A Report on Mongo DB

Introduction:

MongoDB is an open source document-oriented database system. It is part of the NoSQL family of database systems. It provides high performance, high availability, and easy scalability.

Features:

- Scalability
- Document based queries
- Map-Reduce
- GridFS
- Indexing
- Geospatial indexing

Data Modelling:

MongoDB server can contain multiple databases. Each database has collections which are analogous to tables in relational databases. Each of these collections contains documents that are analogous to rows in relational databases. Each document contains fields that are relevant and actually resemble the objects of the application. Data in MongoDB is schemaless. The implications of this is

- documents in the same collection may not necessarily have the same set of fields or structure, and
- common fields in a collection's documents may hold different types of data.

Data modeling decisions involve determining how to structure the documents to model the data effectively. The primary decision is whether to embed or to use references.

- Embedding: To de-normalize data, store two related pieces of data in a single document. For relations such as "contains" relationships between entities, oneto-many relationships where the "many" objects always appear with or are viewed in the context of their parent documents.
- Referencing: To normalize data, store references between two documents to indicate a relationship between the data represented in each document.
 Referencing is appropriate in cases such as, representation of more complex many-to-many relationships, modelling of large hierarchical data sets.

GridFS:

GridFS is the file system layer built on top of MongoDB. GridFS is the specification for chunking files. Instead of storing a large file in a single document, it is chunked into smaller files and each of them are stored as separate documents.

GridFS stores these in two collections. One collection stores the chunks and other stores the metadata related to the file. Hence the metadata of the file is tightly coupled with the data file. When a query is made to the GridFS stored file, the driver or client will reassemble the chunks.

The advantages of chunking mechanism are -

- When a part of file is required to be accessed, then the appropriate chunk is loaded into memory
- Range operation queries can be performed easily
- Arbitrary sections of the files can be accessed

Replication:

Replication in mongoDB ensures automated failover, redundancy and backup. It occurs in Replica Set. Replica Set will consists of two or more instances of mongod. One among them is designated as Primary and others are secondary. Clients direct Writes to Primary.

In an application that involves intensive Reads from database, one mongod instance which is primary, holds all the data. Other secondary mongod instances in the Replica Set will replicate the contents of the primary asynchronously and let more clients read from database.

Failover mechanism is achieved when the primary goes offline and other members of the Replica Set can connect to each other, a new primary is elected from the secondary instances based on the pre assigned priority to the secondary instances. If two secondary servers are assigned same priority then the secondary instance whose dataset is most up to date is chosen as the primary instance. When the previously designated primary comes back online, it becomes a secondary instance. Figure below shows the Replication mechanism in MongoDB.

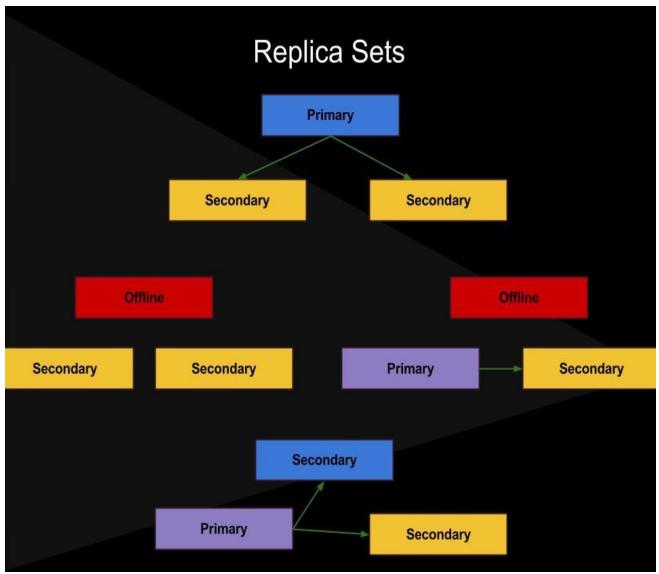


Figure: Represents the Replication process in MongoDB

Sharding:

Sharding is the MongoDB way of realizing scaling. It distributes a single logical database across multiple machines.

Sharding occurs in a sharding cluster. The components are -Shards: A shard holds a subset of the collections data. It is either an instance of mongod(the MongoDB Server) or Replication Set.

Config Servers: It holds metadata of the cluster. It maps metadata to the chunks that the shard holds. These could be single or multiple instances of mongod.

mongos instances: It routes the reads and writes to application transparently and to shards. It does not persist the data by itself.

Consider a database collection that is larger than the existing storage. This single collection is divided into chunks that are provided with a shard key. A process called balancer will distribute the chunks among the shards thus balancing the load. The Config servers will hold the metadata of the shard key and the mapping to the corresponding shard. The monogos will route the reads and writes to appropriate shards by consulting Config servers.

Advantages of Sharding -

- Automatic failover
- Auto balancing of load
- Facilitates additional write capacity

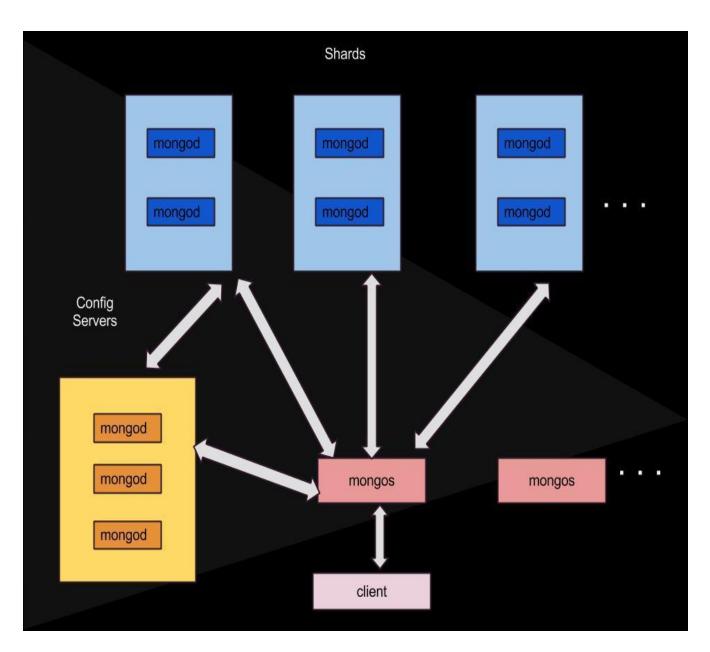


Figure: Sharding cluster

ACID Properties:

- Atomicity: Ensured at single document level.
- Consistency: Eventually consistent reads, from a replica set are only possible with a write concern that permits reads from secondary members.
- Isolation: The multi update/write to multiple documents is not atomic and may interleave with other write operations. The isolation operator isolates the update/write operation and blocks other write operations during update.
- Durability:

- MongoDB uses write ahead logging to an on-disk journal to guarantee write operation, durability and to provide crash resiliency.
- Before applying a change to the data files, MongoDB writes the change operation to the journal.
- there is an up-to 100 millisecond window between journal commits where the write operation is not fully durable.
- Requiring journaled write concern in a replica set only requires a journal commit of the write operation to the primary of the set regardless of the level of replica acknowledged write concern.

Aggregation:

Aggregation Framework provides a means to calculate the aggregated values. Using aggregation, we can add computed fields, create new virtual sub-objects, and extract sub-fields into the top-level of results. Aggregation is provided by following two methods -

- Pipelines: Documents from a collection pass through an aggregation pipeline, the output of one process is provided as input to the next process.
- Expressions: produce output documents based on calculations performed on input documents.

Complex aggregation tasks are handled by a method called Map-Reduce.

Programing in Mongo DB:

Insert, delete and update operations in Mongo DB.

Insert

Primary method to insert a document or documents into a Mongo DB collection

Syntax: db.collection.insert(<document>)

Corresponding Operation in SQL: The insert() method is analogous to the INSERT statement.

Example:

If the collection 'bios' does not exist, then the insert operation will create this collection: db.bios.insert(

{

_id: 1, name: { first: 'John', last: 'Backus' }, birth: new Date('Dec 03, 1924'), death: new Date('Mar 17, 2007'), contribs: ['Fortran', 'ALGOL', 'Backus-Naur Form', 'FP'],

```
awards: [
{
award: 'W.W. McDowell Award',
year: 1967,
by: 'IEEE Computer Society'
},
{
award: 'National Medal of Science',
year: 1975,
by: 'National Science Foundation'
},
{
award: 'Turing Award',
year: 1977,
by: 'ACM'
},
{
award: 'Draper Prize',
year: 1993,
by: 'National Academy of Engineering'
}
]
```

})

To confirm the insert query the bios collection: db.bios.find(). It will return the content of the collection.

```
If the document does not contain an id, we can find doing db.bios.find( { name: { first: 'John', last: 'McCarthy' } } )
```

```
{
    "award" : "Kyoto Prize",
    "year" :1988,
    "by" : "Inamori Foundation"
    },
    {
    "award" : "National Medal of Science",
    "year" : 1990,
    "by" : "National Science Foundation"
    }
  ]
}
```

We can also pass an array of documents to the insert() method, the insert() performs a bulk insert into a collection.

```
Inserting using save()
```

The save() method performs an insert if the document to save does not contain the _id field.

db.bios.save(

{

})

Delete

The remove() method has the following syntax: db.collection.remove(<query>, <justOne>)

Corresponding operation in SQL:

The remove() method is analogous to the DELETE statement, and: the <query> argument corresponds to the WHERE statement, and the <justOne> argument takes a Boolean and has the same effect as LIMIT 1. remove() deletes documents from the collection. If you do not specify a query, remove() removes all documents from a collection, but does not remove the indexes. [1]

For large deletion operations, it may be more efficient to copy the documents that you want to keep to a new collection and then use drop() on the original collection.

Deletes all documents from the bios collection where the subdocument name contains a field first whose value starts with G:

db.bios.remove({ 'name.first' : /^G/ })

The following operation deletes a single document from the bios collection where the turing field equals true: db.bios.remove({ turing: true }, 1)

Delete all documents from the bios collection: db.bios.remove()

Update

Syntax: db.collection.update(<query>, <update>, <options>)

Corresponding operation in SQL The update() method corresponds to the UPDATE operation in SQL, and:

<query> argument corresponds to the WHERE statement, and <update> corresponds to the SET ... statement.

The default behavior of the update() method updates a single document and would correspond to the SQL UPDATE statement with the LIMIT 1.

With the multi option, update() method would correspond to the SQL UPDATE statement without the LIMIT clause.

Modify

Use \$set to update a value of a field.

The following operation queries the bios collection for the first document that has an _id field equal to 1 and sets the value of the field middle, in the subdocument name, to Warner:

```
db.bios.update(
{_id: 1 },
{
    $set: { 'name.middle': 'Warner' },
}
)
```

Add new field

If the <update> argument contains fields not currently in the document, the update() method adds the new fields to the document.

The following operation queries the bios collection for the first document that has an _id field equal to 3 and adds to that document a new mbranch field and a new aka field in the subdocument name:

```
Remove field
```

If the <update> argument contains \$unset operator, the update() method removes the field from the document.

The following operation queries the bios collection for the first document that has an _id field equal to 3 and removes the birth field from the document:

```
db.bios.update(
  { _id: 3 },
  { $unset: { birth: 1 } }
)
```

Upsert flag

The update() operation accepts an "upsert" flag that modifies the behavior of update() from updating existing documents, to inserting data.

These update() operations with the upsert flag eliminate the need to perform an additional operation to check for existence of a record before performing either an update or an insert operation. These update operations have the use <query> argument to determine the write operation:

If the query matches an existing document(s), the operation is an update. If the query matches no document in the collection, the operation is an insert. An upsert operation has the following syntax:

```
db.collection.update( <query>,
<update>,
{ upsert: true } )
```

If no document matches the <query> argument, the upsert performs an insert.

If the <update> argument includes only field and value pairs, the new document contains the fields and values specified in the <update> argument.

If query does not include an _id field, the operation adds the _id field and generates a unique ObjectId for its value.

```
db.bios.update(
{ name: { first: 'Dennis', last: 'Ritchie'} },
{
        name: { first: 'Dennis', last: 'Ritchie'},
        birth: new Date('Sep 09, 1941'),
       death: new Date('Oct 12, 2011'),
       contribs: [ 'UNIX', 'C' ],
       awards: [
       {
       award: 'Turing Award',
       year: 1983,
       by: 'ACM'
       },
       {
       award: 'National Medal of Technology',
       year: 1998,
       by: 'United States'
```

```
},
       {
       award: 'Japan Prize',
       year: 2011,
       by: 'The Japan Prize Foundation'
       }
       1
},
{ upsert: true }
)
```

Insert a document that contains update operator expressions

If no document matches the <query> argument, the update operation inserts a new document. If the <update> argument includes only update operators, the new document contains the fields and values from <query> argument with the operations from the <update> argument applied.

The following operation inserts a new document into the bios collection:

```
db.bios.update(
```

{

{

```
_id: 7,
       name: { first: 'Ken', last: 'Thompson' }
},
       $set: {
       birth: new Date('Feb 04, 1943'),
       contribs: [ 'UNIX', 'C', 'B', 'UTF-8' ],
       awards: [
              {
              award: 'Turing Award',
              year: 1983,
              by: 'ACM'
              },
              {
              award: 'IEEE Richard W. Hamming Medal',
              year: 1990,
              by: 'IEEE'
              },
              {
              award: 'National Medal of Technology',
              year: 1998,
              by: 'United States'
              },
```

```
{
    award: 'Tsutomu Kanai Award',
    year: 1999,
    by: 'IEEE'
    },
    {
        award: 'Japan Prize',
        year: 2011,
        by: 'The Japan Prize Foundation'
        }
      ]
    },
    {
        upsert: true }
)
```

Update operations with save()

The save() method is identical to an update operation with the upsert flag performs an upsert if the document to save contains the _id field. To determine whether to perform an insert or an update, save() method queries documents on the _id field.

The following operation performs an upsert that inserts a document into the bios collection since no documents in the collection contains an _id field with the value 10:

```
db.bios.save(
{
    __id: 10,
    name: { first: 'Yukihiro', aka: 'Matz', last: 'Matsumoto'},
    birth: new Date('Apr 14, 1965'),
    contribs: [ 'Ruby' ],
    awards: [
    {
        awards: [
        {
            awards: [
            {
            award: 'Award for the Advancement of Free Software',
            year: '2011',
            by: 'Free Software Foundation'
        }
      ]
}
```

import com.mongodb.*;

import java.util.ArrayList;

public class MongoExample {

// in the URI. Ex: "mongodb://username:password@localhost:27017/mongoquest" private static String uriString = "mongodb://localhost:27017/mongoquest";

public static void main(String[] args){

// We opt to use the MongoURI class to access MongoDB connection methods. MongoURI uri = **new** MongoURI(uriString); DB database = null;

DBCollection locations = null;

try {

// The MongoURI class can connect and return a database given the URI above. database = uri.connectDB();

// If running in auth mode and have provided user info in your URI, you can use this line.

// database.authenticate(uri.getUsername(), uri.getPassword());

} catch(UnknownHostException uhe) { System.out.println("UnknownHostException: " + uhe);}

catch(MongoException me) { System.out.println("MongoException: " + me);}

if (database != null) {

locations = database.getCollection("locations"); // Retrieve the collection we'll be working with.

// In this example, we build BasicDBObjects describing two locations, Arganis and

Kent.

- // {'name': 'Arganis',
- // 'weather': 'temperate',
- // 'terrain': ['forests', 'plains'],
- // 'benefits': ['lodging', 'trade', 'justice'],
- // 'dangers': ['bandits', 'rebels', 'goblins', 'ghosts']}
- BasicDBObject arganis = **new** BasicDBObject(); ArrayList<String> arganisTerrain = new ArrayList<String>(); ArrayList<String> arganisBenefits = new ArrayList<String>(); ArrayList<String> arganisDangers = **new** ArrayList<String>(); arganis.put("name", "Arganis"); arganis.put("weather", "temperate"); arganisTerrain.add("forests"); arganisTerrain.add("plains"); arganis.put("terrain", arganisTerrain);

arganisBenefits.add("lodging"); arganisBenefits.add("trade");

arganisBenefits.add("justice");

arganis.put("benefits", arganisBenefits);

```
arganisDangers.add("bandits"); arganisDangers.add("rebels");
arganisDangers.add("ghosts");
```

arganis.put("dangers", arganisDangers);

- // {'name': 'Kent',
- // 'weather': 'temperate',
 // 'terrain': ['hills', 'plains'],
- // 'benefits': ['lodging', 'trade']
- // 'dangers': ['bandits', 'rebels', 'famine', 'goblins']}

BasicDBObject kent = **new** BasicDBObject();

```
ArrayList<String> kentTerrain = new ArrayList<String>();
ArrayList<String> kentBenefits = new ArrayList<String>();
ArrayList<String> kentDangers = new ArrayList<String>();
kent.put("name", "Kent"); kent.put("weather", "temperate");
kentTerrain.add("hills");kentTerrain.add("plains");
kent.put("terrain", kentTerrain);
kentBenefits.add("lodging"); kentBenefits.add("trade");
kent.put("benefits", kentBenefits);
kentDangers.add("bandits"); kentDangers.add("rebels");
kentDangers.add("famine");kentDangers.add("goblins");
kent.put("dangers", kentDangers);
// Pass the BasicDBObjects to the .insert() function in our collection object.
locations.insert(arganis);
locations.insert(kent);
```

new BasicDBObject("leader",

"King Argan III")));

// Query for locations with forests. System.out.println("Total number of locations " + locations.count()); // Assign the results of a find operation to a DBCursor object. // Cursors can be iterated through using familiar next/hasNext logic. DBCursor results = locations.find(new BasicDBObject("terrain", "forests")); while(results.hasNext()){ DBObject result = results.next(); System.out.println((String) result.get("name") + " has forests."); System.out.println("Leader (optional) " + (String) result.get("leader")); } // Clean up after ourselves. locations.drop(); }}}

PHP example

```
<?php
$m = new Mongo('mongodb://localhost:27017');
$db = $m->mongoguest;
/*First we get our desired collection.*/
$collection = $db->Spells;
/*$collection = $db->createCollection("Collection Name", true, 10*1024, 10);*/
/* We insert by first creating an array, and passing that array to the collection's insert
function.
We use arrays to construct JSON-like objects.*/
$obj = array('name' => 'Poke', 'level' => 1);
$collection->insert($obj);
$obj2 = array('name' => 'Zap', 'level' => 1);
$collection->insert($obj2);
$obj3 = array('name' => 'Blast', 'level' => 2);
$collection->insert($obj3);
/* At level 1, we only know level 1 spells.*/
echo 'Level 1 spell list: <br/>';
query = array(|evel' = > 1);
$cursor = $collection->find($query);
foreach($cursor as $obj) {
 echo 'Spell name: ' .$obj['name'] .'<br/>';
}
```

```
/*We can use array syntax in-line to create JSON-like queries.*/
$collection->update(array('name' => 'Poke'), array('$set' => array('flavor' => 'Snick
snick!')));
$collection->update(array('name' => 'Zap'), array('$set' => array('flavor' => 'Bzazt!')));
$collection->update(array('name' => 'Blast'), array('$set' => array('flavor' =>
'FWOOM!')));
```

```
/*query again with flavor!*/
echo '<br/>Level 1 spell list, with flavor:<br/>';
$query2 = array('level' => 1);
$cursor2 = $collection->find($query2);
foreach($cursor2 as $obj2) {
    echo 'Spell name: '.$obj2['name'];
    echo ' Flavortext: '.$obj2['flavor'] .'<br/>';
}
```

```
/*clean up after ourselves.*/
$collection->drop();
```

?>

Use Cases:

MongoDB, with its flexible schema, distributed deployment, aggregation and low latency is typically suited for the following kind of applications:

- Content Management
- Inventory Management
- Game Development
- Social Media Storage
- Database for sensor streams

A Few MongoDB Users

MongoDB is widely used by many enterprises. To name a few:

- Craigslist
- Disney
- MTV
- EA Sports

Our Application - Forum Archives

We used MongoDB to create an archive for a collection of massive forums. We downloaded the entire content of seven different forums, each with about 16,000 threads and each thread containing about 100 messages on an average. The detailed statistics of the forums we archived is given below:

Number of forums: 7 Average size of each forum 280 MB Average of threads in each forum: 25716 Average number of message in each thread: 110

Rationale for choosing MongoDB

- The reason we chose MongoDB for our application is because of the nature of content stored in the forums. The structure of a forum is not always constant. A few posts can contain additional information like Images or Videos. A traditional relational database that operates on a fixed schema is not suitable for storing such varied content. Hence, MongoDB with its schema-less architecture suited our purposes.
- The other factor was the absence of complex processing or transactions. Since concurrency was not our priority, MongoDB, with its incomplete implementation of the ACID properties did not pose any restrictions.

• Since we are dealing with a massive amount of data, MongoDB, with its low latency helped us efficiently implement in-memory operations like caching and pagination.

Schema Design

We used three collections – users, forums and threads. The "users" collection stored the login information for the users. The "forums" collection is used to store the general details of the forum statistics and the "threads" collection stores all the threads from all the forums with a reference to the forum id. The schema we used is as follows:

- Users
 - \circ _id
 - Password
- Forum
 - ∘ _id
 - o title
 - o website
 - o description
 - o startdate
 - o endDate
 - o numOfMembers
 - o numOfThreads
 - o numOfMsgs
- Threads
 - ∘ _id
 - o forum
 - o title
 - numOfMsgs
 - o tags
 - o [Comments]
 - _id
 - author
 - date
 - post

A few screen shots:

TCV Forum Archives ×	-	
← → C 🗋 tcv.cs.gsu.edu/php/index.php		९. ८२ 🖃
🔠 IEEE 🏟 ACM Digital Library 🗋 How to dual-boot	U 🗋 date	C Other bookmarks
	TCV FORUM ARCHIVES Collection of Extremist Forums	
	USERNAME	
	PASSWORD	
	LOG IN RESET	E
		*

Figure 1 Login Screen

CV Forum Archives - D₂ ×
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C

TCV FORUM ARCHIVES

GAWAHER (GAMAHER)		ISLAMIC AWAKENING (ISLAMICAWAKENING)		TURN TO ISLAM (TURNTOISLAM)	
NUMBER OF MESSAGES:	372499	NUMBER OF MESSAGES	201287	NUMBER OF MESSAGES	335338
NUMBER OF THREADS:	53225	NUMBER OF THREADS:	32879	NUMBER OF THREADS:	41654
NUMBER OF MEMBERS	9269	NUMBER OF MEMBERS:	3964	NUMBER OF MEMBERS	10858
TART DATE	24-Oct-2004	START DATE	28-Apr-2004	START DATE	02-jun-2006
END DATE	07-jun-2012	END DATE	22-May-2012	END DATE	20-May-2012
FORUM URL	http://www.gawaher.com/	FORUM URL	http://forums.islamicawakening.com/	FORUM URL	http://www.turntoislam.com/forum/
ANSAR ALIMAD NETW	ORX (ANSAR1)	TSLAMIC NETWOR	R (ISLAMECHETWORK)		(ontanventalist militant groups.
				ISLAMIC WEB-CO	OWMUNELA (MAIMC)
WMBER OF MESSAGES	29492	NUMBER OF MESSAGES	91974	ISLAMIC WEB-CO	сания (ауума) 2016
KUMBER OF MESSAGES KUMBER OF THREADS				ISLAMIC WEB-CO	DMMUNITY (MYRWC)
WMEER OF MESSAGES NUMEER OF THREADS NUMEER OF MEMBERS START DATE	29492 11244	NUMBER OF MESSAGES. NUMBER OF THREADS:	91874 13995 2082	ISLAMIC WEB CO NUMBER OF MESSAGES NUMBER OF THREADS	амиллиту (музик;) 25016 5310
WIMBER OF MESSAGES NUMBER OF THREADS NUMBER OF MEMBERS START SATE END DATE	29492 11244 382	NUMBER OF MESSAGES NUMBER OF THREADS NUMBER OF MEMBERS	91274 13995	ISLAMIC WEB-CO NUMBER OF MESSAGES NUMBER OF THEADS NUMBER OF MEMBERS	онкуму утакима (Эккум) 3005 265 265
ANSAR ALIMAD NEW WUNDER OF NESSAGES NUMBER OF NERBAS STATE SATE FORUM URL	25482 11244 382 03-0ec-2008	NUMBER OF MESSAGES NUMBER OF THREADS NUMBER OF MEMBERS START DATE	91874 13995 2083 091jun-2004	ISLAMIC WEB-CO MUMBER OF MISSAGES NUMBER OF MEMORY NUMBER OF MEMORYS START DATE	омина) 25016 6210 756 6300 756

NUMBER OF MESSAGES	1481957
NUMBER OF THREADS:	91527
NUMBER OF MEMBERS:	21013
START DATE	01-Apr-2002
END DATE	18-May-2012
FORUM URL	http://www.ummah.com/forum/

Figure 2 Forum Dashboard Screen

TCV Forum A	Archives - Thi x	- 6 <mark>- × -</mark>
← → C	C tox.cs.gsu.edu/php/hhreadBoard.php?id=1	ବ୍ 😭 ≓
🛄 IEEE 🗔 ACM	M Digital Library 🗋 How to dual-boot U 🗅 date	C Other bookmarks
	TCV FORUM ARCHIVES	
FORUMS	i → GAWATER (GWATER)	
THREAD IE		NUMBER OF MESSAGES
1	Hd Signt Of The Hour Deexonine	1
2	From Behind The Curtain	2
3	Syria, Alah Has Purchases Your Lives For Parados	1
4	Mfa World Cup 2014 Qualification	8
5	War in The Holy Books	1
6	Peace Be Upon You	4
7	The Best Of Muslims Are The Ones With The Best Character	3
8	Helo	15
9	Handi Fujh Of Fasting And Zallati: Prepare For Ramadan	1
10	Exposing One's Own Sits	2
- 11	Where Can I Find Inexpensive, Even Used, Abayes?	6
14	A News Story About Vitchcraft From The Kitat	1
15	Tearing Pages Off The Word Of God (Alah) By An Infidei	16
16	Quran + Arrogent Athest Fooleny Challenging Alant' What	1
18	A Story For You All	1
First P	Provious 1 2 3 4 5 6 7 Next Last Go Page 1 of 3549	

Figure 3 Forum Threads Screen

Q ☆ ≡

CV Forum Arc		and the state of the second seco	
	tcv.cs.gsu.edu/php/messageBoard.php?id=8		ବ୍ ର୍ଯ
EE 🚯 ACM D	Digital Library 🗋 How to dual-boot U 🗋 date		Ci Other book
		TCV FORUM ARCHIVES	
		Collection of Extremist Forums	
FORUMS	>> GAWAHER (GAWAHER) >> HELLO!		
POSTID		POST	
	Author/Skierunner		
	Date:05-Jue-2012		
		us about converting to Barm. This is very scary for me because I grew up in a Christian community and have been agruasic for most of my He– God was evide as that has a nearly non-existence Muslim population, which means I am nearly completely on my own, fortunately, there is the internet. I have great hopes for	
	Author/Saraces/21stC		
2			
	Welcome to the forum Skierunner. I am glad that you have decided to join this forum. We will try our lev	wei best to present Islam in true form to you. Here is an informative thread for new members http://www.gawaher.cbers-read-this/ Here is a section if you w	want to ask any question. http://www.gawaher.cnon-muskm-qa/
	Author:Silerunner		
3	Date:05-Jun-2012		
	Thank you very mucht		
4			
	Helo and welcome from someone who recently joined here. I also have been searching and learning abs	bout Islam and the people here are very helpful. And believe me, I know how it can be difficult as the community I ive in is all fundamentalist Christian churches	s but God (Allah) is very good at guiding us to where He wants us.
	Author: ParadiseLoct		
	Date:05-Jun-2012		
5	welcome to the forum Angle. Great to hear you are interested in Islam - I know it is a scary thought thin	inking to become Muslim especially if you don't live in an area with other Muslims. I became a Muslim over 2 years ago and I was pretty much on my own for th	the beginning and it is difficult but what makes it easier is knowing that choosing the way with
		rophets of God peace be upon them - at times in their life they felt alone - imagine they had to tell people that they were a prophet of God yet many people did we have then we certainly can get through it too with faith in God Weil if you have any questions don't hesitate to ask.	dn't believe them and thought they were insane with even their family members turning
	Author Sternmer	e new one we versamy, van ges un vogin a voorman maan in daar neen goornane any gestatens son cinstance to anti	
6		we certainly can get through it too with faith in God!' That does make it seem a lot better. It's been done before, so I can achieve it as well Thank you all for th	
	* If the prophets of God made it through these hardships with extra responsibilities than we have then v been stated before asking questions, though.	we certainly can get through it too with faith in God. That does make it seem a lot better, it's been done before, so I can achieve it as well thank you all for th	the support, I tigure I's just lurk on the forums for a while collecting knowledge that's aready
	AuthoriPadre5		
7	Date:05-jun-2012		
		ed a lot from them! I have come to see Slaim as a source for both much that is very good, and, unfortunately, much that is very evil, too. I Suspect that the Mi rrolism that appall me. However, I'm happy that you have found the good part, so far! Good luck and peace on your journey to new famil Edited by Padre5, 05	
	Author/ParadiseLoss		
8			
		Figure 4 Forum Messages Screen	