APACHECON
NORTH AMERICA

Apache httpd v2.4:
What's New, Pussycat?

This should be pretty good!

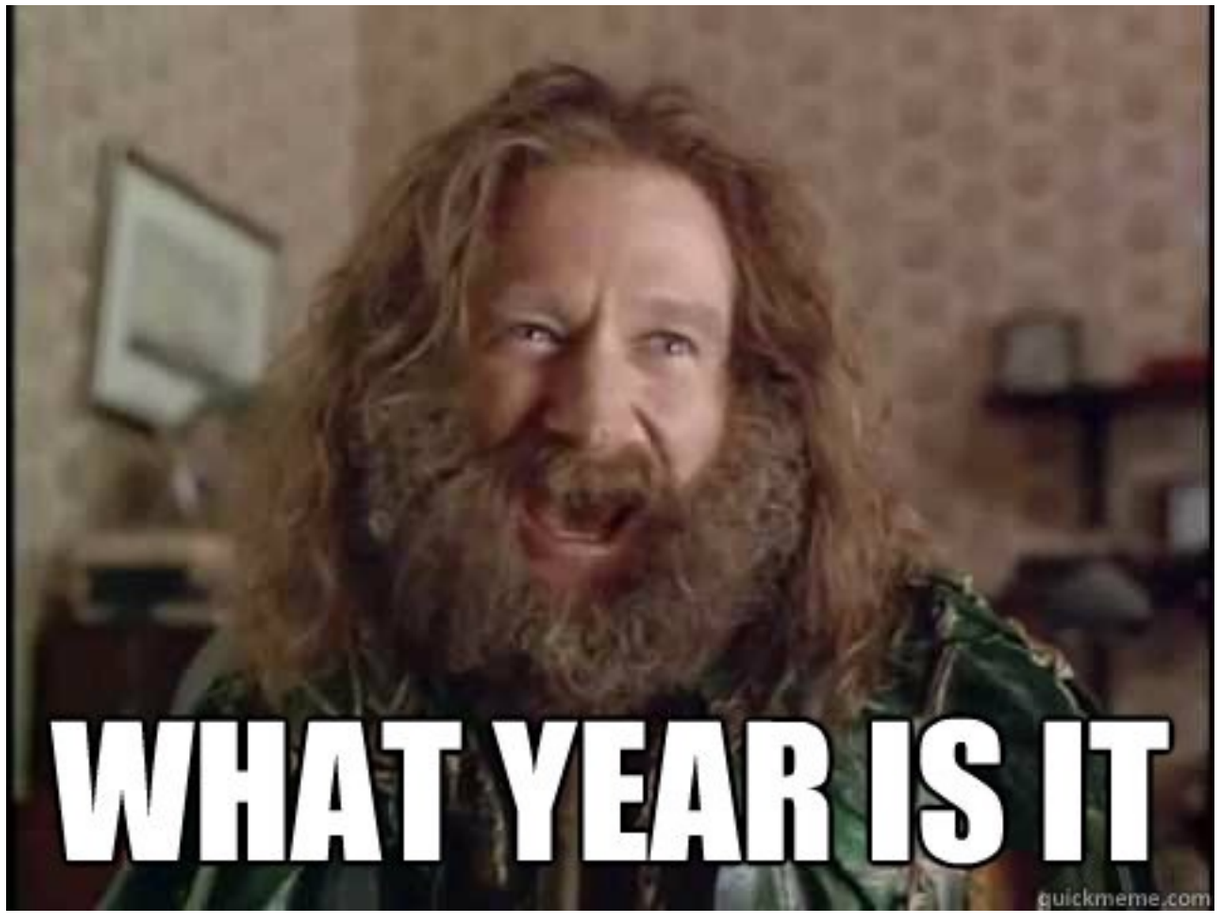
Jim Jagielski

About me

- Jim Jagielski
  - Hacker and developer
    - Wearer o' many hats at the ASF
    - Director and President: Outercurve
    - Council member: MARSEC-XL
  - Consulting Engineer with Red Hat
  - @jimjag

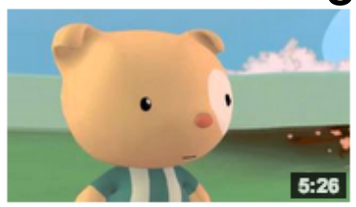
Hold on a tic

- How do you define “new”??



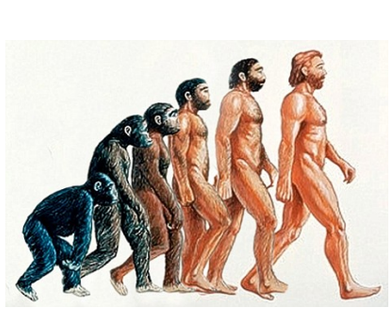
httpd is sooo old school *(aka fud)*

- Apache doesn't scale (its SLOW)
  - http://www.youtube.com/watch?v=bzkRVzciAZg



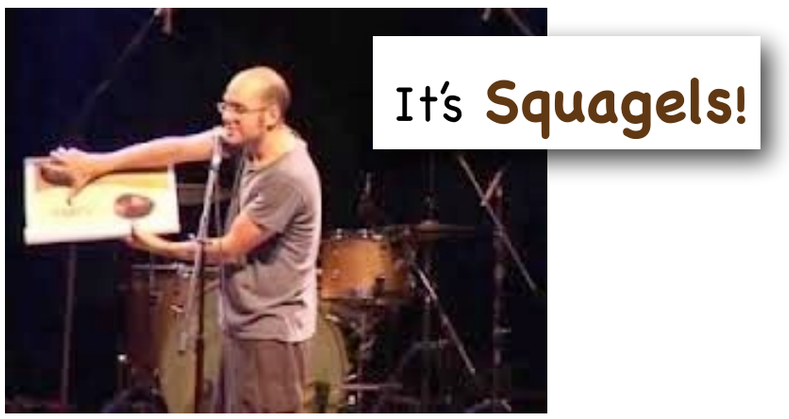
Node.js Is Bad Ass Rock Star Tech
by **gar1t** • 1 year ago • 52,419 views
A Q&A session on web servers turns existential.

- Apache is too generalized



VS

- Apache is too complex (config file)
  - really?
- Apache is too old
  (yeah, just like Linux)



Apache httpd 2.4 - design drivers

- New features and improve old ones
- Support for async I/O w/o dropping support for older systems
- Larger selection of usable MPMs: added Event, etc...
- Leverage higher-performant versions of APR
- Increase performance
- Reduce memory utilization
- The Cloud

Currently at version 2.4.12 (2.4.1 went GA Feb 21, 2012)

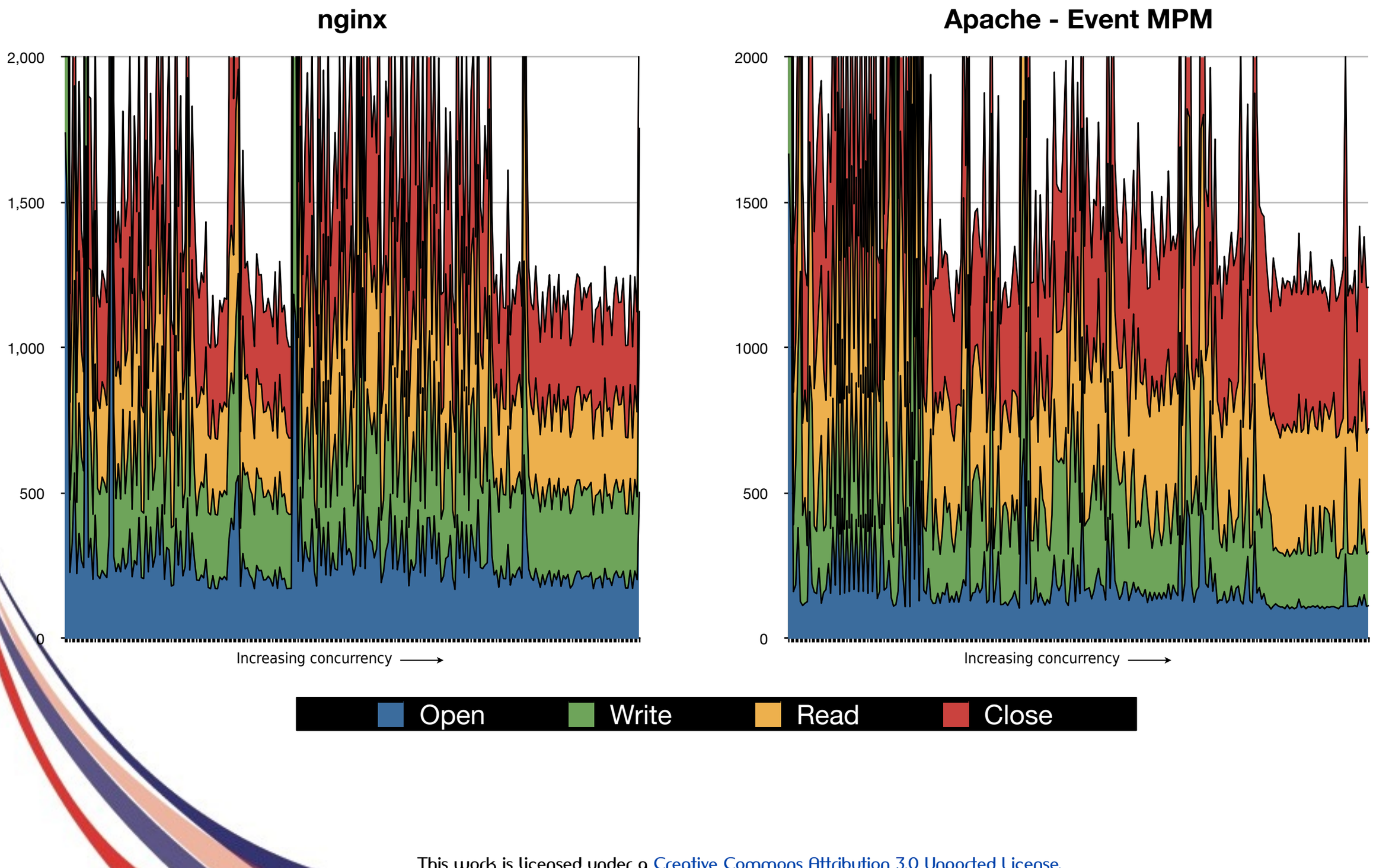
What's New: Apache httpd 2.4

- Performance Increases
- Configuration / Runtime Improvements
- New Modules / Capabilities
- Cloud / Proxy Enhancements

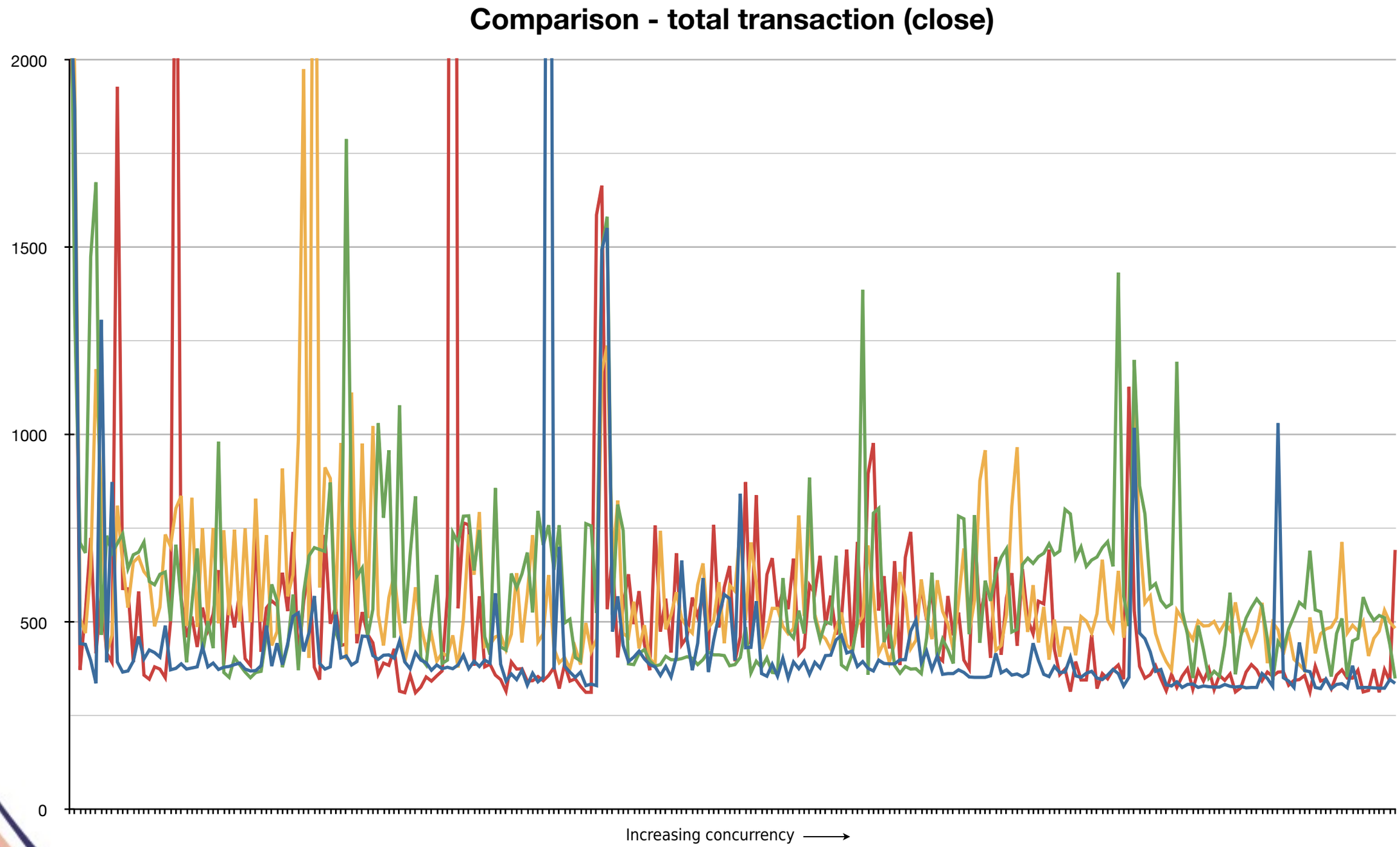
Performance

- Event MPM : no longer experimental
- Faster, more efficient APR
- Smaller memory footprint
- More efficient data structures (worker and event)

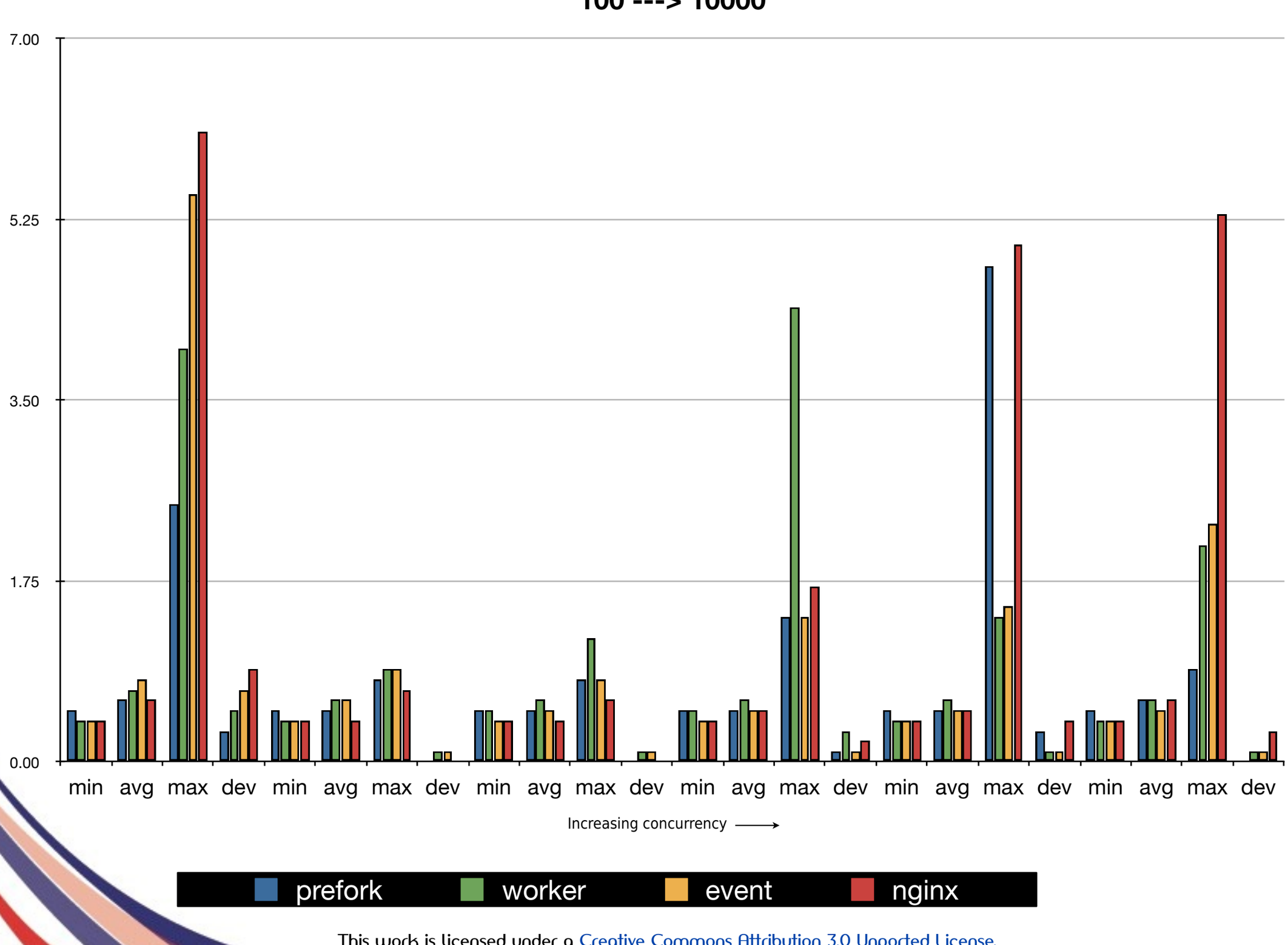
nginx vs Event (typical)



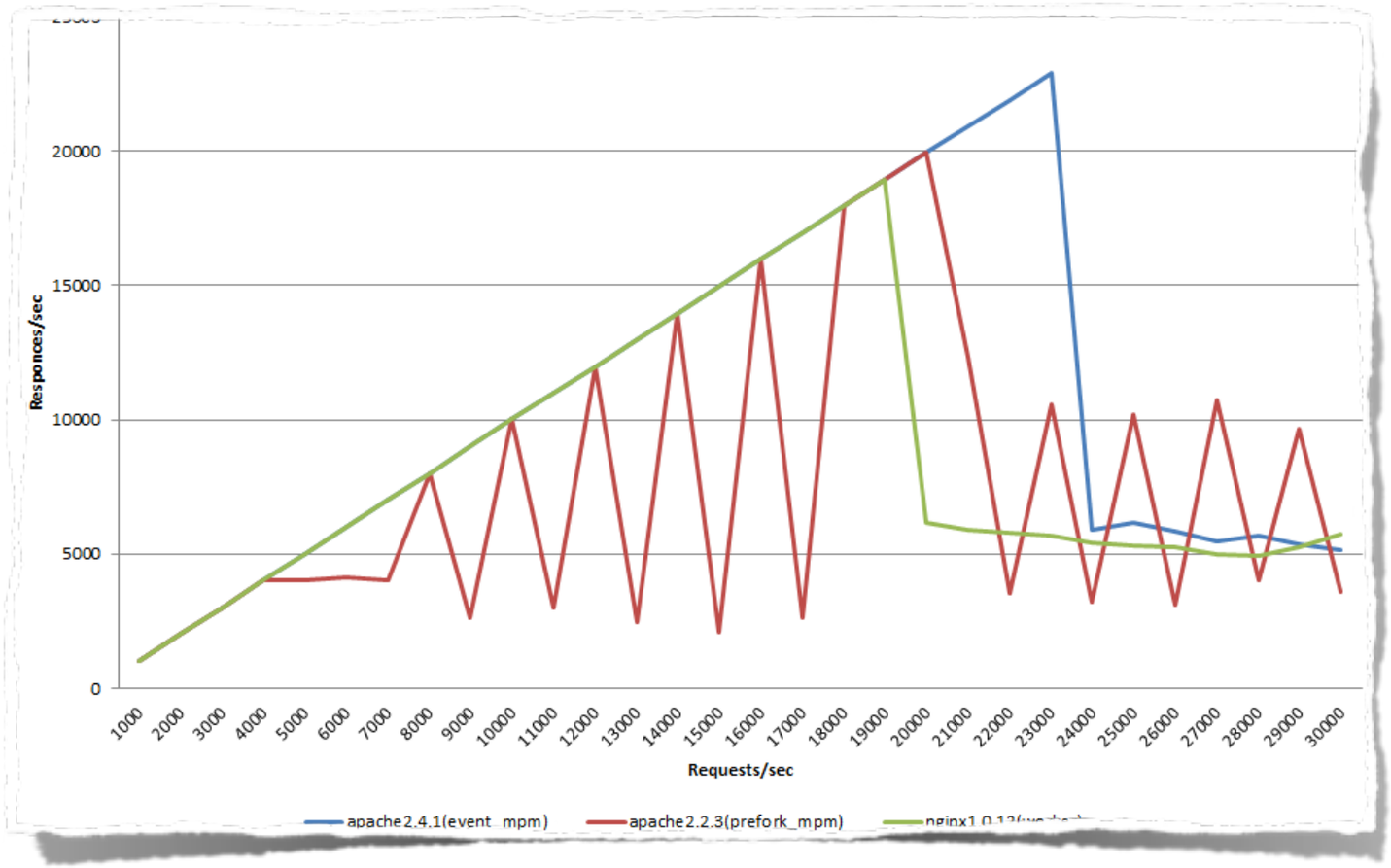
Total req/resp time



Resp to Req. Bursts - httpert



Independent benchmark




```
#!/bin/sh
RESULT='./result.txt'

for port in 80 8080 8888
do
    #for count in 1000 2000 3000 4000 5000 6000 7000 8000
9000 10000
    #for count in 11000 12000 13000 14000 15000 16000 17000
18000 19000 20000
    for count in 21000 22000 23000 24000 25000 26000 27000
28000 29000 30000
    do
        echo -n "$port $count " >> $RESULT
        httperf --rate $count --num-conns 25000 --server
ipaddr --port $port \
                --uri=/test.html | grep "Request rate:" >>
$RESULT.$port
        sleep 60
    done
done
```

Source: Ryosuke Matsumoto : http://blog.matsumoto-r.jp/?p=1812

Benchmark Conclusions

- Events, polling and fork/spawn creates overhead: good for "more bang for buck" system, bad for performance for ***that*** request
- For concurrency, Event & Worker on par with nginx*
- For transaction speed, prefork shines
- Let's work on leaner MPM (more streamlined)
- *Main Caveats:
  - Apache is never resource starved
  - If memory is a scarce resource, nginx still better *(for now ;))*
  - More work can (and should) be done

Configuration - Runtime

- Finer control of timeouts, esp. during requests
  - **mod_reqtimeout**

```
RequestReadTimout notice=10 body=30
```

  - **KeepAliveTimout** down to the millisecond
- Finer control over logging
  - per module/per directory
  - new logging levels (TRACE[1-8])

```
LogLevel notice
LogLevel info ssl:warn
<Directory "/usr/local/apache/htdocs/foo">
    LogLevel debug
</Directory>
```

Configuration - Runtime

- **<If>** supports per-request conditions
- General purpose expression parser (BNF compatible)

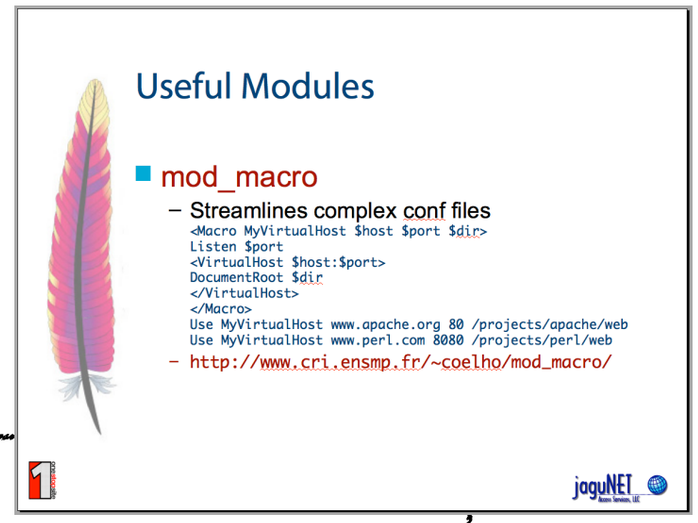
```
# Compare the host name to example.com and
# redirect to www.example.com if it matches
<If "%{HTTP_HOST} == 'example.com'">
    Redirect permanent / http://www.example.com/
<ElseIf "%{HTTP_HOST} == ‘foobarfoo.com'">
    Redirect permanent / http://www2.example.com/
</If>

<If “%{QUERY_STRING} =~ /dohtml/“>
    ForceType text/html
</If>
```

Configuration - Runtime

➡ mod_macro

From my ApacheCon 2000 Preso




```
<Macro VHost $name $domain>
<VirtualHost *:80>
    ServerName $domain
    ServerAlias www.$domain

    DocumentRoot /var/www/vhosts/$name
    ErrorLog /var/log/httpd/$name.error_log
    CustomLog /var/log/httpd/$name.access_log combined
</VirtualHost>
</Macro>

Use VHost example example.com
Use VHost myhost hostname.org
Use VHost apache apache.org

UndefMacro VHost
```

Configuration - Runtime

- Simple config-file variables: **<Define>**

```
<IfDefine TEST>
  Define servername test.example.com
</IfDefine>
<IfDefine !TEST>
  Define servername www.example.com
  Define SSL
</IfDefine>

DocumentRoot /var/www/${servername}/htdocs
```

Configuration - Runtime

- Other stuff:
  - No more **NameVirtualHost**
  - **AllowOverrideList**

```
AllowOverride None
AllowOverrideList Redirect RedirectMatch Header
```

- Loadable MPM modules
  - Recall that different MPMs have different config directives!

```
./configure –enable-mpms-shared=all
LoadModule mpm_event_module modules/mod_mpm_event.so
```

Configuration - Runtime

- **Require**
  - Removes order/deny insanity!


```
AuthType Basic
AuthName "Restricted Resource"
AuthBasicProvider file
AuthUserFile /web/users
AuthGroupFile /web/groups
Require group admin
<Directory /www/docs>
    <RequireAll>
        Require group alpha beta
        Require not group reject
    </RequireAll>
</Directory>
<Directory /www/docs2>
    Require all granted
</Directory>
```

New Modules

- **mod_lua** (semi-experimental, but we use it!)

```
<Files *.lua>
    SetHandler lua-script
</Files>
…
example.lua
require "string"
function handle(r)
    r.content_type = "text/plain"

    if r.method == 'GET' then
        r:puts("Hello Lua World!\n")
        for k, v in pairs( r:parseargs() ) do
            r:puts( string.format("%s: %s\n", k, v) )
        end
    elseif r.method == 'POST' then
        r:puts("Hello Lua World!\n")
        for k, v in pairs( r:parsebody() ) do
            r:puts( string.format("%s: %s\n", k, v) )
        end
    elseif r.method == 'PUT' then
        r:puts("Unsupported HTTP method " .. r.method)
        r.status = 405
        return apache2.ok
    else
       return 501
    end
    return apache2.OK
end
```

New Proxy (sub)Modules

- mod_proxy submodules:
  - **mod_proxy_fcgi**
  - **mod_proxy_scgi**
  - **mod_proxy_wstunnel**
  - **mod_proxy_html**
  - **mod_proxy_express**

```
ProxyExpressEnable on
ProxyExpressDBMFile emap
…
##
##express-map.txt: httxt2dbm -i express-map.txt -o emap
##

www1.example.com      http://192.168.002.2:8080
www2.example.com      http://192.168.002.12:8088
www3.example.com      http://192.168.002.10
 ...
www6341.example.com   http://192.168.211.26
```

New Modules

- **mod_buffer**
  - buffer the i/o stacks w/i httpd
- **mod_sed**
  - True sed functionality, alternate to mod_substitute

```
<Directory "/var/www/docs/status">
    AddOutputFilter Sed html
    OutputSed "s/complete/DONE/g"
    OutputSed “s/in-progress/TODO/g"
</Directory>
```

- **mod_remoteip**
  - allow access to the ***real*** client IP address

```
RemoteIPHeader X-Client-IP
```

New Modules

- **mod_session**
  - easily maintain application server state
- **mod_auth_form**
  - Form-based auth can now be handled internally

```
<Location /dologin.html>
    SetHandler form-login-handler
    AuthFormLoginRequiredLocation http://example.com/login.html
    AuthFormLoginSuccessLocation http://example.com/success.html
    AuthFormProvider file
    AuthUserFile conf/passwd
    AuthType form
    AuthName realm
    Session On
    SessionCookieName session path=/
    SessionCryptoPassphrase secret
</Location>
```

New Modules

- mod_log_debug
  - Add debug logging at any hook

```
<Location /foo>
  LogMessage “subreq to foo” hook=type_checker expr=%{IS_SUBREQ}
</Location>
```

- mod_ratelimit
  - (basic) bandwidth limiting for clients

```
<Location /downloads>
    SetOutputFilter RATE_LIMIT
    SetEnv rate-limit 400
</Location>
```

Even more!

- **mod_cache**
  - Can serve stale data if required
  - `X-Cache-Header` now supports `HIT/MISS/REVALIDATE`
  - Can cache `HEAD`
  - `htcacheclean` improvements
- **mod_socache / mod_slotmem**
  - Data object/blog storage mechanisms

Why Dynamic Proxy Matters

- Apache httpd still the most frequently used front-end
- Proxy capabilities must be cloud friendly
- Front-end must be dynamic friendly

Apache httpd 2.4 proxy

- Reverse Proxy Improvements
  - Supports FastCGI, SCGI, Websockets in balancer
  - Additional load balancing mechanisms
  - Runtime changing of clusters w/o restarts
  - Support for dynamic configuration
  - mod_proxy_express
  - mod_fcgid and fcgistarter
  - Brand New: Support for Unix Domain Sockets

Putting it all together

```
<Proxy balancer://foo>
  BalancerMember http://php1:8080/     loadfactor=1
  BalancerMember http://php2:8080/     loadfactor=4
  BalancerMember http://phpbkup:8080/  loadfactor=1 status=+h
  BalancerMember http://phpexp:8080/   lbset=1
  ProxySet lbmethod=bytraffic
</Proxy>
<Proxy balancer://javaapps>
  BalancerMember ajp://tc1:8089/     loadfactor=1
  BalancerMember ajp://tc2:8089/     loadfactor=4
  ProxySet lbmethod=byrequests
</Proxy>
ProxyPass        /apps/         balancer://foo/
ProxyPassReverse /apps/         balancer://foo/
ProxyPass        /serv/         balancer://javaapps/
ProxyPass        /images/       http://images:8080/
ProxyPass        /foo           unix:/home/www.socket|http://localhost/bar/
```

HeartBeat / HeartMonitor

- New LB (load balance) method
  - Uses multicast between gateway and reverse proxies
  - Provides heartbeat (are you there?) capability
  - Also provides basic load info
    - This info stored in shm, and used for balancing
- Multicast can be an issue
- Use mod_header with %l, %i, %b (loadavg, idle, busy)
  - but no LBmethod currently uses this :(
- We need a universal “load” measure

balancer-manager

- Embedded proxy admin web interface
- Allows for real-time
  - Monitoring of stats for each worker
  - Adjustment of worker params
- Allows for real-time
  - Addition of ***new*** workers/nodes
  - Change of LB methods
  - Can be ***persistent!***
  - More RESTful
  - Can be CLI-driven

What's next?

- Support for HTTP/2 (*mod_h2*)
- Support for ALPN (TLS)
- Better async support
- More MPMs
  - *motorz*:
    - Streamlined event driven MPM
    - Prelim benchmarks: 50% faster, 33% the size
- You tell us!

Thanks

Twitter: @jimjag

Emails:

jim@jaguNET.com
jjagielski@outercurve.org
jim@apache.org
jimjag@redhat.com

http://www.slideshare.net/jimjag/