Math 581d: Overview of the Modular Forms Database Architecture

William Stein

December 6, 2010

Abstract

I Have Data

- Have data; Sage can generate much more.
- Problem: put it into a web-accessible database server.
- Have plenty of disk space and computers.

Today's Technology is Better

- Last time I tried: 2003
- Technology improved: 64-bit, big disks, better databases
- How to put it together to make a web-based database?

Database

Servers

- MongoDB master – disk.math.washington.edu (in Seattle)
- MongoDB slave 1 – my OS X desktop
- MongoDB slave 2 – OS X desktop

Disk Space?

- Limit database footprint for this project to 4 terabytes, so single $\sim$ \$300 USB disk plugged into any computer can server as a redundant MongoDB slave.
- (If not, use "sharding".)

Security and Users

- The *master MongoDB server* runs on our Linux fileserver, listening only on 'localhost'. Port forwarded to the machines in the sage.math cluster via ssh, so any of those machine can securely connect.
- MongoDB user accounts. Users needing write access get account on MongoDB server.
- A single MongoDB server can *simultaneously* serve numerous completely independent databases, and independent requests from different users.

Scalable Web Interface

Software

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

- Use the Flask micro web framework.
- Create indexes to optimize queries.
- Run *read-only MongoDB slave servers* for queries.
- Deploy our Flask application using Apache's `mod_wsgi`: http://code.google.com/p/modwsgi/.

MongoDB: a Documented Oriented Database

MongoDB (http://www.mongodb.org/):

1. Relatively new free open source *documented-oriented* database management system, written in C++, built on Boost, PCRE, and SpiderMonkey (Javascript) libraries.
2. Much different than a SQL database such as SQLite, MySQL, or PostgreSQL.
3. Easily build indexes and do elaborate optimized queries.
4. Replication and sharding.
5. FAST.

WARNING: MongoDB documents are limited

A MongoDB document must be at most 4MB in size (will increase to 16MB in next major release).

Pushing the limits

```
sage: import pymongo
sage: R = pymongo.Connection('localhost:29000').research
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 limited to 4 MB
```

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

- MongoDB documents can be at most 4MB in size.
- GridFS stores gigantic data using MongoDB
- GridFS is just a key:value store, built on top of MongoDB.

Using GridFS

```
sage: import pymongo, gridfs
sage: R = pymongo.Connection('localhost:29000').research
sage: G = gridfs.GridFS(R)
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 Sage Objects

- Store Sage objects using Python's pickle:

Store a mathematical object

```
sage: import pymongo, gridfs
sage: R = pymongo.Connection('localhost:29000').research
sage: G = gridfs.GridFS(R)
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 systematically name keys.

Flash: a web development “micro-framework”

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
$ python hello.py
```

Using Flask

- http://flask.pocoo.org/docs/ is excellent.
- Use decorators to construct the URL mapping.
- Put static/ and templates/ subdirectories in your Python project.
- Use Jinja2 templating engine: http://jinja.pocoo.org/2/.

Demo Sites

1. See http://db.modform.org/
   - Mike Hansen and I built this.
   - Illustrates the architecture sketched above
   - Provides access to several hundred million elliptic curves (Cremona plus Stein-Watkins)
2. See http://www.modform.org/
   - Also uses Flask + MongoDB.
   - Created at the Paris workshop in October.

Apache Setup

/etc/apache2/sites-available/lmfdb

```
<VirtualHost *>
    ServerName db.modform.org

    WSGIDaemonProcess lmfdb threads=5
    WSGIScriptAlias / /home/wstein/db/lmfdb/site/lmfdb.wsgi

    <Directory /home/wstein/db/lmfdb/site>
        WSGIProcessGroup lmfdb
        WSGIApplicationGroup %{GLOBAL}
        WSGIScriptReloading On
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

And a symbolic link:

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

WSGI Setup

The WSGI application is defined by this file:

http://sage.math.washington.edu/home/wstein/db/lmfdb/site/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 am using for my new web-based databases:

- **Python/Sage**: high quality programming environment
- **MongoDB**: scalable documented-oriented database
- **Flask**: "micro-framework" for Python-based web apps
- **Apache + WSGI**: scalable web server