УДК 519.687

ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ СИМУЛЯТОРА ОБЛАЧНОЙ СИСТЕМЫ ПО ПРЕДОСТАВЛЕНИЮ ГИБРИДНЫХ ОБЛАЧНЫХ УСЛУГ ОБРАЗОВАТЕЛЬНЫМ ОРГАНИЗАЦИЯМ

Болодурина И.П., Легашев Л.В.

ФГБОУ ВО «Оренбургский государственный университет», Оренбург, e-mail: silentgir@gmail.com

Настоящая статья посвящена описанию функциональной модели симулятора облачной системы по предоставлению гибридных облачных услуг образовательным организациям. Реализация схемы DaaS (Desktop as a Service) предоставляет конечным пользователям виртуальный рабочий стол с требуемым программным обеспечением. В статье проведена формализация работы облачной системы. На основе проведенной формализации представлена функциональная модель облачной системы. Описаны основные модули функциональной модели: модуль генерации заявок, модуль интерфейса, модуль планирования и модуль назначения виртуальных машин на серверы. Приведена UML диаграмма классов симулятора облачной системы, описывающая классы симулятора и отношения между ними. Представлены результаты симуляции по генерации и размещению заявок в расписании облачной системы. Доказана эффективность использования разработанных эвристических алгоритмов на основе имитации отжига и генетического программирования.

Ключевые слова: функциональное моделирование, генерация заявок, составление расписаний, виртуальный рабочий стол как сервис

FUNCTIONAL MODEL OF CLOUD SYSTEM SIMULATOR FOR PROVIDING HYBRID CLOUD SERVICES FOR EDUCATIONAL INSTITUTIONS

Bolodurina I.P., Legashev L.V.

Federal State Educational Institution of Higher Education «Orenburg State University», Orenburg, e-mail: silentgir@gmail.com

This article is dedicated to functional model of cloud system simulator as the means of providing hybrid cloud services for educational institutions. DaaS (Desktop as a service) scheme implementation provides a virtual desktop with needed software for users. Cloud system formalization is provided. The functional model of cloud system is presented. The main modules of the functional model are described as: requests' generation module, interface module, scheduling module, virtual machines assignment module. Cloud system simulator UML diagram is presented. It depicts classes and relationships between them. Simulation results of requests' generation and scheduling are presented. The efficiency of implemented heuristics algorithms is proved.

Keywords: functional modeling, requests generation, scheduling, desktop as a service

В настоящее время многие образовательные учреждения испытывают проблемы, связанные со своевременным обновлением компьютерного оборудования, а также с закупкой и настройкой лицензионного программного обеспечения. Чаще всего причинами выступают недостаточное финансирование образовательных учреждений и отсутствие квалифицированных кадров, занимающихся установкой и поддержкой программного обеспечения. Одним из путей решения поставленных проблем является развертывание в центре обработки данных облачной системы, которая реализует услугу DaaS (Desktop as a Service), и предоставляет в аренду конечным пользователям виртуальные рабочие столы с установленным программным обеспечением [1, 2]. Существует множество исследований, касающихся вопросов составления расписаний, планирования размещения виртуальных машин в облачных системах, а также использования облачных технологий в процессе образования. Публикация [4] Питера Бракера является известной работой по теории расписаний. В ней представлены различные классификации задач планирования и показано, что в общем случае проблема планирования задач для реальных вычислительных систем является NP-полной. Составление эффективного расписания системы может быть выполнено за экспоненциальное время, что недопустимо для планировщика облачной системы. В связи с этим необходимо использовать эвристические алгоритмы планирования, которые строят оптимальные расписания за приемлемое полиномиальное время. Авторы статьи [5] рассмотрели вопросы балансировки нагрузки ресурсов виртуальной машины в облачной вычислительной среде. Ими была предложена стратегия планирования размещения виртуальных машин на основе генетического алгоритма, который учитывает текущее и предыдущие состояния системы. К основным достоинствам алгоритма можно отнести скорость его работы, среди недостатков – отсутствие учета специфики образовательных учреждений, ограничений на программное обеспечение и ограничений по времени выполнения виртуальных машин. Реализация механизма SaaS (Software as a Service) в системе образо-

вания Китая описана в статье [3]. Учащиеся получают доступ к офисным пакетам, библиотекам и прочим ресурсам посредством облачных приложений GoogleApps и Zoho Office. Среди недостатков можно отметить использование в процессе обучения только открытого программного обеспечения. Авторы статьи [6] предлагают два вида реализации схемы DaaS – для одного пользователя и для нескольких. При возрастании нагрузки на облачную систему авторы предлагают накладывать ролевые и сессионные ограничения для удалённых рабочих столов.

Анализ существующих публикаций показывает отсутствие эффективных решений по предоставлению услуги DaaS образовательным учреждениям, учитывающих ограничения по времени работы пользователей и по числу лицензий на платное программное обеспечение.

Для построения функциональной модели проведена формализация работы облачной системы. Вектор CRD = (ScheduleTemplates, Software, Requests, FLs, Datacenters) представляет собой совокупность шаблонов расписаний ScheduleTemplates = {ScheduleTemplate$_i$}, доступного программного обеспечения Software = {Software$_j$}, заявок пользователей Requests = {Request$_k$}, шаблонов конфигураций виртуальных машин FLs = {FL$_m$} и центров обработки данных облачных провайдеров Datacenters = {Datacenter$_p$}. Пользователи, число которых выражается величиной UsersCount, оставляют по одной заявке на Web-портале облачной системы. Каждая заявка представляет собой кортеж Request$_k$ = (ScheduleTemplate$_k$, VMCount$_k$, s_k, $t_{arrive.k}$, t_k, w_k, Status$_k$), который содержит: используемый шаблон расписания образовательного учреждения ScheduleTemplate$_k$ ∈ ScheduleTemplates, количество экземпляров виртуальных машин VMCount$_k$, подмножество запрашиваемого ПО $s_k \subseteq$ Software из списка доступного программного обеспечения, время появления заявки $t_{arrive.k}$, набор индексов временных слотов $t_k = \{In_{1k}, In_{2k}, ..., In_{r_k k}\}$, где $In_{lk} \in Intervals_k$, $(l = \overline{1, r_k}$, временные слоты принадлежат шаблону заявок), весовой коэффициент пользователя w_k, а также статус размещения заявки пользователя в расписании Status$_k$ = {– 1, 0, 1}. Значение Status$_k$ = 1 показывает, что заявка размещена в расписании в указанное пользователем время. Иначе, если планировщику не удалось разместить заявку пользователя в указанное им время, и он подобрал другие временные слоты, то статус заявки принимает значение Status$_k$ = 0. Значение Status$_k$ = – 1 соответствует случаю, когда заявку пользователя вообще не удалось разместить в текущем расписании.

Предложена следующая функция оптимизации, отражающая степень выполнения суммарных пожеланий всех координаторов образовательных учреждений по запуску виртуальных машин в указанное ими время:

$$F(S) \to \max,$$

$$F(S) = \sum_{k=1}^{UsersCount} w_k [\alpha \cdot PrefferedTS_k + \beta \cdot notPrefferedTS_k - \gamma \cdot unsTS_k], \quad (1)$$

где α – коэффициент поощрения при размещении виртуальной машины в указанные пользователем временные интервалы; $PrefferedTS_k$ – количество таких машин в k-й заявке; β – коэффициент поощрения при размещении машины в интервалы времени, не указанные пользователем; $notPrefferedTS_k$ – количество таких машин в k-й заявке; γ – штрафной коэффициент при невозможности размещения виртуальной машины пользователя в указанное время; $unsTS_k$ – количество таких машин в k-й заявке.

Для всех единиц программного обеспечения и всех временных интервалов пользователей выполняются ограничения на максимальное число лицензий программного обеспечения в каждую минуту времени:

$$\forall j = 1..|Software|, \forall l = [480, 1080]: AssignedVMsC(S, j, l) \le LicenseCount_j, \quad (2)$$

где AssignedVMsC(S,j,l) – общее число виртуальных машин, выполняющих j-е программное обеспечение в l-й интервал времени согласно расписанию S.

На основе проведенной формализации предложена следующая функциональная модель облачной системы (рис. 1):

1. На первом этапе происходит генерация параметров заявок конечных пользователей облачной системы. Каждый пользователь формирует одну заявку, поэтому количество заявок RequestsCount совпадает с количеством пользователей UsersCount. Индекс шаблона расписания

$ScheduleTemplate_k$, набор индексов временных слотов t_k и количество требуемого программного обеспечения $|s_k|$ представляют собой равномерно распределенные случайные величины. Множество требуемого программного обеспечения s_k формируется путем случайной перестановки элементов множества Software с последующим выбором первых $|s_k|$ элементов. Количество экземпляров виртуальной машины пользователя $VMCount_k$ распределено равномерно на отрезке [5, 15], который соответствует минимальному и максимальному количеству учеников/студентов в классе/группе.

2. На втором этапе используется интерфейс облачной системы для связи конечных пользователей с web-порталом облачной системы. Сгенерированные параметры заявок пользователей используются в качестве входных данных при заполнении полей заявки. В соответствии с выбранным программным обеспечением для каждого пользователя подбирается подходящий шаблон конфигурации виртуальной машины FL_m, которая способна запустить на выполнение затребованный виртуальный рабочий стол и обеспечить возможность его функционирования.

3. Модуль планирования состоит из двух объектов: «Очередь» и «Планировщик». Сгенерированные запросы пользователей Requests добавляются в очередь модуля планирования. В качестве базовых алгоритмов составления расписания выбраны эвристические методы имитации отжига и генетическое программирование. Задача составления расписания облачной системы в условиях ограничений представима базовыми операциями метода имитации отжига и генетического программирования. Основной структурной единицей работы алгоритмов является один из готовых вариантов расписания облачной системы. Для инициирования работы эвристических алгоритмов составления расписания облачной системы требуется первоначальный неоптимизированный вариант расписания, который строится с помощью метода Round-robin. В ходе работы алгоритмов заявки пользователей извлекаются из очереди и размещаются в расписании S облачной системы, при этом проверяется выполнение существующих ограничений (2) и происходит назначение соответствующих статусов заявок пользователей.

4. На последнем этапе работы облачной системы происходит подбор физических серверов выбранного облачного провайдера для запуска требуемых шаблонов виртуальных машин на выполнение в указанные временные сроки. Возможность подключения подобных типов облаков провайдеров (как публичных, так и приватных) позволяет сделать вывод о том, что разрабатываемая облачная система является гибридной.

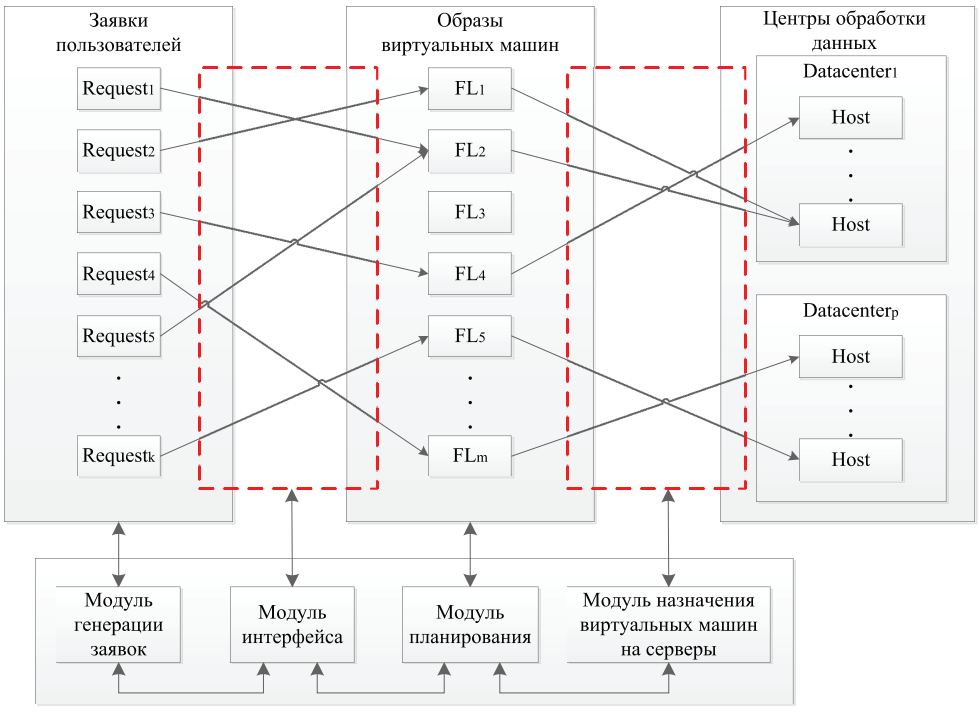

Рис. 1. Функциональная модель облачной системы

Симулятор облачной системы представляет собой совокупность двух модулей функциональной модели – модуля генерации заявок и модуля планирования. Симулятор реализован с использованием высокоуровневого языка C++ в среде Visual Studio 2012. На рис. 2 представлена UML диаграмма классов, описывающая основные классы симулятора облачной системы и отношения между ними.

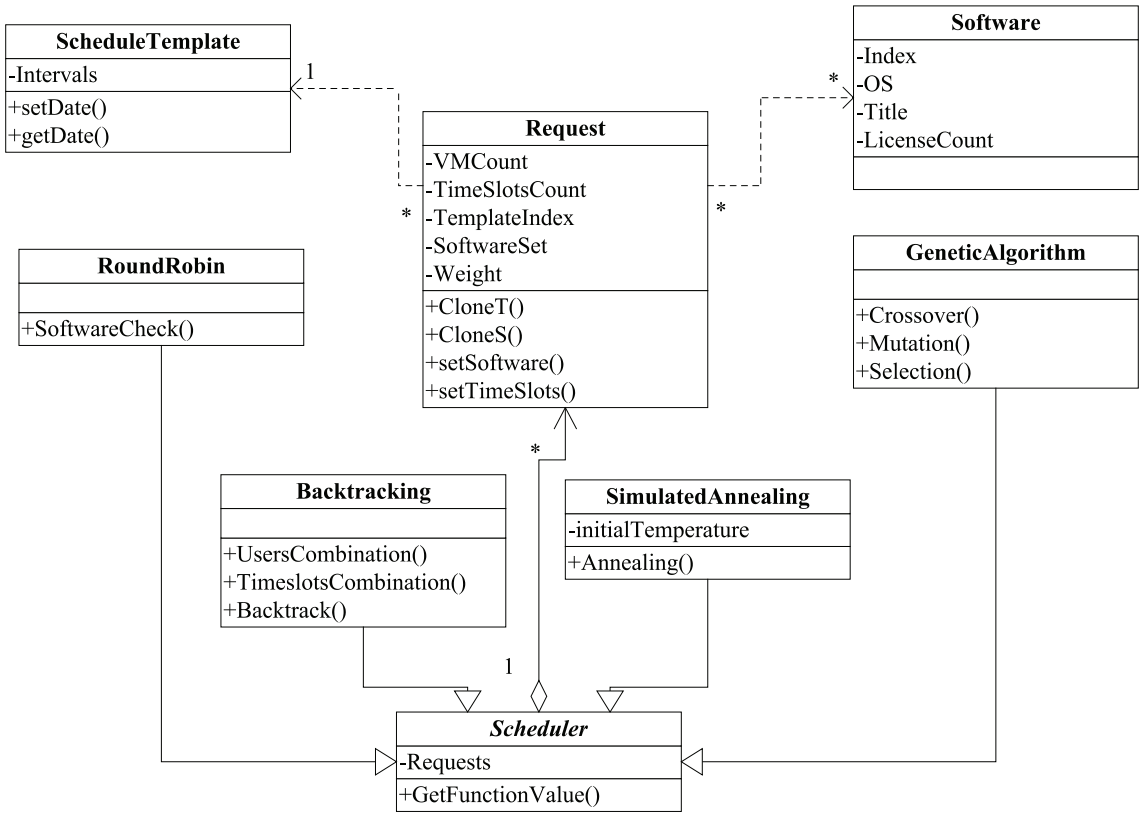

Рис. 2. UML диаграмма классов симулятора облачной системы

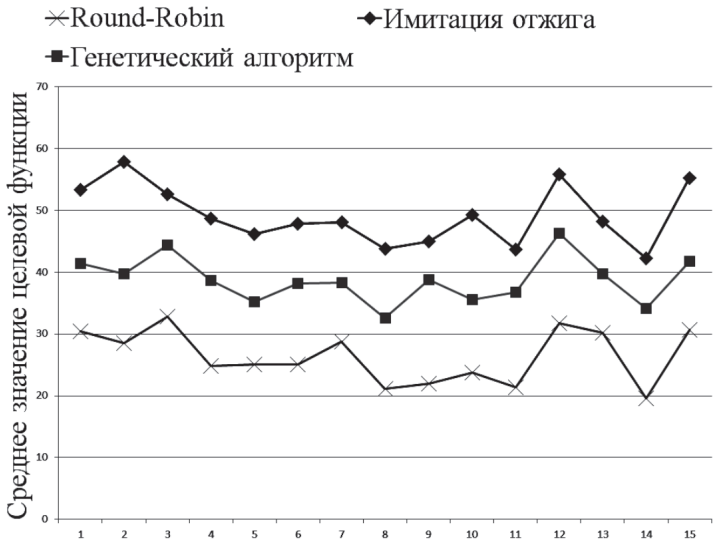

Рис. 3. Среднее значение целевой функции для разработанных алгоритмов составления расписания облачной системы

Совокупность классов ScheduleTemplate, Request и Software описывает реализацию модуля генерации заявок. Класс ScheduleTemplate имеет поле Intervals, которое содержит множество всех временных слотов каждого шаблона расписания образовательного учреждения. Методы setDate() и getDate() используются для выполнения операций чтения/записи над временными слотами. Класс Software содержит информацию о доступном программном обеспечении, такую как: индекс ПО, поддерживаемую операционную систему и максимальное число лицензий. Класс Request используется для описания заявки пользователя. Его свойство TemplateIndex содержит указатель на используемый шаблон расписания, а свойство SoftwareSet содержит список индексов запрашиваемого программного обеспечения. Метод setSoftware() случайным образом генерирует множество индексов программного обеспечения для заявки каждого пользователя. Метод setTimeSlots() случайным образом генерирует множество индексов временных слотов для заявки каждого пользователя. Метод CloneT() используется для копирования выбранных индексов временных слотов. Метод CloneS() используется для копирования индексов выбранного программного обеспечения.

Совокупность классов RoundRobin, Backtracking, SimulatedAnnealing, GeneticAlgorithm и Scheduler описывает реализацию модуля планирования. Вызов метода SoftwareCheck() класса RoundRobin инициирует построение первоначального варианта расписания облачной системы. Класс Backtracking описывает реализацию алгоритма перебора, который используется для оценивания эффективности алгоритма имитации отжига и генетического алгоритма. Его методы UsersCombination() и TimeSlotsCombination() вызываются для построения всех возможных комбинаций размещения заявок пользователей в расписании, что представляет собой декартово произведение сочетаний без повторений из всех временных слотов шаблона расписания по количеству временных слотов пользователя. Метод Annealing() класса SimulatedAnnealing создает оптимальное расписание облачной системы за полиномиальное время. Свойство initialTemperature регулирует число итераций работы алгоритма. Методы Crossover(), Mutation() и Selection() класса GeneticAlgorithm реализуют операции скрещивания, мутации и селекции генетического алгоритма. Класс Scheduler является абстрактным классом, от которого наследуются остальные классы симулятора облачной системы: RoundRobin, Backtracking, SimulatedAnnealing и GeneticAlgorithm. Метод GetFunctionValue() используется для вычисления значения целевой функции (1).

Проведены исследования симулятора облачной системы по генерации и размещению заявок пользователей в расписании. На рис. 3 представлены результаты 150 экспериментов. Каждая точка на оси X представляет собой отрезок усреднения значения целевой функции (1) по 10 экспериментам. Разработанный метод на основе имитации отжига в среднем размещает 89,7 % от общего числа заявок пользователей. Разработанный метод на основе генетического программирования в среднем размещает 86,5 % от общего числа заявок пользователей.

В результате исследования формализована работа облачной системы. Представлена функциональная модель облачной системы. Описаны основные модули функциональной модели. Приведена UML диаграмма классов симулятора облачной системы. Представлены результаты симуляции по генерации и размещению заявок пользователей в расписании облачной системы.

Исследования проведены при финансовой поддержке Министерства образования Оренбургской области (грант № 37 от 30 июня 2016 г.), РФФИ и Правительства Оренбургской области (проекты № 16-47-560335, № 16-07-01004), Президента Российской Федерации, стипендии для молодых ученых и аспирантов (СП-2179.2015.5), гранта Президента Российской Федерации для государственной поддержки молодых российских ученых МК-1624.2017.9.

Список литературы

1. Болодурина И.П., Парфёнов Д.И. Исследование моделей оптимизации управления трафиком сервис-ориентированных облачных приложений в программно-управляемой инфраструктуре виртуального центра обработки данных // Современные наукоемкие технологии. – 2016. – № 12–2. – С. 229–235.

2. Болодурина И.П., Полежаев П.Н., Ушаков Ю.А., Шухман А.Е. Концепция регионального центра коллективного доступа к образовательным программным продуктам на основе облачных технологий // VIII Международная научно-практическая конференция «Информационные технологии в образовании» г. Саратов, 2–3 ноября 2016 г., ИТО-Саратов, 2016. – С. 332–336.

3. Bo Wang, Hong Yu Xing. The Application of Cloud Computing in Education Informatization // IEEE International Conference on Cloud Computing Technology and Science. – 2011. – P. 2673–2676.

4. Brucker P. Scheduling Algorithms [Текст] / P. Brucker. – Берлин: Springer, 2007. – 371 p.

5. Jinhua Hu, Jianhua Gu, Guofei Sun, Tianhai Zhao. A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment // 3rd International Symposium on Parallel Architectures, Algorithms and Programming. – 2010. – P. 89–96.

6. Shinichiro Kibe, Teruaki Koyama, Minoru Uehara. The Evaluations of Desktop as a Service in an Educational Cloud // 15th International Conference on Network-Based Information Systems – 2012. – P. 621–626.