2. LBLRTM. http://rtweb.aer.com/line_param_frame.html

3. RTTOV. http://research.metoffice.gov.uk/research/interproj/nwpsaf/rtm/

4. *Matricardi M.*: The generation of RTTOV regression coefficients for IASI and AIRS using a new profile training set and a new linebyline database, ECMWF Research Dept. Tech. Memo. 564, 2008.

5. *Головин Ю.М., Завелевич Ф.С., Никулин А.Г., Козлов Д.А., Монахов Д.О., Козлов И.А., Болмосов И.В., Архипов С.А., Целиков В.А., Романовский А.С.* Информационные характеристики летного образца аппаратуры ИКФС-2 // Современные проблемы дистанционного зондирования Земли из космоса: сборник статей. 2012. Т.9, № 5. С. 291-300.

6. *Brunnel P., Turner S.* On the use of Planck-weighted transmittances in RTTOV // Poster in the 13h International TOVS study conference, Ste. Adele, Canada, 29 October – 4 November 2003 (http://cimss.ssec.wisc.edu/itwg/itsc/itsc13/thursday/brunel_poster.pdf).

Technology for fast simulation of infrared sounders measurements

Valeriy Pavlovich Pyatkin, Doctor of Technical Sciences, Head of laboratory
Evgeniy Vladimirovich Rusin, Candidate of Technical Sciences, Senior Research Associate
Institute of Computational Mathematics and Mathematical Geophysics RAS

The computational technologies for fast and accurate modeling of infrared spectra to be measured by IRFS-2 and MSU-MR sounders which are planned to be installed on board of Russian meteorological Meteor-M satellites are under consideration. Methodological aspects of fast radiative transfer model development are briefly described. Interface and functionality of the developed software is presented, its accuracy and performance are assessed.

Keywords: computational technologies, remote sensing, software, satellite measurements simulation, high-performance computing, RTTOV.

УДК 004.438

СТРУКТУРА ДАННЫХ ДЛЯ ХРАНЕНИЯ ИНФОРМАЦИИ В СОЦИАЛЬНЫХ СЕТЯХ

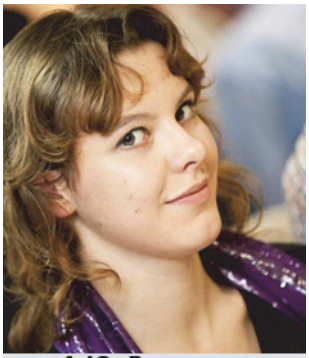
***Вера Львовна Волушкова**, канд.техн.наук, доц.,*
Тел.: +7 (920) 694 13 72, e-mail: w2l@pisem.net
Тверской государственный университет
http://university.tversu.ru
***Александра Юрьевна Волушкова**, студент*
Тел.:+7(926)153-53-30, e-mail: sasha.volushkova.92@mail.ru
Московский авиационный институт
http://www.mai.ru

Предложена структура данных для хранения «постов» в социальных сетях с небольшим количеством пользователей. Перед разработчиками стоит задача выбора удобной и эффективной системы хранения информации. Созданные в работе структуры данных учитывают особенности no-SQL и SQL хранилищ данных. Производительность каждой из предложенных моделей проверяется на запросах разной сложности к базам данных разной структуры. Эффективность предложенных структур проверяется на конкретных данных.

Ключевые слова: базы данных, NoSQL базы данных, MongoDB.

Данные в социальных сетях представляют собой текстовые сообщения различных пользователей, которые вывешиваются(post up) на страничках членов сети или страничках сообществ сети. Поэтому такие сообщения называют «постами». Посты принадлежат страничкам пользователей или сообществам сети. Данные постов не требуют синхронизации. Помимо страничек в социальных сетях, которые мы рассматриваем, существует понятие «новостная лента». «Новостная лента» - это подборка «постов» ав-


В.Л. Волушкова


А.Ю. Волушкова

торов, на которые подписан пользователь, отфильтрованная по дате. Между авторами в сети существуют отношения «подписка» и «друг». Авторы-«друзья» могут комментировать сообщения друга, а авторы-«подписчики» могут лишь просматривать сообщения определённого пользователя сети.

Задача заключается в выборе способа хранения «постов», который обеспечивал бы минимальное время формирования странички пользователя сети или минимальное время формирования «новостной ленты».

Так как структура данных рассматриваемой социальной сети не является сложной, а основная задача заключается в минимизации времени отклика, можно рассмотреть не только традиционный способ хранения данных – реляционная база данных, но и NoSQL базы данных, в которых вопросы, связанные с обеспечением вычислительной мощности системы, успешно решаются разделением задач между узлами системы.

Впервые термин NoSQL был употреблён в 1998 году Карло Строззи (Carlo Strozzi) для обозначения своей базы данных, не поддерживавшей язык запросов SQL. В современном значении термин был выдвинут в 2009-м, когда Йохан Оскарссон (Johan Oskarsson) решил организовать встречу разработчиков для обсуждения распределенных баз данных. Сам акроним NoSQL принято расшифровывать как «Not only SQL». Это должно означать, что сообщество разработчиков NoSQL не позиционирует нереляционные хранилища как универсальное решение и подразумевает использование реляционных моделей хранения данных там, где это имеет смысл. Разработчики NoSQL хранилищ опираются на теорему Брюера.

Теорема Брюера или CAP теорема (акроним от Consistency, Availability, Partition tolerance)[1] заключается в следующем – невозможно создать распределенный веб-сервис, удовлетворяющий сразу трём требованиям: согласованности (consistency), доступности (availability) и устойчивости к разделению на части (partition tolerance).

Согласованность и доступность при отсутствии устойчивости к разделению – к этому классу относятся все традиционные реляционные базы данных.

Согласованность и устойчивость к разделению при отсутствии доступности – такой подход обеспечивает согласованность данных в ущерб доступности (в редких случаях даже в ущерб сохранности последних совершенных транзакций – см. Redis). Такой подход используют продукты BigTable, HBase, MongoDB, MemcacheDB, BerkeleyDB и другие.

Доступность и устойчивость к разделению при отсутствии согласованности – к этому классу относятся системы, направленные на быструю обработку больших объемов данных в условиях, когда появление этих данных на всех узлах системы в определенном порядке не имеет большого значения. Главными представителями этого класса являются потенциально согласованные (eventually consistent) системы, такие как Cassandra, Dynamo, Voldemort, CouchDB, SimpleDB и другие.

В дальнейшем будем рассматривать реляционную БД – Postgres и нереляционное хранилище MongoDB, т.к. для рассматриваемых социальных сетей не требуется разделение на узлы.

MongoDB - это документо-ориентированное нереляционное хранилище данных, использующее для хранения данных формат BSON – расширение формата JSON (JSON – JavaScript Object Notation, BSON – Binary JSON)[2]. Каждый экземпляр сервера MongoDB содержит несколько баз данных. Каждая база данных состоит из коллекций. Коллекция содержит в себе документы. Документы являются наборами полей, представляющих из себя пары ключ-значение. Структура данных лишена схемы. Так, один

документ может иметь разные поля, а поля документа с одинаковым именем не обязательно будут иметь один тип.

Сравним организацию данных в реляционной базе и в MongoDB.

Реляционная модель социальной сети содержит сущности: Users (user and community), UserToUser (для описания связей между пользователями), Post (реализована связь, отражающая иерархическую зависимость между постами – посты в ответ на другие посты). Такая модель позволяет формировать достаточно широкий набор запросов разной сложности.

Для работы с MongoDb используется Orm(*Object-relational mapping*) Google.morphia. ORM - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». В случае использование Google.morphia достаточно создать ряд объектов, описав их с помощью соответствующих классов. В каждом классе есть поля и методы работы с ними, а также обрабатываются возникающие в процессе работы исключительные ситуации согласно модели, приведенной в [3].

Модель в MongoDB представляет собой набор следующих сущностей:

```
@Entity("users")
public class User {
   @Id private ObjectId userId;
   @Property("user_name")  private String name;
   @Embedded  private Wall userWall;
   @Reference(ignoreMissing = true)
   private List<User> friends;
   }

Entity("posts")
@Indexes({
   @Index(name = "wall_id_index", value = "wall_id"),
   @Index(name = "author_id_index", value = "author")} )
public class Post extends AbstractPost {
   @Property("theme")  private String theme;
   @Property("wall_id")  private ObjectId wallId;}

public abstract class AbstractPost {
   @Id  private ObjectId id;
   @Property ("text")  private String text;
   @Property ("author")  private ObjectId author;
   @Property ("parent")  private ObjectId redactor;
   }

public class Wall {
@Property
   private ObjectId id;

   @Property
   private String ownerType;
   }
```

Приведенная модель использует аннотированные объекты и DAO(data access object) подход, предоставляющий абстрактный интерфейс к базе MongoDb. Вся конфигурация задается аннотациями, XML файлы не используются.

В работе была сымитирована социальная сеть, имеющая 100000 user'ов. Каждый user имел в среднем по 50 друзей и 100 подписчиков. В post'ах было сделано в среднем по 70 сообщений на страничку каждого пользователя.

Генерация такой модели в Postgres достаточно длинная процедура. Важно правильно сформировать ключи и индексы во всех трех таблицах. Для таблицы UserToUser создаем индекс по двум полям user1,user2. Для таблицы Post создаем индекс по владельцу «стены» и индекс по автору «поста».

В MongoDB формирование модели занимает несколько минут за счет того, что наборы данных можно писать сначала с оперативную память, а затем скидывать на диск. Для этого использовалась JDK 1.8 и следующие настройки VM -d64 -Xms512m -Xmx4g.

Рассмотрим ряд типичных запросов, необходимых для формирования странички пользователя.

Для поиска всех «постов» пользователя с id = 100054 с учетом иерархии использовался следующий рекурсивный запрос:

```
WITH RECURSIVE temp1 ( "id","parentPost","authorId","subject","content",PATH, LEVEL ) AS (
SELECT T1."id",T1."parentPost",T1."authorId", T1."subject",T1."content", CAST (T1."id" AS VARCHAR (50)) as PATH, 1 FROM "socialNework".post T1 WHERE T1."parentPost" IS NULL and T1."ownerId" = 100054
union select T2."id",T2."parentPost", T2."authorId", T2."subject",T2."content", CAST ( temp1.PATH ||'->'|| T2."id" AS VARCHAR(50)) ,LEVEL + 1 FROM "socialNework".post T2 INNER JOIN temp1 ON( temp1."id"= T2."parentPost") where T2."ownerId" = 100054 )
select * from temp1 ORDER BY PATH LIMIT 100
```

Получился следующий результат

Drag a column header here to group by that column

id	parentPost	authorId	subject	content	path	level
209 360	Null	162677	subject162677	message from 162677	209360	1
209 361	209360	162677	subject162677	message from 162677	209360->209361	2
209 363	209361	124399	subject124399	message from 124399	209360->209361->209363	3
209 366	209363	106990	subject106990	message from 106990	209360->209361->209363->209366	4
209 367	209360	100054	subject100054	message from 100054	209360->209367	2
209 362	Null	187310	subject187310	message from 187310	209362	1
209 364	Null	161748	subject161748	message from 161748	209364	1
209 365	Null	161748	subject161748	message from 161748	209365	1

Grid View | Form View

Records fetched: 8/8

```
8 rows returned (141 ms)
```

Рис. 1. Результат рекурсивного запроса

В MongoDB этот результат получается с использованием запроса: db.test.find({{wall_id :"100054"},{parent: 'branch_root_id'}}); Время выполнения этого запроса в среднем 70мс.

Результаты тестирования отражены в таблице 1.

Таблица 1

Среднее время выполнения запросов

	Время выполнения запроса на формирование странички user'а(mc)	Время выполнения запроса на формирование новостной ленты(mc)	Время выполнения запроса на добавление поста(mc)
postgres	100	80	0.4
mongoDB	70	78	0.1

Для нагрузочного тестирования использовалось приложение Apache JMeter.

План тестирования:

1. Thread group создаёт 100 потоков и выполняет 5 циклов.
2. Созданные потоки соединяются с базой данных с помощью JDBC-драйвер.
3. Потоки делают по одному запросу на формирование странички user'а к postgres и mongoDB.
4. Данные запросов отображаются на графике и в агрегированном отчете.

Таблица 2

Время выполнения запроса под нагрузкой

	Количество пользователей						
	1	2	3	5	10	50	100
Время выполнения, postgres мс	100	100	100	100	100	105	107
Время выполнения mongoDB мс	70	70	70	70	70	83	88
Ошибки, %	0	0	0	0	0	0	0

Качественные показатели по работе с БД отражены в таблице 3.

Таблица 3

Качественное сравнение БД

	Простота запросов	Маштабируемость	Работа под нагрузкой	Целостность данных
postgres	-	+	+	+
mongoDB	+	+	+	-

Результаты эксперимента позволяют сделать вывод о том что, производительность Postrges и MongoDB при формировании странички пользователя и новостной ленты в социальных сетях небольшой размерности примерно одинакова. Поэтому, если у разработчиков сети есть привязка к реляционной базе в силу каких либо причин, например исторических, то не стоит переходить на модные в настоящий момент NoSQL БД. Если такой привязки нет и в команде нет специалистов SQL, то есть смысл воспользоваться MongoDB, так как формирование запросов в такой БД проще и связь с БД осуществляется с использованием непосредственно объектов, т.е. пропадает необходимость в создании слоя DAO.

Авторы считают, что в данной работе новыми являются следующие положения и результаты:

1. Модель структуры данных для хранения «постов» в социальных сетях, реализованная для Postgres и MongoDB. Оценка производительности БД с помощью созданной модели.

2. Программный продукт, который реализует модель БД, и позволяет рассчитать различные параметры Postgres и MongoDB при использовании их в социальных сетях

Литература

1. *Brewer Eric A.* Towards robust distributed systems // Proceedings of the XIX annual ACM symposium on Principles of distributed computing. − Portland, OR: ACM, 2000. Т. 19. № 7.

2. *Кристина Чодороу, Майкл Дирольф* MongoDB: The Definitive Guide. – O'Reilly Media, 2010. – 216 с.

2. *Волушкова А.Ю.* Анализ работы с исключениями в различных языках программирования // Образовательные ресурсы и технологии. 2014. № 1. С. 62-67

Structure of data for information storage in social networks

***Vera Lvovna Volushkova**, Candidate of Technical Sciences, Associate Professor*
Tver state university
***Alexandra Yurevna Volushkova**, student*
Moscow Aviation Institute

The structure of «posts» for storage in social networks with a small amount of users is offered. The developers face the problem of choosing an effective system of information storage. The data structures created in the work consider features of no-SQL and SQL storages of data. Productivity of each models is checked on inquiries of different complexity to databases of different structure. The efficiency of the offered structures is checked on concrete data.

Key words: databases, NoSQL databases, MongoDB.