My Next Modular Forms Database

William Stein (joint work with Mike Hansen)

October 2010

Abstract

Database Architecture
W. Stein
The Overall Architecture
The Database – MongoDB
The Web Interface – Flask, mod_wsgi, Apache
Demo Site
Summary

I Have Data

I have oodles of data on web pages and in files only I know how to use, and with Sage I can generate much more. I am putting all of this data into a web-accessible database server. Thanks to the NSF I can allocate terabytes of disk space to this database, and have money to buy extra computers for redundancy.

Today's Technology is Better

Last time I tried putting together a database like this was in 2003; technology has dramatically improved since then. Most computers are 64-bit, which removes numerous annoying barriers, and there are good documented-oriented databases. This talk is about how I intend to to put together this database, and will be of interest to others with similar goals.

Database

Database Architecture
W. Stein
The Overall Architecture
The Database – MongoDB
The Web Interface – Flask, mod_wsgi, Apache
Demo Site
Summary

Servers

- MongoDB master – disk.math.washington.edu (in Seattle)
- MongoDB slave 1 – in Seattle on William Stein's OS X desktop (?)
- MongoDB slave 2 – in Waterloo on Mike Rubinstein OS X computer

Disk Space?

- Try to limit the database footprint for this project to 4 terabytes, so that a single $\sim$ \$350 USB disk plugged into any computer (Linux, OS X, etc.) can server as a redundant MongoDB slave.
- But, if things get too big, I'll use "sharding".

Security and Users

- The *master MongoDB server* will run directly on our big Ubuntu Linux fileserver, listening only on localhost.
- A user who needs direct *write* access to the database will have their ssh key added to a limited account on this machine, and via ssh port forwarding, they will be able to access the database, using a login and password that gives them access to a subset of the databases or collections served by MongoDB.
- A single MongoDB server can *simultaneously* serve numerous completely independent databases, and independent requests from different users.

Web Interface

Software

- Flask microframework: http://flask.pocoo.org/
- Apache: via `mod_wsgi`

- Use the Flask Python library (Flask is from the same group that brought us Jinja, Sphinx, etc.) to develop a web front end for to the database.
- Webpage will enable anybody to easily make fast queries.
- Will create indexes in the MongoDB database that optimize queries available through the web interface.
- Will deploy our Flask application using Apache's `mod_wsgi` module, which is scales well.

Use MongoDB from C, C++, Javascript, Python (Sage), Perl, etc.

- Will run several *read-only MongoDB slave servers*. Good for arbitrary queries against the database. Some queries involve server side javascript run on millions of documents, and can take a long time and put a heavy load on the database server.
- MongoDB officially supports accessing a MongoDB server from any of C, C++, Java, Javascript, Perl, PHP, Python (hence Sage!), and Ruby. There are numerous other languages that are not officially supported, but are here: http://www.mongodb.org/display/DOCS/Drivers
  No math software besides Sage, e.g., none of Magma, Mathematica, Maple, or Matlab, is in that list.

Web Upload?

- Data upload is done by connecting to the *master* database via a programming language.
- There is *no web page upload for data* as part of my planned architecture, due to security issues and time constraints. However, if somebody else makes a web upload system, they could act as an "editor" and submit the results of uploads to my MongoDB database.

MongoDB: a Documented Oriented Database

At the Paris meeting, David Farmer has put forth an idea that "the basic building blocks in the project are the individual homepages of each object of interest."
MongoDB (http://www.mongodb.org/)

1. is a new free open source *documented oriented* database management system, written in C++.
2. is much different than a SQL database such as SQLite, MySQL, or PostgreSQL.
3. data model corresponds to Farmer's idea of homepages.
4. easily builds indexes and does elaborate optimized queries.
5. automatically *replicates* to any number of backup servers.

MongoDB makes your data "feel smaller"

- This summer I tested using MongoDB to deal with masses of data I generated related to modular forms for a research project with Barry Mazur.
- I also tested putting all of the Cremona and Stein-Watkins tables of elliptic curves in a single big MongoDB database.
- It made a vast amount of data (hundreds of gigabytes) feel "small".

I have *never* had this feeling before with huge number theory tables using any other database, including PostgreSQL, MySQL, sqlite, ZODB, and custom filesystem based stores.

How to Learn MongoDB

Go to http://www.mongodb.org/ and start browsing.

- Tons of quickstarts, tutorials, articles, and videos of talks, slides, etc.
- A Company is behind MongoDB; but don't worry, MongoDB is free and open source
- Some not-quite-finished *books* about MongoDB; I read them by temporarily signing up for an O'Reilly Safari books membership (http://my.safaribooksonline.com), reading them, then unsubscribing.

Setting up a MongoDB Server

Step by step (more details below)

1. Binaries (Linux, OS X, Windows, Solaris) from `http://www.mongodb.org/downloads`.
2. Start a MongoDB server running by typing `mongod`.
3. Connect by typing `mongo` in another window.
4. Connect from Python (or Sage) using `pymongo`.

Starting a MongoDB server

I run my mongod server by typing:

```
mongod --dbpath /lvm/array/lmfdb/mongodb  \
          --bind_ip localhost  --port 29000
```

The dbpath option specifies where the files for the database are stored and the bind_ip and port options makes it so mongod accepts connections on localhost port 29000; otherwise, anybody in the world could just connect to your mongodb and delete all your data!! If you want to run mongod on a remote server somewhere, but easily connect to it from your laptop (say), setup an ssh tunnel by simply typing:

```
ssh -L 29000:localhost:29000 remote.computer.edu
```

It's also possible to create accounts with various permissions from the mongo console.

Connecting to MongoDB via the console

Connect to your new MongoDB server with the Mongo console

```
wstein@disk$ mongo localhost:29000
MongoDB shell version: 1.6.1
connecting to: localhost:29000/test
> show dbs
admin
local
research
> help
        db.help()     help on db methods
        ...
```

Connecting to MongoDB from Sage (Python)

Database Architecture
W. Stein
The Overall Architecture
The Database – MongoDB
The Web Interface – Flask, mod_wsgi, Apache
Demo Site
Summary

Install Pymongo (about 10 seconds)

```
sage: !easy_install pymongo
...
sage: quit    # important!
```

Use it

```
sage: import pymongo
sage: C = pymongo.Connection('localhost:29000')
sage: C.database_names()
[u'research', u'admin', u'local']
sage: R = C.research; R
Database(Connection('localhost', 29000), u'research')
sage: R.[tab key] ...
sage: R.collection_names()
[u'mazur_irreg.done', ..., u'fs.chunks', u'fs.files']
```

MongoDB's structure: Databases, Collections and Documents

- A MongoDB server serves a collection of *independent* databases.
- A *database* is a set of collections, and a *collection* is a set of documents.
- A *document* is like a Python dictionary, but only a limited number of datatypes are allowed.
- Technically, a document is a "BSON" document, where BSON is a format very similar to JSON.

MongoDB documents are limited

A MongoDB document must be at most 4MB in size. Let's push the limits, to see what this means in practice:

Use it

```
sage: foo = R.foo
sage: foo.insert({'test':'a'*(4*10^6)})
ObjectId('4cae369075688b3eab000006')
sage: foo.insert({'test':'a'*(5*10^6)})
Traceback (most recent call last):
...
InvalidDocument: document too large - BSON documents are lim
```

So you could store a string with 4 million characters, but not 5 million; for reference, 4 million characters is about *1,000 typed pages of text*.

How to Store Huge Stuff: GridFS

- Recall: MongoDB documents can be at most 4MB in size!
- GridFS get arounds this; stores gigantic data in MongoDB
- No indexing and searching capabilities
- GridFS is just a key:value store, built on top of MongoDB.

Using GridFS

```
sage: import gridfs
sage: G = gridfs.GridFS(R)  # we defined the db R above
sage: G.put('a'*(5*10^6), filename='test1')
ObjectId('4cae3ba075688b3eab000008')
sage: x = G.get_last_version('test1').read(); len(x)
5000000
sage: x[:30]
'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'
```

Using GridFS to Store Pickles

You can store arbitrary Sage objects using dumps and loads, which are wrappers around Python's pickle module:

Store a mathematical object

```
sage: M = ModularSymbols(389, 2)
sage: G.put(dumps(M), filename='modsym389')
ObjectId('4cae3c1a75688b3eab00001d')
sage: loads(G.get_last_version('modsym389').read())
Modular Symbols space of dimension 65 for Gamma_0(389)
of weight 2 with sign 0 over Rational Field
```

You get one GridFS per database, so if you have documents in all sorts of collections that somehow point to GridFS "files", you'll need to choose some systematic way of naming the files.

Flask: a web development "micro-framework"

Database Architecture
W. Stein
The Overall Architecture
The Database – MongoDB
The Web Interface – Flask, mod_wsgi, Apache
Demo Site
Summary

FLASK

http://flask.pocoo.org/

"Hello world" written using Flask

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__": app.run()
```

Put the above in a file hello.py and...

```
$ easy_install Flask    # 30 seconds?
$ python hello.py
```

Using Flask

- I don't have time in this talk to go into detail about how to use Flask in general.
- The documentation at http://flask.pocoo.org/docs/ is excellent.
- You use decorators to construct the URL mapping, deal with GET and POST requests, etc.
- You can also put static/ and templates/ subdirectories in your Python project, and relevant files will get pulled.
- You need to learn the Jinja2 templating engine: http://jinja.pocoo.org/2/.

Demo Site

The DEMO is here

http://db.modform.org/

- Mike Hansen and I built a demo site.
- Illustrates the architecture sketched above by providing access to a large table of over a hundred million elliptic curves (Cremona plus Stein-Watkins)
- Will form the core for the new modular forms database.

Try it out! It's running on boxen.math.washington.edu, in Seattle.

Apache Setup

/etc/apache2/sites-available/lmfdb

```
NameVirtualHost db.modform.org:80
<VirtualHost db.modform.org:80>
    ServerName db.modform.org
    WSGIDaemonProcess lmfdb threads=5
    WSGIScriptAlias / /home/mhansen/lmfdb/lmfdb.wsgi
    <Directory /home/mhansen/lmfdb>
        WSGIProcessGroup lmfdb
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

And a symbolic link:

```
/etc/apache2/sites-available/lmfdb --->
                /etc/apache2/sites-enabled/lmfdb
```

WSGI Setup

The WSGI application is defined by this file:

http://sage.math.washington.edu/home/mhansen/lmfdb/lmfdb.wsgi

The main thing that this file has to do is define some object called "application" which will obey the WSGI protocol. There are a few other things in there to let it know about the environment. Here are the contents:

```
import os, sys
sys.path.append('/home/mhansen/lmfdb')
os.environ['PYTHON_EGG_CACHE'] = '/home/mhansen/lmfdb/.pytho
activate_this = '/home/mhansen/lmfdb/env/bin/activate_this.p
execfile(activate_this, dict(__file__=activate_this))
from lmfdb import app as application
```

(It is important to look at the files mentioned above.)

Python/Flask Code

Look at the files in

http://wstein.org/talks/2010-10-lmfdb/demo.tar.bz2

In addition to the templates, there's a file lmfdb.py:

```
from flask import Flask, url_for, render_template, request
app = Flask(__name__)
from pymongo import Connection
db = Connection(port=int(29000)).research
...
@app.route('/ellcurves/rank/<int:rank>/')
@ellcurves_list
def ellcurves_of_rank(rank):
    curves = db.ellcurves.find({'r':rank}).sort('level')
    return locals()
...
```

This file defines what happens when a URL is accessed.

Summary

This talk has laid out the architecture that I will be using for my new web-based databases. It uses the following free open source tools together in a natural way:

- **Python**: a high quality programming language
- **MongoDB**: a scalable documented-oriented database
- **Flask**: a mmicro-frameworkffor Python-based web apps
- **Jinja2**: a general purpose templating language
- **Apache + WSGI**: scalable web server

You can try out a demo that combines the above right now at: http://db.modform.org