# Scripting libdrizzle with Lua inside Nginx

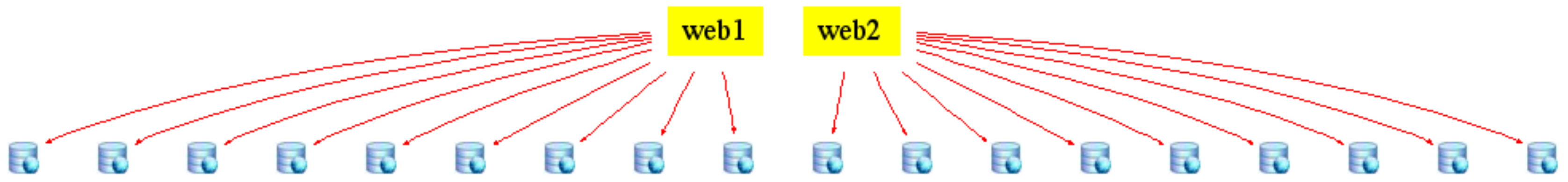
# Scripting libdrizzle with Lua inside Nginx

*章亦春 (agentzh)*

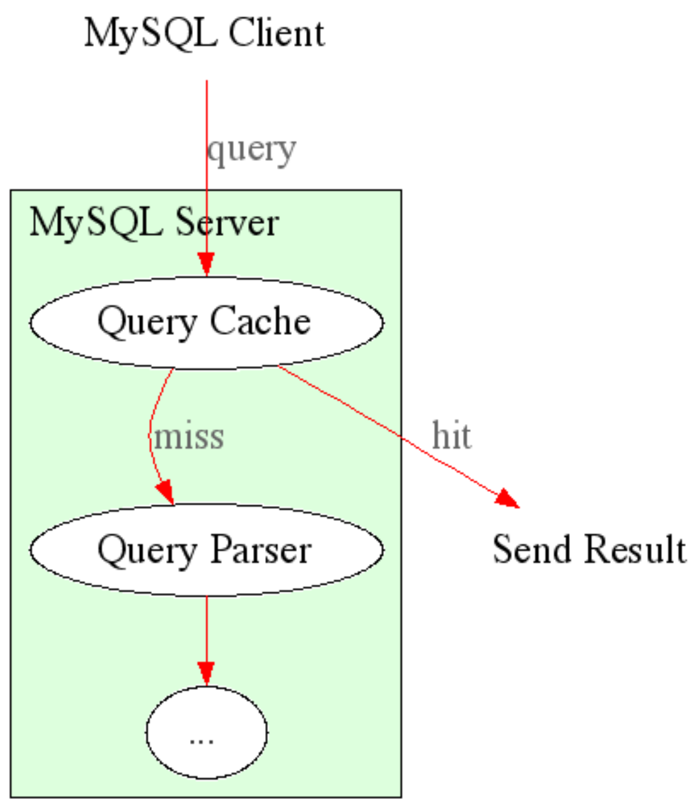
☺ *agentzh@gmail.com* ☺

*2012.4*

"MySQL is always the *bottleneck*!"
"Really?!"

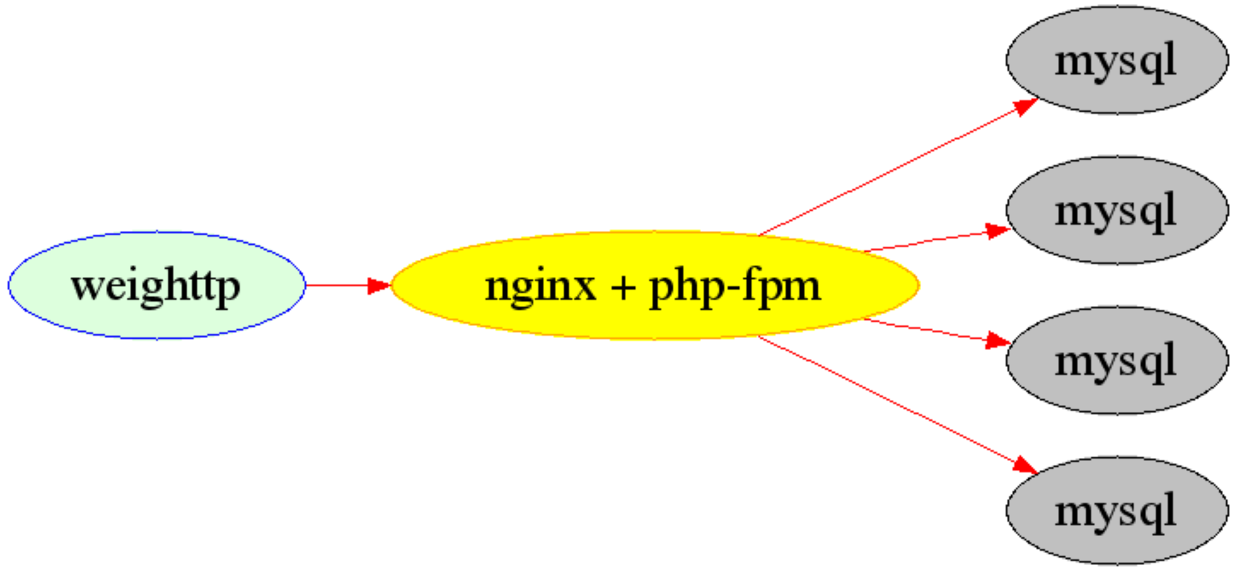


A Cluster of Frontend Web Servers and MySQL Backend Servers



Hitting MySQL Query Cache

☺ Some *benchmarks* on
Amazon EC2 Small instances

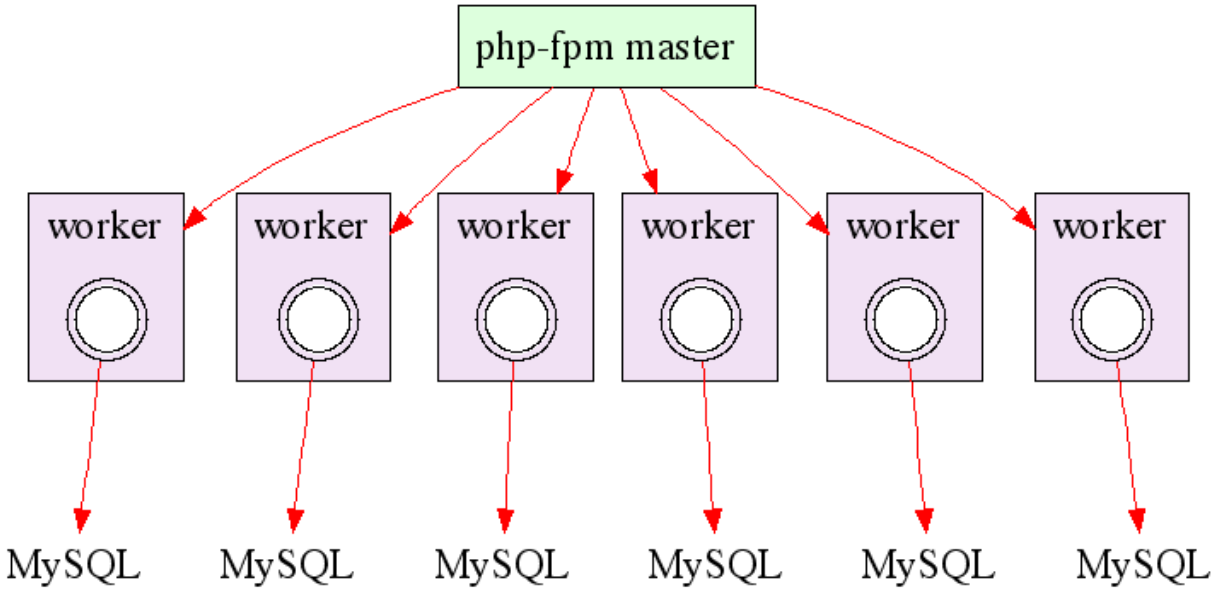


A Test Cluster of Amazon EC2 Small Instances

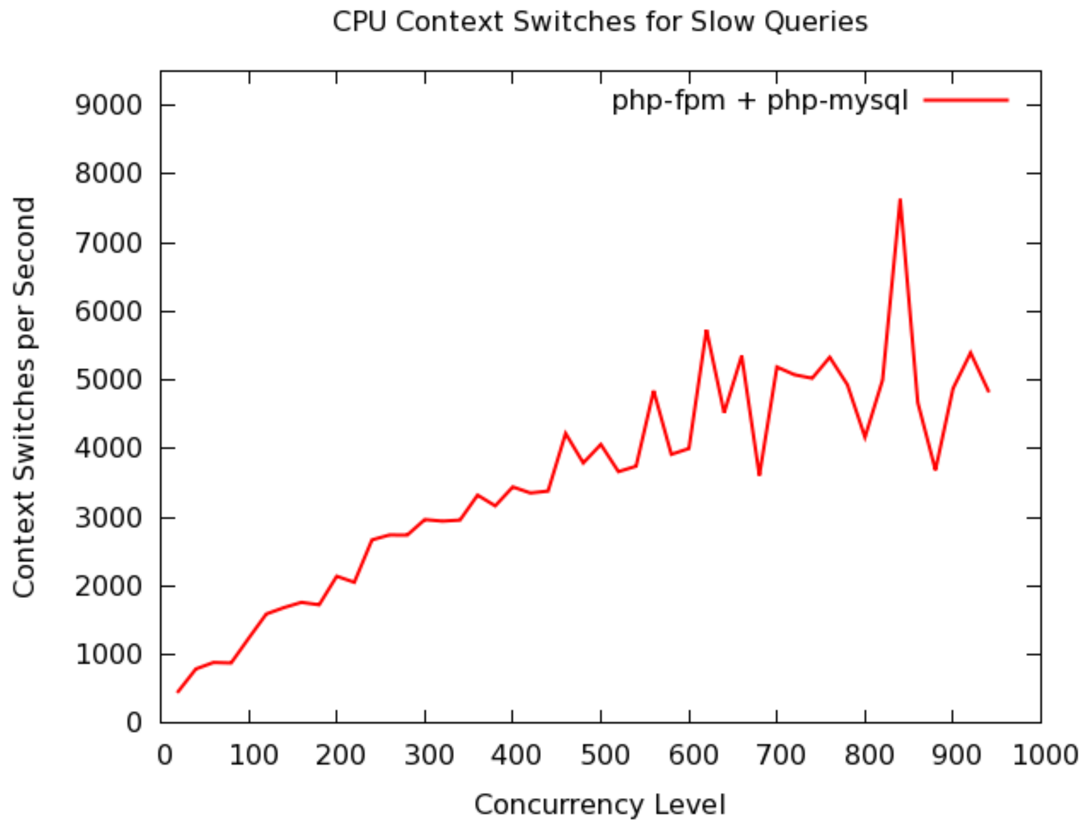
## ☺ A *Slow* MySQL Query

```
select sleep(1)
```

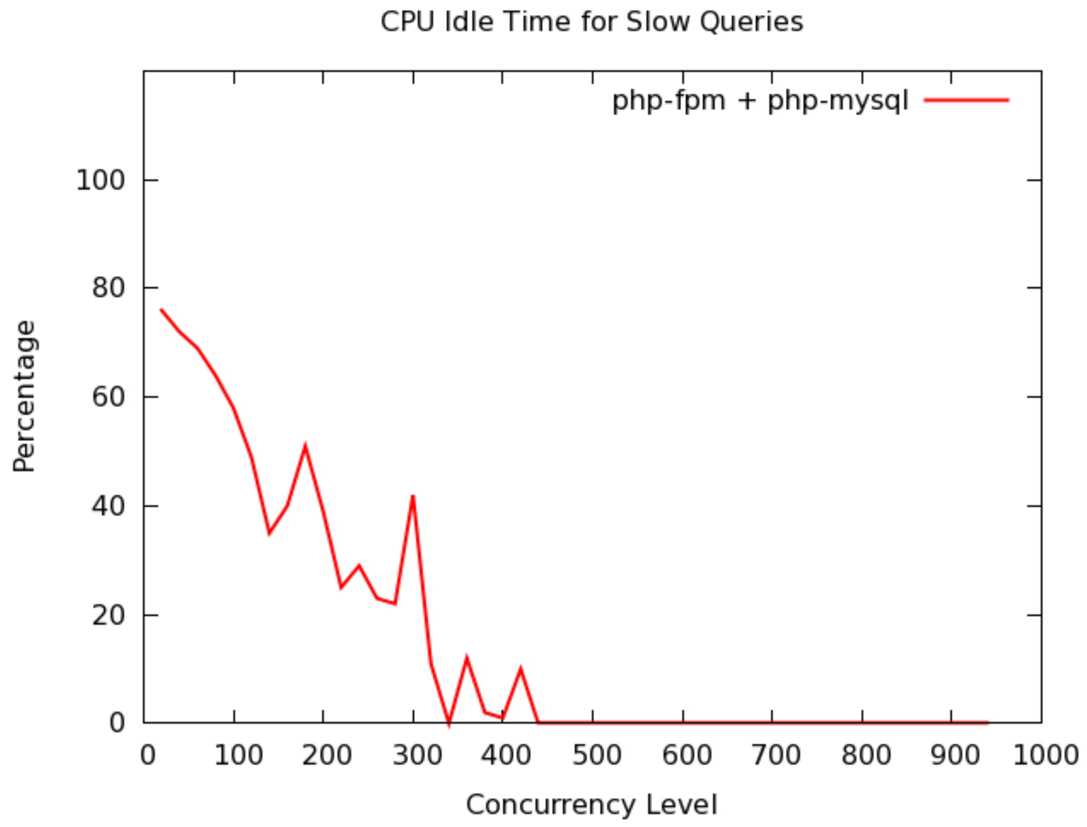
- ♡ **Amazon Linux AMI** *2011.09*
- ♡ **nginx** *1.0.14*
- ♡ **php-fpm** *5.3.10*

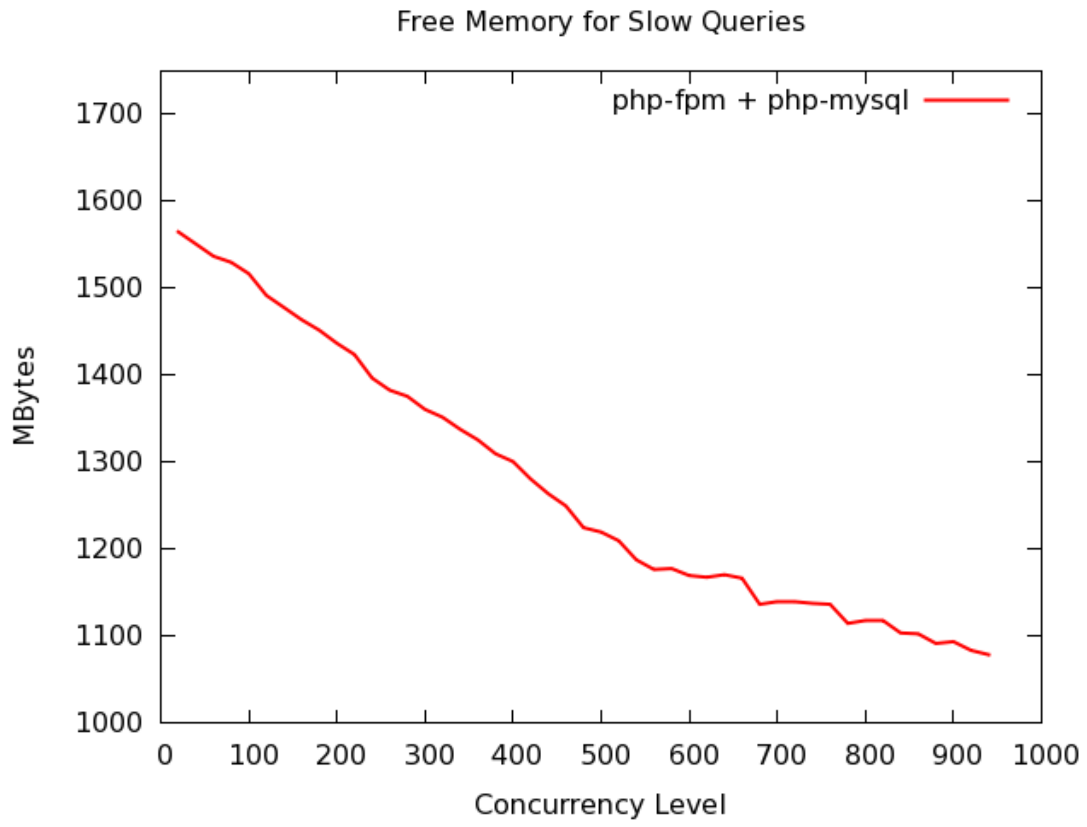


PHP-FPM's Multi-Worker Model and Blocking MySQL Connections



CPU Context Switches for Slow Queries
php-fpm + php-mysql
Context Switches per Second
Concurrency Level
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
100
200
300
400
500
600
700
800
900
1000

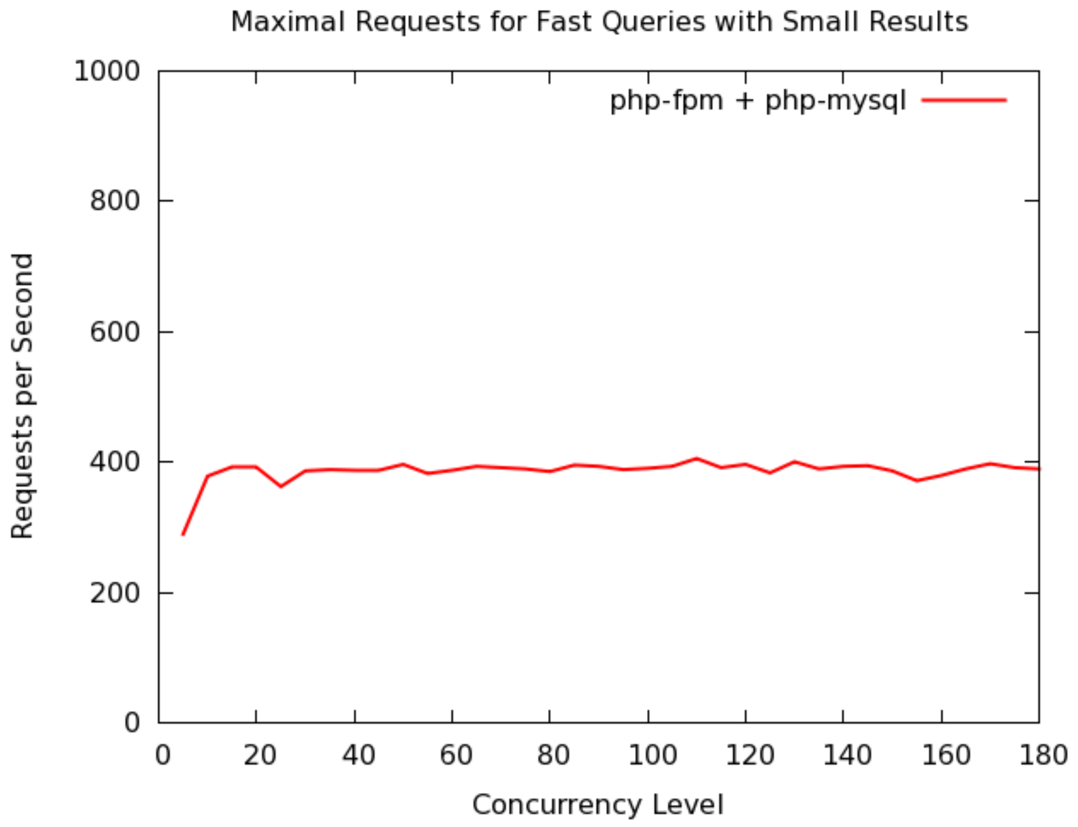

CPU Idle Time for Slow Queries
php-fpm + php-mysql
Percentage
0
20
40
60
80
100
0
100
200
300
400
500
600
700
800
900
1000
Concurrency Level



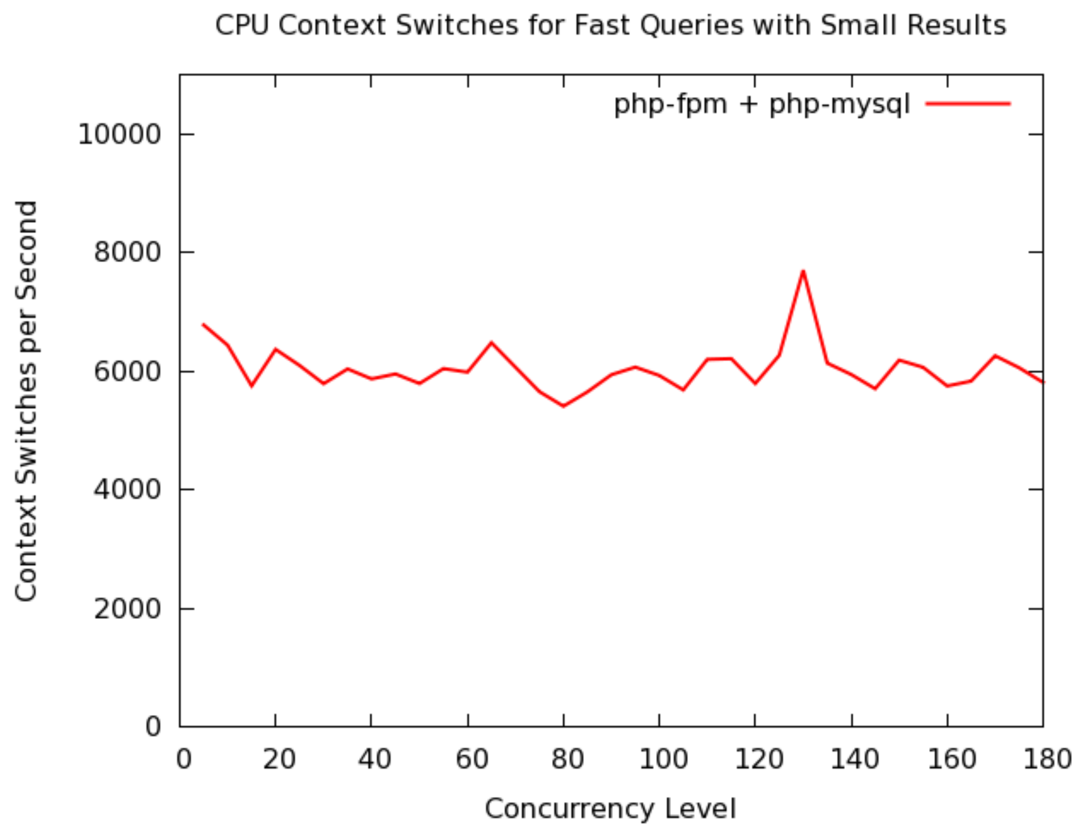
Free Memory for Slow Queries
php-fpm + php-mysql
MBytes
1700
1600
1500
1400
1300
1200
1100
1000
0
100
200
300
400
500
600
700
800
900
1000
Concurrency Level

# ☺ A *Fast* MySQL Query with a Small Resultset

```
select *
from world.City
order by ID
limit 1
```



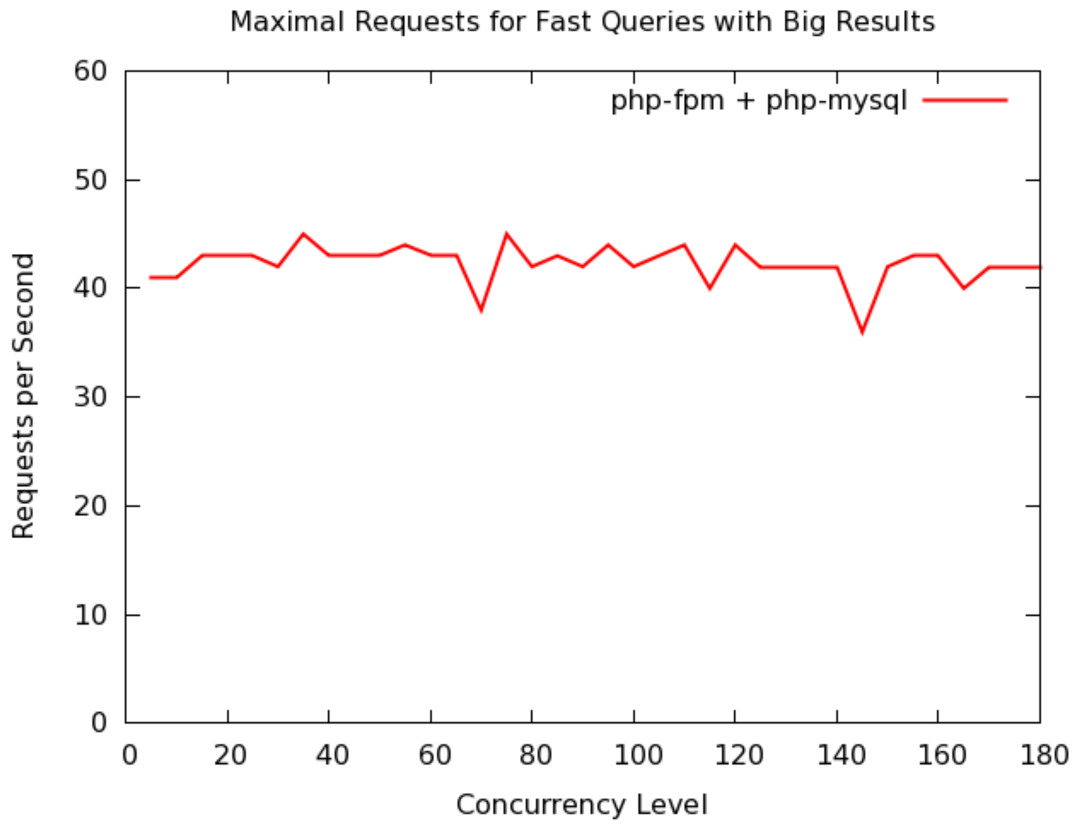
Maximal Requests for Fast Queries with Small Results
php-fpm + php-mysql
Requests per Second
Concurrency Level
0
200
400
600
800
1000
0
20
40
60
80
100
120
140
160
180



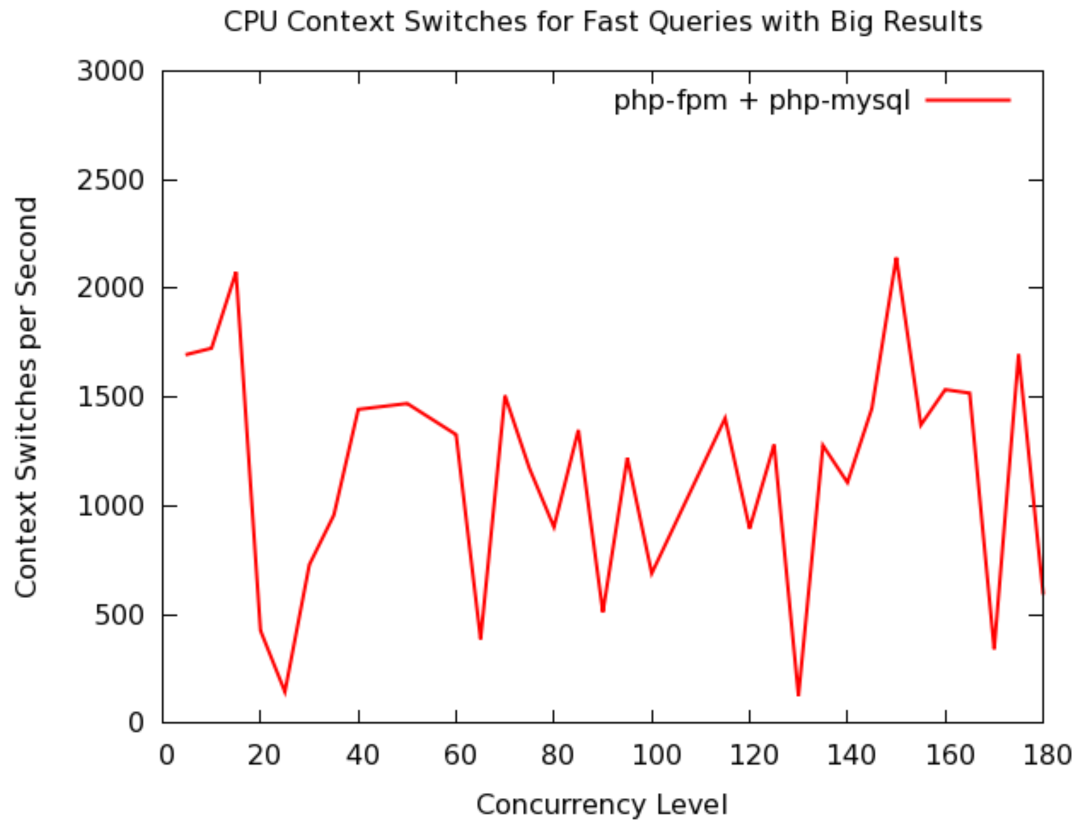
CPU Context Switches for Fast Queries with Small Results
php-fpm + php-mysql
Context Switches per Second
Concurrency Level
0
2000
4000
6000
8000
10000
0
20
40
60
80
100
120
140
160
180

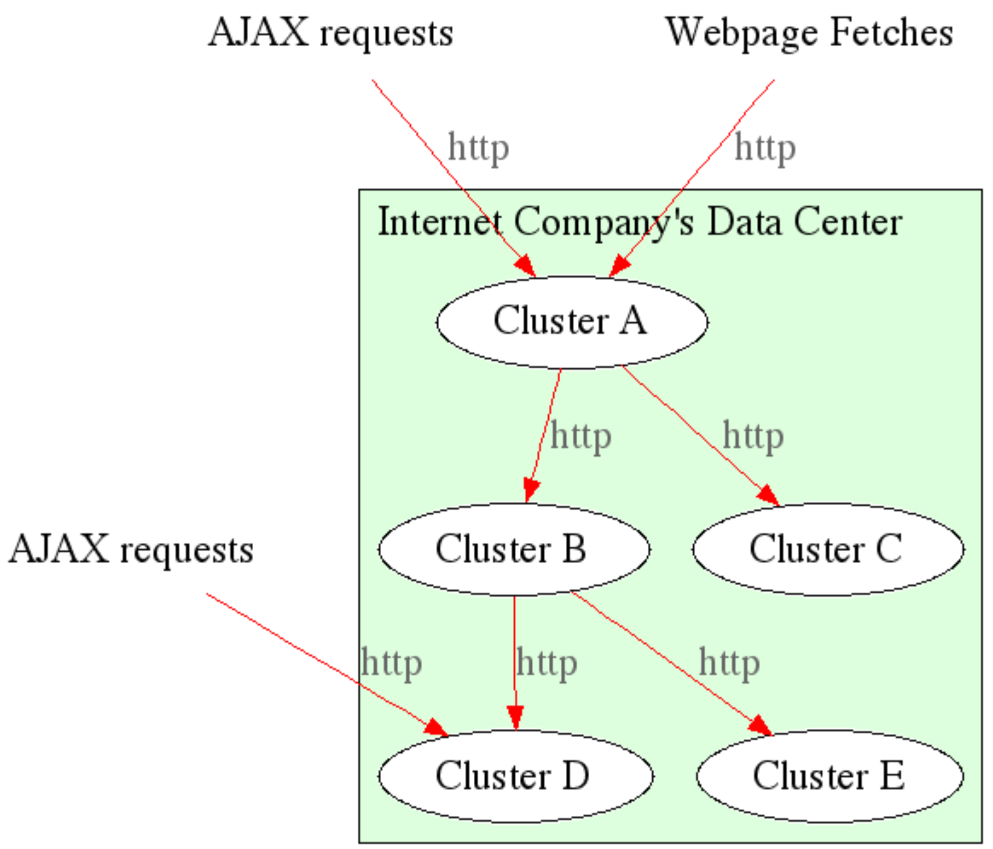
## ☺ A *Fast* MySQL Query with a Big Resultset (100 KBytes)

```
select *
from world.City
order by ID
limit 1000
```



Maximal Requests for Fast Queries with Big Results
php-fpm + php-mysql
Requests per Second
Concurrency Level
0
10
20
30
40
50
60
0
20
40
60
80
100
120
140
160
180

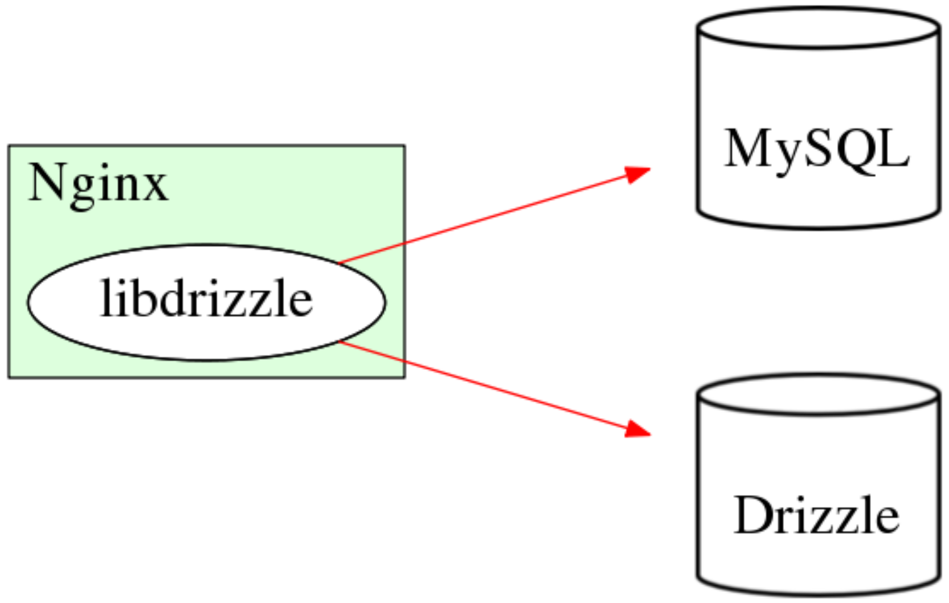

CPU Context Switches for Fast Queries with Big Results
php-fpm + php-mysql
Context Switches per Second
Concurrency Level
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120
140
160
180



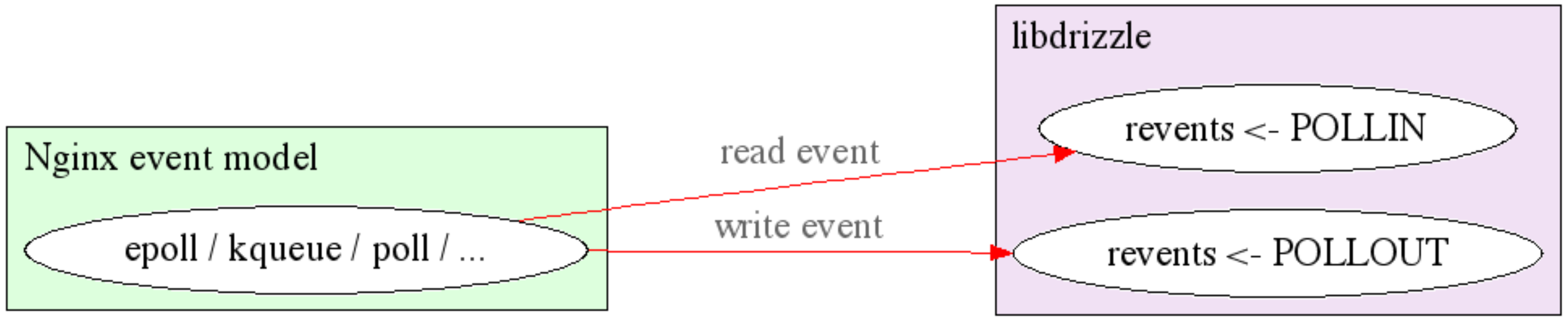
Service-ization Happening in Some Internet Companies

☺ We integrated *libdrizzle* directly into Nginx!

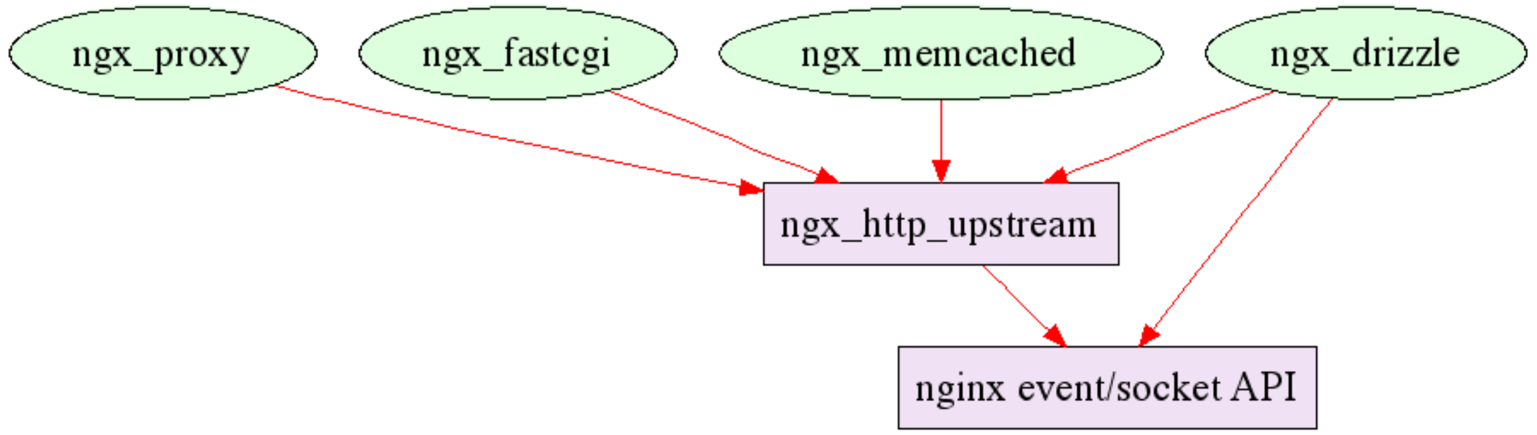
http://wiki.nginx.org/HttpDrizzleModule



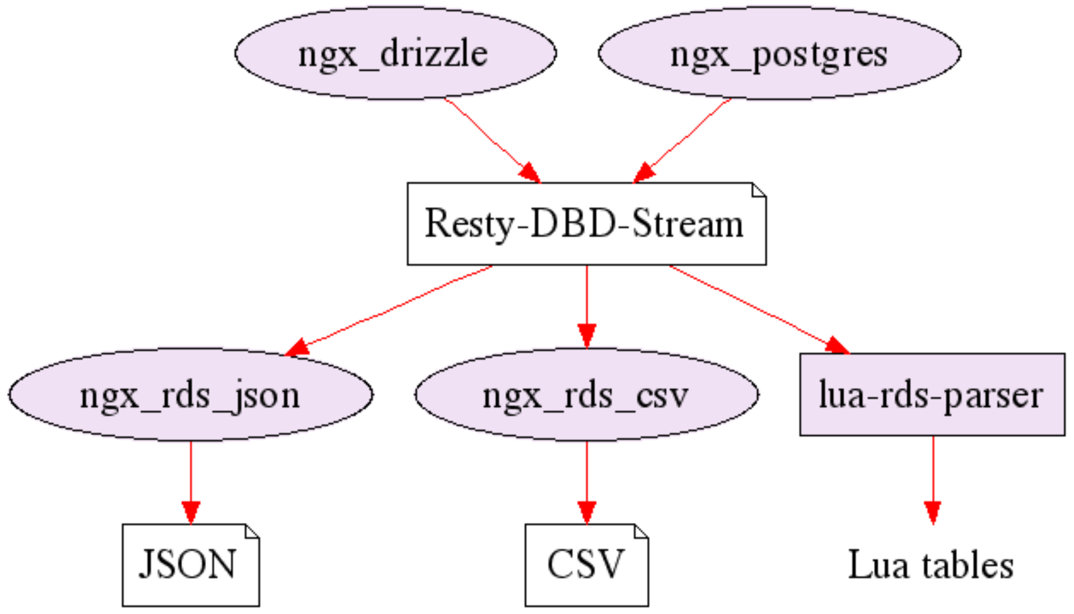
Using libdrizzle to talk to MySQL or Drizzle servers



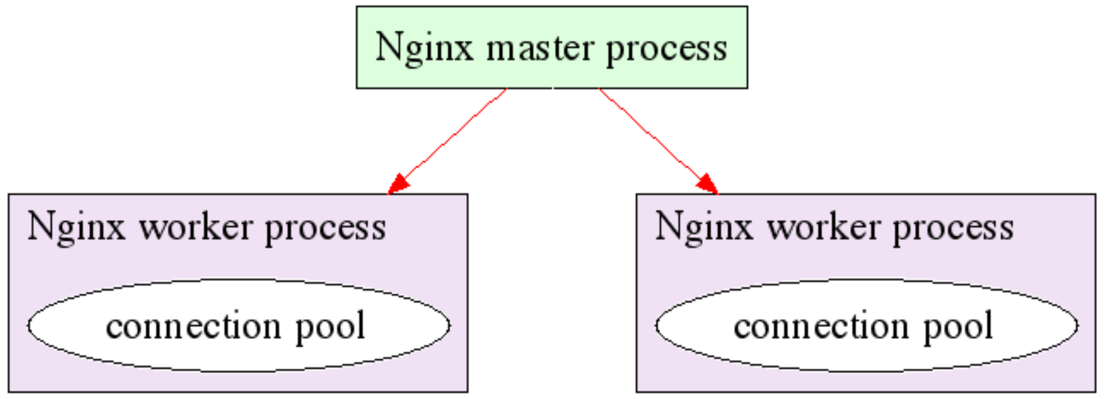
Integrating libdrizzle with Nginx events



Nginx Upstream Architecture



Resty-DBD-Stream Data Flow



Nginx Multi-Worker Model and Connection Pools

☺ Let's just mud with *nginx.conf*, the Nginx configuration file

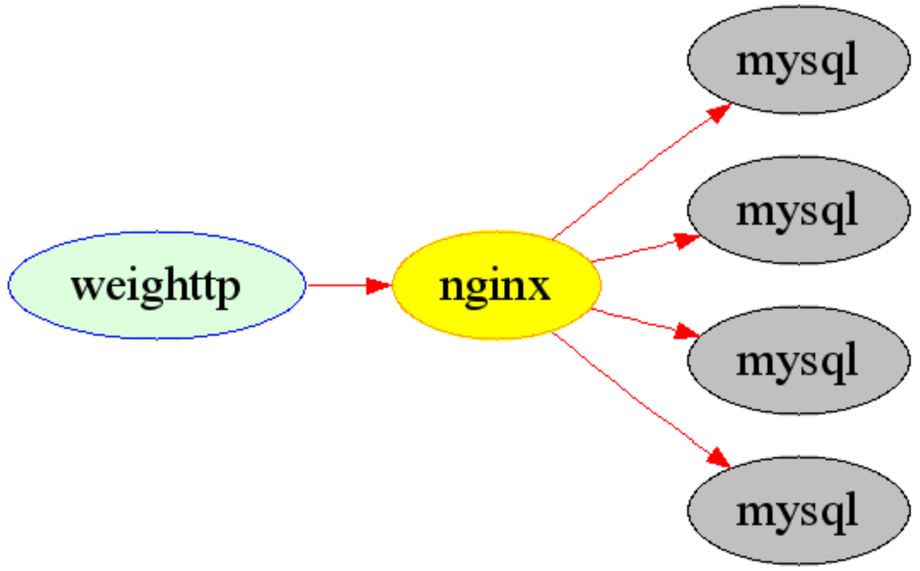
```
upstream my_mysql_backend {
    drizzle_server 127.0.0.1:3306 dbname=test
                password=some_pass user=monty
                protocol=mysql;

    # a connection pool that can cache up to
    #   200 mysql TCP connections
    drizzle_keepalive max=200 overflow=reject;
}
```

```
location ~ '^/cat/(.*)' {
    set $name $1;
    set_quote_sql_str $quoted_name $name;
    drizzle_query "select *
        from cats
        where name=$quoted_name";

    drizzle_pass my_mysql_backend;

    rds_json on;
}
```

```
$ curl 'http://localhost/cat/Jerry'
[{"name":"Jerry","age":1}]
```

## ☺ The *dynamic* SQL Query for This Request

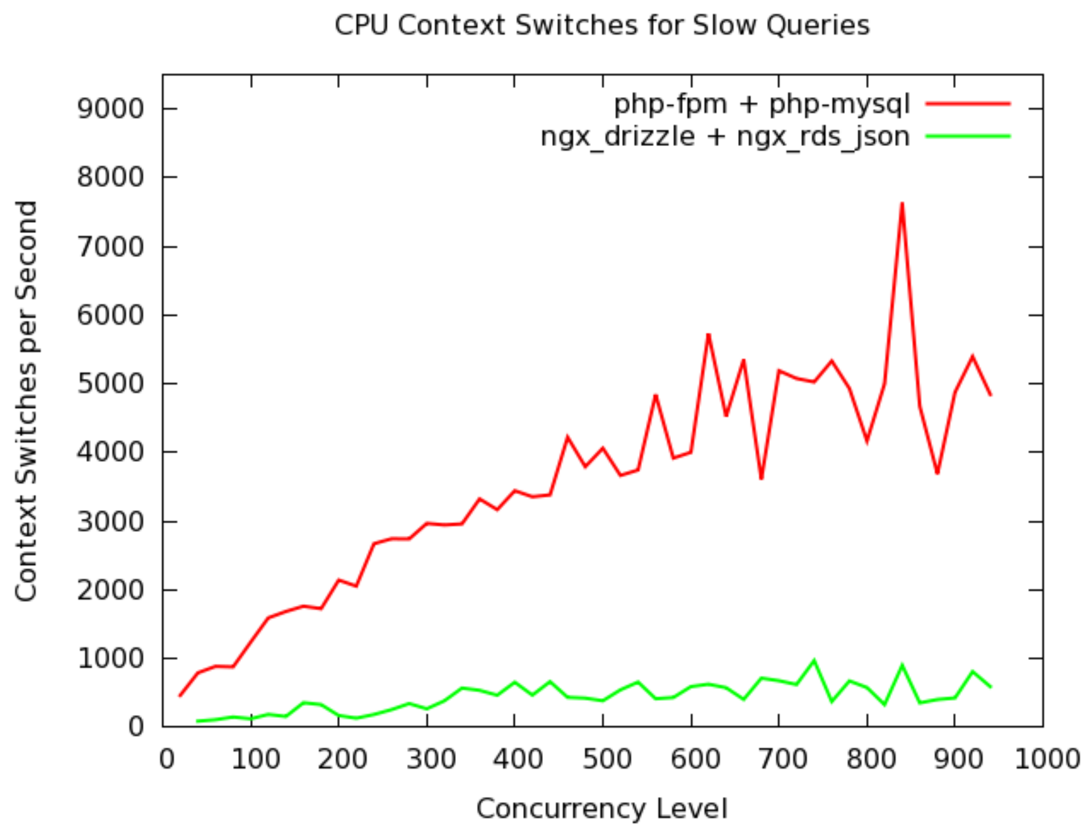
```
select *
from cats
where name='Jerry'
```

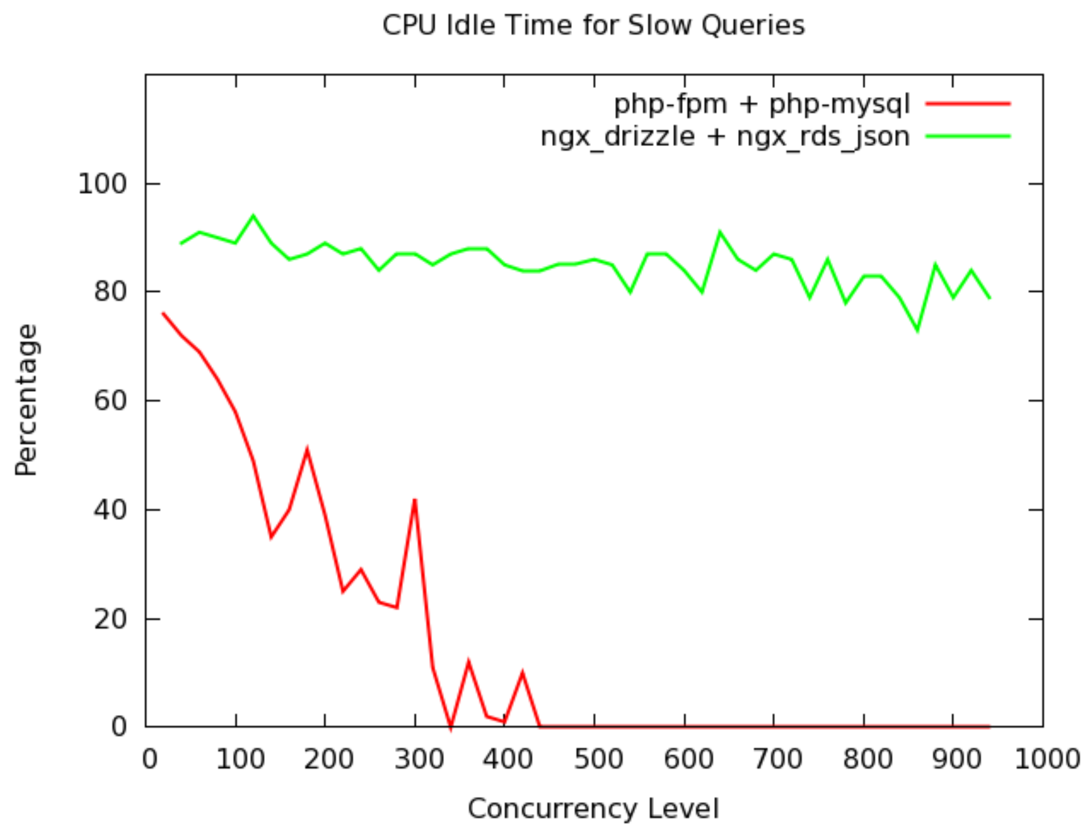


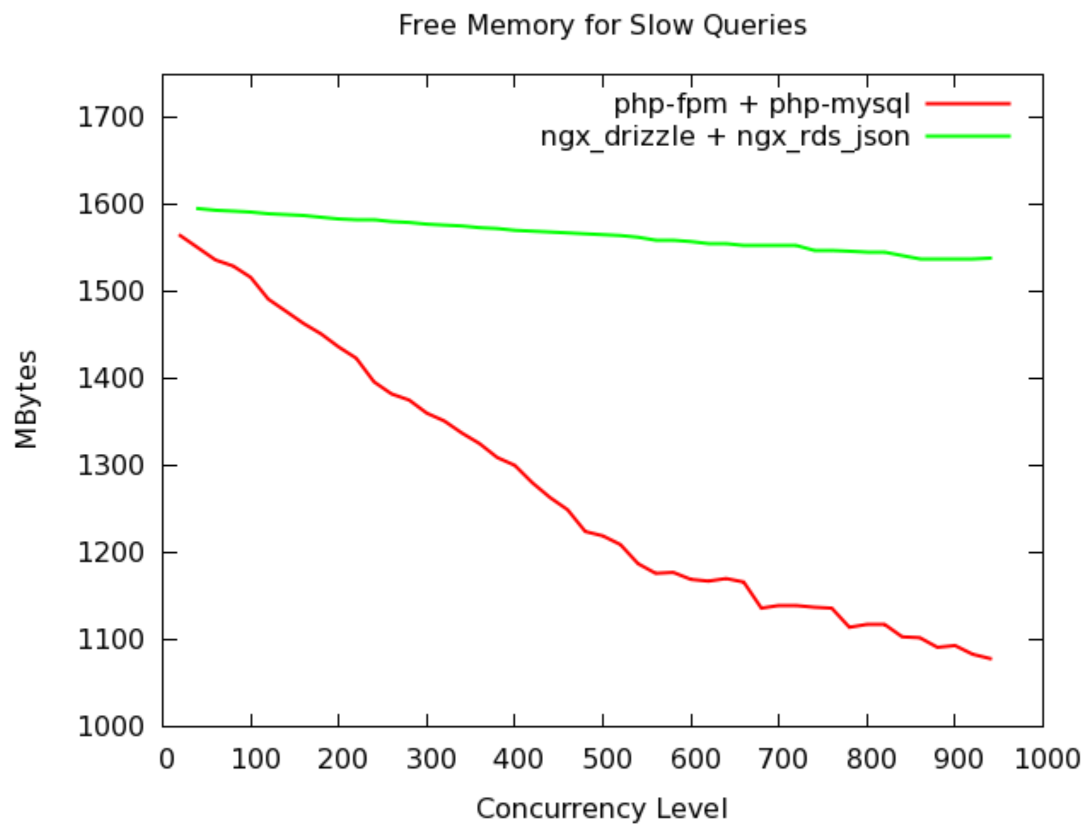
A Test Cluster of Amazon EC2 Small Instances

- ☺ The *Slow* MySQL Query again!

```
select sleep(1)
```

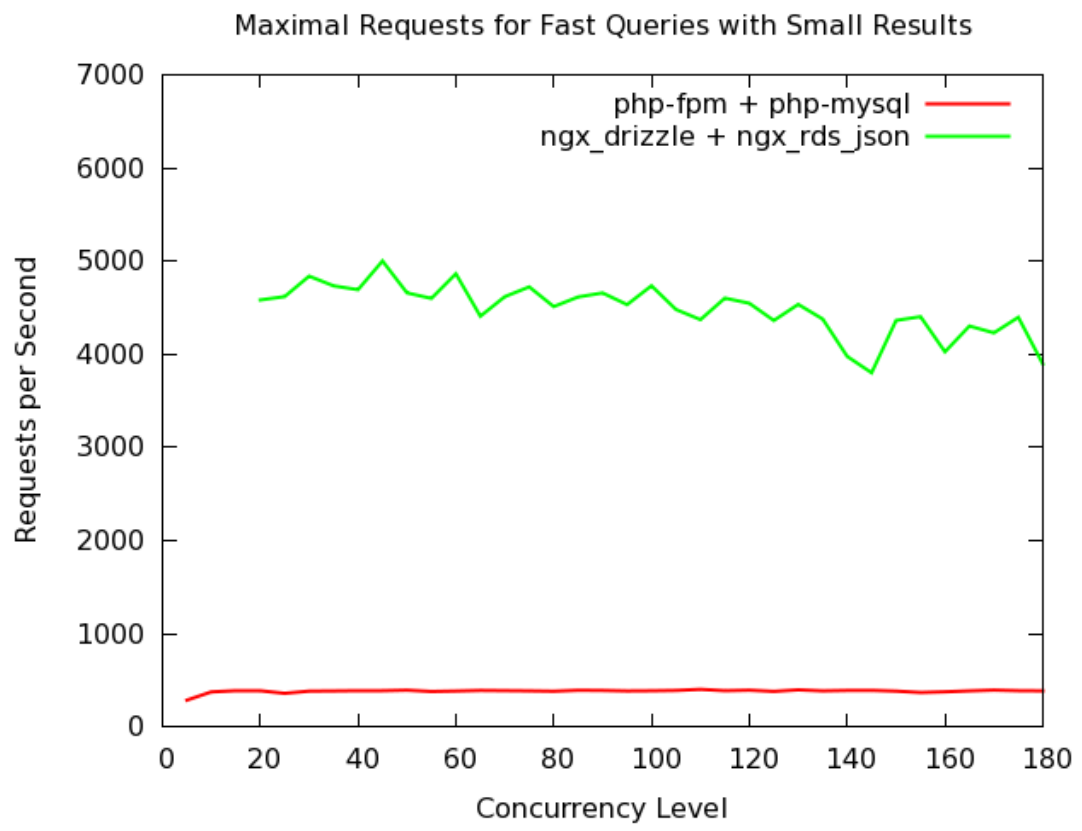

CPU Context Switches for Slow Queries
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
Context Switches per Second
Concurrency Level
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
0
100
200
300
400
500
600
700
800
900
1000

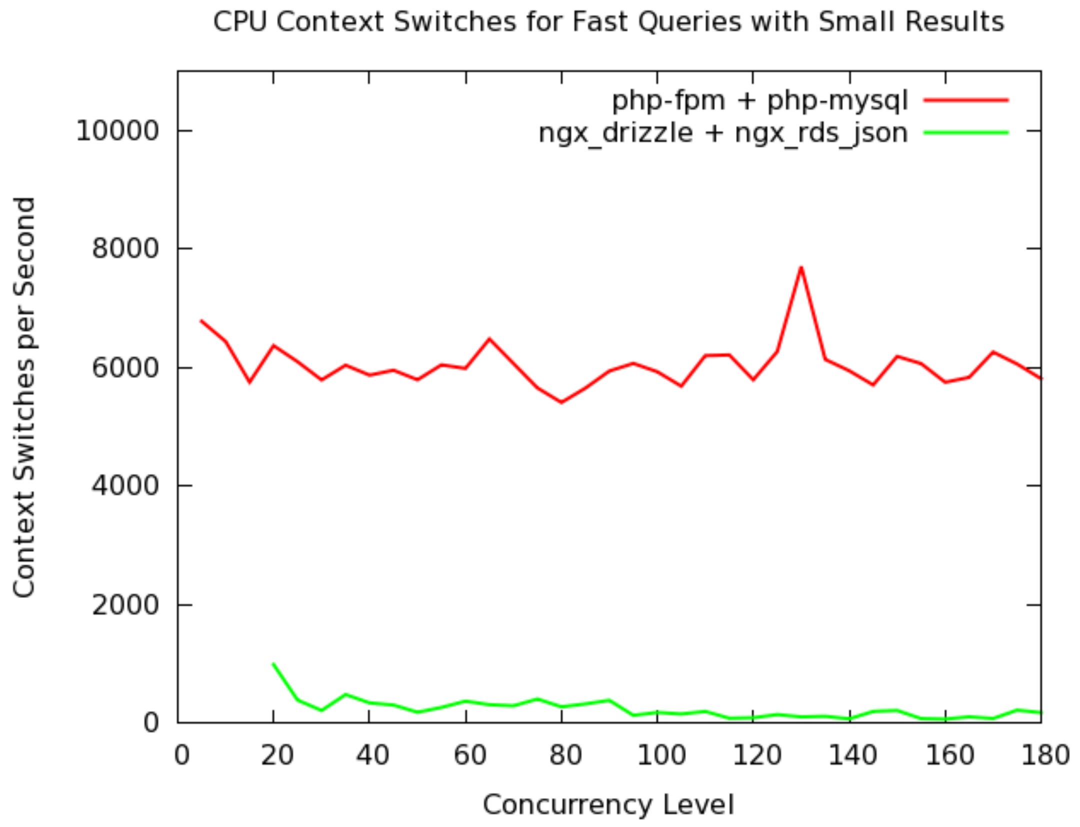

CPU Idle Time for Slow Queries
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
Percentage
Concurrency Level
0
20
40
60
80
100
0
100
200
300
400
500
600
700
800
900
1000


Free Memory for Slow Queries
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
MBytes
Concurrency Level
1000
1100
1200
1300
1400
1500
1600
1700
0
100
200
300
400
500
600
700
800
900
1000

# ☺ The *Fast* MySQL Query with a Small Resultset Again!

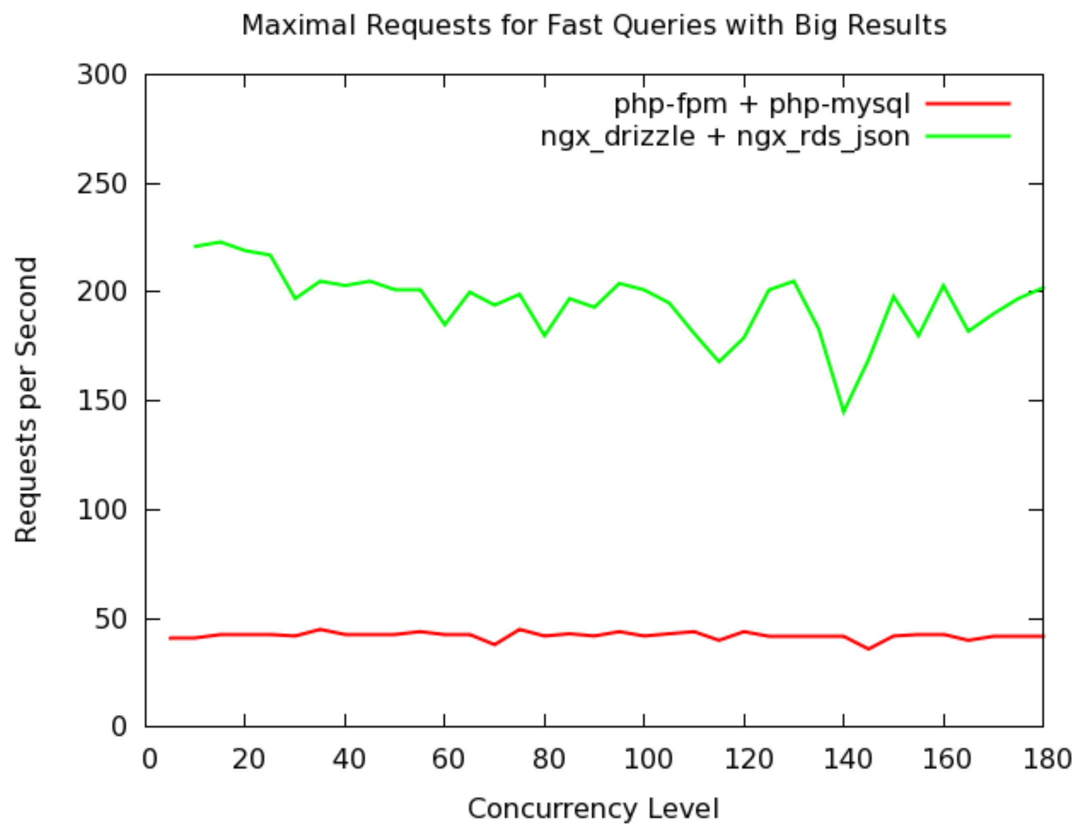
```
select *
from world.City
order by ID
limit 1
```

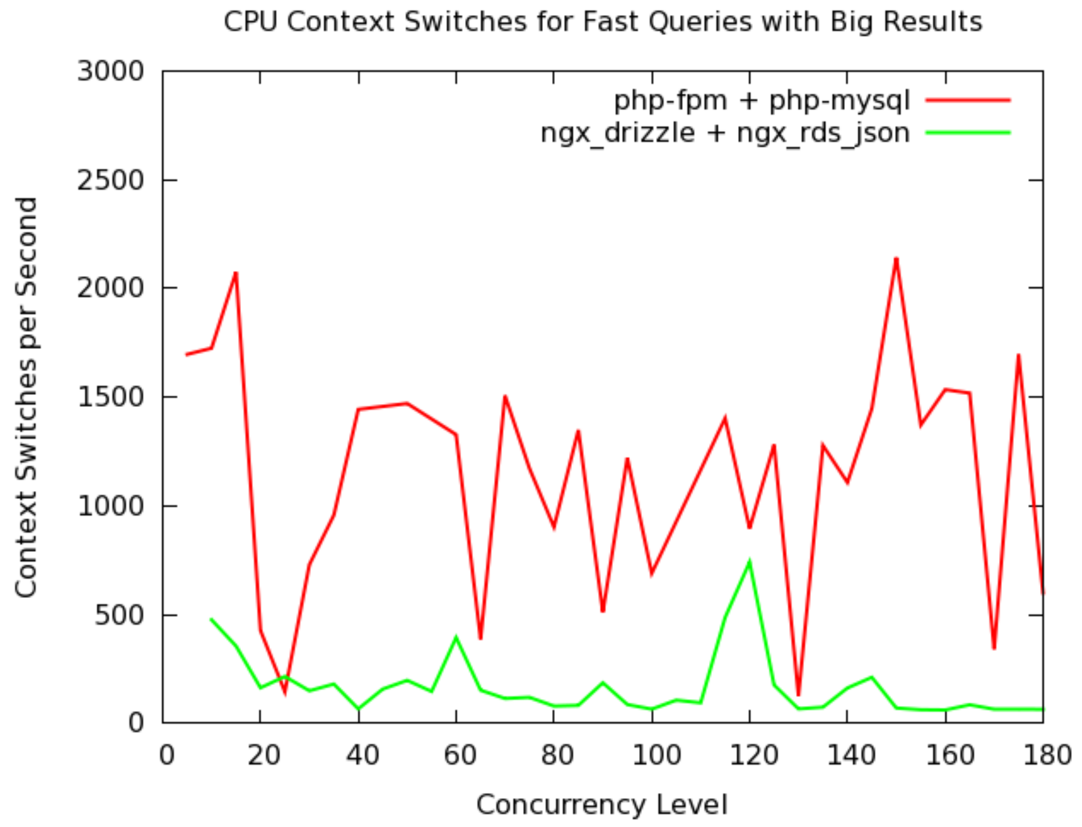

Maximal Requests for Fast Queries with Small Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
Requests per Second
Concurrency Level
0
1000
2000
3000
4000
5000
6000
7000
0
20
40
60
80
100
120
140
160
180


CPU Context Switches for Fast Queries with Small Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
Context Switches per Second
Concurrency Level
0
2000
4000
6000
8000
10000
0
20
40
60
80
100
120
140
160
180

## ☺ The *Fast* MySQL Query with a Big Resultset (100 KBytes) Again!

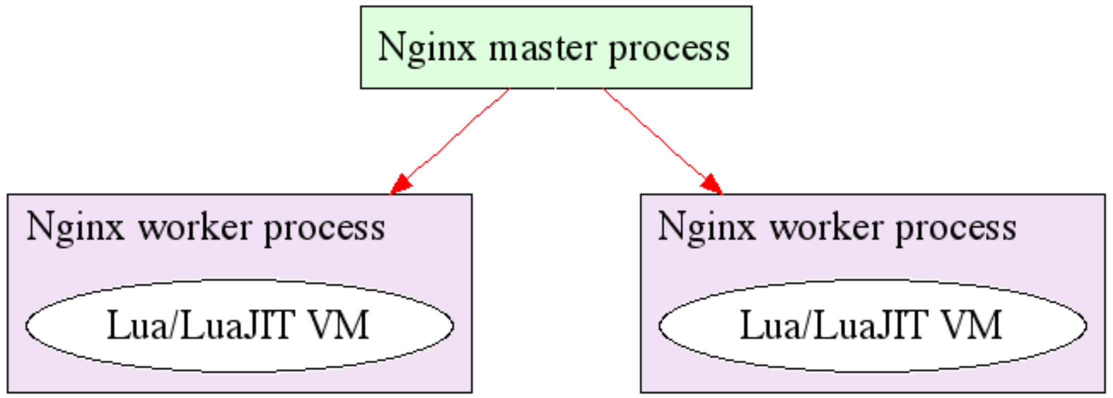
```
select *
from world.City
order by ID
limit 1000
```


Maximal Requests for Fast Queries with Big Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
Requests per Second
Concurrency Level
0
50
100
150
200
250
300
0
20
40
60
80
100
120
140
160
180


CPU Context Switches for Fast Queries with Big Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
Context Switches per Second
Concurrency Level
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120
140
160
180

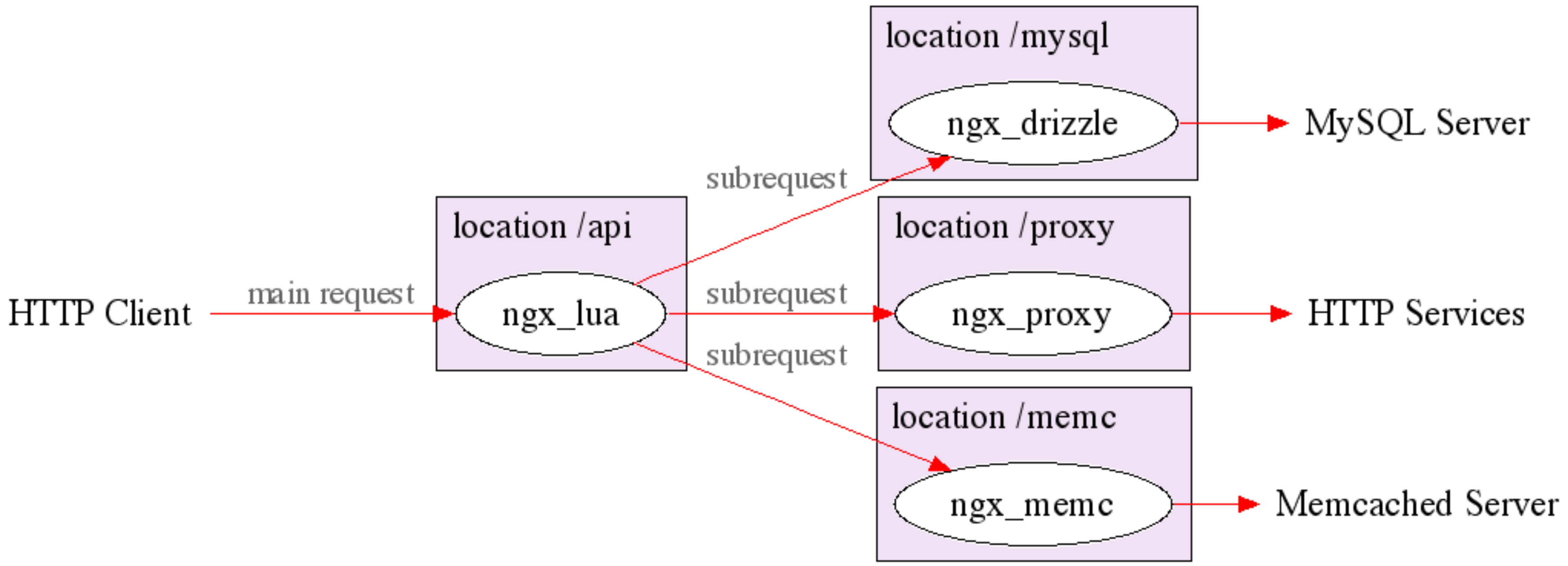
☺ We also embedded *Lua* and *LuaJIT* directly into Nginx!

http://wiki.nginx.org/HttpLuaModule



Nginx Multi-Worker Model and Lua/LuaJIT VMs

☺ Use the *Lua* language to access the ngx_drizzle module!



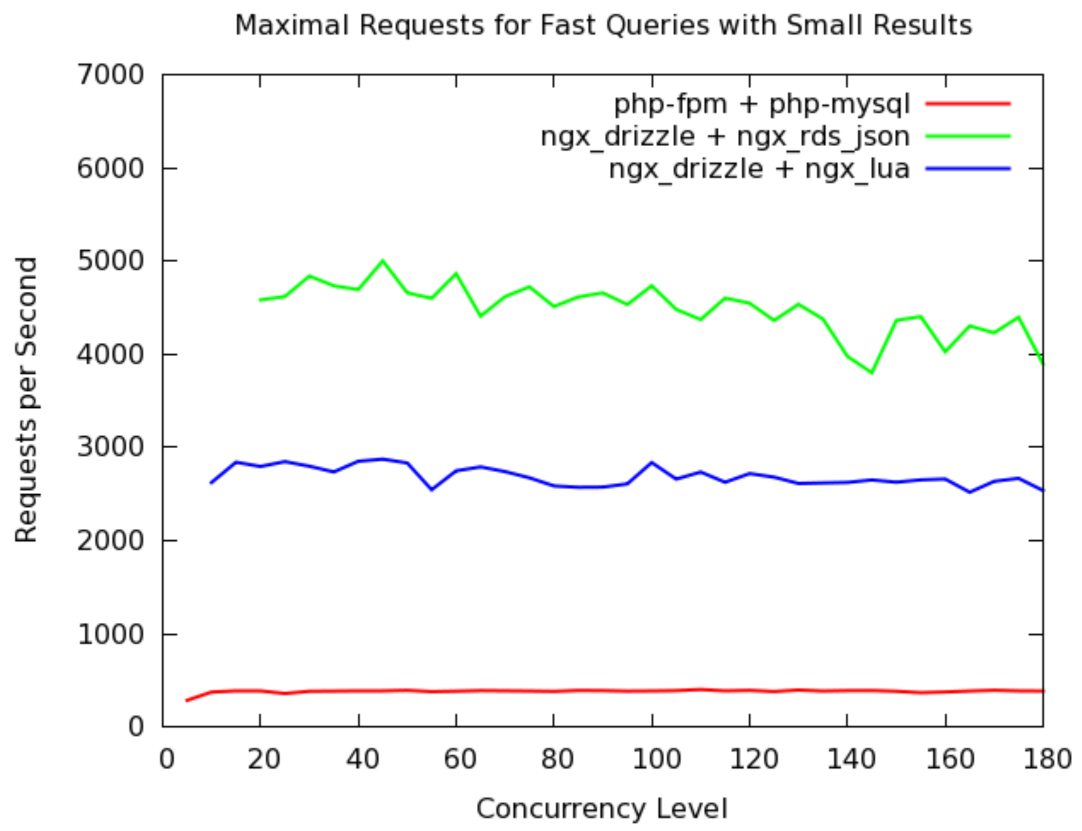
Nginx Subrequest Model

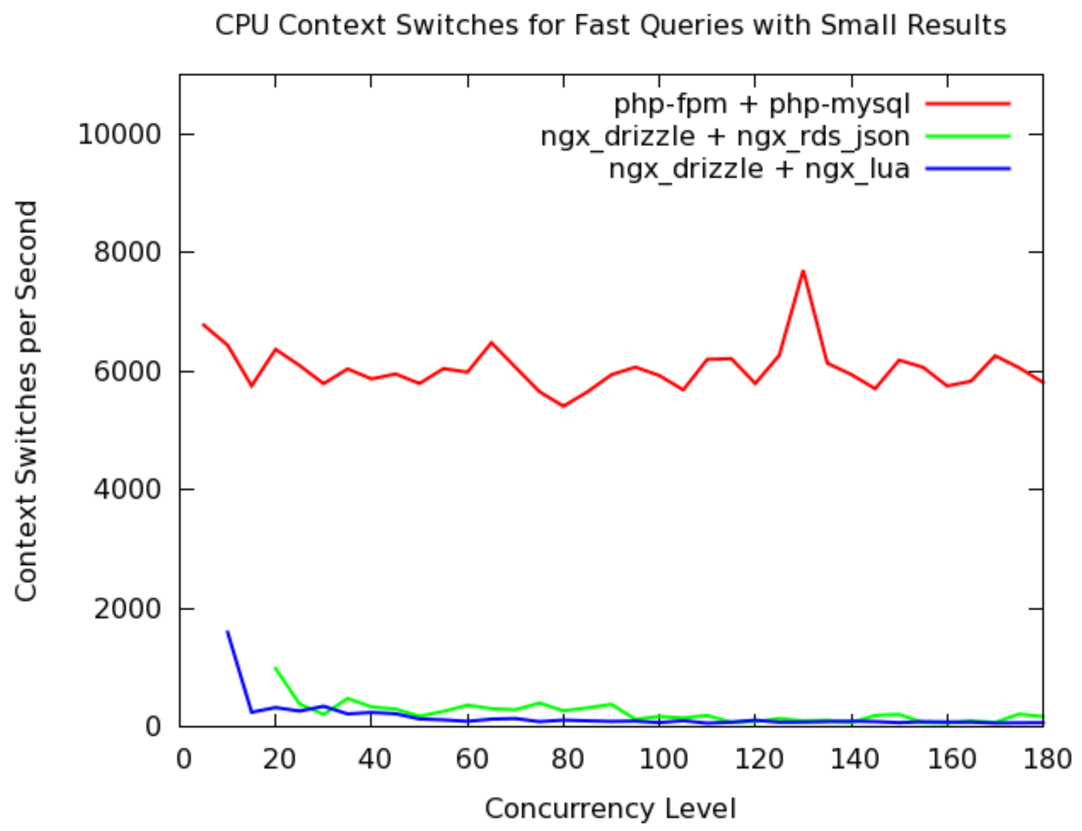
```
location = /api {
    content_by_lua '
        local rds_parser = require "rds.parser"
        local cjson = require "cjson"

        local resp = ngx.location.capture("/cat/Jerry")
        local data, err = rds_parser.parse(res.body)
        ngx.print(cjson.encode(data.resultset))
    ';
}
```

```
$ curl 'http://localhost/api'
[{"name":"Jerry","age":1}]
```

## ☺ The *Fast* MySQL Query with a Small Resultset Revisited!

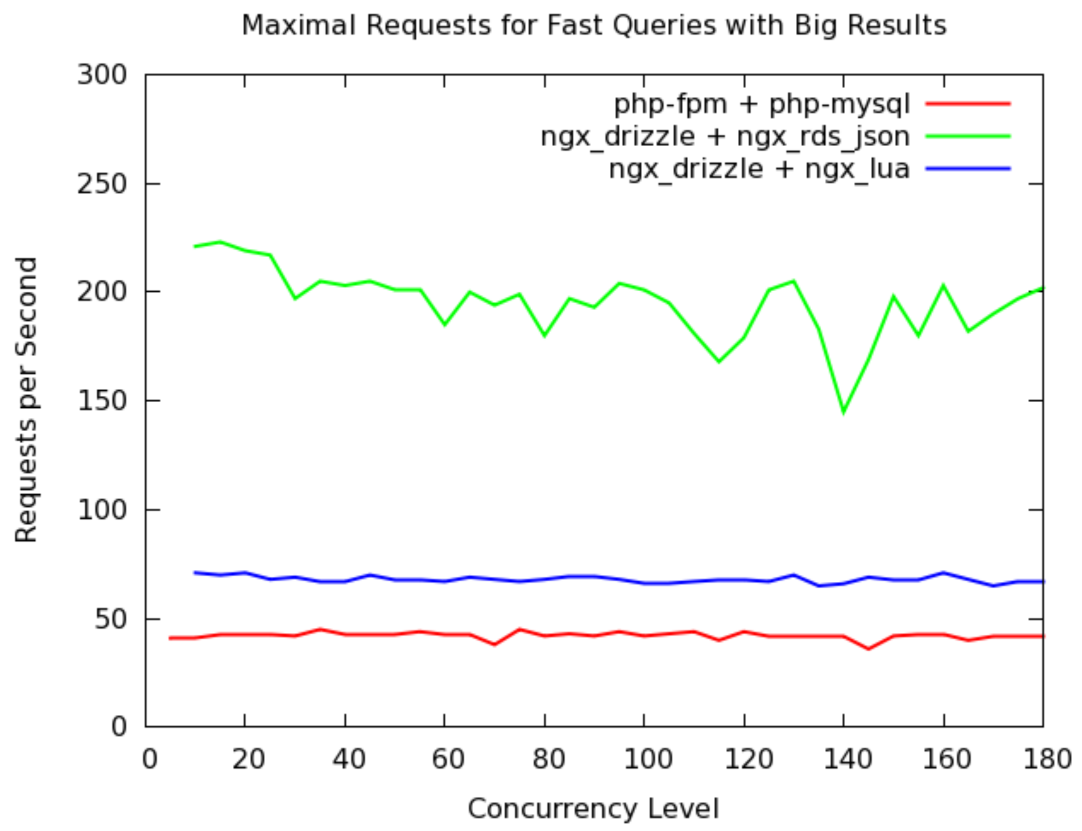
```
select *
from world.City
order by ID
limit 1
```

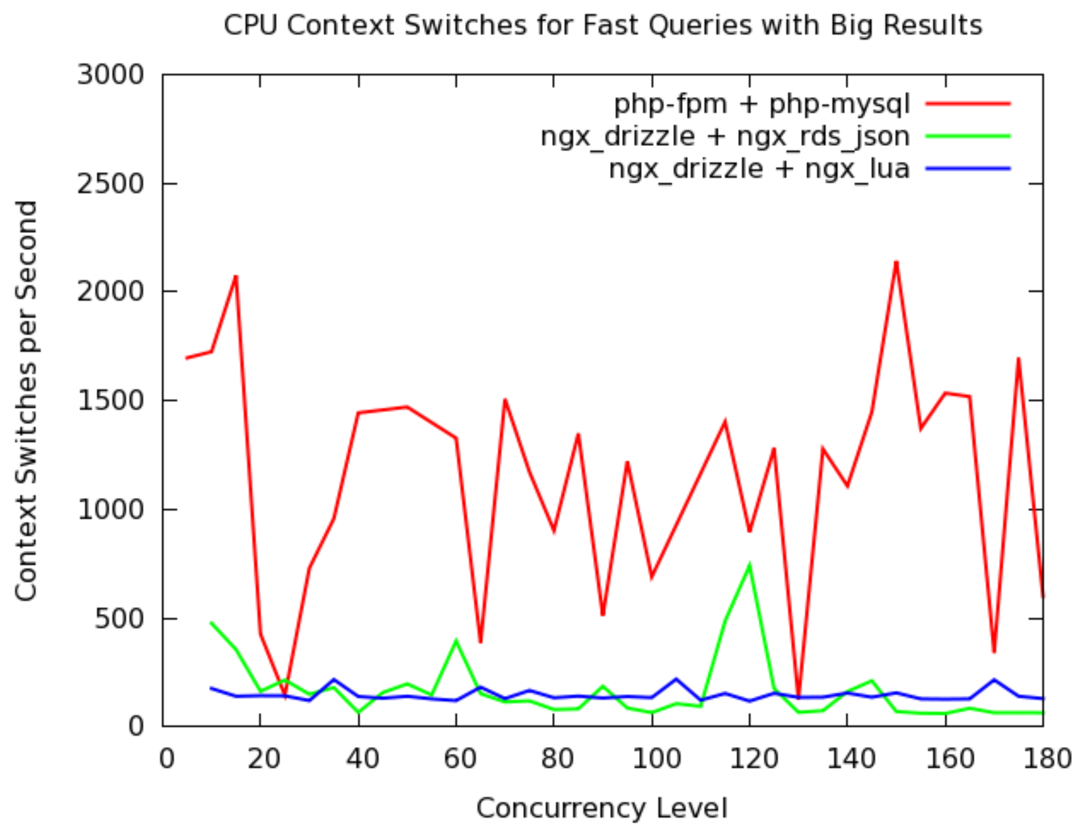

Maximal Requests for Fast Queries with Small Results
Requests per Second
Concurrency Level
0
1000
2000
3000
4000
5000
6000
7000
0
20
40
60
80
100
120
140
160
180
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua


CPU Context Switches for Fast Queries with Small Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
Context Switches per Second
Concurrency Level
0
2000
4000
6000
8000
10000
0
20
40
60
80
100
120
140
160
180

## ☺ The *Fast* MySQL Query with a Big Resultset (100 KBytes) Again!

```
select *
from world.City
order by ID
limit 1000
```


Maximal Requests for Fast Queries with Big Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
Requests per Second
Concurrency Level
0
50
100
150
200
250
300
0
20
40
60
80
100
120
140
160
180

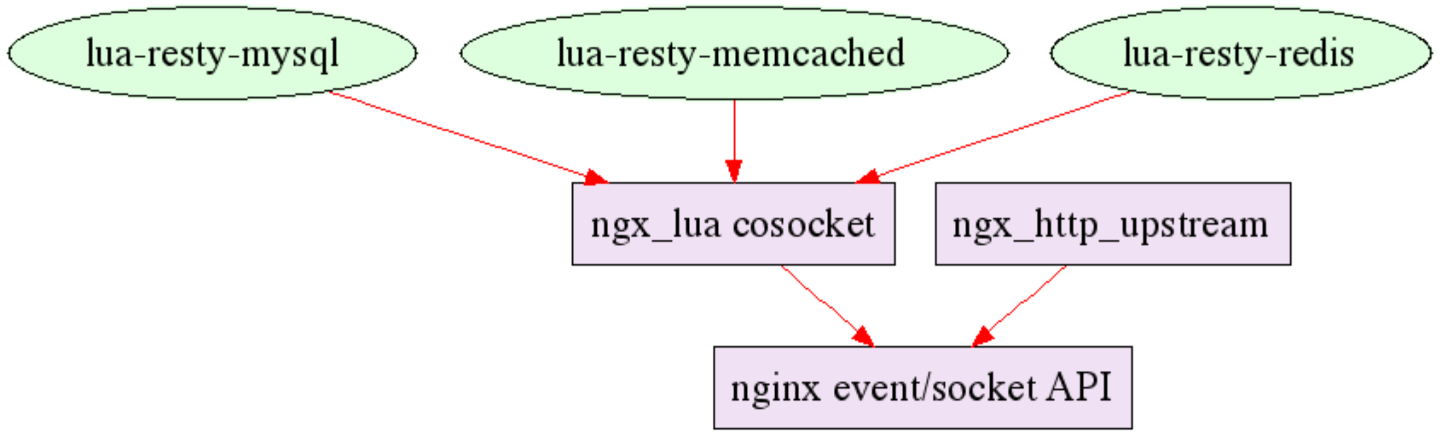


CPU Context Switches for Fast Queries with Big Results
Context Switches per Second
Concurrency Level
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
0
500
1000
1500
2000
2500
3000
0
20
40
60
80
100
120
140
160
180

☺ I just implemented the Lua *cosocket API*!

http://wiki.nginx.org/HttpLuaModule#ngx.socket.tcp

- ✓ a socket API based on Lua *coroutines*
- ✓ a socket API that is *synchronous*
- ✓ a socket API that is *nonblocking*



ngx_lua cosocket mechanism

☺ I wrote the lua-resty-mysql library based on the *cosocket* API.

http://github.com/agentzh/lua-resty-mysql

☺ It is a *pure Lua* MySQL driver written from scratch!

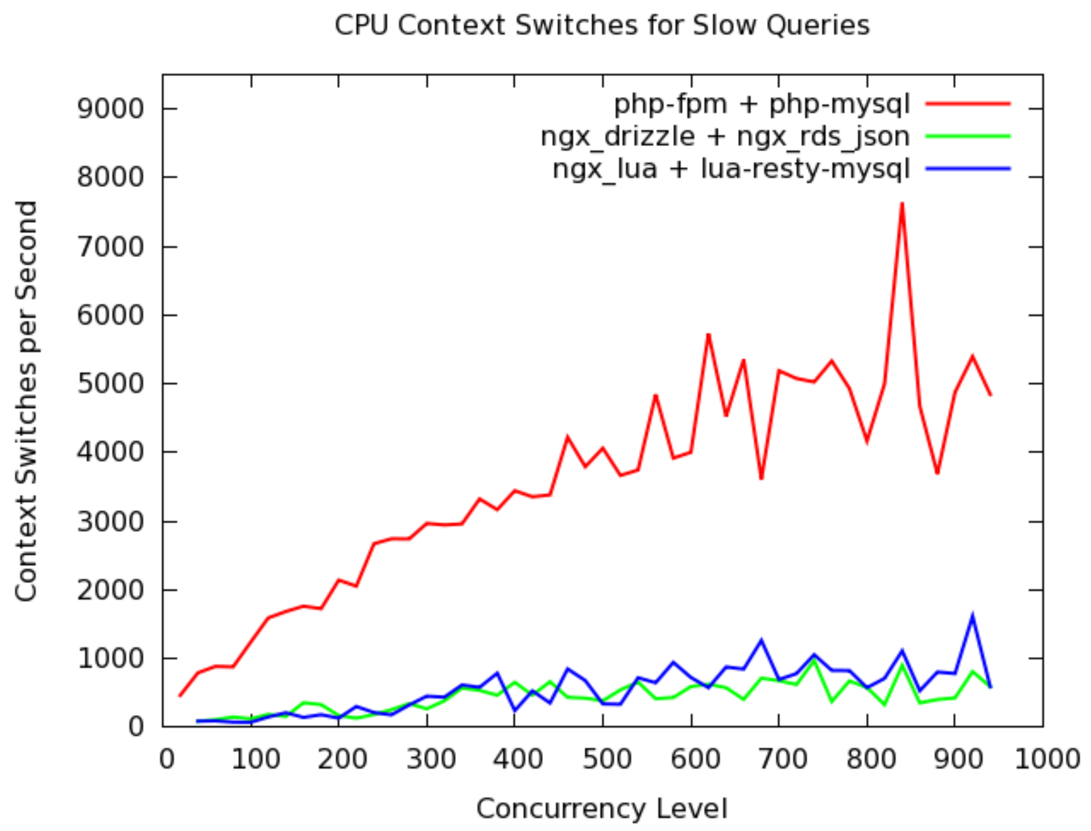
```lua
local resty_mysql = require "resty.mysql"

local mysql = resty_mysql:new()

local ok, err = mysql:connect{
    host = "127.0.0.1",
    port = 3306,
    database = "world",
    user = "monty",
    password = "some_pass"
}
```

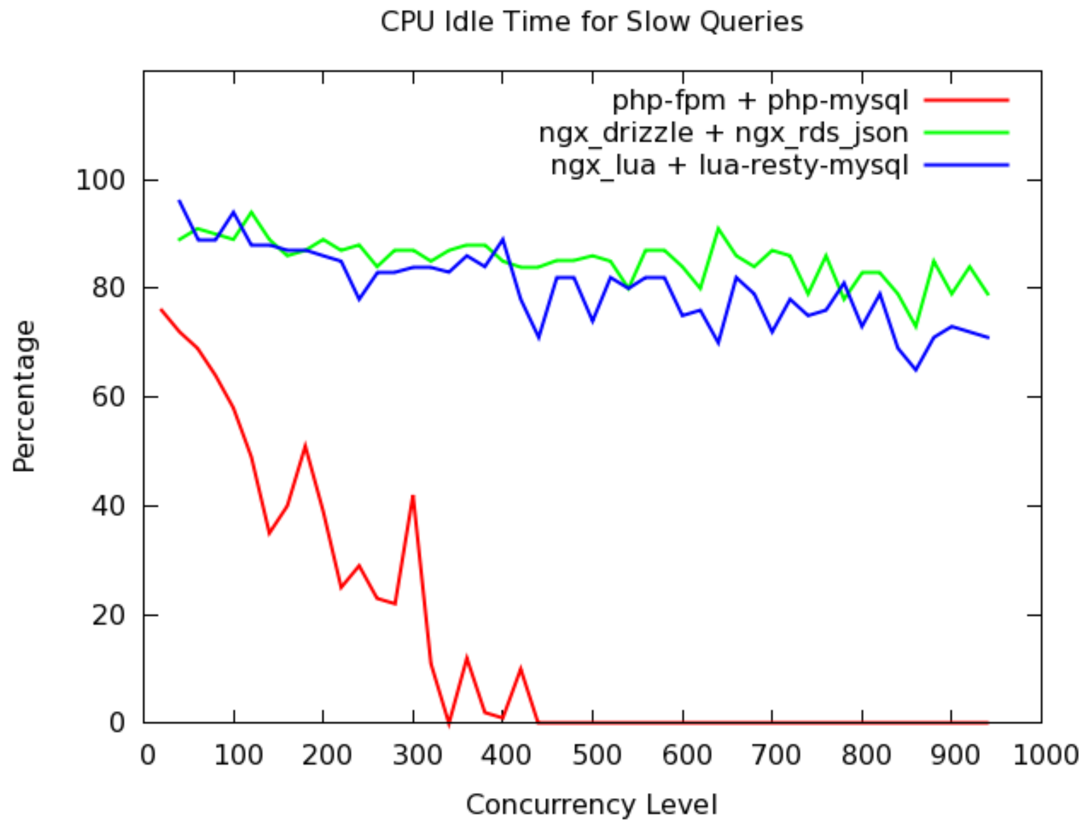
```lua
local query = "select * from cats"

local rows, err, errno, sqlstate =
    mysql:query(query)

for i, row in ipairs(rows) do
    -- process the row table
end
```

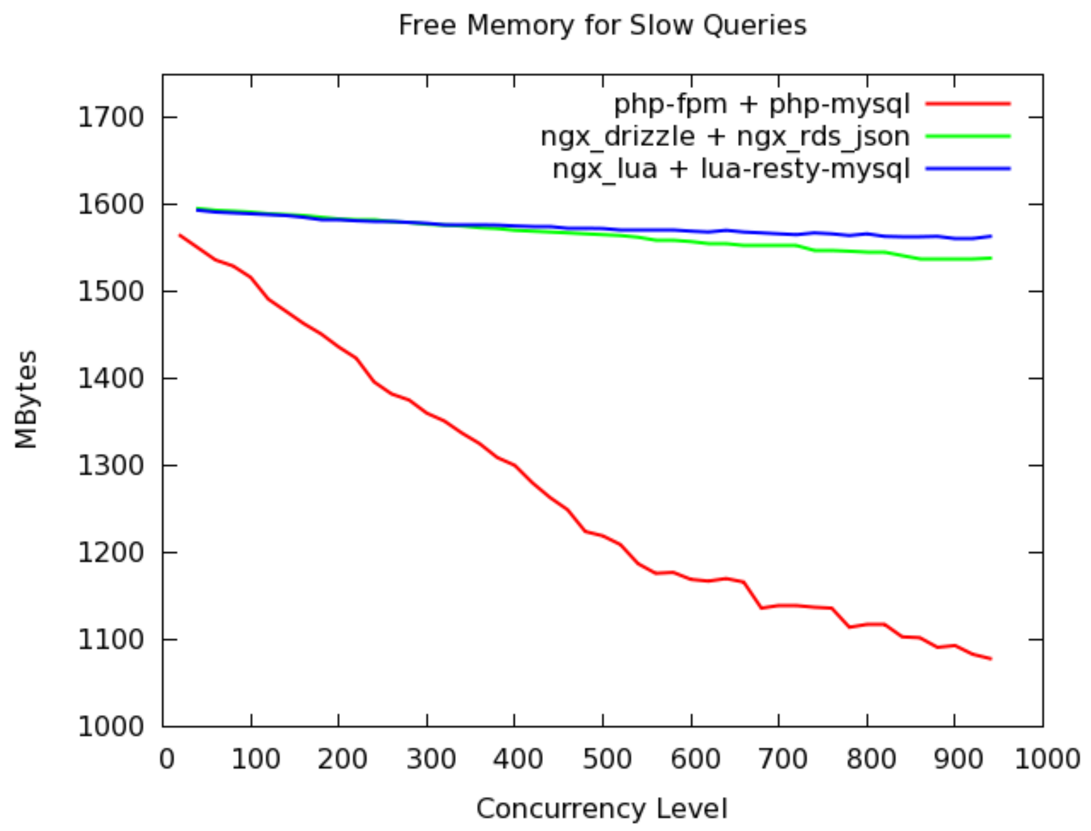
```lua
-- add the current MySQL connection
-- into the per-worker connection pool,
-- with total capacity of 1024 connections and
-- 60 seconds maximal connection idle time

local ok, err = mysql:set_keepalive(60000, 1024)
```

☺ The *Slow* MySQL Query Revisited!

```
select sleep(1)
```

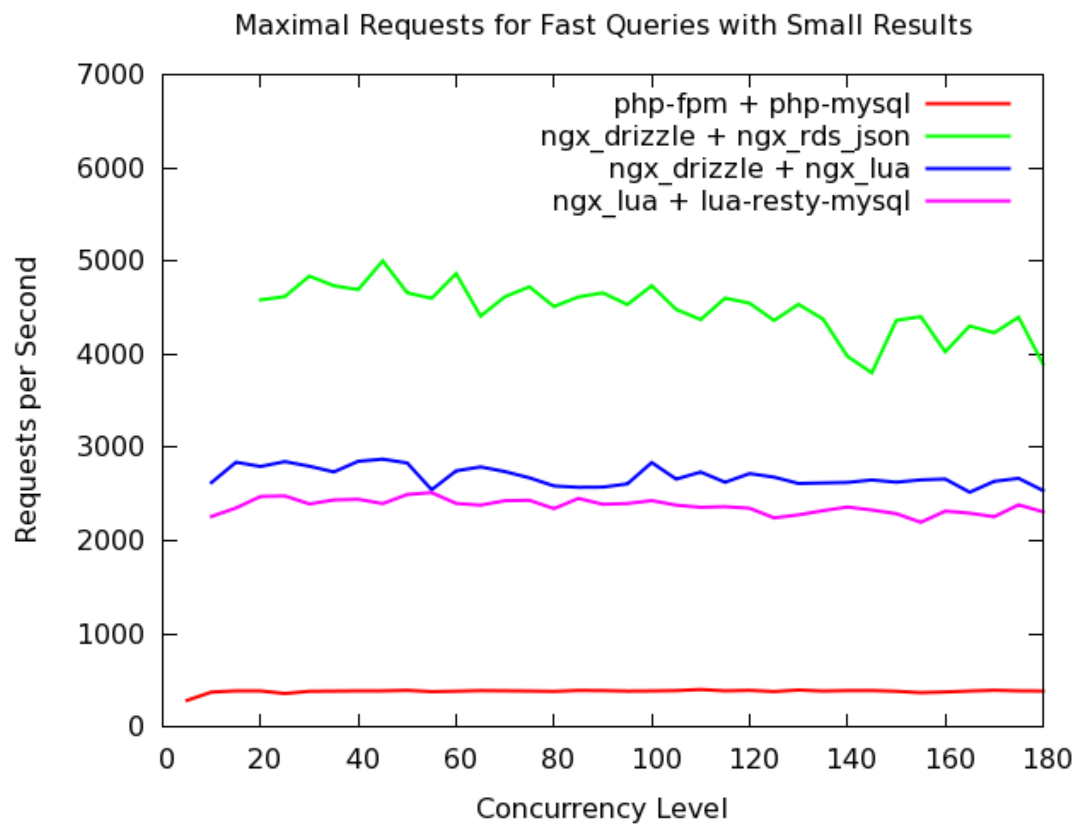

CPU Context Switches for Slow Queries
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_lua + lua-resty-mysql
Context Switches per Second
Concurrency Level
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
100
200
300
400
500
600
700
800
900

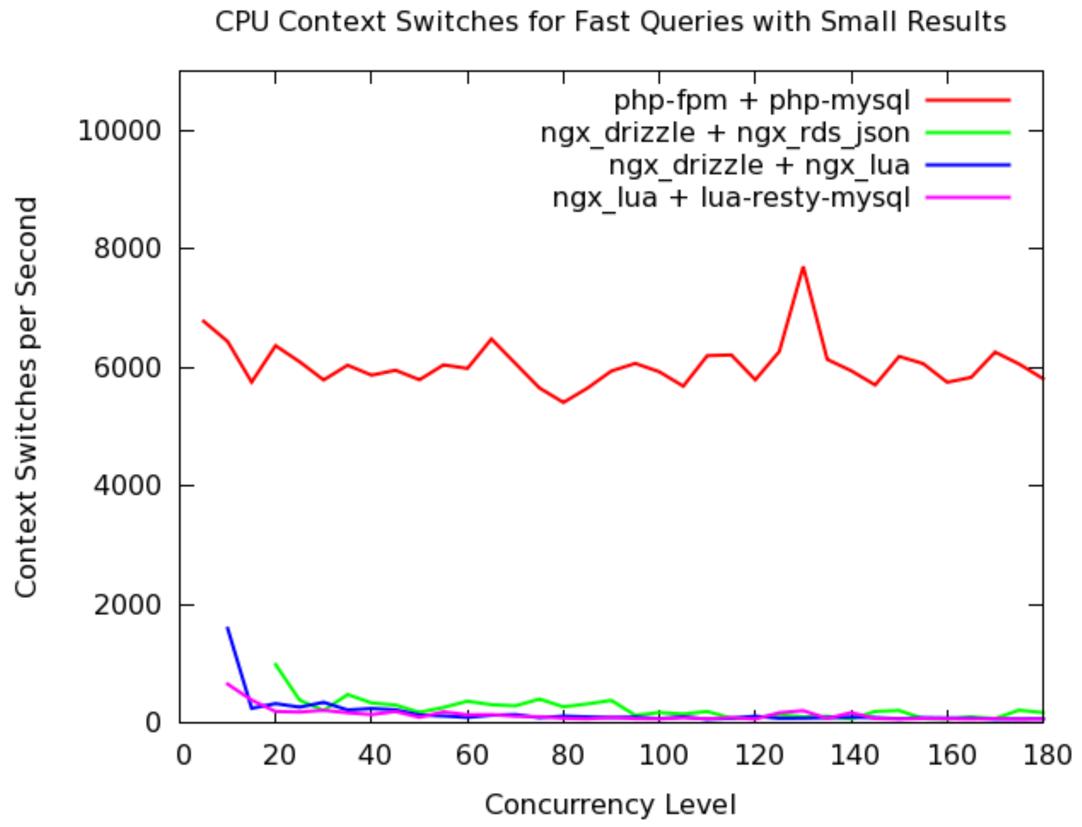

CPU Idle Time for Slow Queries
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_lua + lua-resty-mysql
Percentage
0
20
40
60
80
100
0
100
200
300
400
500
600
700
800
900
1000
Concurrency Level


Free Memory for Slow Queries
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_lua + lua-resty-mysql
MBytes
1000
1100
1200
1300
1400
1500
1600
1700
0
100
200
300
400
500
600
700
800
900
1000
Concurrency Level

## ☺ The *Fast* MySQL Query with a Small Resultset Revisited!

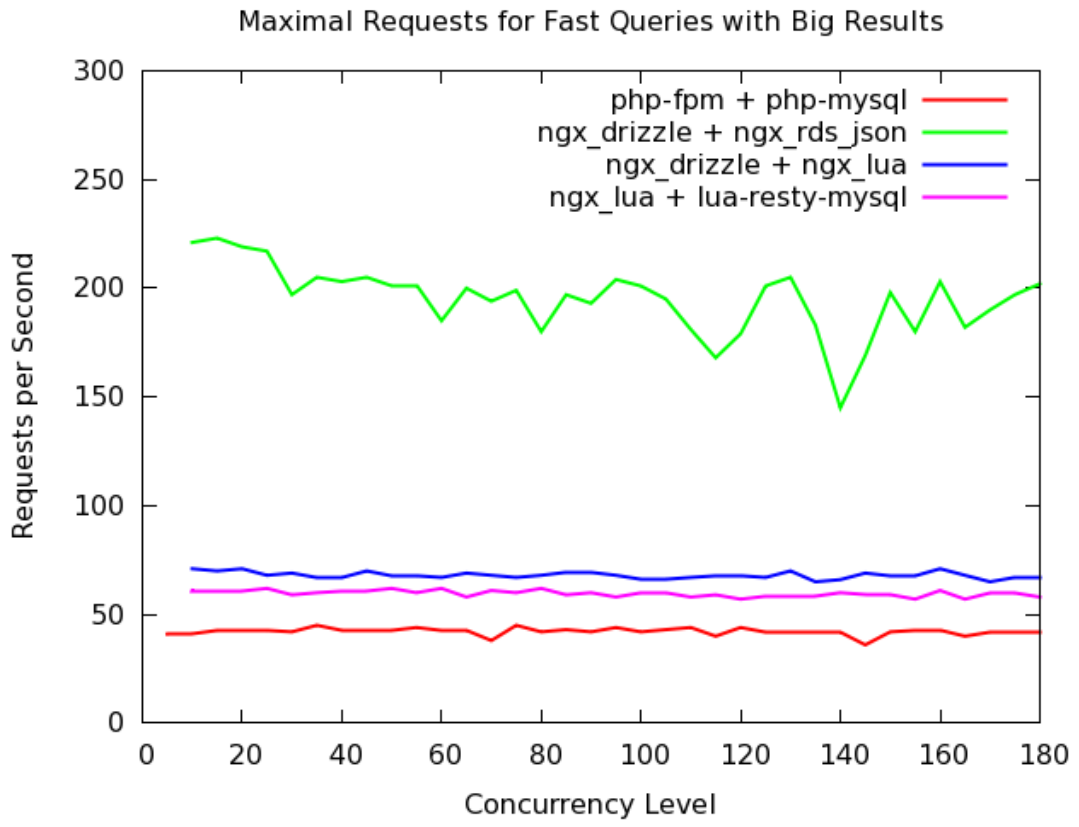
```
select *
from world.City
order by ID
limit 1
```

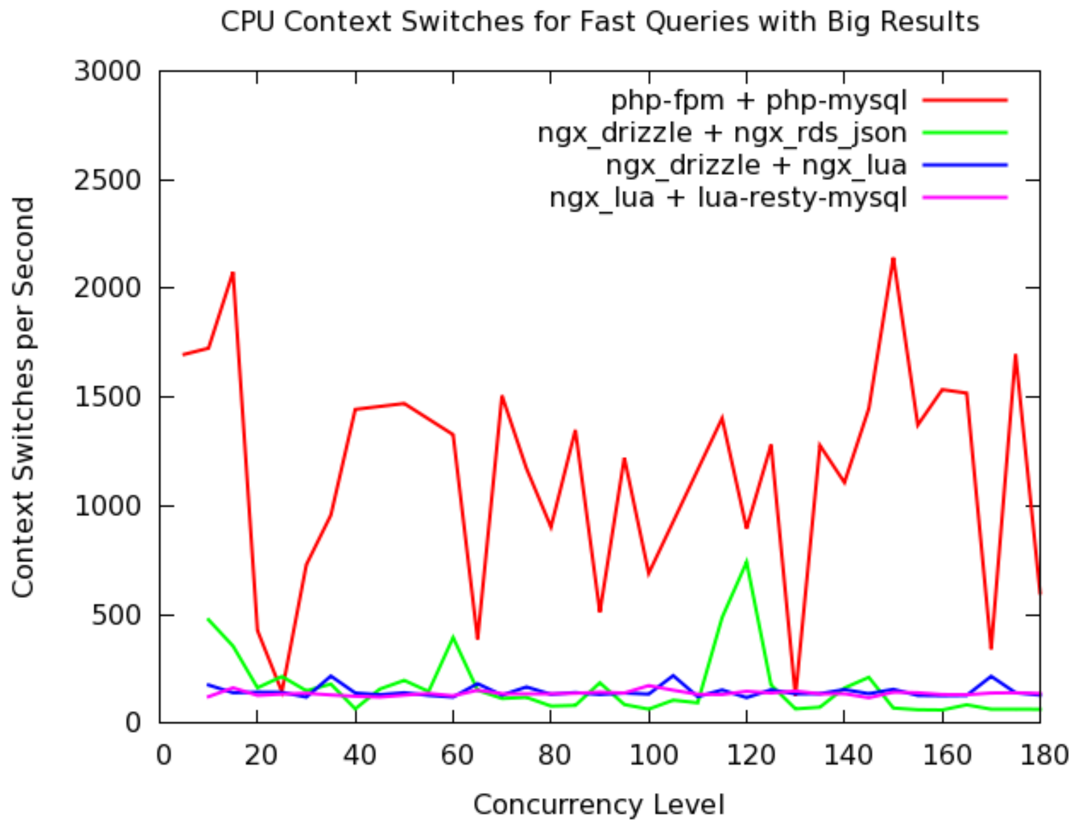

Maximal Requests for Fast Queries with Small Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
Requests per Second
Concurrency Level
0
1000
2000
3000
4000
5000
6000
7000
0
20
40
60
80
100
120
140
160
180


CPU Context Switches for Fast Queries with Small Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
Context Switches per Second
10000
8000
6000
4000
2000
0
0
20
40
60
80
100
120
140
160
180
Concurrency Level

## ☺ The *Fast* MySQL Query with a Big Resultset (100 KBytes) Revisited!

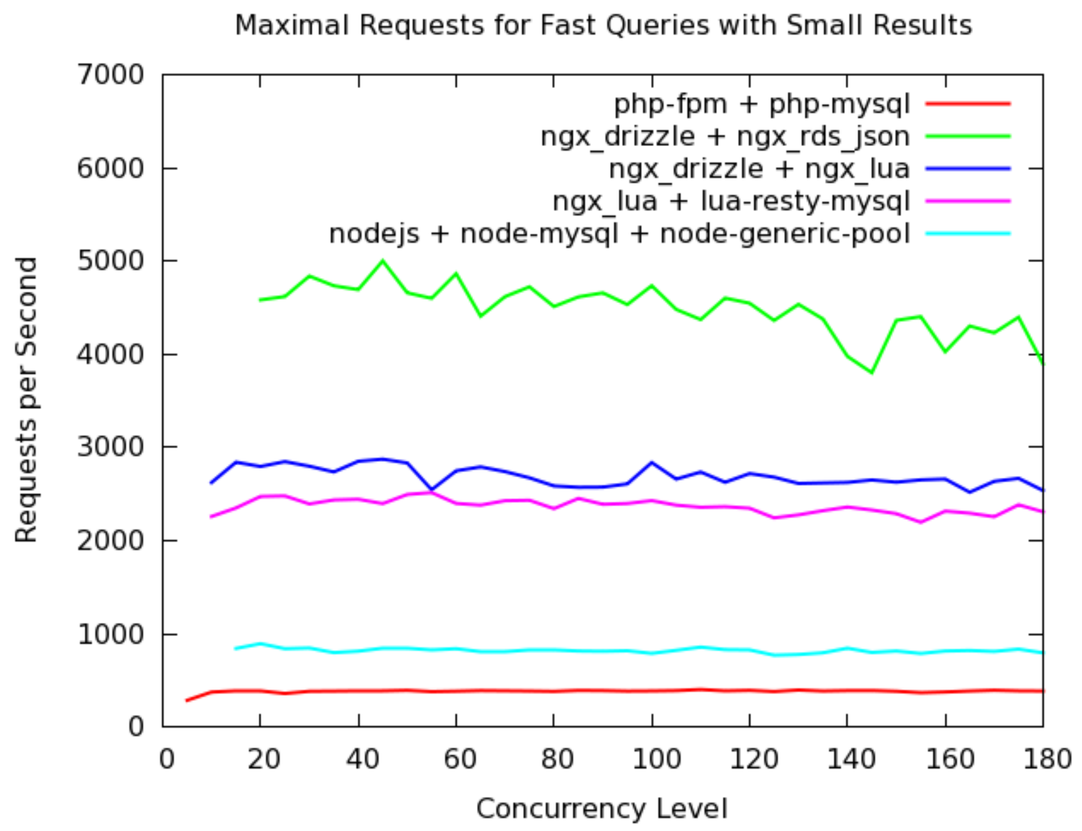
```
select *
from world.City
order by ID
limit 1000
```

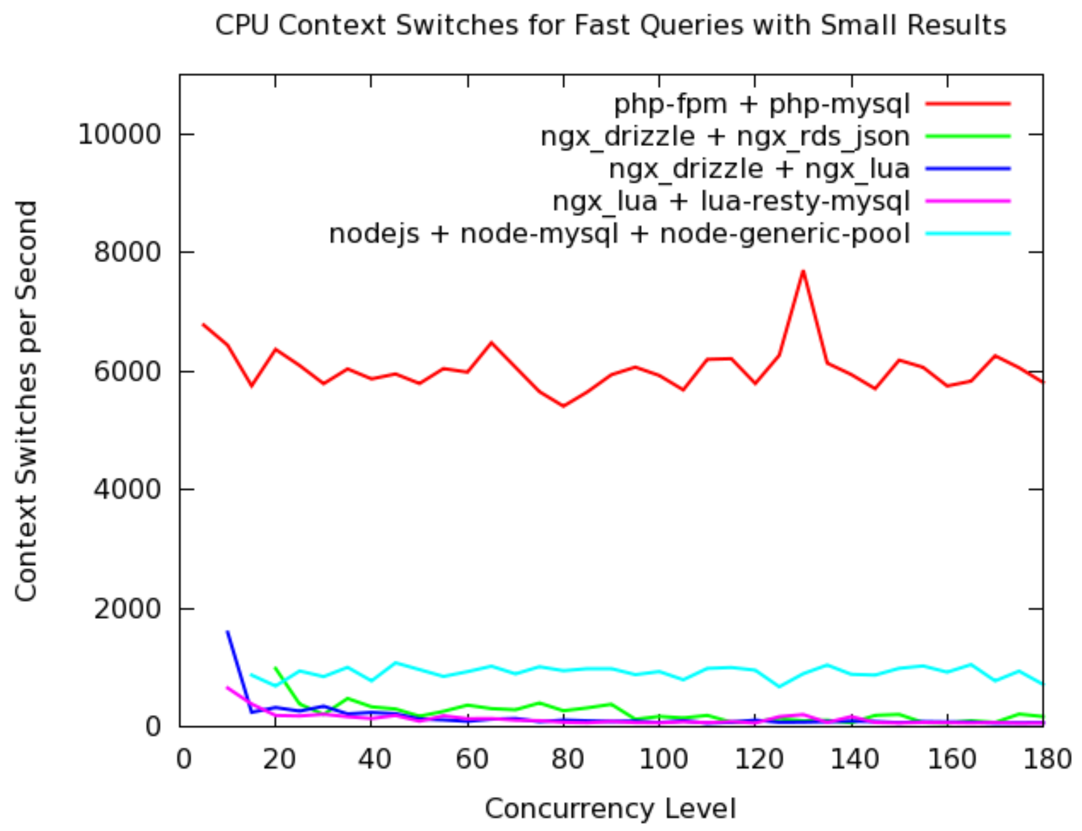

Maximal Requests for Fast Queries with Big Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
Requests per Second
Concurrency Level
0
50
100
150
200
250
300
0
20
40
60
80
100
120
140
160
180

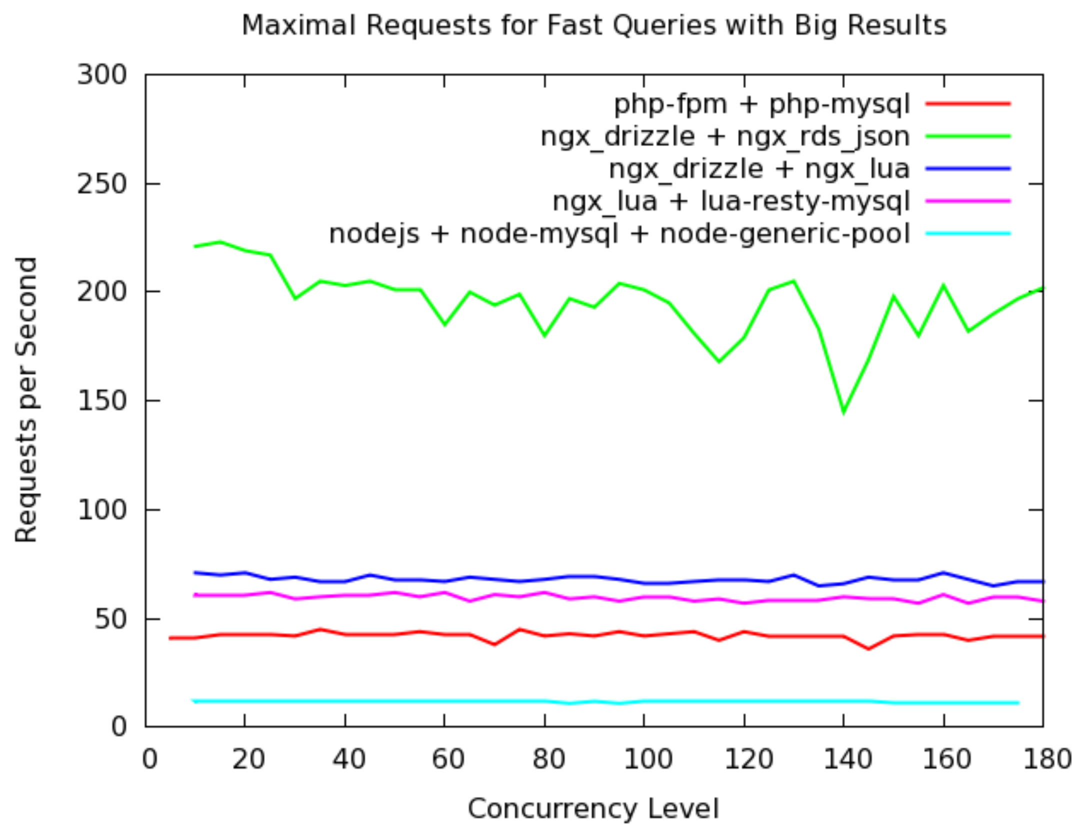

CPU Context Switches for Fast Queries with Big Results
Context Switches per Second
Concurrency Level
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
0
500
1000
1500
2000
2500
3000
0
20
40
60
80
100
120
140
160
180

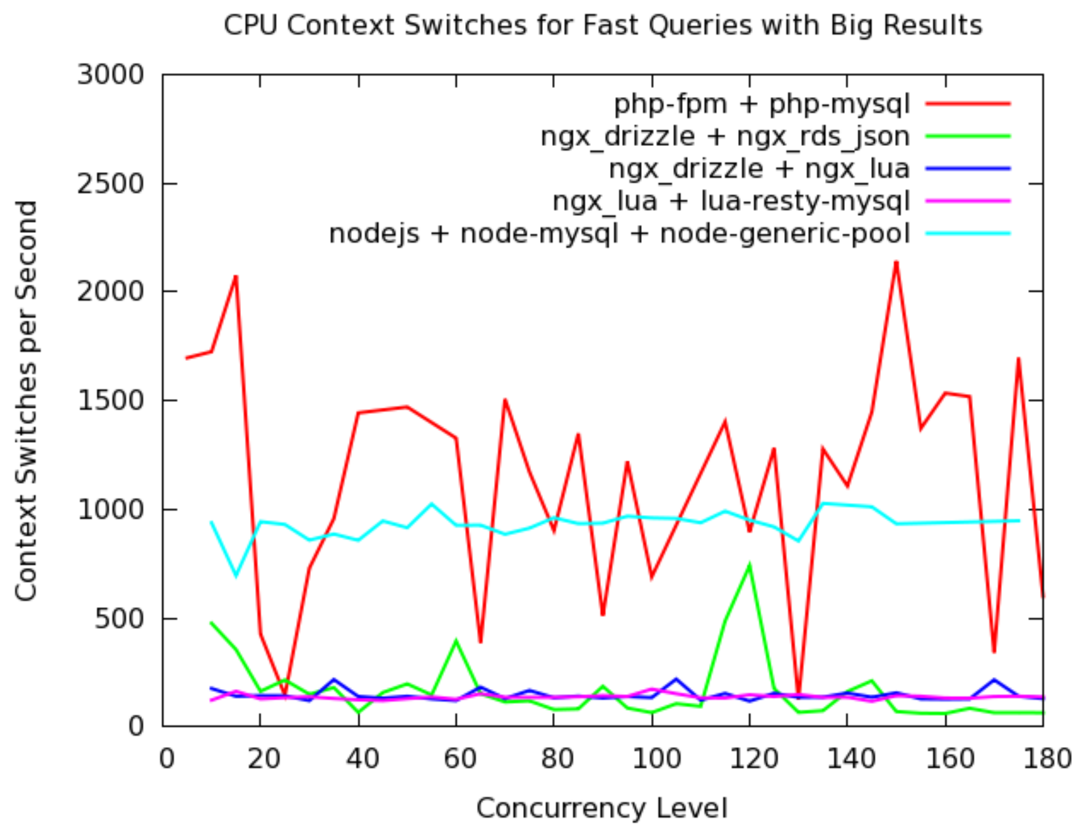
☺ How about *comparing* with the NodeJS world?

- ♡ node *0.6.14*
- ♡ node mysql *0.9.5*
- ♡ node generic pool *1.0.9*


Maximal Requests for Fast Queries with Small Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
nodejs + node-mysql + node-generic-pool
Requests per Second
Concurrency Level
0
1000
2000
3000
4000
5000
6000
7000
0
20
40
60
80
100
120
140
160
180

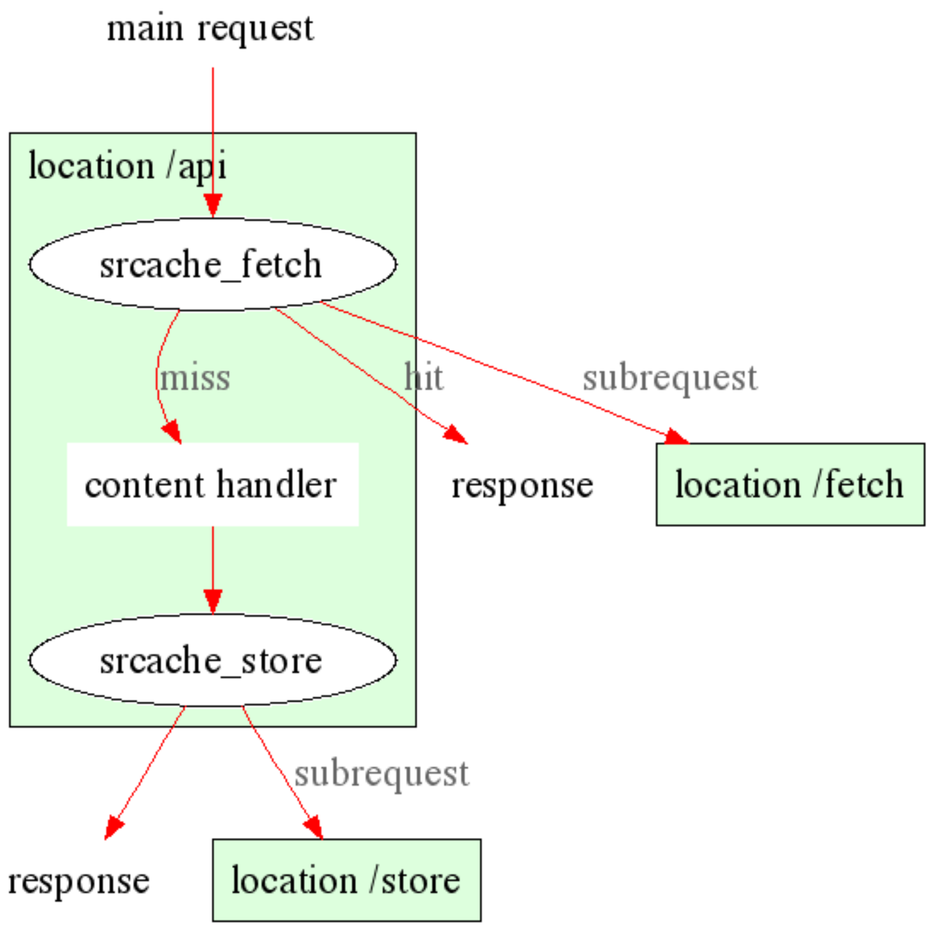

CPU Context Switches for Fast Queries with Small Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
nodejs + node-mysql + node-generic-pool
Context Switches per Second
Concurrency Level
0
2000
4000
6000
8000
10000
20
40
60
80
100
120
140
160
180


Maximal Requests for Fast Queries with Big Results
Requests per Second
Concurrency Level
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
nodejs + node-mysql + node-generic-pool
0
50
100
150
200
250
300
0
20
40
60
80
100
120
140
160
180


CPU Context Switches for Fast Queries with Big Results
Context Switches per Second
Concurrency Level
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
nodejs + node-mysql + node-generic-pool
0
500
1000
1500
2000
2500
3000
20
40
60
80
100
120
140
160
180

☺ *Caching* responses with
*ngx_srcache* + *ngx_memc*

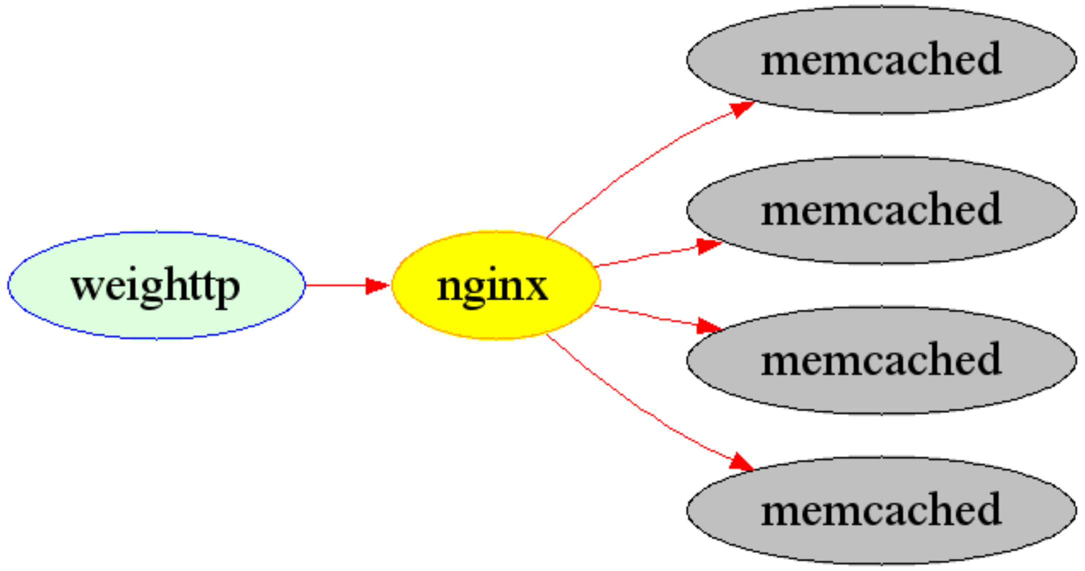
http://wiki.nginx.org/HttpSRCacheModule
http://wiki.nginx.org/HttpMemcModule



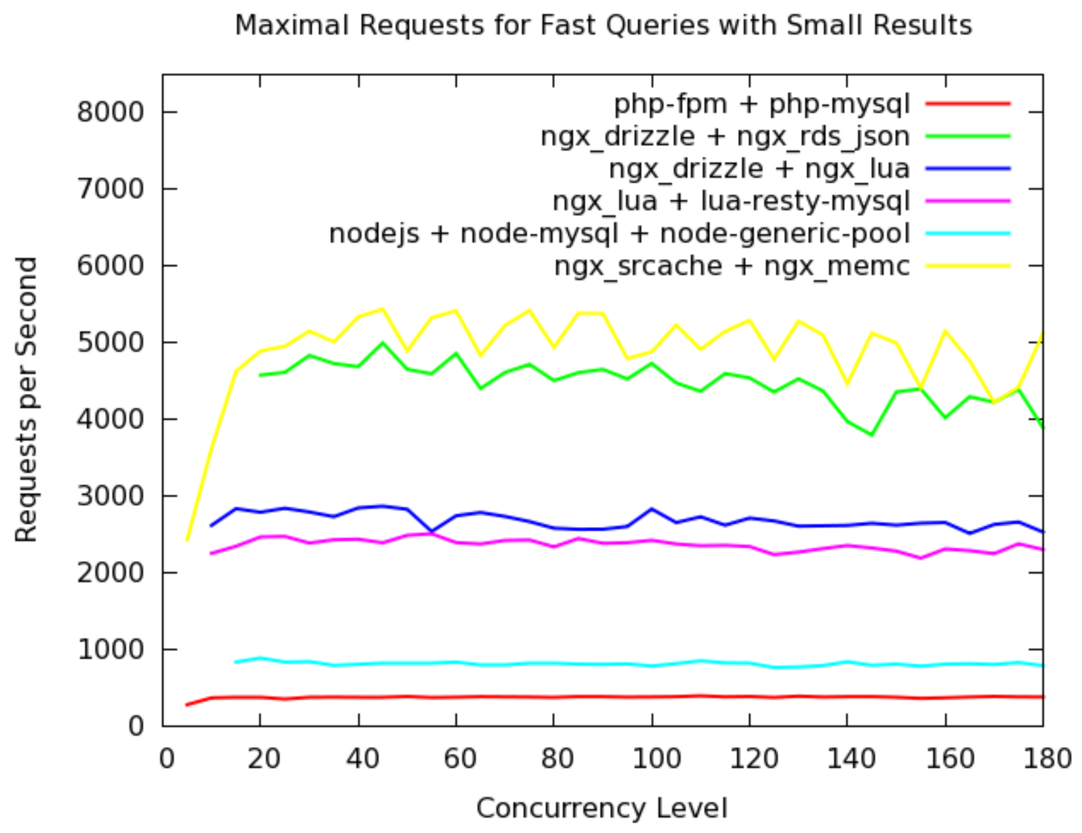
The ngx_srcache Module's Workflow

```
# configure the cache storage location
location /memc {
    internal;

    set $memc_key $query_string;
    set $memc_exptime 300;

    memc_pass 127.0.0.1:11211;
}
```

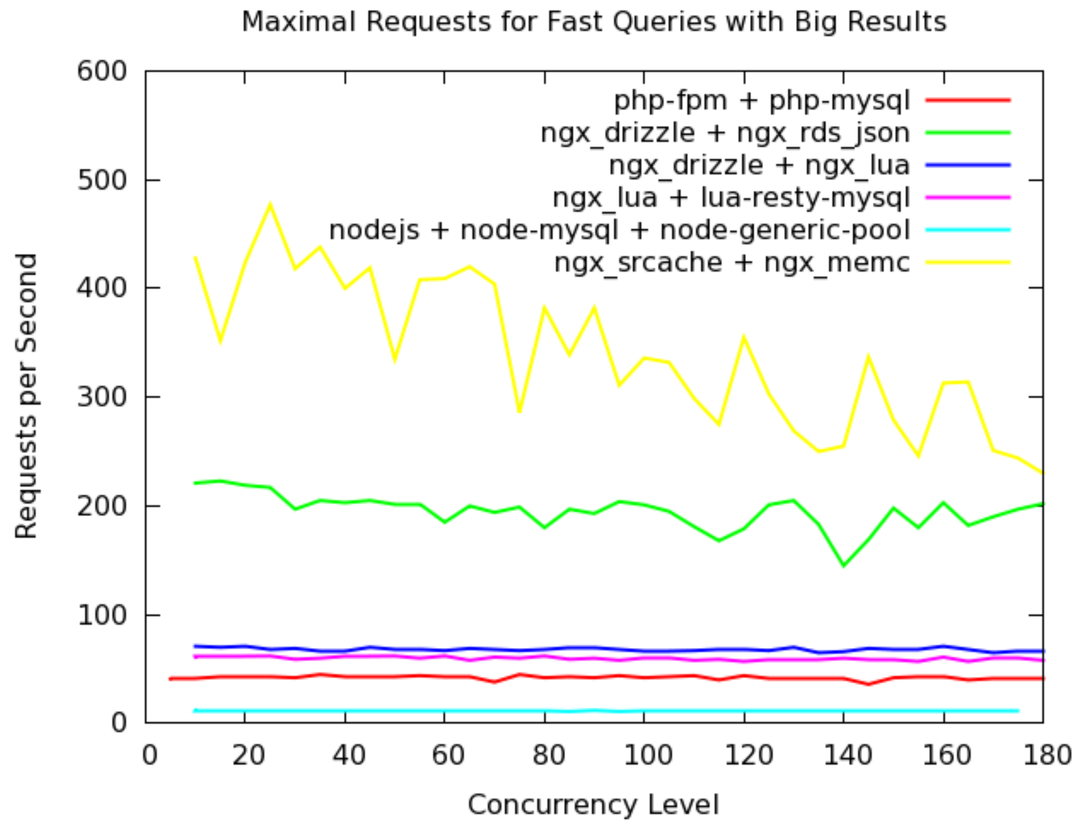
```
location = /api {

    set $key "$uri?$args";

    srcache_fetch GET /memc $key;
    srcache_store PUT /memc $key;

    # drizzle_pass/fastcgi_pass/content_by_lua/...
}
```



A Test Cluster of Amazon EC2 Small Instances (Using ngx_srcache + ngx_memc)



Maximal Requests for Fast Queries with Small Results
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
nodejs + node-mysql + node-generic-pool
ngx_srcache + ngx_memc
Requests per Second
Concurrency Level
0
1000
2000
3000
4000
5000
6000
7000
8000
0
20
40
60
80
100
120
140
160
180



Maximal Requests for Fast Queries with Big Results
Requests per Second
Concurrency Level
php-fpm + php-mysql
ngx_drizzle + ngx_rds_json
ngx_drizzle + ngx_lua
ngx_lua + lua-resty-mysql
nodejs + node-mysql + node-generic-pool
ngx_srcache + ngx_memc
0
100
200
300
400
500
600
0
20
40
60
80
100
120
140
160
180

☺ Find the *source* for
all the benchmarks given here:

http://github.com/agentzh/mysql-driver-benchmark

☺ *Any questions*? ☺

http://openresty.org

https://groups.google.com/group/openresty