Обзор статей по направлению «Симуляторы облака»

Тагир Кускаров, БПМИ176

Содержание

1 Введение

Облачные вычисления сегодня стали крайне популярны по ряду общеизвестных причин — из-за выгоды в цене, простоты использования, надежности и некоторых других. Отсюда возникает спрос на развитие технологий в данной области. Многие исследователи в мире занимаются разработкой новых подходов к построению облаков, а именно в части организации инфраструктуры и создания нового программного обеспечения.

Для тестирования новых разработок необходима система, которая обеспечивает воспроизводимость экспериментов, полный контроль за своим состоянием и, желательно, невысокую стоимость. Тестирование на реальных облачных системах не вполне отвечает данным требованиям — даже если речь идет о внутренних разработках компании-провайдера, то обеспечить воспроизводимость в системе, использующей физическую сеть и серверы, полностью невозможно. Для исследователей, не обладающих доступом к ресурсам, задача еще более осложняется — стоимость тестирования при высокой нагрузке или на больших кластерах может быть крайне высокой.

Для подобных задач разработаны *симуляторы облаков*, которые позволяют моделировать работу физического кластера и ПО поверх него и на этой основе запускать новые алгоритмы. Задача симулятора — максимально точно приблизить условия работы в физическом мире.

2 Основные задачи симулятора

В чём заключается воссоздание условий реального мира в симуляторе? Во-первых, это включает моделирование физических сущностей — серверов, серверных стоек, сети и их возможных отказов. Во-вторых, необходимо эмулировать запуск пользовательских приложений в облачной парадигме — например, с автоматическим масштабированием и динамической реконфигурацией. В частности, это ограничивает возможность использования уже написанных симуляторов grid-вычислений.

Возможны реализации в симуляторе различных моделей обслуживания — это имеет значение, так как чем ближе модель к SaaS, тем больше на стороне разработчика облака остаётся контроля, а значит и возможностей для оптимизации.

Авторам симулятора нужно продумать и описать представление приложения — его характеристику с точки зрения ресурсов (CPU, RAM и др.) и требования (SLA), которые на практике могут отличаться, так как зависят от тарифа. Дополнительным плюсом симулятора будет являться возможность эмуляции кластеров разных размеров — как будет видно далее, на такое способны далеко не все симуляторы, так как часто они зависят от асимптотики используемых алгоритмов оптимизации.

Наконец, симулятор должен обладать понятным API, мощной системой мониторинга и анализа результатов для удобства проведения и воспроизведения экспериментов. Многие из изученных систем обладают графическим интерфейсом, что является плюсом в данном пункте.

3 Структура обзора

В данном обзоре рассмотрены несколько симуляторов — CloudSim [1] как наиболее известный, основанный на нём ElacticSim [2], а также симулятор под названием SCORE [3].

Описание каждого симулятора включает обзор архитектуры, API и возможностей для настройки. После описания будет приведён сравнительный анализ разобранных систем и собственная оценка автора.

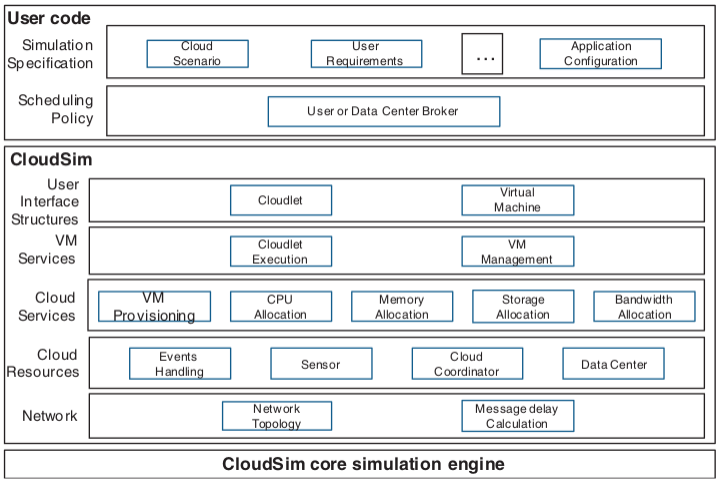
4 Описание симуляторов

4.1 CloudSim

Данный симулятор впервые был описан в 2010 году и является, по впечатлению автора, наиболее известным. Статья о нём имеет более 2000 цитирований. На базе CloudSim создано более 10 других систем, которые так или иначе дополняют его функционал.

Система имеет многоуровневую архитектуру. В основе всего лежит *core simulation engine* — модуль, который отвечает за саму симуляцию. В CloudSim основным примитивом является событие — ими обмениваются дата-центры, серверы и виртуальные машины. Simulation engine хранит и обрабатывает очереди событий, генерируемых другими компонентами.

Над simulation engine расположен модуль с основной логикой CloudSim — классы, описывающие физические объекты (дата-центры, серверы, VM), сеть и правила их взаимодействия. Сеть моделируется простым способом — с помощью *latency matrix* (матрицы, где для каждой пары объектов указана задержка сети), а топология в виде маршрутизаторов и свичей отсутствует.

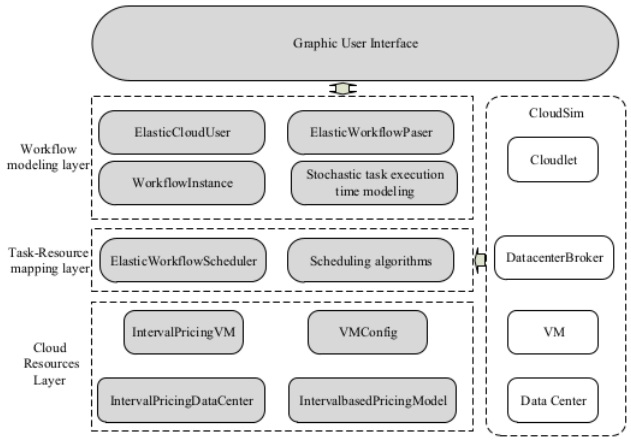


Над внутренними классами CloudSim расположен слой API — средства задания конфигурации для приложений и прочие настройки. Кроме того, в API входит и возможность переопределять политики, например, принцип размещения виртуальных машин по серверам или распределение ресурсов на одном сервере по размещенным на нем виртуальным машинам. Переопределение реализовано с помощью Java ООП.

CloudSim предоставляет возможность динамически менять конфигурацию облака — добавлять и удалять серверы, реализуя таким образом эмуляцию отказов. Кроме того, присутствует важный функционал по автоматическому измерению энергопотребления оборудования на основе заранее заданных параметров.

4.2 ElasticSim

Данный симулятор описан в статье 2016 года. В нём сделан акцент на автоматическом масштабировании количества выделенных ресурсов на задачу, что является важным именно в контексте облачных вычислений. Авторы показывают, что использование статических, заранее заданных ограничений на время работы задач и статического расписания приводит к неэффективному расходованию ресурсов.



ElasticSim основан на CloudSim, поэтому его архитектура — это лишь дополнительные слои поверх уже описанного выше. Поверх него добавлены классы для прогнозирования времени исполнения задачи, обновлённой системы ценообразования и динамического составления расписания.

По сравнению с CloudSim, в ElacticSim есть графический интерфейс, который позволяет по шагам прослеживать ход выполнения алгоритма. Исследователи могут определять свои политики прогнозирования, переопределяя Java классы и методы у слоя ElasticSim — аналогично тому, как это реализовано в CloudSim.

4.3 SCORE

Симулятор SCORE не является надстройкой над CloudSim, как ElasticSim и многие другие аналоги. Его авторы задавались целью создать систему для сравнения различных стратегий построения расписания задач на очень больших кластерах, что недоступно с CloudSim. Кроме того, данный симулятор содержит функционал для измерения энергопотребления.

Существует несколько подходов к тому, как нужно проектировать модули для составления расписания задач в облаке. В простейшей модели есть единый сервис, который статически или динамически генерирует время и место запуска для каждой задачи, однако это решение не является масштабируемым.

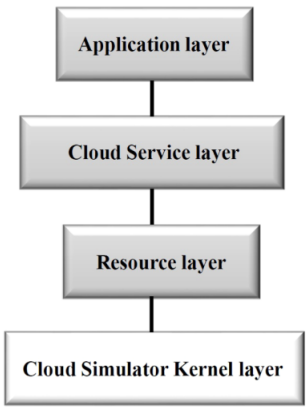
В двухслойной модели есть модуль, который выделяет ресурсы блоками и передает управления малым узлам, которые делят выделенные им ресурсы между задачами. Существуют и более сложные распределенные системы планировщиков, которые также могут быть исследованы с помощью SCORE.

SCORE основан на легковесном симуляторе Google Omega, который предоставляет основу для симулятора. Поверх неё на языке Scala реализованы классы непосредственно SCORE, отвечающие за стратегии расписания, а также для измерения потребления электроэнергии — то, чего нет в Omega.

В данном симуляторе отдельно исследуются модели, оптимизирующие энергопотребление, например, предлагающие отключать простаивающие (idle) серверы. Вообще говоря, такой подход сильно снижает производительность, так как отключенный сервер не способен быстро взять на исполнение новую задачу, однако оптимальное составление расписания позволяет это оптимизировать.

5 Сравнительный анализ

Проведём сравнение трёх рассмотренных систем. Стоит сразу отметить, что они разрабатывались с разными целями и ни одна из них не претендует на универсальность. Каждая из систем имеет свои преимущества и свои ограничения, из которых следует область их применения.



Главное сходство между симуляторами — это их многослойная структура, примерная схема которой изображена на иллюстрации. Это позволяет четко абстрагировать внутреннюю часть системы от пользовательского кода. Однако есть и отличия.

Например, CloudSim позволяет достаточно комплексно моделировать работу кластеров небольших и средних размеров за счёт множества возможностей для определения своих реализаций ключевых функций. К минусам стоит отнести уже упомянутую чрезмерную упрощённую модель сети, которая не даёт смоделировать полноценный дата-центр и более точную функцию энергопотребления. Кроме того, этот симулятор не может быть масштабирован на очень большие кластеры из-за единого планировщика, а также из-за своей сложной структуры. Гибкость CloudSim можно назвать как достоинством, так и недостатком — большое количество настроек может помешать пользователю, особенно в отсутствие GUI; это объясняет такое большое количество «унаследованных» от него симуляторов, позиционирующих себя как легковесных.

ElasticSim, в частности, является примером такого «наследника». Используя мощь CloudSim как уровня обмена сообщениями и менеджмента событий, данная система фокусируется на resource autoscaling — динамическом определении потребности приложений и изменения выделенных на них ресурсов, что отсутствует в исходном CloudSim. Создаётся впечатление, что для исследования именно этого параметра облачных систем более целесообразно брать ElasticSim, чем CloudSim.

В работе [4] описаны еще несколько систем, которые на основе CloudSim строят свой фреймворк для определенного класса алгоритма.

Симулятор SCORE также решает две узконаправленные (но связанные между собой) задачи — исследования паттернов проектирования систем планирования в облаке и оптимизации энергопотребления. Авторы прямо оговаривают, что ради быстродействия и простоты использования они отказались от «лишних» абстракций, например, от моделирования сети. С одной стороны, это облегчает работу с симулятором, с другой — может привести к высоким погрешностям результатов и расхождениям с реальной жизнью.

6 Заключение

Из проведённого анализа можно сделать логичный вывод — симулятор нужно подбирать под цели, которыми задаётся исследователь. Не существует абсолютно точной модели реального облака, любая абстракция вносит свою погрешность.

Общим местом изученных работ является детерминизм систем и «прозрачность» их моделей — в статьях приведены точные формулы, по которым вычисляются параметры алгоритма. Трендом последних лет является применение машинного обучения, в том числе в вопросах планирования. Это уже нашло отражение в ряде подобных статей, не вошедших в данный обзор. Возможно, изученные системы имеет смысл также развивать в данном направлении.

Список литературы

[1] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: Cloudsim: a toolkit for modeling and simula- tion of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41(1), 23–50 (2011)

[2] Z. Cai, Q. Li, X. Li, ElasticSim: a toolkit for simulating workflows with cloud resource runtime auto-scaling and stochastic task execution times, J. Grid Comput. 15 (2016) 257–272.

[3] D. Fernández-Cerero, A. Fernández-Montes, A. Jakóbik, J. Kołodziej, M. Toro, SCORE: simulator for cloud optimization of resources and energy consumption, Simul. Model. Pract. Theory 82 (2018) 160–173.

[4] Mansouri, N., Ghafari, R., & Zade, B. M. H. (2020). Cloud computing simulators: A comprehensive review. Simulation Modelling Practice and Theory