igvita.com

A goal is a dream with a deadline.

Asynchronous DB: DBSlayer & HTTP

You hit a web 2.0 nerve and that magical 'viral coefficient' is working with full force, except now you' have a problem: scaling the database. No problem, you say, we have a few tricks up our sleeve: faster disks, loads of memory, dataset sharding, load balancing, connection pooling, master-master and master-slave replication schemes, multiple caching layers... anything to minimize the latency of the database call.

The problem is that dreaded dynamic request which cuts through the cache and requires a blocking database query, which has an unfortunate side effect of locking up the resources on your application server as well. Wouldn't it be nice if we could fetch that data asynchronously instead?

Over the years, there have been many attempts at asynchronous database drivers, with Asymy as the most recent Ruby variant by Thomas Ptacek. However, every time someone starts another one of these projects, you have to wonder: why are we duplicating our efforts? After all, we already have a battle-tested protocol that we're all familiar with: HTTP. Hence, when I saw the announcement, and later sat on a presentation of DBSlayer at MySQL Conf '08, I knew that the guys at NYT were onto something big.

Database calls over HTTP

Instead of choosing a specific language or platform, DBSlayer speaks and understands JSON - something that any language (even your browser!) can easily produce and consume. With failover support, connection pooling, round-robin slave distribution, and a MySQL interface, I'm surprised it hasn't received more attention! Let's take it for a test drive:

```
# connect to localhost database, expose DBSlayer on port 81
dbslayer -s localhost -c mysql.conf -p 81 -l db.log

# Check if it's live
curl http://192.168.0.198:81/stats

{"current_time" : 1218011845 , "hits" : [8 , 1] , "slices" : [1218011724 , 1218011784] , "start_time" : 1218011724 ,
"total_requests" : 9}
```

Now that DBSlayer is up and running, accessing your database is as simple as making an HTTP call:

> dbslayer.rb

```
require 'rubygems'
require 'net/http'
require 'json'
require 'cgi'

# instead of querying the database, send a HTTP call to DBSlayer
def http_request(query)
  Net::HTTP.start('localhost', 81) { |http|
      req = Net::HTTP::Get.new('/db?'+ CGI.escape(query.to_json))
      response = http.request(req)
      return JSON.parse(response.body)
```

```
    }
end

# select the database. Output:
#  >> {"HOST_INFO" : "Localhost via UNIX socket" , "RESULT" : {"SUCCESS" : true} , "SERVER" : "localhost"}
http_request({'SQL' => 'use dbslayer'})

# issue the real sql query! Output:
# >> ["bobblehead", 5]
# >> ["toy", 2]
# >> ["gadget", 3]
http_request({'SQL' => 'select * from widgets'})['RESULT']['ROWS'].each do |row|
  p row
end
```

Asynchronous DB with DBSlayer and EventMachine

HTTP comes with some overhead, but it also offers the potential for hundreds of battle tested scalability tools. One of which is the ability to easily turn any HTTP request into a non-blocking request! Borrowing some sample code from my previous post on Ruby EventMachine, we have an easy Ruby web-server (port 8082) which makes an asynchronous DB call to fetch the data:

> em_dbslayer.rb

```
require 'rubygems'
require 'eventmachine'
require 'evma_httpserver'
require 'json'
require 'cgi'

class Handler  < EventMachine::Connection
  include EventMachine::HttpServer

  def process_http_request
    resp = EventMachine::DelegatedHttpResponse.new(self)

    # connection pool on DBSlayer, tell the connection which DB we're accessing.
    query = {"SQL" => "USE dbslayer; SELECT * from widgets"}

    http = EM::Protocols::HttpClient2.connect("localhost", 81)
    d = http.get "/db?" + CGI.escape(query.to_json)

    # defer the response until we get response from DBSlayer
    d.callback {
      resp.status = 200
      resp.content = d.content
      resp.send_response
    }
  end
end

EventMachine::run {
  EventMachine.epoll
  EventMachine::start_server("0.0.0.0", 8082, Handler)
  puts "Listening..."
}
```

dbslayer-code.zip (Sample Net-HTTP, EventMachine, MySQL Ruby code)

Downloads: 444 File Size: 85.9 KB

Scaling over HTTP

DBSlayer is a young project (albeit deployed in production at NYTimes) and there is definitely room for improvement: there are