Санкт-Петербургский государственный университет

Направление “Математическое обеспечение и администрирование информационных систем”

Профиль “Информационно-аналитические системы”

Тимофеев Богдан Михайлович

Разработка масштабируемой системы управления данными, обладающей высокой доступностью

Бакалаврская работа

Научный руководитель:

доктор физико-математических наук профессор Нестеров В.М.

Рецензент:

старший преподаватель Луцив Дмитрий Вадимович

Санкт-Петербург

2016

SAINT-PETERSBURG STATE UNIVERSITY

Main Field of Study “Software and administration of information systems”

Area of Specialisation “Analitical information Systems”

Timofeev Bogdan Mikhailovich

Development of scalable high-availability data management platform

Bachelor’s Thesis

Scientific supervisor:

doctor of physico-mathematical sciences professor Nesterov V.M.

Reviewer:

D.V.Luciv

Saint-Petersburg

2016

Оглавление

Введение

В связи со стремительным ростом объемов хранимой информации становятся все более актуальны системы, позволяющие хранить и оперировать большим количеством слабоструктурированных данных. В данной работе рассмотрен подход к хранению информации, основанный на использовании различных способов хранения и использования метаданных, а также примеры применения этой методологии.

Данная выпускная квалификационная работа заключается в разработке некоммерческого программного решения для раздельного управления данными и их метаданными. Проектируемая система будет иметь доступ к хранилищу данных , в качестве которого будет использоваться один из существующих продуктов с открытой лицензией, а также к хранилищу метаданных, способ хранения которых предстоит определить в процессе исследования. Предлагаемое решение будет использовать систему управления метаданными, которая обеспечит быстрый доступ к ним (в том числе с использованием распределенного кэша). Среди основных свойств разрабатываемой платформы важную роль играет масштабируемость (сохранение производительности при увеличениях количества хранимых данных и количества подключенных хранилищ) и высокая доступность — способность восстанавливаться после сбоев. Система должна также поддерживать работу многих пользователей, их аутентификацию и разделение прав доступа, а также обеспечивать консистентность данных при помощи одного из существующих механизмов. Среди перспективных направлений в разработке платформы следует выделить создание модуля политиками, который позволит пользователям с помощью специального интерфейса самим определять правила, по которым будут хранится данные и метаданные, а также реализацию балансировщика нагрузки на систему.

Постановка задачи заключается в разработке системы для хранения данных на основе программных компонент с открытым исходным кодом, использующей метаданные для поиска информации, обладающей свойствами масштабируемости и высокой доступности

В работе производится анализ существующих современных компонентов для программного хранения данных, выбор походящих для обеспечения вышеперечисленных требований к системе, а также совмещение их с теми частями системы, которые будут реализованы “с нуля”.

Архитектура

В данном разделе описываются основные архитектурные особенности систем наподобие реализованной в ходе данной квалификационной работы.

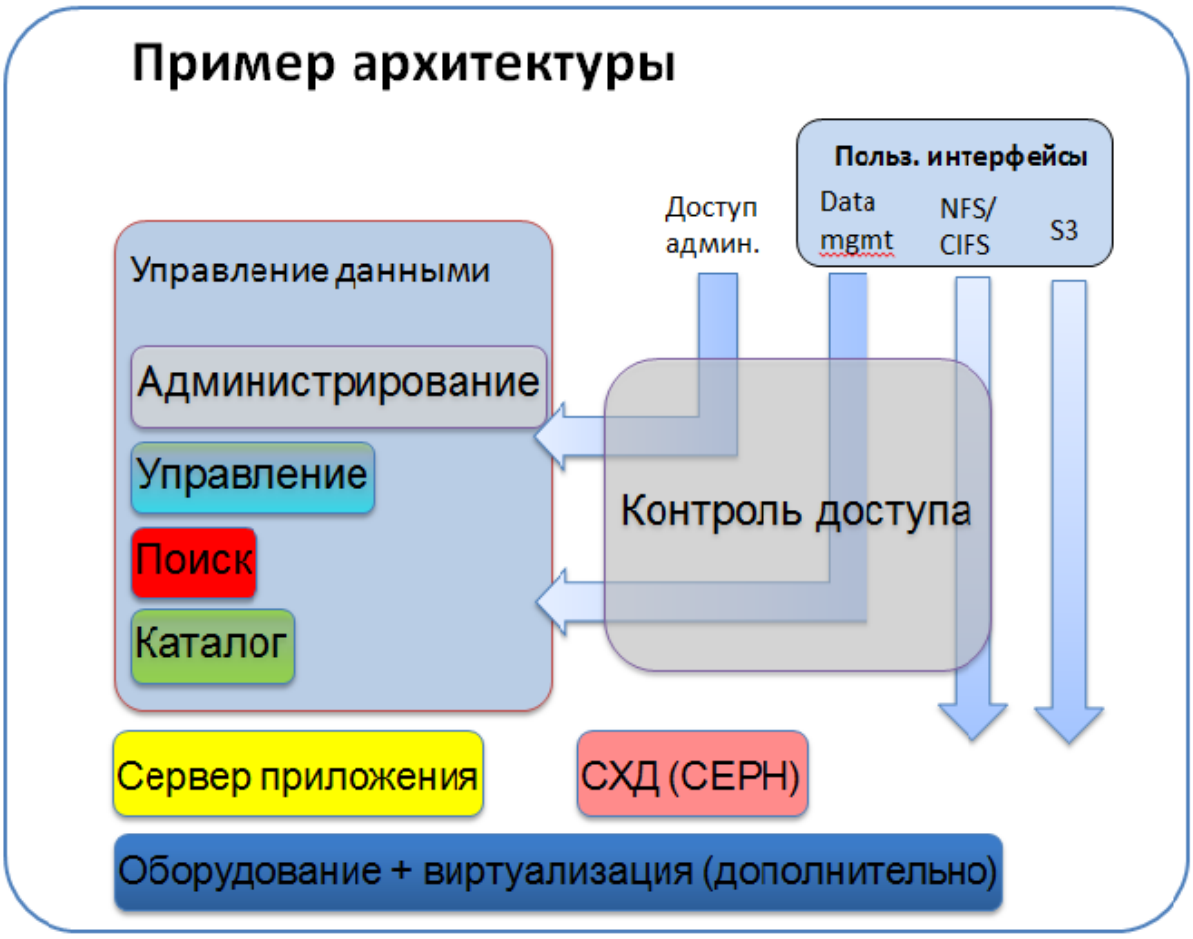

Рис.1 Пример архитектуры подобного рода систем

Основные компоненты системы

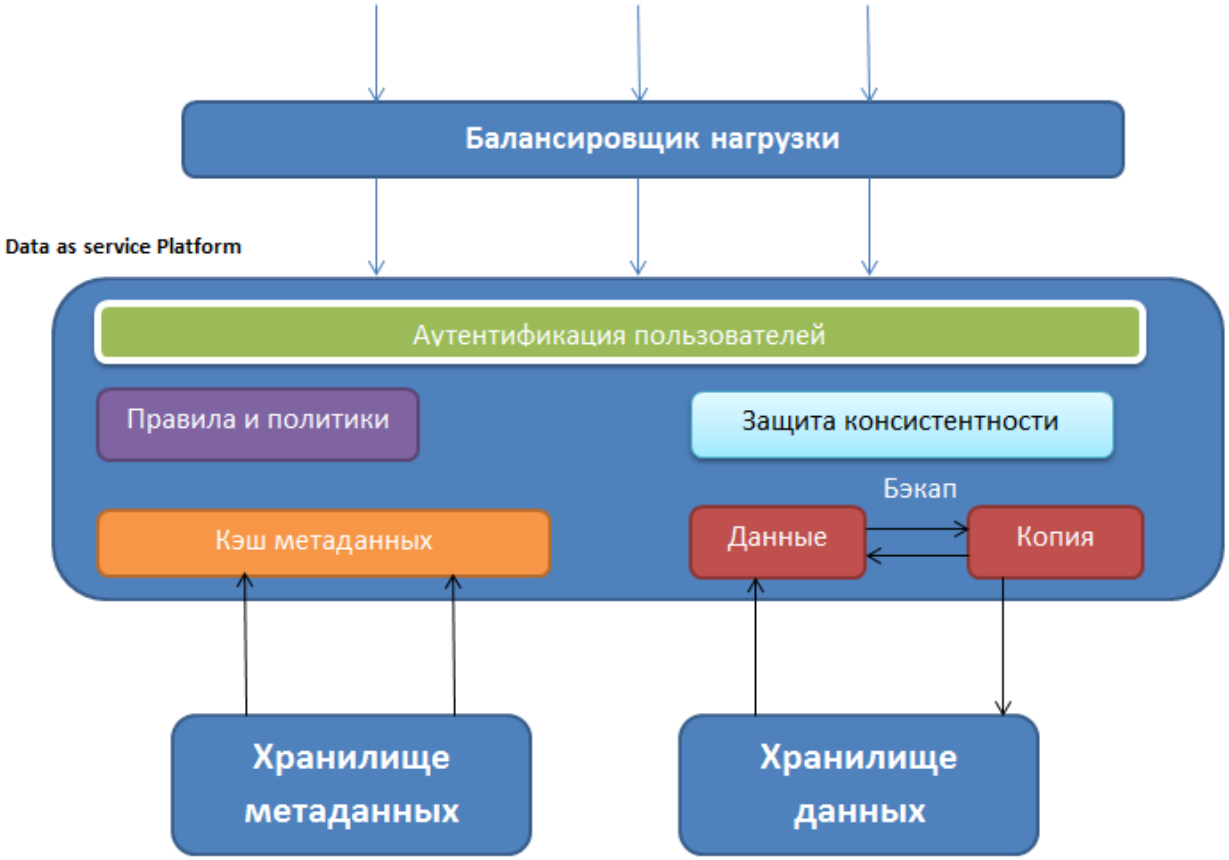

Рис. 2 Основные компоненты системы

Непосредственно платформа управления данными состоит из следующих основных компонентов:

- Механизм поддержки консистентности данных
- Система аутентификации пользователей
- Механизм резервного копирования данных
- Модуль управления политиками и правилами
- Балансировщик нагрузки
- Распределенный кэш метаданных

Data Storage Platform

Данный компонент системы представляет собой существующую систему хранения данных. Определяющую роль в ее выборе играет не тип хранения информации (файловый, блочный или объектный), а масштабируемость, также предполагается, что эта система должна быть в открытом доступе (open-source).

Возможные решения:

- Ceph (http://ceph.com/)
- OpenStack Swift (http://docs.openstack.org/developer/swift/)
- Apache Hadoop FS (https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html)
- EMC Isilon (https://www.emc.com/en-us/storage/isilon/index.htm), VNX (https://www.emc.com/en-us/storage/vnx.htm) — коммерческие системы

Требуемая платформа должна также иметь собственный модуль авторизации пользователей для контроля доступа к хранимым данным. Для получения данных пользователям необходимо использовать API используемой системы хранения. Возможны 2 основных типа доступа к системе хранения: файловый и объектный, реализация платформы управления данными не должна зависеть от конкретного интерфейса.

В случае использования коммерческого решения надежность хранения данных существенно возрастет, также будет предоставлена поддержка по конфигурации и экспулатации, однако это негативно скажется на стоимости системы, что и определяет использование платформ с открытым исходным кодом.

Для удобства оперирования данными, в системах управления данных может быть использована древовидная структура, напоминающая файловую систему современных операционных систем, в связи с этим будут введены два определения:

Объект – неделимая единица информации, не содержащая в себе ссылок на другие части данных, может представлять из себя файл или множество файлов, объект в смысле Amazon S3, таблицу, рисунок и т.п.

Датасет – “множество данных”, логическая единица информации, содержащая в себе ссылки на другие датасеты или объекты. Для датасетов также определены права доступа, о которых написано в разделе “Средство аутентификации пользователей”.

Metadata storage system

Metadata Storage system — компонент системы, в котором содержатся атрибуты сущностей из Data Storage System. С точки зрения реализации, может быть представлен в виде базы данных, в которой хранятся данные в виде “название атрибута : значение”, а также уникальный идентификатор соответствующей сущности из Data Storage System.

Платформа должна уметь находить метаданные по ключу (этот ключ может представлять из себя путь файла в Data Storage System, или же его хэш-значение). Сами же метаданные представляют из себя список из пар “ключ : значение”, таким образом, эту структуру будет достаточно удобно хранить в формате JSON, или в подобных форматах.

Также при выборе платформы необходимо учесть индексатор, который будет использоваться в системе. Отдельно стоит выделить Lucene (https://lucene.apache.org/core/).

Возможные решения:

- Mongo DB (https://www.mongodb.com/)
- Arango DB (https://www.arangodb.com/)
- …

В качестве облачной платформы, в которой будет запущена база данных, подходящим вариантом является Pivotal Cloud Foundry (http://pivotal.io/platform) в силу удобства и простоты использования.

Модуль управления метаданными (работающий с сущностями из системы хранения метаданных) должен обрабатывать запросы пользователей, связанные с поиском информации по ее атрибутам, соответсвенно необходима возможность обработки и выполения поисковых запросов разного уровня сложности.

Средство аутентификации пользователей

С помощью этого компонента система защищается от несанкционированного доступа: работать с содержимым хранилища могут лишь авторизованные пользователи. Возможно создание нескольких ролей пользователей в зависимости от их прав. Каждая коллекция (датасет) имеет список пользоваталей, которым разрешен доступ к ней. Права доступа к объектам определяются датасетами, в которых они содержатся.

Аутентификация может быть реализована не с нуля, а за счет авторизации с использованием аккаунтов Google или Facebook. В таком случае пользователь может получить минимальные права на использование системы, только системный администратор может изменить его полномочия.

В случае использования языка Java удобным инструментом для реализации аутентификации является Spring Security (http://projects.spring.io/spring-security/).

Механизм резервного копирования данных

Механизм резервного копирования данных создает резервные копии данных, хранящихся в системе, с помощью методологии continuous protection. Также, в случае потери файла (или же наличия его метаданных без самого файла), должно производится его восстановление (recovery) из копии.

Таким образом, механизм восстановления должен вызываться всякий раз когда система по пути файла нашла метаданные, не обнаружив самого файла. Восстановление должно также производится в случае наличия информации о том, какие данные были утеряны (их пути, или пути коллекций).

Одним из средств защиты данных от потери является счётчик ссылок — частный случай сборщика мусора, эффективный при не очень частом обращении к объектам и при отсутствии циклических ссылок. . Основная идея подобного механизма состоит в том, чтобы вести подсчет ссылок на единицу данных, и удалять ее в том и только в том случае, когда количество ссылок стало равным нулю.

Утилита для скачивания данных

В первоначальной версии системы загрузка данных в локальное хранилище пользователя будет производиться напрямую через интерфейс системы хранения данных. Пользователь будет запрашивать доступ к датасету (объекту) по его уникальному идентификатору, и в случае успешной авторизации он получит ссылку, по которой он по протоколу HTTP сможет получить необходимую информацию.

В дальнейшем планируется разработка специального программного обеспечения (возможно, с графическим интерфейсом пользователя), которое будет производить скачивание данных в какую-то директорию пользователя, требуя для этого лишь авторизацию пользователя. Возможно построение решения с использованием других протоколов передачи файлов (например FTP или WebDAV).

Модуль управления политиками и правилами

Модуль управления политиками и правилами позволяют пользователям самим устанавливать алгоритимы, согласно которым будет производится хранение и доступ к данным. Подобный механизм существует в системе iRODS - The Integrated Rule-Oriented Data System (http://irods.org/). Само средство управления политиками можно представить в виде правил, которые вводятся пользователем. По введенным правилам система определяет набор команд, которые необходимо выполнять для их соблюдения.

Пример правила (система iRODS):

```
acPostProcForPut {
     on($objPath like "*.txt") {
          msiDataObjCopy($objPath,"$objPath.copy");
     }
}
```

В данном правиле, acPostProcForPut – название события, обрабатываемого в системе iRODS), система автоматически исполняет это правило (копирование файла в директорию, если это текстовый файл) в случае наступления события загрузки файла, вызов похож на вызов триггера в СУБД.

Данный компонент не планируется в первоначальной версии системы.

Балансировщик нагрузки

Данный компонент системы выполняет роль сохранения быстрого отклика системы. В случае, если нагрузка на одно сетевое устройство значитаельно превышает нагрузку на другие, то балансировщик нагрузки перераспределит задания для выравнивания загруженности хостов. Для этого хосты должны быть универсальными, т. е. иметь унифицированный доступ к хранилищу данных и метаданных.

Существует много различных алгоритмов для балансировки нагрузки, выбор зависит от конкретных требований к системе. В целом, балансировка нагрузки должна обладать следующими свойствами:

- справедливость (равный подход к обработке каждого запроса)
- эффективность (все серверы должны быть заняты на 100%)
- сокращение времени выполнения запроса
- сокращение времени отклика

Одним из наиболее известных алгоритмов балансировки нагрузки является алгоритм Round Robin (алгоритм кругового обслуживания): первый запрос выполняется первым сервером, второй – вторым и т.д., затем процесс повторяется. Данный алгоритм не требует связи между серверами.

Распределенный кэш метаданных

С помощью распределенного кэша метаданных система может увеличить время работы за счет локального сохранения данных из хранилища метаданных. Специально для таких целей в системе Ceph существует компонент под названием Cache pool (pool — каталог с данными, cache pool — кэш конкретного пула, может быть использован при хранении метаданных в системе Ceph), позволяющий обеспечить быстрый доступ к необходимым метаданным. Система должна определить какие метаданные с большой долей вероятности понадобятся пользователю. Возможно сохранять в кэш все запрошенные метаданные, и удалять их оттуда по истечению какого-то промежутка времени (использовать таймер для каждой единицы метаданных).

Сценарии использования

Разрабатываемая система имеет большое количество разных типов сценариев использования в зависимости от целей ее пользователей. Ниже будут разбраны основные из них.

Общий сценарий использования

В начальном состоянии системы у нее существует единственный пользователь – системный администратор. Далее в системе могут зарегистрироваться новые пользователи, у каждого из которых имеется некоторая квота в системе (максимально допустимый объем данных и метаданных). Каждый пользователь получает свой userspace(пространство для хранения данных в пределах квоты), в котором он может создавать датасеты и объекты. В случае, если им необходима большая квота, они могут обратиться к системный администратору, который может их устанавливать для каждого пользователя. В случае, если какому-то из администраторов данных (пользователей, имеющих права создавать датасеты) нужна помощь в управлении своими данными, он может настроить права доступа к своим датасетам необходимым ему образом. В случае, если пользователь имеет доступ к датасету, он имеет доступ ко всем содержащимся внутри него объектам и вложенным датасетам.

Для скачивания данных из объектов, пользователи должны предоставить их идентификатор, после этого система принимает решение о предоставлении доступа. В случае, если авторизация прошла успешно, пользователь получает ссылки на расположение файлов, откуда их можно скачать.

Все данные, хранимые в системе, можно попытаться найти с помощью поиска по метаданным. Пользователь вводит поисковой запрос, в котором указываются параметры поиска (например, размер файлов). Таким образом можно будет искать лишь датасеты. В случае, если у пользователя нет прав на чтение метаданных у найденного датасета, он сможет получить лишь их урезанный набор, который будет включать в себя контакты владельца, с которым нужно договориться о предоставлении доступа.

В случае удаления пользователем данных из системы, система не удаляет их окончательно, а проверяет наличие ссылок на них. В случае, если с ними работает какой-либо другой пользователь, он не потеряет данные. Если данные должны быть удалены для всех пользователей, их системный администратор должен сначала очистить список пользователей, которые могут с ними работать.

Пример развёртывания, конфигурации и использования системы

Данная платформа может найти применение в различных областях, требующих управление данных с разграничением доступа. Одним из таких примеров может являтся научное объединение в сфере биоинформатики.

Одной из ключевых задач геномных исследований является проверка влияния того или иного гена на здоровье человека. Допустим, что какому-то научно-исследовательскому институту необходимо обнаружить связь между мутацией в каком-либо гене и заболеванием. Для того, чтобы проводить такие исследования, необходимо секвенирование большого количества билогического материала.

Пользователями системы в данном примере являются специалист по биоинформатике (ученый), который является Data Steward данной системы, а также сотрудники лаборатории (обычные пользователи), которые будут работать с материалом.

На начальном этапе системный администратор системы создает аккаунт для ученого и подтверждает его права как Data Steward-а. Системный администратор также выделяет ему квоту — 1000TB, за превышение которой данный НИИ будет оштрафован. После этого системный администратор регистрирует сотрудников лаборатории как обычных пользователей.

После регистрации пользователей специалист по биоинформатке создает в данной системе датасет для результатов исследования (датасеты могут быть вложенными). Сам датасет обладает необходимыми метаданными — дата создания, имя руководителя эксперимента, имена лаборантов и т.д. .Также ученый указывает тип видимости датасета для других пользователей (конфиденциальный или нет). Если датасет помечен как скрытый, то другие пользователи не смогут его увидеть при поиске по датасетам. После этого ученый задает список пользователей, которым разрешено что-либо записывать в данную коллекцию (и читать из нее). В данном примере такими пользователями являются сотрудники лаборатории.

На следующем этапе лаборанты получают биологический материал и производят его секвенирование, записывая все полученные результаты в датасет. Поскольку вся информация в ней является конфиденциальной, специалист по биоинформатике запрещает им дальнейший доступ к коллекции. С помощью API данной системы ученый настраивает свои приложения для работы с ней. В случае, если его опыта оказывается недостаточно для анализа предоставленной информации, он может прибегнуть к помощи других ученых, открыв им доступ на чтение к датасету.

В это время системный администратор системы может с помощью системы логирования за потреблением ресурсов системы. В частности, он может заметить что

какой-либо из пользователей злоупотребляет своей квотой, используя ее, судя по всему, не по назначению, либо же сильно превышает ее.

В общем случае все пользователи системы могут просматривать датасеты (получать их метаданные), которые в ней содержатся (кроме скрытых), имея доступ на чтение или запись лишь к тем, которые относятся к их работе.

Одно из отличий данной платформы от обычной системы – возможность существования двух сущностей с одинаковыми именами в одном датасете, в таком случае будут различаться лишь уникальные идентификаторы сущностей.

Работа с метаданными

Типичный сценарий работы с метаданными выглядит следующим образом:

1) Пользователь, которому необходимы метаданные, отправляет поисковой запрос к системе, например поиск по всем датасетам, которые имеют отношение к котировкам акций.

2) После обработки запроса пользователь получает ответ в виде списка идентификаторов датасетов и их системный администраторов.

3) Пользователь обращается к системный администраторам датасетов для получения доступа к ним.

4) Менеджеры датасетов принимают решение о предоставлении доступа к их данным (возможно, на коммерческих основаниях).

5) Пользователь получает доступ к некоторым датасетам и по их идентификатору может получить метаданные.

6) Если после анализа метаданных пользователь решит, что это данные из датасета ему пригодятся, то он может сделать запрос на получение доступа на чтение.

Для обработки пользовательских запросов система должна обладать каким-то API, в разработанной мной системе его роль играет REST API.

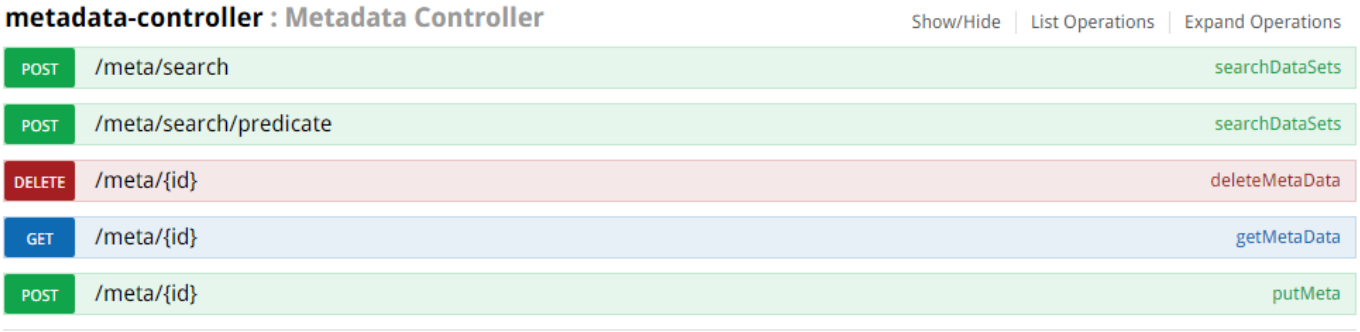

Рис. 3 HTTP-методы для работы с метаданными в разработанной системе, документация составлена при помощи Swagger(http://swagger.io/swagger-ui/)

Разделение прав доступа

Данная система предлагает пользователям достаточно большой диапазон возможностей по разграничению прав доступа к данным и метаданным.

Предположим, что в системе зарегестрирован пользователь, являющийся системный администратором некоторого датасета с информацией о торгах на бирже. Данный датасет обладает метаданными, которые системный администратор предпочитает скрывать от посторонних (например, сведения о назначении датасета).

После создания вышеописанной коллекции пользователю может понадобится каким-то образом поделиться созданными данными с другими пользователями системы. Для этого он предоставляет доступ на чтение данных и метаданных для биржевых аналитиков, доступ на чтение метаданных потенциальным покупателям этой информации, доступ на запись для брокера, отслеживающего текущую ситуацию на бирже. Системный администратор может также разграничить права на создание и редактирование файлов, а также создать новых системный администраторов датасета.

Пользователи, не имеющие отношения к данной коллекции не смогут получить данные, а при запросе на получение метаданных получат лишь контактную информацию системный администраторов, с которыми можно связаться для получения необходимого доступа.

Немаловажную роль в разделении прав доступа может играть наличие такого механизма на уровне системы хранения данных. Таким образом, в случае существования средства разделения прав доступа, можно проверять наличие у пользователя прав на использование данных при их скачивании, то есть их после первой проверки в системе управления данными. В таком случае, защищенность системы от несанкционированного доступа возрастает.

Защита от несанкционированного доступа

Пользователи системы, нуждающиеся в скачивании данных, должны для этого воспользоваться API системы хранения. В случае использования системы хранения Ceph, пользователи с подтвержденным доступом, получают ссылку, по которой они могут получить необходимые данные.

Допустим, пользователь системы, обладающий необходимыми правами, нуждается в загрузке определенных данных. Пройдя авторизацию, он получает ссылку на них в системе хранения. После этого он предоставляет эту ссылку пользователю, не имеющему необходимых прав, и этот пользователь не сможет скачать информацию из-за синхронизации прав доступа к реализуемой платформе и системы авторизации системы хранения.

Восстановление данных

Одно из наиболее важных свойст разрабатываемой системы — защита данных от потерь. В контексте данной работы будут рассматриваться более высокоуровневые конфигурации для защиты данных от потери чем, например, RAID.

Пусть пользователю понадобилось скачать какие-то данные. В таком случае он получает расположение желаемых данных, воспользовавшись поиском по метаданным. Далее, используя полученные адреса, он обращается к системе хранения данных. Система хранения, не обнаружив данные по адресам, вызывает механизм восстановления данных.

Данные также могут восстановлены в случае незапланированного их удаления пользователем (своеобразный аналог “корзины” из ОС Windows). Пользователь системы, удалив какие-то данные, сможет восстановить их в течение какого-то промежутка времени, по прошествии которого копии данных будут также удалены.

Окончательное удаление данных должно производится только в том случае, если ссылки на них отсутствуют. Таким образом, если были удалены все метаданные, в которых содержалась ссылка на место хранения соответствующей информации, то это означае то, что хранимые данные больше не востребованы и могут быть удалены. Для проверки наличия ссылок существует 2 подхода: подсчет количества ссылок раз в определенный период времени и использование счетчика ссылок, который хранится в метаданных.

Роли пользователей

Подобного рода системы предполагают наличие ролей пользователей, которые определяют их права в системе (например, право создания датасетов). Права пользователей также могут определяться отдельно для каждого датасета.

В контексте данной работы предполагается наличие трех типов пользователей:

- Системный администратор
- Data Steward (Системный администратор данных)
- Обычный пользователь

Все эти роли обладают различными правами.. Точное количество ролей пользователей может варьироваться в зависимости от системы, также возможно предоставление системному администратору прав на создание дополнительных пользовательских ролей.

Системный администратор

Системный администратор отвечает за управление правами всех пользователей платформы. Он может наделять их дополнительными полномочиями, а также эти полномочия снимать, определять квоты на использование хранилища, просматривать список созданных ими датасетов (в том числе и скрытых). Также системный администратор может создавать группы пользователей и наделять их определенными полномочиями по управлению данными.

Data steward

Data Steward (администратор данных) – пользователь, имеющий все права обычных пользователей, а также право администрировать датасеты. Он может в пределах выделенной ему квоты (ее превышение увидит системный администратор), создавать и удалять датасеты, менять их метаданные, а также настраивать список пользователей которые имеют доступ к данному датасету (доступ подразумевает возможность создавать объекты в датасетах и удалять их). Data Steward также в праве помечать датасеты как невидимые (их нельзя обнаружить при поиске по метаданным).

Обычный пользователь

Обычный пользователь системы может воспользоваться поиском по метаданным для поиска датасета в системе (просматривать его содержимое он не может без одобрения Data Steward-a, в том числе и вложенные датасеты). Он также может получить метаданные всех датасетов (первого уровня вложенности) системы, кроме тех, которые Data Steward-ом как невидимые. В случае получения доступа к датасету, пользователь может создавать и удалять из него файлы, но для доступа к вложенным датасетам нужно будет разрешение Data Steward.

Поддержка консистентности данных

Существует два основных подхода для защиты данных от ошибок, которые может повлечь за собой параллельный доступ разных пользователей:

- использование блокировок
- использование системы контроля версий

Оба этих подхода обладают своими достоинствами и недостатками, выбор того или иного метода должен быть обоснован исходя из конкретных требований к системе, ниже представлено более подробное описание каждого из них.

Контроль версий

На первоначальном этапе разработки предполагается, что объекты внутри датасетов нельзя изменять, можно лишь удалять и добавлять новые. В дальнейшем имеет смысл реализовать местную систему контроля версий, заключающуюся в том, что каждый пользователь работает со своей версией объекта.

Пусть пользователь 1 (П1) и пользователь 2 (П2) имеют доступ на запись к одному и ту же объекту. Тогда возможна следующая ситуация

- П1 запускает задачу, которая через 2 часа должна взять данные из объекта
- В это время П2 просматривает объект и меняет в нем какие-то данные
- Задача, запущенная П1, выдает неправильный результат из-за измененного файла

Такого случая можно было бы избежать, используя контроль версий. Пользователь 2 тогда бы изменил бы не сам объект, а лишь свою версию объекта. Такой сценарий похож на систему контроля версий, где каждый пользователь обладает веткой, куда он вносит свои изменения, при необходимости объединяясь с ветками другими пользователь.
После работы с датасетом (каждый со своей версией), пользователи могут принять решение об объединении своих версий в одну (главную), система в таком случае сравнит их и попытается создать единую модель (указав на возможные конфликты), а их версии для экономии памяти будут удалены.

Блокировки

Еще одной методологией для поддержки консистенстности объектов при параллельном доступе является использование блокировок. Таким образом, алгоритм, приведенный в предыдущем разделе, изменился бы следующим образом:

- П1 запускает задачу, которая через 2 часа должна взять данные из объекта, и ставит на него блокировку на изменение
- В это время П2 просматривает объект, но ничего поменять не может (но может его скачать)
- Задача, запущенная П1, выдает правильный результат, блокировка на запись снимается

После этого, пользователь 2, в случае необходимости (и согласования с пользователем 1), меняет содержимое объекта.

В случае, если обязательным требованием к системе является масштабируемость, использование блокировок является нежелательным, в случае немасштабируемых решений блокировки являются простым и надежным способом поддержки консистентности.

Компоненты реализованной системы

Данный раздел посвящен описанию компонентов реализованной системы. Система реализована на языке Java в силу большого количества библиотек с открытым исходным кодом для работы с данными.

Модуль управления метаданными

Данный компонент системы отвечает за получение, создание, поиск и удаление метаданных.

1)Добавление метаданных
Для добавления метаданных к находящейся в системе сущности (объекта или датасета), необходимо указать ее id, а также JSON-представление метаданных.

2) Для получения и удаления метаданных необходим только идентификатор сущности. Если у пользователя нет прав доступа к метаданным, то он получит лишь список системный администраторов сущности (чтобы обратиться к ним).

3) Поиск метаданных производится с помощью отправления списка полей и значений, которые эти поля должны принимать (например: “дата создания”:”14/02/1996” и т. д.). Пользователь, инициировавший запрос, получит список метаданных объектов.

Данный компонент системы тесно интегрирован с модулем авторизации пользователей, поскольку все операции над метаданными требуют определенных прав доступа.

В разработанной системе для управления метаданными разработан механизм на языке Java, для хранения метаданных используется SQL-база данных HSQLDB (http://hsqldb.org/). в силу удобства ее интеграции с языком Java.

Модуль управления данными

Модуль управления данными — компонент системы, в котором реализованы основные методы пользовательского доступа к данным, такие как создание и удаление. В готовом прототипе данные считаются немутабельными.

Для хранения данных на 4-х виртуальных машинах была сконфигурирована система хранения данных Ceph (в силу ее масштабируемости).

Модуль преобразования данных

Одним из основных свойств подобного рода систем является независимость от того, как хранятся данные на самом деле (не в контексте датасетов и объектов). Несмотря на то, что в реализованном прототипе системы данные хранятся в объектном (S3) представлении, система может также работать с файловыми интерфейсами (интерфейс обычной файловой системы компьютера и файловый интерфейс системы хранения данных iRODS). Такая возможность существует благодаря наличию модуля преобразования данных, который отображает данные в терминах файлов, директорий и объектов к данным в терминах датасетов и объектов.

REST-интерфейс

Реализованная система на данном этапе не имеет графического интерфейса пользователя, все операции с данными выполняются через встроенный REST-интерфейс. Таким образом, существует возможность выполнять операции с данными и метаданными при помощи HTTP-запросов. Все операции, для которых определен REST-интерфейс, задокументированы с помощью утилиты Swagger UI (web-страница с подробным описанием метода, http://swagger.io/swagger-ui/).

Система авторизации пользователей

Механизм авторизации пользователей реализован при помощи компонента Spring Security (http://projects.spring.io/spring-security/) фреймворка Spring. Таким образом, при выполнении внешних REST-запросов производится проверка наличия у пользователя необходимой роли (администратор данных или системный администратор), а также необходимых прав на выполение операции.

Заключение

Построение конкретной системы управления данными во многом определяется предъявляемыми к ней требованиями. Например, количество ролей пользователей может быть расширено, или могут быть реализованы дополнительные компоненты (графический интерфейс пользователя и т.д.).

В ходе данной работы был реализован прототип системы с тремя основными модулями, использована методология раздельного хранения данных и метаданных. Разработанная система условно называется Data managment platform (Data-as-Service), с ее исходным кодом можно ознакомится на https://github.com/timbog/Data-as-Service (Исходный код модуля преобразования данных находится отдельно, в публичном доступе его нет). Сама платформа для работы с базой данных для метаданных использует библиотеку Hibernate (http://hibernate.org/), позволяющую оперировать сущностями из бд независимо от ее типа, в тестовом окружении используется база данных HSQLDB (http://hsqldb.org/). Для хранения данных была развернута система хранения данных Ceph (http://ceph.com/). В процессе разработки системы была использована система непрерывной интеграции Drone (https://drone.io/). В дальнейшем исследование может быть продолжено в сторону увеличения количества компонентов посредством использования готовых (балансировщик нагрузки, бэкапирование и т.д.) средств, а также разработки новых (модуль управления политиками и правилами).

Разработанная мною система может служить как единое ядро для разработчиков, которым нужна система хранения данных, она может быть доработана или изменена в зависимости от конкретных требований к платформе.

Список литературы

1. Сайт системы хранения данных iRODS [электронный ресурс] – Режим доступа: http://irods.org/, свободный.

2. Сайт платформы ATTIVO [электронный ресурс] – Режим доступа: http://www.attivio.com/platform, свободный.

3. Сайт системы хранения данных Ceph [электронный ресурс] – Режим доступа: http://www.ceph.com/, свободный.

4. Статья с обзором алгоритмов балансировки нагрузки [электронный ресурс] – Режим доступа: https://habrahabr.ru/company/selectel/blog/250201/, свободный.

5. Сайт базы данных Mongo DB[электронный ресурс] – Режим доступа: https://www.mongodb.com/, свободный.

6. Сайт базы данных Arango DB[электронный ресурс] – Режим доступа: https://www.arangodb.com/, свободный.

7. Сайт базы данных HSQLDB[электронный ресурс] – Режим доступа: http://hsqldb.org/ , свободный.