



**Apache**  
HTTP SERVER PROJECT

Apache *httpd* v2.4  
Reverse Proxy  
The “Hidden” Gem

Jim Jagielski

@jimjag

# About Me

- ➔ **Apache Software Foundation**
  - ➔ **Co-founder, Director, Member and Developer**
- ➔ **Director**
  - ➔ **Outercurve, MARSEC-XL, OSSI, OSI (ex)...**
- ➔ **Developer**
  - ➔ **Mega FOSS projects**
- ➔ **O'Reilly Open Source Award: 2013**
- ➔ **European Commission: Luminary Award**
- ➔ **Sr. Director: Tech Fellows: Capital One**



# *Apache httpd 2.4*

- **Currently at version 2.4.23 (2.4.1 went GA Feb 21, 2012)**
- **Significant Improvements**
  - **high-performance**
  - **cloud suitability**

# *Apache httpd 2.4 - design drivers*

- **Support for async I/O w/o dropping support for older systems**
- **Larger selection of usable MPMs: added *event*, *motorz*, etc...**
- **Leverage higher-performant versions of APR**
- **Increase performance**
- **Reduce memory utilization**
- **The Cloud and Reverse Proxy**

# *httpd is sooo old school* (aka fud)

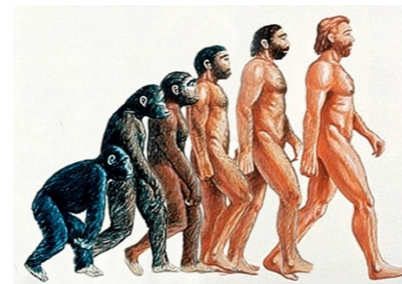
→ Apache doesn't scale (its SLOW)

→ <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)



It's **Squagels!**

# Cloud and Dynamics

- **The Cloud is a game changer for web servers**
  - **The cloud is a dynamic place**
  - **automated reconfiguration**
  - **horizontal, not vertical scaling**
  - **self-aware environments**



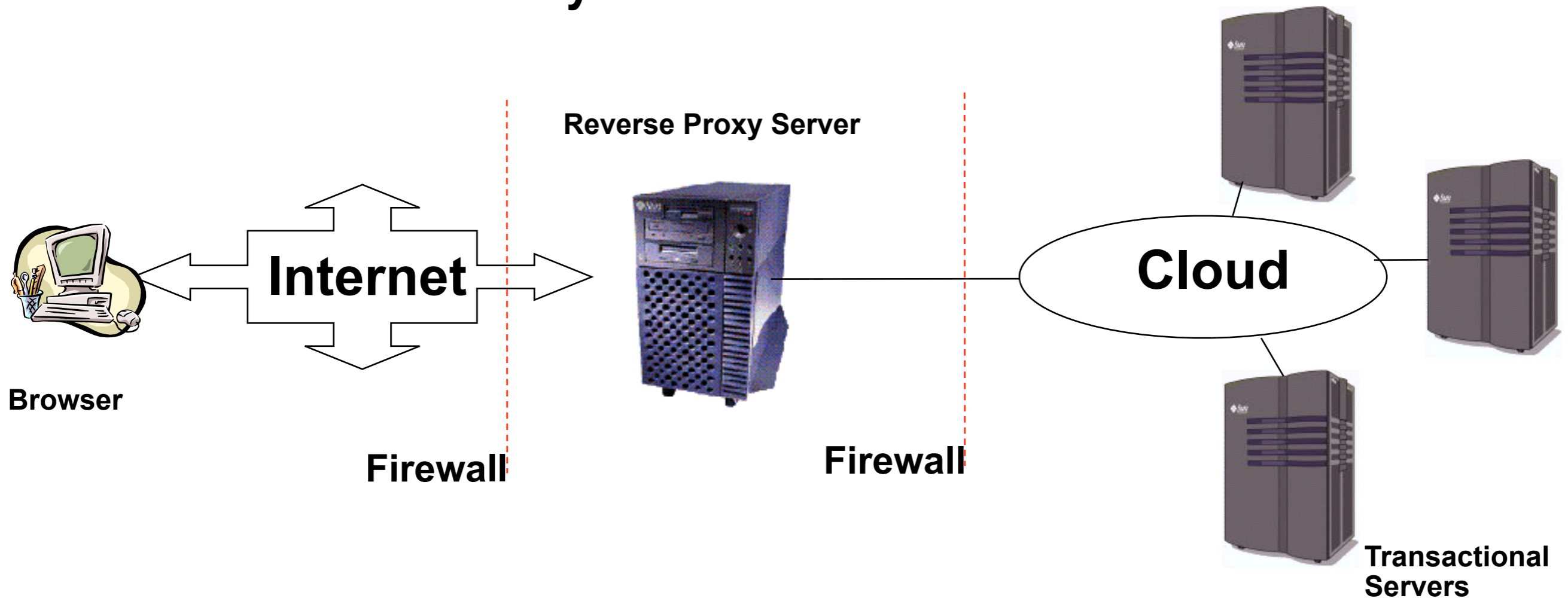
OK, maybe not THAT self-aware

# *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**

# Reverse Proxy

- Operates at the server end of the transaction
- Completely transparent to the Web Browser – thinks the Reverse Proxy Server is the real server





# *Features of Reverse Proxy Server*

## → Security

Uniform security policy can be administered

The real transactional servers are behind the firewall

## → Delegation, Specialization, Load Balancing

## → Caching

## → Performance, HA

# *Proxy Design Drivers*

- **Becoming a robust but generic proxy implementation**
- **Support various protocols**
  - HTTP, HTTPS, HTTP/2, CONNECT, FTP
  - AJP, FastCGI, SCGI, WSGI
  - Load balancing
- **Clustering, failover**
- **Performance**

# *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**
  - **Brand New: HTTP/2**

# *Configuring Reverse Proxy*

- **Set ProxyRequests Off**
- **Apply ProxyPass, ProxyPassReverse and possibly RewriteRule directives**

# Reverse Proxy Directives: ProxyPass

- Allows remote server to be mapped into the space of the local (Reverse Proxy) server
- There is also *ProxyPassMatch* which takes a regex
- Example:
  - `ProxyPass /secure/ http://securerouter/`
  - Presumably “securerouter” is inaccessible directly from the internet
  - `ProxyPassMatch ^/(.*\.js)$ http://js-storage.example.com/bar/$1`

# Reverse Proxy Directives: *ProxyPassReverse*

- Used to specify that redirects issued by the remote server are to be translated to use the proxy before being returned to the client.
- Syntax is identical to *ProxyPass*; used in conjunction with it
- Example:
  - `ProxyPass /secure/ http://secureserver/`
  - `ProxyPassReverse /secure/ http://secureserver/`

# Simple Rev Proxy

- All requests for /images to a backend server

```
ProxyPass /images http://images.example.com/
```

```
ProxyPass <path> <scheme>://<full url>
```

- Useful, but limited

- What if:

images.example.com dies?

traffic for /images increases

# *Load Balancing*

- **mod\_proxy\_balancer.so**
- **mod\_proxy can do native load balancing**
  - **weight by actual requests**
  - **weight by traffic**
  - **weight by busyness**
  - **lb factors**



# Create a balancer “cluster”

- Create a balancer which contains several host nodes
- Apache *httpd* will then direct to each node as specified

```
<Proxy balancer://foo>  
  BalancerMember http://www1.example.com:80/ loadfactor=1  
  BalancerMember http://www2.example.com:80/ loadfactor=1  
  BalancerMember http://www3.example.com:80/ loadfactor=4 status=+h  
  ProxySet lbmethod=bytraffic  
</Proxy>
```

# *Some config params*

## → For BalancerMembers:

### → **loadfactor**

→ normalized load for worker [1]

### → **lbset**

→ worker cluster number [0]

### → **retry**

→ retry timeout, in seconds, for non-ready workers [60]

# Some config params

- **For BalancerMembers (cont):**
  - **connectiontimeout/timeout**
    - Connection timeouts on backend [ProxyTimeout]
  - **flushpackets \***
    - Does proxy need to flush data with each chunk of data?
      - on : Yes | off : No | auto : wait and see
  - **flushwait \***
    - ms to wait for data before flushing

# *Some config params*

## → For BalancerMembers (cont):

### → ping

- Ping backend to check for availability; value is time to wait for response

### → status (+/-)

- D : Disabled
- S : Stopped
- I : Ignore errors
- H : Hot standby
- E : Error
- N: Drain
- C: Dynamic Health Check

# *Some config params*

- **For Balancers:**
  - **lbmethod**
    - load balancing algo to use [byrequests]
  - **stickysession**
    - sticky session name (eg: PHPSESSIONID)
  - **maxattempts**
    - # failover tries before we bail
  - **growth**
    - Extra BalancerMember slots to allow for

# Some config params

- **For Balancers:**
  - **nofailover**
    - pretty freakin obvious
- **For both:**
  - **ProxySet**
    - Alternate method to set various params

```
ProxySet balancer://foo timeout=10
...
ProxyPass / balancer://foo timeout=10
```

# Connection Pooling

- **Backend connection pooling**
- **Available for named workers:**
  - **eg: ProxyPass /foo http://bar.example.com**
- **Reusable connection to origin**
  - **For threaded MPMs, can adjust size of pool (min, max, smax)**
  - **For prefork: singleton**
- **Shared data held in shared memory**

# *Some config params*

- **For BalancerMembers - connection pool:**
  - **min**
    - Initial number of connections [0]
  - **max**
    - Hard maximum number of connections [1|TPC]
  - **smax:**
    - soft max - keep this number available [max]



# *Some config params*

- **For BalancerMembers - connection pool:**
  - **disablereuser/enablereuse:**
    - bypass/enable the connection pool (firewalls)
  - **t1**
    - time to live for connections above **smax**

# Sessions

- **Sticky session support**
  - aka “session affinity”
- **Cookie based**
  - `stickySession=PHPSESSID`
  - `stickySession=JSESSIONID`
- **Natively easy with Tomcat**
- **May require more setup for “simple” HTTP proxying**
- **Use of `mod_session` helps**

# *Failover control*

- **Cluster set with failover**
- **Group backend servers as numbered sets**
  - balancer will try lower-valued sets first
  - If no workers are available, will try next set
- **Hot standby**

# 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=10
  BalancerMember ajp://tc2:8089/        loadfactor=40
  ProxySet lbmethod=byrequests
</Proxy>
ProxyPass          /apps/                balancer://foo/
ProxyPassReverse   /apps/                balancer://foo/
ProxyPass          /serv/                balancer://javaapps/
ProxyPass          /images/              http://images:8080/

ProxyPass          /dyno                 h2c://pappy:80/

ProxyPass          /foo                  unix://home/www.socket | http://localhost/bar/
```

# Mass Reverse Proxy

- We front-end a LOT of reverse proxies
  - What a httpd.conf disaster!
  - Slow and bloated
  - mod\_rewrite doesn't help

```
<VirtualHost www1.example.com>
  ProxyPass / http://192.168.002.2:8080
  ProxyPassReverse / http://192.168.002.2:8080
</VirtualHost>

<VirtualHost www2.example.com>
  ProxyPass / http://192.168.002.12:8088
  ProxyPassReverse / http://192.168.002.12:8088
</VirtualHost>

<VirtualHost www3.example.com>
  ProxyPass / http://192.168.002.10
  ProxyPassReverse / http://192.168.002.10
</VirtualHost>

...
<VirtualHost www6341.example.com>
  ProxyPass / http://192.168.211.26
  ProxyPassReverse / http://192.168.211.26
</VirtualHost>
```

# Mass Reverse Proxy

- Use the new `mod_proxy_express` module
  - ProxyPass mapping obtained via db file
  - Fast and efficient
  - Still dynamic, with no config changes required
  - micro-services? You betcha!

## ProxyExpress map file

```
##  
##express-map.db:  
##  
  
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
```

## httpd.conf file

```
ProxyExpressEnable On  
ProxyExpressDBMFile express-map.db
```

# *HeartBeat / HeartMonitor*

- **Experimental 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**
- **Can we leverage nanomsg (MIT licensed!)**

# *balancer-manager*

- **Embedded proxy admin web interface**
- **Allows for real-time**
  - **Monitoring of stats for each worker**
  - **Adjustment of worker params**
    - **lbset**
    - **load factor**
    - **route**
    - **enabled / disabled**
    - **...**



# *Embedded Admin*

- **Allows for real-time**
  - Addition of *new* workers/nodes
  - Change of LB methods
  - Can be *persistent!*
  - More RESTful
  - Can be CLI-driven


# Easy setup

```
<Location /balancer-manager>  
    SetHandler balancer-manager  
    Require 192.168.2.22  
</Location>
```

Welcome to The Apache Software Foundation

www.apache.org

Foundation Projects People Get Involved Download Support



**The Apache Software Foundation**

Home

*Community-led development since 1999.*

**We consider ourselves**  
not simply a group of projects sharing a server, but rather a community of developers and users.

**The Apache Software Foundation**  
provides support for the Apache community of open-source software projects, which provide software products for the public good.

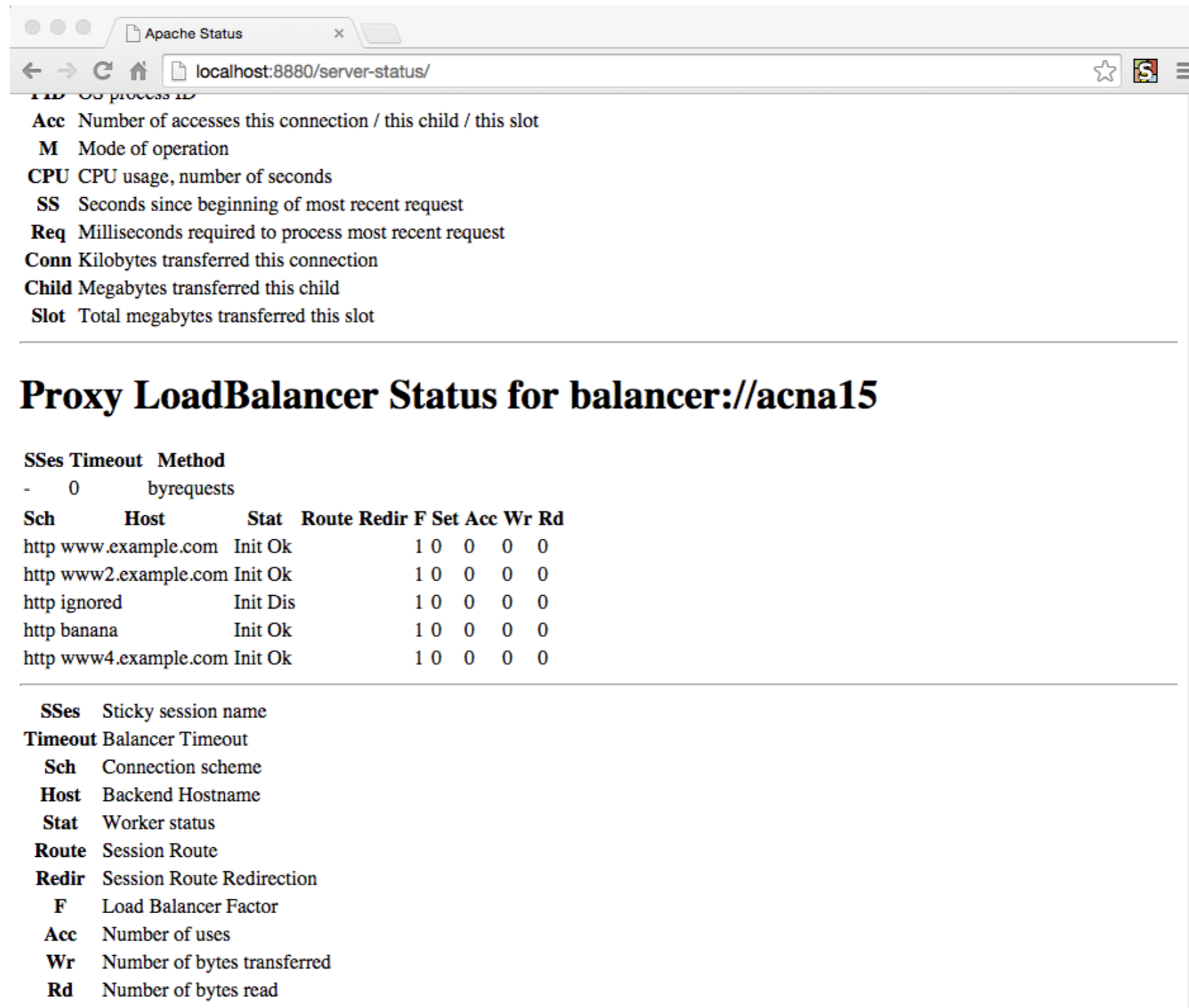
**The Apache projects are defined**  
by collaborative consensus based processes, an open, pragmatic software license and a desire to create high quality software that leads the way in its field.

Featured Projects » Apache Directory Apache OFBiz Apache Drill

**The ASF is made up of more than 150 top level projects** which cover a wide range of technologies. Chances are if you are looking for a rewarding experience in Open Source, you are going to find it here.

**Apache Directory**  
LDAP and Kerberos, entirely in Java

# server-status aware



The screenshot shows a web browser window with the title "Apache Status" and the URL "localhost:8880/server-status/". The page content includes a legend for various status indicators, a section for "Proxy LoadBalancer Status for balancer://acna15", and a table of backend server status.

**Legend:**

- Acc** Number of accesses this connection / this child / this slot
- M** Mode of operation
- CPU** CPU usage, number of seconds
- SS** Seconds since beginning of most recent request
- Req** Milliseconds required to process most recent request
- Conn** Kilobytes transferred this connection
- Child** Megabytes transferred this child
- Slot** Total megabytes transferred this slot

---

## Proxy LoadBalancer Status for balancer://acna15

**SSes Timeout Method**  
- 0 byrequests

Sch	Host	Stat	Route	Redir	F	Set	Acc	Wr	Rd
http	www.example.com	Init Ok			1	0	0	0	0
http	www2.example.com	Init Ok			1	0	0	0	0
http	ignored	Init Dis			1	0	0	0	0
http	banana	Init Ok			1	0	0	0	0
http	www4.example.com	Init Ok			1	0	0	0	0

---

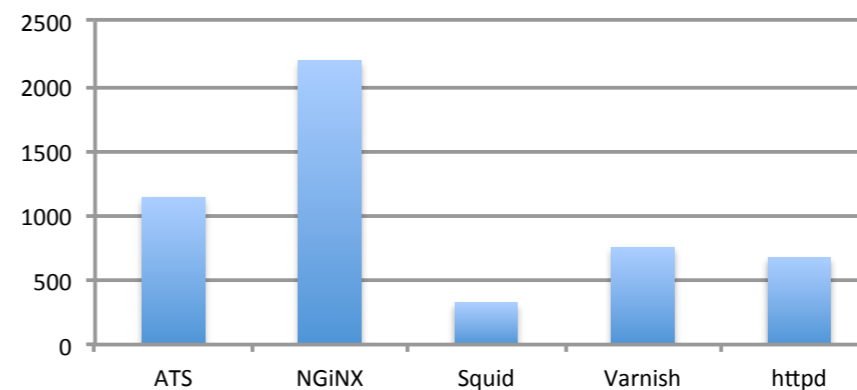
**SSes** Sticky session name  
**Timeout** Balancer Timeout  
**Sch** Connection scheme  
**Host** Backend Hostname  
**Stat** Worker status  
**Route** Session Route  
**Redir** Session Route Redirection  
**F** Load Balancer Factor  
**Acc** Number of uses  
**Wr** Number of bytes transferred  
**Rd** Number of bytes read

# Performance

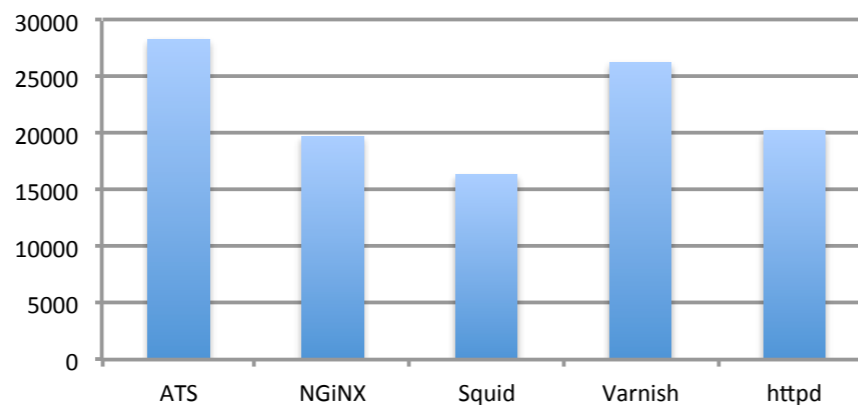
→ From Bryan Call's 2014 ApacheCon preso  
([http://www.slideshare.net/bryan\\_call/choosing-a-proxy-server-apachecon-2014](http://www.slideshare.net/bryan_call/choosing-a-proxy-server-apachecon-2014))

- Squid used the most CPU again
- NGiNX had latency issues
- ATS most throughput

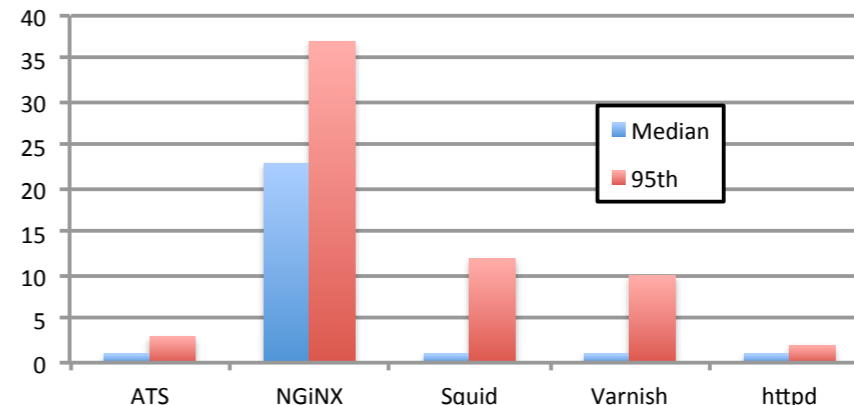
RPS / CPU Usage



Requests Per Second

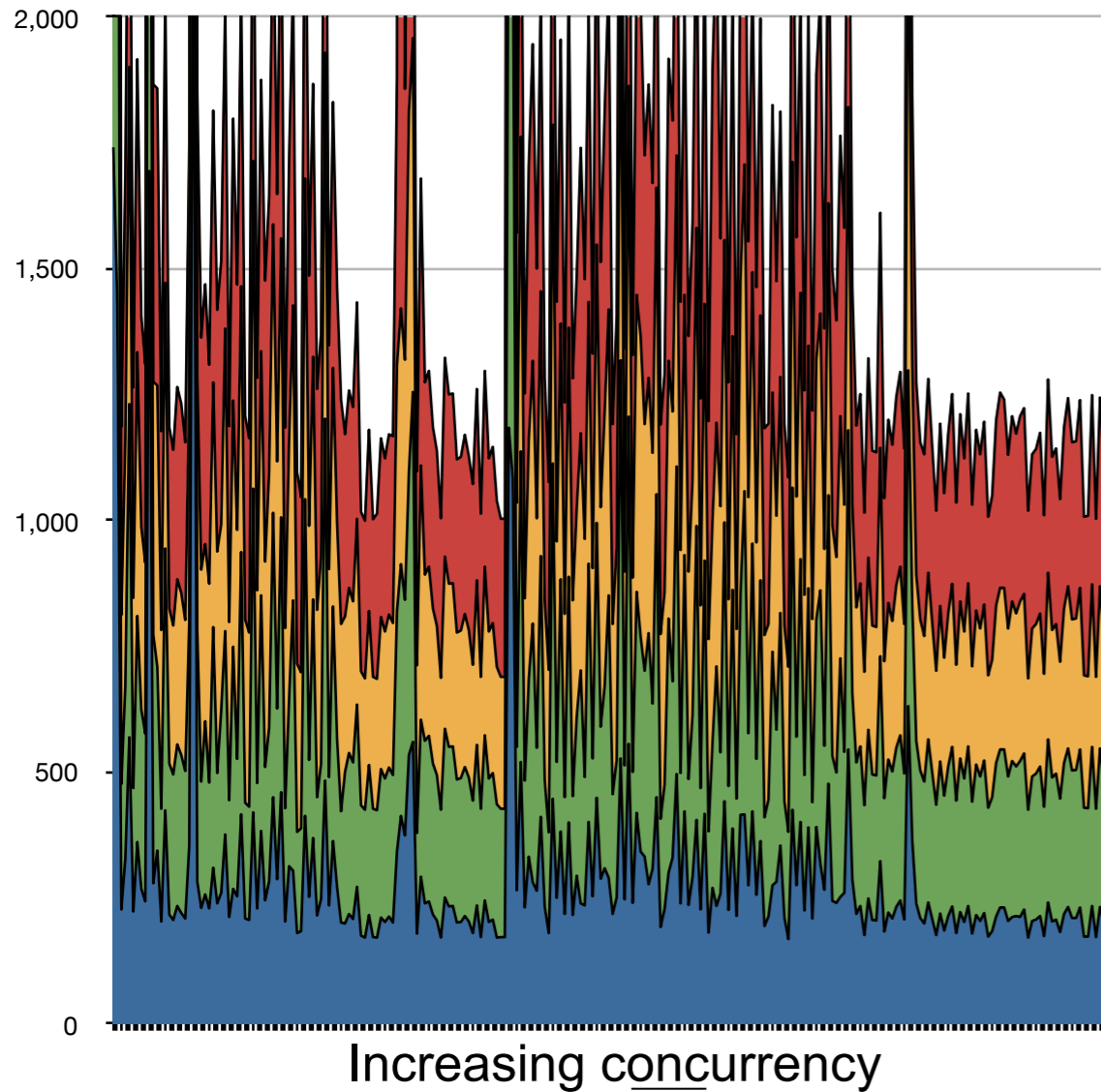


Latency

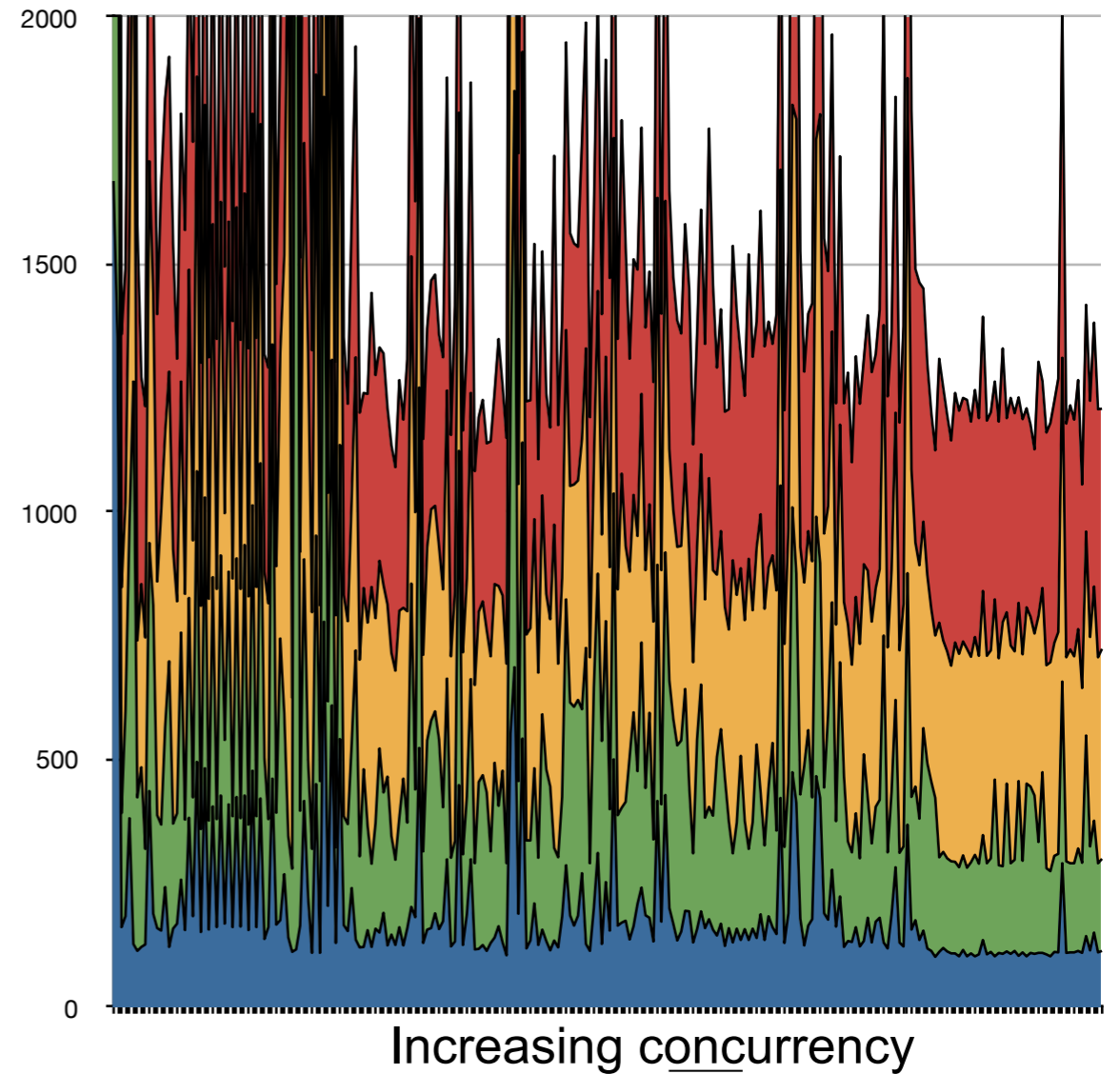


# nginx vs Event (typical)

nginx



Apache - Event MPM

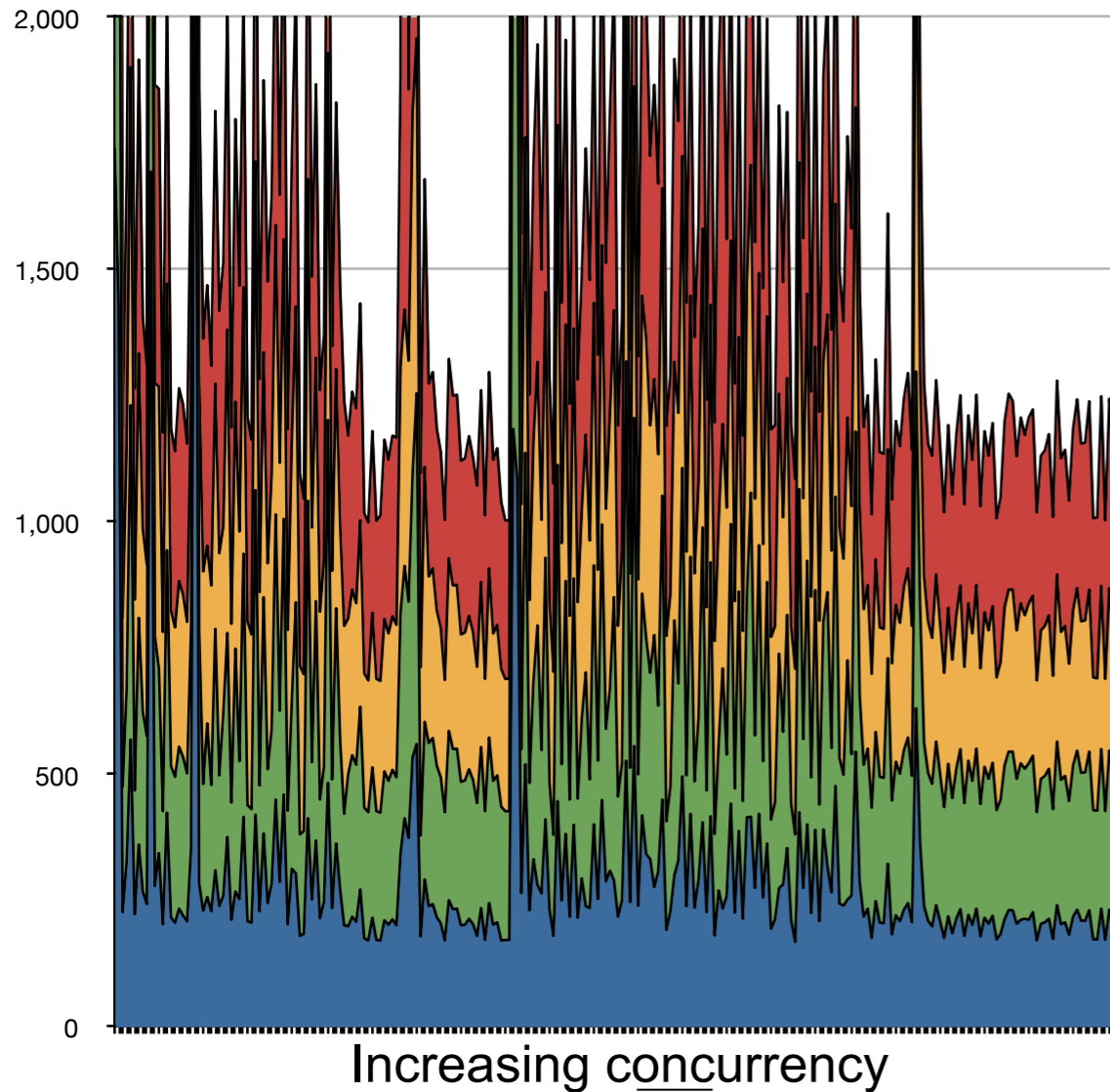


■ Open ■ Write ■ Read ■ Close

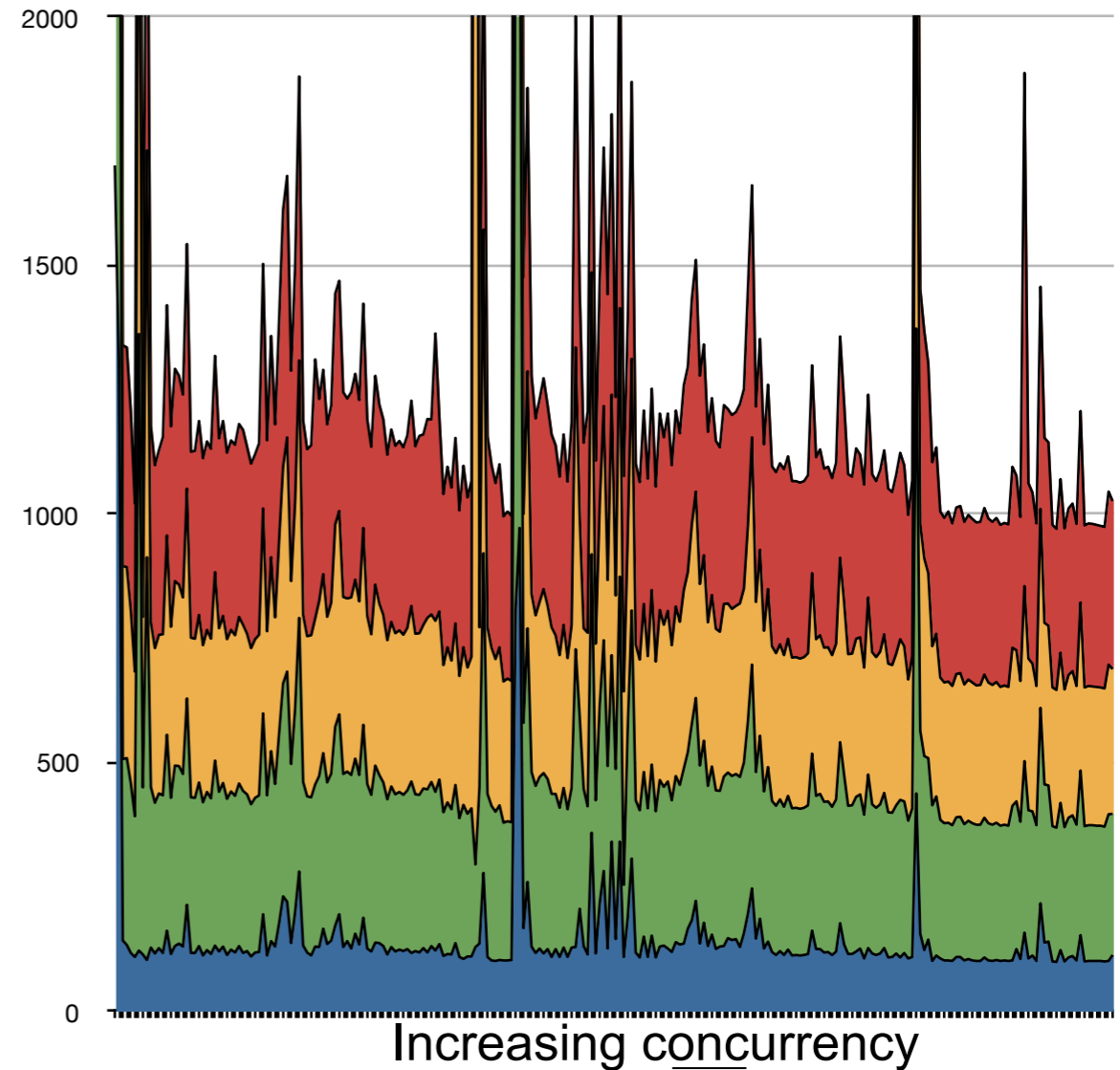


# nginx vs Prefork (typical)

nginx



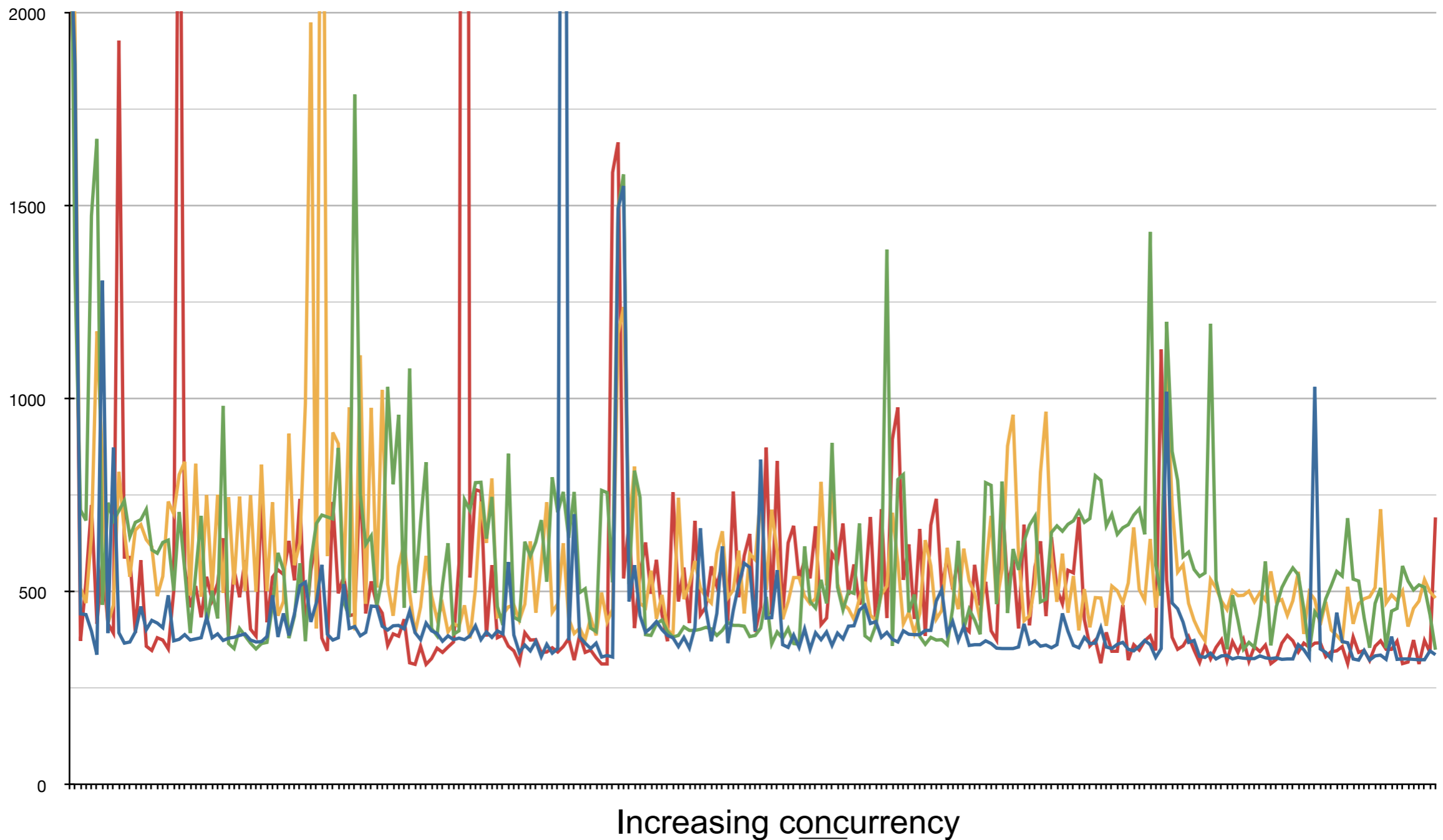
Apache - Prefork MPM



Open Write Read Close

# Total req/resp time

Comparison - total transaction (close)

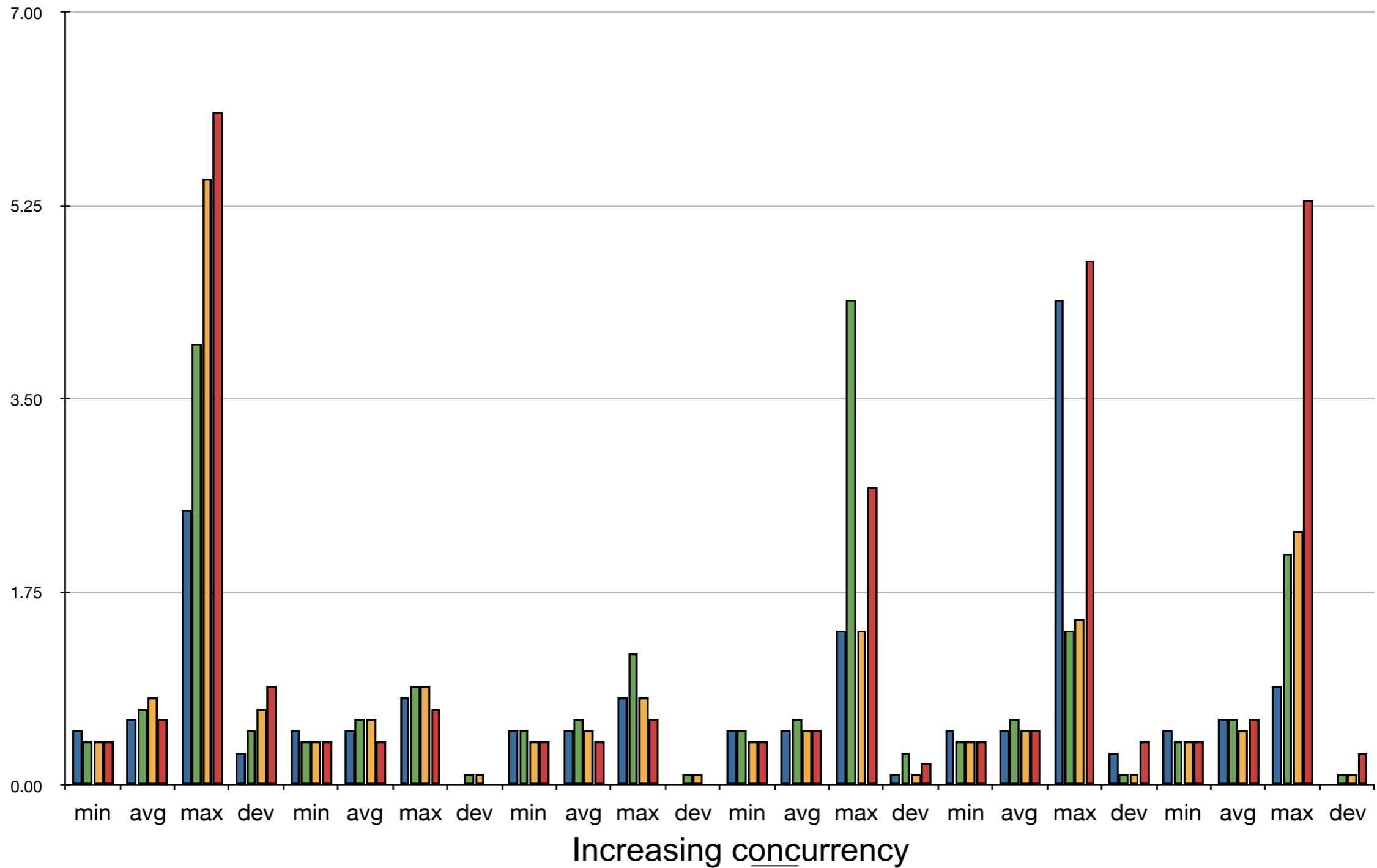


— Prefork — Worker — Event — nginx



# Resp to Req. Bursts - httperf

100 ----> 20000



■ prefork   ■ worker   ■ event   ■ nginx

# Backend Status

- **Dynamic Health Checks !**
  - **TCP/IP Ping**
  - **OPTIONS**
  - **HEAD**
  - **GET**

```
ProxyHCEExpr ok234 {%{REQUEST_STATUS} =~ /^[234]/}  
ProxyHCEExpr gdown {%{REQUEST_STATUS} =~ /^[5]/}  
ProxyHCEExpr in_maint {hc('body') !~ /Under maintenance/}
```

```
<Proxy balancer://foo/>  
  BalancerMember http://www.example.com/ hcmethod=GET hcexpr=in_maint hcuri=/status.php  
  BalancerMember http://www2.example.com/ hcmethod=HEAD hcexpr=ok234 hcinterval=10  
  BalancerMember http://www3.example.com/ hcmethod=TCP hcinterval=5 hcpasses=2 hcfails=3  
  BalancerMember http://www4.example.com/  
</Proxy>
```

```
ProxyPass "/" "balancer://foo/"  
ProxyPassReverse "/" "balancer://foo/"
```

# *What's on the horizon?*

- **Extend mod\_proxy\_express**
- **Adding additional protocols**
- **More dynamic configuration**
  - **Adding balancers!**
- **Extend/improve caching**
  - *Redis*
  - *Memcache* now mod\_status aware
  - **Apache Geode?**
- **Performance, of course!**

# *In conclusion...*

- **For cloud environs and other, the performance and dynamic control of Apache httpd 2.4 in reverse proxies is just what the Dr. ordered (and flexibility remains a big strength)**

# Thanks

Twitter: @jimjag

Emails:

jim@jaguNET.com

jim@apache.org

jim.jagielski@capitalone.com

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