



POLYTECHNIC OF NAMIBIA

School of Information Technology

Masters of Information Technology Programme

Private Bag 13388, 13 Storch Street, Windhoek, Namibia

Tel: +264-(0)61-207-2888 Fax: +264-(0)61-207-2051

Simon Muchinenyika (200996290)

ADS510S Cassandra Report

1. Introduction

Trends in databases are working towards non-SQL approaches that include column families, document-oriented databases, key-value tuple stores, etc. the main goal of this effort is to reduce the complexity of the relational model (Quenum, 2010), especially for applications where other models are more suitable. In this report, two frameworks of MongoDB and Cassandra are used to support the modeling and design of database systems for a blogging application and a message management system.

Besides reducing the complexity of relational model, Cassandra for example is an open-source that is capable of building a high-performance (Malik, 2010), write-intensive application on data set that is growing quickly. Cassandra has huge-availability advantages and no single point of failure. Cassandra also has the advantage of a more advanced data model, allowing for a single 'row' to contain billions of column/value pairs – enough to fill a machine. Cassandra was designed to run on cheap commodity hardware and handle high write throughput while not sacrificing read efficiency (Lakshman & Malik, 2010).

2. Message Management System

For the for a message management system application, Cassandra was used as the database management system and Thrift as Cassandra's external client-facing API. Details on running Cassandra from Windows can be found on <http://coderjournal.com/2010/03/cassandra-jump-start-for-the-windows-developer/>. Cassandra's main API/RPC/Thrift port is 9160. Thrift is a software library and set of code-generation tools developed at Facebook to expedite development and implementation of efficient and scalable backend services (Slee, Agarwal, & Kwiatkowski, 2010). The message management system application supports users sending messages to each other. Below are other use cases fulfilled by the application:

2.1 Login

The code for the welcome page displayed in Fig 1 is saved under login.php. The user can also create other users by clicking “Create a user”.

Select user or create user to use the content management system[email based]:

Default

Default

test

abc

def

simon

Create A User

Fig 2.1: The landing interface of the content management system

2.2 Compose and send messages

For composing and sending messages, the code is under message.php.

Message Management System For simon

Inbox

compose

Outbox

To: Dr.Jos

Search

Subject:

Compose

Logout

Send

Fig 2.2: The interface for composing and sending messages

2.3 View incoming messages

In order to display the incoming messages together with the sender’s address, the code in incoming.php was written.

Message Management System For simon				
<div><div>Inbox</div><div>Outbox</div><div>Search</div><div>Compose</div><div>Logout</div></div>	Inbox	From	Subject	Date
		abc@123.com	hi	August 30, 2010, 4:04 pm
		test@123.com	hi simon	August 30, 2010, 4:46 pm
				[DELETE]
				[DELETE]

Fig 2.3: Inbox showing incoming messages together with the view of the detailed message thread

2.4 View outgoing messages

For the outgoing messages, outgoing.php was written.

Message Management System For simon			
<div><div>Inbox</div><div>Outbox</div><div>Search</div><div>Compose</div><div>Logout</div></div>	Outbox	To	Subject
		abc@123.com	Cass
			Date
			August 30, 2010, 3:33 pm

Fig 2.4 showing simon’s outbox

2.5 Search

To search for users the code was saved under search.php.



Fig 2.5 depicting the interface when searching for users

3.5 Limitations

One use case of a reply button soon after reading a message could have been included in the application. Other use cases such as insert and delete messages could also have been included. All these functionalities were not included due to limited time that the developer had. However, it is the developer's interest to continue working on the project in order to improve it.

3.6 Dataset for testing

Username	E-mail address	Password
simon	simon@cassandra.com	simon
abc	abc@123.com	abc
def	def@123.com	def
test	test@123.com	test

3.7 Data Model

The column families were defined in the storage-conf.xml file. For the message management system, the presentation of key => column structure is:

```
ContentManagementSystem{
  "user1": {
    "username": {"name": "username", "value": "simon"},
    "emailaddress": {"name": "emailaddress", "value": "simon@cassandra.com" },
    "password":{"name": "password" , "value": "simon"}
  }
  "user2": {
    "username": {"name": "username", "value": "simon"},
    "emailaddress": {"name": "emailaddress", "value": "simon@cassandra.com" },
    "password":{"name": "password" , "value": "simon"}
  }
}
```

Generally the design goals that are considered in Cassandra are:

- High availability
- Eventual consistency (trade-off strong consistency in favour of high availability)

- Incremental scalability
- Optimistic Replication
- Low total cost of ownership
- Minimal administration

References

Lakshman, A., & Malik, P. (2010). *Cassandra - A Decentralized Stuctured Storage System*. ACM.

Malik, O. (2010, March 11). *Why Digg Diggs Cassandra*. Retrieved August 31, 2010, from Bloomberg Businessweek Web site: www.bussinessweek.com

Quenum, J. (2010). *A taste of cassandra and mongodb*. Windhoek: Polytechnic of Namibia.

Slee, M., Agarwal, A., & Kwiatkowski, M. (2010). *Thrift: Scalable Cross-Language Services Implementation*. Palo Alto: Facebook.