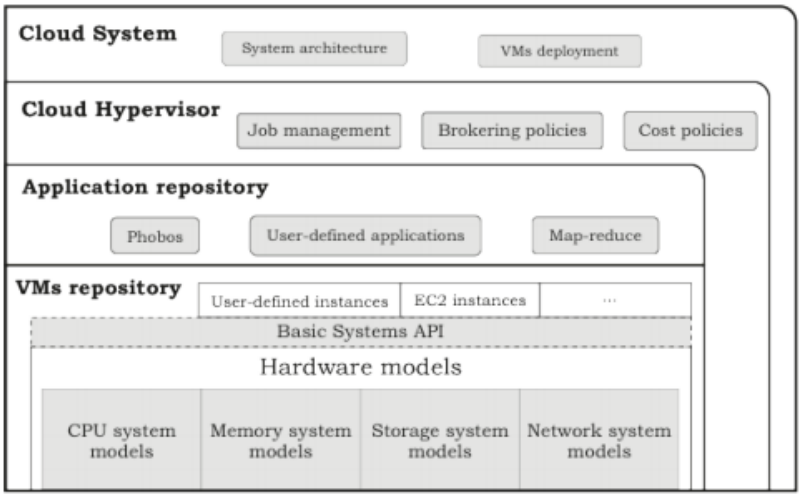

Рис. 2. Базовая многоуровневая схема архитектуры iCanCloud

Нижний уровень архитектуры *iCanCloud* состоит из моделей аппаратной части. На этом уровне моделируются аппаратные части системы: жесткие диски, модули памяти и центральные процессоры. В свою очередь, этот уровень разделен на четыре группы, каждая из которых соответствует определенной части системы: подсистема обработки (*CPU*), подсистема оперативной памяти, подсистема хранения данных и подсистема сети.

Базовый системный модуль API напрямую соединен с моделями уровня аппаратной части. В основном этот модуль содержит набор системных вызовов, которые представлены как *API* (интерфейс прикладного программирования), для всех приложений, выполняемых в виртуальных машинах с использованием *iCanCloud*. Таким образом, эти системные вызовы обеспечивают интерфейс между приложениями и сервисами, предоставляемые моделями аппаратной части. Кроме того, исследователи могут писать приложения для симуляции в *iCanCloud*, используя *API*. В целях поддержания совместимости, *API* разработано по стандартам *POSIX*.

На уровне выше находится хранилище ВМ. Это хранилище содержит в себе коллекцию ВМ, которую определяет пользователь. Изначально, симулятор предоставляет несколько моделей существующих ВМ в известных облаках *Amazon* (*EC*2). Пользователь может добавлять, удалять и редактировать ВМ в этом хранилище. Каждая ВМ моделируется путем настройки каждой соответствующей нижестоящей моделей аппаратной части.

Следующий уровень содержит в себе хранилище приложений. Аналогично хранилищу виртуальных машин, в этот репозиторий добавлен набор заранее определенных моделей приложений. Эти модели будут использоваться для того, чтобы настроить соответствующие задачи, которые будут выполнены в определенном экземпляре ВМ. Кроме того, пользователи могут легко добавлять собственные модели приложений используя *API*.

Уровень выше называется облачным гипервизором. Этот уровень состоит из модуля, отвечающего за управление всех поступающих задач и за создание ВМ для выполнения этих задач. После того, как работа над задачей закончена, этот модуль останавливает ту ВМ, на которой была выполнена работа по задаче, и повторно распределяет ресурсы системы для выполнения оставшихся задач. Этот модуль также содержит политики затрат для назначения входящим задачам на конкретных экземплярах ВМ, рассчитанные на соответствующих эвристических правилах.

На самом верхнем уровне располагается модуль облачной системы. Этот модуль содержит определение всей сущности облачной системы, которая в основном состоит из определения гипервизора, определения каждой ВМ, которая наполняет эту систему.

1.2.2 Функциональность iCanCloud

Исходя из архитектуры *iCanCloud* система обладает следующим функционалом:

- моделирование и симуляция существующих и не существующих облачных систем;
- простая интеграция и тестирования новых и готовых политик брокера ресурсов, путем гибкой настройки модуля гипервизора;
- настройка ВМ может быть использована для быстрого моделирования одноядерных и многоядерных систем
- настройка широкого набора конфигураций системы хранения данных позволяет моделировать локальные системам хранения, удаленные системы хранения, такие как NFS и параллельные системы хранения данных, такие *RAID* и параллельные файловые системы;
- графический интерфейс пользователя предоставляет простую генерацию и настройку больших распределенных моделей. Этот графический интерфейс нацелен прежде всего на управление и изучение ранее настроенных ВМ, облачных систем, гипервизоров, приложений и для получения графического отчета;
- предоставленный *POSIX* совместимый *API* и адаптированная библиотека *MPI* позволяет моделировать и проводить симуляцию приложений. Также используя функциональность моделирования приложений в *iCanCloud*, приложения можно реализовывать и моделировать на основе статических графов, используя трассировку реальных приложений и программирование новых приложений напрямую в платформе моделирования.

В базовом комплекте симуляционного пакета *iCanCloud* уже реализованы некоторые сущности для быстрого создания модели облачной инфраструктуры. Создания модели может быть реализовано

с применением готовых сущностей как снизу-вверх по уровням архитектуры, так и сверху вниз. Таким образом, *iCanCloud* позволяет оперировать готовыми облаками, на примере частного облака, распределенных виртуальных машин, а также известного облачного сервиса *Amazon_EC*2 [14].

1.2.3 Результаты моделирования в iCanCloud

Благодаря графическому интерфейсу, в результате моделирования в *iCanCloud* пользователь получает наглядную структуру спроектированной облачной системы. Используя компонент *INET* для моделирования сети физических и виртуальных вычислительных узлов, пользователь получает структуру сети. После симуляции модели облачной системы пользователю доступны статистические данные как в графическом, так и в файле формата *.csv*. В последнем отображены точки времени выполнения приложений, освобождение ресурсов и жизненный цикл ВМ. Информация в таком виде позволяет создавать необходимые графики и диаграммы для дальнейшего анализа модели.

1.3 CReST

CReST – инструментарий, разработанный для моделирования облачного обеспечения, на основе дискретно-событийной симуляции моделей [15]. Причиной создания *CReST* послужила необходимость использования робастной системы моделирования для исследования и изучения методик управления дата-центрами и предоставления облачных сервисов. Инструментарий *CReST* стремится выйти на функциональность, которая позволяет в короткие сроки и с минимальными трудозатратами выявить проблемы облачной инфраструктуры и указать на возможные ошибки в политиках предоставления и управления дата-центрами. Примером таких ошибок является так называемая «*Leap Day Bug*», которая 29 февраля 2012 года стала причиной отключения функциональности по управлению всеми кластерами *Azure* компании *Microsoft* во всем мире больше чем на 10 часов [16].

CReST представляет собой автономную систему, написанную на языке программирования *Java* и является свободно распространяемой. В отличие от других существующих систем моделирования облачных инфраструктур, *CReST* позволяет запускать симуляцию нескольких абстрактных уровней, начиная от физического уровня, энергопотреблением и тепловыми потоками в пределах дата-центра, до сетевой инфраструктуры и уровня виртуализации облачных сервисов, выполняющиеся по запросу пользователя.

1.3.1 Архитектура CReST

Система *CReST* спроектирована как набор связанных модулей, которые могут быть независимо отключены или включены в зависимости от требований к модели. Схема архитектуры *CReST* представлена на рисунке 3. Входными данными для запуска *CReST* является конфигурационный файл формата *XML*. В этом файле содержится полная спецификация аппаратной части каждого дата-центра.*CReST* «*Builder*» - это графическое приложение для созданий и редактирования таких конфигурационных файлов. Также система может читать необязательный файл «*Parameters*», в котором пользователь указывает вручную интересующие его параметры. Те в свою очередь перезаписывают соответствующие параметры в *XML*-файле. Такой подход позволяет быстро изменять конфигурационный файл или запускать множественную симуляцию с различными конфигурационными параметрами. Дополнительно пользователи могут задавать их собственные события через текстовый файл «*User Events*», в которых определены тип события и время события. *CReST* обладает графическим интерфейсом. Во время симуляции в режиме реального времени пользователь может следить за сбоями, нагрузкой серверов, тепловым потоком на карте дата-центра через графический интерфейс.

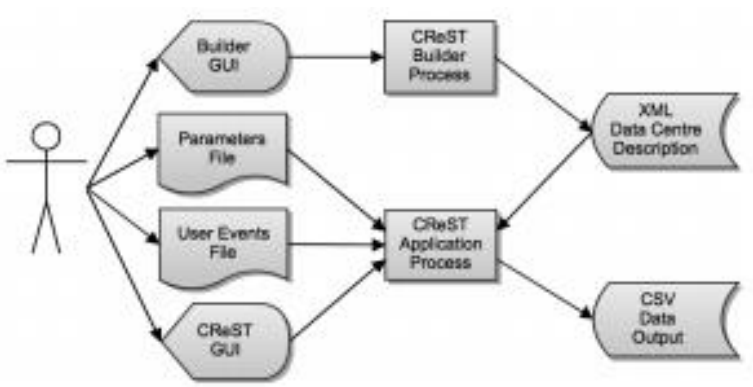

Рис. 3. Схема архитектуры CReST

Для обеспечения расширяемости и модульной независимости *CReST* реализована на базе паттерна проектирования *Model-View-Controller*. Каждый независимый модуль имеет абстрактный метод *ModuleRunner*, который просматривает очередь событий через интерфейс *Java Observer-Observable*. Модули, извлекая события из очереди событий *EventQueue*, просматривают их и делают игнорируют их или принимают на исполнение, что в свою очередь может привести к генерированию новых событий и помещению их в очередь событий. Таким образом модули – это независимые наблюдатели очереди событий, которые взаимодействуют между собой через события. Такой подход позволяет включать и выключать модули, добавлять модули и быстро их регистрировать в системе.

Каждая симуляция начинается созданием *World object*, который содержит в себе хотя бы один или несколько объектов дата-центров. Каждый дата-центр в свою очередь содержит список абстрактных блочных объектов. Блоки реализованы в четырех конкретных типах: *Aisle, Container, AirCon* и *Rack*. *Aisle* и *Container*, содержит в себе список объектов *Rack*. В свою очередь *Rack* содержит список объектов *Server*. *Serve*r и *AirCon* реализуют интерфейс сбоев *Failable*. Сервер содержит в себе жесткий диск, оперативную память, программное обеспечение и хотя бы один центральный процессор. Server может запускать сервисы (*Service*) и виртуальные машины (*VirtualMachine*), которые запускаются и останавливаются через методы *start*() и *stop*().

Все обновления статуса в *World object* инициализируются объектами *Event*, созданные каждым *ModelRunner*. События кладутся в очередь событий *EventQueue* и сортируются по времени. Когда событие извлекается из очереди, оно выполняется и имеет возможность создавать новые события, которые также кладутся в очередь. Как только событие получает статусы «выполненный» или «сгенерированный», оно просматривается каждым объектом *ModuleRunner*. Если *ModuleRunner* заинтересован в событии, тогда он выполняет соответствующее действие, в ином случае событие игнорируется.

1.3.2 Функциональность CReST

На данный момент в *CReST* реализованы следующие модули с определенным набором функций [15]:

- *Thermal* – тепловой модуль. Этот модуль моделирует генерацию тепла, распределение и распространение в пределах дата-центра. Модуль реализует свой графический интерфейс, на котором отображено распространение тепла на вычислительных узлах в режиме реального времени.
- *Energy* – энергетический модуль моделирует потребление энергии дата-центром в целом и каждым вычислительным узлом по отдельности.
- *Failures* – моделирует постоянные и временные отказы работы аппаратной части.
- *Replacements* – модуль позволяет заменять аппаратные части на другие с новыми техническими характеристиками.
- *Subscriptions* – моделирует сетевые подписки
- *Services* – модуль моделирует планировщика запуска виртуальных машин и их расположение в дата-центре
- *Pricing* – проводит оценку расходов на операции и цен за услуги.

- *Demand* – моделирует требования пользователя и брокера ресурсов в облаке.
- *UserEvents* – задает определенные пользователем события через текстовый файл.
- *GUI* – графический интерфейс пользователя отображает результаты симуляции модели в режиме реального времени.

Таким образом разработчикам облачной инфраструктуры предоставлен широкий функционал по моделированию аппаратной части и ее поведение при различных вариантах использования. Благодаря своей модульности, разработчик, используя программный интерфейс в *CReST*, имеет возможность с минимальными трудозатратами по времени и ресурсам реализовать собственный модуль, который будет интегрирован в систему автоматически.

1.3.3 Результаты моделирования в CReST

Пользователю доступен графический интерфейс, на котором отображается карта дата-центра и есть возможность наблюдения в режиме реального времени за сбоями в аппаратной части, нагрузкой серверов и потоками тепла. Расширяя существующие модули, пользователь может быстро реализовать графический интерфейс для наблюдения изменения данных, которые его интересуют. Кроме этого, все симуляционные данные записываются в *CSV*-файл - один для одного включенного модуля. Анализируя этот файл, разработчик может выявить слабые места в модели и откорректировать их. А возможность чтения пользовательского файла параметров позволяет делать правки в кратчайшие сроки и повторно запускать всю цепочку моделирования.

2. Сравнительный анализ пакетов

2.1 Критерии сравнительного анализа

Для проведения сравнительного анализа пакетов моделирования облачных инфраструктур необходимо определить ряд критериев. Текущий анализ описанных пакетов и существующих облачных инфраструктур, таких как *Microsoft Azure* [1], *Amazon EC2, Google App Engine*, и *Aneka* [2], определил следующих набор критериев для сравнительного анализа:

- Моделирование аппаратной части (уровень *PaaS*). Задание и учет технических характеристик вычислительных узлов, анализ энергопотребления, анализ тепловыделения, моделирование сбоев вычислительных узлов.
- Моделирование уровня виртуализации (уровень *IaaS*). Задание и учет технических характеристик ВМ, моделирование политик предоставления ресурсов, моделирование политик распределения задач по ВМ, миграция ВМ в рамках облака.
- Моделирование поведения программного обеспечения в облаке (уровень *SaaS*). Задание и учет сложности выполняемой задачи, симуляция реальных приложений.
- Моделирование сети. Моделирование топологии сети, моделирование задержек между объектами, моделирование различных протоколов передачи данных.
- Моделирование федерации облаков (частные, публичные, гибридные).
- Моделирование расходов на использование облачных ресурсов.
- Наличие графического интерфейса.
- Техническая поддержка.
- Тип лицензии / Открытый код.

2.2. Сравнительный анализ систем моделирования

В таблице 1 представлен список критериев и их реализация в рассматриваемых пакетах.

Таблица 1. Сравнительная таблица результатов анализа пакетов моделирования

Критерий	*CloudSim*	*iCanCloud*	*CReST*
Моделирование аппаратной части (уровень PaaS)			
Задание и учет технических характеристик вычислительных узлов	Да	Да	Да
Анализ энергопотребления	Да	Нет	Да
Анализ тепловыделения	Нет	Нет	Да
Моделирование сбоев вычислительных узлов	Да	Нет	Да
Моделирование уровня виртуализации (уровень IaaS)			
Задание и учет технических характеристик ВМ	Да	Да	Да
Моделирование политик предоставления ресурсов	Да	Да	Нет
Моделирование политик распределения задач по ВМ	Да	Да	Нет
Миграция ВМ в рамках облака	Да	Нет	Нет
Моделирование поведения программного обеспечения в облаке (уровень SaaS)			
Задание и учет сложности выполняемой задачи	Да	Да	Да
Симуляция реальных приложений	Нет	Да	Нет
Моделирование сети			
Моделирование топологии сети	Да	Да	Да
Моделирование задержек между объектами	Да	Да	Да
Другие критерии			
Моделирование различных протоколов передачи данных	Нет	Да	Да
Моделирование федерации облаков (частные, публичные, гибридные)	Да	Нет	Нет
Моделирование ценовых политик и учет использования облачных ресурсов	Нет	Да	Нет
Наличие графического интерфейса	Нет	Да	Да
Техническая поддержка	Да	Нет	Да
Тип лицензии / Открытый код	*GNU*/Да	*GNU*/Да	*GNU*/Да

Основной функциональностью системы *CloudSim*, которая выделяет ее среди остальных, являются: моделирование собственных политик распределения ВМ по дата-центру и задач по ВМ, моделирование федерации облаков и связки между частными и публичными облаками, моделирование динамической нагрузки с учетом отказа и сбоях в аппаратной части дата-центра. Также, стоит отметить, наличие большого числа реализованных проектов, которые позволяют провести моделирование интересующей области облачной инфраструктуры. К недостаткам системы относятся: отсутствие

графического интерфейса и, как следствие, знания языка программирования *Java* как инструмента для моделирования, а также сложность оценки трудоемкости реальных приложений в формате *MIPS*.

Мощным средством моделирования работы приложений в *iCanCloud* является базовый системный модуль с набором системных вызовов, разработанных по стандарту *POSIX*. Это позволяет внедрять в модель облачной инфраструктуры существующие приложения и анализировать их работу в облаке. Такой подход снижает накладные расходы на подготовку задач для моделирования, и, как следствие, уменьшает время моделирования в целом. Благодаря среде моделирования событий *OMNeT++* и симуляционному пакету *INET*, на которых основан *iCanCloud*, пользователь имеет большой набор функциональности по моделированию существующих протоколов передачи данных (*UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP,* 802.11, *MPLS, OSPF*) и системы хранения данных (локальные системам хранения, удаленные системы хранения, *RAID*). Графический интерфейс уменьшает сложность процесса моделирования и запуска на симуляцию, а также облегчает анализ результатов симуляции. Основным достоинством *iCanCloud* перед представленными системами моделирования облачных инфраструктур является наличие модуля ценовой политики, которая позволяет устанавливать цену на использование ресурсов облака и проводить их учет. Однако, в *iCanCloud* нет такого же широкого набора функциональности для моделирования готовых политик предоставления ВМ и распределения задач по ним, как для протоколов передачи данных или системы хранения.

В пакете симуляции *CReST* наиболее полно реализован функционал по моделированию аппаратной части. При сравнении с CloudSim, анализ энергопотребления в *CReST* реализован лучше. Этому способствует и наличие графического интерфейса, позволяющий в режиме реального времени наблюдать за изменениями параметров в дата-центре. Функциональность по учету выделения тепла отсутствует в других пакетах, представленных в данной статье. Но *CReST*, как и *iCanCloud* не позволяет моделировать политики предоставления ВМ и распределения задач, что сужает круг область использования данной системы.

Стоит отметить, что системы моделирования *CloudSim* и *CReST* активно развиваются. На базе *CloudSim* реализованы узконаправленные проекты такие как: *CloudSimEx* [17], *WorkflowSim* [18], *DynamicCloudSim* [19] и другие. Последние изменения в *CloudSim* были сохранены в публичном репозитории проекта 02.05.2013 г.

Стабильная версия *CReST* появилась относительно недавно – 23.03.2014, поэтому дополнительные проекты на базе этой системы еще не реализованы. При этом обновления системы производится каждые 5-6 месяцев.

Неизвестно текущее состояние *iCanCloud*, т.к. последнее обновление на публичном репозитории было 30.11.2011. На сайте разработчиков и в публичном репозитории нет информации о дальнейшем развитии проекта. Поэтому анализ и устранение возможных ошибок в работе программе необходимо проводить собственными силами.

Выводы

В настоящее время требования к функциональности программного обеспечения изменяются крайне быстро. Для адаптации к современным тенденциям программные пакеты должны обладать гибкой настройкой и модульностью. Тогда это позволяет им изменять и расширять свой функционал в зависимости от новых запросов.

Каждый из рассмотренных в данной статье программных пакетов моделирования и симуляции облачных инфраструктур обладают механизмом добавления и удаления программных модулей. Таким образом, использование этих систем при исследование облачных инфраструктур оправдано с точки зрения возможного расширения функционала для поставленной задачи.

Как видно из таблицы 1, ни одна из представленных систем моделирования не охватывает полного списка функциональности, который позволяет моделировать облачную инфраструктуру достаточно подробно на всех возможных уровнях (*PaaS, IaaS, SaaS*). Каждая система ориентирована на определенную область исследования, в той или иной мере затрагивая другие области для формирования более подробной модели.

Несомненно, для моделирования инфраструктур на основе облачных технологий при исследовании новых политик предоставления виртуальных машин, политик распределения задач по виртуаль-

ным машинам, поведения облачной инфраструктуры при динамической нагрузке на дата-центр наиболее подходит *CloudSim*. Для быстрого анализа при переводе пользовательского приложения в облако и оценки расходов за использование облачных ресурсов стоит воспользоваться пакетом моделирования *iCanCloud*. Анализ работы облака, с точки зрения использования аппаратной части, сбоев работы дата-центра, потребления электроэнергии и выделения тепла, наиболее эффективно проводить в системе *CReST*.

Заключение

В статье рассмотрены программные пакеты моделирования облачных инфраструктур *CloudSim*, *iCanCloud* и *CReST*. Каждый из представлен пакетов ориентирован на моделирование определенного уровня облака. Функциональность *CloudSim* позволяет наиболее подробно моделировать уровни *SaaS* и *IaaS*. Для анализа работы уровней *PaaS* и *SaaS* облачной инфраструктуры необходимо использовать *iCanCloud*. Разработку дата-центра с минимальными затратами электроэнергии и эффективным охлаждением можно реализовать в *CReST* который подробно моделирует *PaaS* уровень.

Представленные системы моделирования рассчитаны на решение своих узкоспециализированных задач. Пакеты моделирования CloudSim, iCanCloud и CReST не обладают набором функционала для полноценного моделирования облачных вычислительных центров. При использовании данных пакетов для решения своих задачах необходимо проводить разработку дополнительных модулей.

Список литературы

1. Chappell D. Introducing the Azure services platform // White paper, Oct. 2008.
2. Vecchiola C., Chu X., Buyya R., Aneka: A Software Platform for .NET-based Cloud Computing, High Speed and Large Scale Scientific Computing / W. Gentzsch, L. Grandinetti, G. Joubert (Eds.) // ISBN: 978-1-60750¬073-5, IOS Press. – Amsterdam, Netherlands, 2009.
3. Howell F., Mcnab R. SimJava: A discrete event simulation library for java // Proceedings of the first International Conference on Web-Based Modeling and Simulation, 1998.
4. Buyya R., Yeo C. S., Venugopal S., Broberg J., Brandic I. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility // Future Generation Computer Systems. – June 2009. – Vol. 25. – №6. – Pp. 599-616.
5. Medina A., Lakhina A., Matta I., Byer J. BRITE: An Approach to Universal Topology Generation // Proceedings of the 9th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2001). – Cincinnati, OH, USA, 15-18 August 2001.
6. Avetisyan A. et. al. Open Cirrus: A Global Cloud Computing Testbed // IEEE Computer, April 2010.
7. Yahoo Homepage. – [Electronic resource]. URL: https://www.yahoo.com (дата обращения: 20.05.2014).
8. HP Homepage. – [Electronic resource]. URL: https://www.hp.com (дата обращения: 20.05.2014).
9. Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms // Software: Practice and Experience. – January 2011. – Vol. 41. – №1. – Pp. 23-50.
10. CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services. – [Electronic resource]. URL: http://www.cloudbus.org/cloudsim (дата обращения: 20.05.2014).
11. iCanCloud Homepage [Electronic resource]. – URL: http://www.arcos.inf.uc3m.es/~icancloud/Home.html (дата обращения: 20.05.2014).
12. OMNeT++ Community Homepage. – [Electronic resource]. URL: http://www.omnetpp.org (дата обращения: 20.05.2014).

13. INET Framework Homepage. – [Electronic resource]. URL: http://inet.omnetpp.org (дата обращения: 20.05.2014).
14. Amazon Elastic Compute Cloud Homepage [Electronic resource]. – URL: http://aws.amazon.com/ec2/ (дата обращения: 20.05.2014).
15. J. Cartlidge & D. Cliff, Comparison of Cloud Middleware Protocols and Subscription Network Topologies using CReST, the Cloud Research Simulation Toolkit // Proceedings 3rd Int. Conf. Cloud Computing and Services Science. – Germany: SciTePress, May 2013. – Pp. 58-68.
16. Laing, B. Summary of Windows Azure service disruption on Feb 29th, 2012 // MSDN Windows Azure Team Blog, March 2012. – [Electronic resource]. URL: http://bit.ly/AfdqyL (дата обращения: 20.05.2014).
17. CloudSimEx Documentation. – [Electronic resource]. URL: http://nikolaygrozev.wordpress.com/category/cloudsimex (дата обращения: 20.05.2014).
18. WorkflowSim Homepage. – [Electronic resource]. URL: http://www.workflowsim.org (дата обращения: 20.05.2014).
19. Bux M., Leser U. DynamicCloudSim: Simulating Heterogeneity in Computational Clouds // Int. Workshop on Scalable Workflow Enactment Engines and Technologie, in conjunction with ACM SIGMOD Conference. – New York, USA.