36. *Bukhanovskiy A.V., Zhitnikov A.N., Petrosyan S.G.* Vysokoproizvoditel'nye tekhnologii ekstrennykh vychisleniy dlya predotvrashcheniya ugrozy navodneniy [High-performance technologies of urgent computing for preventing the threat of flooding], *Izvestiya vysshikh uchebnykh zavedeniy. Priborostroenie* [Journal of Instrument Engineering], 2011, Vol. 54, No. 10.

Статью рекомендовал к опубликованию д.ф.-м.н., профессор М.В. Якобовский.

**Сухорослов Олег Викторович** – Институт проблем передачи информации им. А.А. Харкевича Российской академии наук; e-mail: sukhoroslov@iitp.ru; 117218, Москва, Нахимовский проспект, 36, корпус 1; тел.: +74957189631; лаборатория распределенных вычислительных систем; с.н.с.; к.т.н.

**Sukhoroslov Oleg Viktorovich** – Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute); e-mail: sukhoroslov@iitp.ru; 36/1, Nakhimovskiy ave., Moscow, 117218, Russia; phone: +74957189631; Laboratory of Distributed Computing Systems; senior researcher; cand. of eng. sc.

УДК 004.052.34, 004.75 DOI 10.18522/2311-3103-2016-12-130140

**И.В. Богомолов, А.В. Алексиянц, О.Д. Борисенко, А.И. Аветисян**

# ПРОБЛЕМЫ МАСШТАБИРУЕМОСТИ ОБЛАЧНЫХ СРЕД И ПОИСК ПРИЧИН ДЕГРАДАЦИИ ЦЕНТРАЛЬНОГО СЕРВИСА ИДЕНТИФИКАЦИИ OPENSTACK KEYSTONE*

*Облачные среды приобретают все большую популярность, и они являются одним из наиболее удобных средств организации вычислений для большого спектра задач. Устройство систем разделения прав доступа и предоставления доступа для клиентских приложений, работающих с облаками напрямую, в открытых облачных средах отличается от промышленных проприетарных облачных сред. Атомарной сущностью в открытых средах является не индивидуальный пользователь, а группа пользователей. Данный подход накладывает ограничения на построение системы аутентификации и авторизации (идентификации) пользователей. Кроме того, одна и та же система отвечает как за работу с пользователями системы, так и за работу с остальными сервисами облака: это необходимо для защиты облака от добавления фальшивых узлов. Системы идентификации в открытых облачных платформах опираются на два общих компонента: реляционные СУБД для хранения данных и на алгоритмы шифрования в применении к хранимых данным. Разработчики открытых облачных платформ не ставят на первое место производительность системы идентификации и предлагают лишь подходы, обеспечивающие отказоустойчивость системы идентификации с точки зрения сохранности данных. В данной работе мы показываем важность масштабирования системы идентификации с точки зрения производительности на примере Openstack Keystone. Приводятся эксперименты, доказывающие, что за счет централизации системы хранения данных и сложности алгоритмов шифрования, число аутентификационных запросов в секунду, которое Openstack Keystone может обработать без ошибок, растет не линейно с увеличением доступных ресурсов. Как следствие, система идентификации не позволяет наращивать вычислительные мощности облачных систем на базе Openstack как относительно числа пользователей системы, так и относительно физических узлов облачной системы. В данной работе мы описываем новый подход, позволяющий избежать проблем масштабируемости за счет замены РСУБД для хранения данных на решения класса In-Memory Data Grid (IMDG). Реализован прототип на базе IMDG Tarantool.*

*Сервис идентификации и авторизации; Openstack Keystone; облачные вычисления; Denial of Service; масштабируемость облачных сред.*

* РФФИ 15-29-07111 офи_ м Исследование методов обеспечения масштабируемости систем в облачных средах и разработка высокопроизводительного отказоустойчивого центрального сервиса идентификации.

**I.V. Bogomolov, A.V. Aleksiyants, O.D. Borisenko, A.I. Avetisyan**

## SCALABILITY PROBLEMS IN CLOUD ENVIRONMENTS AND REASONS FOR PERFORMANCE DEGRADATION ON IDENTITY SERVICE OPENSTACK KEYSTONE

*Cloud services are becoming increasingly popular and they are among the most convenient means for computations in wide range of tasks. Authorization and authentication services for resource management in open-source clouds are distinct from proprietary solutions. An «atom» in open-source solutions is not a personal account but a grouping entity (tenant/project/etc). This approach imposes restrictions on building authorization and authentication service. Besides, users and other cloud services interact with the same system which prevents the cloud from adding fake nodes. Identification systems in open cloud platforms rely on two common components: relational database management system and common algorithms of cryptography on stored data. None of the system developers put performance first in such systems but they provide some means for reliability and high availability. This article shows importance of scaling identification system in context of performance for Openstack Keystone. We provide experiments and analytics suggesting that using RDBMS is wrong for such a task due to the complexity of employed crypto-algorithms and centralized storage system. We show that this common approach leads to non-linear scalability on nodes count and number of users. We describe a new approach which allows to solve scalability issues by replacing RDBMS with In-Memory Data Grid(IMDG) solutions. Prototype solution has been implemented using Tarantool IMDG.*

*Identity service; Openstack Keystone; cloud computing; Denial of Service; cloud scalability.*

**Введение.** В настоящее время облачные среды становятся все более востребованными как среди разработчиков, так и среди конечных пользователей. Современные облачные среды предоставляют свои ресурсы в моделях обслуживания IaaS [1] (инфраструктура как услуга), PaaS (платформа как услуга), SaaS (ПО как услуга) и DaaS (рабочий стол как услуга).

В рамках модели IaaS обязательно предоставляются ресурсы вычислений (виртуальные машины), возможности организации сетей (виртуальные сети и их связь с внешними сетями) и ресурсы хранения (виртуальные блочные устройства, хранилища объектных данных, загрузочные образы операционных систем). В зависимости от среды, могут предоставляться услуги: управление балансировкой трафика, управление масштабированием виртуальных машин в зависимости от условий, управление службами доменных имен, агрегация метрик использования ресурсов.

Услуги моделей PaaS, SaaS, DaaS предоставляются прозрачным для пользователя образом с использованием инфраструктурных объектов среды. К таким услугам относятся обработка больших данных по запросу, предоставление доступа к облачной СУБД, предоставление доступа к средам исполнения для развертывания приложений, предоставление доступа к графическим рабочим столам в браузере или при помощи тонкого клиента.

Облачные среды могут использоваться для множества вычислительных задач, поскольку позволяют создавать потенциально неограниченные виртуальные инфраструктуры, оптимизированные для каждой конкретной задачи.

Существует множество провайдеров облачных услуг на коммерческой основе, таких как Amazon (EC2) [2], Microsoft (Azure) [3], Google (Compute Engine) [4]. Тем не менее, коммерческие провайдеры предоставляют публичные облачные среды только в качестве сервиса и не дают полного контроля над данными.

В связи с этим является актуальной задача построения собственных облачных систем с полным контролем над данными пользователей и физической инфраструктурой. Единственным способом построить такое облако на базе доступных компонентов является использование открытых платформ, которые представлены проектами Openstack [5], Eucalyptus [6], OpenNebula [7], CloudStack [8] и другими.

Облачные платформы должны предоставлять описанную функциональность и возможность при необходимости масштабировать развернутую систему как относительно физических узлов системы, так и относительно количества пользователей системы с учетом разделения прав доступа. Архитектура облачных платформ включает в себя множество компонентов [9]; сервисы в системе изолированы и разделены по предоставляемой функциональности и должны поддерживать варианты развертывания на независимых физических узлах. Большинство сервисов взаимодействует с ограниченным числом смежных подсистем, однако во всех облачных платформа существует сервис, который используется всеми подсистемами.

В состав облачных платформ входит специальный сервис (сервис идентификации), который отвечает за аутентификацию и авторизацию пользователей и подсистем платформы. Контроль подсистем платформы обеспечивается с двумя целями: во-первых, для защиты от подмены аппаратных узлов на скомпрометированные, во-вторых, для проверки прав доступа при прямых обращениях к ресурсам системы. Чем больше сервисов предоставляет облачная среда и чем больше экземпляров подсистем работает в системе, тем большая нагрузка ложится на сервис идентификации.

Проект Openstack предоставляет наибольшее число возможностей среди открытых платформ и состоит из 8 основных подсистем и 10 дополнительных, каждая из которых взаимодействует с центральным сервисом идентификации – Openstack Keystone. В данном проекте мы исследуем проблемы проекта Openstack, поскольку он является наиболее перспективным [10] и сталкивается с проблемами, которые еще не актуальны для остальных платформ, поскольку предоставляет наибольшее число возможностей [11]. Основным объектом наших исследований является подсистема Keystone [12].

В данной работе мы демонстрируем, что Openstack имеет проблемы масштабируемости как по количеству физических узлов системы, так и по числу активных пользователей.

В рамках предыдущей работы [14] мы выявили и получили предварительные данные о некоторых проблемах данного сервиса идентификации:

1. Openstack Keystone обладает низкой производительностью с точки зрения обработки запросов к системе в секунду (RPS).
2. Производительность Openstack Keystone деградирует со временем вплоть до отказа в обслуживании вне зависимости от используемого метода развертывания сервиса.
3. Openstack Keystone является одной из причин, не позволяющих создавать облачные системы на базе Openstack с более чем 200 физическими узлами штатными средствами[1].

Эти проблемы являются важными при развертывании крупных облачных систем.

В данной работе мы проводим анализ причин выявленных проблем.

**Подходы к обнаружению проблем.** Нами был разработан и выложен в открытый доступ[2] инструментарий, предоставляющий следующие возможности:

1. Автоматическое развертывание из исходных кодов любой версии (согласно меткам git) сервиса Openstack Keystone во всех возможных комбинациях средств, обеспечивающих его работу:

[1]Существуют способы создавать облачные системы на базе Openstack с большим количеством узлов, но эти способы не универсальны и могут применяться с существенными ограничениями: либо искусственным разбиением инфраструктуры на несколько отдельных облаков [14], либо отказом от обычной модели аутентификации [15].

[2]https://github.com/ispras/keystone_full_deployment.

I. Apache2+mod_wsgi, Nginx+uwsgi в качестве HTTP/WSGI фронтенда,
II. MariaDB, PostgreSQL в качестве СУБД
III. С поддержкой использования разных устройств хранения: HDD, SSD, tmpfs (полностью в памяти)

2. Автоматическое развертывание на отдельном узле системы генерации нагрузки на сервис и анализа результатов (на базе проекта Rally [17]).
3. Генерация сценариев нагрузочного тестирования.
4. Запуск сценариев и автоматическое определение критического значения RPS для заданного временного окна тестирования, после которого время отклика сервиса начинает линейно возрастать, либо случается отказ в обслуживании.

Приведем в качестве примера один из результатов, иллюстрирующих проблему (рис. 1):

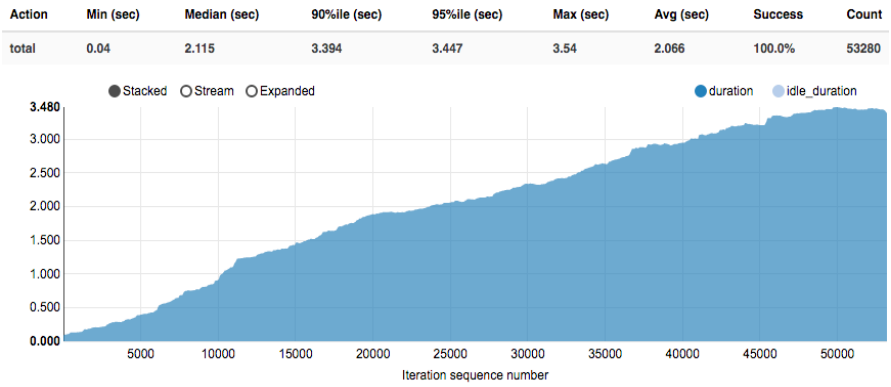


*Рис. 1. Время отклика Keystone при RPS=148, Apache2 + PostgreSQL (tmpfs)*

На оси абсцисс на этом графике расположены номера запросов в порядке их фактического исполнения, по оси ординат – фактическое время ответа на запрос в секундах. RPS=148 является значением, при котором начинает наблюдаться деградация производительности при фиксированной длительности тестирования (в данном примере – 6 минут). Отметим, что в данном примере исключается замедление из-за внешней памяти, поскольку сама база данных располагается в оперативной памяти целиком. Если в качестве внешней памяти использовать HDD, деградация наблюдается при меньшем значении RPS (рис. 2):

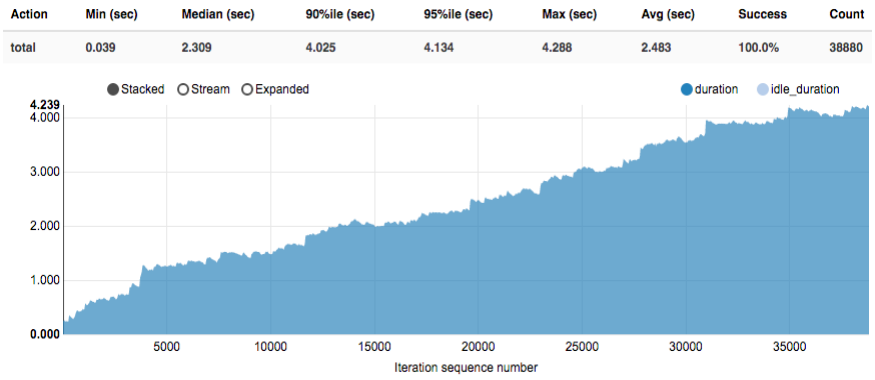


*Рис. 2. Время отклика Keystone при RPS=108, Apache2 + PostgreSQL (HDD)*

На этом графике приводится найденное критическое значение RPS на том же временном окне, с базой данных, расположенной на обычном HDD. После данного этапа исследований стало очевидно, что первостепенной задачей является устранение эффекта деградации, поскольку значительное снижение производительности при фиксации временного окна наступает при сравнительно небольшом числе запросов в секунду.

Также продемонстрируем, как выглядит тот же график в случае низкой нагрузки на сервис идентификации (рис. 3):

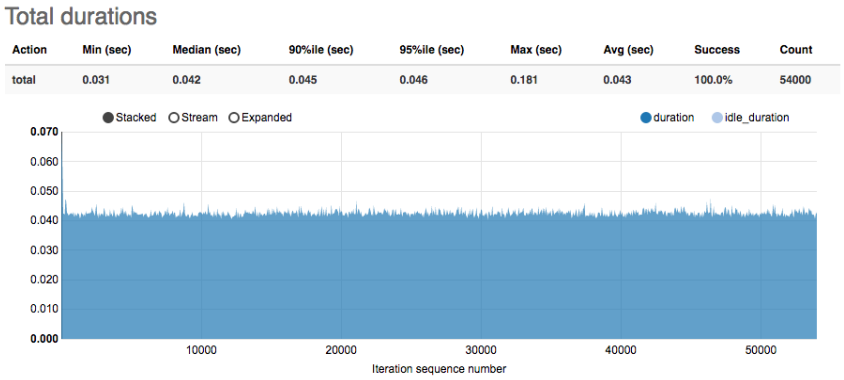


*Рис. 3. Время отклика Keystone при RPS=24, uWSGI + PostgreSQL (tmpfs)*

Для конфигурации с одним узлом мы выявили предельные значения RPS для каждой из возможных конфигураций развертывания (результаты в табл. 1).

В ходе тестирования Keystone разворачивался на одном физическом узле с процессором Intel Core i7-4790 (4 ядра, 3.6ГГц), 32Гб оперативной памяти DDR3 с частотой 1600МГц, жестким диском Western Digital WD3000F9YZ (7200RPM), RAID-0 массивом SSD накопителей Kingston SV300S3D7/120G с использованием промышленного RAID-контроллера. Система тестирования запускалась на выделенном физическом узле с теми же характеристиками, между узлами использовался прямой канал связи с пропускной способностью 1Гбит/с. Нагрузка на сетевые интерфейсы ни в один момент времени не превосходила 80Мбит/с. Для каждой новой конфигурации и каждого значения RPS все данные уничтожались и производилась автоматическая настройка Keystone с нуля. Параметры ядра Linux были настроены таким образом, чтобы не возникало ограничений на число TCP-соединений со стороны операционной системы.

Таблица 1

**Предельные значения RPS, при которых Keystone корректно функционирует в режиме одного узла**

| | | tmpfs | SSD | HDD |
|---|---|---|---|---|
| Apache2 | PostgreSQL | 148 | 138 | 105 |
| Apache2 | MariaDB | 150 | 148 | 65 |
| Nginx | PostgreSQL | 175 | 173 | 142 |
| Nginx | MariaDB | 180 | 178 | 97 |

Отдельно были проведены дополнительные измерения зависимости 95-й перцентили времени отклика и процента успешных запросов в зависимости от количества запросов в секунду в различных конфигурациях. На рис. 4 можно увидеть два графика: верхний график на оси ординат содержит 95-й перцентиль времени

отклика в секундах, а нижний – процент успешных ответов Keystone на запросы в ходе тестирования. Значения процента успешных ответов обрезаны до 90 % для большей наглядности.

Графики демонстрируют описываемое поведение: до определенного момента система функционирует корректно, затем наблюдается скачок задержки времени ответов на запросы и затем число ошибочных ответов растет вместе с временем задержки.

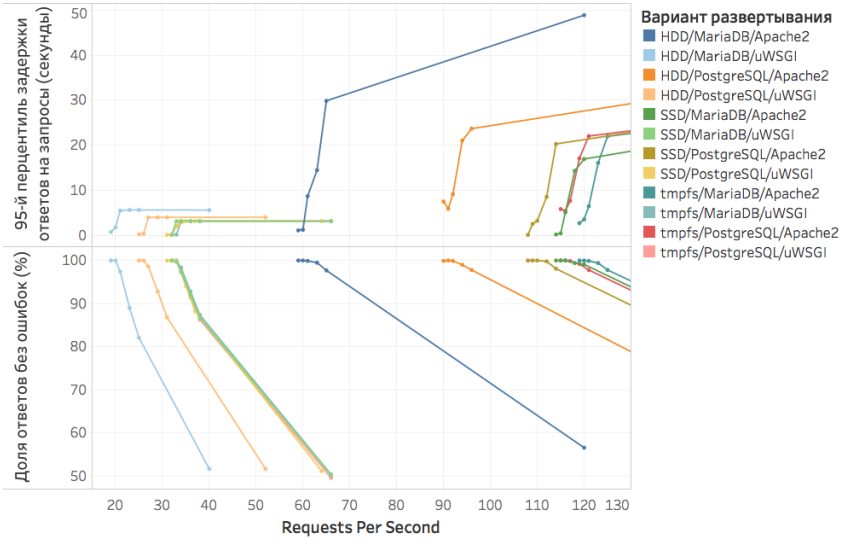


*Рис. 4. Рост времени задержки ответов на запросы и рост числа ошибок в зависимости от значения RPS для разных вариантов развертывания Keystone*

С учетом того, что сервис идентификации используется и пользователями, и самими подсистемами облачной среды, порядок показателей производительности говорит о невозможности эффективно масштабировать облако. Данные вопросы также затрагиваются в работах [17, 18].

**Подход к изоляции причин проблем производительности Keystone.** В качестве подхода к поиску причин проблем производительности мы используем инструментацию исходных кодов Openstack Keystone для выявления основных проблемных функций, вызывающих деградацию системы.

Мы построили полуавтоматическую систему, позволяющую производить профилировку всех вызовов внутри Keystone с предсказуемой степенью замедления системы. В результате влияния работы разработанного инструмента мы получили меньшее абсолютное значение критического RPS, но характер замедления системы сохраняется. В качестве результата работы инструмента мы получили полный список функций с их временем исполнения для одного и того же временного окна для значений RPS равных 10, N и [(N-10)/2], где N – критическое значение RPS. Основная идея заключается в том, чтобы найти отдельные функции, которые увеличивают время исполнения быстрее остальных. Полученные значения позволили выявить такие функции. Затем мы построили полный граф вызовов для данного сценария тестирования и провели автоматическое соответствие между узлами в графе вызовов и выходными данными профилировщика. В обоих случаях мы получили одинаковое покрытие кода в точности соответствующее данному сценарию использования.

Также была построена система очистки графа, позволяющая отследить функции, вызывающие деградацию системы, и получить компактный граф вызовов с группировкой по подсистемам, содержащим проблемные участки, причем не только внутри кода Keystone, но и его зависимостей.

На основе полученного графа вызовов был проведен анализ участков кода, вызывающих замедление системы.

Наибольшее влияние оказал участок программного кода, отвечающий за выдачу временного ключа доступа к системе. Выдача данного ключа производится следующим образом:

1. Пользовательское приложение или внутренний сервис Openstack совершает HTTP-запрос, содержащий логин и пароль пользователя или сервиса по специальному URI.
2. Сервис Keystone принимает запрос и строит хэш от конкатенации логина и пароля по алгоритму SHA512 с числом раундов 5000 (до ноября 2015 года использовалось 40000 раундов[3]).
3. Сервис обращается к базе данных для поиска записи, соответствующей
   1) Тенанту (пользовательской группе) к которому относится данный пользователь [обязан присутствовать в теле запроса].
   2) Пользовательскому хэшу пароля.

В случае нахождения такой записи, от присланных данных строится хэш SHA512 [19] с 5000 раундов и сравнивается с найденной записью в таблице. В случае совпадения, в качестве ответа пользователь получает JSON ответ, содержащий сгенерированный UUID4 хэш в качестве токена и срок действия данного токена. По умолчанию срок действия токена равен часу.

Мы провели исследование масштабируемости Openstack Keystone. В качестве вычислительных ресурсов для проверки мы использовали облачную систему ИСП РАН, основанную на Openstack Liberty.

Характеристики одного узла:

1. 8ГБ оперативной памяти (частота 1600МГц) без перерасхода ресурсов (overcommit) в облачной среде.
2. 2 виртуальных CPU (напрямую соответствующих реальным ядрам процессора без перерасхода ресурсов в облачной среде).
3. 10ГБ дискового пространства.
4. 10Гбит/с – пропускная способность сети.

Кроме того, использовались дополнительные узлы. Первый узел обеспечивал распределение нагрузки согласно документации Openstack: узел содержал только HAProxy, 32ГБ памяти, 16VCPU, 10ГБ. Также мы использовали дополнительный узел с СУБД с такими же характеристиками.

Стоит отдельно пояснить причины, по которым мы используем ровно один узел для СУБД. Существует официально поддерживаемый способ масштабирования: использование Galera [20] для обеспечения распределенного режима работы СУБД MySQL/MariaDB.Исходя из официальной документации, данный способ не предоставляет никаких возможностей к улучшению производительности и позволяет системе работать со скоростью худшего из её узлов.

Также на основе полученных результатов мы выявили, что наилучшую производительность Openstack Keystone показывает в сочетании с СУБД PostgreSQL.

В связи с этими наблюдениями мы решили проводить тестирование с единственным узлом, отвечающим за хранение данных под управлением PostgreSQL с настройками, оптимизированными под большое количество активных клиентских соединений (т.е. соединений между процессами Keystone и СУБД).

[3]https://review.openstack.org/#/c/165054/.

Также важно отметить, что Openstack Keystone запускается таким образом, что поведение двух независимых узлов неотличимо от узла с удвоенным количеством CPU и RAM, поскольку архитектурно каждый процесс на физическом узле не зависит от остальных. Таким образом (были проведены соответствующие эксперименты), 2 узла с 8Гб оперативной памяти и 2 CPU на каждом работают идентично так же, как и один узел с 16 Гб оперативной памяти и 4 CPU в случае, когда СУБД находится на выделенном узле. Максимальная пропускная способность в ходе тестирования не превышала 80 Мбит/с, так что влияние сетевой части является несущественным.

Мы провели тестирование с использованием уже описанного сценария для установления характера масштабируемости системы Openstack Keystone. Поведение системы в зависимости от числа используемых узлов Openstack Keystone проиллюстрировано на рис. 5 и 6.

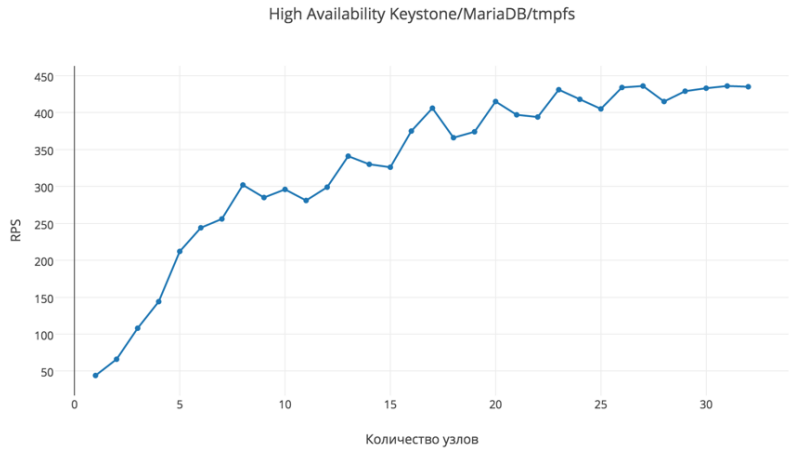


*Рис. 5. Предельное число RPS без эффекта деградации для конфигурации MariaDB/tmpfs в зависимости от числа узлов*

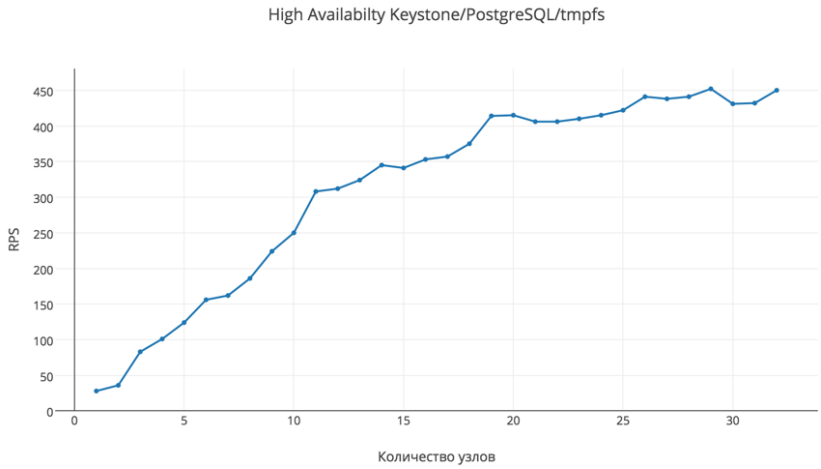


*Рис. 6. предельное число RPS без эффекта деградации для конфигурации PostgreSQL/tmpfs в зависимости от числа узлов*

Полученные в ходе тестирования результаты говорят о том, что, во-первых, системы масштабируются не линейно в зависимости от числа узлов, а во-вторых, предельные значения RPS малы и не позволяют масштабировать облачную среду.

**Выводы и предлагаемое решение проблемы.** Результаты тестирования показывают, что узким местом с точки зрения масштабируемости является РСУБД и связь процессов Keystone с ней. Задачи шифрования задействуют только процессорные мощности, поэтому добавление процессорных мощностей позволяет увеличивать производительность расчетов линейно. Однако каждый новый рабочий процесс Keystone устанавливает отдельное соединение с РСУБД, и чем больше процессов взаимодействует с РСУБД, тем меньший прирост производительности наблюдается в системе в целом.

Предлагаемый подход заключается в том, чтобы использовать решения класса In-Memory Data Grid вместо РСУБД. Данный подход обладает как рядом преимуществ, так и рядом недостатков.

Преимущества:

1. Мы выяснили, что некоторые решения данного класса масштабируются практически линейно и позволяют добиваться высокой производительности в распределенном транзакционном режиме (Tarantool [21] – 45000 RPS для 8 узлов, Apache Ignite [22] – 25000 RPS для 8 узлов).
2. Показатели производительности IMDG в нераспределенном режиме высоки по сравнению с РСУБД.

Недостатки:

1. Невозможно использовать в режиме одного узла с точки зрения сохранности данных, поскольку данные решения ориентированы на работу в оперативной памяти с отложенной выгрузкой на диск.
2. Поскольку все сервисы идентификации для открытых облачных сред зависят от РСУБД, нет возможности прозрачной замены поддерживаемых систем хранения на новую.
3. Основной причиной замедления производительности все равно является шифрование данных; даже заменив систему хранения данных, невозможно добиться значительного прироста производительности. Тем не менее, возможно обеспечить линейный рост производительности подобных систем по количеству узлов и количеству пользователей.

Был реализован прототип демонстрирующий наш подход на базе Tarantool. Исследование результатов его работы является темой для дальнейшей работы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Moreno-Vozmediano R., Montero R.S., Llorente I.M.* Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures // Computer. – 2012. – Vol. 45, No. 12. – P. 65-72.
2. Официальный сайт Amazon Elastic Compute Cloud. – https://aws.amazon.com/ec2/.
3. Официальный сайт Microsoft Azure. – https://azure.microsoft.com/en-us/.
4. Официальный сайт Google Compute Engine. – https://cloud.google.com/compute/.
5. Официальныйсайт проекта OpenStack. – https://www.openstack.org/.
6. Официальныйсайт проекта Eucalyptus. – https://www.eucalyptus.com/.
7. *Fontán J. et al.* OpenNEbula: The open source virtual machine manager for cluster computing // Open Source Grid and Cluster Software Conference, San Francisco, CA, USA. – 2008.
8. *Kumar R. et al*. Apache cloudstack: Open source infrastructure as a service cloud computing platform // Proceedings of the International Journal of advancement in Engineering technology, Management and Applied Science. – 2014. – P. 111-116.
9. *Luo J.Z. et al.* Cloud computing: architecture and key technologies // Journal of China Institute of Communications. – 2011. – Vol. 32, No. 7. – P. 3-21.
10. *Freet D. et al.* Open source cloud management platforms and hypervisor technologies: A review and comparison // SoutheastCon, 2016. – IEEE, 2016. – P. 1-8.

11. *Lynn T. et al.* A Comparative Study of Current Open-source Infrastructure as a Service Frameworks // CLOSER. – 2015. – P. 95-104.
12. Описание архитектуры Openstack Keystone. – http://docs.openstack.org/developer/ keystone/architecture.html.
13. *Богомолов И.В., Алексиянц А.В., Шер А.В., Борисенко О.Д., Аветисян А.И.* Метод тестирования производительности и стресс-тестирования центральных сервисов идентификации облачных систем на примере Openstack Keystone: Труды Института системного программирования РАН. – 2015. – Т. 27. – Вып. 5. – С. 49-58.
14. *Bell, Tim, et al.* Scaling the CERN OpenStack cloud // Journal of Physics: Conference Series. – Vol. 664, No. 2. IOP Publishing, 2015.
15. Marek Denis, Jose Castro Leon, Emmanuel Ormancey, Paolo Tedesco (CERN, Geneva, Switzerland). Identity federation in OpenStack - an introduction to hybrid clouds. 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015). Doi:10.1088/1742-6596/664/2/022015.
16. Официальная страница проекта Rally. – https://wiki.openstack.org/wiki/Rally.
17. *Almási G. et al.* Toward building highly available and scalable OpenStack clouds // IBM Journal of Research and Development. – 2016. – Vol. 60, No. 2-3. – P. 1-5.
18. *Baojiang Cui, Tao Xi.* Security Analysis of Openstack Keystone // in Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS): 9th International Conference 8-10 July 2015. – P. 283-288. Doi: 10.1109/IMIS.2015.44.
19. *Bharati P.V., Mahalakshmi T.S.* A Combinational Approach for securing the data in cloud storage using HMAC-SHA512 and Information Secured Algorithm (ISA) // International Journal of Applied Engineering Research. – 2016. – No. 6. – P. 4081-4084.
20. *Kemme B., Alonso G.* Database replication: a tale of research across communities // Proceedings of the VLDB Endowment. – 2010. – No. 1-2. – P. 5-12.
21. *Abramova V., Bernardino J., Furtado P.* Experimental evaluation of NoSQL databases // International Journal of Database Management Systems. – 2014. – No. 3. – P. 1.
22. *Dinsmore T.W.* In-Memory Analytics // Disruptive Analytics. – Apress, 2016. – P. 97-116.

REFERENCES

1. *Moreno-Vozmediano R., Montero R.S., Llorente I.M.* Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures, *Computer*, 2012, Vol. 45, No. 12, pp. 65-72.
2. Ofitsial'nyy sayt Amazon Elastic Compute Cloud [Amazon Elastic Compute Cloud official page]. Available at: https://aws.amazon.com/ec2/.
3. Ofitsial'nyy sayt Amazon Elastic Compute Cloud [Microsoft Azure official page]. Available at: https://azure.microsoft.com/en-us/.
4. Ofitsial'nyy sayt Amazon Elastic Compute Cloud [Google Compute Engine official page]. Available at: https://cloud.google.com/compute/.
5. Ofitsial'nyy sayt Amazon Elastic Compute Cloud [OpenStack project official page]. Available at: https://www.openstack.org/.
6. Ofitsial'nyy sayt Amazon Elastic Compute Cloud [Eucalyptus project official page]. Available at: https://www.eucalyptus.com/.
7. *Fontán J. et al.* OpenNEbula: The open source virtual machine manager for cluster computing, *Open Source Grid and Cluster Software Conference, San Francisco, CA, USA*, 2008.
8. *Kumar R. et al*. Apache cloudstack: Open source infrastructure as a service cloud computing platform, *Proceedings of the International Journal of advancement in Engineering technology, Management and Applied Science*, 2014, pp. 111-116.
9. *Luo J.Z. et al.* Cloud computing: architecture and key technologies, *Journal of China Institute of Communications*, 2011, Vol. 32, No. 7, pp. 3-21.
10. *Freet D. et al.* Open source cloud management platforms and hypervisor technologies: A review and comparison, *SoutheastCon, 2016*. IEEE, 2016, pp. 1-8.
11. *Lynn T. et al.* A Comparative Study of Current Open-source Infrastructure as a Service Frameworks, *CLOSER*, 2015, pp. 95-104.

12. Opisanie arkhitektury Openstack Keystone [Openstack Keystone architecture description]. Available at: http://docs.openstack.org/developer/keystone/architecture.html.
13. *Bogomolov I.V., Aleksiyants A.V., Sher A.V., Borisenko O.D., Avetisyan A.I.* Metod testirovaniya proizvoditel'nosti i stress-testirovaniya tsentral'nykh servisov identifikatsii oblachnykh sistem na primere Openstack Keystone [A performance testing and stress testing of cloud platform central identity: OpenStack Keystone case study]: *Trudy Instituta sistemnogo programmirovaniya RAN* [Proceedings of the Institute for System Programming], 2015, Vol. 27, Issue 5, pp. 49-58.
14. *Bell, Tim, et al.* Scaling the CERN OpenStack cloud, *Journal of Physics: Conference Series*, Vol. 664, No. 2. IOP Publishing, 2015.
15. Marek Denis, Jose Castro Leon, Emmanuel Ormancey, Paolo Tedesco (CERN, Geneva, Switzerland). Identity federation in OpenStack - an introduction to hybrid clouds. 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015). Doi:10.1088/1742-6596/664/2/022015.
16. Ofitsial'naya stranitsa proekta Rally [Rally project official page]. Available at: https://wiki.openstack.org/wiki/Rally.
17. *Almási G. et al.* Toward building highly available and scalable OpenStack clouds, *IBM Journal of Research and Development*, 2016, Vol. 60, No. 2-3, pp. 1-5.
18. *Baojiang Cui, Tao Xi.* Security Analysis of Openstack Keystone, *in Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS): 9th International Conference 2015, 8-10 July 2015*, pp. 283-288, Doi: 10.1109/IMIS.2015.44.
19. *Bharati P.V., Mahalakshmi T.S.* A Combinational Approach for securing the data in cloud storage using HMAC-SHA512 and Information Secured Algorithm (ISA), *International Journal of Applied Engineering Research*, 2016, No. 6, pp. 4081-4084.
20. *Kemme B., Alonso G.* Database replication: a tale of research across communities, *Proceedings of the VLDB Endowment*, 2010, No. 1-2, pp. 5-12.
21. *Abramova V., Bernardino J., Furtado P.* Experimental evaluation of NoSQL databases, *International Journal of Database Management Systems*, 2014, No. 3, P. 1.
22. *Dinsmore T.W.* In-Memory Analytics, *Disruptive Analytics*. Apress, 2016, pp. 97-116.

Статью рекомендовал к опубликованию д.т.н. В.В. Кореньков.

**Богомолов Игорь Владимирович** – Институт системного программирования Российской академии наук; e-mail: bogomolov@ispras.ru; Москва, ул. Александра Солженицына, 25; тел.: +79168251426; стажер-исследователь.

**Алексиянц Александр Владимирович** – e-mail: aleksiyantsa@ispras.ru; тел.: +79268505598; стажер-исследователь.

**Борисенко Олег Дмитриевич** – e-mail: borisenko@ispras.ru; тел.: +79037082057; м.н.с.

**Аветисян Арутюн Ишханович** – e-mail: arut@ispras.ru; тел.: +79859979718; д.ф.-м.н.; директор ИСП РАН.

**Bogomolov Igor' Vladimirovich** – Institute for System Programming of the Russian Academy of Sciences; e-mail: bogomolov@ispras.ru; 25, Alexander Solzhenitsyn street, Moscow, Russia; phone: +79168251426; research Intern.

**Aleksiyants Alexandr Vladimirovich** – e-mail: aleksiyantsa @ispras.ru; phone: +79268505598; research Intern.

**Borisenko Oleg Dmitrievich** – e-mail: borisenko@ispras.ru; phone: +79037082057; junior researcher.

**Avetisyan Arutyun Ishhanovich** – e-mail: arut@ispras.ru; phone: +79859979718; dr. of phys.-math. sc.