An Open Source NoSQL solution for Internet Access Logs Analysis

A practical case of why, what and how to use a NoSQL Database Management System instead of a relational one

José Manuel Ciges Regueiro <jmanuel@ciges.net> *- Student of the V Master on Free Software Projects Development and Management 2011-2012*

The author's personal web page is http://www.ciges.net and for contacting him preferred method is by email at jmanuel@ciges.net.

I am also available at social networks like Facebook[1], Google+[2], Twitter[3] or Identi.ca[4]

1 http://www.facebook.com/ciges

2 https://plus.google.com/105050850707469524247/posts

3 https://twitter.com/ciges

4 http://identi.ca/ciges

Index

Foreword

This work makes part of the fifth edition of the “Master on Free Software Projects Development and Management”[5], created by the galician open source consultancy Igalia[6] and the “Universidad Rey Juan Carlos”[7] university at Madrid. This master is composed from live sessions imparted by professionals from different specialized open source solutions enterprises and university researchers, practical works and a final project (called “practicum”) which could be made in an enterprise.

As I am working in an open source department at “PSA Peugeot Citroën”[8] (as a worker for the IT consulting Seresco[9]) I thought it could be a good idea to apply part of the knowledge acquired in the master to a project that is interesting for PSA. Our department manages open source server solutions over Unix servers (mainly but not only Linux) and at that time (early 2012) was considering making a study of NoSQL and how this class of database management systems could be useful for enterprise needs.

So what it's presented next is the result of studying a few open source NoSQL solutions, choosing and deploying one of them in PSA's servers and making a comparative between the database management system used at this moment for Internet access log management and the chosen one, MongoDB.

All of this would not have been possible, at least the way it has been done, without a lot of people, but specially:

- Igalia's and Libresoft's[10] people who had been made possible that in a distant city located in the northwest corner of Spain eight students could enjoy the company of open source experts
- Seresco and PSA Peugeot Citroën's EAMP department, who has given all the collaboration possible to make compatible the requirements of the Master with everyday work
- My wife and my young daughter, who have had a lot of patience with “that little guy behind his computer” they are living with
- My father, who always has been there when I needed to delegate my “father functions”

“Be Free! Be Wild! Be Open!”

José M. Ciges

5 http://www.mastersoftwarelibre.com/

6 http://www.igalia.com/

7 http://www.urjc.es/

8 http://www.psa-peugeot-citroen.com

9 http://www.seresco.es/

10 Libresoft is the libre software and open communities research group from “Universidad Rey Juan Carlos” http://libresoft.es/

Notation used for the references and bibliography

Most of affirmations made in this document are supported by Internet references (blogs of experts, webs of enterprises which made the technologies cited), books or published papers.

I have used the following criteria to include the references

- Studies shown on blog's posts, books, papers or official documentation from supporting enterprises are shown as part of the bibliography (at the end of the document). The reference to the bibliography is made with a number between [and]
- URLs to products official web pages and links to Internet cited are shown as a little number, which leads to a note on the footer of the same page. As a footnote is shown what normally would be a link (in a digital document). I have preferred this format to avoid loosing information in case this work is printed on paper (please don't kill trees only for a question of comfort)

In the following example we can see utilisation of both:

> Apache Cassandra[11] is an open source distributed NoSQL database management system. It is an Apache Software Foundation top-level project*[1]*.

11 http://cassandra.apache.org/

Introduction and description

1. Author's description

I am a Spanish systems engineer born in 1976 who has been working last 9 years in PSA Peugeot Citroën (from now on PSA) EAMP's department hired by Seresco (an IT Consulting enterprise).

This department gives support for some Open Source servers on Unix machines (mainly Linux) for the needs of any PSA's worker worldwide. The products we work on a daily basis are Apache web server, MySQL, PHP, Tomcat, FreeRadius, MediaWiki and Typo3.

I discovered Linux at University. At that time I knew nothing about Free Software, I installed Linux just because somebody told me with that I could make the Unix exercises at home. Later, when Internet become part of my life, I become interested in everything around Linux and now I am a “GNU/Linux & Free Software fan boy”.

Apart from that I am also a father, husband, and in my free time I try to make some sport, pay attention to what happens around and learn everyday something new :-)

My personal data are

- Name: José Manuel Ciges Regueiro
- Born date: 25 February 1976
- Education Title: “Technical Engineering in Computer Systems” at University of La Coruña (Spain)
- City in which I live: Vigo

2. Company's description

As I said I work for Seresco[12], an IT Consulting Spanish company born in 1969 with around 500 employees in the country. Seresco's main activities are software development, technical assistance, consultancy and is specialized in the areas of human resources and geographical information. Clients of Seresco are other enterprises and Spanish regional governments.

In Galicia one of this clients is “PSA Peugeot Citroën”[13], as this multinational automobiles and motorcycles manufacturer has a factory at Vigo.

PSA is the second largest European automaker and the eighth largest in the world measured by 2010 unit production. With its 32 manufacturing facilities PSA employees 186.000 and makes around 3,5 millions of vehicles per year.

From an IT Systems point of view from Vigo's factory we give service to every technical team to install, configure and update some Open Source products as Apache web server, MySQL, PHP, Tomcat, FreeRadius, MediaWiki and Typo3.

The IT Systems at PSA employees 2.600 people in 26 different countries. A rough summary of the facilities could be:

- Servers: 6.500 instances of Unix, 3.200 instances of Windows servers, and also a few tens of Mainframe z/OS, VMS and Tandem / Guardian
- Office equipment: 87.000 client computers, most of them (over 70.000) windows

12 http://www.seresco.es

13 http://www.psa-peugeot-citroen.com/

The project's tutor from Seresco will be Andrés Riveiro Sestelo, the director of company's Galician area. At PSA's side this work will be verified by David Fernández Fernández, the head of the department.

3. Project's objectives

The term “NoSQL” is fairly popular in last years to designate a class of database management systems (DBMS from now) where the relations between groups of data are not really important. NoSQL database systems rose alongside major internet companies, such as Google, Amazon, Twitter, and Facebook[14] which had significantly different challenges that the traditional RDBMS solutions could not cope with.

The main raison to work with this DBMS is to manage **huge amounts of data** where we need to generate reports and make calculations but due to the nature of the problem the traditional data schema where the data is stored in records with the identical structure grouped in tables and related between them is not useful. We could think in problems as statistical data analyse, logs information, document or web page indexing, geographical information management ….

When we say big amounts of data we mean

- **the data must be distributed** between an undetermined number of computers and the architecture must be fault tolerant
- the performance of retrieving and appending operations must be very high, as the **read & write performance are critical**
- **operations over the data must be distributed also**, the system must be able to coordinate different nodes and the results got by them to get the final results

4. Realisation conditions

All the project has been realized on PSA's IT installations, as part of my work in the office or at home using a network connection via VPN.

The work has begun the 5 of March with an estimated completion date of late July and an estimated work charge of 300 hours. My time at PSA will not be fully dedicated to this NoSQL study, as there are other work tasks that require my time. So the number of hours per day will be variable between 0 and 8.

The hardware used for the tests will be two identical machines with the following specifications:

- RAM: 10 GB
- Cores: 32 (AMD Opteron Processor 6128)
- Other info: this machines are virtual machines hosted on a physical Dell™ PowerEdge™ R815[15] server

The initial study and information searching will be done reading articles and comparatives from the Internet. For the development and tests phase almost all the technologies used will be server-side. In particular:

Server-side software technologies used

- Operating System: **Suse Linux Enterprise Server 10** (Virtual Server on Xen)

14 Cassandra, now maintained by Apache was initially a Facebook's project. See “Cassandra – A structured storage system on a P2P Network” at http://www.facebook.com/note.php?note_id=24413138919&id=9445547199&index=9

15 http://www.dell.com/us/enterprise/p/poweredge-r815/pd

- The servers to compare will be **MongoDB 2.2.0**[16] and **MySQL**[17] **5.0.26** with MyISAM tables
- The development of scripts will be done on **PHP**, **JavaScript** and **SQL**. The code of the PHP classes will be documented using **phpDocumentor**[18].
- The code editor will be **VIM**

From the client-side I will use

- Automation tasks and single user load tests will be done using **Shell scripts**
- **Notepad++** as code editor
- **JMeter**[19] for making multiuser load tests, using the plugin "Stepping Thread Group"[20] for some of them
- The results from JMeter will be represented graphically using the statistical software **R**[21]
- **LibreOffice** for writing the documentation
- **ArgoUML**[22] for showing a diagram with the architecture of PHP classes

I have made this work as part of the "Master on Free Software Projects Development and Management" and I think **it's important and coherent with its spirit that, when possible, the documentation created, methodology used and conclusions reached should be of public availability** (and all the tools used should be Open Source). The only limit to this consideration is the internal data of PSA and those tools obliged by the work place conditions (as Windows for the workstation :-).

Internal data (as configurations or custom installation directories) has been carefully replaced in the documentation by fake data (not real, but valid anyway).

The tools used to make this had been the following:

- A summary of the information told in this document is available as **posts at my personal web page**[23] (in Spanish). This web has been done with **TextPattern**[24], a PHP Content Management System. This document and a presentation made are also available.
- The code developed in PHP is available in **Github** at "Ciges/internet_access_control_demo"[25]. The HTML documentation created with phpDocumentor is at http://www.ciges.net/phpdoc/iacd/InternetAccessLog/
- The presentation has been uploaded to **SlideShare**[26]

16 2.2.0 is the production release from August 2012 http://www.mongodb.org

17 MyISAM is the light engine for web-based http://www.mysql.com/

18 http://www.phpdoc.org/

19 http://jmeter.apache.org/

20 http://code.google.com/p/jmeter-plugins/wiki/SteppingThreadGroup

21 http://www.r-project.org/

22 http://argouml.tigris.org/

23 http://www.ciges.net

24 Textpattern is a PHP Based Content Management System http://textpattern.com/

25 https://github.com/Ciges/internet_access_control_demo

26 http://www.slideshare.net/Ciges/no-sql-projectmswlpresentation

5. Brief description of the work plan

The work plan is composed of four main parts:

NoSQL State of the Question

As at this moment we know really nothing of NoSQL technologies first we will have to become familiar with this kind of database management systems.

So this part is a **summary of what I have learned reading articles and books**, and the first general conclusions reached.

We are studying to apply NoSQL technologies for log information management, so in this part **we will describe one of the use cases, the management of the access log from the enterprise network to Internet**. From this use case we will choose a NoSQL solution between the products available in the Open Source world.

Installation of chosen solution and database design for an Internet access control software

This part will be more technical. It will have

- An explanation of some details of **how the product has been installed**, its configuration and scripts developed if we have the need to develop some.
- **A database schema design for NSQL** based on MySQL actual solution. The structure chosen and the information fields will be shown

Comparative of a MySQL based solution versus new NoSQL one

With a very similar database on MySQL and on the new NoSQL DBMS[27] I will try to do performance tests on both using generated random data (I can not use real data because of confidentiality).

To make this tests valid the volume of data should be high. As we can not use real data some scripts will be developed to make it possible:

- **Generate fake random data** similar to real one (URLs, user ids, timestamps, IPs …)
- Create tables on the database and save all this data in a very similar structure regardless of the DBMS used
- **The classes developed will have the same interface, so the DBMS chosen will be transparent** to to the applications that use them

Then, using this classes a list of write and read tests will be developed to compare both solutions. A description of each test and how to play them will be made in this part

Conclusions and last words

Well, then objective of this study is to learn about NoSQL and to have concrete data to decide if it's a good idea to use NoSQL for some of PSA's needs.

Here the results obtained, its limitations if there are any and future work to do will be detailed.

27 Database Management System

NoSQL State of the Question

1. What are we talking about

As I said in the previous chapter NoSQL is a class of database management systems that differ from the classic model of the relational database management systems:

- **They do not use SQL**

  NoSQL database systems rose alongside major internet companies, such as Google, Amazon, Twitter, and Facebook which had significantly different challenges in dealing with huge quantities of data that the traditional RDBMS solutions could not cope with.

  The kind of problems this databases are developed for are the management of really big amounts of data that do **not follow necessarily a fixed schema**. The data is partitioned between different machines (for performance questions and due its size) so **JOIN operations and are not usable** and ACID guarantees are not given. DBMS and SQL are not valid tools

- **May not give full ACID guarantees**

  Usually **only eventual consistency is guaranteed** or transactions limited to single data items. This means that given a sufficiently long period of time over which no changes are sent, all updates can be expected to propagate eventually through the system.

- Usually they have a **distributed architecture** and are **fault tolerant**

  Several NoSQL systems employ a distributed architecture, with the data held in a redundant manner on several servers. In this way, the system can easily scale out by adding more servers, and failure of a server can be tolerated.

  This type of databases typically **scale horizontally** and are used for managing with **big amounts of data**, when the performance and real-time nature is more important than consistency (as indexing a large number of documents, serving pages on high-traffic websites, and delivering streaming media).

NoSQL Databases are often highly optimized for retrieve and append operations and often offer little functionality beyond record storage (e.g. key-value stores). The reduced run time flexibility compared to full SQL systems is compensated by significant gains in scalability and performance for certain data models.

In short, NoSQL database management systems are useful when we work with a huge quantity of data, and the data's nature does not require a relational model for the data structure. The data could be structured, but it is minimal and what matters is the ability of storing and retrieving great quantities of data, and not the relations between the elements.

By example, we want to store millions of pairs key-value in one or a few associative arrays or we want to store million of data records. This is particularly useful for statistical or real-time analyses for growing list of elements (think in posts at Twitter or the logs of access to Internet from a big group of users).

NoSQL databases are categorized according to the way they store the data. The main categories we could consider are

- “Document-oriented databases”

- “Key-value databases”
- “Graph databases”
- “Tabular databases”, also called “Columnar databases”

2. Document-oriented databases

A document-oriented database stores, retrieves, and manages semi structured data. **The element of data is called “document”**.

Different implementations offer different ways of organizing and/or grouping documents:

- Collections
- Tags
- Non-visible Metadata
- Directory hierarchies

Compared to relational databases we could say collections are as tables, and documents are as records. But there is one big difference: every record in a table have the same number of fields, **while documents in a collection could have completely different fields**.

Encodings in use include XML, YAML, JSON, and BSON, as well as binary forms like PDF and Microsoft Office documents (MS Word, Excel, and so on).

Documents are addressed in the database via a unique **key** that represents that document. One of the other defining characteristics of a document-oriented database is that, beyond the simple key-document (or key-value) lookup that you can use to retrieve a document, the database will offer an **API or query language that will allow you to retrieve documents based on their contents**.

Data example

For example, MongoDB uses a binary form JSON to store data. An example MongoDB collection could be described as

```
{
"_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
"Last Name": "DUMON",
"First Name": "Jean",
"Age": 43
},

{
"_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
"Last Name":  "PELLERIN",
"First Name": "Franck",
"Age": 29,
"Address": "1 chemin des Loges",
"City": "VERSAILLES"
}
```

Some Open Source solutions

Example of Open Source document-oriented NoSQL databases that we will study are:

- **MongoDB:** A 10gen[28] project which store structure data as JSON-like documents with dynamic schemas (MongoDB calls the format BSON). MongoDB provides dynamic queries, indexes, geospatial indexes and master-slave replication with auto-failover. **MongoDB is being developed as a business with commercial support available.**

  Best used: If you need dynamic queries. If you prefer to define indexes, not map/reduce functions. If you need good performance on a big DB. If you wanted CouchDB, but your data changes too much, filling up disks. *[2]*

  If your DB is 3NF and you don't do any joins (you're just selecting a bunch of tables and putting all the objects together, AKA what most people do in a webapp), MongoDB would probably kick ass for you. *[3]*

  For example *[2]*: For most things that you would do with MySQL or PostgreSQL, but having predefined columns really holds you back.

- **CouchDB:** An Apache Software Foundation project which uses JSON over REST/HTTP. CouchDB provides **master-master replication** and **versioning**.

  Best used: For accumulating, occasionally changing data, on which pre-defined queries are to be run. Places where versioning is important. *[2]*

  For example: CRM, CMS systems. Master-master replication is an especially interesting feature, allowing easy multi-site deployments. *[2]*

3. Key-value databases

Data is stored as pairs key-value, in a schema-less way. A value could be of any data type or object.

Example of Open Source key-value NoSQL databases that we will study are:

- **Cassandra:**[29] It is an **Apache Software Foundation top-level project** *[1]*, initially developed by Facebook. It is distributed and designed for using commodity servers *[4]*. It is possible to use Map Reduce with Apache Hadoop.

  Best used: When you write more than you read (logging). *[2]*

  For example: Banking, financial industry. Writes are faster than reads, so one natural niche is real time data analysis. *[2]*

> Apache Hadoop is a software framework that supports data-intensive distributed applications which is becoming a standard for data storing and analyse.

- **Membase**:[30] **Memcache like compatible database** (it uses Memcache protocol) but with persistence to disk and master-master replication (all nodes are identical)

28 http://10gen.com/

29 http://cassandra.apache.org/

30 http://www.couchbase.com/membase

Best used: Any application where low-latency data access, high concurrency support and high availability is a requirement. *[2]*

For example: Low-latency use-cases like ad targeting or highly-concurrent webapps like online gaming *[2]*

- **Redis:**[31] A very fast NoSQL database that **keeps most data in memory**. Provides master-slave replication and transactions. As Memcached it does not scale, can do sharding by handling it in the client, and therefore, you can't just start adding new servers and increase your throughput. Nor is it fault tolerant. Your Redis server dies, and there goes that data. Redis also supports replication. *[3]*

  Best used: For rapidly changing data with a foreseeable database size (should fit mostly in memory). *[2]*

  For example: Stock prices. Analytics. Real-time data collection. Real-time communication. *[2]*

- **Riak:**[32] Riak is a NoSQL database implementing the principles from **Amazon's Dynamo storage system**. **Riak provides built-in Map Reduce** support, full-text search, indexing & querying. Comes in "open source" and "enterprise" editions.

  Best used: If you want something Cassandra-like, but no way you're gonna deal with the bloat and complexity. If you need very good single-site scalability, availability and fault-tolerance, but you're ready to pay for multi-site replication. *[2]*

  For example: Point-of-sales data collection. Factory control systems. Places where even seconds of downtime hurt. Could be used as a well-update-able web server. *[2]*

4. Graph databases

This kind of databases are thought for data whose relations are well represented with a graph-style (elements interconnected with an undetermined number of relations between them). The kind of data could be s**ocial relations**, **public transport links**, **road maps** or **network topologies**, by example.

Examples of Open Source Graph NoSQL databases could be:

- **Neo4j:**[33] Graph database with full ACID conformity, transactions, indexing of nodes and relationships and advanced path-finding with multiple algorithms.
- **FlockDB:**[34] Graph database created by Twitter for managing plus de 13 billion of relationships between its users.

5. Tabular databases

In this kind of DBMS data is organized in columns. Each rows has one or more values for a number of possible columns.

Example of Open Source Tabular NoSQL databases are:

- **HBase:**[35] An alternative to Google's Big Table that uses Hadoop's HDFS as storage. Offers Map/reduce with

31 http://redis.io/

32 http://wiki.basho.com/Riak.html

33 http://neo4j.org

34 https://github.com/twitter/flockdb

35 http://hbase.apache.org/

Apache Hadoop.

> **Best used:** When you use the Hadoop/HDFS stack. When you need random, real-time read/write access to Big Table-like data. *[2]*
>
> **For example:** For data that's similar to a search engine's data. *[2]*

- **Cassandra**, aforementioned, could be considered also a tabular database due to keys map to multiple values, which are grouped into **column families**

6. MySQL as NoSQL database

Recently a preview version of next MySQL 5.6 *[5]* at MySQL Developer Zone has been released by Oracle. This version has a NoSQL interface.

With this interface applications could write and read to a InnoDB storage using a Memcache-type API. The data could be in memory or stored in the InnoDB Storage Engine, and in the value multiple columns could be stored.

This software is yet experimental, but in the future could be interesting. More info can be read at the following articles:

- "MySQL 5.6 preview introduces a NoSQL interface" at The "H" web. *[6]*
- "NoSQL to InnoDB with Memcached" at "Transactions on InnoDB" blog. *[7]*

7. Other interesting readings and feature comparisons

- "Consistency Models in Non-Relational Databases" by Guy Harrison : A good explanation of CAP Theorem, Eventual consistency and how consistency problems can be handled in distributed environments. *[8]*
- The appendix of the O'Really book "Cassandra: The Definitive Guide" makes a very good description of the NoSQL current status. *[9]*

The following articles done good explanations to know which NoSQL solutions is the right choice for the scenario we are facing

- "CouchDB vs MongoDB" by Gabriele Lana: A good comparison between CouchDB and MongoDB with an excellent explanation of Map/Reduce. *[10]*
- "Should I use MongoDB or CouchDB (or Redis)?", by Riyad Kalla. *[11]*

Detailed description of some NoSQL DBMS

I have read a lot over a few NoSQL Database Management Systems which could be interesting for our needs (log management). In particular there are three that caught initially my attention: Cassandra, CouchDB and MongoDB.

In this section I will made a brief description of each one.

1. Cassandra

Apache Cassandra[36] is an open source distributed NoSQL database management system. It is an Apache Software Foundation top-level project *[1]* designed to handle very large amounts of data spread out "'across many commodity servers'" while providing a highly available service with "'no single point of failure'".

Cassandra provides a **structured key-value store with tunable consistency**. Keys map to multiple values, which are grouped into column families. Different keys can have different numbers of columns. This makes Cassandra a hybrid data management system between a **key-value and a tabular database**.

History

Apache Cassandra was developed at Facebook to power their Inbox Search feature by Avinash Lakshman (one of the authors of Amazon's Dynamo) and Prashant Malik.

It was released as an open source project on Google code in July 2008. In March 2009, it became an Apache Incubator project. On February 17, 2010 it graduated to a top-level project. *[12]*

Facebook abandoned Cassandra in late 2010 when they built Facebook Messaging platform on HBase. *[13]*

Licensing and support

Apache Cassandra is an Apache Software Foundation project, so it has an Apache License (version 2.0)[37].

There are professional grade support available from a few companies. In the official wiki of Apache Cassandra's project *[14]* the following ones, which collaborate with developers to the project, are mentioned

- Acunu[38]
- Datastax[39]

Main features

- **Decentralized**

  Every node in the cluster has the same role. There is **no single point of failure**. Data is distributed across the cluster (so each node contains different data), but there is no master as every node can service any request.

- **Supports replication and multi datacenter replication**

36 http://cassandra.apache.org/

37 http://www.apache.org/licenses/LICENSE-2.0.html

38 http://www.acunu.com/

39 http://datastax.com/

Replication strategies are configurable *[15].* Cassandra is designed as a distributed system, for deployment of large numbers of nodes across multiple data centers. Key features of Cassandra's distributed architecture are specifically tailored for multiple datacenter deployment, for redundancy, for failover and disaster recovery.

- **Elasticity**

  Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.

- **Fault-tolerant**

  Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime.

- **Tunable consistency**

  Writes and reads offer a tunable level of consistency, all the way from "writes never fail" to "block for all replicas to be readable", with the quorum level in the middle.

- **Map Reduce support**

  Cassandra has Hadoop integration, with Map Reduce support. There is support also for Apache Pig[40] and Apache Hive[41]. *[16]*

- **Query language**

  CQL (Cassandra Query Language) was introduced, an SQL-like alternative to the traditional RPC interface. Language drivers are available for '''Java''' (JDBC) and '''Python''' (DBAPI2).

Enterprises who use Cassandra

A very brief list of known enterprises who uses Cassandra could be:

- **Netflix**, uses Cassandra as their back-end database for their streaming services *[17]*
- **Twitter**, announced it is planning to use Cassandra because it can be run on large server clusters and is capable of taking in very large amounts of data at a time. *[18]*
- **Urban Airship**, uses Cassandra with the mobile service hosting for over 160 million application installs across 80 million unique devices. *[19]*
- **Constant Contact**, uses Cassandra in their social media marketing application. *[20]*
- **Rackspace** *[21]*
- **Cisco's WebEx** uses Cassandra to store user feed and activity in near real time. *[22]*

A list more complete can be looked up at Datastax "Cassandra Users" page[42]

Data manipulation: keys, row keys, columns and column families

As said in "NoSQL State of the Question" section we could consider Cassandra **a hybrid between a key-value and a tabular database**.

For each key in Cassandra corresponds a value which is an object. Each key has values as columns, and columns are grouped together into sets called column families. Also, each column families can be grouped in super column families.

40 http://pig.apache.org/

41 http://hive.apache.org

42 http://www.datastax.com/cassandrausers

So each key identifies a row, of variable elements number. This column families could be considered then as tables. A table in Cassandra is a distributed multi dimensional map indexed by a key.

Furthermore, applications can specify the sort order of columns within a Super Column or Simple Column family.

Tools for Cassandra

Cassandra has built in tools for accessing Cassandra from the direct download such **cassandra-cli** and **node-tool**.

There are third party tools available, as the following: *[23]*

Data browsers

- Chiton[43], a GTK data browser.
- cassandra-gui[44], a Swing data browser.

Administration tools

- OpsCenter[45], OpsCenter is a tool for management and monitoring of a Cassandra cluster. The Community Edition of OpsCenter is free for anyone to download and use. There is also an Enterprise Edition of OpsCenter that includes additional features.
- "Cassandra Cluster Admin"[46], Cassandra Cluster Admin is a GUI tool to help people administrate their Apache Cassandra cluster, similar to PHPMyAdmin for MySQL administration.

Client interfaces and language Support

Cassandra has a lot of high-level client libraries for Python, Java, .Net, Ruby, PHP, Perl, C++, etc. *[24]*

For a detailed list of client software go to "Client Options" article on Cassandra's Wiki[47]

Integration with other tools

There are other tools worth mentioning like **Solandra**[48], a Cassandra backend for Apache Solr[49], a web application built around Lucene, for full text indexing and search.

For monitoring purposes Cassandra is well integrated with Ganglia *[25]* and there are plugins for other monitoring system as, by example, Nagios.

Conclusions

If we need to handle very big amounts of data, with more writes than reads (as for real time data analysis, by example) Cassandra could be a good NoSQL solution.

I will emphasize the following:

- Hadoop's utilisation for Map Reduce is integrated in Cassandra [16], but the architecture will be fairly complex (simpler than HBase according to Dominic Williams [26])

43 http://github.com/driftx/chiton

44 http://code.google.com/p/cassandra-gui

45 http://www.datastax.com/products/opscenter

46 https://github.com/sebgiroux/Cassandra-Cluster-Admin

47 http://wiki.apache.org/cassandra/ClientOptions

48 https://github.com/tjake/Solandra Solandra source at Github

49 http://lucene.apache.org/solr/

- It's tunable consistency and multi datacenter support

Interesting readings

- “Cassandra - A Decentralized Structured Storage System”, a 2009 paper presenting Cassandra by their creators Avinash Lakshman and Prashant Malik. *[25]*
- “4 Months with Cassandra, a love story”, a chronicle and main reasons why Cassandra was adopted at CloudKick. *[27]*
- “HBase vs Cassandra: why we moved” post from Dominic Williams blog, where he explains why they moved from HBase to Cassandra. *[26]*)
- “HBase vs Cassandra”, from Adku blog. *[28]*
- “Cassandra vs. (CouchDB | MongoDB | Riak | HBase)”, from Brian O'Neill blog. *[29]*
- “Introduction to Cassandra: Replication and Consistency” presentation by Benjamin Black. *[30]*

2. CouchDB

CouchDB[50] is an open source **document-oriented NoSQL database system**. It's similar to MongoDB

Created in 2005, began an Apache Software Foundation project in 2008, and makes part of the "new" NoSQL family of database systems. Instead of storing data in tables as is made in a "classical" relational database, CouchDB **store structured data as JSON documents** with dynamic schemas, making easier and faster the integration of data in certain type of applications.

CouchDB is interesting in part due to its Multi-Version Concurrent Control. This means that we have **versioning support** for the data and that readers will not block writers and writers will not block readers.

CouchDB uses a RESTful JSON API for accessing data, which allows accessing data using HTTP request.

Other features are ACID semantics with **eventual consistency**, **Map Reduce**, incremental replication, and fault-tolerance. It comes with a web console.

History

CouchDB ("Couch" is an acronym for "cluster of unreliable commodity hardware") *[31]* is a project created in April 2005 by Damien Katz, former Lotus Notes developer at IBM. Damien Katz defined it as a "storage system for a large scale object database". His objectives for the database were for it to become the database of the Internet and that it would be designed from the ground up to serve web applications. He self-funded the project for almost two years and released it as an open source project under the GNU General Public License.

In February 2008, it became an Apache Incubator project and the license was changed to the Apache License *[32]*. A few months after, it graduated to a top-level project. *[33]*

Currently, CouchDB is maintained at the Apache Software Foundation with backing from IBM. Katz works on it full-time as the lead developer.

First stable version was released in July 2010 *[34]*. Last version is 1.2, released in April 2012.

Licensing and support

CouchDB is an Apache Software Foundation project, and so it has an Apache License 2.0

CouchDB has commercial support, by the enterprises Couchbase[51] and Cloudant[52].

Main features

A summary of main features could be the following

- **Document Storage**

CouchDB stores data as "documents", as one or more field/value pairs expressed as JSON. Field values can

50 http://couchdb.apache.org/

51 http://www.couchbase.com

52 https://cloudant.com/

be simple things like strings, numbers, or dates. But you can also use ordered lists and associative arrays. Every document in a CouchDB database has a unique id and there is no required document schema.

- **ACID Semantics**

  CouchDB provides ACID semantics *[35]* It does this by implementing a form of Multi-Version Concurrency Control, meaning that CouchDB can handle a high volume of concurrent readers and writers without conflict.

- **Map/Reduce Views and Indexes**

  The data stored is structured using **views**. In CouchDB, **each view is constructed by a JavaScript function** that acts as the Map half of a map/reduce operation. The function takes a document and transforms it into a single value which it returns. CouchDB can index views and keep those indexes updated as documents are added, removed, or updated.

- **Distributed Architecture with Replication**

  CouchDB was designed with **bi-direction replication (or synchronization) and off-line operation** in mind. That means multiple replicas can have their own copies of the same data, modify it, and then sync those changes at a later time.

- **REST API**

  All items have a unique URI that gets exposed via HTTP. REST uses the HTTP methods POST, GET, PUT and DELETE for the four basic (Create, Read, Update, Delete) operations on all resources.

- **Eventual Consistency**

  CouchDB guarantees eventual consistency to be able to provide both availability and partition tolerance.

- **Built for Offline**

  CouchDB can replicate to devices (like smartphones) that can go offline and handle data sync for you when the device is back online.

CouchDB offers also a **built-in admin interface accessible via web** called Futon *[36]*.

Use cases & production deployments

Replication and synchronization capabilities of CouchDB make it ideal for using it in mobile devices, where network connection is not guaranteed but the application must keep on working offline.

CouchDB is well suited for applications with accumulating, occasionally changing data, on which pre-defined queries are to be run and where versioning is important (CRM, CMS systems, by example). Master-master replication is an especially interesting feature, allowing easy multi-site deployments *[2]*.

Enterprises who use CouchDB

CouchDB is used in certain **applications for Android** like “SpreadLyrics”[53] and **applications for Facebook** like “Will you Kissme” or “Birthday Greeting Cards” or webs like “Friendpaste” *[37]*.

A few examples of enterprises that used or are using CouchDB are:

53 https://play.google.com/store/apps/details?id=br.com.smartfingers.spreadlyrics

- Ubuntu[54] for its synchronization service “Ubuntu One” until November 2011 *[38]* but was discontinued because of scalability issues. *[39]*
- The BBC[55], for its dynamic content platforms. *[40]*
- Credit Suisse[56], for internal use at commodities department for their marketplace framework. *[37]*
- Meebo[57], for their social platform (web and applications)

For a complete list of software projects and web sites that use CouchDB, read the “CouchDB in the wild” *[37]* article of the product's web.

Data manipulation: Documents and Views

CouchDB is similar to other document stores like MongoDB. **CouchDB manages a collection of JSON documents. The documents are organised via views**. Views are defined with aggregate functions and filters are computed in parallel, much like Map Reduce.

Views are generally stored in the database and their indexes updated continuously. CouchDB supports a view system using external socket servers and a JSON-based protocol. *[41]* As a consequence, view servers have been developed in a variety of languages (JavaScript is the default, but there are also PHP, Ruby, Python and Erlang).

Accessing data via HTTP

CouchDB provides a set of RESTful HTTP methods (e.g., POST, GET, PUT or DELETE). We could access to the data, using cURL, by example.

A few examples of data accessing via HTTP could be:

- For accessing CouchDB server info:

```
curl http://127.0.0.1:5984/
```

The CouchDB server returns a response in JSON format:

```
{"couchdb":"Welcome","version":"1.1.0"}
```

- For creating a database we could:

```
curl -X PUT http://127.0.0.1:5984/wiki
```

If the database does not exist, CouchDB will reply with

```
{"ok":true}
```

or, with a different response message, if the database already exists:

```
{"error":"file_exists","reason":"The database could not be created, the file already exists."}
```

Conclusions

For knowing if CouchDB is for us I will emphasize the following:

- For getting results we have to define **views**. This means that **if our problem could not be resolved with a set of predefined queries CouchDB is not for us**, as it lacks flexibility in the way of querying data
- For this reason the initial learning curve is harder

54 http://www.ubuntu.com/

55 http://www.bbc.co.uk/

56 http://www.credit-suisse.com

57 https://www.meebo.com/about/

- CouchDB has **master-master replication support**, ideal for a different data centers multi-node setup. Also it can **replicate to devices that can go offline** (like smartphones) and handle data sync for you when the device is back online (making it a good solution for applications working on distributed environments with non guaranteed connection)
- CouchDB has **multiple versions support**
- The interface is HTTP/REST, so is easily accessible by any application/language/server

For what is said in a few blogs CouchDB is not a very mature project. Last version is the 1.2.0 and it has breaking changes with regard to the previous version. *[42]*

Interesting readings

- “Why CouchDB?”, from the "CouchDB The Definitive Guide *[43]*
- “Comparing MongoDB and CouchDB”, from MongoDB web *[44]*
- “MongoDB or CouchDB - fit for production?”, question and responses at StackOverflow *[45]*
- “3 CouchDB Case Studies”, post on Alex Popescu NoSQL blog *[46]*
- “CouchDB for access log aggregation and analysis”, post on UserPrimery.net blog *[47]*

3. MongoDB

MongoDB (from “hu*mongo*us”) is an open source **document-oriented NoSQL database system**.

MongoDB makes part of the “new” NoSQL family of database systems. Instead of storing data in tables as is made in a “classical” relational database, MongoDB store structure data as JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making easier and faster the integration of data in certain type of applications.

Development of MongoDB began in October 2007 by 10gen[58]. It is now a mature and feature rich database ready for production use. It's used, by example, by MTV Networks *[48]*, Craigslist *[49]* or Foursquare *[50]*.

History

Development of MongoDB began at 10gen in 2007, when the company was building a “Platform as a Service” similar to “Google App Engine” *[51]*. In 2009 MongoDB was open sourced as a stand-alone product *[52]*, with an GNU Affero General Public License (or **AGPL**[59]**)** license.

In March 2011, from version 1.4, MongoDB has been considered production ready *[53]*.

The last stable version is 2.2.0, released in August 2012.

Licensing and support

MongoDB is available for free under the GNU Affero General Public License *[54]*. The language drivers are available under an Apache License.

MongoDB is being developed by 10gen] as a business with commercial support available *[55]*.

Main features

A summary of main features could be the following

- **Ad hoc queries**

  MongoDB supports search by field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.

- **Indexing**

  Any field in a MongoDB document can be indexed (indexes in MongoDB are conceptually similar to those in RDBMS). Secondary indexes and “geospatial indexes are also available”.

- **Replication**

  MongoDB supports **“master-slave replication”**. A master can perform reads and writes. A slave copies data from the master and can only be used for reads or backup (not writes). The slaves have the ability to elect a new master if the current one goes down.

- **Load balancing**

  MongoDB scales horizontally using a system called sharding *[56]*. The developer chooses a shard key, which

58 http://en.wikipedia.org/wiki/10gen

59 http://www.gnu.org/licenses/agpl.html

determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more slaves.)

MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure. Automatic configuration is easy to deploy and it's possible to add new machines to a running database.

- **File storage**

  MongoDB could be used as a file system, taking advantage of load balancing and data replication features over multiple machines for storing files.

  This function, called GridFS *[57]*, is included with MongoDB drivers and available with no difficulty for development languages. MongoDB expose functions for manipulate files and their contents to developers. GridFS is used, by example, in plugins for NGINX *[58]* and lighttpd *[59]*.

  In a multiple machines MongoDB system, files could distributed and copied multiple times between machines transparently, having then a load balanced & fault tolerant system.

- **Aggregation**

  Map Reduce can be used for batch processing of data and aggregation operations. The aggregation framework enables users to obtain the kind of results SQL group-by is used for

- **Server-side JavaScript execution**

  JavaScript can be used in queries, aggregation functions (**such as Map Reduce**), are sent directly to the database to be executed.

- **Capped collections**

  MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a circular queue.

Use cases & production deployments

According to "Use Cases" article at product's web MongoDB *[60]* is well suited for following cases:

- **Archiving** and **event logging**
- **Document and Content Management Systems**. as a document-oriented (JSON) database, MongoDB's flexible schemas are a good fit for this.
- E-Commerce. Several sites are using MongoDB as the core of their e-commerce infrastructure (often in combination with an RDBMS for the final order processing and accounting).
- **Gaming**. High performance small read/writes are a good fit for MongoDB; also for certain games geospatial indexes can be helpful.
- **High volume problems**. Problems where a traditional DBMS might be too expensive for the data in question. In many cases developers would traditionally write custom code to a file system instead using flat files or other methodologies.
- **Mobile**. Specifically, the **server-side infrastructure of mobile systems**. Geospatial key here.
- **Operational data store of a web site**. MongoDB is very good at real-time inserts, updates, and queries. Scalability and replication are provided which are necessary functions for large web sites' real-time data stores. Specific web use case examples

- **Projects using iterative/agile development methodologies**. Mongo's BSON data format makes it very easy to store and retrieve data in a document-style / “schemaless” format. Addition of new properties to existing objects is easy and does not generally require blocking "ALTER TABLE" style operations.
- **Real-time stats/analytics**

Enterprises who use MongoDB

Between the enterprises who use MongoDB there are: “MTV Networks”, Craigslist, “Disney Interactive Media Group”, Wordnik, Diaspora, Shutterfly, foursquare, bit.ly, “The New York Times”, SourceForge, “Business Insider”, Etsy, “CERN LHC”, Thumbtack, AppScale, Uber or “The Guardian”

For a complete list and references on each particular use case visit the article "Production Deployments" on MongoDB's web *[61]*

Data manipulation: Collections and Documents

MongoDB store structure data as JSON-like documents with dynamic schemas (called BSON, with no predefined schema.

The element of data is called **documents**, stored in **collections**. One collection may have any number of documents.

Compared to relational databases we could say collections are as tables, and documents are as records. But there is one big difference: every record in a table have the same number of fields, while documents in a collection could have completely different fields.

One table of a few records with the fields “Last name”, “First name” and “Age” and possible others like “Address” or “City” could be described as the following MongoDB collection:

```
{
"_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
"Last Name": "DUMON",
"First Name": "Jean",
"Age": 43
},

{
"_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
"Last Name":  "PELLERIN",
"First Name": "Franck",
"Age": 29,
"Address":
     {
      "Street" : "1 chemin des Loges",
      "City": "VERSAILLES"
     }
"City": "VERSAILLES"
}
```

Documents in a MongoDB collection could have different fields (note: “_id” field is obligatory, automatically created by MongoDB, it's a unique index which identify the document

In a document, new fields could be added, existing ones suppressed, modified or renamed at any moment. There is no predefined schema. A document structure is really simple and composed of key-value pairs like associative arrays in programming languages JSON format.

The key is the field name, the value is its content. As value we could use numbers, strings and also binary data like images **or another key-value pairs**.

Language Support

MongoDB has official drivers for: C[60], C++[61], C# / .Net[62], Erlang[63], Haskell[64], Java[65], JavaScript[66], Lisp[67], Perl[68], PHP[69], Python[70], Ruby[71] and Scala[72].

There are also a large number of unofficial drivers for ColdFusion[73], Delphi[74], Lua[75], node.js[76], Ruby[77], Smalltalk[78] and much others.

Management and graphical frontends

MongoDB tools

In a MongoDB installation there are available the following commands

- **mongo**

  MongoDB offers an interactive shell called "mongo" *[62]*, which lets developers view, insert, remove, and update data in their databases, as well as get replication information, setting up sharding, shut down servers, execute JavaScript, and more.

  Administrative information can also be accessed through a **web interface**, a simple web page that serves information about the current server status. By default, this interface is 1000 ports above the database port (28017).

- **mongostat**

  "mongostat" is a command-line tool that displays a simple list of stats about the last second: how many inserts, updates, removes, queries, and commands were performed, as well as what percentage of the time the database

60 http://github.com/mongodb/mongo-c-driver

61 http://github.com/mongodb/mongo

62 http://www.mongodb.org/display/DOCS/CSharp+Language+Center

63 https://github.com/TonyGen/mongodb-erlang

64 http://hackage.haskell.org/package/mongoDB

65 http://github.com/mongodb/mongo-java-driver

66 http://www.mongodb.org/display/DOCS/JavaScript+Language+Center

67 https://github.com/fons/cl-mongo

68 http://github.com/mongodb/mongo-perl-driver

69 http://github.com/mongodb/mongo-php-driver

70 http://github.com/mongodb/mongo-python-driver

71 http://github.com/mongodb/mongo-ruby-driver

72 https://github.com/mongodb/casbah

73 http://github.com/virtix/cfmongodb

74 http://code.google.com/p/pebongo/

75 http://code.google.com/p/luamongo/

76 http://www.mongodb.org/display/DOCS/node.JS

77 http://github.com/tmm1/rmongo

78 http://www.squeaksource.com/MongoTalk.html

was locked and how much memory it is using.

- **mongosniff**

  “mongosniff” sniffs network traffic going to and from MongoDB.

Monitoring plugins

There are MongoDB monitoring plugins available for the following network tools: Munin[79], Ganglia[80], Cacti[81], Scout[82]

Cloud-based monitoring services

“MongoDB Monitoring Service”[83] is a free, cloud-based monitoring and alerting solution for MongoDB deployments offered by 10gen, the company who develops MongoDB

Web & Desktop Application GUIs

Several GUIs have been created by MongoDB's developer community to help visualize their data. Some popular ones are:

Open Source tools

- RockMongo[84]: PHP based MongoDB administration GUI tool
- phpMoAdmin[85]: another PHP GUI that runs entirely from a single 95kb self-configuring file
- JMongoBrowser[86]: a desktop application for all platforms
- Mongo3[87]: a Ruby-based interface
- Meclipse[88]: Eclipse plugin for interacting with MongoDB

Proprietary tools

- MongoHub[89]: a Freeware native Mac OS X application for managing MongoDB
- “Database Master”[90]: development and administration tool for Oracle, SQL Server, MySQL, PostgreSQL, MongoDB, SQLite ... which allows run SQL, LINQ and JSON queries over databases. Developed by Nucleon Software[91] for Windows systems
- “BI Studio”: business intelligence and data analysis software which allows design database reports, charts and

79 http://github.com/erh/mongo-munin
80 http://github.com/quiiver/mongodb-ganglia
81 http://tag1consulting.com/blog/mongodb-cacti-graphs
82 http://scoutapp.com/plugin_urls/291-mongodb-slow-queries
83 http://www.10gen.com/mongodb-monitoring-service
84 http://code.google.com/p/rock-php/wiki/rock_mongo
85 http://www.phpmoadmin.com/
86 http://www.edgytech.com/jmongobrowser/
87 http://mongo3.com/
88 http://update.exoanalytic.com/org.mongodb.meclipse/
89 http://mongohub.todayclose.com/
90 http://www.nucleonsoftware.com
91 http://www.nucleonsoftware.com

dashboards, from the same company that Database Master

Conclusions

For knowing if MongoDB is for us I will emphasize the following:

- MongoDB **has a query language**, which makes getting the data dynamic and flexible
- The interface is a custom protocol over TCP/IP, with native drivers for a lot of languages. **The utilisation of a binary protocol make the operations faster than others** (like CouchDB)
- **Replication is master-slave ONLY**, as with MySQL. If you need multiple masters in a Mongo environment, you have to set up sharding.

Other consideration I will do are:

- the documentation on the product's web page is very good (by example, "Use cases" documentation *[60]*)
- as said there is commercial support available.

Interesting Readings

- "MongoDB Schema Design: How to Think Non-Relational" Jared Rosoff's presentation at Youtube *[63]*
- "Real-time Analytics with MongoDB", presentation by Jared Rosoff *[64]*
- "Web Analytics using MongoDB" of "PHP and MongoDB Web Development Beginner's Guide" book *[65]*

Looking for a NoSQL solution for our needs

1. Introduction and initial approach

As I said the main justification we have for looking at NoSQL technologies is log management.

In the enterprise I work, we have two use cases:

- **Analysis of Internet access logs**

  Right now this access are stored in MySQL tables. We're talking about tens of **thousands of users with Internet access** and a few logs that record every URL and download size. **As several Gigabytes are generated daily** what is done is to store logs in different tables partitioning the data vertically by month (a table for January, one for February, etc ...)

- **Log analysis of geographically distributed Unix servers**

  Used for communication with sales offices, we are talking of logs of the various services (FTP, SSH, Apache, MySQL, file transfer services developed by the company ...) of about 1000 Linux servers

For this analyse we will chose the first one

2. Analyse of Internet access logs, describing the problem

What we want is to achieve is an efficient storage and analysis of the logs of communications made by employees (tens of thousands) with Internet access. At least more efficient that our actual solution.

Now has been decided to divide the data in different tables by month. This decision has been taken for reasons of volume. One immediate consequence is that quite complicated to make queries asking for data of different months as the developer will have to think carefully how to design the queries and the server will have to access multiple tables.

The critical problem here is to handle huge amounts of data. Do we need relations? Yes and no. If we gain performance we don't mind to repeat certain data (as user name, by example).

We will get statistics and reports based on log analysis, each one of this record will include information such as:

- Date and time of access
- User name
- URL accessed
- Size in bytes of the network transmission

The questions we want to answer are the like the following type:

- What are the most visited pages?
- And the most visited per month? And the last week?
- What users spend more time online?
- What are the 100 users whose traffic volume is greater?

- What is the average daily volume of traffic from the corporate network to the Internet?

Data size estimation

The size and number of records of data that is being stored by month are:

- Data size: between 150 and 300 GB
- Log entries number: between 650 millions and and 1.700 millions

For a user population of around 70.000 who access to 1,5 millions of domains

> So, given the actual traffic size, in a year we could reach a volume stored of **3.600 GB for 20 billions of entries** in the log

3. Oh my God! We have a lot of options!

Well, well, well ....

As said in the second chapter “NoSQL, State of the Question”, we have a lot of NoSQL Database Management Systems and for a beginner it seems difficult to know where to begin.

After reading a lot on the subject (see read recommendations at the end of this section) the most known Open Source NoSQL DBMS are: “MongoDB”[92], “CouchDB”[93], “Cassandra”[94], “Membase”[95], “Redis”[96], “Riak”[97], “Neo4J”[98], “FlockDB”[99] and “HBase”[100], among others.

The first thing I would recommend would be to give a quick read to the article “NoSQL” on English Wikipedia[101] (with contributions from myself :-)) and the first article in the series “Picking the Right NoSQL Database Tool”[102]

A quick summary (as reminder, it's explained with more detail on second chapter) would be that, depending on the way data is organized, NoSQL databases are divided into ...

- **Document oriented**

  As “MongoDB” or “CouchDB”. Data is stored in structured formats (records) as “JSON”. Each data unit is called a “document” (here this word have nothing to do with a file type).

- **Key-Value**

  As “Cassandra”, “Membase”, “Redis” or “Riak”. Data is stores in key-value pairs (a value might be an object)

- **Graph oriented**

92 http://www.mongodb.org/

93 http://couchdb.apache.org/

94 http://cassandra.apache.org

95 http://www.couchbase.com/membase

96 http://redis.io/

97 http://wiki.basho.com/

98 http://neo4j.org/

99 http://github.com/twitter/flockdb

100 http://hbase.apache.org/

101 http://en.wikipedia.org/wiki/NoSQL

102 http://blog.monitis.com/index.php/2011/05/22/picking-the-right-nosql-database-tool/

As “Neo4J” or “FlockDB”. They store the elements and their relationships with a graph style (for social networks, transport networks, road maps, network topologies, for example)

- **Tabular**

  As “Cassandra” or “HBase”. Data is stored in rows with several columns that correspond to a key, with a similar result to a table

4. Questions we should answer before making a choice

There is no “one fits all” NoSQL solution, as this term apply to a wide range of database management systems. It's perfectly possible and reasonable to use several systems at the same time (a relational DBMS as MySQL and one or more NoSQL DBMS), depending on the type of data to store and to query.

In the end the choice will depend on the nature of the problem we want to solve.

A NoSQL solution does not replace a relational database, complements it for a kind of problems where relational DBMS have not enough performance. Unless, of course, we're using a SQL database for the wrong problem.

We should be able to answer the following questions before looking for a product:

- What type of data will be handled? This data could be naturally organized in associative Arrays? Or in key-value pairs? It is data which will fit in a XML or similar structure?
- Do we need transactions?
- Do we need to use “Map Reduce”?

And when reviewing the different options:

- The latest version is considered stable?
- Does it have commercial support?
- What is the learning curve?
- Is good documentation available? Is there an active community?

5. Description of the data for an Internet Access Log management system

Description of the problem

As said will be the kind of data we manage, its structure and the nature of the problem which will lead us to one or other NoSQL solution.

The data we want to manage are access logs generated by several HTTP proxies of the company for several tens of thousands of users.

We have two different type of records: records from FTP access and from the rest (mainly HTTP):

- For each FTP access we will save
  - IP of the host that makes the request
  - date and time of access

    - Internet domain accessed
    - URI
    - size of the transfer
- For each NonFTP access:
    - IP of the host that makes the request
    - the user id
    - date and time of access
    - the HTTP method used
    - protocol
    - Internet domain accessed
    - URI
    - HTTP return code
    - size of the transfer

Besides storing the data **the following statistical reports will be created**

- Hits number and volume of data transferred **by Internet domain**, daily and monthly
- Hits number and volume of data transferred **by user**, daily and monthly

Definition of our needs

So we can reach to the following first definition of our needs:

- each data entry could be represented as an associative array
- each record in unrelated to each other
- each entry is stored in a log table as it grows indefinitely
- accesses to the database are mostly writing
- each access means a change in the statistical values which reflect daily and monthly access by domain and user
- the list of queries sent by our application is known (anyway, the schema should be defined as new ones can be easily made)

What lead us to the following conclusions:

- The data are records with multiple fields, so we need a document-oriented database or tabular (multiple columns for a record)
- Map Reduce is desired. For having reports in real time each access will update the daily and monthly statistics for domain and user
- We don't need master-master replication (proxies in different geographic areas manage accesses from different users)
- We don't need support for multiple versions (there is no such a thing in a log)
- We don't need real data consistency

- We don't need transactions (data will be added one after another, isolated)

And also the product chosen must be:

- Open Source
- Ready for production environments (stable)
- With professional support

Not bad

6. Choosing between several NoSQL products

If we discard the databases that hold data in memory, as key-value pairs and graphs we are left with the following options: “MongoDB”, “CouchDB”, “Cassandra”, “HBase” and “Riak”.

To what is told in all read documents I will add the following thoughts (read also the “Interesting readings” sections of different chapters):

MongoDB[103]

PROS:

- It's a document-oriented database therefore very flexible in structuring the data (uses JSON)
- It has a dynamic query language
- Has professional support by the company that developed the product, 10gen[104]
- It has a **large and active community** (present at conferences, I have seen them at FOSDEM[105] in Brussels this year)
- It has support for Map Reduce
- It's a **mature product**, considered production ready (current version is 2.2)
- The **documentation on their website is really good**
- There are native drivers for multiple languages made by 10gen

CONS:

- Although it is not difficult to install and run MongoDB is not a simple product. An installation of MongoDB has several types of services: data servers, configuration servers, servers that route the client requests
- Replication is only master-slave

103 http://www.mongodb.org/

104 http://www.10gen.com/

105 https://fosdem.org

CouchDB[106]

PROS:

- It is a document-oriented database, so very flexible in structuring the data
- Concurrent Versions System
- It has **master-master replication**

CONS:

- For achieving versioning data is not modified, each time a modification is done a new version is added. This takes a lot of disk space and batch processes are necessary for data compaction operations
- **It is not very mature**. The latest version is 1.2.0 and has changes that make it incompatible with the previous versions
- **To exploit the data is necessary to define views**, which means that queries must be defined in advance (not very flexible)

Riak[107]

PROS:

- It is a hybrid database, store documents and key-value pairs
- There is no central controller and therefore no single point of failure
- It has support for Map-Reduce
- It has support for transactions

CONS:

- It has two versions, one open source and a commercial one with multi-site replication

Cassandra[108]

PROS:

- It is an Apache project, considered of maximum importance
- It is a tabular database (can store multiple key columns) making it flexible and valid for our case
- **Designed for situations where there is more writes than reads**. Scale very well in these cases (ideal for log analysis)
- Designed to **replicate data between multiple data centers**
- Provides **integration with Hadoop[109] for Map Reduce**
- The consistency level is configurable

106 http://couchdb.apache.org/

107 http://wiki.basho.com/

108 http://cassandra.apache.org/

109 http://hadoop.apache.org/

- There is no central controller and therefore **no single point of failure**
- It has **support for transactions** (with ZooKeeper[110])

CONS:

- Maybe too complex

HBase[111]

PROS:

- It is an Apache project (subproject of Hadoop)
- Similar to Cassandra (can store multiple key columns)
- Provides **integration with Hadoop for Map Reduce**

CONS:

- Too complex

7. And the Winner is ..... MongoDB!

After reading everything I opted for MongoDB, mainly because:

- meets all the requirements stated at the beginning: document-oriented, with Map-Reduce, Open Source, stable and professionally supported
- support is given by the same company that developed the product, 10gen, which is clear in the know :-)
- has a complete website with extensive documentation
- are very active, they are present in many conferences and lectures (as seen in the article "events"[112] of their web)
- comparatively this product does not seem too complex

As this will be the first deployment of a NoSQL database in the company and made by someone with no previous experience, **I consider vital the availability documentation and comprehensive guides**.

In particular, and for our use case I will highlight the following articles from their web

- "MongoDB is Fantastic for Logging" *[66]*
- "Using MongoDB for Real-time Analytics" *[67]*

There is a good collection of interesting presentations on 10gen's web[113]

110 http://zookeeper.apache.org/

111 http://hbase.apache.org/

112 http://www.mongodb.org/display/DOCS/Events

113 http://www.10gen.com/presentations

8. What we will do from here ....

Once chosen NoSQL management system from here so we have to do is

1. Install and configure MongoDB
2. Designing the MongoDB schema database for Internet Access Logs
3. Develop some code which will allow us to use the schema from MongoDB or MySQL transparently to applications
4. Make some performance tests on the same conditions for verifying that using MongoDB is not only more flexible than MySQL, but also a better idea from a performance point of view

There are other alternatives that seem equally interesting and valid as Cassandra or Riak, and that could be interesting to test in a further study.

9. Interesting readings

- “Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase vs Membase vs Neo4j comparison” from Kristof Kovacs Blog *[2]*
- “Picking the Right NoSQL Database Tool”: post from Monitis' blog *[68]*
- "Consistency Models in Non-Relational Databases" by Guy Harrison : A good explanation of CAP Theorem, Eventual consistency and how consistency problems can be handled in distributed environments. *[8]*

Installation of MongoDB

In this chapter I will describe the technical details for deploying a MongoDB 2.2.0 server in the Linux machines of PSA (Linux distribution is “Suse Enterprise Linux Server”)

In PSA have developed in-house software distribution system (they don't use .deb or .rpm packages neither a known standard for remote installs). The installed software on their servers must meet a strict directory structure.

For reasons of confidentiality in this article we will assume that MongoDB will be installed under “/opt/mongodb-2.2.0”.

1. Deploying MongoDB binaries

After downloading **the legacy static 64 bits version** (version 2.2.0 at September 2012) from MongoDB Downloads page[114] the following commands have been executed:

```
tar -xzvf mongodb-linux-x86_64-static-legacy-2.2.0.tgz
mkdir /opt/mongodb-2.2.0
mkdir /opt/mongodb-2.2.0/bin
mkdir /opt/mongodb-2.2.0/docs
mkdir /opt/mongodb-2.2.0/etc
mkdir /opt/mongodb-2.2.0/logs
mkdir /opt/mongodb-2.2.0/data
mkdir /opt/mongodb-2.2.0/modules
mkdir /opt/mongodb-2.2.0/tmp
mv mongodb-linux-x86_64-static-legacy-2.2.0/bin /opt/mongodb-2.2.0/bin
mv mongodb-linux-x86_64-static-legacy-2.2.0/* /opt/mongodb-2.2.0/docs
chown -Rh root:root /opt/mongodb-2.2.0
chown -Rh nobody:nobody /opt/mongodb-2.2.0/data
chmod -R a+rX /opt/mongodb-2.2.0
```

2. Compiling and installing PHP driver

For the installation of mongo's PHP driver we use the pecl command **for using the PECL repository for PHP extensions**[115]

```
pecl download mongo
tar -xvf mongo-1.2.12.tar
cd mongo-1.2.12
phpize
./configure
make
```

For deploying the PHP module in an Apache installation we will have to copy the **mongo.so** under the “extensions” directory

Last we will have to add the following lines to the php.ini PHP configuration file

114 http://www.mongodb.org/downloads

115 http://pecl.php.net/

```
; Driver pour MongoDB
extension=mongo.so
```

3. Compiling and installing PyMongo, the Python driver for MongoDB

Installing python-devel RPM

For adding new Python modules first we need to install the "python-devel" Linux package (and its dependencies). In a PSA SLES 10 Linux server it means:

```
rpm -i tk-8.4.12-14.12.x86_64.rpm
rpm -i blt-2.4z-222.2.x86_64.rpm
rpm -i python-tk-2.4.2-18.25.x86_64.rpm
rpm -i python-devel-2.4.2-18.25.x86_64.rpm
```

Installing PyMongo

To install Python driver under "/opt/mongodb-2.2.0/modules/python2.4" (without affecting system's Python) **we need to make first a virtual Python installation** following instructions from virtualenv page at Python Package Index[116]

Once the script **"virtualenv.py"** is downloaded the commands for creating the virtual Python environment and installing pymongo, the Python driver for MongoDB, are

```
mkdir -p /opt/mongodb-2.2.0/modules/python2.4
python virtualenv.py /opt/mongodb-2.2.0/modules/python2.4
source /opt/mongodb-2.2.0/modules/python2.4/bin/activate
pip install pymongo
chown -Rh root:root /opt/mongodb-2.2.0/modules/python2.4
```

To use this environment the developers will have to set the following line as the first in his/her Python scripts

```
 #!/opt/mongodb-2.2.0/modules/python2.4/bin/python
```

4. Configuring the server

In addition to accepting Command Line Parameters, MongoDB can also be configured using a configuration file ("/opt/mongodb-2.2.0/etc/mongodb.conf"). The configuration included in our installation is the following:

```
dbpath = /opt/mongodb-2.2.0/data
logpath = /opt/mongodb-2.2.0/logs/mongod.logs
logappend = true
unixSocketPrefix = /opt/mongodb-2.2.0/tmp

#verbose = true
rest = true
fork = true
directoryperdb = true
```

In this configuration the most remarkable element is "directoryperdbd" directive, which means that we are going to use one different directory for each database, for making easier to develop backup scripts later.

116 http://pypi.python.org/pypi/virtualenv

5. Installing RockMongo, a PHP based administration tool

There are a few front-ends for querying and manage MongoDB (a good list can be found on English Wikipedia "MongoDB" article[117]).

After testing a few I have found interesting the PHP web administration tool **"RockMongo"**[118].

So for installing and deploying it we will download the file "rockmongo-v1.1.2.zip" from RockMongo's web page[119] and ...

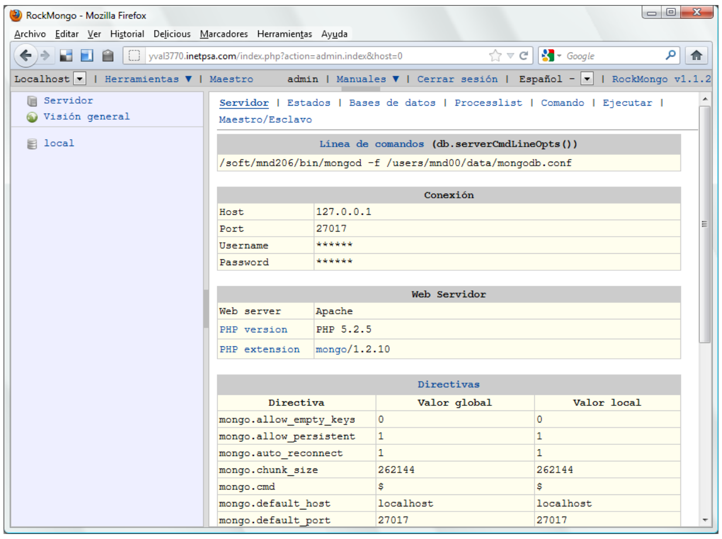
```
mkdir -p /opt/mongodb-2.2.0/web/html
mkdir /opt/mongodb-2.2.0/logs/apache
unzip rockmongo-v1.1.2.zip
cd rockmongo
mv * /users/mnd01/web/html
```

Once the files are deployed we will have to configure Apache. One example for a Virtual Host by name with the (fake) URL myrockmongo.ciges.net could be

```
<VirtualHost *:80>
    DocumentRoot "/opt/mongodb-2.2.0/web/html"
    DirectoryIndex index.php index.html
    ServerName  myrockmongo.ciges.net
    ErrorLog /opt/mongodb-2.2.0/logs/apache/error_mng.log
    CustomLog /opt/mongodb-2.2.0/logs/apache/access_mng common

    <IfModule mod_php5.c>
        php_admin_flag safe_mode Off
    </IfModule>
</VirtualHost>
```

We should see the following screen after making login with de default user "admin" with password "admin":


RockMongo - Mozilla Firefox
Archivo Editar Ver Historial Delicious Marcadores Herramientas Ayuda
yval3770.inetpsa.com/index.php?action=admin.index&host=0
Google
Localhost | Herramientas ▼ | Maestro admin | Manuales ▼ | Cerrar sesión | Español - | RockMongo v1.1.2
Servidor
Visión general
local
Servidor | Estados | Bases de datos | Processlist | Comando | Ejecutar | Maestro/Esclavo

Línea de comandos (db.serverCmdLineOpts())
/soft/mnd206/bin/mongod -f /users/mnd00/data/mongodb.conf

Conexión	
Host	127.0.0.1
Port	27017
Username	******
Password	******

Web Servidor	
Web server	Apache
PHP version	PHP 5.2.5
PHP extension	mongo/1.2.10

Directivas

Directiva	Valor global	Valor local
mongo.allow_empty_keys	0	0
mongo.allow_persistent	1	1
mongo.auto_reconnect	1	1
mongo.chunk_size	262144	262144
mongo.cmd	$	$
mongo.default_host	localhost	localhost
mongo.default_port	27017	27017

117 http://en.wikipedia.org/wiki/MongoDB#Management_and_graphical_front-ends

118 http://code.google.com/p/rock-php/wiki/rock_mongo

119 http://rockmongo.com/?action=downloads

6. Authentication in MongoDB

The current version supports only basic security. We authenticate with a user name and password. By default a normal user has full read and write access to the database. So its important change this in the first connection.

We have to create a database called “admin”, which will contain the users who can have administration rights. The users created on this database will have administration rights on all collections.

Creation of a user with administrator rights

For creating an administrator user called “administrator” the commands will be:

```
> use admin
> db.addUser("administrator", "administrator")
```

To check if the user was created properly

```
> show users
  {
       "_id" : ObjectId("4faa6a53416734d9596ef18d"),
       "user" : "administrator",
       "readOnly" : false,
       "pwd" : "819b3a97e0f6de5ca0b079460c5ea88a"
  }
```

To log with the new user we have to use the client with the following parameters

```
mongo -u administrator -p administrator admin
```

Creation of a user with read only rights

We can have admin users with read only rights or users with read only right for a specific collection

In this example we will create a read only user “test” with access for all the collections

```
> use admin
> db.addUser("test", "test", true)
```

Users with access only to a collection

This kind of users only have rights on their databases. To create them. We need to log as admin and connect to the database where we want to create the new user:

```
> use test
> db.addUser("test", "testpassword")
```

Activation of authentication support

We have two options to activate support in MongoDB:

- We can run mongo script start with the option --auth
- We can add the following line to the configuration file mongodb.conf

```
$ auth = true
```

Activation of authentication support in RockMongo web interface

At last, for activate user recognition in RockMongo we need to set the following variables in “config.php” PHP file:

```
$MONGO["servers"][$i]["mongo_auth"] = true;
$MONGO["servers"][$i]["control_auth"] = false;
```

7. Developed scripts for starting, stopping and verifying MongoDB status

I have developed three shell scripts for starting, stopping and verifying the MondoDB status. Each one of this scripts reads common configuration and code from a file called “profile”. The main features are:

- The processes of the server will be run by user “nobody”
- For verifying that MongoDB is running a process for “nobody” user with the id saved in the PID file is searched in the system
- The start and the stop script store in a log each time they are run
- This scripts must be run as “root”, it's the starting script who runs the server with the user configured using the command “**su**”
- To avoid problems with memory usage by MongoDB we must tell to the operating system that the memory size of the process should be unlimited, using **ulimit -v unlimited** before starting it

Due to this last two reasons the daemon is started with the following line

```
su nobody -c "ulimit -v unlimited; /opt/mongodb-2.2.0/bin/mongod -f /opt/mongodb-2.2.0/etc/mongodb.conf
```

Configuration and common code

```
#!/bin/ksh

#  Installation paths
mng_binpath="/opt/mongodb-2.2.0"
mng_datapath="/opt/mongodb-2.2.0"
mng_dbpath="/opt/mongodb-2.2.0/data"
mng_pidfile="$mng_dbpath/mongod.lock"
mng_logpath="$mng_datapath/logs"
mng_log="$mng_logpath/mongod.log"

#  Daemon properties
mng_daemon="$mng_binpath/bin/mongod"
mng_user="nobody"


#  CONSTANTS
FMT_INFO="\t%s\n"
FMT_ERROR="\tERROR: %s!\n"


#  FUNCTIONS
function mongod_process_running  {
    /bin/ps U $mng_user|grep -v grep|grep -q "$mng_daemon"
}


function mongod_started  {
    if [ -e "$mng_pidfile" ]; then
            if [ -s "$mng_pidfile" ]; then
                    mng_pid=`cat "$mng_pidfile"`
                    /bin/ps U $mng_user|grep -v grep|grep "$mng_daemon"|grep
-q $mng_pid
            else
                    return 1
            fi;
    else
            if mongod_process_running; then
                    printf "$FMT_ERROR" "MongoDB instance running but no pid
file found on \"$mng_pidfile\""
                    exit 1
            else
                    return 1
            fi;
    fi;

}
```

Script to start the server

```
#!/bin/ksh
source /soft/mnd220/fileso/profile

#  Maximum time in second after starting daemon and before returning an
error
MAX_TIME=60

#  This script must be run by root
if [ `id -u` != 0 ]; then
        printf "$FMT_ERROR" "This script must be run by root"
        exit 1
fi;

# Log file
script_log="$mng_logpath/mnd_start.log"
if ! [ -e "$script_log" ]; then
        touch "$script_log"
        chown $mng_user "$script_log"
fi;

printf "\n$FMT_INFO\n" "Starting MongoDB Server at `date`" | tee -a
$script_log

if ! mongod_started; then
    su $mng_user -c "ulimit -v unlimited; $mng_daemon -f
$mng_dbpath/mongodb.conf | tee -a $script_log"
    exit

    sleep 1
    i=1
    while [ $i -lt $MAX_TIME ] && ! mongod_started; do
            sleep 1
    done;
    if ! mongod_started; then
            printf "$FMT_ERROR" "MongoDB server could not be started" | tee
-a $script_log
    else
            printf "$FMT_INFO" "MongoDB server started ok and running" | tee
-a $script_log
    fi;

else
    printf "$FMT_ERROR" "MongoDB server is already running" | tee -a
$script_log
    exit 1
fi;
```

Script to stop the server

```
#!/bin/ksh
source /soft/mnd220/fileso/profile

#  Maximum time in seconds after starting daemon and before returning an
error
MAX_TIME=$((3*60))

#  This script must be run by root
if [ `id -u` != 0 ]; then
        printf "$FMT_ERROR" "This script must be run by root"
        exit 1
fi;

# Log file
script_log="$mng_logpath/mnd_stop.log"
if ! [ -e "$script_log" ]; then
    touch "$script_log"
    chown $mng_user "$script_log"
fi;

printf "\n$FMT_INFO\n" "Stopping MongoDB Server at `date`" | tee -a
$script_log

if mongod_started; then
    su $mng_user -c "$mng_daemon -f $mng_dbpath/mongodb.conf --shutdown|
tee -a $script_log"
    sleep 1
    i=1
    while [ $i -lt $MAX_TIME ] && mongod_started; do
            sleep 1
    done;

    if mongod_started; then
            printf "$FMT_ERROR" "MongoDB server could not be stopped" | tee
-a $script_log
    else
            printf "$FMT_INFO" "MongoDB server sttoped" | tee -a $script_log
    fi;

else
    printf "$FMT_ERROR" "MongoDB server is not running" | tee -a
$script_log

    exit 1
```

```
fi;
```

Script to verify the status

```
#!/bin/ksh
source /soft/mnd220/fileso/profile

if mongod_started; then
   printf "$FMT_INFO" "MongoDB server is running"
    return 0
else
   printf "$FMT_INFO" "MongoDB server is NOT running"
    return 1
fi;
```

NoSQL Schema Design for Internet Access Logs

1. Analysis of logs with NoSQL

The analyse of logs (in real time while the data is being received or processing data already stored) is the type of problem for which NoSQL solutions are particularly suitable. We have a great (or even huge) amount of data that increases without end, and where the relationships are not really important (we don't need to normalise the elements of data).

In this article I will explain the design of the schema chosen for an equivalent solution implemented with MySQL and with MongoDB.

2. Description of an equivalent MySQL database

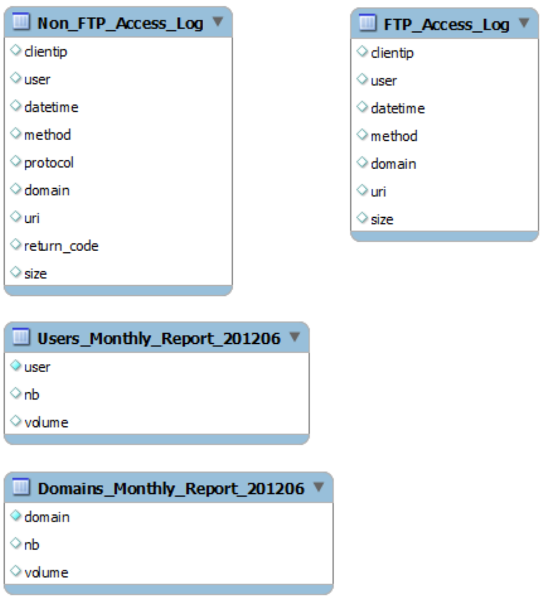
For our comparative tests I have defined the following MySQL tables:

Access Logs

The FTP connections are stored in a different table that the Non FTP (mostly HTTP)

Reports by month

Two totals are stored each month per domain and user: number of access (hits) and volume in bytes downloaded. A report is made by month, what means **we have for each month two tables: one with the users information and a second one with domains information**.



3. Defining a schema for MongoDB NoSQL

We have the following elements to manage:

- Users
- Internet domains
- The access non FTP (mainly HTTP)
- The access using FTP

In MongoDB data is grouped into “collections” (equivalent to tables) and each element of data is called a “document” (equivalent to records). **Unlike relational databases each document could have a different number of fields, and also contain other documents**.

In MongoDB is not necessary to define the fields and the type of each field. The collection, if needed, and the structure of a document is created by the server at the time of saving the data.

We will work with the following collections:

- Two collections for access log, one for each access log table
- Totals (hit number and data transferred) will be calculated in real time using “MongoDB functions”[120]. For each month we will have we will have two collections: one by user and a second one by domain (so in a year we will have 24 collections with aggregation data)

Therefore, the structure of each collection is (shown in pseudocode) ...

NON FTP Connections Log

```
{
    "userip": string,
    "user": string,
    "datetime": Date,
    "method": string,
    "protocol": string,
    "domain": string,
    "uri": string,
    "return_code": integer,
    "size": integer
}
```

FTP Connections Log

```
{
    "userip": string,
    "user": string,
    "datetime": Date,
    "method": string,
    "domain": string,
    "uri": string,
    "size": integer
}
```

120 http://www.mongodb.org/display/DOCS/Updating

Totals calculated by user

As said, for monthly reports we will work with two collections: a collection for users and one for the domains, with totals by month and day.

For each year and month we will have a collection

Each collection will have one document per user. In addition to the user identifier, t**he number of visits and volume of bytes transferred will be stored by month and by day**. Daily totals will be stored as a subdocument (within the document corresponding to the user).

These totals will be updated in real time, as log data is being processed. That is, each time information is received from a visit a new record will be created in the log (FTP or FTP), the number of visits will be incremented by one and the size transferred will be added the total volume for the day and user in the collection of the corresponding month. There will be totals by month that will be updated too.

```
{
    "_id": "Userid"
    "Nb": integer,
    "Volume": integer,
    "Daily": {
        "0": {
            "Nb": integer,
            "Volume": integer
        },
        "1": {
            "Nb": integer,
            "Volume": integer
        },
        "2": {
            "Nb": integer,
            "Volume": integer
        },
        "3": {
            "Nb": integer,
            "Volume": integer
        },
        ....
        "30": {
            "Nb": integer,
            "Volume": integer
        },
    },
}
```

Totals calculated by Domain

This will work exactly as in users but each document will correspond to a domain instead of a user name.

4. Interesting articles and presentations

To understand how to design a NoSQL schema the following presentations have been very useful:

- "Real-Time Analytics Schema Design and Optimization", by Ryan Nitz *[69]*
- "MongoDB for Analytics", by John Nunemaker *[70]*
- "Real Time Analytics with MongoDB Webinar", by Jared Rosoff *[71]*

From a broader perspective, considering MongoDB architecture that would form part, following two links are also interesting:

- "Real-Time Log Collection with Fluentd and MongoDB", article from "Treasure Data" company[121], telling how to use Fluentd[122] for real-time processing of server logs and storing in MongoDB *[72]*
- "Social Data and Log Analysis Using MongoDB", by Takahiro Inoue, telling the architecture deployed for a social game company using MongoDB, Hadoop and Cassandra *[73]*

121 http://treasure-data.com/

122 http://fluentd.org/

Comparative of a MySQL based solution versus MongoDB

1. Work plan for the performance tests

Now that we have the following elements ready:

- A package for the installation of MySQL 5.0.26
- A package for the installation of MongoDB 2.2.0
- A schema design for MySQL and the equivalent for MongoDB

It's time to make tests with data. As **we can't use real data for confidentiality concerns** we will have to:

- Develop code to fill with **“fake but realistic”** data the MySQL and MongoDB databases. This code will have classes with a common interface that will allow applications to use data from MySQL or MongoDB with no code changes
- Define a battery of tests to compare the performance of both solutions
- Make the tests and obtain conclusions

2. PHP Development for testing the database model

The first thing to do is to fill the databases with a big volume of realistic data. So initially I have developed some code to **create a volume of millions of log entries**. For our tests we have created:

- 70.000 visiting users
- 70.000 visiting IP's
- 1.300.000 visited Internet domains
- 90.000.000 of Non FTP log entries
- 4.500.000 of FTP log entries

For creating each log entry we will have to generate random data for the different elements: “Internet domain”, “FTP method”, “HTTP method”, “IP”, “Protocol”, “Return code”, “Size” and “User”.

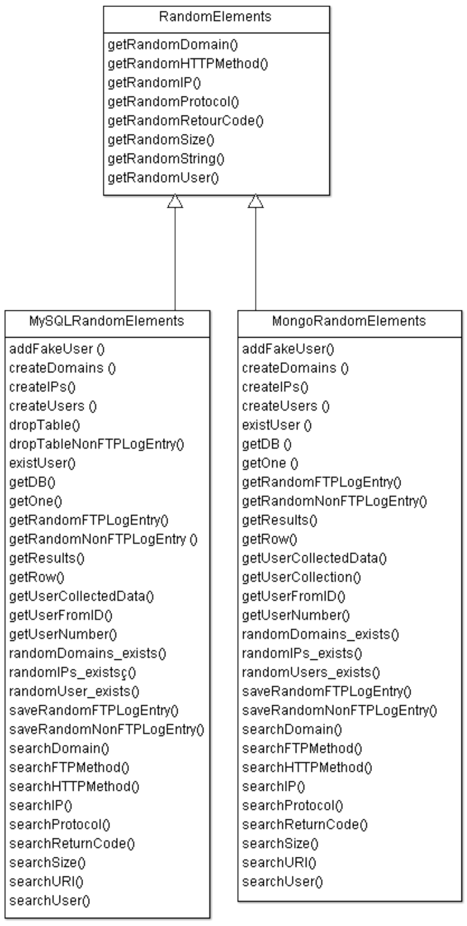
So, I have developed three PHP classes:

- **“RandomElements” class**: with functions like getRandomDomain(), getRandomFTPMethod() … which are used to generate the random elements of data
- **“MongoRandomElements” and “MySQLRandomElements” classes**, which are children classes of the previous one and have added functions to work with each database management system. They have functions to:
    - Save a random user in the database
    - Create lists of random domains, IPs and users and save them in tables/collections
    - Delete tables/collections

- Verify if a user exists in the list of random users
- Send a query that returns one single data and return it
- Send a query that returns a group of records and return them as an array
- Get the number of users created
- Get one random user/domain/IP from the list of created users/domains/IPs
- Create a new FTP log entry getting the (random) elements needed and save it into the database
- Create a new non FTP log entry getting the (random) elements needed and save it into the database
- …

The interface for this two classes is the same, so they can be used with the same code, making the scripts which use them agnostic to the type of database used

The UML diagram of this classes will be:



Use code example

An example of using this classes could be the following PHP code that, starting from an empty database, creates 70.000 random users, 70.000 random IPs, 1.300.000 random domains and after having this elements, generate 30 millions of random log entries for the month of April and save them in the collection for Non FTP log entries:

```
$mre = new MongoRandomElements();
$mre->createUsers(70000);
$mre->createIPs(70000);
$mre->createDomains(1300000);
// Example data for April
$start = mktime(0,0,0,4,1,2012);
$end = mktime(23,59,0,4,30,2012);
for ($i = 0; $i < 30000000; $i++)   {
    $log = $mre->getRandomNonFTPLogEntry($start, $end);
    $mre->saveRandomNonFTPLogEntry($log);
}
for ($i = 0; $i < 1500000; $i++)    {
    $log = $mre->getRandomFTPLogEntry($start, $end);
    $mre->saveRandomFTPLogEntry($log);
}
```

The code for making the same operations on MySQL is exactly de same except we should change the first line

```
$mre = new MongoRandomElements();
```

by

```
$mre = new MySQLRandomElements();
```

Code source available at Github and ciges.net

The source code and the tests scripts who use this classes are available at **"Ciges / internet_access_control_demo" on Github**[123]. Also it has been documented with **phpDocumentor**[124] and it's available at my web page **www.ciges.net**[125]

3. Testing MongoDB vs MySQL performance

As said in the first chapter the tests will be realised in one machine with the following hardware specifications:

- RAM: 10 GB
- Cores: 32 (AMD Opteron Processor 6128)

The comparative will be made between

- **MongoDB 2.2.0** (and 2.2.0-rc0, the tests had begun before the final stable version has been made available)
- **MySQL 5.0.26** (with MyISAM tables)

The objective is to have an objective measure of performance of both solutions for a list of equivalent tests. The tests developed are grouped in:

- Insertion tests
- Multiuser concurrent read tests

123 https://github.com/Ciges/internet_access_control_demo

124 http://www.phpdoc.org/

125 http://www.ciges.net/phpdoc/iacd/

- Multiuser concurrent write tests
- Complex (aggregation) queries read tests

A script (PHP, JavaScript or SQL) has been done for each one. This scripts are run with the Unix command “time” to measure the time taken by each one.

Each test has been repeated three (or more) times to discard abnormal results.

4. Insertion tests

The list of insertion tests is the following:

1. Generation and saving of 70.000 random users without using indexes and allowing repeated values
2. Generation and saving of 70.000 random IPs without using indexes and allowing repeated values
3. Generation and saving of 1.300.000 random domains without using indexes and allowing repeated values
4. Generation and saving of 70.000 random users using indexes and verifying (sending a read query) that the user does not exists before sending the save command
5. Generation and saving of 70.000 random IPs using indexes and verifying (sending a read query) that the IP does not exists before sending the save command
6. Generation and saving of 1.300.000 random domains using indexes and verifying (sending a read query) that the user does not exists before sending the save command
7. Generation and saving of 1 million of non FTP log entries
8. Generation and saving of 5 millions of non FTP log entries
9. Generation and saving of 10 millions of non FTP log entries
10. Generation and saving of 30 millions of non FTP log entries

Insertion tests results

Results given are the average of the different results discarding extreme values.

	MongoDB	**MySQL**
70.000 users	3s	12s
70.000 IPs	3s	12s
1.300.000 domains	58s	4m 36s
70.000 unique users with indexes	23s	28s
70.000 unique IPs with indexes	22s	31s
1.300.000 unique domains with indexes	8m27s	14m13s
1.000.000 log entries	12m7s	26m14s
5.000.000 log entries	1h03m53s	2h10m54s
10.000.000 log entries	1h59m11s	3h27m10s
30.000.000 log entries	5h55m25s	10h18m46s

5. Multi user concurrent tests

The previous insertion tests are coded as a loop which makes an insertion one after the other. This means that there will be only one query at a time.

For the following tests instead of making a loop (it makes little sense for reading tests) I have used the open source tool **JMeter**[126] **with the plugin Stepping Thread Group**[127] to simulate concurrent users.

JMeter is a powerful tool that allows to simulate use cases and loads with virtual users to measure the performance of a web application.

I will simulate virtual users that will access simultaneously to a collection of scripts which make simple read and write operations. This scripts are PHP scripts which will be made available via web.

The tests are composed by six scripts, which will perform the following three tests for MongoDB and for MySQL:

- **Search and show data for a random user (read test)**
- **Add a random user (write test)**
- **Search and show data for a random user or add a random user (read & write test, 80% of times will read and 20% of times will write)**

We will simulate two scenarios:

- **An incrementing load from 0 to 50 users rising by five**. This load will be kept for a few minutes
- **A load of 50 users sending all queries from the beginning**. This load will be kept also for a few minutes before stopping.

The list of tests configured (two times, one for MongoDB and another for MySQL) is the following

1. Concurrent reads, incrementing users from 0 to 50
2. Concurrent reads, 50 users
3. Concurrent writes, incrementing users from 0 to 50
4. Concurrent writes, 50 users
5. Concurrent reads (80%) & writes (20%), incrementing users from 0 to 50
6. Concurrent reads (80%) & writes (20%), 50 users

Each one of this tests have been made three or more times, stopping and starting the server before.

For each one we will get:

- Number of queries sent (value samples)
- Statistical values for **response time** (in milliseconds): average, median, minimum, maximum, standard deviation
- Percentage of errors
- Throughput in queries/second

126 http://jmeter.apache.org/

127 http://code.google.com/p/jmeter-plugins/wiki/SteppingThreadGroup

- KBytes/second received and average bytes per query
- A CSV file with the response time for all the queries, which I will use to made a graphical representation

For the generation of the graphics I have

- Reduced the number of values (and the impact of aberrant ones) obtained getting the mean grouped by second (155 values for MongoDB and the same number for MySQL)
- Represent a linear regression with the R function "loess"[128]

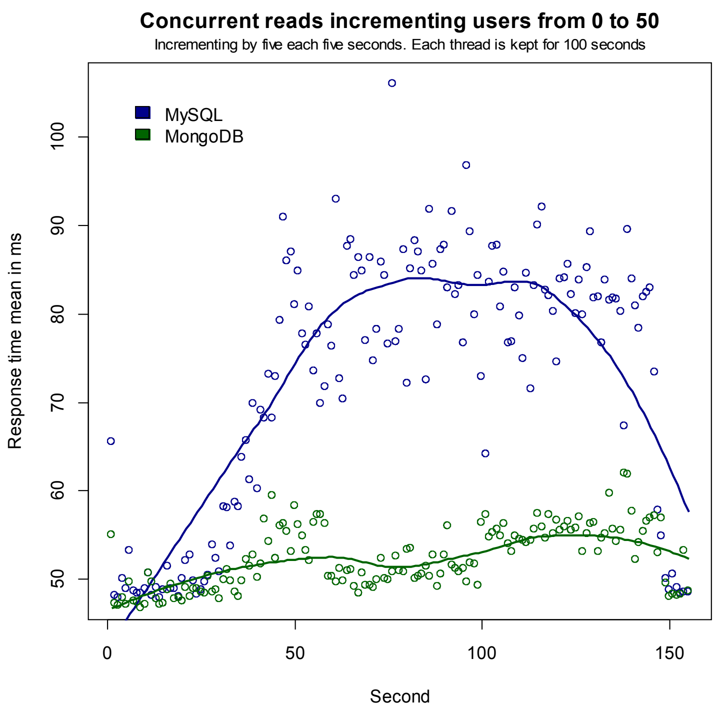
The **JMeter configuration**, **CSV files with the samples results** and **R scripts** are all available in the Github at **"**Ciges / internet_access_control_demo" on Github[129]

Concurrent read tests results

*Concurrent **reads**, incrementing users from 0 to 50 (each thread will be kept for 100 seconds)*

	Samples	Med	Min	Max	Std. Dev.	Throughput	KB/sec
MongoDB	112.165	**48ms**	43ms	3.090ms	18,72ms	728,3 q/s	184,99 kb/s
MySQL	81.179	**67ms**	45ms	3.140ms	40,92ms	528 q/s	134,08 kb/s

Graphically represented this load test will be



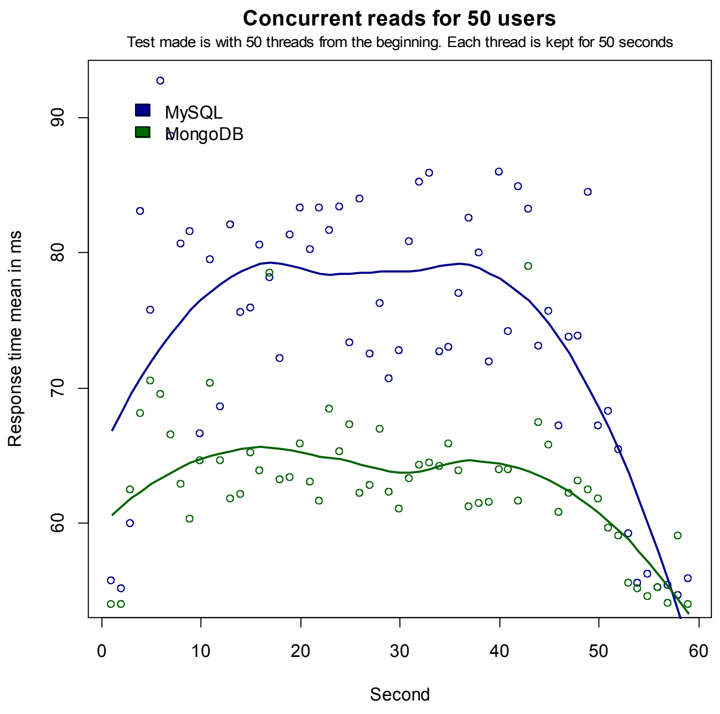
128 http://stat.ethz.ch/R-manual/R-patched/library/stats/html/loess.html

129 https://github.com/Ciges/internet_access_control_demo

*Concurrent **reads**, 50 users(each thread will be kept for 50 seconds)*

	Samples	Med	Min	Max	Std. Dev.	Throughput	KB/sec
MongoDB	37.497	**54ms**	50ms	5.419ms	161,04ms	635,4 q/s	122,88 kb/s
MySQL	32.273	**62ms**	52ms	5.136ms	156,90ms	547,9 q/s	114,54 kb/s

Graphically represented this load test will be

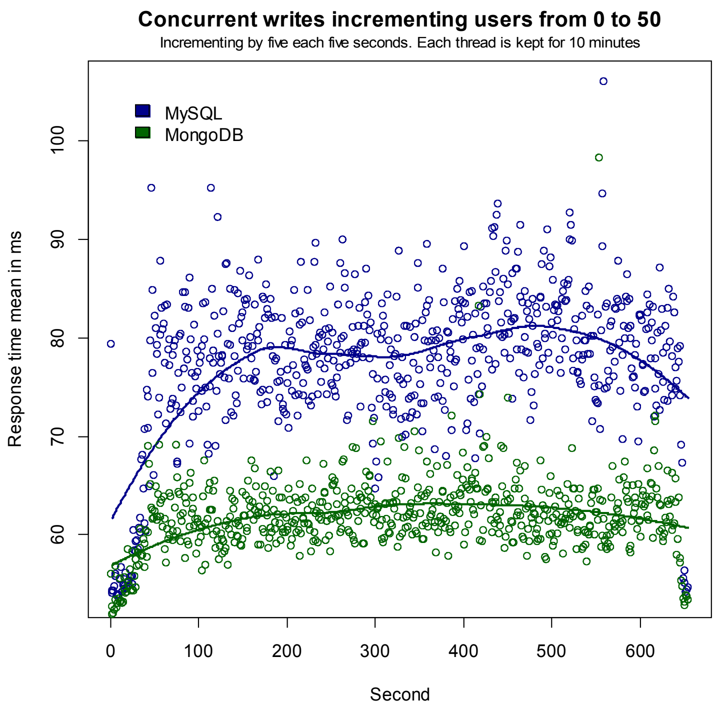


Concurrent writes tests results

*Concurrent **writes**, incrementing users from 0 to 50 (each thread will be kept for 10 minutes)*

	Samples	Med	Min	Max	Std. Dev.	Throughput	KB/sec
MongoDB	464.853	**54ms**	49ms	3.105ms	27,71ms	710,9 q/s	148,67 kb/s
MySQL	383.700	**70ms**	51ms	4.105ms	25,58ms	586,7 q/s	122,64 kb/s

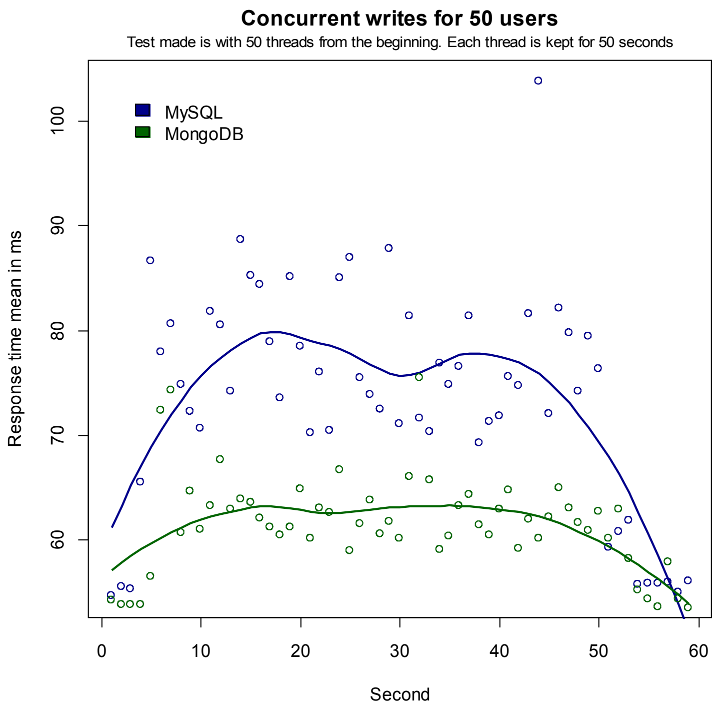
Graphically represented this load test will be



Concurrent ***writes****, 50 users (each thread will be kept for 50 seconds)*

	Samples	Med	Min	Max	Std. Dev.	Throughput	KB/sec
MongoDB	37.497	**54ms**	50ms	5.419ms	161,04ms	635,4 q/s	132,88 kb/s
MySQL	32.273	**62ms**	52ms	5.136ms	156,90ms	547,9 q/s	114,54 kb/s

Graphically represented this load test will be

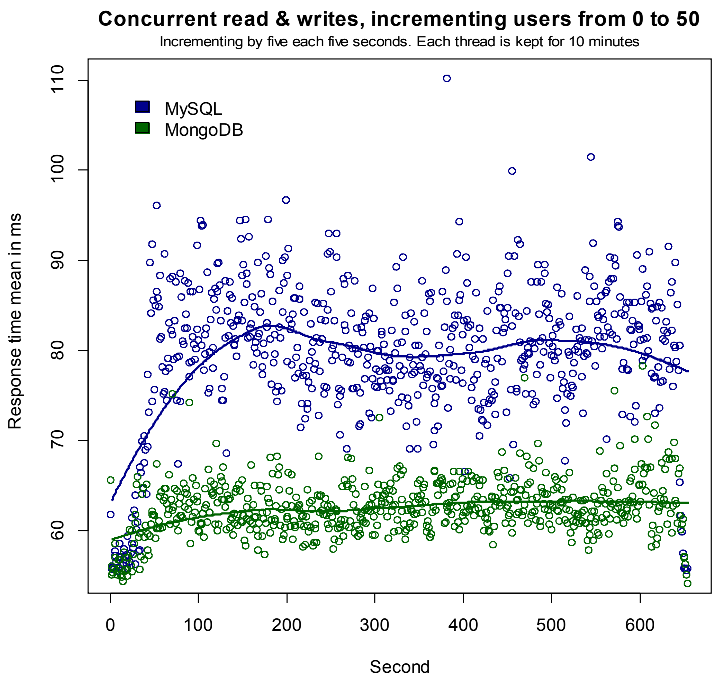


Concurrent reads & writes tests results

Concurrent ***read & writes****, incrementing users from 0 to 50 (each thread will be kept for 10 minutes)*

	Samples	Med	Min	Max	Std. Dev.	Throughput	KB/sec
MongoDB	462.740	**55ms**	48ms	3.111ms	26,24ms	707,7 q/s	173,45 kb/s
MySQL	373.484	**71ms**	52ms	3.152ms	27,98ms	571,1 q/s	139,91 kb/s

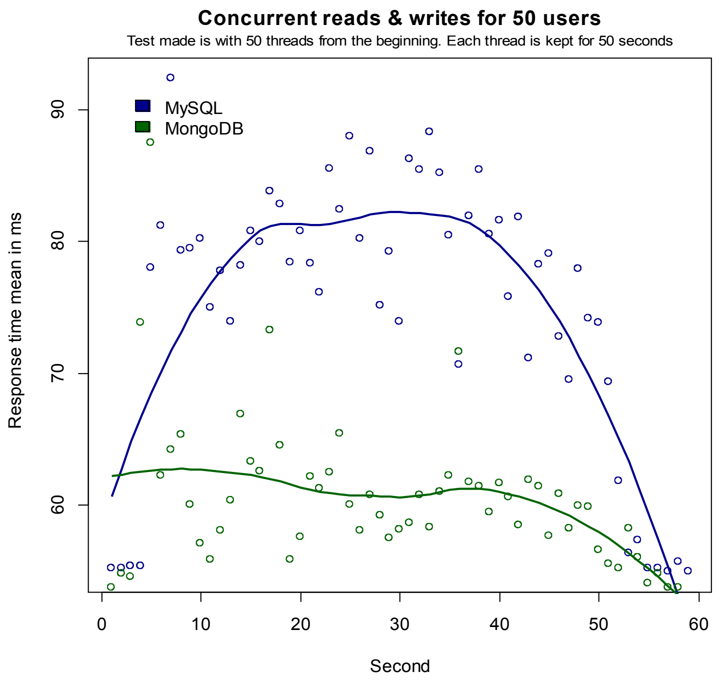
Graphically represented this load test will be



*Concurrent **read & writes**, 50 users (each thread will be kept for 50 seconds)*

	Samples	Med	Min	Max	Std. Dev.	Throughput	KB/sec
MongoDB	39.096	**54ms**	50ms	4.753ms	142,72ms	665,0 q/s	162,93 kb/s
MySQL	31.272	**64ms**	52ms	5.964ms	176,64ms	530,0 q/s	129,8 kb/s

Graphically represented this load test will be



6. Data analyse (aggregation) read tests

This tests are made to compare the aggregation capabilities of both database management systems. I have designed complex queries that will read all the data (90 millions of log records) and obtain different results. For MongoDB I have used the aggregation framework (simpler than Map Reduce functions and enough if we don't need to get a large list of results).

The queries tested are the following:

- Which are the 10 most visited domains and how many visits has each one?
- Which are the 10 most visited domains in the second half of June?
- Which are the 10 users that have more Internet accesses?
- What is the average Internet traffic for June?

In the real world (and with database that could have terabytes) this type of questions would be calculated in real time updating collections created for storing the results (as shown in chapter "NoSQL Schema Design for Internet Access Logs") **or with batch scripts**.

For comparing the performance between MongoDB and MySQL we will compare the time taken for each one which the Unix command "time" as with the insertion tests. Also each test will be repeated three or more times. Results given are the average of the different results discarding extreme values.

Aggregation read tests results

	MongoDB	MySQL
10 most visited domains with visit totals	13m13s	2m37s
10 most visited domains in the second half of June	52m39s	17m43s
10 users with more Internet accesses	24m02s	3m53s
Average Internet traffic for June	12m05s	2m42s

7. Aggregation read tests code

I think it's interesting to show the code used for the aggregation scripts in MongoDB and MySQL. MySQL part is SQL, so it will be familiar to most readers, MongoDB part uses the aggregation framework. The code is mostly PHP code except one of the tests where I have used JavaScript for MongoDB and a SQL script for MySQL.

Which are the 10 most visited domains and how many visits has each one?

MongoDB (JavaScript)

```
db.NonFTP_Access_log.aggregate(
    { $group: {
        _id: "$domain",
        visits: { $sum: 1 }
        }},
    { $sort: { visits: -1 } },
    { $limit: 10 }
    ).result.forEach(printjson
```

MySQL (SQL script)

```
drop table if exists NonFTP_Access_log_domain_visits;
create table NonFTP_Access_log_domain_visits (
    `domain` varchar(255) NOT NULL,
    `value` int unsigned not null,
    PRIMARY KEY  (`domain`),
    KEY `value_index` (`value`)
    ) ENGINE=MyISAM DEFAULT CHARSET=utf8
    select domain, count(*) as value from NonFTP_Access_log group by domain;
select * from NonFTP_Access_log_domain_visits order by value desc limit 10;
```

Which are the 10 most visited domains in the second half of June?

MongoDB (PHP code)

```
$mre = new MongoRandomElements("mongodb", "mongodb", "localhost", "InternetAccessLog");
// First, we get the minimum value of the 10 highest visits per domain
$start = new MongoDate(strtotime("2012-06-15 00:00:00"));
$end = new MongoDate(strtotime("2012-06-30 23:59:59"));
$min_value  = $mre->getOne(array(
            array('$match' => array('datetime' => array( '$gt' => $start, '$lt' =>
$end ))),
            array('$group' => array('_id' => '$domain', 'visits' => array( '$sum' => 1 ))),
            array('$group' => array('_id' => '$visits')),
            array('$sort' => array('_id' => -1)),
            array('$limit' => 10),
            array('$sort' => array('_id' => 1)),
            array('$limit' => 1),
        ), "NonFTP_Access_log");

// Now, we obtain all the domains with at lest that value
$data = $mre->getResults(array(
            array('$match' => array('datetime' => array( '$gt' => $start, '$lt' =>
$end ))),
            array('$group' => array('_id' => '$domain', 'visits' => array( '$sum' => 1 ))),
            array('$match' => array('visits' => array( '$gte' => $min_value)))
        ), "NonFTP_Access_log");

foreach($data as $doc)    {
    print_r($doc);
    }
```

MySQL (SQL code)

```
$mre = new MySQLRandomElements("mysqldb", "mysqldb", "localhost", "InternetAccessLog");

// First, we get the minimum value of the 10 highest visits per domain
$start = "2012-06-15 00:00:00";
$end = "2012-06-30 23:59:59";
$query = "select * from (select distinct(count(*)) as visits from NonFTP_Access_log where
datetime between \"".$start."\" and \"".$end."\" group by domain order by visits desc limit
10) as topten_visits_by_domain order by visits limit 1";
$min_value = $mre->getOne($query);
```

```
// Now, we obtain all the domains with at lest that value
$query = "select * from (select domain, count(*) as visits from NonFTP_Access_log where
datetime between \"".$start."\" and \"".$end."\" group by domain) as visits_by_domain where
visits >= ".$min_value;

$results = $mre->getResults($query);
while($row = $results->fetch_assoc())   {
    print_r($row);
    }
```

Which are the 10 users that have more Internet accesses?

MongoDB (PHP code)

```
$mre = new MongoRandomElements("mongodb", "mongodb", "localhost", "InternetAccessLog");

// First, we get the minimum value of the 10 highest visits per user
$min_value  = $mre->getOne(array(
    array('$group' => array('_id' => '$user', 'visits' => array( '$sum' => 1 ))),
    array('$group' => array('_id' => '$visits')),
    array('$sort' => array('_id' => -1)),
    array('$limit' => 10),
    array('$sort' => array('_id' => 1)),
    array('$limit' => 1),
    ), "NonFTP_Access_log");

// Now, we obtain all the users with at least that value
$data = $mre->getResults(array(
    array('$group' => array('_id' => '$user', 'visits' => array( '$sum' => 1 ))),
    array('$match' => array('visits' => array( '$gte' => $min_value)))
    ), "NonFTP_Access_log");

foreach($data as $doc)    {
    print_r($doc);
    }
```

MySQL (SQL code)

```
$mre = new MySQLRandomElements("mysqldb", "mysqldb", "localhost", "InternetAccessLog");

// First, we get the minimum value of the 10 highest visits per user
$query = "select * from (select distinct(count(*)) as visits from NonFTP_Access_log group
by user order by visits desc limit 10) as topten_visits_by_user order by visits limit 1";
$min_value = $mre->getOne($query);

// Now, we obtain all the users with at least that value
$query = "select * from (select user, count(*) as visits from NonFTP_Access_log group by
user) as visits_by_user where visits >= ".$min_value;
$results = $mre->getResults($query);
while($row = $results->fetch_assoc())   {
    print_r($row);
    }
```

What is the average Internet traffic for June?

MongoDB (PHP code)

```
$mre = new MongoRandomElements("mongodb", "mongodb", "localhost", "InternetAccessLog");
$start = new MongoDate(strtotime("2012-06-01 00:00:00"));
$end = new MongoDate(strtotime("2012-06-30 23:59:59"));

$result = round($mre->getOne(array(
    array('$match' => array('datetime' => array( '$gte' => $start, '$lte' => $end ))),
    array('$project' => array('_id' => 0, 'day' => array ( '$dayOfMonth' => '$datetime' ),
'size' => 1)),
    array('$group' => array('_id' => '$day', 'volume' => array( '$sum' => '$size'))),
    array('$group' => array('_id' => 'all', 'average' => array( '$avg' => '$volume'))),
    array('$project' => array('_id' => '$average'))
    ), "NonFTP_Access_log"));

printf("Traffic volume mean by day in bytes for June: %.0f\n", $result);
```

MySQL (SQL code)

```
$mre = new MySQLRandomElements("mysqldb", "mysqldb", "localhost", "InternetAccessLog");

$start = "2012-06-01 00:00:00";
$end = "2012-06-30 23:59:59";
$query="select round(avg(volume)) from (select sum(size) as volume from NonFTP_Access_log
where datetime between \"".$start."\" and \"".$end."\" group by dayofmonth(datetime)) as
sizebyday";
$result = $mre->getOne($query);

printf("Traffic volume mean by day in bytes for June: %.0f\n", $result);
```

8. How to run this tests

Database and users creation

In the Github repository for " Ciges / internet_access_control_demo"[130] there is a collection of scripts to run the tests.

This scripts use the following database names and users by default:

- For MySQL: database "InternetAccessLog", with user and password "mysqldb"
- For MongoDB: collection " InternetAccessLog", with user and password "mongodb"

So before starting we will have to create both and give the permissions:

Creating database and user in MySQL

```
mysql> create database InternetAccessLog;
mysql> grant all privileges on InternetAccessLog.* to mysqldb@localhost identified by
'mysqldb';
```

130 https://github.com/Ciges/internet_access_control_demo

Creating collection and user in MongoDB

```
> use InternetAccessLog
> db.addUser("mongodb", "mongodb")
```

Generating random data

The following PHP scripts create 3 months of random data as explained before:

- “createData_3months_mongo.php”
- “createData_3months_mysql.php”

To run them we can use the **console PHP interpreter** with

```
php -c path/to/my/php.ini phpscript
```

List of runnable scripts

As with the data generation script, to run them on use the **console PHP interpreter.** There are two PHP scripts per test: one for MySQL and the second one for MongoDB. The relation of scripts is the following:

Scripts	Test
test1_1_mongo.php, test1_1_mysql.php	Generation and saving of 70.000 random users without using indexes and allowing repeated values
test1_2_mongo.php, test1_2_mysql.php	Generation and saving of 70.000 random IPs without using indexes and allowing repeated values
test1_3_mongo.php, test1_3_mysql.php	Generation and saving of 1.300.000 random domains without using indexes and allowing repeated values
test2_1_mongo.php, test2_1_mysql.php	Generation and saving of 70.000 random users using indexes and verifying (sending a read query) that the user does not exists before sending the save command
test2_2_mongo.php, test2_2_mysql.php	Generation and saving of 70.000 random IPs using indexes and verifying (sending a read query) that the IP does not exists before sending the save command
test2_3_mongo.php, test2_3_mysql.php	Generation and saving of 1.300.000 random domains using indexes and verifying (sending a read query) that the domain does not exists before sending the save command
test3_1_mongo.php, test3_1_mysql.php	Generation and saving of 1 million of non FTP log entries
test3_2_mongo.php, test3_2_mysql.php	Generation and saving of 5 millions of non FTP log entries
test3_3_mongo.php, test3_3_mysql.php	Generation and saving of 10 millions of non FTP log entries
test3_4_mongo.php, test3_4_mysql.php	Generation and saving of 30 millions of non FTP log entries
test9_1_mongo.php, test9_1_mysql.php	Analyse query: Gets the 10 domains most visited and the number of visits for each one
test9_2_mongo.php, test9_2_mysql.php	Analyse query: Gets the 10 domains most visited in the second half of June and the number of visits for each one
test9_3_mongo.php, test9_3_mysql.php	Analyse query: Gets the 10 users with most hits
test9_4_mongo.php, test9_4_mysql.php	Analyse query: Gets the mean by day for traffic volume in June

Multi-user concurrent tests

This scripts, under the “web” directory, are though to we hosted in a web server. One we have configured our web server to made then available I have used:

- **Apache JMeter**[131] with **the plugin “Stepping Thread Group”**[132] to run the load tests

131 http://jmeter.apache.org/

132 http://code.google.com/p/jmeter-plugins/wiki/SteppingThreadGroup

- **R**[133] to create graphical representation from CSV files with the data created with JMeter

The scripts available under web directory[134] are:

Scripts	Function	Test
test4_mongo.php, test4_mysql.php	Search and show data for a random user	Concurrent reads
test5_mongo.php, test5_mysql.php	Write a random user	Concurrent writes
test6_mongo.php, test6_mysql.php	MongoDB read/write test. This scripts makes one of two actions: Search and show data for a random user (read test) or Write a new random user in the database (write test). The read test is made 80% of times, the write one the 20%.	Concurrent reads & writes

Once the web server configured if we accede to the URL corresponding to the directory we should see a description message with links to the different scripts

Using JMeter to run load tests

As shown before we have defined **two scenarios for each test and three types of tests**. Then we have six different tests:

- Concurrent reads, incrementing users from 0 to 50
- Concurrent reads, 50 users
- Concurrent writes, incrementing users from 0 to 50
- Concurrent writes, 50 users
- Concurrent reads (80%) & writes (20%), incrementing users from 0 to 50
- Concurrent reads (80%) & writes (20%), 50 users

In the file **“MongoDB vs MySQL.jmx”** there is all the configuration needed for JMeter

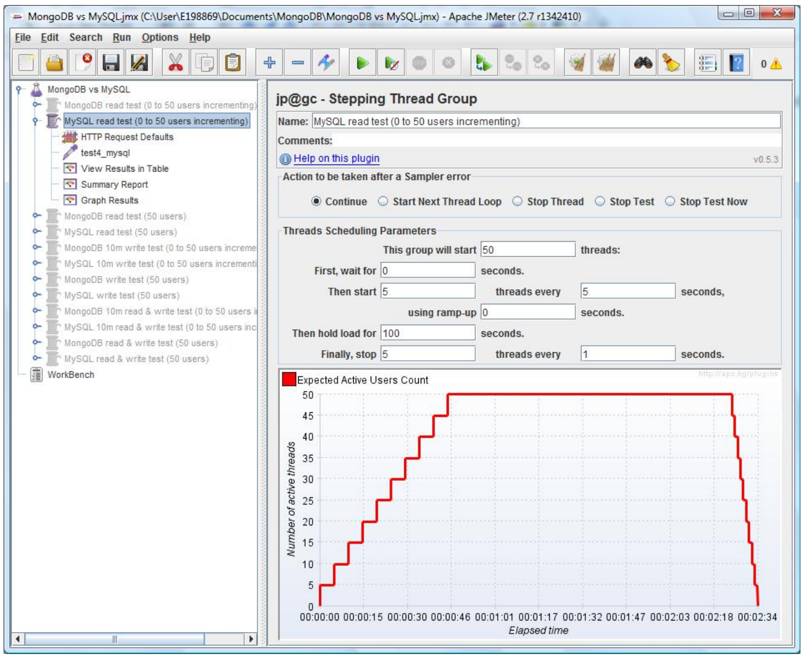
To run each tests we should

- Change the URL of the server to our address
- In “View Results in Table” change the path where the CSV file should be saved
- Enable in JMeter only the test we want to run, disabling the rest (if not more than one test will be run)

133 http://www.r-project.org/

134 https://github.com/Ciges/internet_access_control_demo/tree/master/web

Example of incrementing user configuration with JMeter



MongoDB vs MySQL.jmx (C:\User\E198869\Documents\MongoDB\MongoDB vs MySQL.jmx) - Apache JMeter (2.7 r1342410)

- MongoDB vs MySQL
  - MongoDB read test (0 to 50 users incrementing)
  - MySQL read test (0 to 50 users incrementing)
    - HTTP Request Defaults
    - test4_mysql
    - View Results in Table
    - Summary Report
    - Graph Results
  - MongoDB read test (50 users)
  - MySQL read test (50 users)
  - MongoDB 10m write test (0 to 50 users increme
  - MySQL 10m write test (0 to 50 users incrementi
  - MongoDB write test (50 users)
  - MySQL write test (50 users)
  - MongoDB 10m read & write test (0 to 50 users i
  - MySQL 10m read & write test (0 to 50 users inc
  - MongoDB read & write test (50 users)
  - MySQL read & write test (50 users)
- WorkBench

jp@gc - Stepping Thread Group

Name: MySQL read test (0 to 50 users incrementing)

Comments:

Help on this plugin v0.5.3

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Threads Scheduling Parameters

This group will start 50 threads:
First, wait for 0 seconds.
Then start 5 threads every 5 seconds,
using ramp-up 0 seconds.
Then hold load for 100 seconds.
Finally, stop 5 threads every 1 seconds.

Getting a graphical representation of load tests with R

Each load test will generate **tens of thousands of samples that will be stored in CSV files**. We have **two files for each tests type**, one with MySQL response times and the second one with MongoDB response time.

For each test type I have developed a R script that reads this two files and represents graphically a summary of samples and draws a line that shows response time evolution for both types of servers.

This scripts are available also in the web directory and for running them you have simply to use the command source. Their names are self explanatory. If we had six tests we have then six R scripts, one for showing the comparative results of each one.

To load in R and show the graphic you have simply to load the script. By example for loading the first one:

```
source("Concurrent reads 50 users.R")
```

Conclusions and last words

1. Tests conclusions

Looking at the numbers and graphics we arrive to the following conclusions:

Write performance:

- MongoDB is faster in pure write performance
- In a series of continuous simple writings MongoDB is from 2 to 4 times faster. **In general, for high numbers (millions of record savings) simple writing performance is the double of MySQL**
- In **concurrent writes MongoDB is faster (15% and 30%** in our tests)
- MongoDB is much more scalable, meaning that **when the user load increases the response time keeps stable**. Response time in MySQL, instead, gets worse as the number of users grows

Read performance:

- MongoDB is faster in pure read performance
- In **concurrent reads MongoDB is faster (15% and 40%** in our tests)
- Also, MongoDB is more scalable

Aggregation performance:

- Here MySQL wins over MongoDB's aggregation native framework. **MySQL is much faster in aggregating data, 3 to 6 times faster** for the 4 tests we have done
- In this aggregation queries no relations are involved. MySQL GROUP BY queries have a very high performance

So we could say that, as waited, **for intensive reading and writing data operations MongoDB is a better option that MySQL** when no relations nor aggregation queries performance are important and the data reading/writing performance is critical.

Need to say that aggregation queries on ten of millions of records is not a good idea, **it would be better to calculate in real time values needed as records are processed** (what means read & write operations). **So for problems as log analyse NoSQL technologies are much better**.

2. Initial planning and actual time spent on each task

The initial work charge estimated was 300 hours. The actual time spent on each has been of more of 400 hours divided as follows:

Tasks	Time spent (rounded to hours)
Study of NoSQL articles & books	65
MongoDB installation, configuration, package creation & updates	58
Development of a schema for Internet Access Log	20
Scripts development (PHP, SQL, JavaScript, MySQL stored procedures)	68
Load tests	75
Documentation (memory, posts on ciges.net & presentation)	93
Incidents analyse & resolution	18
Planning, coordination & communication	43
Total	***440***

To keep track of time spent on each task I have used the web application **Paymo**[135], an easy to use time tracking software that allow to create projects and tasks and comfortably start/stop timers for each one.

Also initially was planned to make also tests with sharding capabilities of MongoDB (using more than one machine), but due to lack of time we will do them later.

3. Problems and bugs found

The reasons of the time excess regarding to was initial planned are:

- At first I developed a script to import real data for production server into MongoDB for the tests. But using real data is not allowed, then I could not use it
- **We have used three versions of MongoDB**. The MongoDB study has been started with version 2.0.6. Meanwhile 2.2.0 “release candidate 0” and 2.2.0 final have been published .

  We have upgraded the version due to with 2.2.0 a native aggregation framework is available, and it's easier and more performing than using Map-Reduce to aggregate data.

  Also the second version change has been forced due to a bug found on 2.2.0rc0 that produced and integer overflow on some aggregation functions[136]
- Initially I have tried to make some load tests scripts using directly JavaScript for MongoDB and stored procedures for MySQL. Also I have tried to use map-reduce. **Both initiatives, using native supported JavaScript /stored procedures and map-reduce were wrong.**

  The time invested in learning how to develop with this technologies has been useless due to limitations on JavaScript MongoDB API. Also MySQL stored procedures developing was more complex than thought, so at last I have used PHP for most of the scripts

  Map-reduce functionality included by default in MongoDB is terribly slow[137] and not useful. MongoDB's aggregation framework was a valid option and the one chosen for data analyse tests.
- Due to version change and some errors made when running load tests **I had to repeat the tests battery two or three times**
- We have found some **configurations problems and product's bugs**

135 http://www.paymo.biz/

136 https://jira.mongodb.org/browse/SERVER-6166

137 http://stackoverflow.com/questions/12139149/mapreduce-with-mongodb-really-really-slow-30-hours-vs-20-minutes-in-mysql-for

- **Documentation work time has been underestimated**

Bugs found on MongoDB

Apart of problems commented right now, while we were preparing and testing the MongoDB package for distribution in PSA's server we have found the following notable bugs in MongoDB or problems in our initial configuration:

Memory problems when inserting big amounts of data

When I began with the insertion tests I got the following errors after a few millions of records saved

```
ERROR:   mmap() failed for /users/mng00/instances/primary/data/local/local.4 len:2146435072
errno:12 Cannot allocate memory
ERROR: mmap failed with out of memory. (64 bit build)
assertion 10085 can't map file memory ns:local.system.replset query:{}
```

The problem here is that MongoDB was reserving memory as it was needed and it arrives a moment where the operating system does not allow the process to consume more memory.

After a lot of tests and consulting with my colleagues one of them show me the way to go. In our MongoDB starting script we have to tell Linux not to limit the amount of memory with the command:

```
ulimit -v unlimited
```

Integer overflow when using aggregation functions

When calculating the average of traffic volume the result was a negative number. It was obviously an overflow. After searching in MongoDB's JIRA[138] it is a known problem for the version 2.2.0rc0.

After upgrading the product to the stable 2.2.0 the problem was solved.

Map-Reduce operations on MongoDB are really, really slow

To get the number of visits by domains I tried to use map-reduce functions but I found they were terribly slow (30 hours vs 20 minutes in MySQL for the same kind of test).

After asking in MongoDB's JIRA and in Stack Overflow[139] I received quickly support from Adam Comerford[140], a technical support manager from 10gen (the enterprise who made MongoDB) who explained that it could be normal. **MongoDB uses the JavaScript engine “SpiderMonkey” to compute Map-Reduce functions, and JavaScript is slow and single threaded**.

We have three options for this kind of operations

- **Use the “Aggregation Framework”[141] included from MongoDB's version 2.1**. (this framework has the limitation of not being able of returning data of more than 16 Megabytes)
- **Use Apache Hadoop[142] to make map-reduce operations** with the “MongoDB Hadoop Connector”[143]. In this case MongoDB will be the database from where to read and save the data and Apache Hadoop will do the calculations

138 https://jira.mongodb.org/browse/SERVER-6166

139http://stackoverflow.com/questions/12139149/mapreduce-with-mongodb-really-really-slow-30-hours-vs-20-minutes-in-mysql-for

140 http://www.linkedin.com/in/acomerford

141 http://docs.mongodb.org/manual/applications/aggregation/

142 http://hadoop.apache.org/

143http://api.mongodb.org/hadoop/MongoDB+Hadoop+Connector.html

- Another option (to test) could be to **use Google's JavaScript engine V8** which can be integrated in MongoDB's compiling the product[144]. This engine is faster and multi-threaded.

For our test I have used the first possibility, the “Aggregation Framework” with an updated version of MongoDB.

4. Future work

This work is not really complete. In this project I have compared MongoDB with MySQL for a concrete use case and with a limited number of tests.

To complete this work the following should be done

- **Repeat the tests with a huge quantity of data** (hundreds of millions of records instead of only 90 millions, with a used disk size of hundreds of gigabytes instead of tens)
- Add tests with a multi-machine configuration using **sharding**

Also others future lines of work could be:

- Test map-reduce operations with V8 JavaScript engine
- Test map-reduce operations with Hadoop integration (well, being realistic MongoDB's and Hadoop's integration could be the subject for another work like the presented in this document).
- Add a few more aggregation tests

5. Contributions to the community

All this work, made as part of my paid work as system administrator at PSA, is intended to be publicly available. So, the contribution to the community is documentation and source code.

In particular, while this project was being made the following contributions have been done:

Contributions to the Wikipedia

- Rewriting of English wikipedia articles: “MongoDB”[145], “CouchDB”[146]
- Rewriting of French wikipedia article: “MongoDB”[147]
- Minor edition on other articles like English “CAP theorem”[148], “NoSQL”[149], “Textile (markup language)”[150], “Apache Cassandra”[151]

Creation of a personal blog http://www.ciges.net

- **Series of post with a summary of the work done**, problems found and solutions given:

144 http://www.mongodb.org/display/DOCS/Building+with+V8

145 http://en.wikipedia.org/wiki/MongoDB

146 http://en.wikipedia.org/wiki/CouchDB

147 http://fr.wikipedia.org/wiki/MongoDB

148 http://en.wikipedia.org/wiki/CAP_theorem

149 http://en.wikipedia.org/wiki/NoSQL

150 http://en.wikipedia.org/wiki/Textile_%28markup_language%29

151 http://en.wikipedia.org/wiki/Apache_Cassandra

  - "Buscando una solución NoSQL para el análisis de logs (1 de 4)"[152]
  - "Buscando una solución NoSQL para el análisis de logs (2 de 4)"[153]
  - "Buscando una solución NoSQL para el análisis de logs (3 de 4)"[154]
  - "Buscando una solución NoSQL para el análisis de logs (4 de 4)"[155]
  - "Instalando MongoDB + drivers en SUSE Linux SLES 10, algunos apuntes"[156]
  - "Esquema de datos NoSQL para el análisis de logs de acceso a Internet"[157]

All source code (PHP classes, scripts and configuration files)

- At **Github repository** "Ciges / internet_access_control_demo"[158]
- The **documentation created with phpDocumentor** is on my web page[159]

This document and detailed instructions to repeat the tests done are included in Github and in my web

Questions opened <u>and answered</u> on Stack Overflow (and also in MongoDB's JIRA)

- "Map Reduce with MongoDB really, really slow (30 hours vs 20 minutes in MySQL for an equivalent database)"[160]
- "Simple tool for web server benchmarking?"[161]
- "Simultaneous users for web load tests in JMeter?"[162]

The documentation (LibreOffice documents and posts on my web) have a Creative Commons Attribution – Share Alike 3.0 Unported license. The source code is licensed under the GPL version 3.0

152 http://www.ciges.net/buscando-nosql-1

153 http://www.ciges.net/buscando-nosql-2

154 http://www.ciges.net/buscando-nosql-3

155 http://www.ciges.net/buscando-nosql-4

156 http://www.ciges.net/apuntes-sobre-la-instalacion-de-mongodb

157 http://www.ciges.net/esquema-de-datos-nosql-para-el-analisis-de-logs-de-acceso-a-internet

158 https://github.com/Ciges/internet_access_control_demo

159 http://www.ciges.net/phpdoc/iacd/

160 http://stackoverflow.com/questions/12139149/mapreduce-with-mongodb-really-really-slow-30-hours-vs-20-minutes-in-mysql-for

161 http://stackoverflow.com/questions/12249895/simple-tool-for-web-server-benchmarking

162 http://stackoverflow.com/questions/12392644/simultaneous-users-for-web-load-tests-in-jmeter

6. Personal evaluation of the practicum

This work has been, from an administrator's system point of view, very interesting. What I have tried here is to apply the knowledge acquired in the "Master on Free Software Projects Development and Management"[163]

In particular I have considered very important:

- **The availability of (almost) all the documentation, code and configuration files** in an open license
- **The openness in all the process followed** to justify options chosen and to compare MongoDB and MySQL to allow anyone to repeat it
- **The use of Open Source products** (instead of proprietary ones used by default in my enterprise). I mean particularly
  - LibreOffice instead of Microsoft Office
  - Apache JMeter and R instead of HP LoadRunner

It's clear the real influence of the philosophy and technologies shown at the Master on Free Software on this work, which would be different if it had been simply another project to complete at work.

It has been also the first time I have used from the beginning tools to increase and measure productivity and to follow work time dedicated to each project's task. I have used:

- **Paymo** to measure the real time spent on each part
- **Thinking Rock** as the tool to define tasks, subtasks and to take quick notes about them
- **TiddlyWiki** as a portable notepad to take notes

The initial planning has been optimistic, as usual, but now I have objective data and I hope to improve work time estimation for future projects.

Also this work has been useful to make a first approach to performance testing, a domain were I have never worked before and whose complexity I get now a better idea.

I hope that the "Master on Free Software Projects Development and Management" will make me a better professional with a broader knowledge on Free Software world. In my humble opinion, I think that at least for this time it has been successful.

Regards

José M. Ciges, in Vigo (Spain) at October 2012

163 http://www.mastersoftwarelibre.com/

Bibliography & References

1: "Cassandra is an Apache top level project", by incubator.apache.org . URL: http://www.mail-archive.com/cassandra-dev@incubator.apache.org/msg01518.html

2: "Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase comparison", by Kristóf Kovács . URL: http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis

3: "NoSQL, If Only It Was That Easy", by B. J. Clark . URL: http://bjclark.me/2009/08/nosql-if-only-it-was-that-easy/

4: "Cassandra – A structured storage system on a P2P Network", by Facebook . URL: http://www.facebook.com/note.php?note_id=24413138919&id=9445547199&index=9

5: "What's new in MySQL 5.6", by MySQL Developer Zone . URL: http://dev.mysql.com/tech-resources/articles/whats-new-in-mysql-5.6.html

6: "MySQL 5.6 preview introduces a NoSQL interface", by The "H" web . URL: http://www.h-online.com/open/news/item/MySQL-5-6-preview-introduces-a-NoSQL-interface-1519719.html

7: "NoSQL to InnoDB with Memcached", by TRansactions on InnoDB Blog . URL: http://blogs.innodb.com/wp/2011/04/nosql-to-innodb-with-memcached/

8: "Consistency Models in Non-Relational Databases", by Guy Harrison . URL: http://dbpedias.com/wiki/NoSQL:Consistency_Models_in_Non-Relational_Databases

9: ""Cassandra: The Definitive Guide"", by Eben Hewitt . URL: http://shop.oreilly.com/product/0636920010852.do

10: "CouchDB Vs MongoDB", by Gabriele Lana . URL: http://www.slideshare.net/gabriele.lana/couchdb-vs-mongodb-2982288

11: "Should I use MongoDB or CouchDB (or Redis)?", by Riyad Kalla . URL: https://plus.google.com/107397941677313236670/posts/LFBB233PKQ1

12: "Is this the new hotness now", by Apache . URL: http://www.mail-archive.com/cassandra-dev@incubator.apache.org/msg00004.html

13: "The Underlying Technology of Messages", by Kannan Muthukkaruppan . URL: http://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919

14: ""Third Party Support" article on Apache Cassandra's wiki", by Apache . URL: http://wiki.apache.org/cassandra/ThirdPartySupport

15: ""Deploying Cassandra across Multiple Data Centers" article on Datastax Cassandra Developer Center", by Datastax . URL: http://www.datastax.com/dev/blog/deploying-cassandra-across-multiple-data-centersç

16: ""Hadoop Support" article on Cassandra's wiki", by Apache . URL: http://wiki.apache.org/cassandra/HadoopSupport

17: ""Migrating Netflix from Datacenter Oracle to Global Cassandra" presentation", by Adrian Cockcroft .

URL: http://www.slideshare.net/adrianco/migrating-netflix-from-oracle-to-global-cassandra

18: "Rainbird: Realtime Analytics at Twitter] presentation", by Kevin Weil . URL: http://www.slideshare.net/kevinweil/rainbird-realtime-analytics-at-twitter-strata-2011

19: "From 100s to 100s of Millions presentation", by Erik Onnen . URL: http://www.slideshare.net/eonnen/from-100s-to-100s-of-millions/

20: "Cassandra & puppet, scaling data at $15 per month presentation", by Dave Connors . URL: http://www.slideshare.net/daveconnors/cassandra-puppet-scaling-data-at-15-per-month

21: "Hadoop and Cassandra at Rackspace"] presentation", by Stu Hood . URL: http://www.slideshare.net/stuhood/hadoop-and-cassandra-at-rackspace

22: "mail from Cisco in cassandra-dev mailing list", by Cisco . URL: http://www.mail-archive.com/cassandra-dev@incubator.apache.org/msg01163.html

23: "FAQ on Cassandra's wiki", by Apache . URL: http://wiki.apache.org/cassandra/FAQ#gui

24: ""Client Options" article on Cassandra Wiki", by Apache . URL: http://wiki.apache.org/cassandra/ClientOptions

25: "Cassandra - A Decentralized Structured Storage System", by Avinash Lakshman and Prashant Malik . URL: http://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf

26: ""HBase vs Cassandra: why we moved"", by Dominic Williams . URL: http://ria101.wordpress.com/2010/02/24/hbase-vs-cassandra-why-we-moved/

27: ""4 Months with Cassandra, a love story"", by CloudCick . URL: https://www.cloudkick.com/blog/2010/mar/02/4_months_with_cassandra/

28: ""HBase vs Cassandra"", by Adku . URL: http://blog.adku.com/2011/02/hbase-vs-cassandra.html

29: ""Cassandra vs (CouchDB | MongoDB | Riak | HBase)", by Brian O'Neill . URL: http://brianoneill.blogspot.fr/2012/04/cassandra-vs-couchdb-mongodb-riak-hbase.html

30: ""Introduction to Cassandra: Replication and Consistency" presentation", by Benjamin Black . URL: http://www.slideshare.net/benjaminblack/introduction-to-cassandra-replication-and-consistency

31: ""Exploring CouchDB", article from IBM Developer Works", by Joe Lennon . URL: http://www.ibm.com/developerworks/opensource/library/os-couchdb/index.html

32: "Apache mailing list announcement on mail-archives.apache.org", by Apache . URL: http://mail-archives.apache.org/mod_mbox/incubator-general/200802.mbox/
%3c3d4032300802121136p361b52ceyfc0f3b0ad81a1793@mail.gmail.com%3e

33: ""Re: Proposed Resolution: Establish CouchDB TLP" on mail-archives.apache.org", by Apache . URL: http://mail-archives.apache.org/mod_mbox/incubator-couchdb-dev/200811.mbox/%3c3F352A54-5FC8-4CB0-8A6B-7D3446F07462@jaguNET.com%3e

34: ""CouchDB NoSQL Database Ready for Production Use", article from PC World of Jully 2010", by Joab Jackson . URL: http://www.pcworld.com/businesscenter/article/201046/couchdb_nosql_database_ready_for_production_use.html

35: ""CoachDB, Technical Overview"", by Apache . URL: http://couchdb.apache.org/docs/overview.html

36: ""Welcome to Futon" from "CouchDB The Definitive Guide"", by J. Chris Anderson, Jan Lehnardt and Noah Slater . URL: http://guide.couchdb.org/draft/tour.html#welcome

37: ""CouchDB in the wild" article of the product's web, a list of software projects and websites using CouchDB", by CouchDB . URL: http://wiki.apache.org/couchdb/CouchDB_in_the_wild

38: "Email to the CouchDB-Devel list", by Elliot Murphy . URL: http://mail-archives.apache.org/mod_mbox/couchdb-dev/200910.mbox/%3C4AD53996.3090104@canonical.com%3E

39: "EOL for couchdb and desktopcouch", by . URL: https://lists.ubuntu.com/archives/ubuntu-desktop/2011-November/003474.html

40: ""CouchDB at the BBC as a fault tolerant, scalable, multi-data center key-value store"", by Enda Farrell . URL: http://www.erlang-factory.com/conference/London2009/speakers/endafarrell

41: "View Server Documentation on wiki.apache.org", by Apache . URL: http://wiki.apache.org/couchdb/ViewServer

42: "Backwards_compatibility "Breaking Changes"", by Apache . URL: http://wiki.apache.org/couchdb/Breaking_changes

43: ""Why CouchDB?" from the "CouchDB The Definitive Guide"", by J. Chris Anderson, Jan Lehnardt and Noah Slater . URL: http://guide.couchdb.org/editions/1/en/why.html

44: "“Comparing Mongo DB and Couch DB”, from MongoDB web", by MongoDB . URL: http://www.mongodb.org/display/DOCS/Comparing+Mongo+DB+and+Couch+DB

45: ""MongoDB or CouchDB - fit for production?", question and responses at StackOverflow", by Jason Plank . URL: http://stackoverflow.com/questions/895762/mongodb-or-couchdb-fit-for-production

46: ""3 CouchDB Case Studies" post on Alex Popescu NoSQL blog", by Alex Popescu . URL: http://nosql.mypopescu.com/post/746667801/3-couchdb-case-studies

47: "“CouchDB for access log aggregation and analysis”, post on UserPrimery.net blog", by Seth Falcon . URL: http://userprimary.net/posts/2009/06/13/couchdb-for-access-log-aggregation-and-analysis/

48: "MongoDB Powering MTV's Web Properties", by MongoDB . URL: http://blog.mongodb.org/post/5360007734/mongodb-powering-mtvs-web-properties

49: "MongoDB live at craigslist", by Jeremy Zawodny . URL: http://blog.mongodb.org/post/5545198613/mongodb-live-at-craigslist

50: ""MongoDB at foursquare" Presentation at MongoNYC", by . URL: http://blip.tv/file/3704098

51: "http://www.theregister.co.uk/2011/05/25/the_once_and_future_mongodb/ MongoDB daddy: My baby beats Google BigTable", by . URL:

52: "The MongoDB NoSQL Database Blog, The AGPL", by MongoDB . URL: http://blog.mongodb.org/post/103832439/the-agpl

53: "The MongoDB NoSQL Database Blog, MongoDB 1.4 Ready for Production", by MongoDB . URL: http://blog.mongodb.org/post/472835820/mongodb-1-4-ready-for-production

54: ""The MongoDB NoSQL Database Blog, The AGPL"", by MongoDB . URL: http://blog.mongodb.org/post/103832439/the-agpl

55: ""MongoDB Support" by 10gen", by MongoDB . URL: http://www.10gen.com/subscription

56: "Article "Sharding" on MongoDB Administrator's Manual", by MongoDB . URL: http://www.mongodb.org/display/DOCS/Sharding

57: "GridFS article on MongoDB Developer's Manual", by MongoDB . URL: http://www.mongodb.org/display/DOCS/GridFS

58: "NGINX plugin for MongoDB source code", by Mike Dirolf . URL: http://github.com/mdirolf/nginx-gridfs

59: "lighttpd plugin for MongoDB source code", by Brendan McAdams . URL: http://bitbucket.org/bwmcadams/lighttpd-gridfs/src/

60: ""Use Cases" article at MongoDB's web page", by MongoDB . URL: http://www.mongodb.org/display/DOCS/Use+Cases

61: ""Production Deployents" article on MongoDB web", by MongoDB . URL: http://www.mongodb.org/display/DOCS/Production+Deployments

62: "mongo - The Interactive Shell", by MongoDB . URL: http://www.mongodb.org/display/DOCS/mongo+-+The+Interactive+Shell

63: "“MongoDB Schema Design: How to Think Non-Relational” Jared Rosoff's presentation at Youtube ", by Jared Rosoff . URL: http://youtu.be/PIWVFUtBV1Q

64: "“Realtime Analytics with MongoDB”, presentation by Jared Rosoff", by Jared Rosoff . URL: http://www.slideshare.net/jrosoff/scaling-rails-yottaa

65: "“Web Analytics using MongoDB” of “PHP and MongoDBWeb Development Beginner’s Guide” book", by Rubayeet Islam . URL: http://es.scribd.com/doc/73460113/3623OS-Chapter-5-Web-Analytics-Using-MongoDB-Sample-Chapter

66: "“MongoDB is Fantastic for Logging”", by MongoDB . URL: http://blog.mongodb.org/post/172254834/mongodb-is-fantastic-for-logging

67: "“Using MongoDB for Real-time Analytics”", by MongoDB . URL: http://blog.mongodb.org/post/171353301/using-mongodb-for-real-time-analytics

68: "“Picking the Right NoSQL Database Tool”: post from Monitis' blog", by Monitis . URL: http://blog.monitis.com/index.php/2011/05/22/picking-the-right-nosql-database-tool/

69: ""Real-Time Analytics Schema Design and Optimization"", by Ryan Nitz . URL: http://www.10gen.com/presentations/real-time-analytics-schema-design-and-optimization

70: ""MongoDB for Analytics"", by John Nunemaker . URL: http://www.10gen.com/presentations/mongo-chicago-2011/mongodb-for-analytics

71: ""Real Time Analytics with MongoDB Webinar"", by Jared Rosoff . URL: http://www.10gen.com/presentations/webinar/real-time-analytics-with-mongodb

72: ""Real-Time Log Collection with Fluentd and MongoDB"", by Treasure Data . URL: http://blog.treasure-data.com/post/13766262632/real-time-log-collection-with-fluentd-and-mongodb

73: ""Social Data and Log Analysis Using MongoDB"", by Takahiro Inoue . URL: http://www.slideshare.net/doryokujin/social-data-and-log-analysis-using-mongodb