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 Mongodb Report

1.0 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, Mongodb for example is not limited to querying by key like SQL approaches. This data transparency enables the engine to perform additional work without having to translate the data into intermediary or a format that it understands (Popescu, 2010). Mongodb allows ad-hoc querying, and relies on indexes defined on the document values to help it achieve reasonable performance when data grows large enough.

In terms of security, Mongodb does not support encryption but authentication using site to site VPN through CISCO hardware firewalls thereby removing extra complexity from the database.

2.0 Blogging application

For a blogging application, moadmin.php was used for the Mongodb database management system. A model-view-control PHP framework of kohana was used as the API. For the mongodb data base backend I used moadmin.php where I created my database and could administer it.

For more information concerning kohana mvc, more details can be found here: http://dev.kohanaframework.org/wiki/kohana2/Kohana101. Db.php is a PHP wrapper used for this project to connect to mongodb and to perform other functions like insertion, deletion, updating, etc. Below is the code for connection to the mongodb that is included in every template:

```
$connection = new Mongo();
$db = $connection->selectDB('name_of_database');
$collection = $db->selectCollection('name_of_collection');
```

2.1 Insert Function

```
static function batchInsert($collection, $array)
{       global $mongo;
        $col = $mongo->selectCollection(MONGODB_NAME, $collection);
        return $col->batchInsert($array);
}
```

2.2 Delete Function

```
static function remove($collection, $criteria, $just_one = false)
{
            global $mongo;
             $col = $mongo->selectCollection(MONGODB_NAME, $collection);
             if (!is_array($criteria))
            {
                        $criteria = array('_id' => self::id($criteria));
             }
            return $col->remove($criteria, $just_one);
 }
```

2.3 Update function

```
static function update($collection, $criteria, $newobj, $upsert = false)
 {
        global $mongo;
        $col = $mongo->selectCollection(MONGODB_NAME, $collection);
        return $col->update($criteria, $newobj, $upsert);
   }
```

2.4 Retrieve Function

For retrieving one only the code is:

```
static function findOne($collection, $id)
{
         global $mongo;
         $col = $mongo->selectCollection(MONGODB_NAME, $collection);
        if (!is_array($id))
         {
                    $id = array('_id' => self::id($id));
```

```
        }
        return $col->findOne($id);
    }
```

In order to retrieve all the code used was:

```
    static function find($collection, $query = array(), $options = array())
    {
        global $mongo;
        $col = $mongo->selectCollection(MONGODB_NAME, $collection);
        $fields = isset($options['fields']) ? $options['fields'] : array();
        $result = $col->find($query, $fields);
        if (isset($options['sort']))
        {
            $result->sort($options['sort']);
        }
        if (isset($options['limit']))
        {
            $result->limit($options['limit']);
        }
        if (isset($options['skip']))
        {
            $result->skip($options['skip']);
        }
        return $result;
  }
```

N.B: For already existing articles with or without comments, the $push function inside the update or insert function.

3.0 Limitation

The GUI interfaces for this project was not added as part of the report mainly because part of application was not running. However, this does not mean the whole application is a waste as the developer was still working on the project. When developing this project , additional PHP drivers for mongodb has to be installed from http://mongodb.org/display/DOCS/PHP+Language+Center .

4.0 Data Model

The collection that will be used for this project together with their objects are indicated below:

Blogs = {$blog\_entry_0$, $blog\_entry_1$, ……., $blog\_entry_n$}

```
Blog_enrty = {
        author = {
                first_name,
                last_name,
                email_address,
                address: = {
                        house_no,
                        street,
                        city,
                        country
                    }
            }

        post = {
                title,
                text,
                date
            }

        tags = [‘tag0’, ‘tag1’, ……., ‘tagn’]

        comments = [ {
                commentator: {
                        user_name,
                        first_name,
                        last_name,
                        address
                        }
                comments: { …….}, ……
              }]
    }
```

Generally with Mongodb, the attributes that are considered when designing a key/value store are:

- The Key to store/retrieve
- The Value for the given key
- An auto expiry of the cached data
- A key scope enabling multiple namespaces

References

Popescu, A. (2010, August 31). *Comparing Document Databases to Key-Value Stores << mySQL*. Retrieved June 3, 2010, from myNoSQL Web site: http://nosql.mypopescu.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.