Санкт–Петербургский государственный университет

Фундаментальные информатика и информационные технологии

Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Власова Ксения Андреевна

Магистерская диссертация

Мультиагентный подход к моделированию эпидемий в облаке

Научный руководитель:
д.ф.-м.н., проф.
Терехов А. Н.

Рецензент:
ведущий бизнес–аналитик
Ерёмушкина С. Ю.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Fundamental Computer Science and Information Technologies

Software of Computers, Complexes and Networks

Vlasova Kseniia

Master's Thesis

Epidemics modelling using cloud-based multi-agent systems

Scientific supervisor:
Ph. D., professor Andrey Terehov

Reviewer:
Lead business analyst,
postgraduate Eremushkina Svetlana

Saint-Petersburg
2016

Содержание

1. Введение

Эпидемиологические угрозы последних лет требуют для своего решения не только медикаментозных, но и организационных мер. Неожиданное возникновение эпидемий может привести к негативным последствиям в случае непродуманных действий специалистов различных служб (органов здравоохранения, администрации, аптечных сетей и т.д.). При осуществлении качественного прогноза ожидаемого уровня заболеваемости и возможных последствий, становится возможным также заранее выявить приближение эпидемии. Прогнозирование динамики распространения заболевания, полученное в результате построения имитационной модели, позволит специалистам заблаговременно продумать меры борьбы с эпидемией.

Имитационное моделирование — это подход, для которого характерен перевод процессов реального мира в некоторое математическое описание или упрощённое физическое представление [1, 2]. Одним из видов имитационного моделирования является построение мультиагентных систем. Мультиагентные системы представляют собой совокупность взаимодействующих агентов для решения определённой задачи. Агент — это некоторая сущность, которая обладает активностью, автономным поведением, может принимать решения в соответствии с некоторым набором правил, может взаимодействовать с окружением и другими агентами [15].

Для того чтобы модель распространения эпидемии была наиболее точной, необходимо множество различных параметров, таких как возраст агентов, вероятность их пребывания в той или иной локации в зависимости от возраста и времени суток и т. д., что влечёт за собой увеличение вычислительных ресурсов. Вычислительная мощность компьютера, на котором производится запуск модели, является одной из основных причин, по которой детализирование модели становится весьма нецелесообразным. Для преодоления этого ограничения хорошим решением является использование для вычислений облачных технологий. Облачные вычисления — подход

к разработке приложений, в котором общие ресурсы, данные и информация хранятся на мощных масштабируемых кластерах и предоставляются по требованию компьютерам и другим устройствам. Как следует из определения, одним из главных преимуществ облачных вычислений является масштабируемость, позволяющая при необходимости быстрее вычислять сложные задачи по сравнению с персональным компьютером.

Программный комплекс на основе мультиагентных систем в облаке позволит построить прогноз развития эпидемиологической обстановки в городе, а также количественно оценить масштабы потерь.

Цель работы заключается в моделировании эпидемий на примере лихорадки Эбола [9] с помощью мультиагентной системы с учётом распределения вычислительной нагрузки в облаке.

Для её достижения были сформулированы следующие задачи:

- реализовать мультиагентную систему, моделирующую распространение эпидемии;
- адаптировать систему к размещению в облаке, а именно:
  - разделить систему на масштабируемые модули;
  - распределить вычислительную нагрузку;
  - организовать сбор результатов.
- произвести балансировку вычислительной нагрузки перед запуском модели и во время её расчёта.

2. Обзор существующих решений

Для уточнения особенностей заболевания была выбрана модель SEIRFD, как наиболее соответствующая реальности и подходящая данному заболеванию [4]. Она представлена на рисунке 1 в виде конечного автомата.

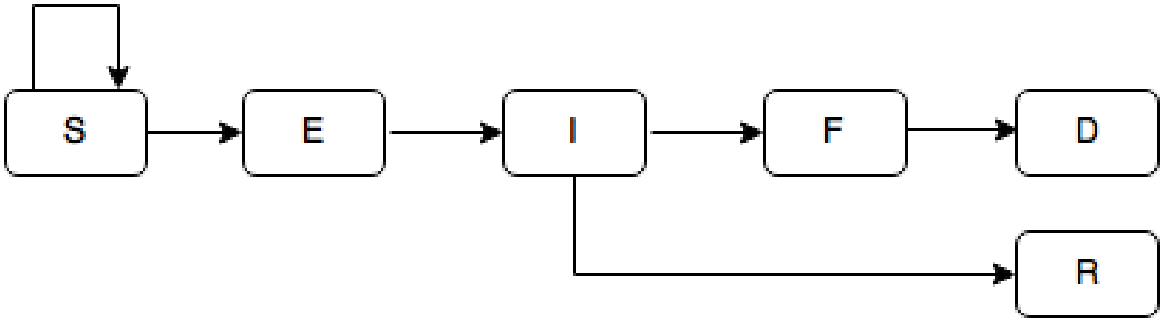

Рис. 1. Конечный автомат модели SEIRFD.

Данная модель является модификацией модели SIR [19], являющейся упрощённым способом описания передачи инфекционных заболеваний [13].

Согласно модели SEIRFD, каждый агент может находиться в одном из шести состояний.

- S (Susceptible) - восприимчивые к заражению;
- E (Exposed) — заражённые, находящиеся в инкубационном периоде;
- I (Infectious) — заражённые;
- F (Funeral) — мёртвые, но по-прежнему являющиеся источником заражения;
- D (Died) — погребённые;
- R (Recovered) — выздоровевшие.

Задача моделирования эпидемий актуальна с давних времён, и её пытались решить неоднократно. Существуют традиционные модели распространения заболеваний, которые, однако, требуют различных допущений (однотипность индивидуумов, их непрерывное равномерное перемешивание на моделируемой территории), вследствие чего являются недостаточно

точными [3].

В работе [3] разрабатывается модель распространения гриппа A на основе агентного подхода при помощи среды AnyLogic [3]. Модель учитывает несколько факторов снижения инфекционной заболеваемости и рассчитывает прогноз на месяц вперёд.

Другая работа [4] реализует модель распространения эпидемии Эбола на основе мультиагентного подхода. Моделирование происходит также с помощью программного инструмента AnyLogic.

Первый недостаток обеих вышеупомянутых моделей заключается в ограничении на количество агентов рассматриваемой локации, а значит, и площади территории, на которой они располагаются. Второй недостаток — затруднительность детализации модели из-за ограниченных возможностей процессора машины, на которой выполняется моделирование.

2.1. Реализованные системы прогнозирования эпидемии

На данный момент существует система, моделирующая динамику развития эпидемий, вызванных особыми патогенами [17], разработанная в Российском Государственном научном центре вирусологии и биотехнологии. Модель включает в себя ряд противодействий: профилактику, неотложную массовую вакцинацию, вакцинацию групп риска, карантин и другое. Возможно проведение моделирований таких эпидемий как грипп, натуральная оспа, Эбола, холера и другие, дающее сравнительно хороший результат. Модель распространения эпидемии относится к классу SEmInRF (S — восприимчивые, E, m, In — разновидности инфицированных, R — выздоровевшие, F — мёртвые), однако в состоянии F заражения не происходит, а с учётом особенностей моделируемой территории это существенно. Стоит отметить, что модель находится в закрытом доступе и как следствие не предполагает детализации.

Имеется также основа для реконструкции динамики развития эпидемий FRED (A Framework for Reconstructing Epidemiological Dynamics). Это система моделирования с открытым исходным кодом, использующая агентное моделирование, основанное на искуственно сформированных группах населения, построенных на данных переписи населения, которые содержат демографическое и географическое распределения популяции. Недостатком модели является невозможность проводить моделирование на другой местности, поскольку встроенные данные предназначены только для территории США [16].

Центр по контролю и профилактике заболеваний США опубликовал исследование на тему моделирования распространения воздействия болезней и вмешательства. Модель позволяет оценить число случаев заболевания вирусом Эбола, однако не ведется статистика смертности [18].

2.2. Вирус Эбола

Болезнь, вызванная вирусом Эбола, — острая вирусная инфекция, являющаяся часто смертельной, с коэффициентом смертности, достигающим 90%. Поражает людей и некоторые виды животных [9]. По последним данным число жертв Эболы составляет примерно одиннадцать тысяч [10]. Первые вспышки болезни, вызванной вирусом Эбола, появились в деревнях Центральной Африки, однако самые последние вспышки в Западной Африке охватили крупные города и сельские районы [11].

Исследованием вируса Эбола занимаются мировые научные организации, и статистические данные по данному заболеванию были выложены в открытый доступ [29]. Нас интересовала статистика зараженных и умерших людей с 23 мая 2014 года. Исследования проводились по городу Кайлахун, как наиболее пострадавшему от эпидемии.

2.2.1. Модель протекания заболевания SEIRFD

В модели, как и в реальной жизни, основная часть агентов находится в состоянии "Susceptible". При непосредственном контакте с больным ("Infectious") или непогребённым ("Funeral") агентом, появляется возможность с определённой вероятностью заразиться. Если заражение произошло, наступает инкубационный период ("Exposed"), в котором агент может находиться от 8 до 12 дней. По истечении этого срока агент считается заболевшим (состояние "Infectious") и становится способным передавать с некоторой вероятностью инфекцию. Из стадии "Infectious" агент может попасть либо в состояние "Funeral" (через 10 дней), либо в конечное состояние "Recovered" (через 20 дней). В первом случае сохраняется способность передавать болезнь до момента погребения тела (около 4 дней), после чего агент переходит в конечное состояние "Died". Во втором случае у агента вырабатывается на всю жизнь иммунитет к вирусу Эбола. Средняя длительность всех состояний, вероятности заболевания в случае возникновения контакта с агентами в состояниях "Infectious" и "Funeral", согласованы с данными из исследований [14] и дополнительно подстроены под нашу модель. Такая модель протекания заболевания подойдет для моделирования другого типа инфекции — входные параметры могут быть быстро и эффективно скорректированы.

3. Описание решения

3.1. Моделирование эпидемий с помощью мультиагентной системы

3.1.1. Агентный подход

В модели агент представляет собой аналог человека, который может быть болен, и при выполнении некоторых условий может заражать других.

Агенты формируют популяцию, каждый агент обладает индивидуальными свойствами, например, принадлежностью к возрастной группе. В модели определенным образом описывается протекание заболевания у каждого агента, например, на основе состояний, предложенных в модели SEIRFD.

3.1.2. Контейнерный подход

В данной модели используется контейнерный подход — набор различных локаций, в которых может находиться агент, представляет собой контейнеры, различающиеся площадью и плотностью. Каждый агент может передвигаться из одной локации в другую, то есть перемещаться из одного контейнера в другой.

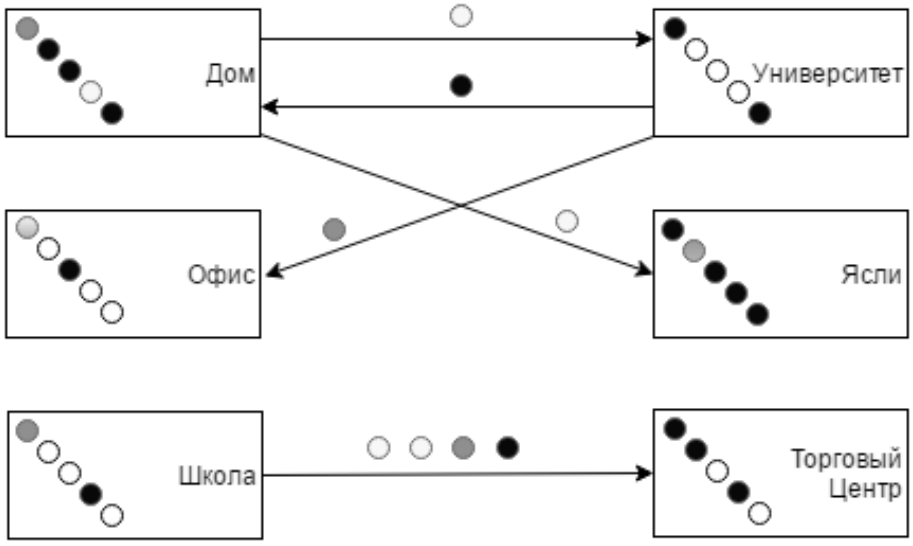

Рис. 2. Контейнерный подход.

Задано семь типов контейнеров: Дом, Больница, Торговый центр, Ясли, Офис, Школа, Университет. В рассматриваемой реализации на основе данных, предоставленных Управлением по координации гуманитарных вопросов Содружества Наций [12] и демографической статистики Сьерра–Леоне, присутствует следующее количество контейнеров:

- Школы: 23;
- Больницы: 4;
- Офисы: 500;

- Дома: 10000;
- Торговые центры: 5;
- Ясли: 2;
- Университет: 1.

В системе присутствует множество контейнеров каждого типа. Стоит отметить, что в каждый момент модельного времени агент находится в одном определённом контейнере.

3.1.3. Возрастные категории

Специалистами НИИ гриппа [3] было предложено следующее разделение на возрастные категории:

- дети до 4 лет;
- подростки 5 – 14 лет;
- молодежь 15 – 24 года;
- взрослые 25 – 54 лет;
- люди преклонного возраста – от 55 лет.

Возрастная группа человека позволяет наиболее точно определить вероятность заболевания с помощью вероятности пребывания в той или иной локации. Для каждой возрастной группы задан определённый набор правил, состоящий из временного интервала пребывания в определённом контейнере, типа контейнера и вероятности пребывания в нём.

3.2. Выбор технологий

Рассмотрим альтернативы.

3.2.1. Akka.Net

Akka.Net — набор инструментов для построения параллельных, распределённых и отказоустойчивых приложений, управляемых событиями.

Основными объектами являются акторы. Акторы представляют собой объекты, способные совершать определённый набор действий и общаться между собой путём обмена сообщениями. Akka.Net можно запустить на одной Worker-роли, однако на нескольких — затруднительно, поскольку синхронизировать и организовывать передачу сообщений придётся вручную.

3.2.2. Orleans

Orleans — платформа, ориентированная на построение распределённых крупномасштабных вычислительных приложений без необходимости изучать и применять сложный параллелизм или другие модели масштабирования. Система, построенная на основе Orleans, состоит из набора легковесных агентов (grains), расположенных на виртуальных машинах (silos). Создана подразделением Microsoft Research и предназначена для использования в облачных приложениях на платформе .NET [7]. Главное преимущество данной платформы — хорошая масштабируемость. Основным её недостатком является невозможность балансировать агентов во время работы системы.

3.2.3. Java, Jade

Jade (Java Agent Development framework) – один из распространенных инструментов для создания мультиагентых систем. Написан на Java, поэтому является кроссплатформенным, может работать на беспроводных мобильных устройствах, поддерживает стандарты FIPA и распространяется бесплатно по лицензии LGPL [20]. Существует реализация облачной мультиагентной платформы на основе Jade в облачном сервисе Google App Engine [8]. Основным недостатком данной работы является отсутствие балансировки нагрузки агентов. Для нашей задачи наличие балансировки является обязательным: агенты могут переходить из контейнера в контейнер, из-за чего может произойти перегрузка узлов.

3.3. Описание программной реализации

3.3.1. C#, Microsoft Azure, Service Bus

В качестве решения был выбран следующий стек технологий: C#, Microsoft Azure, Servise Bus. Azure — облачная платформа от Microsoft, предоставляющая возможность разработки и выполнения приложений и хранения данных на серверах, расположенных в распределённых дата-центрах.

Одним из способов написания приложений для Azure является применение Cloud Servises Microsoft Azure. Он предоставляет разработчику следующие способы развёртывания частей приложения:
Web-роль — слой приложения, выполняющий роль веб-интерфейса, взаимодействующего с пользователем.
Worker-роль — слой приложения, выполняющий роль обработчика данных.

Microsoft Azure обеспечивает поддержку двух типов механизмов очередей — Azure Queues и Service Bus Queues. Для обеспечения взаимодействия компонент была выбрана система Microsoft Azure Service Bus (служебная шина Microsoft Azure) (рис. 3) — многопользовательская облачная служба [28]. Гарантия сохранения очередности сообщений позволяет упростить код программы и избежать реализации дополнительных механизмов поддержания правильного порядка получаемых сообщений.

Для каждого пользователя создаётся пространство имён и затем определяются необходимые механизмы связи. Внутри пространства имен можно использовать один или более экземпляров четырех различных механизмов связи, каждая из которых по-разному связывает приложения. Возможны следующие варианты:

- *Очереди (Queues):* обеспечивают однонаправленную связь. Каждая очередь выступает в качестве посредника, который хранит отправленные сообщения, пока они не будут получены. Каждое сообщение предназначается одному получателю.

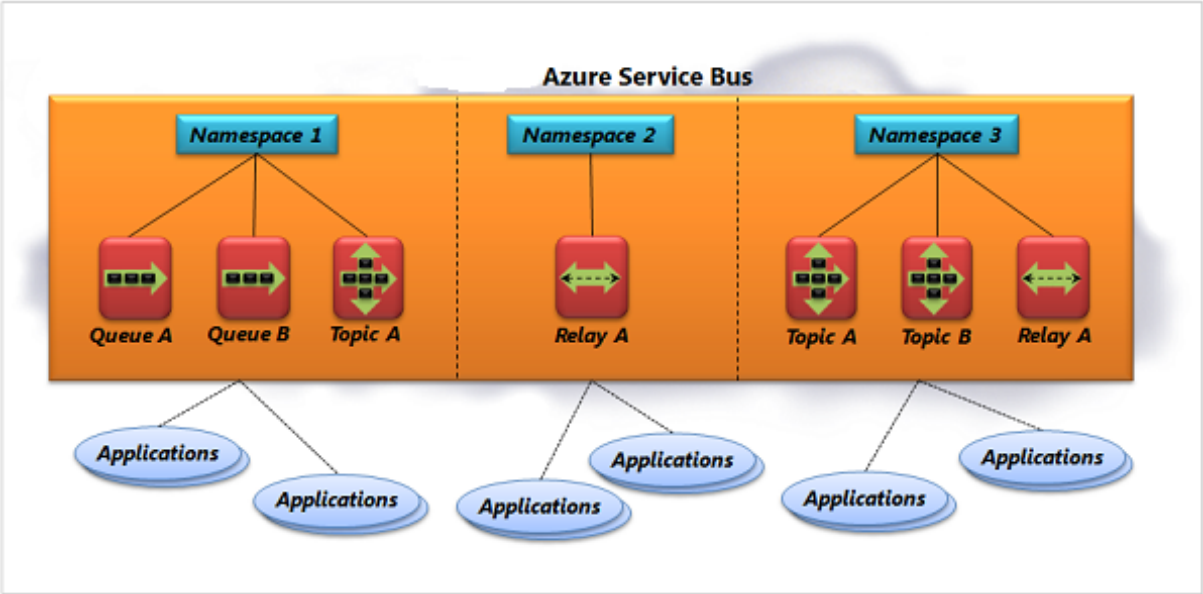

Рис. 3. Service Bus.

[28]

- *Темы (Topics):* предоставляют однонаправленную связь с использованием подписок — одна тема может иметь несколько подписок. Как и очередь, тема выступает в качестве посредника, но каждая подписка может дополнительно использовать фильтр. Такой подход полезен в случае, когда несколько приложений заинтересовано в одних и тех же сообщениях. Достаточно определить правильный фильтр, и каждый подписчик сможет подключиться только на ту часть потока сообщений, которую он должен видеть. Например, на рисунке 4 показан отправитель и тема с тремя подписчиками, каждый со своим собственным фильтром [28]:
  - Подписчик 1 получает только те сообщения, которые содержат свойство Продавец = "Ava".
  - Подписчик 2 получает сообщения, которые содержат свойство Продавец = "Ruby" и/или содержат свойство *Amount*, значение которого превышает 100000.
  - Подписчик 3 установил свой фильтр на "True", а это значит, что он принимает все сообщения.

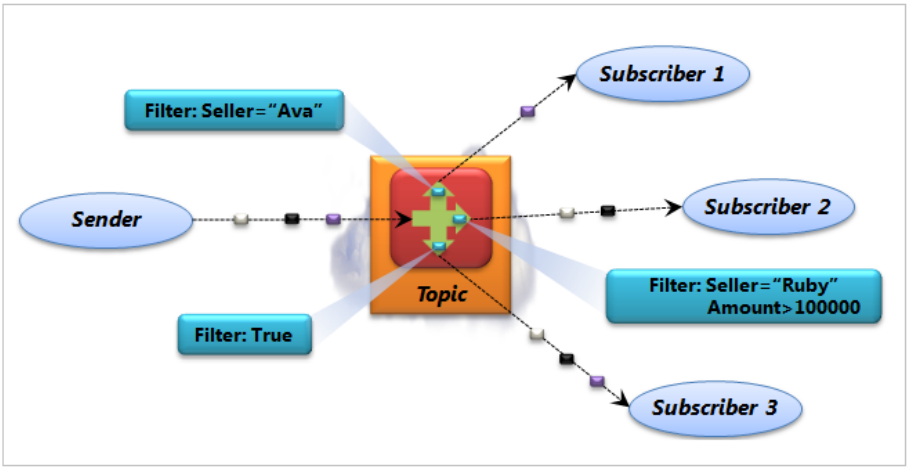

Рис. 4. Фильтрация подписок.

[28]

- *Переключатели (Relays):* обеспечивают двунаправленную связь. В отличие от очередей и тем, переключатель не хранит передающиеся сообщения. Вместо этого он просто отправляет их целевому приложению.
- *Event Hubs:* предоставляет доступ к событиям и телеметрии в облако в массовом масштабе, с низкой задержкой и высокой надежностью.

Microsoft предоставляет Servise Bus SDK для Java, JavaScript (Node.js) и других языков.

3.3.2. Архитектура системы

Система состоит из следующих компонент: нескольких Cloud Simulation (CS) и одного Global Descriptor (GD), общающихся между собой (рис 5). Global Descriptor хранит вспомогательную информацию, необходимую для работы системы. Cloud Simulations содержат контейнеры с агентами.

Взаимодействие между Global Descriptor и Cloud Simulations показано на рис 6 и происходит следующим образом:

Перед началом работы системы каждый Cloud Simulation отправляет Global Descriptor регистрационное сообщение со своим уникальным иден-

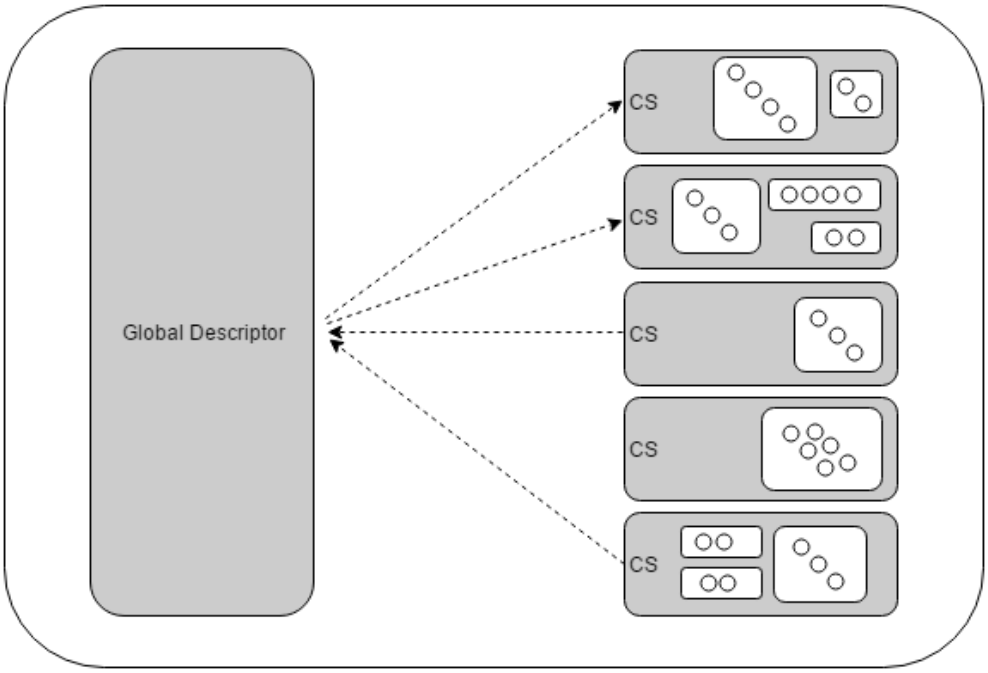

Рис. 5. Компоненты системы.

тификатором. После того как у Global Descriptor появляется список всех Cloud Simulation с их идентификаторами, он распределяет по ним контейнеры с агентами в соответствии с алгоритмом Round-Robin и посылает сообщение о старте.

Во время работы, согласно контейнерному подходу, происходят переходы агентов между контейнерами, в том числе и контейнерами, расположенными на разных Cloud Simulation. Принцип перехода следующий: агент посылает Global Descriptor сообщение о переходе, содержащее информацию о типе контейнера. Global Descriptor, в свою очередь, находит для этого агента контейнер требуемого типа на каком-либо из Cloud Simulations и посылает сообщение о добавлении агента в новый контейнер.

Для обеспечения корректности перехода агентов между контейнерами, расположенными на разных узлах (Cloud Simulation), требуется поддержка одинакового времени на всех узлах. Для этого по истечении каждого модельного часа происходит синхронизация с глобальным временем системы.

Во избежание потери данных при сбое модуля Global Descriptor, они

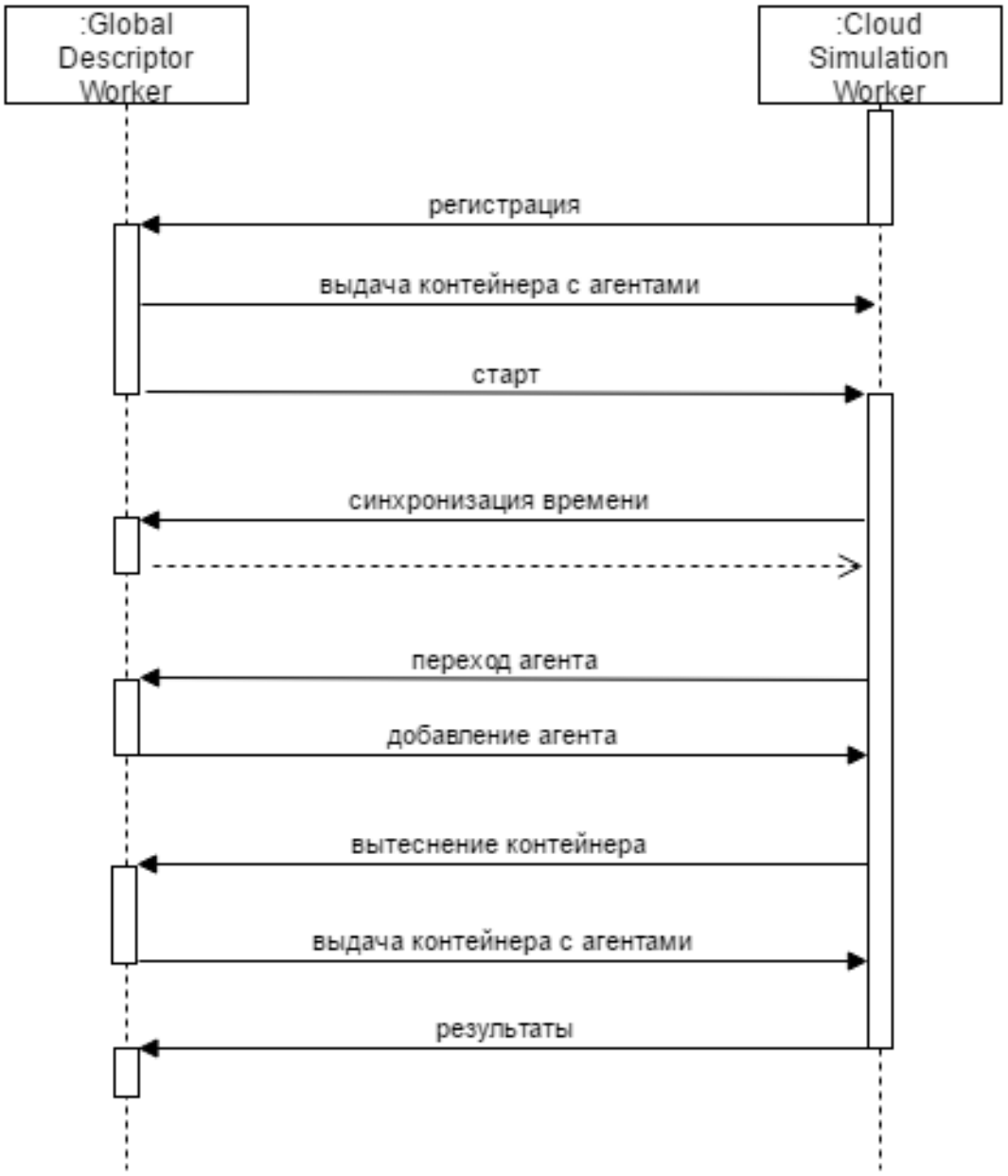

Рис. 6. Взаимодействие компонент.

отправляются в хранилище BLOB-объектов Azure, представляющую собой службу хранения большого количества неструктурированных данных объектов. Сохраняются следующие типы данных: промежуточные — во время синхронизации времени; результаты итерации — после итерации; результат — по окончании работы системы.

Для обеспечения балансировки во время работы существует ещё один тип сообщений — переход контейнера с одного узла на другой. Процесс вытеснения контейнеров рассматривается ниже в разделе "Начальная балан-

сировка".

3.3.3. Дополнение системы балансировщиком нагрузки

Балансировка нагрузки — метод распределения задач между физическими или виртуальными вычислителями. Задача любого алгоритма балансировки нагрузки — увеличить использование незагруженных или слабо загруженных узлов и уменьшить использование узлов с большой нагрузкой. Балансировка нагрузки производится для следующих целей:

- Сокращение времени выполнения и времени ожидания задач
- Повышение доступности услуг
- Оптимальное использование доступных ресурсов
- Поддержание стабильности системы (например, при появлении poison message — задачи, находящейся в очереди, которую узел не может обработать из-за ошибок) [21].
- Адаптация к будущим изменениям (в случае изменения пула ресурсов)

Для оценки достижения этих целей используются следующие формальные критерии качества:

- Накладные расходы - это потребляемые ресурсы балансировщика нагрузки. Этот критерий необходимо минимизировать.
- Учёт гетерогенности облака — способность алгоритма адаптироваться к узлам с разной производительностью. Наличие данного признака является преимуществом.
- Масштабируемость — сохранение качества балансировки при уменьшении или увеличении числа узлов.
- Отказоустойчивость — способность системы продолжать балансировку нагрузки, несмотря на сбой в узле.
- Утилизация ресурсов — способность системы распределять задачи так, чтобы ресурсы (узлы) не простаивали. Этот критерий необходимо максимизировать.

3.3.4. Основные виды алгоритмов балансировки нагрузки

На основании используемого анализа текущего состояния системы, алгоритмы балансировки нагрузки подразделяются на два типа: статические и динамические [23].

В статических алгоритмах текущее состояние узлов не учитывается. Все узлы и их характеристики (например, быстродействие, пропускная способность сетевого канала) известны заранее, и алгоритм работает на основе этого предварительного знания, вследствие чего этот тип алгоритмов проще в реализации.

Динамические алгоритмы учитывают текущее состояние системы и работают в соответствии с изменениями состояний узлов. Алгоритм собирает текущее состояние узлов в таблицу состояний и поддерживает текущий статус всех узлов в облаке. Динамические алгоритмы сложны в реализации, но как правило, балансируют нагрузку эффективнее.

3.3.5. Алгоритмы балансировки нагрузки

В настоящее время распространены следующие алгоритмы балансировки нагрузки в облаках:

1) Алгоритм Round-robin (циклическое обслуживание) [5]. Это статический алгоритм балансировки нагрузки; он случайным образом выбирает первый узел и затем распределяет задачи по всем узлам по кругу следующим образом: перед запуском системы создаётся упорядоченный список узлов и в ходе работы алгоритма задачи назначаются узлам в соответствии с их позицией в списке. Задачи быстро распределяются по узлам и не накапливаются в очереди балансировщика. Время работы любого процесса заранее неизвестно, из-за чего одни узлы могут оказаться перегруженными, а другие — слабо загруженными. По этой причине использование ресурсов у алгоритма низкое. Адаптируемости к гетерогенному

облаку нет — не учитывается производительность узлов, вследствие чего задачи склонны к скапливанию на узлах в ожидании своего исполнения. Накладные расходы низкие, поскольку балансировщику нагрузки требуется хранить лишь упорядоченный список узлов. Вследствие необходимости предварительной нумерации узлов масштабируемость средняя.

2) Весовой алгоритм Round-robin (weighted round-robin) [5]. Этот статический алгоритм, в отличие от предыдущего, распределяет задачи по узлам более эффективно. Это достигается следующим образом: узлу присваивается некоторый весовой коэффициент. Каждый узел, на основе заранее заданных им весов, получает определенную часть запросов. Если назначенные веса равны, каждый узел получает равную порцию входящих задач.

3) Оппортунистический алгоритм балансировки нагрузки (Opportunistic Load Balancing Algorithm) [24]. Этот статический алгоритм балансировки нагрузки предназначен для поддержания каждого узла в занятом состоянии независимо от текущей нагрузки каждого узла. Этот алгоритм быстро раздает невыполненные задачи в случайном порядке текущим свободным узлам, текущая загруженность узла не высчитывается.

4) Алгоритм балансировки нагрузки min-min (Min-Min Load Balancing Algorithm) [6]. В этом статическом алгоритме балансировщик облака в первую очередь вычисляет самые короткие задания и назначает их самым производительным узлам. Недостатком этого алгоритма является то, что задания, имеющие максимальное время выполнения, должны ждать неопределённый период времени, что может повлечь скопление задач в очереди. Также некоторые узлы могут стоять не полностью нагруженными: мелкие задачи выполняются, а свободных ресурсов, несмот-

ря на их потенциально большое количество, недостаточно для обработки больших задач.

5) Алгоритм балансировки нагрузки max-min (Max-Min Load Balancing Algorithm) [22]. Max-min алгоритм является статическим и работает так же, как min-min алгоритм, за исключением того, что после выявления минимального времени выполнения задач балансировщик облака занимается задачами, имеющими максимальное время выполнения. Алгоритм выбирает задачу с максимальным временем выполнения и назначает её узлу, на котором достигается минимальное время её завершения. После чего время выполнения оставшихся задач для этого узла обновляется.

В основном на практике у популярных облачных платформ, таких как Microsoft Azure, Amazon EC2, Google Cloud Platform применяются простые статические алгоритмы или их незначительные модификации [25, 26, 27]. Это связано с тем, что бóльшую часть нагрузки создаёт большое количество мелких запросов, хорошо балансируемых такими алгоритмами как min-min, round-robin и оппортунистическим. Также разработчиками облака решается проблема отсутствия отказоустойчивости, присутствующая во всех алгоритмах. Для каждого облака это делается по-своему, в основном архитектурными методами.

3.3.6. Начальная балансировка

Для начального распределения контейнеров по Cloud Simulation применяется алгоритм round-robin. В начале работы системы агенты равномерно распределены по контейнерам типа Дом, поэтому применение этого алгоритма приводит к равномерному распределению и агентов по различным контейнерам. Применение этого алгоритма в данном случае позволяет эффективно и быстро произвести распределение.

3.3.7. Балансировка во время работы системы

В процессе работы системы агенты перемещаются между контейнерами, тем самым переходя с одного узла на другой. Вследствие этого возникает необходимость перераспределять агентов между узлами. Для этого применяется модифицированный алгоритм балансировки min-min, распределяющий агентов по работающим узлам. Этот алгоритм, в случае перегрузки какого-либо узла, выбирает минимальный по количеству находящихся в нём агентов контейнер и перемещает его на минимально загруженный узел. Таким образом, можно сформулировать следующий алгоритм вытеснения контейнеров:

1. если количество агентов на каком-либо из CS значительно больше среднего значения, тогда происходит вытеснение контейнера с минимальным количеством агентов.
2. на GD отправляется сообщение с контейнером, из CS контейнер удаляется.
3. GD находит минимально загруженный CS и отправляет его туда. Информация, необходимая для третьего пункта, отправляется во время синхронизации времени с CS на GD.

Именно алгоритм min-min, выбранный за основу, хорошо подходит для нашей задачи: остальные алгоритмы сделали бы процесс затруднительным, поскольку выбранный для переноса контейнер может оказаться слишком большим и его перемещение заняло бы очень большое время (в случае с round-robin или оппортунистическим), а с применением алгоритма max-min это произошло бы гарантировано.

3.4. Эксперименты

На рисунках 7, 8 представлены сравнения результатов работы модели за 20 дней с реальными данными и с результатом модели [4]:

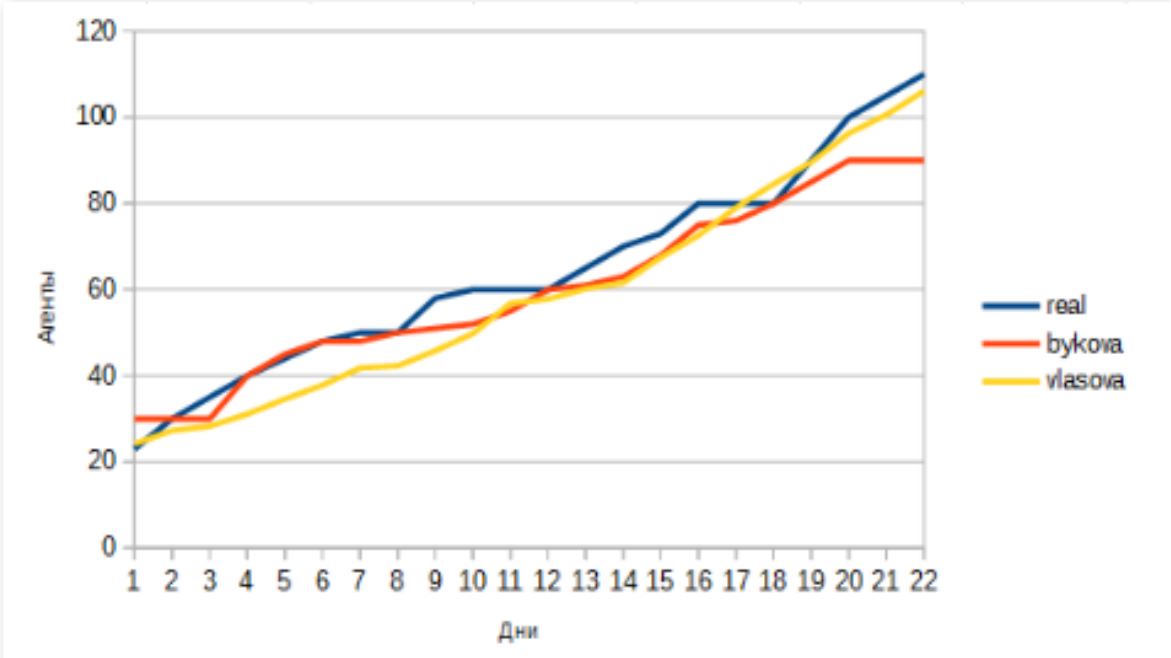

Рис. 7. График заражённости.

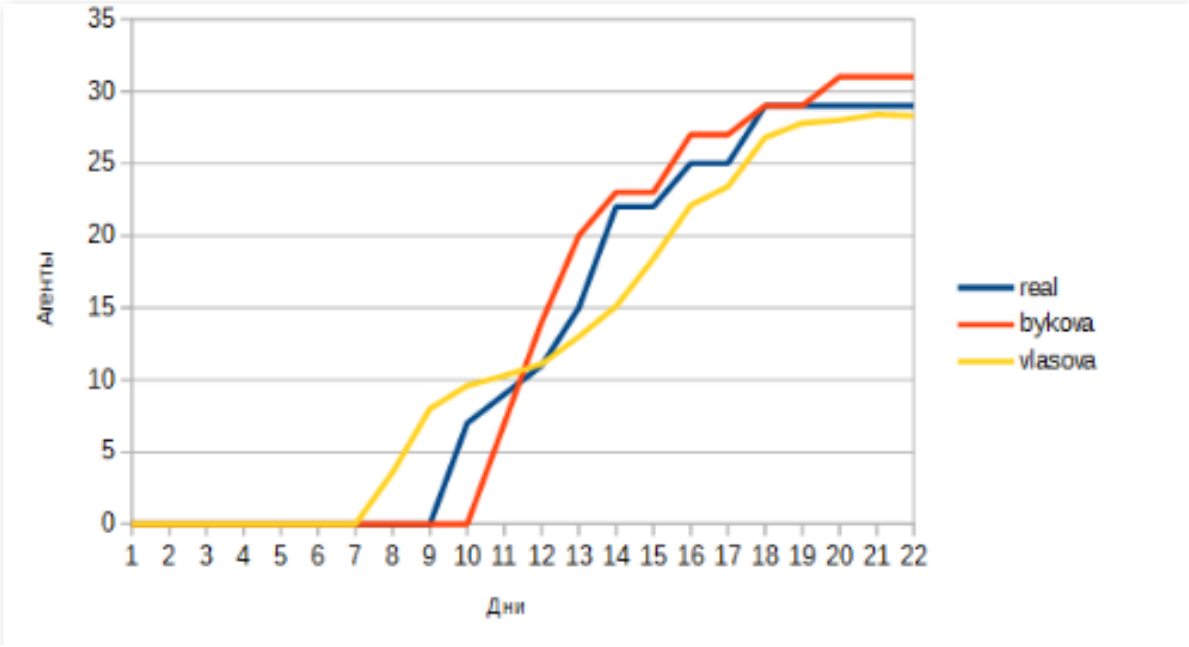

Рис. 8. График смертности.

3.5. Ограничения исследования

Для проведения моделирования крайне желательно выполнение условия обособленности рассматриваемых населённых пунктов в целях удовлетворения контейнерному подходу. При необходимости моделирования

другого заболевания также стоит учесть, что оно должно удовлетворять модели SEIRFD, то есть иметь ярко выраженные формы протекания.

3.6. Дальнейшие пути развития

Данная модель может быть расширена на другие заболевания, удовлетворяющие SEIRFD. Также можно провести моделирование на различных территориях с большей плотностью населения — на это система и рассчитана. В дополнение к этому является возможным добавление новых уточняющих параметров, позволяющих наиболее точно описать модель.

4. Заключение

При создании модели распространения заболевания был выбран агентный подход, позволяющий детализировать модель, учитывать различные особенности агентов. Адаптация мультиагентной системы к облачной среде сделала возможным преодолеть ограничения, связанные с вычислительной мощностью компьютера, а балансировка нагрузки в начале и в процессе работы позволила избежать перегрузки виртуальных машин и производить вычисления надёжнее и оптимальнее.

Моделирование эпидемиологии позволит внести свой вклад в разработку и анализ эпидемиологических исследований, определить тенденцию, сделать общие прогноз, а эластичность облачного решения при таком подходе позволило найти оптимальную по соотношению цена-качество модель.

Список литературы

1. Макаров В.Л., Бахтизин А.Р. Компьютерное моделирование искусственных миров – URL: http://www.xjtek.ru/file/212
2. Паринов С.И. Новые возможности имитационного моделирования социально-экономических систем. — Т. 2, № 3–4. — С. 26–61, 2007.
3. Кондратьев М.А. Разработка модели распространения инфекционных заболеваний на основе агентного подхода в AnyLogic – 2012 г.
4. Быкова Ю.С. МАС в AnyLogic – 2015 г. – URL: se.math.spbu.ru/SE/diploma/2015/s/544-Bykova-report.pdf
5. Pooja Samal, Pranati Mishra. Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing. Pooja Samal et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 (3) , 2013 г., 416 - 419 – URL: http://www.ijcsit.com/docs/Volume%204/vol4Issue3/ijcsit2013040306.pdf
6. Kokilavani T., Amalarethinam G. Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing. International Journal of Computer Applications (0975 – 8887) Volume 20– No.2,India. April 2011 – URL: http://www.ijcaonline.org/volume20/number2/pxc3873197.pdf
7. Bykov S., Geller A., Kliot G., Larus J., Pandya R., Thelin J. – URL: http://research.microsoft.com/pubs/141999/pldi%2011%20submission%20public.pdf
8. Кузнецов К.О. Облачная мультиагентная платформа на основе JADE и Google App Engine – URL: http://www.math.spbu.ru/ru/mmeh/Magistr/2013/sp/kuznetsov_diss.doc
9. Centers for Disease Control and Prevention – URL: http://www.cdc.gov/vhf/ebola/
10. URL: http://www.tvc.ru/news/show/id/67407
11. http://www.who.int/mediacentre/factsheets/fs103/ru/

12. Sierra Leone: Kailahun District Profile (3 December 2015) http://reliefweb.int/sites/reliefweb.int/files/resources/district_profile_kailahun_10_dec_2015am_0.pdf
13. Brauer F., Castillo-Chavez C. Mathematical Models in Population Biology and Epidemiology. — Springer-Verlag New York, 2001.
14. Rivers C. M., Lofgren E. T. Modeling the Impact of Interventions on an Epidemic of Ebola in Sierra Leone and Liberia. — URL: http://currents.plos.org/outbreaks/article
15. Борщев А. В. Практическое агентное моделирование и его место в арсенале аналитика // Exponenta Pro. Математика в приложениях. — 2004. — № 3–4. — С. 38–47.
16. Public Health Dynamics Laboratory. Framework for Reconstructing Epidemiological Dynamics. — URL: http://fred.publichealth.pitt.edu/
17. Alexander G. Bachinsky and Lily Ph. Nizolenko, "A Universal Model for Predicting Dynamics of the Epidemics Caused by Special Pathogens", BioMed Research International, vol. 2013, Article ID 467078, 7 pages, 2013. doi:10.1155/2013/467078
18. Centers for Disease Control and Prevention (U.S.). Generic Ebola response (ER) : modeling the spread of disease impact intervention. — URL: https://stacks.cdc.gov/view/cdc/24900
19. Brauer F., Castillo-Chavez C. Mathematical Models in Population Biology and Epidemiology — New York: Springer-Verlag New York, 2001. — 448 p.
20. Jade site – URL: http://jade.tilab.com/
21. Poison message – URL: https://msdn.microsoft.com/en-us/library/ms789028(v=vs.110).aspx
22. Elzeki O., Reshad M., Elsoud M. Improved Max-Min Algorithm in Cloud Computing. International Journal of Computer Applications (0975 – 8887) Volume 50 – No.12, Egypt, July 2012 – URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.

258.6082&rep=rep1&type=pdf

23. Sandeep Sharma, Sarabjit Singh, Meenakshi Sharma. Performance Analysis of Load Balancing Algorithms. World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:2, No:2, 2008 – URL: http://waset.org/publications/5537/performance-analysis-of-load-balancing-algorithms
24. Che-Lun Hung, Hsiao-hsi Wang, Yu-Chen Hu. Efficient Load Balancing Algorithm for Cloud Computing Network – URL: http://onlinepresent.org/proceedings/vol2_2012/80.pdf
25. Anand Chaudhari, Anushka Kapadia.Load Balancing Algorithm for Azure Virtualization with Specialized VM's. India, June 2013 – URL: http://ijiet.com/wp-content/uploads/2013/07/35.pdf
26. Amazon EC2 Documentation – https://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/how-elb-works.html#request-routing
27. Google Cloud Platform Documentation – https://cloud.google.com/compute/docs/load-balancing/network/#load_distribution_algorithm
28. Azure Servise Bus – URL: https://azure.microsoft.com/en-us/documentation/articles/service-bus-fundamentals-hybrid-solutions/
29. The Liberia Ministry of Health. Data for the 2014 ebola outbeak in West Africa. –– URL: https://github.com/cmrivers/ebola/
30. Centers for Disease Control and Prevention. Signs and Symptoms – URL: http://www.cdc.gov/vhf/ebola/symptoms/