

www.Mcours.com

Site N°1 des Cours et Exercices Email: contact@mcours.com

Scripting libdrizzle with Lua inside Nginx

一淘
www.etao.com

量子恒道

Xiaozhe Wang
chaoslawful@gmail.com
[@chaoslawful](https://github.com/chaoslawful)

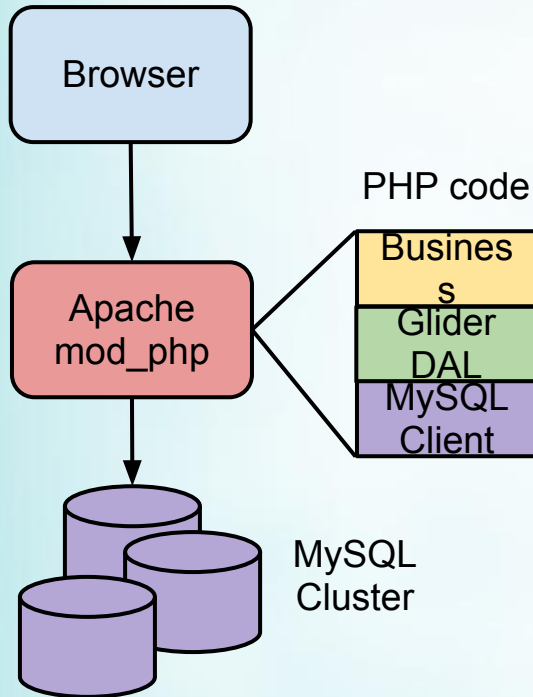
Business Background

- 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

- 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

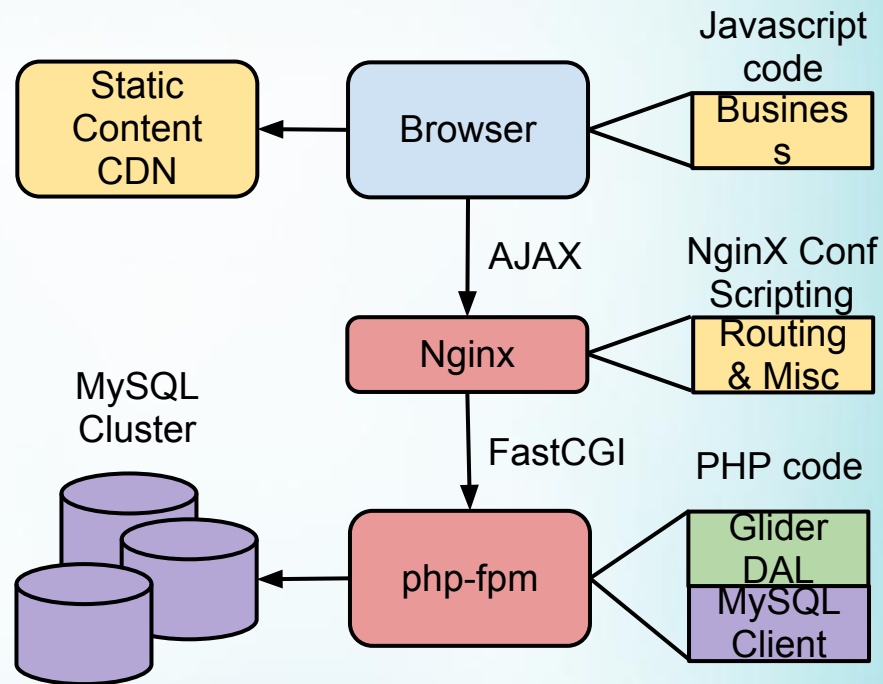
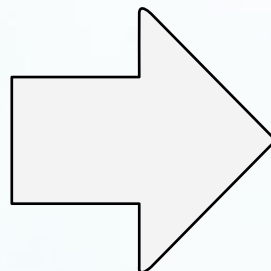
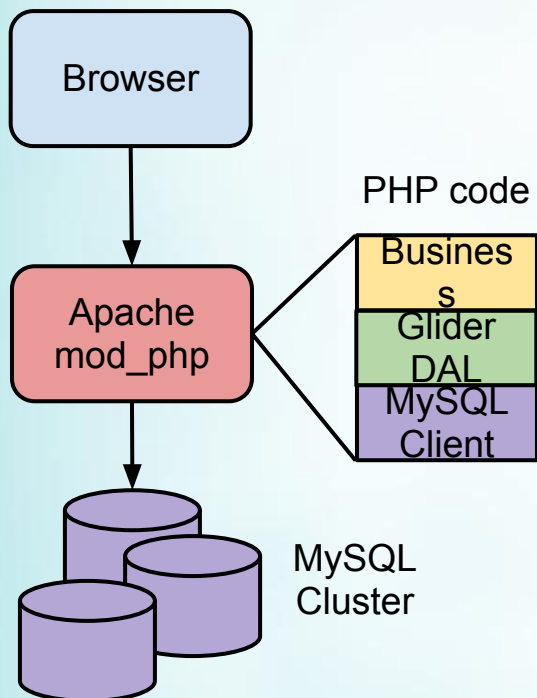
- Std LAMP, well understood
- Easy to learn for newcomers

Cons

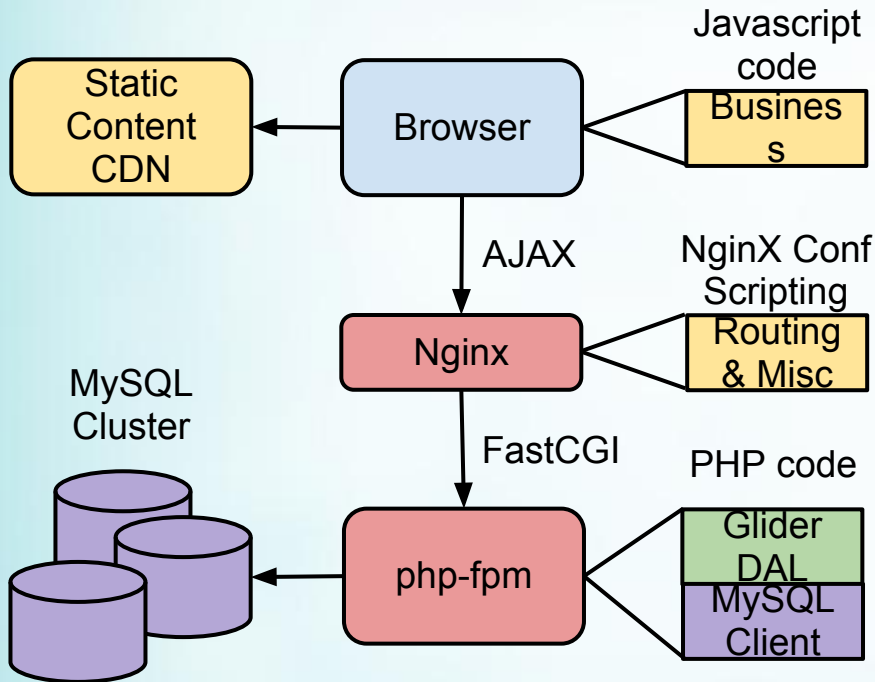
- Repeatedly transfer page outline
- Limited concurrency
- Slow-connection attack risk
- PHP is slow for large data processing

- 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



Architecture (improved)



Pros

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

Cons

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

Solution



量子恒道

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

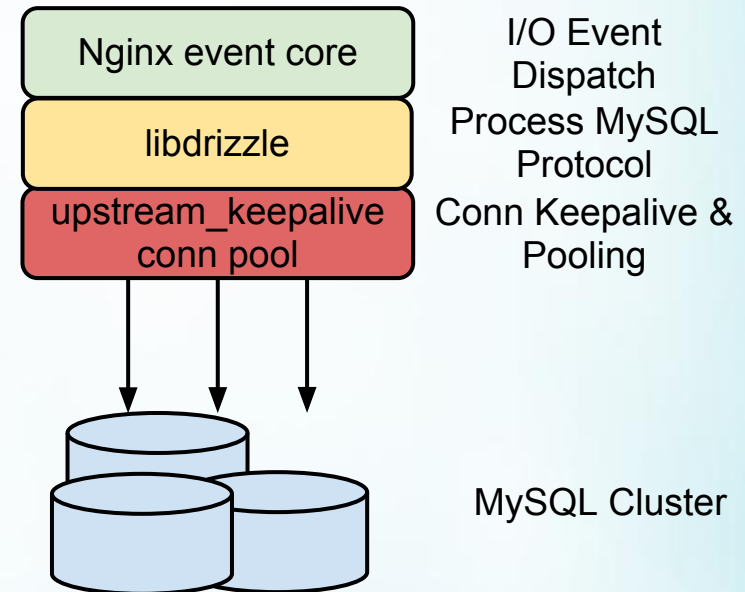
- Need efficient accessing to MySQL in Nginx
 - **ngx_drizzle was developed!**

- 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;  
        }  
    }  
}
```

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?

www.Mcours.com

Site N°1 des Cours et Exercices Email: contact@mcours.com

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

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

ngx_lua implemented the **proactor** pattern:

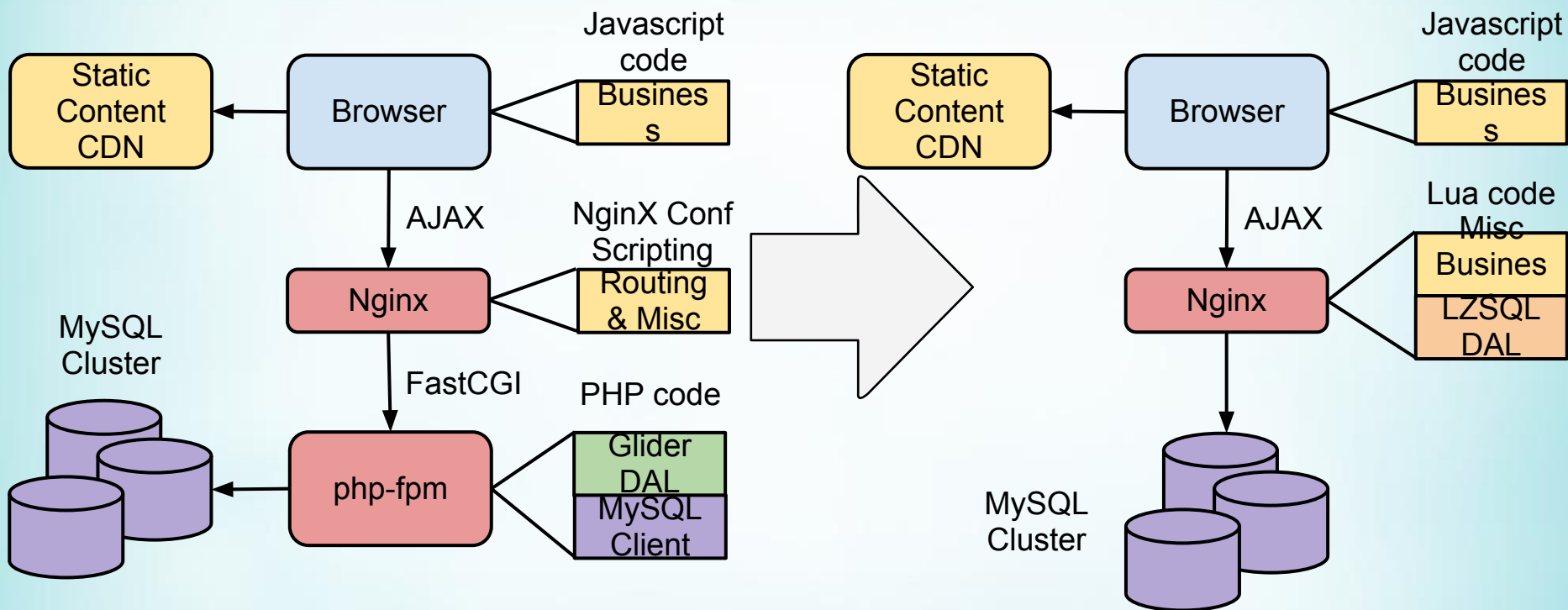
- Business logic is written in natural linear order
- Process resource would not be wasted on blocking I/O waiting
- Gain high concurrency capability automatically

ngx_lua example

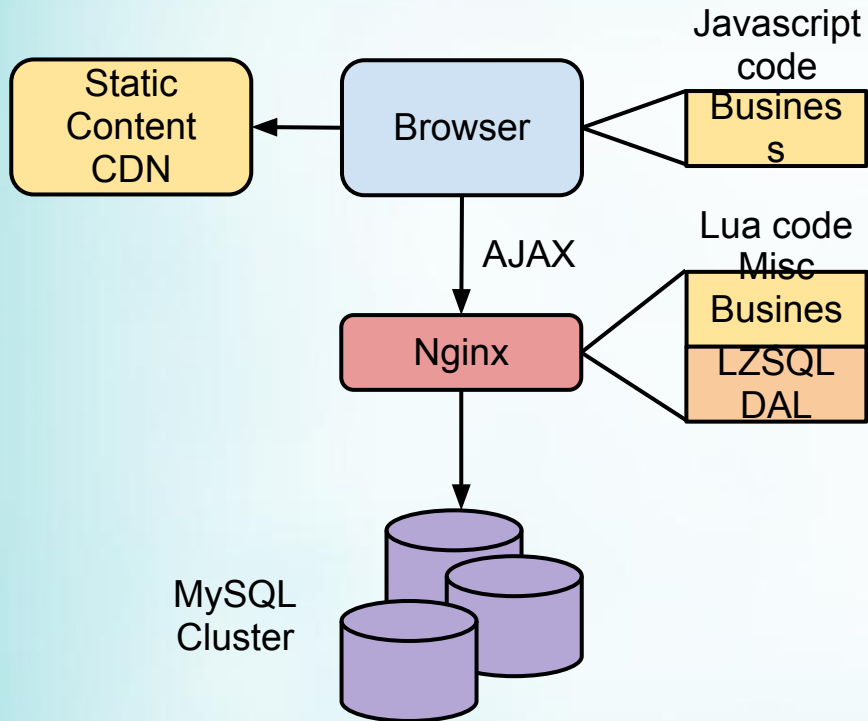


```
resolver ip.to.dns.server;
location /http_client {
    internal;
    proxy_pass $arg_url;
}
location /web_iconv {
    content_by_lua '
        local from, to, url = ngx.var.arg_f, ngx.var.arg_t, ngx.var.arg_u
        local capture = ngx.location.capture
        local iconv = require "iconv"
        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



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/nginx_openresty
- <http://luajit.org/luajit.html>

www.Mcours.com

Site N°1 des Cours et Exercices Email: contact@mcours.com

www.Mcours.com

Site N°1 des Cours et Exercices Email: contact@mcours.com



Thanks!

<http://bit.ly/lhuWkP>