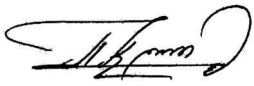
ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

На правах рукописи



Кочовски Петар

РАЗРАБОТКА ТЕХНОЛОГИИ ОБЕСПЕЧЕНИЯ ВЫСОКОСКОРОСТНЫХ ВЫЧИСЛЕНИЙ НА ОСНОВЕ ОБЛАЧНОЙ ИНФРАСТРУКТУРЫ

Специальность
05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Санкт Петербург
2021

Работа выполнена в федеральном государственном автономном образовательном учреждении высшего образования «Санкт-Петербургский политехнический университет Петра Великого».

Научный руководитель: **ДРОБИНЦЕВ Павел Дмитриевич**, кандидат технических наук, доцент, директор высшей школы программной инженерии

Официальные оппоненты: **КОЗНОВ Дмитрий Владимирович**, доктор технических наук, профессор, профессор кафедры системного программирования, ФГБОУ ВО «Санкт-Петербургский государственный университет»

АБРАМСКИЙ Михаил Михайлович, кандидат технических наук, доцент, директор института информационных технологий и интеллектуальных система, ФГАОУ ВО «Казанский (Приволжский) федеральный университет»

Ведущая организация: ФГАОУ ВО «Самарский национальный исследовательский университет имени академика С.П.Королева»

Защита состоится 16 сентября 2021 г. в 16 часов на заседании диссертационного совета У.05.13.11 при федеральном государственном автономном образовательном учреждении высшего образования «Санкт-Петербургский политехнический университет Петра Великого» в удаленном интерактивном режиме на платформе MS Teams: https://bit.ly/3xQ5qlx.

С диссертацией можно ознакомиться в библиотеке ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого» и на сайте https://www.spbstu.ru/science/the-department-of-doctoral-studies/defences-calendar/the-degree-of-candidate-of-sciences/kochovski_petar_/.

Автореферат разослан «__» _____ 2021 года.

Ученый секретарь
диссертационного совета
У.05.13.11,
кандидат технических наук, доцент Дробинцев Павел Дмитриевич

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. С появлением концепции Интернета вещей возникла необходимость создания приложений в различных областях, таких как умный дом, умные города и населенные пункты, промышленность 4.0 и подобных, требующих высокой вычислительной мощности и быстрой передачи данных. Следуя стандартам разработки программного обеспечения для архитектуры микросервисов, эти приложения могут быть построены как набор модульных сервисов, которые можно виртуализировать в форме виртуальных машин или контейнеров. Следовательно, они могут быть развернуты по всему вычислительному спектру, от облачных центров обработки данных до вычислительных узлов на периферии.

При реализации Интернета вещей в настоящее время существует богатый выбор различного рода программного обеспечения. При этом на основе различных сценариев использования программное обеспечение запрашивает удовлетворения различных требований, включая требования предъявляемые к качеству обслуживания. Примером такого программного обеспечения является приложение для раннего предупреждения. Чтобы своевременно прогнозировать и предотвращать катастрофу, такое приложение взаимодействует с множеством датчиков и должно быстро обрабатывать большие объемы данных и адекватно отреагировать. С другой стороны, в контексте Интернета вещей, также могут быть разработаны приложения для хранения и управления данными, или даже для видеоконференций WebRTC в реальном времени и других применений. Из-за огромной разницы между сценариями все эти приложения имеют разные требования к качеству. Например, система раннего предупреждения потребует высокой скорости отклика, приложение хранения потребует большой емкости хранилища, а приложение для видеоконференций в реальном времени потребует высокой пропускной способности и низкой задержки. Поэтому, принятие решения о том где развернуть программное обеспечение, которое должно удовлетворять требованиям к высокой вычислительной производительности, в континууме от узла до облака, где много различных доступных инфраструктур для развертывания, является особенно сложным.

Прежде чем принять решение о развертывании, разработчик программного обеспечения должен рассмотреть различные требования, связанные с качеством обслуживания, включая большой выбор инфраструктур, различные технологии виртуализации, географическое распределение, качество сети (например, задержка, пропускная способность) и другие требования на уровне приложения и пользователя. Имея в виду

количество доступных инфраструктур и большое количество требований к качеству, разработчик программного обеспечения практически не может вручную выбрать оптимальную инфраструктуру для развертывания программного обеспечения.

Следовательно, появляется необходимость в разработке новых подходов, методов и технологий, для улучшения процесса принятия решений и подбора адекватных и оптимальных или близким к оптимальным решений по развертыванию приложений от облачных центров обработки данных до вычислительных узлов на периферии.

Степень разработанности темы. На сегодняшний день, существует множество исследований, в которых изучался выбор оптимальных поставщиков облачных инфраструктур с учетом ожидаемого качества обслуживания. Во всех рассмотренных исследованиях, таких как: J. Hu, L.E. Li, M. Randles, Z. Chaczko, S.S. Manvi, B. Jennings, N.C. Luong, N. Jain, S. Singh, S. Chaisiri, L. Zhang, предлагаются многокритериальные методы принятия решений, в которых облачные вычисления рассматриваются как детерминированный процесс. В отличие от рассмотренных исследований, настоящая работа рассматривает природу облачных вычислений как стохастическую из-за неопределенности надежности инфраструктуры, ее доступности, предоставления ресурсов, предоставления услуг и так далее. Кроме того, эта работа представляет технологию, автоматизирующую процесс развертывания на основе метода, обеспечивающего оптимальное решение на основе большого количества критериев, в частности таких, как: местоположение инфраструктуры, стоимость, пропускная способность, качество пользовательского опыта и т.п., снижая вычислительную сложность процесса принятия решения и предоставляя формальные гарантии на соответствие входным требованиям разработчика программного обеспечения.

Целью диссертационного исследования является разработка методов и средств для балансировки использования ресурсов и увеличения вычислительной производительности облачной системы с помощью автоматического ранжирования доступных облачных инфраструктур. Для достижения поставленной цели должны быть решены следующие **задачи:**

1. Создание методики выбора оптимальной облачной инфраструктуры;
2. Создание методики разделения облачных инфраструктур на классы эквивалентности с целью сокращения вычислительной сложности процесса принятия решения;
3. Создание методики оценки корректности результата с проверкой выполнения обязательных ограничений качества (жесткие

ограничения) и выполнения максимального количества необязательных ограничений качества (мягкие ограничения);
4. Объединение предложенных методик в формализованный эффективный метод развертывания программного обеспечения в облачных инфраструктурах;
5. Разработка программного комплекса на основе предложенного метода, автоматизирующего процесс развертывания приложений в облачных инфраструктурах с использованием нефункциональных требований для достижения высокого качества обслуживания;
6. Выполнение экспериментального исследования результатов применения программного комплекса на реальных сценариях развертывания облачных приложений.

Постановка цели и задач исследования соответствует следующему пункту паспорта специальности 05.13.11: Модели, методы, алгоритмы и программная инфраструктура для организации глобально распределенной обработки данных.

Научная новизна состоит в следующем:
1. Разработана методика выбора оптимальной облачной инфраструктуры на основе Марковского процесса принятия решения;
2. Предложена методика сокращения вычислительной сложности путем формирования классов эквивалентности на основе выбранных ограничений нефункциональных требований;
3. Предложена методика оценки корректности результата на основе темпоральной логики;
4. Предложен формализованный метод для автоматизации процесса развертывания приложений в облачных инфраструктурах, позволяющий использовать нефункциональные требования для достижения высокого качества обслуживания.

Практическая значимость работы. На базе полученных научных результатов был разработан комплекс программных средств, предназначенных для автоматического ранжирования доступных облачных инфраструктур и развертывания приложений в облачных инфраструктурах. Программный комплекс разработанный в рамках диссертации был использован в рамках международного проекта *BI-RU/16-18-043 Internet of Things and Cloud computing as support for the development of new smart approaches in the construction sector (Интернет вещей и облачные вычисления как поддержка разработки новых интеллектуальных подходов в строительном секторе),* европейских проектов *DECENTER-Decentralised*

technologies for orchestrated Cloud-to-Edge intelligence (*Децентрализованные технологии для управления интеллектуальными вычислениями от облака до периферии*, ID проекта: 815141) и *SWITCH-Software Workbench for Interactive, Time Critical and Highly self-adaptive Cloud applications* (*Программное обеспечение для разработки интерактивных, критичных ко времени и самоадаптирующихся облачных приложений*, ID проекта: 643963). Созданные программные средства могут быть использованы для обеспечения и гарантии качества обслуживания приложений во время выполнения на облачных инфраструктурах различными категориями пользователей.

Объектом исследования являются технологии обеспечения высокоскоростных вычислений на основе облачной инфраструктуры.

Предметом исследования являются нефункциональные требования для высокоскоростных и качественных вычислений, методы принятия решений в средах облачных вычислений, типы инфраструктур облачных вычислений и технологии облачных вычислений.

Методология и методы исследования. Исследование включает использование теории множеств, логики предикатов первого порядка, темпоральной логики, методов принятия решений, а также методов планирования и обработки результатов эксперимента. Для разработки предложенного решения применены инструменты и подходы программной инженерии.

Основные положения, выносимые на защиту:

1. Предложена методика выбора оптимальной облачной инфраструктуры на основе Марковского процесса принятия решения;
2. Предложена методика сокращения вычислительной сложности путем формирования классов эквивалентности на основе выбранных ограничений нефункциональных требований;
3. Предложена методика оценки корректности результата на основе темпоральной логики;
4. Разработан формализованный метод для автоматизации процесса развертывания приложений в облачных инфраструктурах, позволяющий использовать нефункциональные требования для достижения высокого качества обслуживания. Разработано программное обеспечение, реализующее предложенный метод.

Достоверность результатов и сформулированных выводов обеспечивается корректным применением математического аппарата,

полнотой выполненных теоретических и практических исследований. Положения получили положительную оценку на научных конференциях, а также внедрены в практику. По сравнению с работами других авторов, решение, разработанное в рамках диссертации: 1) показало лучшие результаты в отношении вычислительной сложности процесса принятия решения, 2) достигло более высокой точности результата с точки зрения количества нарушений требований к качеству, заданных пользователем, 3) достигло более высокого уровня технологической готовности и технология была внедрена в три проекта, где использовалась для обеспечения автоматизированной оркестровки ресурсов с учетом требований к качеству от периферии к облаку.

Апробация результатов работы. Основные результаты исследований и научных разработок обсуждались на конференциях: международная конференция 33rd IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019, Рио-де-Жанейро (Бразилия); международная конференция IEEE World Congress on Services, IEEE SERVICES 2019, Милан (Италия); международная конференция IEEE Computer Society - XXVII International Conference on Information, Communication and Automation Technologies (ICAT 2019), Сараево (Босния и Герцеговина).

Публикации. Основные результаты по теме диссертации изложены в восьми научных работах, в том числе три статьи в научных журналах, три статьи в сборниках докладов конференции и один раздел книги, входящих в перечень Scopus, а также одна статья в научном журнале, входящем в перечень ВАК.

Личный вклад. Содержание диссертации, включая все основные результаты, выносимые на защиту, получены автором самостоятельно. Из работ, выполненных в соавторстве, в диссертацию включены результаты, которые соответствуют личному участию автора.

Структура и объем диссертации. Диссертация состоит из введения, 6 глав, заключения и списка литературы из 87 наименований. Объем диссертации 96 страниц текста, в том числе 15 рисунков и 14 таблиц.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обоснована актуальность темы исследования, определены цели и задачи, представлена научная новизна и практическая значимость, представлены выносимые на защиту научные положения, описан личный вклад автора, приведен список публикаций автора по теме диссертации.

В **первой главе** диссертации представлены результаты обзора литературы. На основе анализа источников рассмотрены основные парадигмы облачных вычислений, рассмотрены принципы виртуализации в облачных вычислениях, описаны нефункциональные требования и их отношение к надежности облачных вычислений. Кроме того, представлен обзор наиболее часто используемых методов принятия решений в средах облачных вычислений. В **разделе 1.1** рассмотрена парадигма облачных вычислений и продемонстрированы различия между различными типами вычислений от периферии до облака. В данном разделе определены понятия «облачное вычисление», «туманное вычисление» и «периферийное вычисление». В **разделе 1.2** рассмотрены современные технологии виртуализации, такие как: виртуальные машины и контейнеры, позволяющих более эффективно использовать вычислительные ресурсы. Кроме того, в этом разделе рассматриваются нефункциональные требования в облачных вычислениях. Дается определение понятии «нефункциональное требование». Также проводится обзор наиболее важных нефункциональных требований, используемых для достижения высокого качества облачных вычислений. В **разделе 1.3** рассмотрены механизмы выбора оптимальных облачных инфраструктур с учетом ожидаемого качества обслуживания, например, исследования, связанные с балансировкой нагрузки, управлением и распределением ресурсов, предоставлением ресурсов и созданием систем развертывания и управления услугами. Из-за большого количества различных нефункциональных требований, в обзорных исследованиях были предложены многокритериальные методы принятия решеный для различных сценариев развертывания в облачных инфраструктурах. Предложенные методы в большинстве случаев основываются на методе анализа иерархий и генетическом алгоритме недетерминированной сортировки.

На основе анализа в этой главе, были сделаны следующие выводы о текущем положении дел в области исследования:

- Хотя облачные инфраструктуры имеют решающее значение для высокоскоростных вычислений, процесс разработки программного обеспечения не зависит от облаков в том смысле, что приложение может быть разработано до выбора облачной инфраструктуры;
- Нефункциональные требования играют важную роль в удовлетворении высоких стандартов качества;
- Нефункциональные требования облачных приложений сильно различаются, поэтому их необходимо учитывать на этапе развертывания в облаке;
- Из-за большого количества различных нефункциональных требований, которые можно учитывать, в соответствующих исследованиях предлагаются подходы многокритериального

принятия решений для сценариев развертывания приложений в облачных инфраструктурах.

Учитывая тот факт, что наиболее перспективными методами принятия решения, с точки зрения стохастичности облачных инфраструктур, являются марковские процессы принятия решения, следует отметить что в настоящее время отсутствуют технологии, которые на основе Марковского процесса принятия решений, автоматизируют процесс развертывания приложений на оптимальной облачной инфраструктуре, учитывая конкретные требования и контекст использования приложений или программных компонентов, виртуализированных как образы контейнера.

Таким образом, в данной главе была рассмотрена соответствующая теме исследования литература, определены преимущества и недостатки предшественников и обоснован выбор направления исследования.

Во **второй главе** представлена методика выбора оптимальной облачной инфраструктуры на основе Марковского процесса принятия решений. Глава предоставляет информацию о генерации вероятностного автомата, такую как вычисление вероятностей переходов, вычисление наград состояний и вычисление полезности состояний, которые опираются на марковский процесс принятия решений. Марковский процесс принятия решений – это вероятностная структура $M= (S, A, P, R, \gamma)$, где:

- S – конечное множество состояний;
- A – конечное множество действий;
- P – стохастическая матрица переходных вероятностей;
- R – вознаграждение, получаемое после перехода в состояние S' из состояния S в результате действия a;
- γ – коэффициент дисконтирования, который имеет значение в интервале [0,1].

Раздел 2.1 описывает вероятностную модель, которая используется Марковским процессом принятия решений. В **разделе 2.2** представлен способ вычисления вероятностей для каждого перехода и вознаграждений за достижение каждого состояния в вероятностной модели. Переходные вероятности изображаются в виде слоев в вероятностном дереве решений. Первый уровень показывает, что вероятности достижения какой-либо доступной инфраструктуры, когда нет данных мониторинга или знаний о предыдущем использовании, одинаковы. Учитывая, что, N_{tran} является количеством переходов одного действия из каждого состояния в остальные состояния, а n является количеством состояний в автомате, матрица будет иметь следующий вид:

$$P_1 = (P1_{ij}), \tag{1}$$

где $P1_{ij} = \frac{1}{N_{tran}}, i = \overline{1,n}, j = \overline{1,n}$

Второй уровень представляет вероятности перехода между состояниями в автомате, которые рассчитываются при наличии знаний об использовании перераспределения микросервисов между инфраструктурами. Учитывая, N_{chosen} - количество раз, которое инфраструктура выбиралась в качестве цели перераспределения микросервиса из другой инфраструктуры, а N_{listed} - количество раз, когда инфраструктура была указана в классе эквивалентности, матрица будет иметь следующий вид:

$$P_2 = (P2_{ij}), \quad \text{где } P2_{ij} = \frac{N_{chosen}}{N_{listed}}, i = \overline{1,n}, j = \overline{1,n} \tag{2}$$

В общем, если автомат обладает знаниями о предыдущем использовании, матрицу вероятностей окончательного перехода можно рассчитать, как произведение матриц P_1 и P_2:

$$P = P_1 \cdot P_2 \tag{3}$$

Значение вознаграждения для каждого состояния зависит от данных, полученных в результате измерений мониторинга. Чем больше мягких ограничений n_{sc} нарушаются конкретным состоянием (то есть инфраструктурой), тем ниже будет вознаграждение за достижение этого состояния. Учитывая, $count_i$ - количество нарушений, вектор вознаграждений можно рассчитать с помощью следующего уравнения:

$$R(S_i) = \begin{cases} \frac{1}{count_i}, & count_i > 0 \\ 1, & count_i = 0 \\ 0, & count_i = n_{sc} \end{cases} \tag{4}$$

Раздел 2.3 описывает процесс принятия решения. Чтобы рассчитать вектор рейтинговых баллов и определить оптимальную инфраструктуру, необходимо оценить полезность каждого состояния используя уравнения Беллмана:

$$U_i(S) = R(S) + \gamma max_{a \in A(S)} \sum_{S'} P(S'|S,a) U(S_{i+1}) \tag{5}$$

где S является текущем состоянием, а S' является следующим состоянием. Таким образом в данной главе описана методика выбора оптимальной

облачной инфраструктуры на основе Марковского процесса принятия решения.

В **третьей главе** предложена методика сокращения вычислительной сложности. В **разделе 3.1** предоставляется формальное определение классов эквивалентности, основанных на выбранных жестких ограничениях. Класс эквивалентности — это подмножество больших наборов элементов, удовлетворяющих точно определенным жестким ограничениям, которые также известны как отношения эквивалентности. **Раздел 3.2** описывает методику сокращения вычислительной сложности на основе классов эквивалентности. Классом эквивалентности $[e]$ называется множество всех элементов множества инфраструктур $I = \{inf_0, \ldots, inf_m\}$, находящихся в отношении ~ к элементу e, то есть:

$$[e] = \{inf \in I \mid inf \sim e\} \tag{6}$$

Отношение эквивалентности ~ определяется как:

$$\forall i \exists j (nr_j^{inf} = nr_i^e), \tag{7}$$

где nr является элементом подмножества атрибутов качества в множестве $NFR = \{\{nr_{00}, nr_{01}, \ldots, nr_{0n}\}, \ldots, \{nr_{m0}, nr_{m1}, \ldots, nr_{mn}\}\}$. Таким образом в данной главе описана методика сокращения вычислительной сложности путем формирования классов эквивалентности.

В **четвертой главе** представлена методика оценки корректности результата. Оценка корректности результата состоит из верификации предоставленных результатов метода принятия решения, а также оценки вероятности действий по повторному развертыванию микросервисов в некоторых случаях в будущем. Процесс верификации заключается в проверке того, были ли выполнены обязательные ограничения качества (жесткие ограничения) и максимальное количество необязательных ограничений качества (мягкие ограничения). **Раздел 4.1** описывает подход оценки корректности результата на основе проверки на модели (Model Checking). **Раздел 4.2** описывает свойства формулируемые в темпоральной логике PCTL, которые используются для оценки корректности результата. В данной работе с использованием PCTL основным критерием проверки является следующее:

$$P_{=1}[G(hard)\&F(soft)] \tag{8}$$

где $hard$ – конъюнкция всех жестких ограничений:

$$hard = hConstraint_1 \,\&\, hConstraint_2 \,\&\, \dots \,\&\, hConstraint_m, \qquad (9)$$

и *soft* – дизъюнкция всех мягких ограничений:

$$soft = sConstraint_1 \,||\, sConstraint_2 \,||\, \dots \,||\, sConstraint_n. \qquad (10)$$

Для того чтобы сузить проверку, критерий может быть максимизирован следующим образом:

$$P_{=1}[G(hard)\&F(soft)\,\&(nfrViolations \leq minViolations)], \qquad (11)$$

где *nfrViolations* – это количество нефункциональных требований, которые не удовлетворяются текущими инфраструктурами развертывания, а *minViolations* – это минимальное количество нарушений нефункциональных требований среди всех возможных инфраструктур, включенных в модель. Таким образом в данной главе описана методика оценки корректности результата на основе темпоральной логики.

В **пятой главе** описывается формализованный метод для автоматизации процесса развертывания приложений в облачных инфраструктурах. Нередко на практике, после разработки программного обеспечения разработчики должны вручную выбрать инфраструктуру, в которой будет размещено разработанное программное обеспечение. Поэтому часто эти инфраструктуры не являются оптимальными хостами для программного обеспечения, что приводит к снижению качества обслуживания. Чтобы решить эту проблему, в данной работе предлагается концепция процесса автоматического ранжирования всех доступных вычислительных инфраструктур с учетом нефункциональных требований и уникального контекста использования микросервисов. В **разделе 5.1** описывается концепция процесса автоматического ранжирования инфраструктур и развертывания приложений. Концепция состоит из пяти общих шагов (см. Рисунок 1), где входными параметрами являются все доступные инфраструктуры, данные мониторинга инфраструктуры и требования к качеству заданные разработчиками программного обеспечения; а выходными данными является список ранжированных инфраструктур, где первая инфраструктура считается оптимальной для работы программного обеспечения. Подробное объяснение рабочего процесса приведено в пунктах ниже.

Шаг 1: На этом этапе программный инженер разрабатывает облачное приложение из контейнерных микросервисов, которые необходимо

развернуть в облаке. Он выбирает все важные нефункциональные требования (например, конкретная геолокация, стоимость и т.д.). Кроме того, программному инженеру дается возможность определить пороговые значения для любых метрик качества (например, пропускная способность, задержка, производительность и т.д.). Тем не менее, этот выбор может значительно отличаться для различных типов микросервисов. Другими словами, инженер должен также решить, какие требования представляют жесткие ограничения и должны постоянно выполняться во время выполнения, а какие требования являются желательными, но не обязательными (мягкие ограничения). Как только жесткие и мягкие ограничения определены, они используются на двух заключительных этапах автоматизированного процесса принятия решений. Жесткие ограничения используются в качестве входных параметров для классификации эквивалентности (второй шаг), в то время как мягкие ограничения используются на этапе генерации и проверки вероятностной модели (третий шаг).

Рисунок 1 - Концепция процесса принятия решений для развертывания микросервисов

Шаг 2: На этом шаге уменьшается количество необходимых вычислений при определении оптимальной облачной инфраструктуры. Для

дальнейших вычислений рассматриваются только инфраструктуры, которые удовлетворяют жестким ограничениям разработчика программного обеспечения. В начале этого этапа, генерируется конечный автомат с использованием списка всех доступных инфраструктур для развертывания облачного приложения (например, 1000 доступных облачных инфраструктур), где каждое состояние этой исходной модели представляет один вариант развертывания. Затем следует автоматизированный процесс, в котором инфраструктуры классифицируются по классам эквивалентности. Например, класс эквивалентности может содержать все инфраструктуры, которые содержат минимум 8 процессоров, или все инфраструктуры, для которых загрузка центрального процессора составляет менее 80%.

Шаг 3: Целью этого шага является создание вероятностного автомата, который представляет собой вероятностную модель случайно-изменяющихся систем. Для построения вероятностной модели используются различные метрики качества обслуживания, которые представляют прошлое и настоящее состояние производительности инфраструктур. Они собираются и хранятся в базе данных с использованием многоуровневой системы мониторинга. В процессе используются только те нефункциональные требования, которые являются частью жестких и мягких ограничений. Затем метод вычисляет рейтинговые оценки для всех инфраструктур и ранжирует их.

Шаг 4: После генерации вероятностной модели полученный результат проверяется с использованием метода проверки на модели (model checking). Этот шаг, используя формальные критерии, проверяет, насколько нефункциональные требования удовлетворяются доступными инфраструктурами в каждом классе эквивалентности. Рассчитанное значение проверки модели является выходом вероятностной модели и предоставляет формальную гарантию.

Шаг 5: Инфраструктура первого ранга автоматически выбирается и микросервис развертывается с помощью инструмента оркестровки, такого как Kubernetes. Этот этап выполняется в предположении, что полученная формальная гарантия для инфраструктуры первого ранга, то есть той, которая имеет наивысшую оценку, приемлема для разработчика программного обеспечения.

В **разделе 5.2** предложена системная архитектура программного комплекса для развертывания микросервисов на оптимальной облачной инфраструктуре (Рисунок 2). Программный комплекс, разработанный в ходе диссертации, основан на использовании существующих методов принятия

решений, верификации и мониторинга. Многоуровневая архитектура, соответствует стандартам совместимости, установленным организациями, Cloud Native Computing Foundation (CNCF), Edge Computing Consortium Europe (ECCE) и OpenFog Consortium. Разработанный программный комплекс состоит из графического интерфейса, модулей принятия решений и «Edge-to-Cloud» вычислений для развертывания контейнерных микросервисов.

Рисунок 2 - Системная архитектура высокого уровня

Графический интерфейс – это входная точка для разработчика программного обеспечения. В графическом пользовательском интерфейсе программный инженер разрабатывает приложение из контейнерных микросервисов, управляет требованиями к качеству и входными параметрами для процесса развертывания и инициирует процесс развертывания.

Модуль принятия решения отвечает за определение оптимальной инфраструктуры для развертывания контейнерных микросервисов на основе Марковского процесса принятия решения. В данном модуле также происходит верификация решения о развертывании и анализ возможных сценариев перераспределения микросервисов из одной инфраструктуры в другую в некоторый момент времени в будущем. Кроме того, модуль также включает в себя оркестратор, который запускается для развертывания приложения после получения оптимальной инфраструктуры и верификации решения.

Модуль вычисления «Edge-to-Cloud» состоится из инфраструктур, для развертывания контейнерных микросервисов и данных, компонентов мониторинга, компонентов оркестровки инфраструктур и IoT устройств. Инфраструктуры в этом модуле используются для хранения и обработки данных в вычислительном континууме. В зависимости от цели и требований развертываемых программных компонентов, предложенная архитектура позволяет разворачивать контейнеры программных компонентов на периферии (Edge), инфраструктурах в тумане (Fog) и в инфраструктурах в облаке (Cloud).

В **разделе 5.3** описывается механизм принятия решений о развертывании. Модуль принятия решений разбивается на две системы компонентов: система Марковского процесса принятия решений и верификации и система мониторинга (Рисунок 3). Первый компонент Генератор Классов Эквивалентности (ГКЭ), инициирует работу механизма. Этот компонент распределяет все доступные инфраструктуры и создает начальную модель, которая используется в качестве входного параметра для создания классов эквивалентности. Затем он проверяет, какие инфраструктуры удовлетворяют жестким ограничениям, и назначает их элементам класса эквивалентности. Полученный класс эквивалентности направляется в компонент Генератор Вероятностной Модели (ГВМ). ГВМ генерирует конечный вероятностный автомат, размер которого равен размеру набора, который был предоставлен с ГКЭ. ГВМ рассчитывает значения вероятностей переходов, награды состояний и полезности состояний. Оба компонента, ГКЭ и ГВМ, разработаны с использованием технологий разработки на основе Java, таких как Java Jersey для RESTful веб-сервисов и Apache Maven для управления программным обеспечением.

На основе вероятностной логики ветвящегося времени (PCTL), Верификатор Решений (ВР) проверяет модель и выходные параметры компоненты ГВМ. Таким образом, ВР проверяет степень, в которой оптимальная инфраструктура будет удовлетворять нефункциональным

требованиям микросервиса в соответствии с конкретными ограничениями, которые были установлены программным инженером. Следовательно, этот компонент обеспечивает инженеру уверенность в том, что система выделяет оптимальную инфраструктуру для работы микросервиса. Этот компонент использует средство проверки модели PRISM, которое применяется для анализа вероятностных моделей.

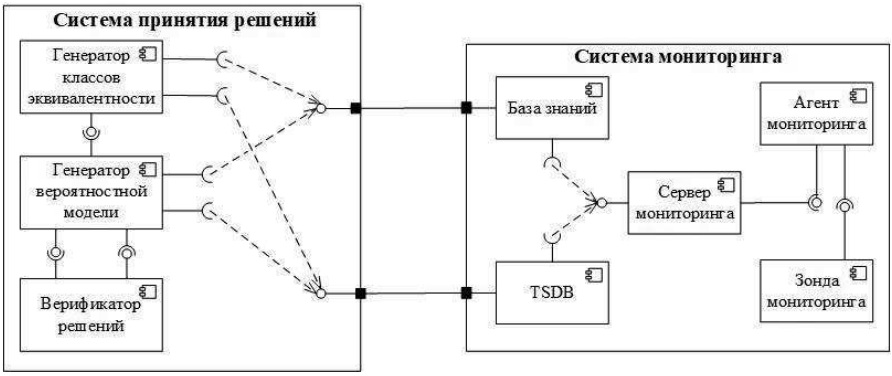


Рисунок 3 - Диаграмма компонентов

Правильное функционирование вероятностной вычислительной логики сильно зависит от системы мониторинга. Система мониторинга представляет собой набор компонентов мониторинга, таких как зонды для метрик различных атрибутов, агентов и сервера, которые динамически собирают метрики во время работы приложения. Она играет важную роль в предлагаемой системе, так как используется для измерения метрик (например, пропускной способности, задержки, загрузки процессора и памяти), и это гарантирует, что любые микросервисы удовлетворяют необходимым требованиям к качеству обслуживания во время выполнения. В качестве систем мониторинга используются Jcatascopia и Prometheus. Агенты мониторинга (Monitoring Agent) являются легкими компонентами, которые управляют сбором метрик из виртуальных экземпляров машин и контейнеров. Зонды мониторинга (Monitoring Probes) являются компонентами, созданными для сбора низко- и высокоуровневых метрик. Сервер мониторинга (Monitoring Server) используется для сбора отслеживаемых данных от агента мониторинга и передачи их в базу данных. База данных временных рядов — это база данных, которая используется для хранения метрик QoS. В экспериментах данной диссертации используется Apache Cassandra с открытым исходным кодом. База данных временных рядов (TSDB) представляет собой базу данных, которая используется для

хранения метрик о качестве обслуживания. База знаний используется для сбора комплексной информаций, которая требуется ГКЭ и ГВМ в качестве входных параметров. База знаний Apache Jena Fuseki собирает информацию о выбранной инфраструктуре, такую как: геолокация инфраструктуры, информация о типе приложения, оценка качества опыта и информация о развертывании. В **разделе 5.4** описывается процесс автоматического развертывания приложений, который начинается после того, как система принятия решений определила и проверила инфраструктуру развертывания. В этой работе для автоматического развертывания приложений реализуется механизм оркестровки ресурсов Kubernetes.

Таким образом, в данной главе описывается предлагаемый сценарий и формализованный метод, который автоматизирует процесс развертывания приложений в облачных инфраструктурах, позволяющий использовать нефункциональные требования для достижения высокого качества обслуживания.

Шестая глава посвящена созданию и экспериментальному исследованию разработанных методик программного комплекса для принятия решений о развертывания микросервисов на оптимальной облачной инфраструктуры. Предложенные методики и средства были экспериментально протестированы на работоспособность с тремя различными микросервисами, которые предъявляют высокие требования к качеству: загрузка файлов, видеоконференцсвязь и использование базы данных. Цель практической реализации состоит в том, чтобы доказать, что предлагаемая методика является полностью функциональной и что, используя множество ограничений к достижению качества, методика может обеспечить автономное развертывание контейнерных приложений на оптимальных инфраструктурах, которые удовлетворяют требованиям качества. В состав методов и средств технологии, используемых в тестировании, входили:

1. методика сокращения вычислительной сложности путем формирования классов эквивалентности на основе выбранных ограничений нефункциональных требований;
2. методика выбора оптимальной облачной инфраструктуры на основе Марковского процесса принятия решения;
3. методика оценки корректности результата на основе темпоральной логики.

В контексте экспериментальной оценки выбранных методик необходимо было ответить на следующие вопросы:

1. на сколько сократилось среднее время выполнения всего процесса принятия решения путем формирования классов эквивалентности?
2. удовлетворяют ли инфраструктуры, которые были приняты во внимание для развертывания, жестким ограничениям, установленным разработчиком программного обеспечения?
3. удовлетворяет ли оптимальная инфраструктура для развертывания приложения максимальному количеству мягких ограничений?
4. сколько было нарушений порогов к качеству обслуживания у предложенной методики выбора оптимальной облачной инфраструктуры?

Раздел 6.1 описывает сценарии тестирования. В **разделе 6.1.1** описан сценарий реализации, который определяет оптимальную инфраструктуру для микросервиса загрузки больших файлов в облачные инфраструктуры File Upload. Хотя операция загрузки файлов представляет собой довольно простой сценарий использования, она имеет строгие требования к характеристикам сети. Поэтому такие атрибуты, как местоположение, потеря пакетов, пропускная способность и задержка, являются важными требованиями к качеству, которые должны быть выполнены. В **разделе 6.1.2** описан сценарий реализации, который определяет оптимальную инфраструктуру для микросервиса, обеспечивающего видеоконференций WebRTC. Микросервис был разработан на основе технологии с открытым исходным кодом Jitsi Meet. В этом сценарии контейнер микросервиса развертывается в выбранной инфраструктуре, где он запускается, и уничтожается для каждого сеанса видеоконференцсвязи. В этом сценарии используются те же атрибуты NFR, что и в File Upload сценарии. Из-за различного характера микросервисов, различные пороговые значения используются в качестве ограничений. В **разделе 6.1.3** описан сценарий реализации, который определяет оптимальную инфраструктуру для развертывания контейнеров базы данных Apache Cassandra под различными рабочими нагрузками, такими как: 50000, 200000, 500000 операций чтения/записи. В **разделе 6.2** описывается экспериментальная среда, которая использовалась при практической реализации предложенных методик. В **разделе 6.3** приводятся выводы, полученные на основе анализа результатов эксперимента. В **разделе 6.4** приводится сравнение и соотнесение предлагаемого решения с общепринятыми методами, рассмотренными в разделе 1.3. Экспериментальные исследования показали работоспособность метода. Именно, был сделан вывод что: (1) методика сокращения вычислительной сложности путем формирования классов эквивалентности

сократила среднее время выполнения всего процесса до 62%; (2) во всех сценариях тестирования, все инфраструктуры развертывания, которые были приняты во внимание удовлетворяли всем жестким ограничениям установленным разработчиком программного обеспечения; (3) во всех сценариях тестирования, выбранная инфраструктура развертывания удовлетворяла максимальному количеству мягких ограничений; (4) методика выбора оптимальной облачной инфраструктуры имеет тенденцию выбирать инфраструктуру развертывания с минимальным количеством нарушений пороговых значений по сравнению с часто используемым методом МАИ. В **заключении** приведены выводы по всей работе и планы дальнейших исследований.

Таким образом в данной главе представлены выводы на основе результатов экспериментального исследования разработанных методик программного комплекса для принятия решений о развертывания трех микросервисов на оптимальных облачных инфраструктурах.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

В диссертационной работе поставлена и решена проблема обеспечения высокого уровня качества в рамках организации вычислений «Edge-to-Cloud», путем удовлетворения критерия качества, заданного программным инженером для определения оптимальной вычислительной инфраструктуры. В ходе выполнения диссертационной работы получены следующие результаты:

1. предложена методика группировки инфраструктур, удовлетворяющих жестким ограничениям качества, определенным программным инженером;
2. предложена методика определения оптимальной инфраструктуры для контейнерного приложения, которая удовлетворяет мягким ограничениям, определенным программным инженером, на основе текущих измерений инфраструктуры и предшествующих знаний об использовании;
3. предложена методика проверки оптимального решения с помощью PCTL логики и оценки вероятностей возможных сценариев перераспределения приложений в будущем на основе предыдущих знаний об использовании;
4. проведена оценка эффективности предложенных методик на базе использования с различными типами приложений, с различными требованиями к качеству обслуживания;
5. разработана системная архитектура, объединяющая вышеупомянутые методики в систему, позволяющую автоматически

развертывать программные компоненты в «Edge-to-Cloud» континууме, гарантируя высокое качество обслуживания и исключение человеческих ошибок, возникающих при ручном развертывании;

6. разработан программный комплекс, прошедший апробацию в рамках трех проектов, автоматизирующей процесса развертывания приложений в инфраструктурах от периферии до облаков, с гарантией высокого качества обслуживания.

Результаты, полученные в процессе выполнения проектов, использующих методики данной диссертации, позволили сделать выводы о работоспособности разработанных методов.

Дальнейшие исследования целесообразно проводить в области расширения базы знаний информацией о минимальных требованиях к качеству различных типов приложений. В таком случае классы эквивалентности будут группировать доступные инфраструктуры по типу приложения, что позволит разработчику программного обеспечения выбрать тип приложения, а система предоставит ему оптимальную инфраструктуру для развертывания.

ПУБЛИКАЦИИ АВТОРА ПО ТЕМЕ ДИССЕРТАЦИИ

Публикации в зарубежных научных изданиях, индексируемых в системах ВАК, Scopus или Web of Science:

1. Kochovski, P. An approach for automated deployment of cloud applications in the «Edge-to-Cloud» computing continuum satisfying high Quality of Service requirements / P. Kochovski, P.D. Drobintsev // Computing Telecommunications and Control. — 2020. — Т. 13, № 1. — С. 8-18.

2. Kochovski, P. Trust management in blockchain based fog computing platform with trustless smart oracles / P.Kochovski, S. Gec, V. Stankovski, M. Bajec, P.D. Drobintsev // Future Generation Computer Systems. — 2019. — № 101. — С. 747-759.

3. Kochovski, P. Formal Quality of Service assurances, ranking and verification of cloud deployment options with probabilistic model checking method / P. Kochovski, P.D. Drobintsev, V. Stankovski // Information and Software Technology. — 2019. — № 109. — С. 14-25.

4. Kochovski, P. Supporting smart construction with dependable edge computing infrastructures and applications / P. Kochovski, V. Stankovski // Automation in Construction. — 2018. — № 85. — С. 182-192.

5. Kochovski, P. Dependability of Container-Based Data-Centric Systems / P. Kochovski, V. Stankovski // В кн.: Security and Resilience in Intelligent Data-Centric Systems and Communication Networks. Academic Press. — 2018. — С. 7-27.

Публикации в сборниках трудов конференций:

6. Kochovski, P. An Architecture and Stochastic Method for Database Container Placement in the Edge-Fog-Cloud Continuum / P. Kochovski, R. Sakellariou, M. Bajec, P.D. Drobintsev, V. Stankovski // 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). Rio de Janeiro. — 2019. — С. 396-405.

7. Kochovski, P. A Smart and Safe Construction Application Design for Fog Computing / P. Kochovski, M. Bajec, R. Sakellariou, V. Stankovski // 2019 IEEE World Congress on Services (SERVICES). Milan. — 2019. — С. 378-379.

8. Holste, B. Blockchain Based Variability Management Solutions for Fog Native Open Source Software / B. Holste, V. Stankovski, P. Kochovski, A. Puliafito, P. Massonet // 2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT). Sarajevo. — 2019. — С. 1-6.