УДК 004.05

МЕТОД ПРИНЯТИЯ РЕШЕНИЙ ПО УПРАВЛЕНИЮ РИСКАМИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ В ПУБЛИЧНЫХ ОБЛАКАХ

Лаврентьев А.В., аспирант, E-mail: lavrentev@hush.com
Кирюхин В.А., студент. E-mail: v.a.kir@yandex.ru
МГТУ МИРЭА, Москва, Россия

Аннотация. В статье предложен метод, позволяющий автоматизировать процесс выбора конфигураций облачных систем, оптимальных по стоимости итоговой системы и рискам информационной безопасности, укладываясь при этом в заданные ограничения на качество обслуживания. Показана возможность применения этого метода к задаче противодействия EDoS-атакам. Поиск дешевых облачных конфигураций осуществляется посредством решения задачи линейного программирования, а оценка рисков информационной безопасности проводится на основе предложенной теоретико-игровой модели. Приведены результаты экспериментов, показывающие эффективность разработанного метода в смысле достижения оптимальных соотношений скорости анализа и итоговых значений рисков информационной безопасности.
Ключевые слова: облачные вычисления, теория игр, линейное программирование, Парето-оптимальность, EDoS-атаки.

IFORMATION SECURITY RISK MANAGEMENT METHOD IN PUBLIC CLOUD SYSTEMS

Lavrentev A.V., postgraduate student, Email: lavrentev@hush.com
Kiryukhin V.A., student, Email: v.a.kir@yandex.ru
MSTU MIREA, Moskow, Russia

Abstract. In article, the high-level decision method is proposed in order to automatize selection process of cloud systems configurations that are optimized by information security risks and system costs saving within a set of restrictions on quality of service. Application of this method to a real problem is shown on the example of counteraction to EDoS-attacks. Search of cheap cloud configurations is carried out by means of a problem of linear programming, and the assessment of risks is based on game-theoretic model. The results of experiments are showing efficiency of the method in terms of selection speed and resulting information security risks values.
Keywords: cloud computing, game theory, linear programming, Pareto-optimality, EDoS-attacks.

1. Введение.

В корпоративной среде сложные и повторяющиеся информационные процессы часто реализуются как отдельные программные приложения.

В связи с распространенностью сервисно-ориентированных архитектур такие приложения часто представляют собой композиции многократно используемых сервисов, формирующих поддержку технологических процессов компаний-владельцев. Собственные аппаратные средства хранения и обработки могут быть сильно

ограничены, что может привести к тому, что качество выполнения технологического процесса становится неустойчивым к пиковым нагрузкам и, кроме того, увеличивается время обработки. Если даже соответствующие ресурсы доступны в достаточных объемах, то они могут простаивать вне периодов пиковых нагрузок, и тем самым ресурсы тратятся непроизводительно.

Облачные вычисления рассчитаны на изменяемые потребности в необходимой для выполнения определенных операций производительности. Они обеспечивают необходимое качество обслуживания и низкую стоимость архитектуры посредством использования масштабируемых вычислительных ресурсов, предоставляя их по требованию и оплачивая в объемах их использования. Несмотря на эти преимущества, массовое использование публичных облаков в качестве идеального инструментария выполнения массовой обработки информации замедляют новые свойственные исключительно им угрозы безопасности [1]. Хранение данных и приложений для технологических процессов в публичных облаках часто означает, что управление безопасностью полностью переходит из рук клиента в область ответственности поставщика облачных услуг. Это не позволяет гарантировать выполнение политики информационной безопасности владельца данных.

Реализация приложений, поддерживающих технологические процессы в публичных облаках, должна осуществляться с наименьшими затратами, необходимыми для поддерживания требуемой политики информационной безопасности (ИБ) владельца процессов. Как следствие, предприятия, пользующиеся услугами облачных провайдеров, считают, что обеспечение безопасности и соответствие политикам ИБ в публичных облаках осуществляется не достаточно прозрачно. Высокая динамичность изменения параметров качества обслуживания облачных систем, недостаточность контроля и прозрачности обеспечения ИБ требуют разработки новых оптимизационных методов и технологий, позволяющих создавать эффективные стратегии управления информационной безопасностью технологических процессов.

Кроме того, управление использованием облачных ресурсов и выбор стратегии развертывания публичного облака являются сложными ресурсоемкими процессами, т.к. требуют от лиц, принимающих решения, рассмотрения огромного множества предоставляемых облачными провайдерами альтернатив. Текущие облачные технологии и их стоимостные модели могут быть настолько комплексными и отличными друг от друга, что поиск дешевых решений, гарантирующих требуемое качество обслуживания, является крайне трудоемкой задачей. Архитектор приложений, поддерживающих технологические процессы, сталкиваясь с подобным вопросом

должен рассмотреть большое количество вариантов и быть в состоянии оценить затраты и качество обслуживания для каждого из них. Хотя существуют некоторые аналитические модели приблизительной оценки качества работы программных систем в распределенных системах, в большинстве своем они не соответствуют специфике функционирования облачной среды.

Конечно, имеются классические модели, разработанные с целью предсказания поведения статичных систем и ориентированные на постоянные во времени параметры, такие как уровень рабочей нагрузки, конфигурацию оборудования и показатели качества. Динамичность облачных систем делает подобные методы неприменимыми к ним. При моделировании таких систем для оценки их качества, стоимости и рисков ИБ требуется учитывать параметры, крайне зависимые от времени (эластичность облачных архитектур подразумевает значительные суточные колебания ресурсов, поддерживающих выполняемые задачи). Очевидно, что это порождает дополнительные трудности, связанные с появлением проблем реализации процессов автоматизации и вычислительной эффективности при проведении анализа пространства возможных конфигураций облачных вычислительных сред для решения конкретных прикладных задач.

Эта работа посвящена разработке метода, позволяющего в автоматическом режиме получать оптимальные соотношения параметров механизмов безопасности облачных систем и стоимостных характеристик итоговой прикладной системы с учетом изменяемости параметров операционных нагрузок и качества обслуживания пользователей.

2. Метод управления рисками ИБ в публичных облачных системах.

Задача управления рисками информационной безопасности в публичных облачных системах может быть сформулирована как проблема многокритериального принятия решений. Целью разрабатываемого метода является нахождение множества решений, оптимальных с точки зрения достижения требуемых характеристик качества обслуживания, стоимости системы и рисков информационной безопасности.

Предложим метод поиска таких решений. Он состоит из реализации трех основных этапов (Рис. 1): выбор характеристик качества обслуживания, поиск на основе решения задачи линейного программирования оптимальной по стоимости системы, теоретико-игровая оценка рисков ИБ.

1) На основе требуемых функциональных характеристик облачной среды формируется пространство всех доступных альтернатив.

Рис. 1. **Метод принятия решений по управлению рисками ИБ.**

2) Ограничения характеристик качества выбираются в пределах верхней и нижней допустимых границ. Как правило, нижняя граница описывает наихудшее значение качества обслуживания, при котором система сохраняет свою функциональность (например, отклик на запрос пользователя к системе происходит не дольше 1 секунды). Верхняя граница заведомо фиксирована и связана с возможностями аппаратных средств, используемых для реализации анализируемой облачной среды.

3) Посредством использования оптимизационного алгоритма, минимизирующего стоимость, ищется облачная система, удовлетворяющая выбранным ограничениям качества.

4) На основе применения теоретико-игровой модели с двумя игроками оцениваются риски информационной безопасности.

5) Найденное решение добавляется в итоговое множество оптимальных решений. Значения качества, стоимости и рисков сравниваются со значениями ранее найденных решений. Решения, не соответствующие условиям Парето-оптимальности, исключаются.

6) В зависимости от результатов предыдущего шага изменяется ограничение на характеристики качества обслуживания и происходит возврат к пункту 3, либо алгоритм завершает свою работу.

По результатам работы этого алгоритма лицо, принимающее решения, получает список Парето-оптимальных облачных конфигураций, соответствующих заданным ограничениям качества обслуживания, и может в зависимости от своей склонности к риску выбрать более дорогое, но менее рисковое решение или наоборот.

3. Применение предложенного метода к выбору параметров механизмов защиты при противодействии EDoS-атакам.

Уникальные возможности облачных вычислений привели к появлению новых критических угроз информационной безопасности. Одной из них является атака EDoS [2, 3] (Economical Denial of Sustainability), свойственная исключительно облачной среде. Она заключается в том, что злоумышленник, желающий нарушить работу приложения, размещенного в облаке, формирует большое количество запросов, что приводит к увеличению объема используемых облачных ресурсов и, как следствие, к ускоренному расходованию денежных средств владельца приложения. В момент, когда денежные средства заканчиваются, работа системы приостанавливается, что и является целью злоумышленника. Если же владелец приложения устанавливает ограничение на использование его ресурсов, то большое количество запросов приводит к значительному снижению качества обслуживания (выраженному, например, в параметре доступности приложения или среднем времени ответа на запрос).

Чаще всего в Service Level Agreement (SLA) между владельцем приложения и облачным провайдером указывается, что провайдер обязан обнаруживать такие атаки и возмещать потерянные ресурсы. Между тем при реализации атаки злоумышленнику достаточно посылать запросы, ничем не отличающиеся от легитимных, что приводит к новым нерешенным задачам их обнаружения и к трудностям с доказательством того, что была осуществлена атака, а не увеличен поток запросов легальных пользователей.

Это приводит к необходимости включения в процесс анализа облачных конфигураций оценок их устойчивости к подобным атакам.

3.1. Модель доверенной среды.

Для примера иллюстрации работоспособности метода рассмотрим приложение Apache Open For Business [4] (далее - OfBiz). Рисунки 2 и 3 отражают необходимую информацию об этом приложении, представленную посредством UML-подобного языка PCM [5].

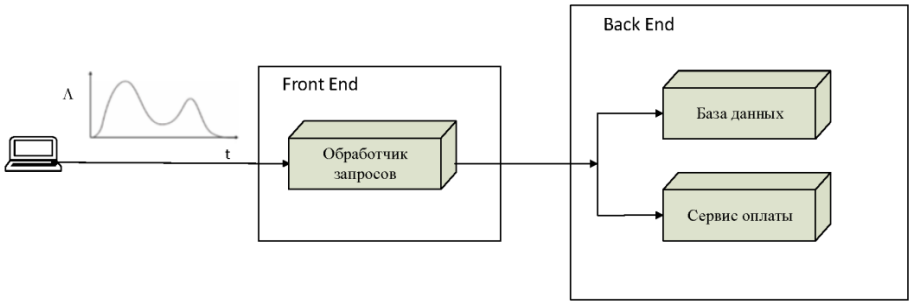

Рис. 2. **Приложение OfBiz.**

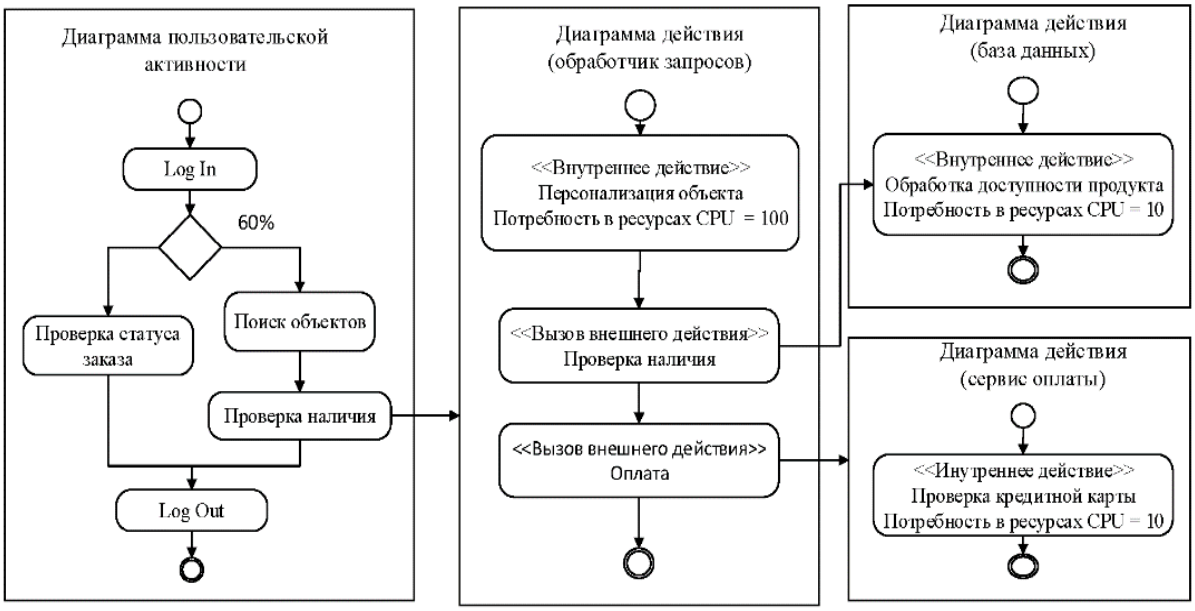

Рис. 3. **Диаграммы активности пользователей в приложении OfBiz.**

Левая диаграмма рисунка 3 показывает обычное поведение пользователей при работе с этим приложением. В данном примере 60% пользователей используют приложение для оформления покупки некоторого продукта, в то время как оставшиеся 40% проверяют статус доставки продукта, заказанного ранее. Входящая рабочая нагрузка представлена в виде количества запросов в секунду. Она задается отдельно для каждого из 24 часов. Все запросы, сформированные пользователями, обрабатываются компонентом <Обработчик запросов>. Поведение функционала <Проверка наличия> описывается диаграммой деятельности, соответствующей обработчику. Для выполнения запроса front-end виртуальной машине требуется взаимодействие с некоторыми компонентами, расположенными на back-end виртуальной машине, и внутренние вычисления (например, для получения цены продукта), влияние которых на физические ресурсы хоста обозначено <Потребность в ресурсах CPU>. В нашем примере запрос обращается к компоненту <База данных> для проверки доступности выбранного продукта и к компоненту <Сервис оплаты> для

проверки корректности информации о кредитной карте, введённой пользователем. Рисунок 2 демонстрирует размещение компонентов между виртуальными машинами.

Представим процесс поиска оптимальной конфигурации облачной среды (без учета действий злоумышленника) как решение проблемы использования частично-целочисленного линейного программирования, в которой качество обслуживания развертываемого решения вычисляется посредством M/G/1-PS модели очередей. Такая модель позволяет вычислить среднее время ответа на запрос в закрытой форме. Целью оптимизации будет нахождение облачной конфигурации минимальной цены при условии сохранения гарантированного качества обслуживания, выраженного средним временем ответа. В настоящей работе под стоимостью понимается сумма цен арендованных ресурсов, оформленных на условиях оплаты по факту использования. В этом случае виртуальная машина арендуется на час, по истечении которого ее использование может быть снова продлено и оплачено, либо она будет отключена. Этот тип ценовой политики является наиболее общим для всех облачных провайдеров.

Приложение может быть представлено как композиция нескольких компонентов C, каждый из которых соответствует некоторому множеству классов запросов K с заданной потребностью в ресурсах $D_{k,c}$. Каждый компонент размещен в некотором пуле ресурсов (уровне архитектуры) *I*, сформированном множеством гомогенных виртуальных машин. Такое множество не статично, но может масштабироваться в соответствии с изменением входящей рабочей нагрузки. Т.к. дневная рабочая нагрузка периодична для многих приложений, дальнейший анализ будет проводиться с учетом временного горизонта в один день. Многие облачные провайдеры взимают почасовую оплату за пользование ресурсами, так что будет разумно разбить временной горизонт на 24 части по одному часу каждая. Для простоты в дальнейшем будем полагать, что ограничения на время ответа для оптимизируемого приложения заданы заранее.

Пользователи взаимодействуют с приложением, формируя запросы. Каждый класс запросов характеризуется вероятностью его выполнения α_k и множеством компонентов, его поддерживающих (т.е. расположенных на пути выполнения). Частоту использования классом *k* компонента *c* обозначим $p_{k,c}$. Наконец, будем предполагать, что запросы обрабатываются согласно типичной политике планирования PS (processor sharing), которая равномерно распределяет нагрузку между всеми виртуальными машинами в пуле. Требования к качеству обслуживания в модели представлены задаваемым разработчиком приложения ограничением на среднее время ответа $\{R\}$.

Типы виртуальных машин V характеризуются стоимостью C_v и скоростью $Speed_v$. Примем за опорную виртуальную машину (ВМ) с наименьшей скоростью. Рассчитаем для ВМ интенсивность обслуживания каждого класса запросов каждым компонентом $\mu_{k,c}$ (максимальное количество обрабатываемых запросов в секунду). Она равна отношению скорости опорной ВМ к требуемой потребности в ресурсах $D_{k,c}$. Теперь, введя дополнительный параметр S_v, равный отношению скорости ВМ типа *v* к скорости опорной, можем выразить интенсивность обслуживания ВМ типа *v* как $S_v\mu_{k,c}$.

Доступное время ответа $\{R\}$ для запроса распределим между классами запросов и компонентами следующим образом. Для каждого класса запросов *k* и каждого компонента *c* рассчитаем совокупную ожидаемую потребность в ресурсах. Затем для каждого класса распределим $\{R\}$ между компонентами пропорционально ожидаемой потребности в ресурсах. Обозначим через γ матрицу $|K|x|C|$ пропорций, а $\{R\}_{k,c}$ – итоговые ограничения. Очевидно, что $\{R\}_{k,c} := \gamma_{k,c}\{R\}$.

Задача оптимизации заключается в выборе типа *v* и количества виртуальных машин $N_{i,v,t}$ для каждого пула ресурсов. Она может быть сформулирована как:

$$\min \sum_{i\in I}\sum_{v\in V}\sum_{t\in T} C_{i,v}N_{i,v,t}, \tag{1}$$

при условии:

$$\sum_{v\in V} w_{i,v} = 1 \quad \forall i \in I, \tag{2}$$

$$w_{i,v} \le N_{i,v,t} \le \overline{N}\, w_{i,v} \quad \forall i \in I, v \in V, t \in T, \tag{3}$$

$$\sum_{v\in V}\left(S_v - \frac{1}{\mu_{k,c}\{R\}_{k,c}}\right)N_{i,v,t} \ge G_{c,t}\Lambda_t \quad \forall c \in C, i \in I_c, t \in T, k \in K. \tag{4}$$

Здесь I_c – пул ресурсов, поддерживающий компонент *c*, а

$$G_{c,t} = \sum_{\tilde{c}\in I_c}\sum_{\tilde{k}\in K}\frac{\alpha_{\tilde{k}}p_{\tilde{k},\tilde{c}}}{\mu_{\tilde{k},\tilde{c}}} \tag{5}$$

Первые два условия означают, что для каждого пула ресурсов будет выбран только один тип ВМ. Третье условие гарантирует сохранение необходимого качества обслуживания. Оно получено с помощью простых арифметических операций из

известной в теории массового обслуживания формулы для времени ответа очереди M/G/1:

$$\{R\}_{k,c} \geq R_{k,c} = \frac{\frac{1}{\mu_{k,c} S_v}}{1 - \frac{\Lambda_t}{S_v N_{i,v,t}} \sum_{\tilde{c} \in I_c, \tilde{k} \in K} \frac{p_{\tilde{k},\tilde{c}} \alpha_{\tilde{k}}}{\mu_{\tilde{k},\tilde{c}}}} \quad (6)$$

Следует отметить, что предложенная модель линейна и может быть эффективно проанализирована с помощью специализированных решателей (например, CPLEX [6]).

Обозначим как OP(R) указанную оптимизационную задачу, где R – матрица размера $|K| x |C|$, элементами которой являются вводимые ограничения $\{R\}_{k,c}$. Обозначим через $C*$ и $X*$ полученные итоговые стоимость и конфигурацию.

3.2. Теоретико-игровая оценка риска EDoS-атак.

Типичной стратегией противодействия EDoS-атакам является разбиение пользователей на классы на основе их поведения. Например, пользователи, совершающие покупки, относятся к классу «хороших», а пользователи, которые просто просматривают сайт, к классу «сомнительных». Пользователи класса «сомнительных» получают дополнительную задержку перед ответом на их запросы. Очевидно, что при этом легитимные пользователи, которые могли бы в будущем совершить покупку, на текущий момент также попадают в класс «сомнительных». Поэтому значения параметров стратегии должны выбираться оптимально, чтобы не слишком снижать удовлетворенность пользователей качеством сервиса.

Эта стратегия была предложена в 2011 году в [3] и остается единственной, в достаточной мере исследованной международным научным сообществом. Тем не менее, до сих пор не было предложено методов, позволяющих количественно рассчитать параметры данной стратегии, например, значение задержки и/или выражение для условия, при котором она должна применяться.

Осуществим оценку одного из этих параметров, а также значения рисков EDoS на основе теоретико-игровой модели. Пусть есть два игрока: архитектор приложения PD и злоумышленник PA. Злоумышленник использует класс запросов k для проведения атаки, в то время как архитектор приложения выбирает компонент, при взаимодействии с которым будет применена политика разделения пользователей на классы. Для этого перед игрой разработчик приложения составляет множество допустимых альтернатив DS. Тогда в терминах теории игр для игрока PA (злоумышленника) чистыми

стратегиями будут элементы множества K (т.е. классы запросов), а чистыми стратегиями для игрока PD (архитектора приложения) – элементы множества DS.

Для каждой конкретной пары стратегий игроков посредством использования оптимизационной модели раздела 3.1 данной работы может быть получена соответствующая оценка риска, отражающая изменения стоимости системы под действием атаки. Для этого зафиксируем типы виртуальных машин $w_{i,v}$ и подставим измененные значения α_k и $\{R\}_{k,c}$. Пусть Q -итоговая стоимость новой системы. Тогда значение риска для данной пары стратегий равно $Z := Q - C*$, где $C*$ - стоимость исходной системы в условиях доверенной среды. Рассчитав значения для каждой пары стратегий получим матрицу выплат игроков $Z^{|DS|x|K|}$.

Решение игры будем искать в смешанных стратегиях, т.е. результатом игры будет пара распределений вероятностей (по одному для каждого игрока), определяющая вероятности выборов игроками тех или иных чистых стратегий. Концепция равновесия Нэша в смешанных стратегиях подразумевает, что ни один участник не сможет увеличить свой выигрыш, изменив свое решение в одностороннем порядке, когда другие участники не меняют своего решения [7]. Согласно известной теореме Нэша в матричной игре всегда существует хотя бы одно равновесие Нэша в смешанных стратегиях. Рассматриваемая матричная игра двух игроков является одним из самых простых классов игр, для которых было разработано множество эффективных методов поиска решений.

Цена игры, т.е. ожидаемое значение выплаты игроков, если они будут играть найденные стратегии равновесия, является выражением для итогового риска $Z*$. Обозначим $GP(\{PD; PA\}; \{DS; K\}; Z^{|DS|x|K|})$ игру оценки рисков EDoS-атак. Тогда $(Z*; P*)$ – итоговый риск и стратегия разработчика соответственно.

3.2. Алгоритм поиска Парето-оптимальных конфигураций облачной среды при условии EDoS-атак.

Наконец, сформулируем алгоритм поиска Парето-оптимальных конфигураций (Рис. 4).

Шаг 0. Построим множество допустимых альтернатив DS. Пусть Pareto – множество искомых Парето-оптимальных альтернатив. Сделаем его пустым. Вычислим матрицу γ. Пусть R^l и R^u – нижняя и верхняя границы допустимых значений времени ответа на запросы пользователей соответственно. Примем исходное значение $R*$ равным R^u.

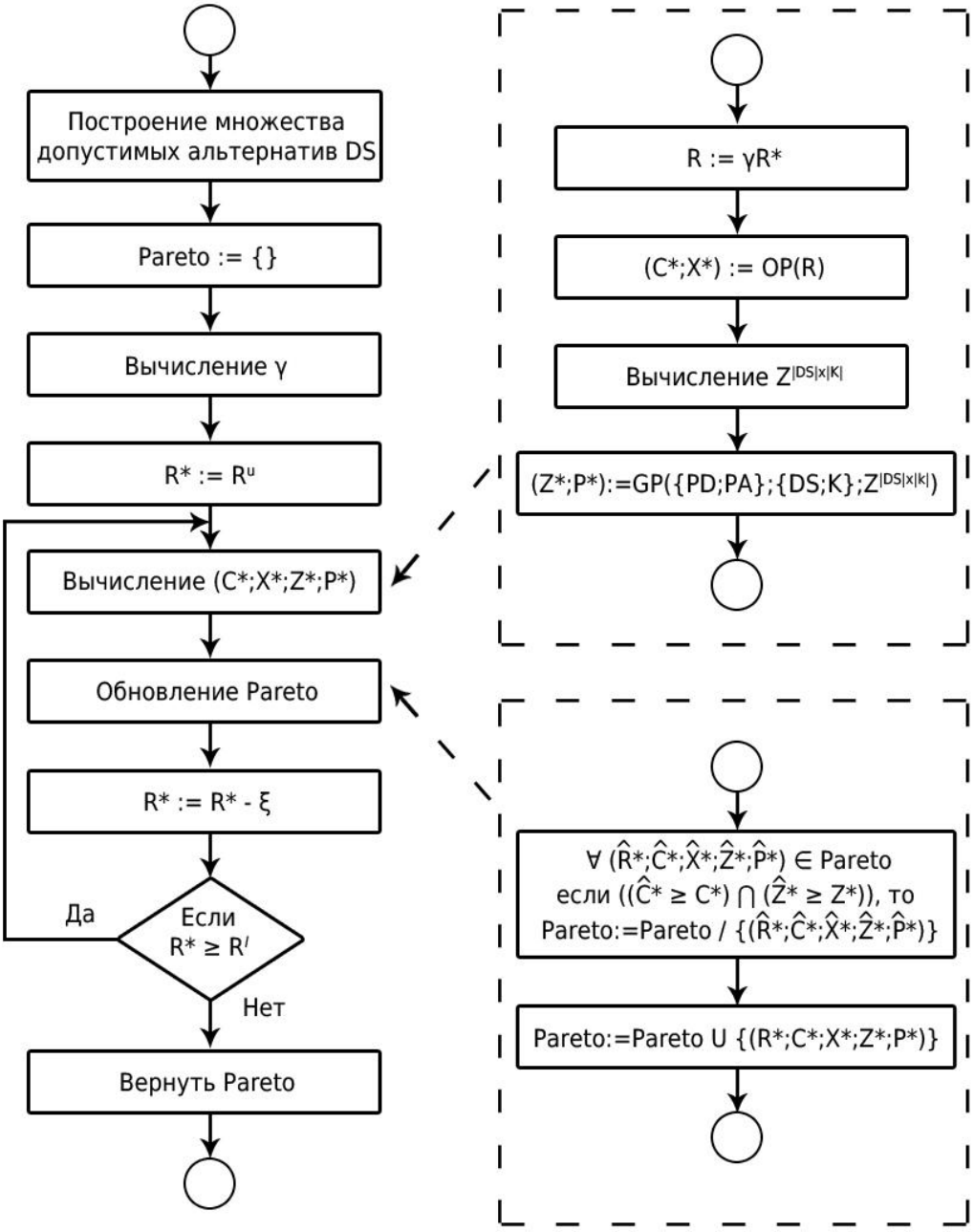

Рис. 4. **Алгоритм получения множества Парето-оптимальных конфигураций.**

Шаг 1. Для выбранного $R*$ вычислим матрицу R ограничений времени ответа для компонентов и классов. Найдем с помощью решения оптимизационной задачи частично-целочисленного линейного программирования OP стоимость $C*$ и конфигурацию $X*$.

Шаг 2. Вычислим $Z^{|DS|x|K|}$. Посредством построения теоретико-игровой модели GP найдем значение риска $Z*$ и параметры безопасности $P*$.

Шаг 3. Удалим из множества Pareto все конфигурации, стоимость и риски которых больше C^* и Z^* соответственно. Добавим в него элемент $(R^*, C^*, X^*, Z^*, P^*)$.

Шаг 4. Уменьшим R^* на величину шага поиска ξ. Если R^* больше или равен минимальному значению R^l, то вернуться к шагу 1. Иначе завершаем алгоритм, множество Pareto содержит искомые конфигурации.

3.4. Экспериментальные результаты.

Мы оценили предложенный метод на 200 различных примерах, отражающих реальные облачные приложения. Кроме того, мы использовали очень большое множество случайно сгенерированных распределений рабочих нагрузок и системных параметров моделей, изменяя их согласно диапазонам, представленным в таблицах 1 и 2. Стоимости виртуальных машин и их скорости были взяты из реальных предложений рынка публичных облаков.

Параметр	**Диапазон**
α_k	[0.1;1] %
$p_{k,\tilde{c}}$	[0.01;0.5]
Λ_t	[50;300] запросов/сек
$\mu_{k,c}$	[0,1;10] запросов/сек
$\{R\}$	[0,6; 1,4] сек

Таблица 1 **Границы параметров**

Описание	**Диапазон**
Контейнеры	[1;9]
Классы запросов	[1;24]
Компоненты	[1;10]
Типы ВМ	[1;12]

Таблица 2 **Количества элементов**

Операционная нагрузка была сгенерирована согласно логам реальной облачной Веб-системы, использующей более 100 серверов. Логи содержат годовую статистику количества пользовательских сессий в час. Распределение нагрузки в течение дня имеет бимодальное распределение с пиками в 11.00 и 16.00. Различные распределения нагрузок были получены добавлением случайного шума в каждую тестируемую конфигурацию.

Тесты проводились на виртуальной машине VirtualBox с операционной системой Ubuntu Server 13.10 на процессоре Intel Core i7 2.2GHz с 6 Гб RAM.

Для решения оптимизационной проблемы использовался CPLEX 12.6.0.0 [8], а для получения результатов теоретико-игровой модели Gambit 13.1.2 [9].

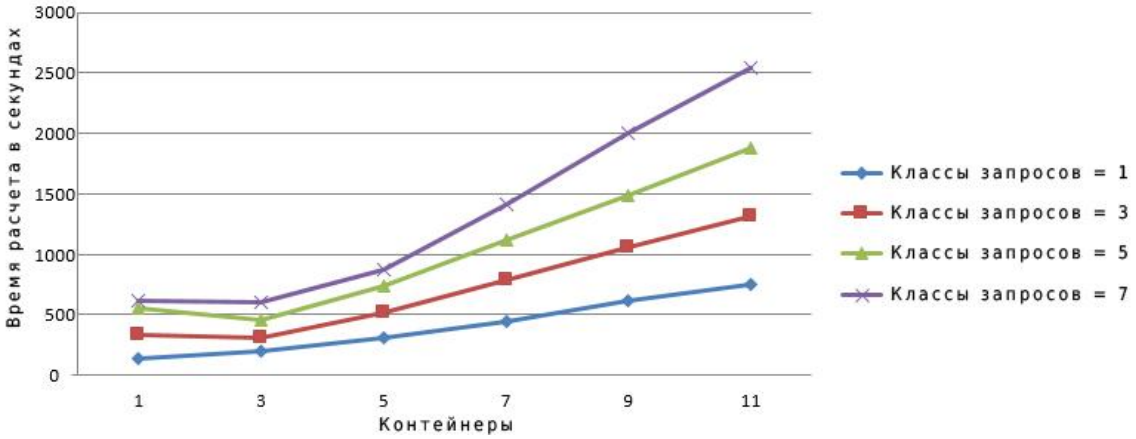

Рис 5. **Временные оценки работы алгоритма в зависимости от количества контейнеров и классов запросов в модели.**

Пример временных оценок работы алгоритма в зависимости от количества контейнеров и компонентов представлен на рисунке 5. В течение применяемых тестов время, затрачиваемое на расчет каждой из предложенных моделей, не превышало 57 минут. Учитывая, что у лица, принимающего решения, как правило, есть один день или даже несколько суток на проведение подобного анализа, можно сделать вывод, что предложенный нами алгоритм позволяет получать оптимальные в смысле сделанных допущений решения достаточно быстро.

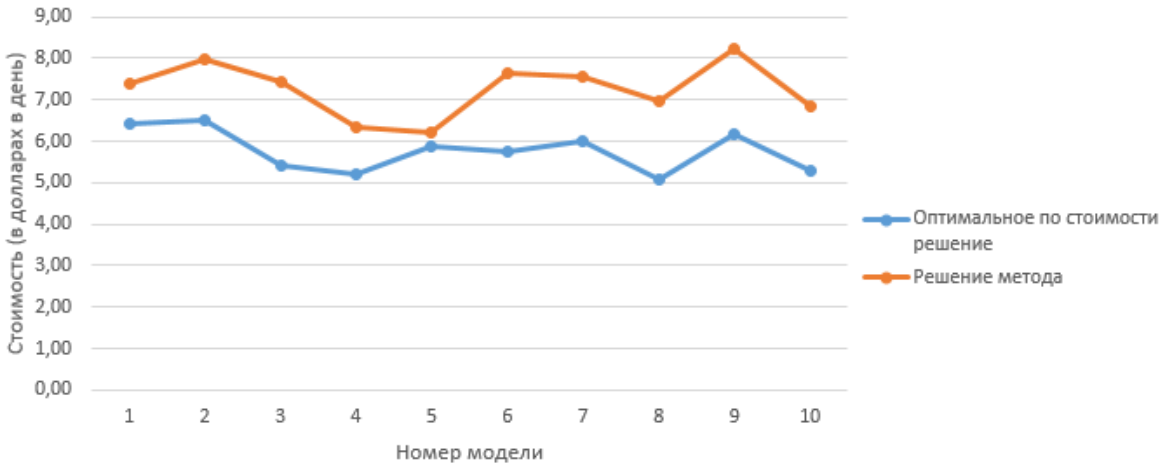

Рис. 6. **Соотношение стоимостей решений метода и глобально оптимальных решений.**

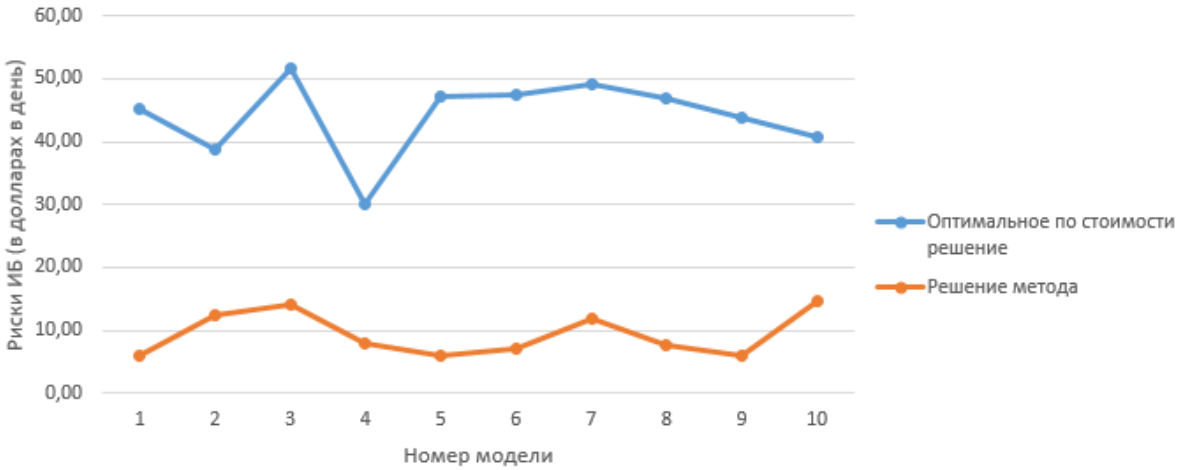

Рис. 7. **Соотношение рисков ИБ решений метода и глобально оптимальных решений.**

На Рис. 6 показаны результаты работы алгоритма для модели с 5 контейнерами, 10 компонентами и 10 классами запросов. Линии на рисунке отражают значения рисков информационной безопасности в зависимости от выбранных ограничений на время ответа. Оранжевая линия соответствует реализации нашего алгоритма, синяя отражает принятый в литературе подход, при котором выбирается самое оптимальное в плане цены решение, но риски информационной безопасности не учитываются. Рисунок 7 отражает соотношения для достижимой стоимости соответствующих решений с учетом рисков информационной безопасности. При проведении наших экспериментов стоимость выбранного с учетом рисков решения была больше стоимости оптимального по цене решения в среднем на 26%, а значение рисков информационной безопасности уменьшилось на 62%.

4. Заключение.

В статье был предложен метод, позволяющий архитектору облачных систем получать в автоматическом режиме конфигурации, оптимизированные по стоимости и рискам информационной безопасности, при этом выполняя требования по качеству обслуживания. Была проведена экспериментальная оценка метода, показывающая его эффективность в смысле достижения роста скорости получения результата и уменьшения конечных значений рисков информационной безопасности на примере задачи противодействия EDoS-атакам.

Список литературы

1. Ponemon Institute and CA. ”Security of Cloud Computing Users: A Study of Practitioners in the US & Europe.” May 12, 2010. – P. 34

2. *Alosaimi W., Al-Begain K.* An Enhanced Economical Denial of Sustainability Mitigation System for the Cloud. Seventh International Conference on Next Generation Mobile Apps, Services and Technologies. IEEE, 2013.

3. *Sqalli M. H., Al-Haidari F., Salah K.* EDoS-Shield - A Two-Steps Mitigation Technique against EDoS Attacks in Cloud Computing. Fourth IEEE International Conference on Utility and Cloud Computing, 2011.

4. *Hoffman R.* Apache OFBiz Cookbook. Packt Publishing, 2010. – P.300

5. Becker S., Koziolek H., Reussner R. The Palladio component model for model-driven performance prediction, Journal of Systems and Software, v.82 n.1, p.3-22, 2009

6. *Mittelmann, H. D.* Recent Benchmarks of Optimization Software, 22nd European Conference on Operational Research (EURO XXII Prague, Czech Republic: Dept. of Math and Stats Arizona State University), 2007.

7. *Nash Jh.* Equilibrium points in n-person games. Proceedings of the National Academy of Sciences, 36(1):48-49, 1950.

8. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/ (Дата обращения: 12.05.2014)

9. *McKelvey R. D., McLennan A. M., and Turocy T. L.* (2014). Gambit: Software Tools for Game Theory, Version 13.1.2. http://www.gambit-project.org.