РАЗРАБОТКА АЛГОРИТМОВ ДЛЯ ВЫБОРКИ ДАННЫХ В КОМПОНЕНТАХ ОБЛАЧНОЙ ИНФРАСТРУКТУРЫ

DOI: 10.36724/2072-8735-2023-17-10-20-27

Manuscript received 30 July 2023;
Accepted 18 September 2023

Петухов Денис Алексеевич,
Московский Технический Университет Связи и Информатики, г. Москва, Россия,
d.a.petukhov@edu.mtuci.ru

Ключевые слова: *облачная инфраструктура, OpenStack, форматы данных, алгоритмы, SQL*

В облачных инфокоммуникационных системах, построенных на базе программного обеспечения с открытым исходным кодом, присутствует вариативность форматов данных и их структур. Такие представления данных могут быть удобны и обоснованы на этапе разработки облачных компонентов, но работа с ними в процессе эксплуатации системы сопровождается проблемой несогласованности друг с другом и отсутствием специализированных инструментов, которые позволяли бы осуществлять выборку с использованием нескольких форматов данных. В статье рассматриваются такие форматы данных, как YAML, JSON и XML и выдвигается гипотеза о существовании возможности их преобразования в единый формат. По результатам анализа реальных выборок данных от облачных компонентов OpenStack предлагается три варианта типовых структур облачных форматов. Другая гипотеза заключается в предположении о наличии возможности объединения данных из нескольких источников, представляющих разнородные форматы. Для проверки предположений разработаны алгоритмы преобразования исходных данных в табличный формат. Быстрота преобразований проверяется на большой выборке данных и является удовлетворительной для использования в процессе эксплуатации облачных инфокоммуникационных систем. После применения предложенных алгоритмов к исходным выборкам данных облачных компонентов предложен SQL запрос и способ выборки данных из нескольких источников. Полученные результаты могут быть востребованы в решениях обеспечения работоспособности облачной инфраструктуры в условиях работы с разнородными фор-матами данных.

Для цитирования:
Петухов Д.А. Разработка алгоритмов для выборки данных в компонентах облачной инфраструктуры // T-Comm: Телекоммуникации и транспорт. 2023. Том 17. №10. С. 20-27.

For citation:
Petukhov D.A. (2023) Cloud infrastructure components data query algorithms. *T-Comm*, vol. 17, no.10, pp. 20-27. (*in Russian*)

1 Введение

Цифровая трансформация компаний ведёт к переносу многих технологических и бизнес-процессов в облако [1,2], что приводит к новым рискам конфиденциальной информации, представленной в разных форматах данных в облачной инфраструктуре [3-7].

Это во многом обусловлено следующими факторами – сложностью и разнородностью облачной инфраструктуры.

Облачная инфраструктура на базе платформы OpenStack в промышленной эксплуатации представляет собой набор компонентов и гетерогенных систем, которые, в целом, обладают слабой связью друг с другом.

В основном, каждый компонент в такой системе решает свой набор функций и имеет разный интерфейс взаимодействия с человеком, разные инструменты для обслуживания и отличающийся алгоритм взаимодействия с ним. Этот эффект может быть не заметен на этапе разработки каждого из компонентов, но и не всегда принимается во внимание при построении архитектур с использованием этих компонентов, поскольку не всегда напрямую влияет на функционирование платформы, однако вызывает неудобства при её обслуживании. Совокупность этих факторов не позволяет использовать какое-либо единое решение для оценки качества работы каждого из компонентов или всей системы в целом. Кроме того, внесение изменений в систему или добавление новых функций в отдельных её частях, сопровождающееся непрерывным изменением архитектуры или алгоритмами обслуживания, приводит к усложнению поддержки такой системы в целом. В связи с этим появляется актуальная проблема поиска алгоритмов и подходов, благодаря которым обслуживание и управление облачной инфраструктурой опиралось бы на заранее определённый план. Весь комплекс предлагаемых мер в целом призван обеспечить повышение быстроты и уменьшение сложности разработки подобных алгоритмов.

2 Обзор структур представления данных, встречаемых в программном обеспечении облачной инфраструктуры

При большом разнообразии применяемого программного обеспечения в облачной инфраструктуре встречается и одновременно применяется большое количество форматов данных. Данное обстоятельство одновременно отличает гибкость системы, и, в то же время, предъявляет высокие требования как к знаниям специалиста, работающего с системой, так и к разрабатываемым приложениям автоматизации.

К сожалению, даже централизованный сбор метрик в предназначенные системы, такие как Prometheus, Logstash дает недостаточные возможности в применении для проблемы анализа этих данных. Техники статистического анализа не применимы к обозначенной проблеме, поскольку большая часть обрабатываемых данных не является численной. В связи с этим, подобный контекст ближе связан с науками о данных в общем понятии.

В ходе исследования был проведён анализ существующей инфраструктуры и компонентов на базе OpenStack на предмет встречаемых и применяемых форматов данных. В листингах 1а,1б,1в,1г приведены оригинальные примеры данных, хранимых и передаваемых компонентами оркестрации, управления, контейнеризации и виртуализации.

По результатам анализа этих и других источников данных в таблице 1 были сведены различные форматы таких данных и описание их структуры.

3 Разнородность форматов и сложность их обработки на примере ПО Ansible

Как можно заметить, в следствие большого разнообразия форматов данных и необходимости ими оперировать в ходе оперативных, плановых или внеплановых мероприятий обслуживания инфраструктуры возникает задача разработки алгоритма обработки данных, который менее сложен для применения человеком с учетом задействования нескольких из приведённых типов данных. В инфраструктуре OpenStack для автоматизации конфигурирования пользуется популярностью программное обеспечение Ansible. Так, к примеру проект Kolla-Ansible позволяет установить компоненты инфраструктуры OpenStack используя несколько видов входных данных: переменные конфигурации, описанные в формате YAML, переменные серверов с возможностью объединения в группы в формате INI и последовательность задач рабочей книги (ansible playbook) в формате YAML.

В основном, алгоритмы Ansible предполагают линейную сложность. В простом случае результирующая сложность решения рабочей книги из N задач эквивалентна $O(N)$. Время выполнения рабочей книги будет составлять сумму длительностей выполнения каждой из задач. Основным усложнением алгоритма или времени выполнения рабочей книги будет являться наличие циклов, способ их реализации, а также работа с несколькими источниками данных.

Простота создания рабочих книг обеспечивает прозрачность и последовательность выполнения действий, однако, когда количество значений в цикле исчисляется в порядке тысяч, обработка и оценка результатов может быть затруднена. Изменение параметров для тысячи дисков приводит к излишнему объему журналов. Кроме того, если необходимо изменить несколько параметров, многократные итерации по спискам объектов, совершаемые для изменения каждого из параметров становятся проблемой. В связи с этим, актуальна разработка специальных алгоритмов для осуществления выборки данных.

4 Формализация структур форматов данных пяти компонентов облачной инфраструктуры

Как показывает результат исследования, множество систем создаются без оглядки на возможность дальнейшей обработки и сопоставления с другими системами. Многообразие таких форматов, как YAML, XML, JSON представляет нереляционные данные, что делает невозможным применение SQL-запросов, применяемых в случае работы с табличными структурированными данными. Ввиду того, что облачная инфраструктура на текущий момент определяется непрерывно развивающимся и разрабатывающимся набором технологий, то компетенции работы с такими системами или обработки данных в них требуют некоторой алгоритмизации и вариативности. В частности, выделение единого подхода по работе данными является трудоемкой задачей, поэтому более рациональными являются предложения по обработке частных случаев реализаций некоторых структур данных.

Листинг 1: Различные форматы данных, встречающиеся в компонентах облачной инфраструктуры:

(а) Шаблон OpenStack Heat в формате YAML

```
heat_template_version: 2015-04-30
description: Simple template
resources:
  machine1:
    type: OS::Nova::Server properties:
      key_name: mtuci_researcher
      image: smolensk-vanilla-1.6.9.qcow2
      flavor: general-m8-c4
  machine2:
    type: OS::Nova::Server properties:
      key_name: mtuci_researcher
      image: smolensk-vanilla-1.6.9.qcow2
      flavor: general-m2-c2
```

(б) Список виртуальных машин OpenStack в формате JSON

```
[
    {
        "ID": "18467d72-5a4e-4887-94c1-e24fd2881354",
        "Name": "machine1",
        "Status": "ACTIVE",
        "Networks": {
            "public": [ "172.24.4.14",
                "2001:db8::184"
        ]
    },
        "Flavor": "general-m8-c4"
    },
    {
        "Comment": "Further output is truncated"
    }
]
```

(в) Конфигурация сети Docker в формате JSON

```
[
  {
    "Name": "docker_1",
    "Comment": "Some fields are skipped",
    "Containers": {
      "688f29a3278fdf1e97dec727...": {
          "Name": "postgres",
          "EndpointID": "7a8a31f28107a7946...",
          "MacAddress": "02:42:ac:19:00:03",
          "IPv4Address": "172.25.0.3/27",
          "IPv6Address": ""
      },
      "6d6e47321600b552c64518c1...": {
          "Name": "zabbix-web",
          "EndpointID": "824205c2e8e194f737...",
          "MacAddress": "02:42:ac:19:00:02",
          "IPv4Address": "172.25.0.2/27",
          "IPv6Address": ""
      }
    }
  }
]
```

(г) Конфигурация домена LibVirt в формате XML

```
<domain type='kvm' id='1'>
        <name>instance-00000003</name>
  <uuid>18467d72-5a4e-4887-94c1-e24fd2881354</uuid>
        <metadata>
            <nova:instance xmlns:nova="http://openstack...">
            <nova:package version="27.1.0"/>
            <nova:name>machine1</nova:name>
          </nova:instance>
        </metadata>
        <devices>
          <emulator>/usr/bin/qemu-system-x86_64</emulator>
          <disk type='block' device='disk'>
            <source dev='/dev/sdb' index='1'/>
            <backingStore/>
            <target dev='vda' bus='virtio'/>
            <alias name='virtio-disk0'/>
          </disk>
          <interface type='ethernet'>
            <mac address='fa:16:3e:69:e5:07'/>
            <target dev='tapa3a26e07-7e'/>
            <model type='virtio'/>
          </interface>
        </devices>
      </domain>
```

В листинге 1 показаны примеры структур данных в форматах YAML и JSON из реальных систем, являющихся компонентами облачной инфраструктуры. В листинге 1а приведен пример структуры шаблона создания виртуальных машин в OpenStack Heat, в листинге 1б – результат команды вывода списка серверов в OpenStack, в листинге 1в – результат команды вывода сетевых настроек Docker, а в листинге 1г – конфигурация домена виртуализации LibVirt. Все из структур данных представляют собой объекты, содержащие ключи и значения. Кроме того, спецификация форматов [8-10] позволяет в значения записывать другие объекты, таким образом достигается логическая древовидность или вложенность данных. При известном ключе, к примеру, имени ресурса, как показано в листинге 1а извлечь данные не составляет труда используя известный ключ machine1. Однако, когда ключом является идентификатор, как в листинге 1в извлечение данных представляет проблему при обработке человеком. В то же время, в листинге 1б приведен пример вывода команды OpenStack Client, отображающей список виртуальных машин. Она представляет собой плоскую структуру, а именно список объектов, содержащих ключ и значение. Поиск по такому списку осуществить достаточно легко при помощи нотации JMESPath [14], которая предоставляет возможность фильтрации по полям.

Таблица 1

Сравнение форматов и их структур данных в облачной инфраструктуре

Название формата	Структура данных	Область применения
CSV, TSV	Структурированные данные. Реляционная модель Значения, отделенные разделителем. Представляют собой табличные данные	Результаты SQL запросов Ввод данных человеком
JSON	Структурированные данные. Нереляционная модель Коллекция пар ключ-значение Массив значений *Примечание:* допустима вложенность, значением может быть как первая, так и вторая структура [8]	Запросы к API сервисов Ответы от API сервисов Параметры конфигураций
YAML	Структурированные данные. Нереляционная модель Пересекается по возможностям с JSON [9]	Параметры конфигураций
INI	Структурированные данные. Нереляционная модель Коллекция пар ключ-значение	Параметры конфигураций
XML	Структура из элементов с аттрибутами и текстовым значением. Может быть преобразован в JSON [11]	Параметры конфигураций
.log	Неструктурированные данные. Нереляционная модель Список из строк, содержащих временные отметки в определённом формате [12]	Диагностические журналы сервисов
.txt	Неструктурированные данные. Нереляционная модель Текст без определенного формата, поддающийся обработке и переводу в табличные данные	Вывод командной строки

Природа того, что разработчики систем организовывают структуры данных такими разными и несогласованными способами, остается явлением, наблюдаемым де-факто, и может иметь несколько причин.

Ведущими языками, используемыми в разработке облачных компонентов, являются Python и GoLang. Исходя из этого известно, к примеру, что в Python одной из классификаций типов данных являются хешируемые и нехешируемые. Тип данных словарь (dict) является хешируемым и по структуре соответствует виду формата JSON. В то же время, список (list) является нехешируемым. Более конкретно, в словаре индексируются ключи для целей быстрого обращения к ним. Это означает, что использование идентификаторов или имен виртуальных машин в качестве ключа носит природу технического решения, обеспечивающего хеширование поля для последующей быстроты доступа по известному ключу, однако сложность для человека при обработке таких данных остается заметной.

Разнообразие структур данных, встречаемых в компонентах облачной инфраструктуры, приводит к задаче их формализации и выделении разновидностей. В таблице 1 сведены сравнения форматов и их структур данных. Было предпринято несколько векторов исследований, чтобы проверить два предположения. Первое предположение ставит вопрос о существовании определённой единой структуры данных в которую возможно было бы преобразовать каждый из форматов. Второе предположение состоит в существовании инструмента, который позволял бы задействовать несколько источников данных и извлекать расширенную информацию об искомых запросах.

Было получено несколько выборок данных в форматах JSON, YAML и XML. В качестве инструмента обработки данных был выбран язык Python, как наиболее часто используемый в разработке облачных инфокоммуникационных систем.

В результате эксперимента было установлено, что каждый из форматов JSON, YAML, XML может быть преобразован в тип данных dict, который в свою очередь соответствует структуре, определяемой форматом JSON. Эта возможность открывает положительные перспективы в проверке второго предположения. JSON де-факто находит более широкое применение и использование, что означает наличие инструментария, входящего в большее число дистрибутивов ОС семейства Linux. После преобразования выборок в формат JSON был проведён анализ полученных структур на предмет выделения разновидностей.

В результате анализа исходных данных в листингах 1,2,3 были формализованы различные виды структур данных в формате JSON. Для работы с такими данными в Linux широкой популярностью пользуется инструмент jq [13]. В Ansible и Python используется нотация библиотеки JMESPath [14].

Оба инструмента обладают языком запросов достаточным, чтобы осуществлять фильтрацию списков, как в листингах 2,3. Тем не менее, фильтрация структуры, приведённой в листинге 1, данными инструментами не представляется возможной при помощи этих инструментов. В случае с листингом 1 возможно только извлечение значения по известному ключу.

Листинг 1: JSON структура вариант №1

```
{
    "key1": {
        "keyA": "valueA",
        "keyB": "valueB"
    },
    "key2": {
        "keyA": "valueA",
        "keyB": "valueB"
    }
}
```

Листинг 2: JSON структура вариант №2

```
{
    "list_of_objects": [
        {
            "name_of_key1": "value_of_key1",
            "keyA": "keyB",
            "keyB": "valueB"
        },
        {
            "name_of_key2": "value_of_key2",
            "keyA": "keyB",
            "keyB": "valueB"
        }
    ]
}
```

Листинг 3: JSON структура вариант №3

```
{
    "nested_list_of_objects": [
        {
            "keyA": "valueA",
            "keyB": "valueB",
            "keyC": {
                "keyCA": "valueCA",
                "keyCB": {
                    "keyCBA": "valueCBA",
                    "keyCBB": "valueCBB"
                },
                "keyCC": [
                    {
                        "keyCCA": "valueCCA",
                        "keyCCB": "valueCCB"
                    }
                ]
            }
        },
        {
            "keyA": "valueA",
            "keyB": "valueB"
        }
    ]
}
```

5 Обработка результатов из нескольких источников данных

Поскольку обозначенная цель состояла в нахождении возможности сопоставления данных из нескольких источников, задача преобразования данных должна проводиться с учетом существующих средств и языков запросов, которыми обладают специализированные для этого СУБД. Так, к примеру, в реляционных СУБД для объединения данных нескольких таблиц используется функция JOIN. Язык запросов нереляционной СУБД MongoDB содержит функции $objectToArray, $unwind, которые позволяют привести структуру к коллекции пар ключ-значение. Попытка привести исходные данные к табличному виду открывает положительные перспективы в дальнейшей обработке информации. СУБД SQLite3 предоставляет возможность работы из командной строки Unix-подобных систем, а также импортировать данные напрямую из текстовых файлов со значениями, разделенными табуляцией.

Дальнейшая выборка данных может быть сведена к использованию типовых SQL запросов.

Путь обработки данных листинга 1 для приведения к табличному виду состоит в преобразовании пар ключ-объект в такие пары, где ключу k соответствует значение имени ключа, а ключу v значение объекта. В листинге 4 отображен результат преобразований в коллекцию пар ключ-значение. Дополнительно в листингах 5, 6 показана разница в представления данных в табличном виде до и после преобразований. Очевидно, что последний вариант наиболее удобен для представления, поскольку предполагает наличие отдельных строк записей по каждому из ключей, в отличие от первого варианта.

Возможность работы с несколькими источниками данных актуальна, когда необходимо разрабатывать автоматизацию обслуживания облачной инфраструктуры. Это могут быть скрипты выгрузки данных, скрипты обеспечения надежности или инструменты, разрабатываемые на лету и необходимые решения возникающих аварий.

Такая автоматизация часто носит прикладной характер задачи и частное решение без необходимости системного решения. Обычно может быть необходимо сопоставлять данные из результатов SQL-запросов, текстовой информации, извлекаемой регулярными выражениями, а также JSON формата. Небольшое ознакомление с предметной областью исследования позволяет понять, что проблематика задачи имеет связь со анализом больших данных, в том числе затрагивается задача сопоставления неструктурированных и структурированных данных.

Стоит заметить, что на российском рынке труда на 2023 год при анализе вакансий не замечены требования к квалификации по обработке больших данных для инженеров по обеспечению надежности облачной инфраструктуры (SRE, Site Reliability Engineer) или инженеров по эксплуатации. Тем не менее, обработка этих данных неизбежна в условиях ограниченного инструментария, доступного из командной строки Linux. Применение специализированных инструментов обработки больших данных или их разработка может выходить за рамки квалификации инженера по обеспечению надежности или инженера по эксплуатации.

Листинг 4: Результат применения преобразования JSON по варианту №1 к коллекции пар ключ-значение

```
[
        {
                "k": "key1",
                "v": {
                        "keyA": "valueA",
                        "keyB": "valueB"
                }
        },
        {
                "k": "key2",
                "v": {
                        "keyA": "valueA",
                        "keyB": "valueB"
                }
        }
]
```

Листинг 5: Преобразование листинга 1 в таблицу формата TSV

```
key1.keyA key1.keyB key2.keyA key2.keyB
valueA   valueB   valueA   valueB
```

Листинг 6: Преобразование листинга 4 в таблицу формата TSV

```
k       v.keyAv.keyB
key1    valueAvalueB
key2    valueAvalueB
```

6 Алгоритм и эксперимент обработки облачных данных

С учетом выводов и наработок, полученных в предыдущих разделах, предлагается следующий алгоритм обработки облачных данных:

1. Определить к какому из вариантов структуры, приведённой в листингах 1,2,3 относятся входные данные в формате JSON.
2. Если структура данных похожа на отображенную в листинге 1, то её возможно преобразовать в коллекцию пар ключ-значение с последующим преобразованием в таблицу.
3. Если структура данных похожа на вариант в листинге 2, то такие данные можно преобразовать в таблицу напрямую.
4. Если структура данных похожа на вариант в листинге 3, то такие данные можно преобразовать в таблицу напрямую, однако вложенные коллекции пар ключ-значение не смогут быть раскрыты.

Для проверки алгоритма на практике рассматривается пример обработки исходных данных в разнородных форматах, определенных в листингах 1а,1б,1г. Используются возможности языка Python версии 3.11 и библиотеки os, sys, pandas, xmltodict, PyYaml и json. Выборка в формате YAML представляет собой структуру, не являющуюся коллекцией ключ-значение и похожа на вариант в листинге 1. Осуществляется чтение данных в тип данных dict с последующим сохранением в формат JSON. Над полученным результатом применяется преобразование значений resources в коллекцию пар ключ-значение с помощью фрагмента, определенного в листинге 12. Далее осуществляется нормализация данных при помощи функции pandas.json_normalize() и экспорт результата в формат значений, разделённых табуляцией (TSV). Результат преобразований исходных данных приведен в листинге 9.

Выборка в формате JSON не требует нормализации или других промежуточных преобразований и похожа на вариант в листинге 2. Поэтому, она преобразуется в формат TSV напрямую. Результат преобразований в формат TSV приведен в листинге 10. Выборка в формате XML была преобразована в dict с помощью библиотеки xmltodict. Дальнейшая последовательность действий состояла в тех же шагах, которые были предприняты с форматом YAML.

Дальнейшие шаги предприняты для проверки возможности осуществления запросов с использованием, преобразованных в формат TSV исходных данных. Среди доступных средств обработки данных из командной строки Unix-подобных систем предлагается использование инструментов GNU join [15] и SQLite3. Так, к примеру, в листинге 7 представлен скрипт, загружающий облачные данные, разделенные табуляцией, полученные после преобразований. Данные загружаются в отдельные таблицы и осуществляется объединение полей. В результате составленного запроса раскрыта возможность выборки данных, оригинальные наборы которых находились в различных форматах. В листинге 8 приведен вывод осуществленного запроса.

Для оценки того, что на выборках данных из реальных систем в соответствующем масштабе инфраструктуры

предлагаемые алгоритмы будут выполняться за разумный интервал времени, были проведены замеры времени выполнения преобразований различных операций. Преобразования осуществлялись на ноутбуке с ARM архитектурой в конфигурации [4×Ядра ЦП, 16×ГБ ОЗУ, SSD], что соответствует конфигурации m4.xlarge или mac2.metal по классификации Amazon AWS. Было проведено преобразование переменных

ПО Ansible, а именно файла hostvars.json объемом 175 МБ, соответствующего по структуре листингу 1. Структура содержимого файла была преобразована в коллекцию пар ключ-значение при помощи листинга 12. Преобразование было выполнено за 24 секунды. Содержащиеся данные о 28 серверах были заведомо избыточны и вложены. Данное время выполнения запроса удовлетворяет ситуациям, когда необходимо обработать данные при разработке краткосрочных скриптов.

Таким образом, было определено, что алгоритм имеет допустимую сходимость несмотря на применение к большой выборке данных.

Листинг 7: Запрос в БД SQLite3 для выборки по нескольким источникам данных в формате TSV

```
1  .mode tabs
2  .import os_servers.tsv os_servers
3  .import os_hot.tsv os_hot
4  .import domains.tsv domains
5  .headers on
6  select
7  os_servers."Name",
8  os_servers."ID",
9  os_servers."Networks.public",
10 os_hot."v.properties.key_name",
11 domains."devices.graphics.@port"
12 from os_servers
13 join domains on os_servers."ID" = domains."uuid"
14 join os_hot on os_hot.k = os_servers."Name";
```

Листинг 8: Результат запроса в БД SQLite3 для выборки по нескольким источникам данных в формате TSV

```
1  denis@mtuci-ws $ sqlite3 -batch < sqlite-q1.sql 2>/dev/null
2  Name                    ID
   Networks.public
         v.properties.key_name
                                 devices.graphics.@port
3  machine1                18467d72-5a4e-4887-94c1-e24fd2881354
['172.24.4.14', '2001:db8::184']
       mtuci_researcher                  5900
4  machine2                02add73d-31b4-4c75-9831-40089b74b6a8
['172.24.4.240', '2001:db8::2ce']
       mtuci_researcher                  5901
5  machine3                c84464cc-83d7-40b1-b9c3-e148c8b33cfa
['172.24.4.83', '2001:db8::31d']
       mtuci_researcher                  5902
```

7 Продолжение исследования

Предлагаемый алгоритм действий находит применение для большинства структур и выборок данных, которые встретились в компонентах облачной инфраструктуры на базе OpenStack. Тем не менее, не рассматривается его применение к структурам данных, имеющим вложенную структуру, как в листинге 3. Обработка таких данных языком реляционных запросов потребует выделения связей между ними для приведения к реляционной модели, либо, напротив, для обработки таких данных найдут применение специализированные языки запросов таких СУБД, как MongoDB или Apache Hive, как было рассмотрено авторами [16].

Кроме того, не рассматривались способы обработки неструктурированных данных, таких как файлы журналов событий. Ранее проблематика и подход к обнаружению событий в OpenStack докладывались авторами на конференции [17]. Авторы [18] предлагают конвейерный подход и собственный алгоритм к обработке неструктурированных данных на примере журналов событий.

Заключение

Компоненты облачной инфраструктуры представляют собой совокупность программного обеспечения, формат хранения и представления данных которого имеет вариативность. При необходимости немедленной автоматизации или алгоритмизации, как, к примеру, в случае аварий на инфраструктуре, консолидация данных из различных источников требует определённых подходов. Сравнительный анализ реальных структур данных, представляемых компонентами облачной инфраструктуры, позволил выделить три варианта представления облачных данных. Были выдвинуты гипотезы о существовании их преобразований в единый формат данных и о возможности обработке данных из нескольких источников.

Для проверки гипотез предложен способ и алгоритм обработки данных для возможности осуществления запросов поиска значений и их дальнейшего использования. Разработанный алгоритм способен понизить сложность разработки скриптов автоматизации, применяемых при обеспечении работоспособности облачной инфраструктуры.

Листинг 9: Результат преобразований выборки Hot Template из YAML в TSV

	k	v.type	v.properties.key_name	v.properties.image	v.properties.flavor
2	machine1		OS::Nova::Server	mtuci_researcher	smolensk-vanilla-1.6.9-qemu-mg8.0.0.qcow2
3	machine2		OS::Nova::Server	mtuci_researcher	smolensk-vanilla-1.6.9-qemu-mg8.0.0.qcow2
4	machine3		OS::Nova::Server	mtuci_researcher	smolensk-vanilla-1.6.9-qemu-mg8.0.0.qcow2

Листинг 10: Результат преобразований выборки Servers из JSON в TSV

	ID	Name	Status	Networks.net1
2	18467d72-...	machine1	ACTIVE	['172.24.4.14', "2001:db8::184"]
3	02add73d-...	machine2	ACTIVE	['172.24.4.45', "2001:db8::2ce"]
4	c84464cc-...	machine3	ACTIVE	['172.24.4.83', "2001:db8::31d"]

Листинг 11: Результат преобразований выборки Domain из XML в TSV

	@type	@id	name	uuid	devices.graphics.@type	devices.graphics.@port
2	kvm	1	instance-00000003	18467d72-...	vnc	5900
3	kvm	2	instance-00000004	02add73d-...	vnc	5901
4	kvm	3	instance-00000005	c84464cc-...	vnc	5902

Дальнейший интерес вызывает исследование возможности осуществления запросов среди вложенных структур данных, а также применение возможностей нереляционных СУБД для решения этой задачи.

Литература

1. *Dokuchaev V. A.* Digital Transformation: New Drivers and New Risks // 2020 International Conference on Engineering Management of Communication and Technology (EMCTECH), Vienna, Austria, 2020, pp. 1-7, doi: 10.1109/EMCTECH49634.2020.9261544.
2. *Petuhov D., Dokuchev V.* The quality indicators analysis on a world cloud market // REDS: Telecommunication systems and equipment, 2020.
3. *Maklachkova V.V., Dokuchaev V.A., Statev V.Y.* Risks Identification in the Exploitation of a Geographically Distributed Cloud Infrastructure for Storing Personal Data // 2020 International Conference on Engineering Management of Communication and Technology (EMCTECH), Vienna, Austria, 2020, pp. 1-6, doi: 10.1109/EMCTECH49634.2020.9261541.
4. *Dokuchaev V.A., Maklachkova V.V., Statev V.Y.* Information security in the big data space // T-Comm: Телекоммуникации и транспорт, 2022, vol. 16, no. 4, pp. 21-28.
5. *Dokuchaev V.A., Makarova D., Maklachkova V.V., Volkova M.L.* Analysis of Data Risk Management Methods for Personal Data Information Systems // 2020 Systems of Signals Generating and Processing in the Field of on Board Communications. 19-20 March 2020, DOI: 10.1109/IEEECONF48371.2020.9078538.
6. *Dokuchaev V.A., Gorban E.V., Maklachkova V.V.* The System of Indicators for Risk Assessment in High-Loaded Infocommunication Systems // 2019 Systems of Signals Generating and Processing in the Field of on Board Communications, Moscow, Russia, 2019, pp. 1-4, doi: 10.1109/SOSG.2019.8706726.
7. *Lopatina K., Dokuchaev V.A., Maklachkova V.V.* Data Risks Identification in Healthcare Sensor Networks // 2021 International Conference on Engineering Management of Communication and Technology (EMCTECH), Vienna, Austria, 2021, pp. 1-7, doi: 10.1109/EMCTECH53459.2021.9619178.
8. Введение в JSON // json.org [сайт]. URL: https://www.json.org/json-ru.html. дата обращения: 28.01.2023
9. YAML Specification // yaml.org [сайт]. URL: https://yaml.org/spec/1.2.2/. дата обращения: 23.02.2023.
10. XML Specification // w3.org [сайт]. URL: https://www.w3.org/TR/ xml. дата обращения: 23.02.2023.
11. xmltodict package // pypi.org [сайт]. URL: https://pypi.org/project/xmltodict/. дата обращения: 23.02.2023.
12. ISO 8601 // ISO [сайт]. URL: https://www.iso.org/ru/iso-8601-date-and-time-format.html. дата обращения: 23.02.2023.
13. JQ Basic Filters // stedolan.github.io [сайт]. URL: https://stedolan.github.io/jq/manual/#Basicfilters. дата обращения: 23.02.2023.
14. JMESPath object Projection // jmespath.org [сайт]. URL: https://jmes path.org/tutorial.html?highlight=projection#object-p rojections
15. Join, die.net, Available: https://linux.die.net/man/1/join.
16. *Зоткин А.С., Ворожцов А.С.* Большие данные: современные технологии обработки информации. МТУСИ. 2012.
17. *Петухов Д.А., Докучаев В.А.* Исследование подходов к обнаружению событий в инфраструктуре Openstack // Электронные системы и технологии : Сборник материалов 58-й научной конференции аспирантов, магистрантов и студентов БГУИР, Минск, 18-22 апреля 2022 года. Минск: Белорусский государственный университет информатики и радиоэлектроники, 2022. С. 179-182. EDN NJHWLL.
18. *Petukhov D., Dokuchaev V.A.* A Research of OpenStack Virtual Machine Backup Service Quality Indicators // 2022 International Conference on Engineering Management of Communication and Technology (EMCTECH), Vienna, Austria, 2022, pp. 1-6, doi: 10.1109/EMCTECH55220.2022.9934035.

CLOUD INFRASTRUCTURE COMPONENTS DATA QUERY ALGORITHMS

Denis A. Petukhov, *Moscow Technical University of Communication and Informatics, Moscow, Russia,* ***d.a.petukhov@edu.mtuci.ru***

Abstract

The digital transformation of companies leads to the transfer of many technological and business processes to the cloud, which leads to new risks of confidential information presented in different data formats in the cloud infrastructure, as well as the complexity and heterogeneity of the cloud infrastructure. Cloud infocommunication systems built upon open-source software components expose variabilities in data formats and structures. Such data formats and structures may be convenient and reasonable when developing cloud components, although working with it during operation and maintenance is accompanied with the problem of a mismatch between the varieties and with the lack of special tools to query among the multiple data formats. In the article such data formats as YAML, JSON and XML are considered and thus an opportunity of its transformation into a single format is taken as a hypothesis. Considering analysis results of original data samples taken from OpenStack components three typical data structures are proposed. The other hypothesis is about the opportunity to join data taken from multiple sources. In order to verify assumptions, the algorithms for data transformation into a tabular format are developed. The quickness of transformations is verified on a large sample and obtained results are acceptable for usage while operating and maintaining cloud infocommunication systems. After applying the proposed algorithms to the original data samples an SQL query and a way to query on multiple data sources are proposed. The results of research may have an application for the cloud infrastructure availability solutions when working with multiple data formats.

Keywords: *cloud infrastructure, OpenStack, data formats, algorithms, SQL*

References

1. V.A. Dokuchaev, "Digital Transformation: New Drivers and New Risks," *2020 International Conference on Engineering Management of Communication and Technology (EMCTECH)*, Vienna, Austria, 2020, pp. 1-7, doi: 10.1109/EMCTECH49634.2020.9261544.
2. D. Petuhov and V. Dokuchev, "The quality indicators analysis on a world cloud market," *REDS: Telecommunication systems and equipment,* 2020.
3. V. . Maklachkova, V.A. Dokuchaev and V.Y. Statev, "Risks Identification in the Exploitation of a Geographically Distributed Cloud Infrastructure for Storing Personal Data," *2020 International Conference on Engineering Management of Communication and Technology (EMCTECH)*, Vienna, Austria, 2020, pp. 1-6, doi: 10.1109/EMCTECH49634.2020.9261541.
4. V.A. Dokuchaev, V.V. Maklachkova and V.Y. Statev, "Information security in the big data space," *T-Comm*, 2022, vol. 16, no. 4, pp. 21-28.
5. V.A. Dokuchaev, D. Makarova, V.V. Maklachkova, M.L. Volkova, "Analysis of Data Risk Management Methods for Personal Data Information Systems," *2020 Systems of Signals Generating and Processing in the Field of on Board Communications*. 19-20 March 2020, DOI: 10.1109/IEEECONF48371.2020.9078538.
6. V.A. Dokuchaev, E.V. Gorban and V.V. Maklachkova, "The System of Indicators for Risk Assessment in High-Loaded Infocommunication Systems," *2019 Systems of Signals Generating and Processing in the Field of on Board Communications*, Moscow, Russia, 2019, pp. 1-4, doi: 10.1109/SOSG.2019.8706726.
7. K. Lopatina, V.A. Dokuchaev and V.V. Maklachkova, "Data Risks Identification in Healthcare Sensor Networks," *2021 International Conference on Engineering Management of Communication and Technology (EMCTECH)*, Vienna, Austria, 2021, pp. 1-7, doi: 10.1109/EMCTECH53459.2021.9619178.
8. Introducing JSON, json.org, Available: https://www.json.org/json-ru.html.
9. YAML Specification // yaml.org [сайт]. URL: https://yaml.org/s pec/1.2.2/.
10. XML Specification // w3.org [сайт]. URL: https://www.w3.org/TR/ xml.
11. xmltodict package // pypi.org [сайт]. URL: https://pypi.org/pro ject/xmltodict/.
12. ISO 8601 // ISO [сайт]. URL: https://www.iso.org/ru/iso-860 1-date-and-time-format.html.
13. JQ Basic Filters // stedolan.github.io [сайт]. URL: https://stedolan.github.io/jq/manual/#Basicfilters.
14. JMESPath object Projection // jmespath.org [сайт]. URL: https://jmes path.org/tutorial.html?highlight=projection#object-p rojections
15. Join, die.net, Available: https://linux.die.net/man/1/join.
16. A.S. Zotkin and A.S. Vorozhtsov, "Big data: relevant data processing techologies", MTUCI, 2012. (in Russian)
17. D.A. Petukhov, V.A. Dokuchaev, "A research of OpenStack events discovery approaches," *Proceedings of 58-th BGUIR Scientific Conference for students, bachelors and masters*, 2022, p. 179-182 (in Russian)
18. D. Petukhov and V.A. Dokuchaev, "A Research of OpenStack Virtual Machine Backup Service Quality Indicators," *2022 International Conference on Engineering Management of Communication and Technology (EMCTECH),* Vienna, Austria, 2022, pp. 1-6, doi: 10.1109/EMCTECH55220.2022.9934035.