Scripting libdrizzle with Lua inside Nginx

Xiaozhe Wang
chaoslawful@gmail.com
@chaoslawful

Business Background

Background

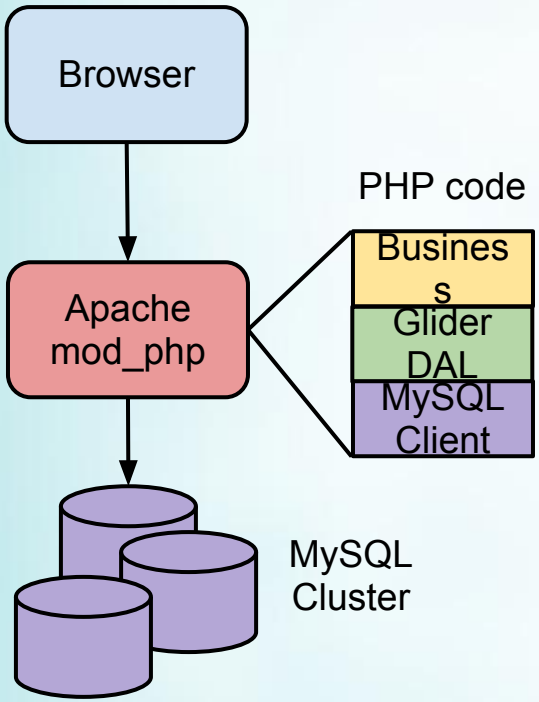
- Linezing data analysis product
  - Provides consumer/sales analysis service for Taobao shops
  - Mainly presents various data reporting pages to end-user
  - Rapidly grows up:
    - Currently **2.3M active shops**, **3k more per day**
    - TCP conns(/min/box): **3.75k/62.69k/109k** (min/avg/max)
    - Daily PV per box: **>30M**
    - Only **4 FE boxes** in total

Background

- Linezing response data features:
  - Invariant reporting page outline
  - Slow DB queries existed: max **3s** execution time
  - Relative large response size: max **>200KB**
  - Few repeat queries: Seller query is independent

Architecture Evolution

Architecture (original)



Pros

- Std LAMP, well understood
- Easy to learn for newcomers

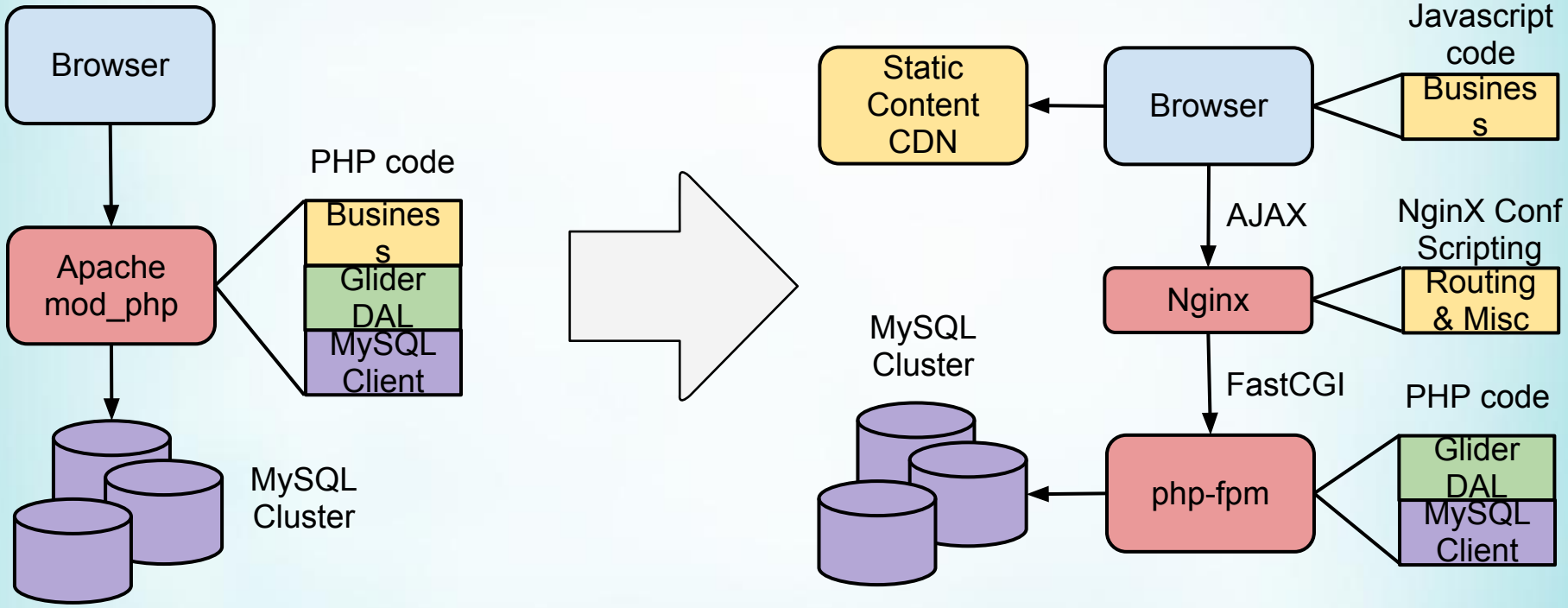
Cons

- Repeatedly transfer page outline
- Limited concurrency
- Slow-connection attack risk
- PHP is slow for large data processing

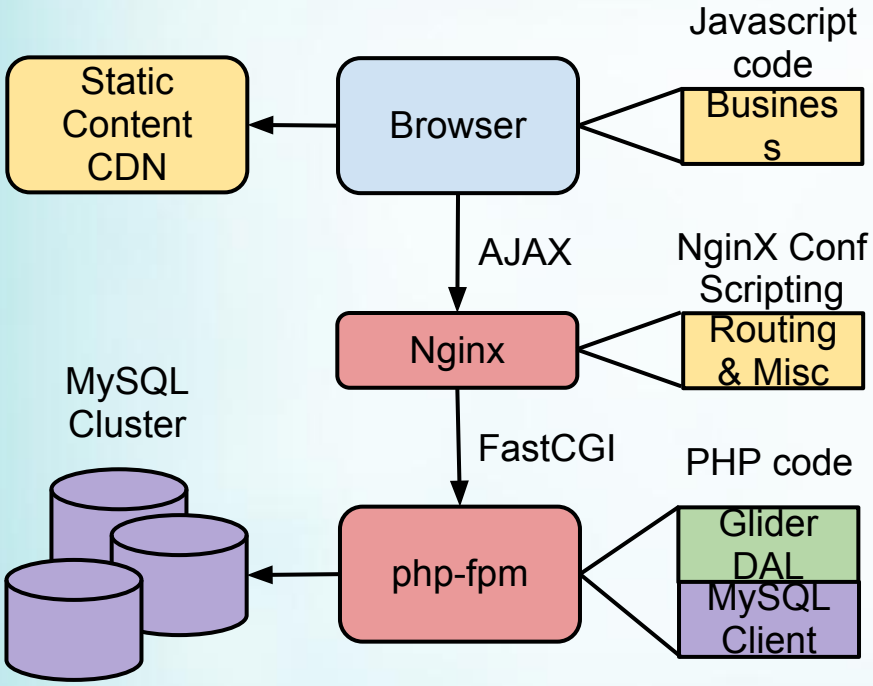
Solution

- Using different web serving paradigm
  - Blocking process/thread v.s. Event driven FSM
- Put invariant resources to CDN
- Filling page content at browser-side instead of server-side
  - Implement business logic using Javascript
  - AJAX communication

Evolution


Browser
Apache
mod_php
PHP code
Busines
s
Glider
DAL
MySQL
Client
MySQL
Cluster
Static
Content
CDN
Browser
Javascript
code
Busines
s
AJAX
NginX Conf
Scripting
Nginx
Routing
& Misc
MySQL
Cluster
FastCGI
PHP code
php-fpm
Glider
DAL
MySQL
Client

Architecture (improved)



Pros

- Saved bandwith
- Reduced server load
- Improved concurrency
- Immune to slow-connection attack

Cons

- Less productive on developing
- PHP/FastCGI restricted overall throughput

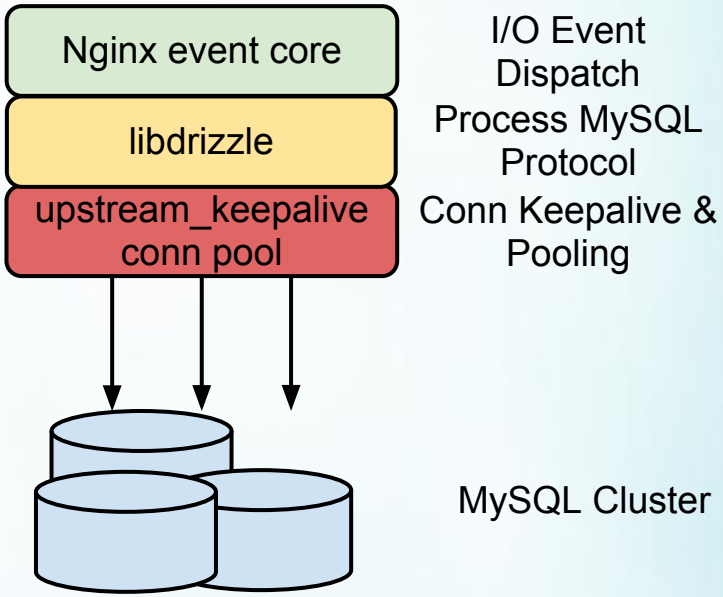
Solution

- Need efficient accessing to MySQL in Nginx
  - **ngx_drizzle was developed!**

- Need productive programming language fitting Nginx I/O model
  - **ngx_lua was developed!**

ngx_drizzle intro

- Accessing MySQL in Nginx in non-blocking synchronous manner
- Keepalive, pooling connections with loadbalancing
- Convert query result to JSON/CSV directly through **ngx_rds_json/csv** modules



ngx_drizzle example


```
http {
    ...
    upstream dbgroup {
        drizzle_server host1:3306 dbname=test password=some_pass user=alice protocol=mysql;
        drizzle_server host2:3306 dbname=test2 password=some_pass user=bob protocol=mysql;
    }
    ...
    server {
        location /mysql {
            set $sql "select * from cats";
            drizzle_query $sql;
            drizzle_pass dbgroup;
            rds_json on;
        }
    }
}
```

ngx_lua motivation

Business logic built solely on Nginx C-modules?

- Developing Nginx C-modules is counter-productive
  - Hard to write, hard to maintain.

- Deploying Nginx C-modules is inflexible
  - Static-linking with Nginx main program, impossible to customization while deploying.

ngx_lua intro

Everyone loves scripting language.

ngx_lua let us writing business logic in Lua!

Why Lua?

- Small memory footprint
- Excellent execution performance
- Interruptible VM with built-in coroutine support

chart reset

compare 2	|-	|---	25%	median	75%	---|	-|
C **GNU gcc**	1.00	1.00	1.04	**1.09**	1.35	1.83	3.80
C++ **GNU g++**	1.00	1.00	1.00	**1.11**	1.24	1.55	1.55
ATS	1.00	1.00	1.00	**1.26**	1.57	2.42	7.19
Java **6 steady state**	1.00	1.00	1.15	**1.57**	1.96	2.06	2.06
Lua **LuaJIT**	1.03	1.03	1.83	**1.98**	8.73	10.51	10.51
Scala	1.14	1.14	1.19	**2.00**	2.39	4.19	6.67
Fortran **Intel**	1.00	1.00	1.56	**2.18**	6.38	10.54	10.54
Java **6 -server**	1.10	1.10	1.36	**2.18**	3.63	6.80	6.80
Ada 2005 **GNAT**	1.00	1.00	1.40	**2.22**	3.66	7.05	7.44
Pascal **Free Pascal**	1.36	1.36	1.89	**2.35**	3.50	4.77	4.77
Haskell **GHC**	1.14	1.14	1.92	**2.38**	3.97	7.04	8.08
Clean	1.29	1.29	1.93	**2.67**	5.76	9.58	9.58
C# **Mono**	1.69	1.69	1.97	**2.84**	5.16	9.95	18.38
OCaml	1.49	1.49	2.52	**3.04**	3.87	4.62	4.62
F# **Mono**	1.93	1.93	1.99	**3.22**	3.95	6.88	10.54
Lisp **SBCL**	1.00	1.00	2.47	**3.29**	7.11	12.44	12.44
Racket	1.45	1.45	2.84	**4.38**	8.01	15.77	19.20
Go **6g 8g**	2.65	2.65	3.70	**4.66**	12.40	25.43	125.45
JavaScript **V8**	1.00	1.00	4.26	**7.19**	20.52	44.89	102.86
Erlang **HiPE**	1.64	1.64	5.60	**7.70**	20.91	34.93	34.93
Clojure	1.55	1.55	4.68	**11.47**	15.59	29.58	29.58
JavaScript **TraceMonkey**	1.73	1.73	4.39	**12.66**	28.53	64.75	898.17
Smalltalk **VisualWorks**	11.27	11.27	12.34	**15.73**	26.42	47.54	71.80
Java **6 -Xint**	7.37	7.37	14.91	**23.85**	32.36	58.53	72.81
Lua	1.03	1.03	21.82	**31.53**	41.70	51.54	51.54
Python **PyPy**	13.90	13.90	26.29	**32.29**	47.29	78.80	122.93
Ruby **JRuby**	16.50	16.50	23.95	**43.08**	153.45	225.45	225.45
Python **CPython**	2.28	2.28	5.74	**47.96**	93.11	106.47	106.47
Python 3	2.23	2.23	7.10	**50.09**	128.75	157.73	157.73
Python **IronPython**	20.41	20.41	33.78	**58.41**	86.05	164.46	190.75
Mozart/Oz	7.57	7.57	32.47	**63.37**	85.64	165.40	197.18
Ruby **1.9**	7.64	7.64	15.33	**63.66**	106.91	244.27	265.59
Perl	2.39	2.39	9.48	**73.69**	154.39	218.38	218.38
PHP	6.90	6.90	48.32	**104.15**	149.90	233.84	233.84
Ruby **MRI**	15.28	15.28	24.72	**158.27**	481.13	837.28	837.28

chart reset

ngx_lua overview

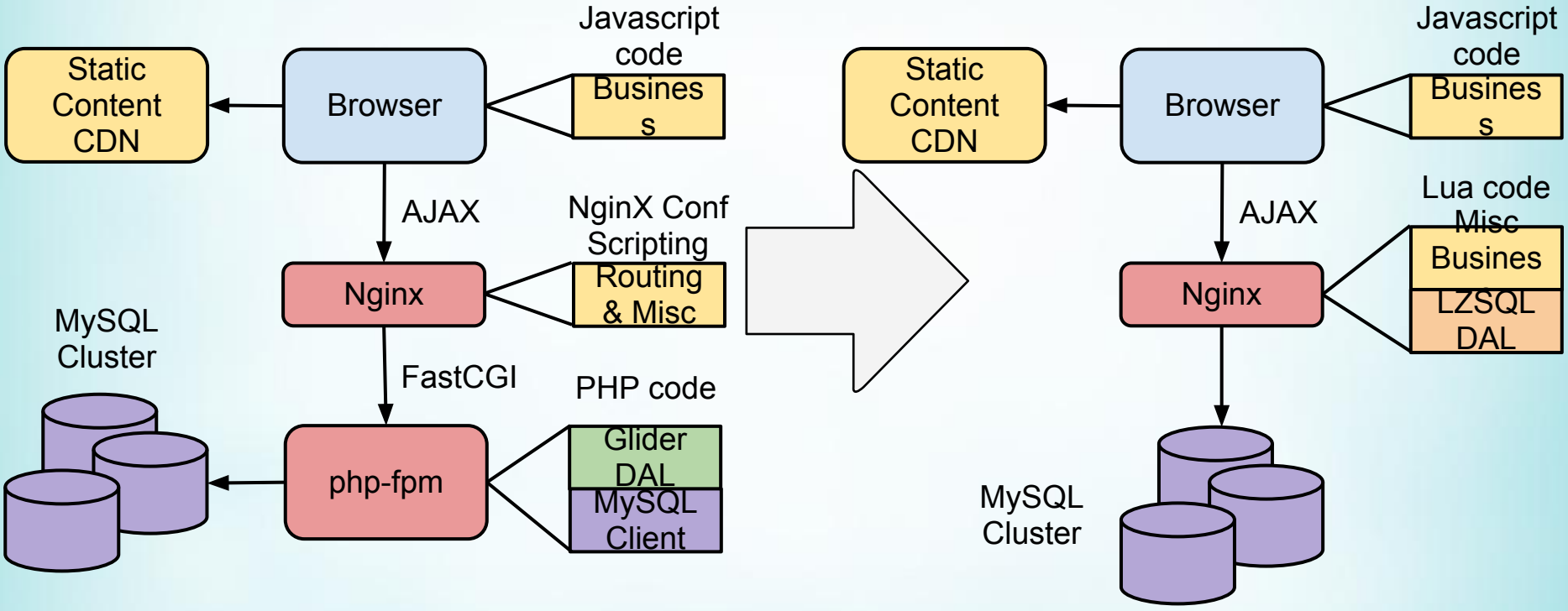
ngx_lua implemented the **proactor** pattern:

- Business logic is written in natural linear order
- Process resource would not be wasted on blocking I/O waiting
- Gain high concurrency capability automatically

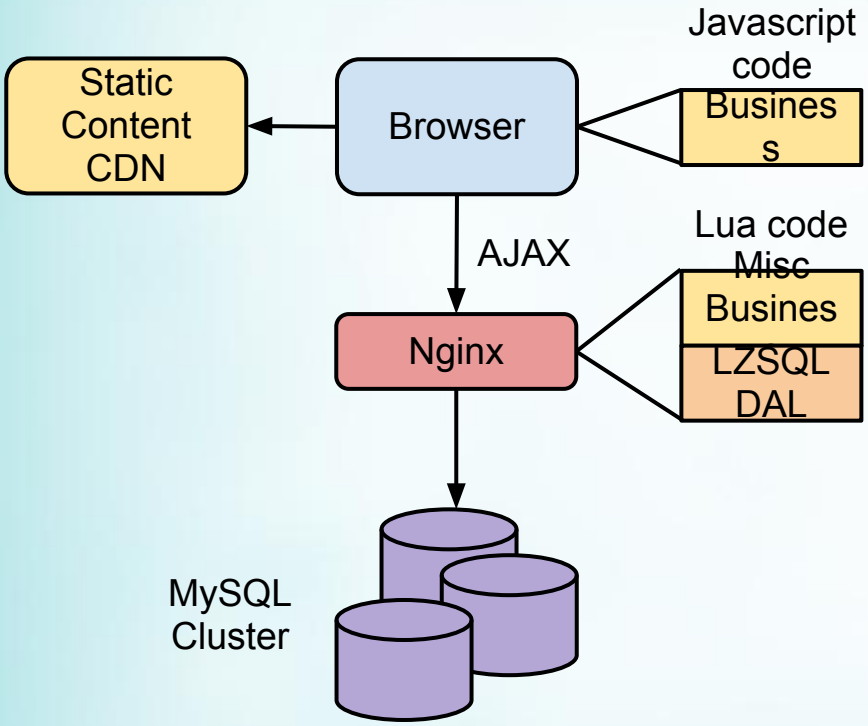
ngx_lua example

```
resolver ip.to.dns.server;
location /http_client {
    internal;
    proxy_pass $arg_url;
}
location /web_iconv {
    content_by_lua '
        local from, to, url = ngx.var.arg_f, ngx.var.arg_t, ngx.var.arg_u
        local capture = ngx.location.capture
        local iconv = require "iconv"
        local cd = iconv.new(to or "utf8", from or "gbk")
        local res = capture("/http_client?url=" .. url)
        if res.status == 200 then
            local ostr, err = cd:iconv(res.body)
            ngx.print(ostr)
        else
            ngx.say("error occured: rc=" .. res.status)
        end
    ';
}
```

Evolution


Static Content CDN
Browser
Javascript code
Busines s
AJAX
NginX Conf Scripting
Routing & Misc
Nginx
FastCGI
PHP code
Glider DAL
MySQL Client
php-fpm
MySQL Cluster
Static Content CDN
Browser
Javascript code
Busines s
AJAX
Lua code Misc
Busines
LZSQL DAL
Nginx
MySQL Cluster

Architecture (Current)



Pros

- Homogeneous server-side language, improving productivity
- Greatly increased large data processing speed

References

- https://github.com/chaoslawful/lua-nginx-module
- https://github.com/chaoslawful/drizzle-nginx-module
- https://github.com/agentzh/ngx_openresty
- http://luajit.org/luajit.html

Thanks!

http://bit.ly/lhuWkP