Реализация сервиса для замены Keystone в качестве центрального сервиса идентификации облачной платформы Openstack[1]

Е.Л. Аксенова <lenaaxenova@ispras.ru>
В.В. Швецова <shvetcova@ispras.ru>
О.Д. Борисенко <al@somestuff.ru>
И.В. Богомолов <bogomolov@ispras.ru>
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В данной работе рассматриваются проблемы масштабируемости проекта Keystone — центрального сервиса авторизации и аутентификации облачной платформы Openstack и новый принцип построения сервиса, позволяющий избегать этих проблем. В более ранних работах был предложен подход к масштабированию Openstack Keystone путем отказа от использования СУБД MySQL/MariaDB и PostgreSQL в качестве хранилища данных сервиса в пользу распределенных NoSQL решений. Данная работа представляет полноценную реализацию сервиса, обеспечивающего полную функциональность Openstack Keystone API V3, на базе API Gateway и использования Apache Cassandra.

Ключевые слова: Openstack Keystone; Apache Cassandra; Kong; API Gateway; Lua; облачные среды.

DOI: 10.15514/ISPRAS-2017-29(6)-11

Для цитирования: Аксенова Е.Л., Швецова В.В, Борисенко О.Д., Богомолов И.В. Реализация сервиса для замены Keystone в качестве центрального сервиса идентификации облачной платформы Openstack. Труды ИСП РАН, том 29, вып. 6, 2017 г., стр. 203-212. DOI: 10.15514/ISPRAS-2017-29(6)-11

1. Введение

В современном мире существует необходимость надежного хранения и оперативной обработки больших объемов данных. Одним из наиболее перспективных подходов в данной области является построение систем

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

хранения и обработки данных над облачными средами. Ресурсы облачной среды могут предоставляться клиенту в моделях обслуживания IaaS (инфраструктура как услуга), PaaS (платформа как услуга), SaaS (программное обеспечение как услуга), DaaS (рабочее окружение как услуга) [1]. В данной работе рассматривается облачная платформа Openstack и ее сервисы, обеспечивающие IaaS модель обслуживания.

Модель IaaS характеризуется предоставлением вычислительных виртуальных ресурсов, возможностью организации сетей, а также предоставлением ресурсов хранения (загрузочные образы операционных систем, хранилища объектных данных и виртуальные блочные устройства). Данная модель подразумевает полноценное использование клиентом вычислительных ресурсов выделенной виртуальной машины и является самой низкоуровневой моделью обслуживания в облачных средах.

Облачные услуги предоставляются на коммерческой основе такими провайдерами как Amazon EC2 [2], Microsoft Azure [3], Google Compute Engine [4] и рядом других. Однако существует множество ситуаций, где использование публичных облачных сервисов недопустимо из-за повышенных требований к контролю над пользовательскими данными. В таких условиях как правило используются приватные облачные среды, построенные на платформах с открытыми исходными кодами. Среди открытых платформ, предоставляющих выделяются следующие проекты: OpenStack [5], Eucalyptus [6], OpenNebula [7].

Кроме описанной выше функциональности облачные платформы должны поддерживать возможность масштабирования как относительно физических узлов системы, так и относительно количества пользователей системы с учетом разделения прав доступа. Архитектура облачных платформ представляет из себя большое число изолированных сервисов, каждый из которых реализует определенную функциональность облачной среды [8]. Все сервисы системы обязаны поддерживать различные варианты развертывания на множество физических узлов.

Обязательным компонентом облачных платформ является специальный модуль, отвечающий за аутентификацию и авторизацию пользователей и сервисов облачной среды (далее — сервис идентификации). С целью защиты от подмены аппаратных узлов и проверки прав доступа на запрашиваемые ресурсы все подсистемы проекта взаимодействуют с сервисом идентификации. В связи с этим данный сервис подвергается существенной нагрузке, которая растет в зависимости от числа работающих подсистем и пользователей проекта.

В данной работе исследуются проблемы открытого проекта OpenStack [9], предоставляющего наибольшую функциональность среди открытых облачных платформ [10]. Основным объектом исследований является сервис идентификации Keystone [11], недостатки которого оказывают существенное влияние на работу всей системы. В частности, сервис Keystone является одной

из основных причин невозможности масштабируемости системы как по количеству физических узлов, так и по числу активных пользователей [12].

В работе [13] подробно описана методика тестирования, в ходе которого были выявлены основные проблемы сервиса идентификации Keystone:

1. Низкая производительность сервиса с точки зрения обработки числа запросов к системе в секунду (RPS).
2. Деградация производительности со временем вплоть до отказа в обслуживании вне зависимости от используемых модулей, обеспечивающих его работу.
3. Невозможность построения монолитной облачной среды на базе открытых облачных платформ с более чем 200 физическими узлами в системе без использования специфических решений, которые не могут являться универсальными. К таким решениям относятся отказ от монолитности облачной среды с использованием федераций и отказ от встроенного сервиса идентификации Keystone.
4. Нелинейный характер масштабирования системы в зависимости от числа используемых ресурсов.

Также был проведен анализ причин подобного поведения. Одной из основных причин наблюдаемых проблем является построение сервиса над классическими РСУБД с ограничениями по количеству одновременных клиентских подключений. Одновременно с этим сама архитектура сервиса не позволяет производить группировку соединений с СУБД.

В других облачных платформах сервис идентификации построен по сходным принципам.

В крупных облачных системах данные проблемы являются существенными, поэтому цель данной работы заключается в устранении вышеперечисленных недостатков за счет перехода к новой модели устройства облачного сервиса идентификации на базе распределенных NoSQL СУБД.

2. Устройство предлагаемого решения

Данный раздел посвящен принципам построения масштабируемого сервиса для замены Openstack Keystone. В качестве основного принципа используется раздельное масштабирование узлов, отвечающих за построение токенов, и узлов, отвечающих за хранение пользовательских данных.

В рамках проверки идеи об использовании распределенных хранилищ для масштабирования сервиса, предоставляющего функциональность Openstack Keystone, был реализован прототип, реализующий функциональность для самого требовательного сценария нагрузки. С точки зрения алгоритма работы необходимых для сценария вызовов, прототип не отличался от Openstack Keystone. Однако для реализации был использован язык Lua и использован

Tarantool [14][15] в качестве распределенного хранилища. Результат тестирования приведен на рис. 1.

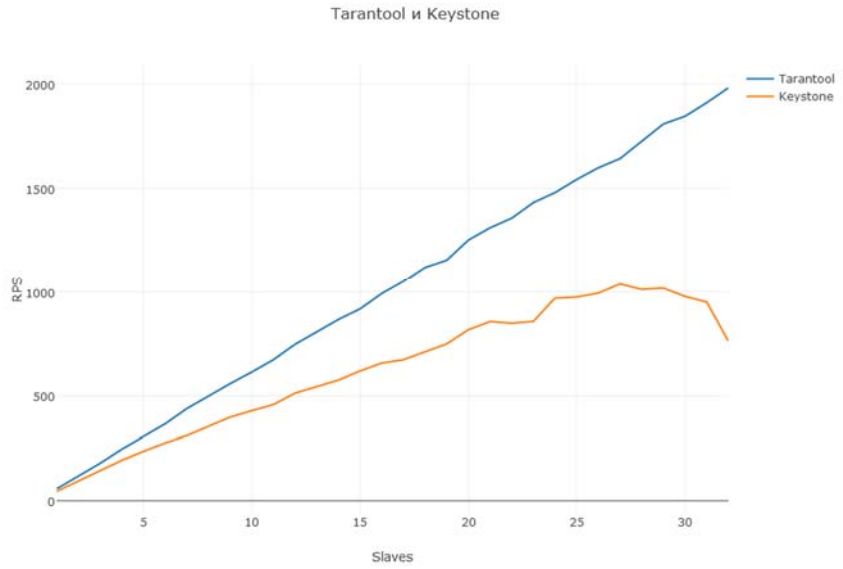

Рис. 1. Зависимость критического значения RPS от числа узлов в кластере для Keystone (PostgreSQL + tmpfs) и прототипа на базе Tarantool
Fig. 1. Critical RPS value to cluster nodes for Keystone (PostgreSQL on tmpfs) and Tarantool-based prototype

В отличие от результатов сервиса Keystone, реализованное решение демонстрирует линейный характер масштабируемости и более высокое значение RPS, что особенно заметно при большом числе узлов кластера. Таким образом, обеспечивается возможность масштабирования облачной платформы по физическим узлам и по числу пользователей, в отличии от сервиса OpenStack Keystone.

Положительные результаты, полученные при отказе от использования централизованной РСУБД в пользу распределенных решений, демонстрируют необходимость полноценной реализации сервиса идентификации, опирающегося на распределенную СУБД. Для решения данной задачи нами была выбрана система API Gateway Kong.

Как было отмечено выше, проект Openstack представлен в виде набора изолированных сервисов и развивается согласно принципам микросервисной архитектуры [16]: каждая из подсистем проекта работает в отдельном процессе и проектируется вне зависимости от других систем, взаимодействие между ними осуществляется с помощью HTTP-протокола. Ввиду разнородности

данных систем пользователю нередко приходится сталкиваться с тем, что интерфейс различных модулей отличается друг от друга, работа с такими системами может вызывать ряд затруднений. Популярным вариантом решения подобных проблем является использование технологии API Gateway, которая представляет из себя дополнительный программный и логический уровень между клиентами и внутренними интерфейсами сервисов [17].

Основными характеристиками таких систем являются простота масштабирования и добавления новых систем в проект, возможность проверки идентификационных данных непосредственно в слое API Gateway, удобная поддержка версий подсистем проекта, а также возможность кэширования запросов и ответов системы и регулирование трафика. Благодаря указанным свойствам, логика работы сервиса идентификации Keystone прозрачно переносится на слой API Gateway.

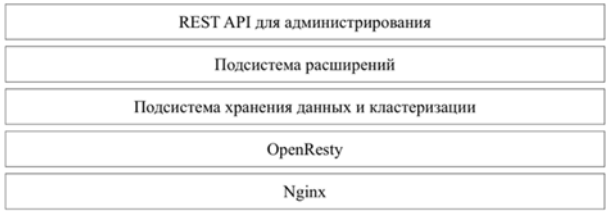

Рис. 2. Слои в API Gateway Kong
Fig. 2. Kong API Gateway layers

Среди открытых платформ, реализующих технологию API Gateway, система Kong [18] предоставляет наибольшую функциональность, а также поддерживает взаимодействие с как с реляционными, так и с нереляционными СУБД. Kong состоит из нескольких слоев (см. рис. 2).

- Верхний слой работает по принципам RESTful API (и перенаправляет полученные запросы и ответы согласно логике, реализованной в следующем слое – плагинах);
- Слой расширений представляет собой программы, написанные на языке Lua, они и обеспечивают работу API Gateway. Kong предоставляет набор готовых расширений, а также поддерживает возможность написания новых;
- Следующий слой отвечает за кластеризацию и хранение данных. Kong поддерживает работу с СУБД PostgreSQL и Apache Cassandra [19], а также допускает использование Redis [20] для кэширования данных. СУБД используются для хранения всех служебных данных проекта Kong и, кроме того, могут быть использованы для хранения пользовательских данных согласно логике, определенной каждым конкретным расширением;

- Следующий слой Openresty является расширением веб-сервера Nginx [21] с помощью функций, написанных на языке Lua. Данный слой позволяет контролировать и обрабатывать пользовательские запросы и ответы на различных стадиях их жизненного цикла (lifecycle);
- Нижний слой представлен веб-сервером Nginx и используется для обработки HTTP/HTTPS и проксирования запросов.

В качестве системы хранения данных для разработанного сервиса выбрана Apache Cassandra.

Авторами работы был разработан и реализован сервис идентификации для облачной среды Openstack в виде плагина на языке Lua, интегрируемого в систему Kong. Для хранения данных используется распределенная СУБД Apache Cassandra поскольку ее модель масштабирования широко известна своей эффективностью [22].

Данные в Cassandra хранятся в соответствии с моделью данных «семейство столбцов». Под семейством столбцов подразумевается структура, содержащая неограниченное число строк, доступ к которым осуществляется по уникальному ключу – имени строки. Основные преимущества Apache Cassandra заключаются в высокой масштабируемости и отказоустойчивости системы, очень высокой пропускной способности операции записи и хорошей пропускной способности для операций считывания, а также в возможности репликации с настраиваемым уровнем согласованности. Но при этом, являясь NoSQL системой, Cassandra не поддерживает транзакции ACID, операции соединения и возможность определения внешних ключей.

Схема данных в авторской реализации практически совпадает с оригинальной схемой данных в проекте Keystone — с поправкой на специфику Cassandra: в схеме отсутствуют внешние ключи.

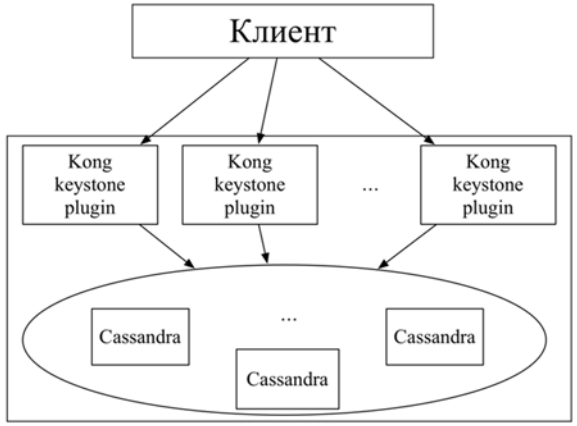

Рис. 3. Схема масштабирования разработанной системы
Fig. 3. Scaling scheme for the system

На рис. 3 схематично демонстрируется возможность масштабирования системы идентификации, реализованной с помощью Kong, на некоторое число узлов кластера. Каждый сервис Kong при этом осуществляет взаимодействие с распределенной СУБД Cassandra.

В следующих работах будет проведено тестирование производительности и сравнение с Openstack Keystone в контексте возможностей масштабирования.

Список литературы

[1]. Moreno-Vozmediano R., Montero R.S., Llorente I.M. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. Computer, vol. 45, no. 12, 2012, pp. 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]. Официальныйсайт проекта OpenNebula. – https://opennebula.org.
[8]. Luo J.Z. et al. Cloud computing: architecture and key technologies. Journal of China Institute of Communications, vol. 32, no. 7, 2011. pp. 3-21.
[9]. Freet D. et al. Open source cloud management platforms and hypervisor technologies: A review and comparison. SoutheastCon, 2016. IEEE, 2016, pp. 1-8.
[10]. Lynn T. et al. A Comparative Study of Current Open-source Infrastructure as a Service Frameworks. CLOSER, 2015, pp. 95-104.
[11]. Описание архитектуры Openstack Keystone. http://docs.openstack.org/developer/keystone/architecture.html.
[12]. Богомолов И.В., Алексиянц А.В., Борисенко О.Д., Аветисян А.И. Проблемы масштабируемости облачных сред и поиск причин деградации центрального сервиса идентификации OpenStack Keystone. Известия ЮФУ. Технические науки, №12 (185), 2016 г., стр. 130-140. DOI: 10.18522/2311-3103-2016-12-130140
[13]. И.В. Богомолов, А.В. Алексиянц, А.В. Шер, О.Д. Борисенко, А.И. Аветисян. Метод тестирования производительности и стресс-тестирования центральных сервисов идентификации облачных систем на примере Openstack Keystone. Труды ИСП, том 27, вып. 5, 2015 г., стр. 49–58. DOI: 10.15514/ISPRAS-2015-27(5)-4
[14]. Официальный сайт проекта Tarantool. - https://tarantool.org/.
[15]. Abramova V., Bernardino J., Furtado P. Experimental evaluation of NoSQL databases, International Journal of Database Management Systems, No. 3, 2014, pp. 1-16.
[16]. Dmitry Namiot, Manfred Sneps-Sneppe. On Micro-services Architecture. International Journal of Open Information Technologies, vol. 2, no. 9, 2014, pp. 24-27.
[17]. Anton Fagerberg. Optimising clients with API Gateways. Department of Computer Science Faculty of Engineering LTH, 2015.
[18]. Официальный сайт проекта Kong. - https://getkong.org.
[19]. Официальный сайт проекта Cassandra. - http://cassandra.apache.org.
[20]. Официальный сайт проекта Redis. - https://redis.io.
[21]. Официальный сайт проекта Nginx. - http://nginx.org.
[22]. Lakshman A., Prashant M. Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, vol. 4, issue 2, 2010, pp. 35-40.

Openstack Keystone identification service drop-in replacement

E.L. Axenova <lenaaxenova@ispras.ru>
V.V. Shvetsova <shvetcova@ispras.ru>
O.D. Borisenko <al@somestuff.ru>
I.V. Bogomolov <bogomolov@ispras.ru>
Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. The paper is dedicated to architecture and scalability principles for developed service intended to be a drop-in replace for Openstack Keystone. Openstack Keystone is the central identification and service catalogue service for clouds based on Openstack. Previous papers indicated problems of this service: it uses RDBMS (MariaDB/MySQL/PostgreSQL) as a data storage. Since each service and each user gets a token to have access to Openstack cloud and tokens are periodically revoked by the system, token generation is a critical function for the whole cloud. As soon as Keystone queries DBMS for getting user or service identification hashes and recomputes this hash upon the user-provided data, there is a bottleneck based on Keystone architecture. Each Keystone process has separate session with DBMS and since the recommended way is to use Galera cluster thus the DBMS part is limited to the slowest DBMS node since Galera provides High-Availability not the performance scale. Our approach is based on API Gateway Kong and its scalability through Apache Cassandra usage as a data store. Drop-in replacement is implemented as Lua plugin inside Kong API Gateway and implements Keystone V3 API.

Keywords: Openstack Keystone; Apache Cassandra; Kong; API Gateway; Lua; cloud platform.

DOI: 10.15514/ISPRAS-2017-29(6)-11

For citation: Axenova E.L., Shvetsova V.V., Borisenko O.D., Bogomolov I.V. Openstack Keystone identification service drop-in replacement. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 6, 2017, pp. 203-212 (in Russian). DOI: 10.15514/ISPRAS-2017-29(6)-11

References

[1]. Moreno-Vozmediano R., Montero R.S., Llorente I.M. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. Computer, vol. 45, no. 12, 2012, pp. 65-72.
[2]. Amazon Elastic Compute Cloud official page. Available at: https://aws.amazon.com/ec2/.
[3]. Microsoft Azure official page. Available at: https://azure.microsoft.com/en-us/.
[4]. Google Compute Engine official page. Available at: https://cloud.google.com/compute/.
[5]. OpenStack project official page. Available at: https://www.openstack.org/.
[6]. Eucalyptus project official page. Available at: https://www.eucalyptus.com/.
[7]. OpenNebula project official page. Available at: https://opennebula.org.
[8]. Luo J.Z. et al. Cloud computing: architecture and key technologies. Journal of China Institute of Communications, vol. 32, no. 7, 2011. pp. 3-21.

[9]. Freet D. et al. Open source cloud management platforms and hypervisor technologies: A review and comparison. SoutheastCon, 2016. IEEE, 2016, pp. 1-8.
[10]. Lynn T. et al. A Comparative Study of Current Open-source Infrastructure as a Service Frameworks. CLOSER, 2015, pp. 95-104.
[11]. Openstack Keystone architecture description. Available at: http://docs.openstack.org/developer/ keystone/architecture.html.
[12]. Bogomolov I.V.., Aleksiyants ., Borisenko O.D., Avetisyan A.I. Scalability problems in cloud environments and reasons for performance degradation on identity service Openstack Keystone. Izvestiya SFedU. Engineering Sciences, №12 (185), 2016, pp. 130-140 (in Russian). DOI: 10.18522/2311-3103-2016-12-130140
[13]. Bogomolov I.V., Aleksiyants A.V., Sher A.V., Borisenko O.D., Avetisyan A.I. A performance testing and stress testing of cloud platform central identity: OpenStack Keystone case study. Trudy ISP RAN / Proc. ISP RAS, vol. 27, issue 5, 2015, pp. 49–58 (in Russian). DOI: 10.15514/ISPRAS-2015-27(5)-4
[14]. Tarantool project official page. Available at: https://tarantool.org/.
[15]. Abramova V., Bernardino J., Furtado P. Experimental evaluation of NoSQL databases, International Journal of Database Management Systems, No. 3, 2014, pp. 1-16.
[16]. Dmitry Namiot, Manfred Sneps-Sneppe. On Micro-services Architecture. International Journal of Open Information Technologies, vol. 2, no. 9, 2014, pp. 24-27.
[17]. Anton Fagerberg. Optimising clients with API Gateways. Department of Computer Science Faculty of Engineering LTH, 2015.
[18]. Kong project official page. Available at: https://getkong.org.
[19]. Cassandra project official page. Available at: http://cassandra.apache.org.
[20]. Redis project official page. Available at: https://redis.io.
[21]. Nginx project official page. Available at: http://nginx.org.
[22]. Lakshman A., Prashant M. Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, vol. 4, issue 2, 2010, pp. 35-40.