O'REILLY®

oscon

open source convention

JULY 16–20, 2012 PORTLAND, OR

#oscon

Apache httpd v2.4:

Watch Out Cloud

-or-

Hello Cloud: Buy you a drink?

Jim Jagielski

About me

- Jim Jagielski
  - Hacker and developer
    - Co-founder of the ASF
    - Member, Director and President
    - Director: Outercurve and OSI
    - Council member: MARSEC-XL
  - Consulting Engineer with Red Hat
  - @jimjag

What we will cover

- Overview of Apache httpd 2.4
  - General improvements
  - Reverse proxy improvements
- How the Cloud is a game-changer for web
- Performance Related Enhancements

Apache httpd 2.4

- Currently at version 2.4.2 (2.4.1 went GA Feb 21, 2012)
- 2.4.3 RSN
- 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, Simple, etc...
- Leverage higher-performant versions of APR
- Increase performance
- Reduce memory utilization
- The Cloud

What's New: Apache httpd 2.4

- Bandwidth control now standard
  - mod_ratelimit
- Finer control of timeouts, esp. during requests
  - mod_reqtimeout
  - KeepAliveTimout down to the millisecond
- Finer control over logging
  - per module/per directory
  - new logging levels (TRACE[1-8])
- <If> supports per-request conditions
- slot-based shared memory capability

What's New: Apache httpd 2.4

- Controllable buffering of I/O
  - mod_buffer
- Support for Lua *(still experimental as of 2.4.2)*
- Loadable MPMs
- General purpose Response Body substitution
  - mod_sed
- Auto-convert Response -> RFC 2397 data URL
  - mod_data
- Config file variables
- Cache improvements
- Proxy improvements (‘natch)

Why Proxy Matters

- Cloud puts big focus on horizontal scaling
- Apache httpd still the most frequently used front-end
- Proxy capabilities must be cloud friendly

Proxy Design Drivers

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

What's New: Apache httpd 2.4 proxy

- Reverse Proxy Improvements
  - Supports FastCGI, SCGI in balancer
  - Additional load balancing mechanisms
  - Runtime changing of clusters w/o restarts
  - Support for dynamic configuration
  - mod_proxy_express
  - mod_proxy_html
  - mod_fcgid

Load Balancer

- mod_proxy_balancer.so
- mod_proxy can do native load balancing
  - weight by actual requests
  - weight by traffic
  - weight by busyness
  - lbfactors

Load Balancer

- 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

Load Balancer

- Sticky session support
  - aka “session affinity”
- Cookie based
  - stickysession=PHPSESSID
  - stickysession=JSESSIONID
- Natively easy with Tomcat
- May require more setup for “simple” HTTP proxying

Load Balancer

- 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=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/
```

Embedded Admin

- Allows for real-time
  - Monitoring of stats for each worker
  - Adjustment of worker params
    - lbset
    - load factor
    - route
    - enabled / disabled
    - ...

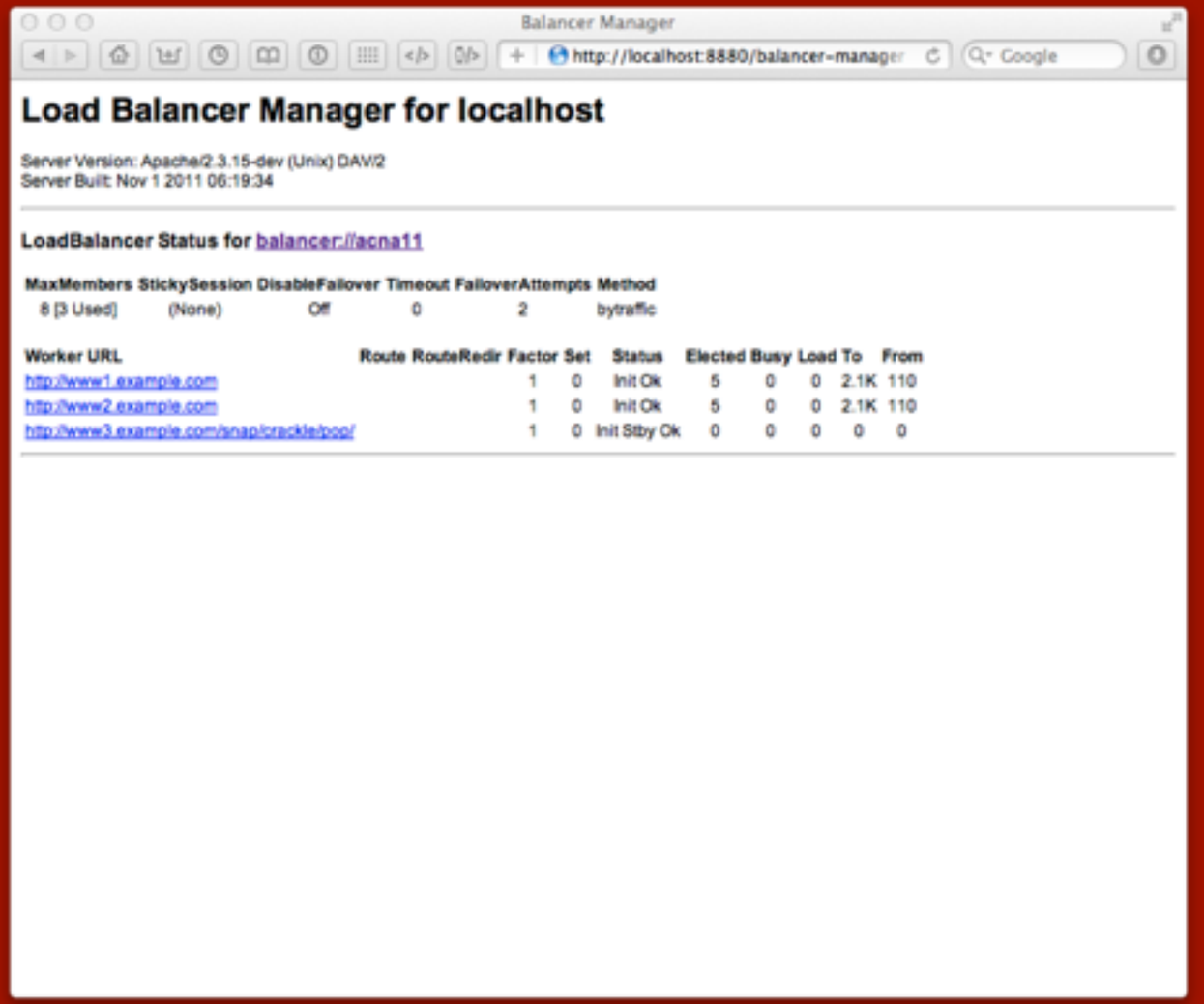
Embedded Admin

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

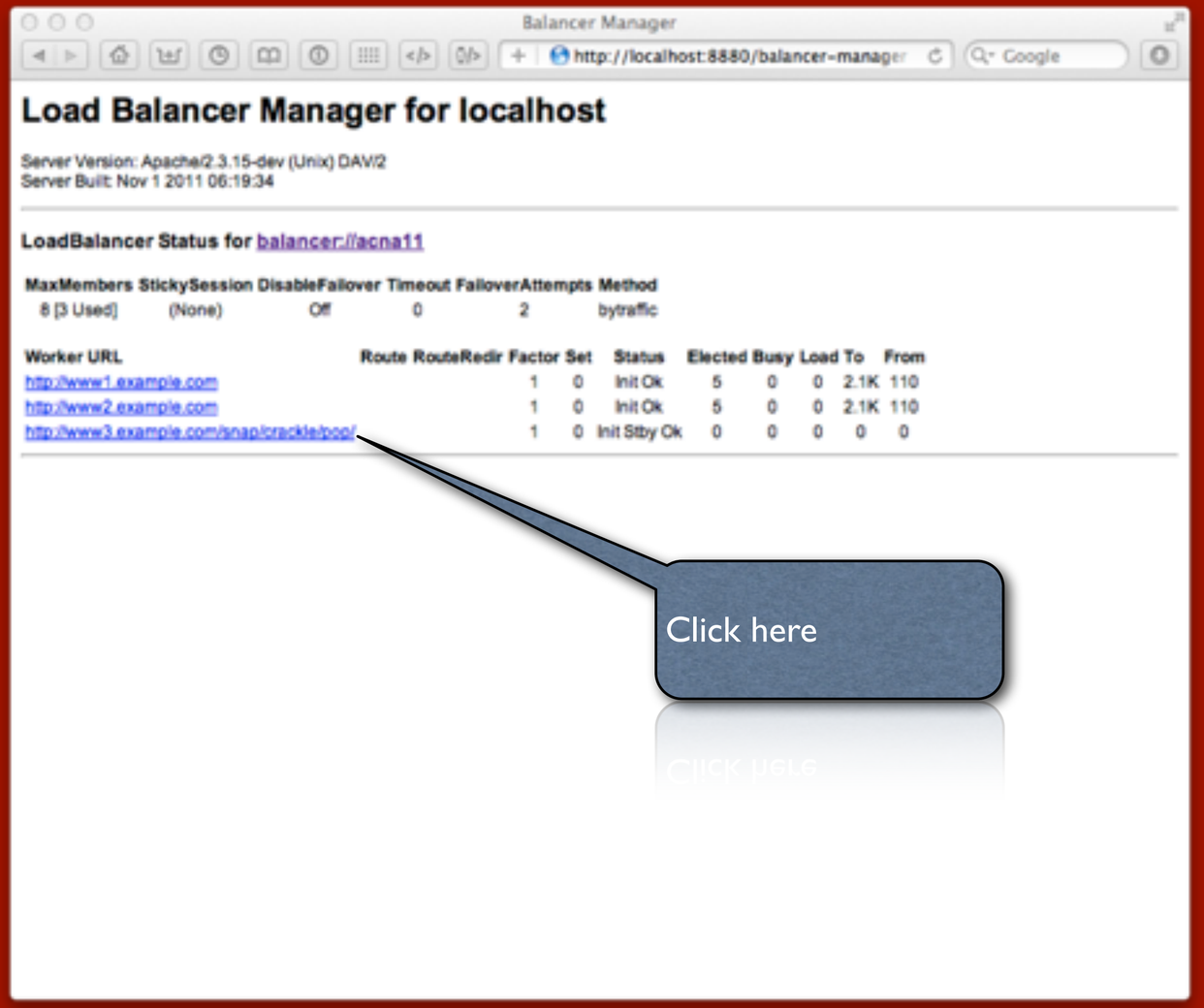
Easy setup

```
<Location /balancer-manager>
  SetHandler balancer-manager
  Order Deny,Allow
  Deny from all
  Allow from 192.168.2.22
</Location>
```

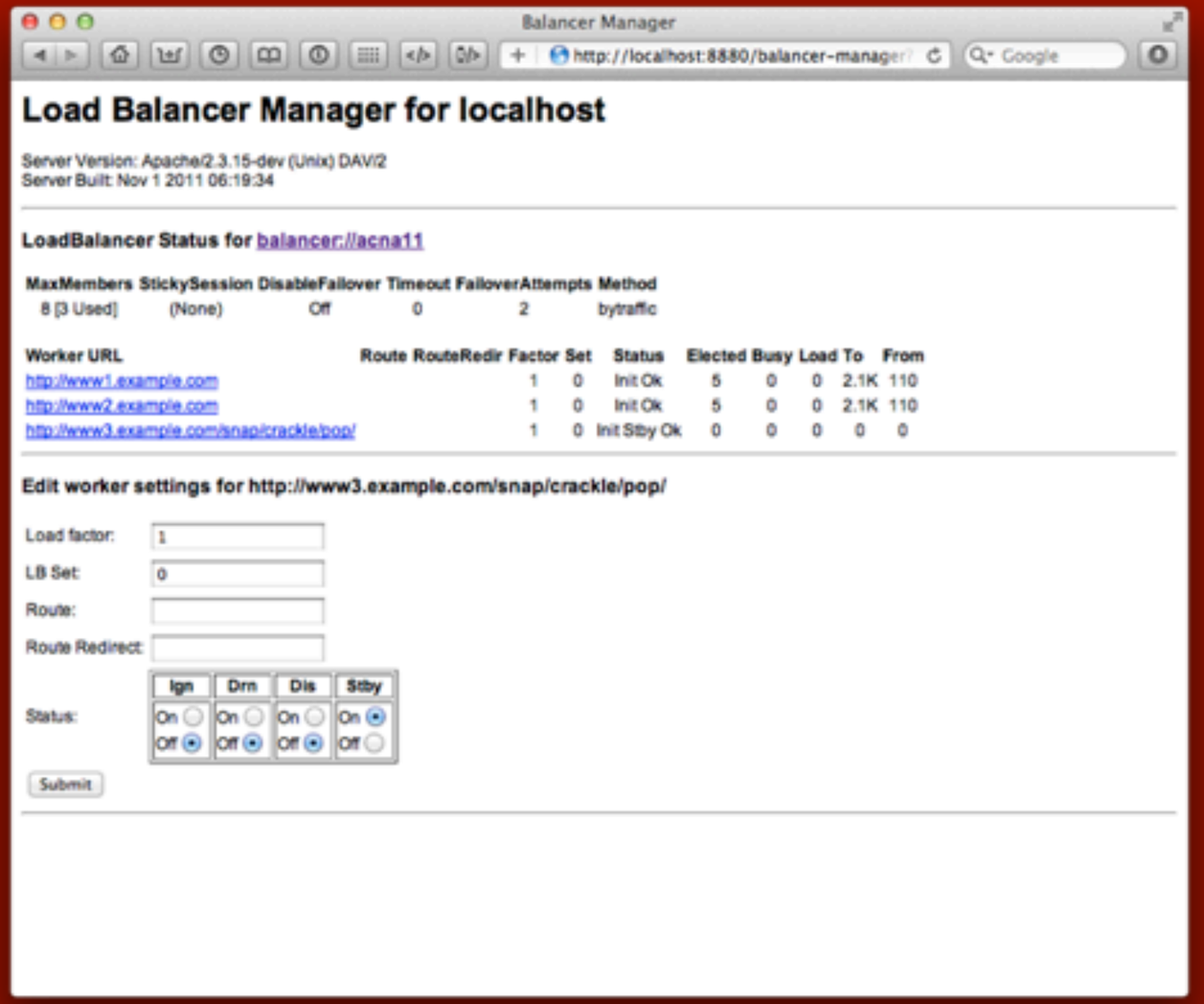
Admin



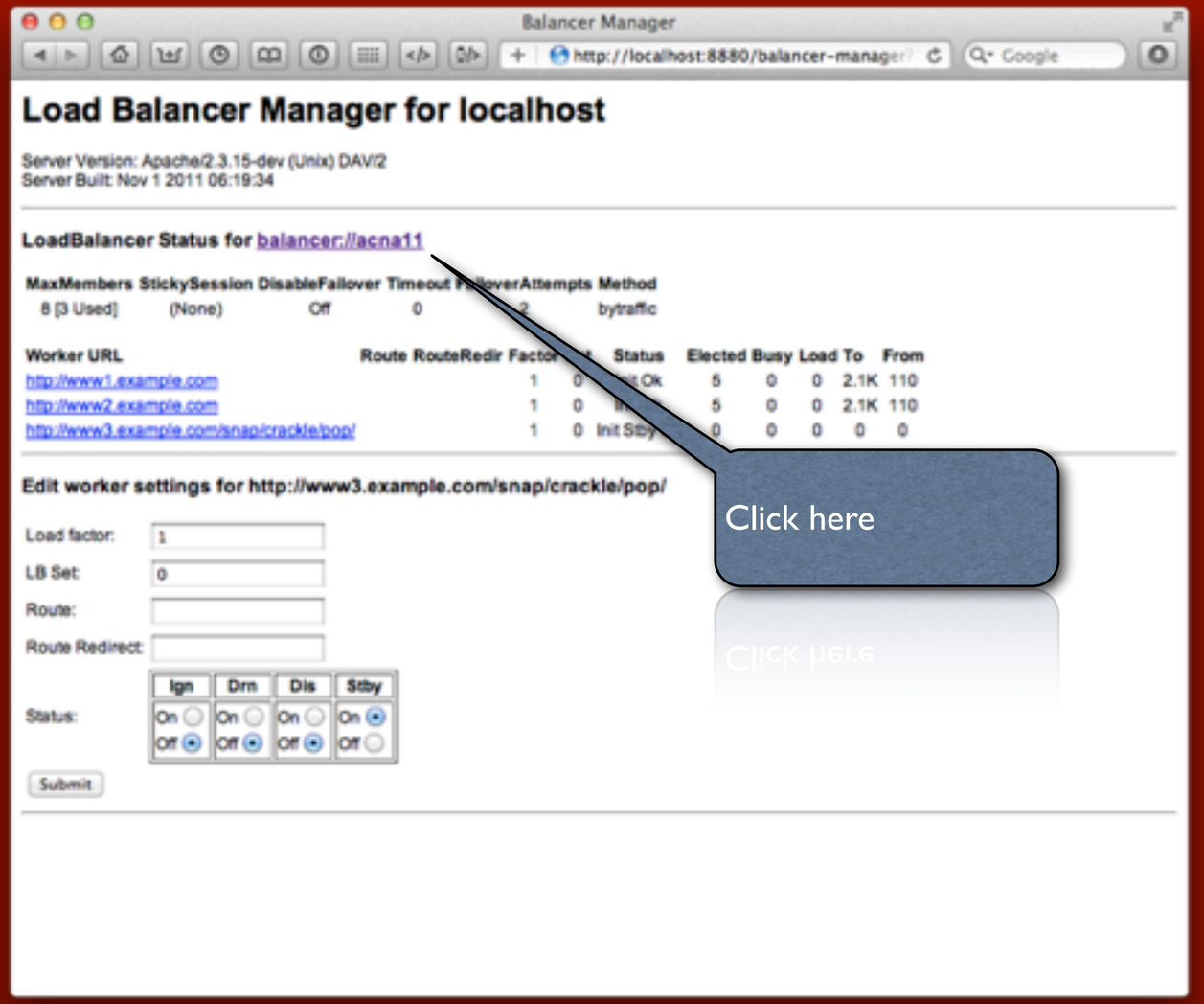
Admin



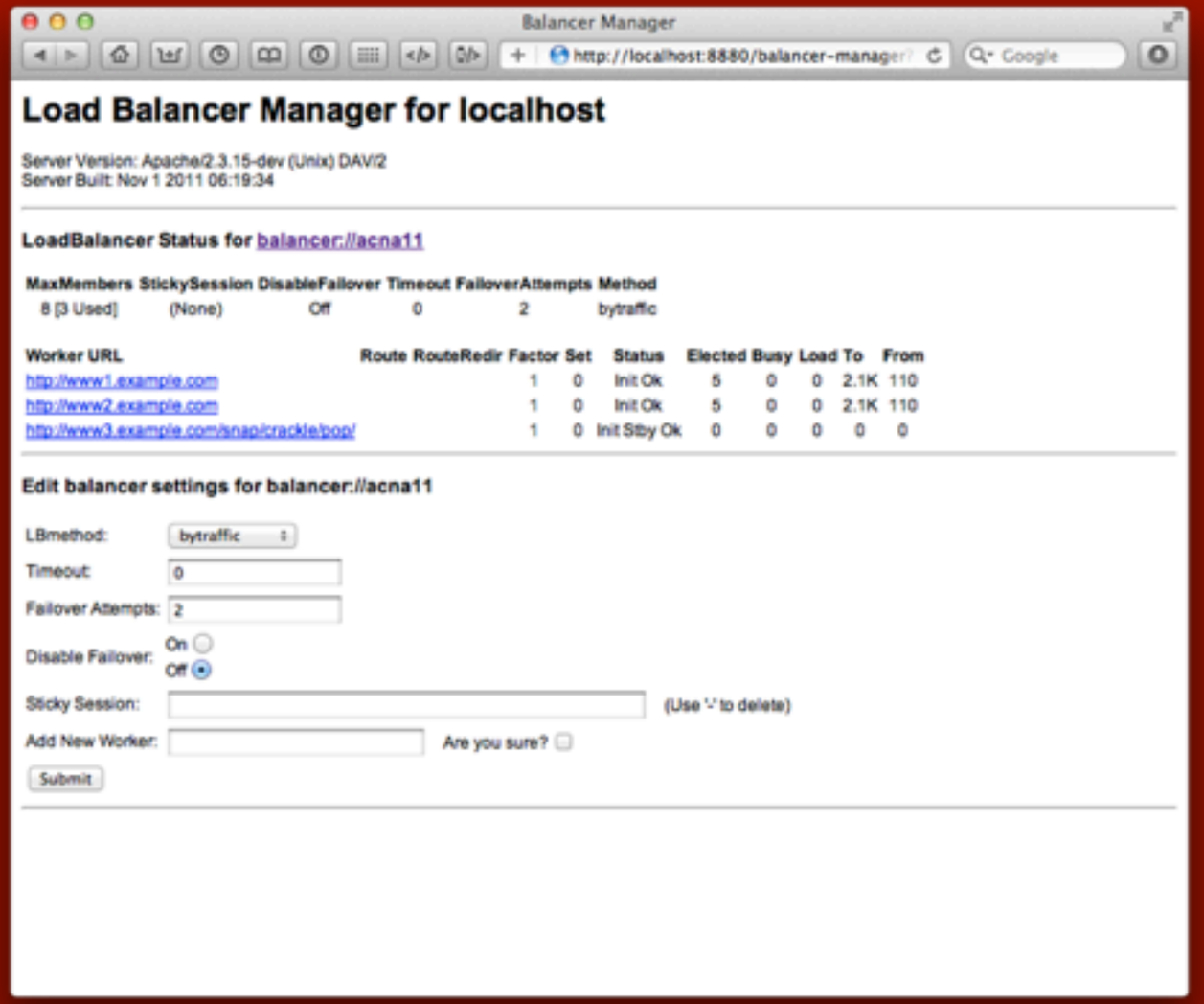
Admin



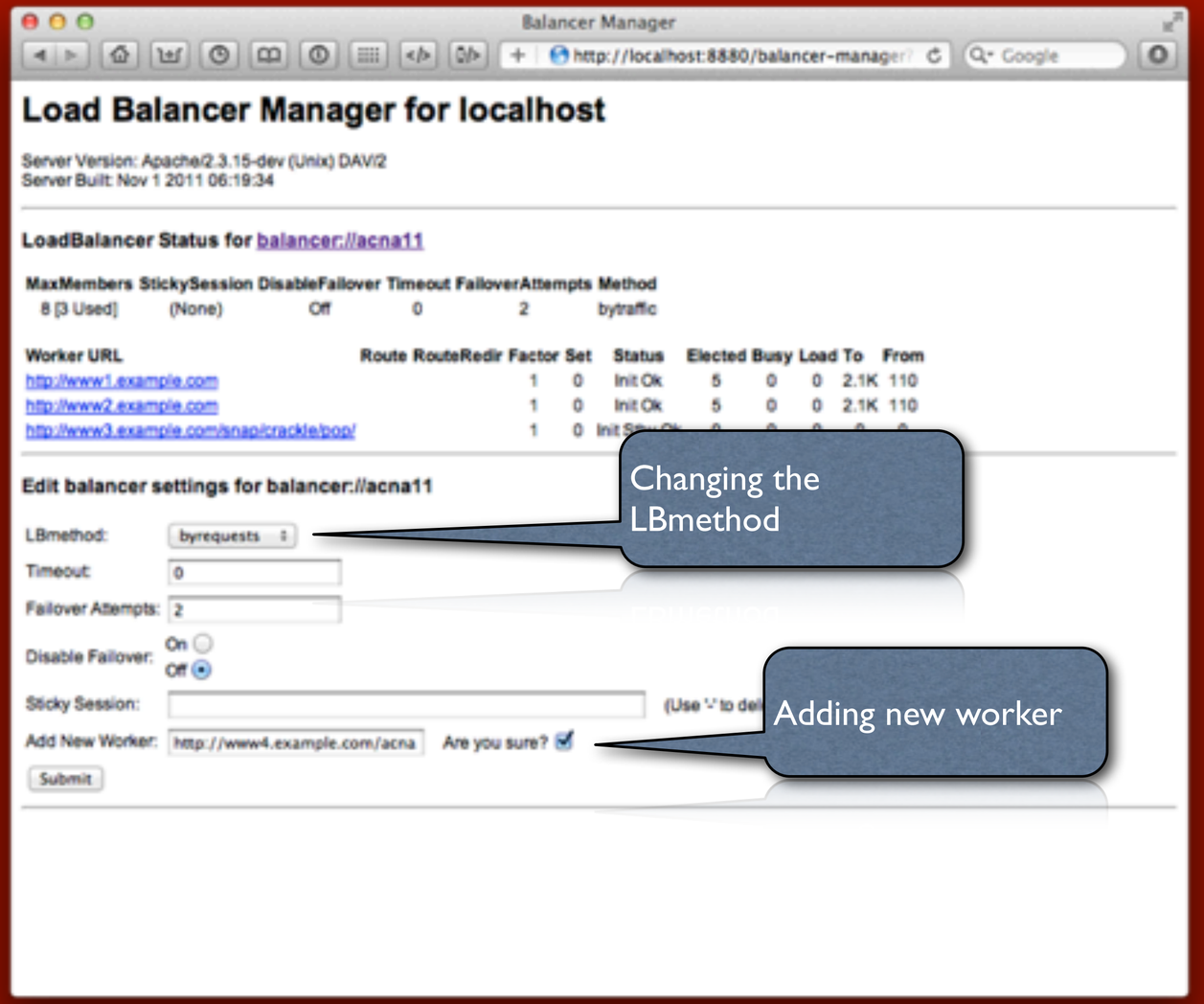
Admin



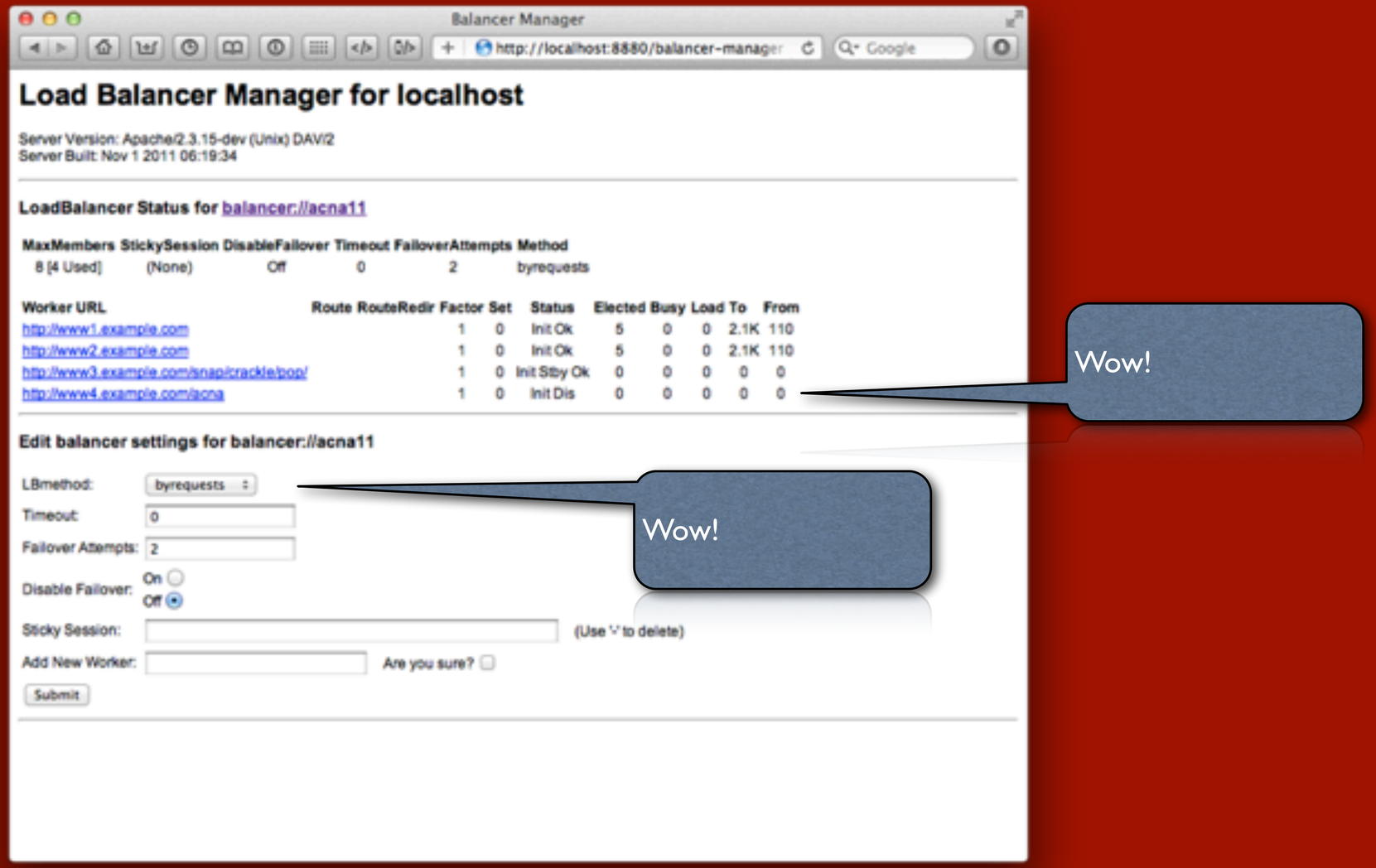
Admin



Admin



Admin



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

ProxyExpress map file

```
##
##express-map.txt:
##

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
```

What's on the horizon?

- Improving AJP
- Adding additional protocols
- More dynamic configuration
  - Adding balancers!

Cloud and Performance

- The Cloud is a game changer for web servers
  - Horizontal scalability is no longer as painful
  - Concurrency is no longer the sole consideration
  - ... or even the primary one
  - What's important now? Transaction Time!
    - Low latency
    - Fast req/resp turnover
  - Does density still matter? *Of course!*
  - Are there environs where concurrency is the bugaboo? *You betcha!* *(but the cloud makes these more and more rare)*

Apache httpd vs nginx

- Why nginx? Everyone asks about it...
- Benchmark: local and reverse proxy transaction times
  - Apache httpd 2.4.1-dev, nginx 1.2.0
  - Fedora 16, Dual Xeon 2.28GHz
  - 4GB memory
  - localhost loopback and external (no firewall)
  - Double checked results: OSX, Ubuntu 10.04

Setup

Setup

Setup 1:

Setup

Setup 1:

Setup 2:

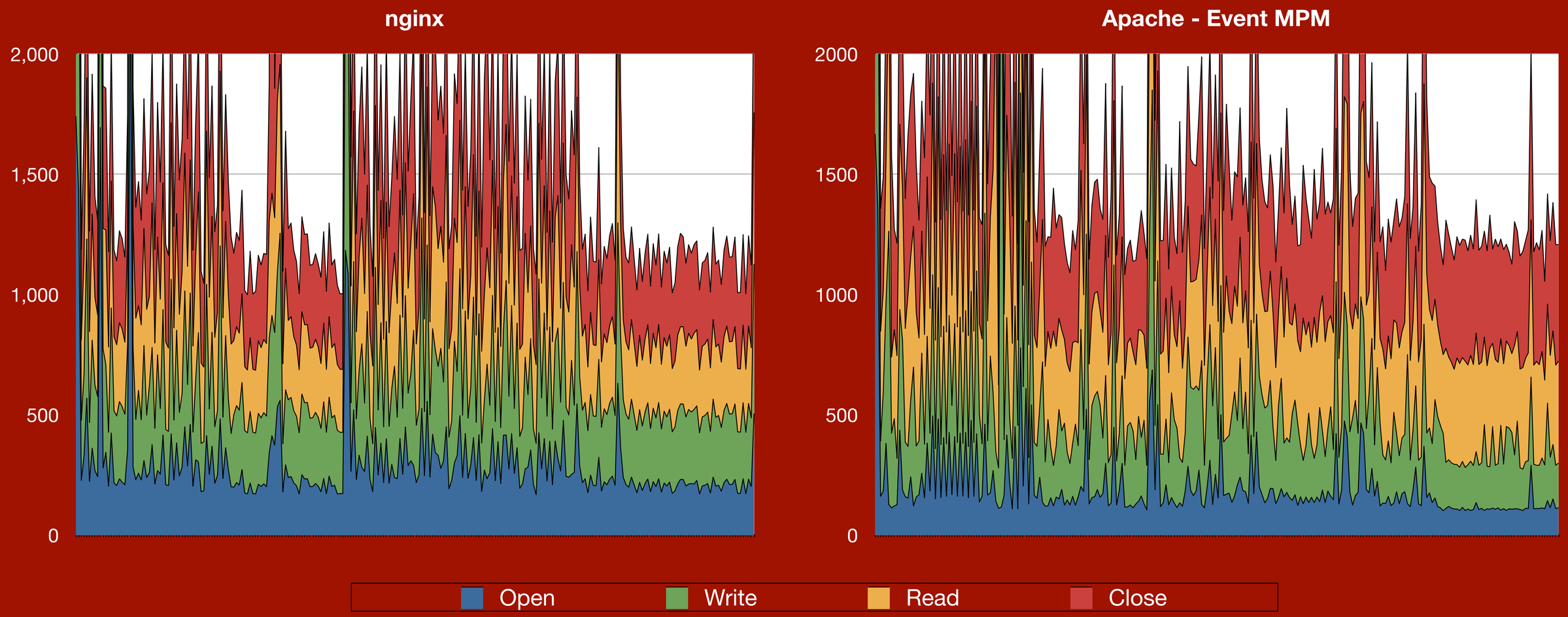
Setup

Setup 1:

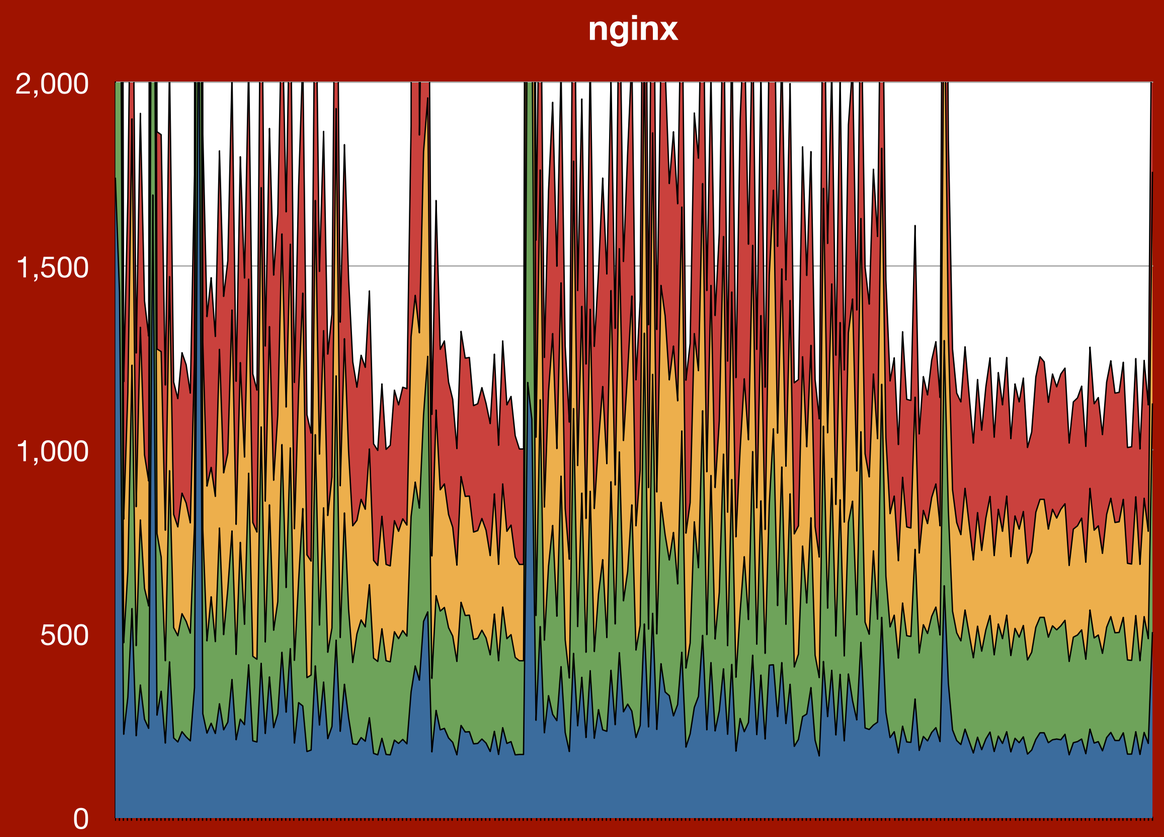
Considerations

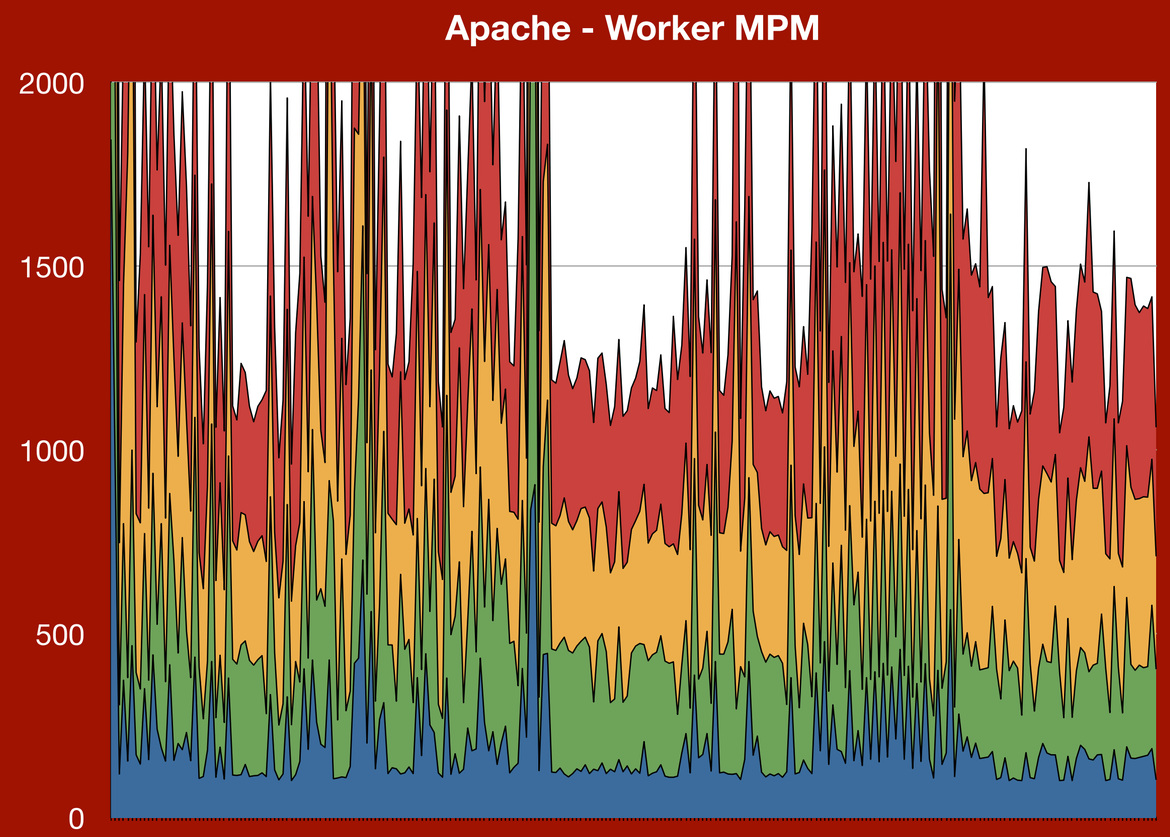
- Multiple benchmarking systems:
  - flood (50/250/5/2, 50/100/5/2, 50/5/5/2)
  - httperf (num-conns=100->10000, numcalls=3)
- Full URL requests (www.example.com/index.html)
- Static local requests
- Static reverse proxy requests
- All Apache httpd MPMs
- No significant “tuning” efforts (mostly out of the box configs)

nginx vs Event (typical)



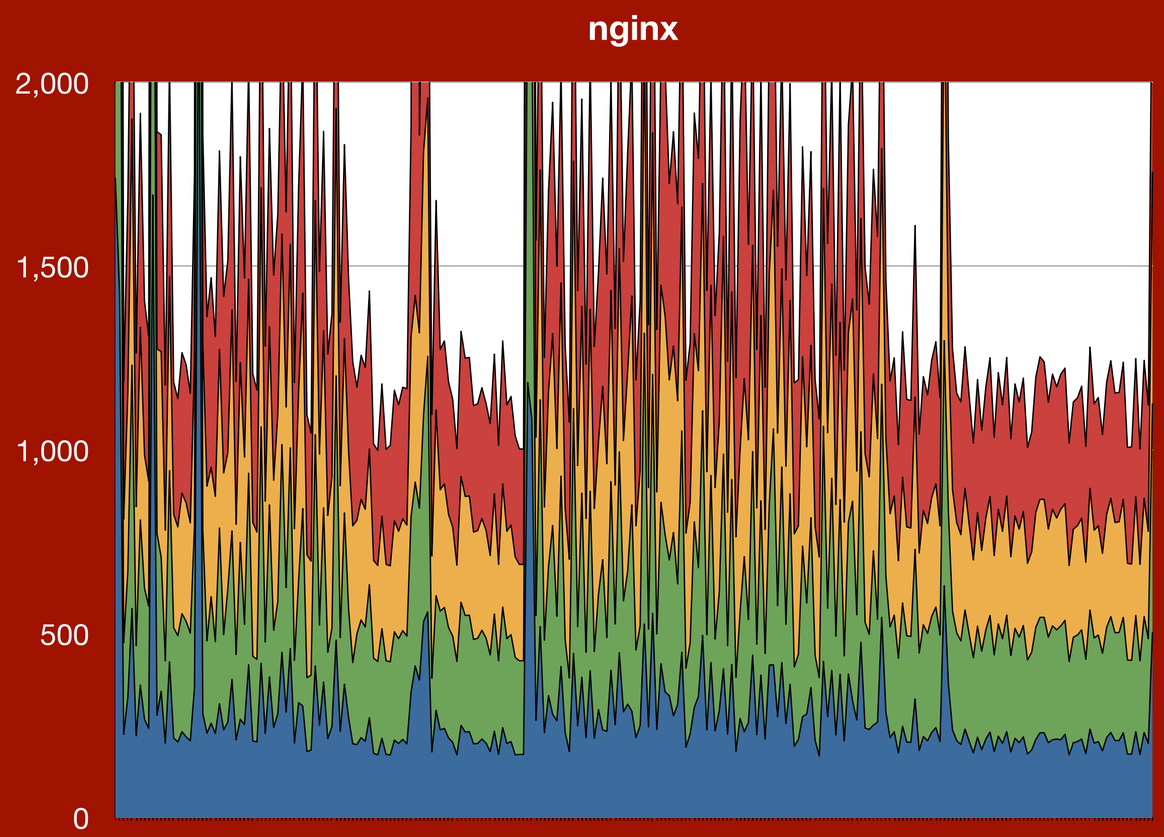
nginx vs Worker (typical)

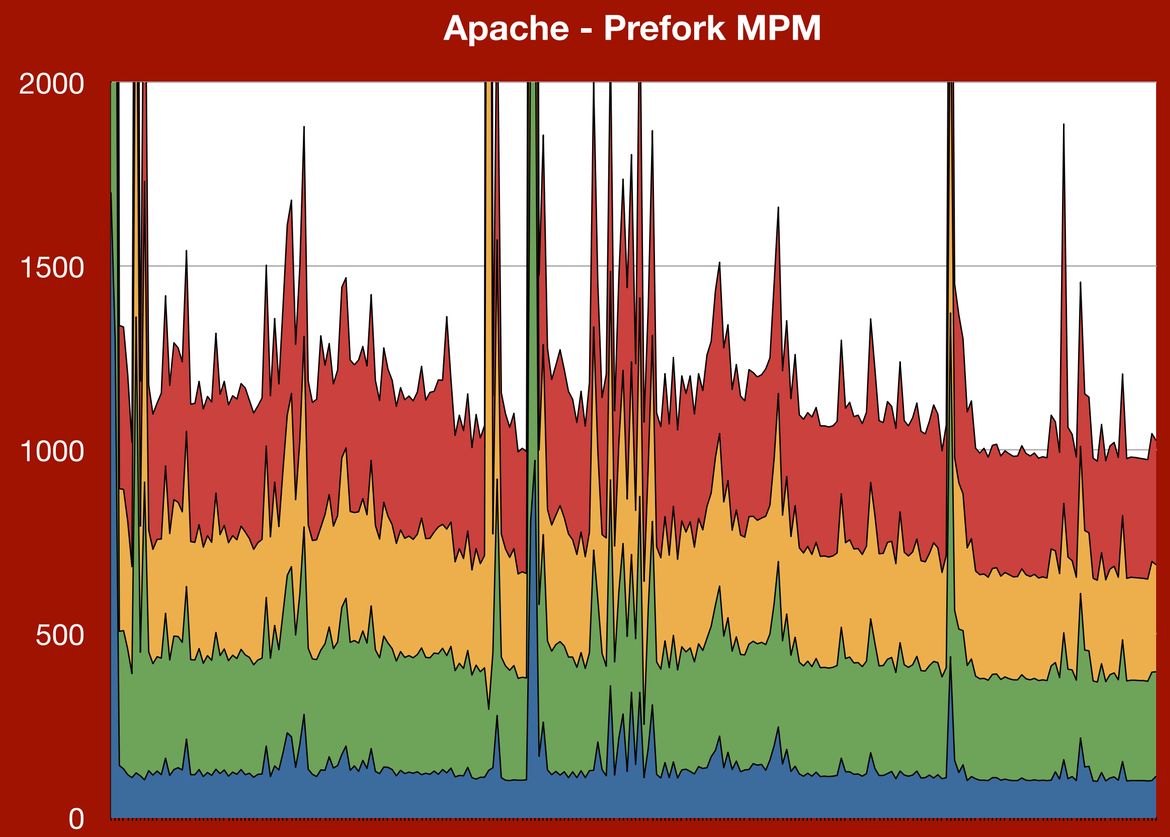




Open Write Read Close

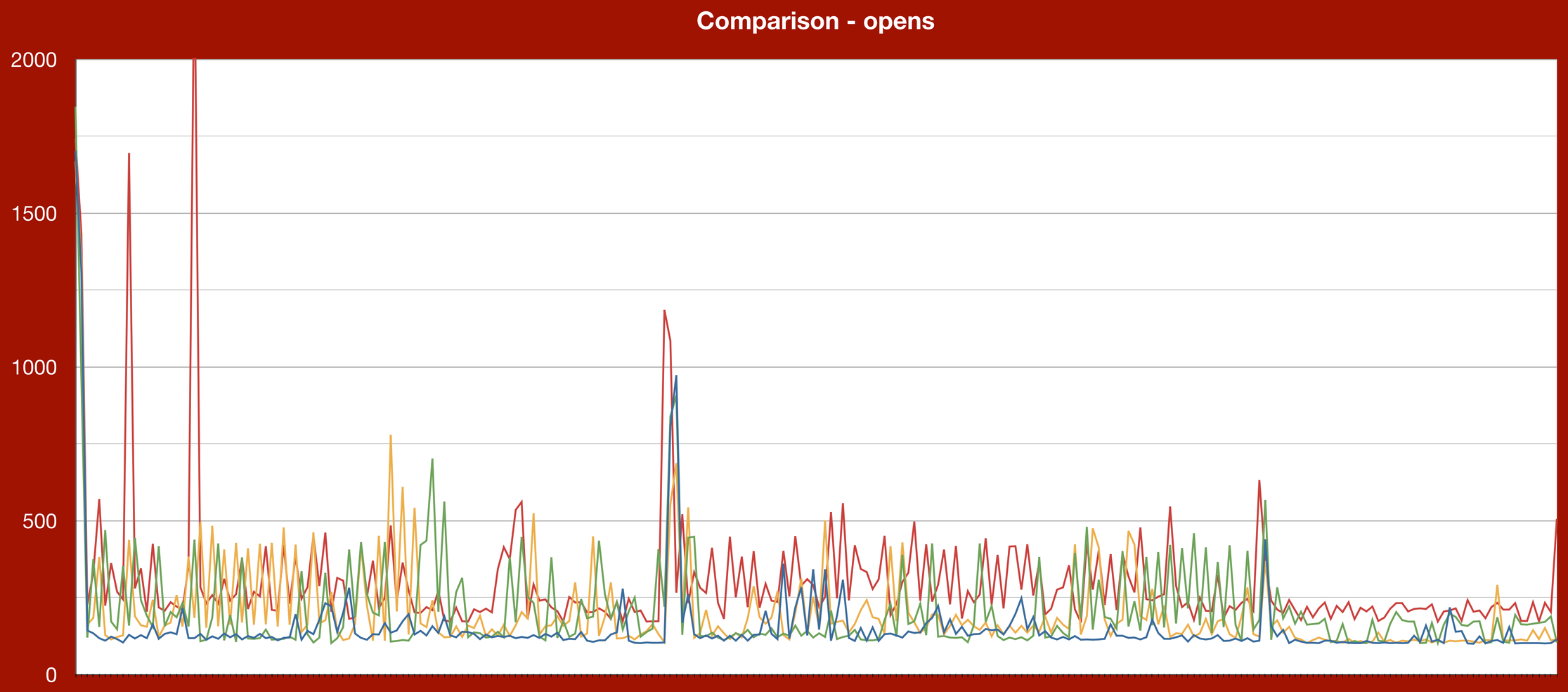
nginx vs Prefork (typical)



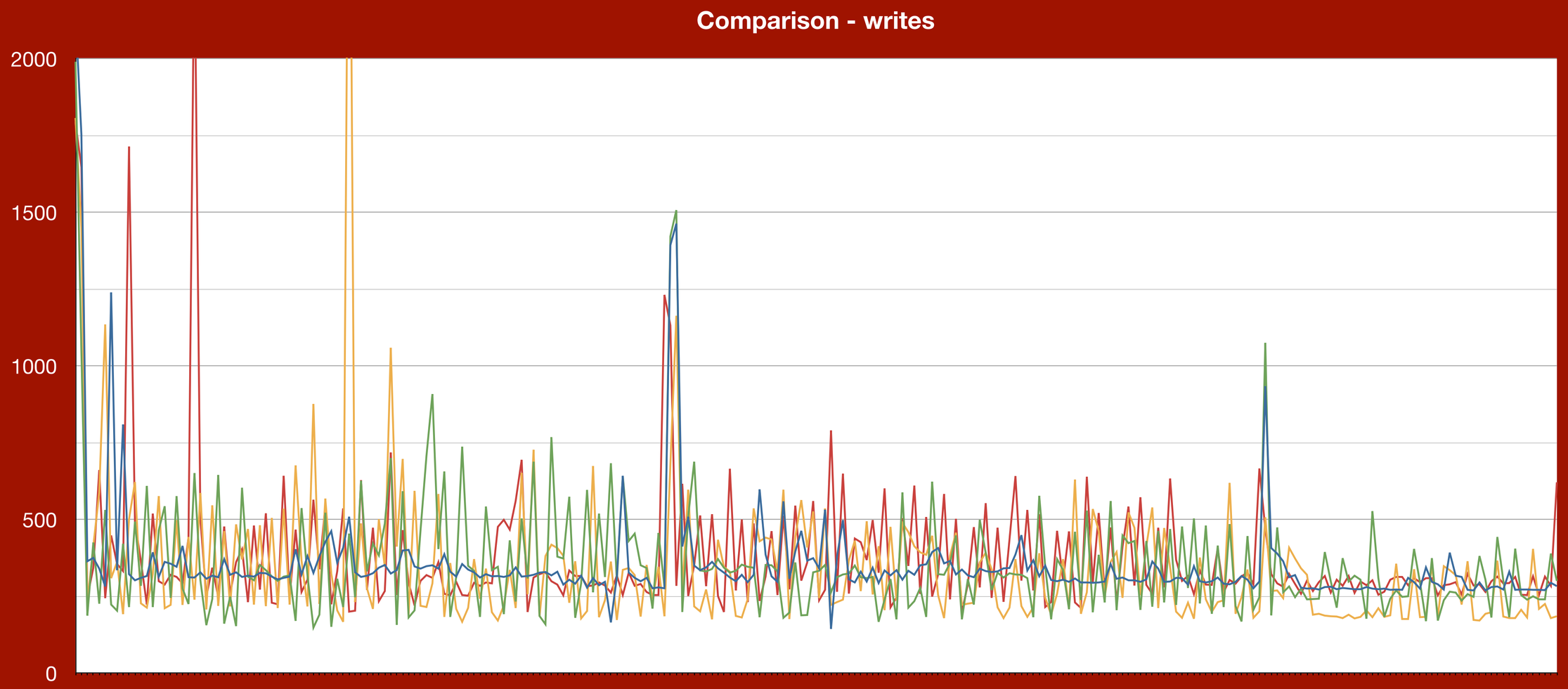


Open Write Read Close

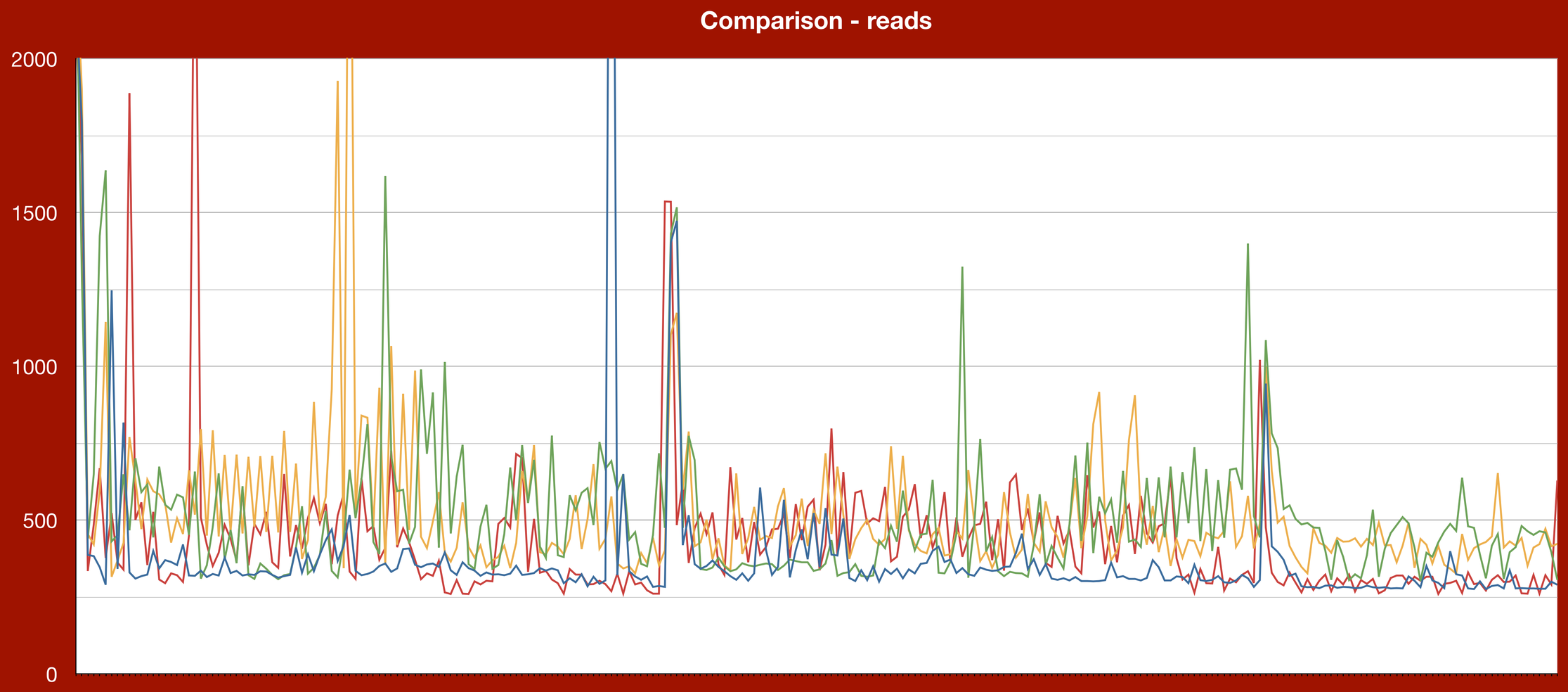
Focus on open()



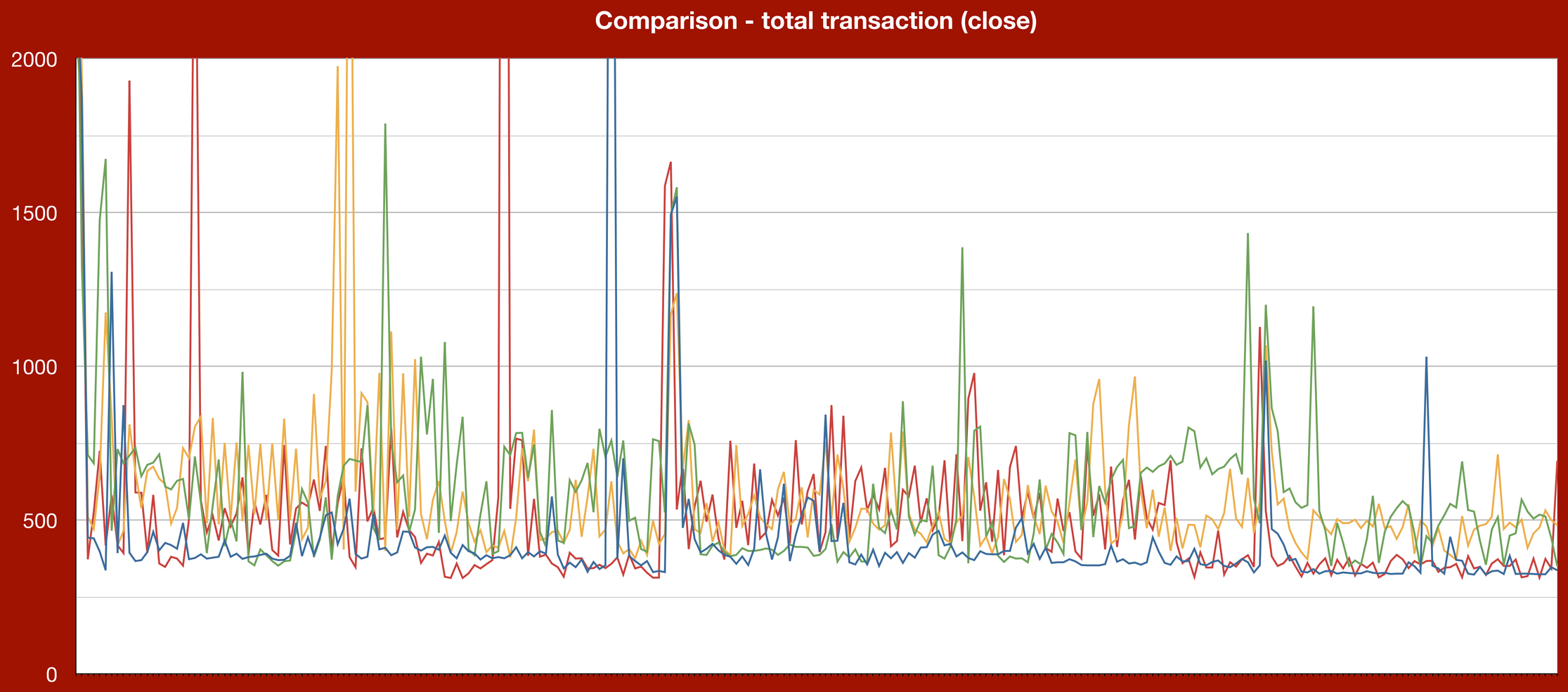
Focus on write()



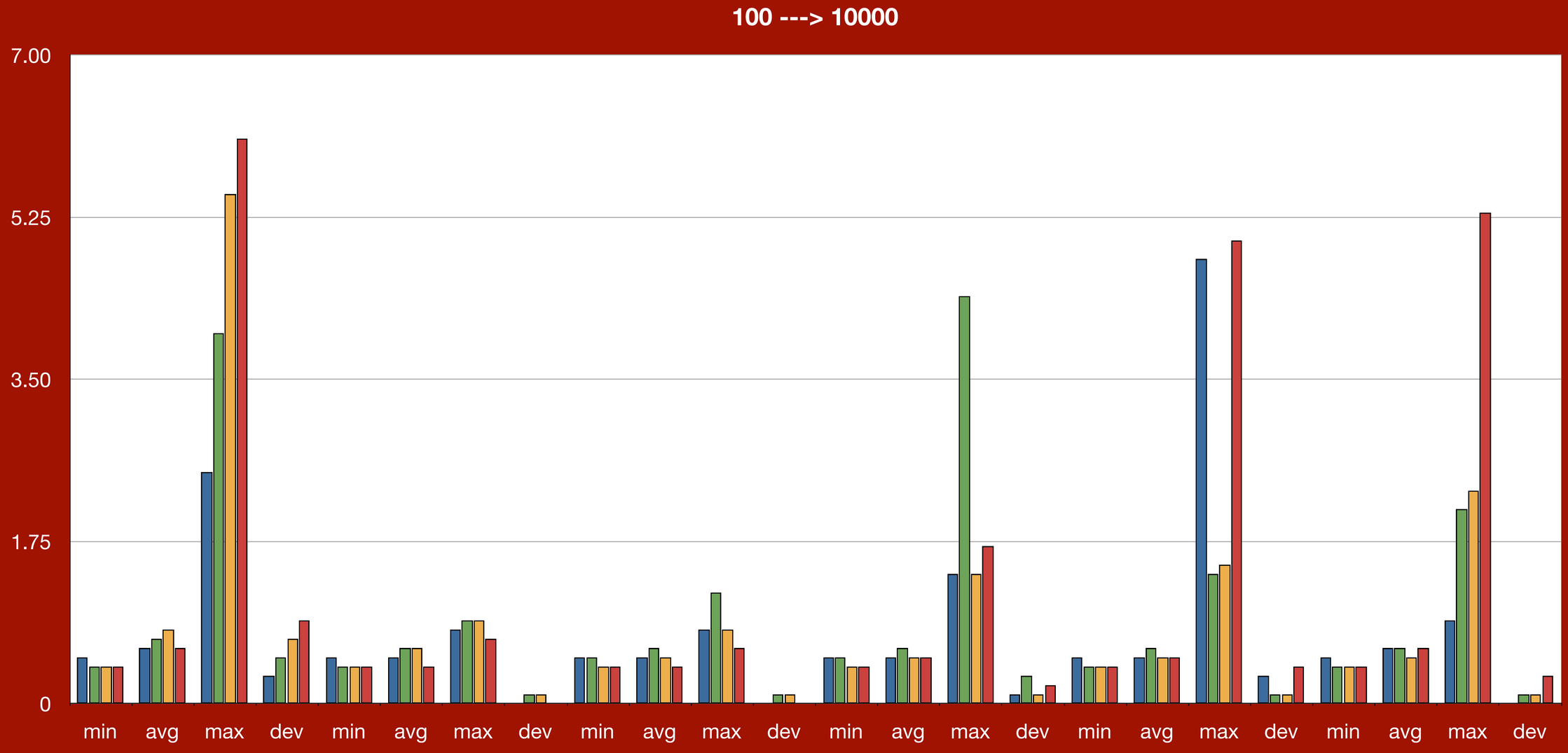
Focus on read()



Total req/resp time



Resp to Req. Bursts - httpref



Independent benchmark

表 1　テスト環境（クライアント）
Table 1　Experimentation Environment for Client

PREV	クライアント
	Intel Core2Duo E8400 3.00GHz
Memory	4GB
NIC	Realtek RTL8111/8168B 1Gbps
OS	CentOS 5.6

表 2　テスト環境（サーバ）
Table 2　Experimentation Environment for Server

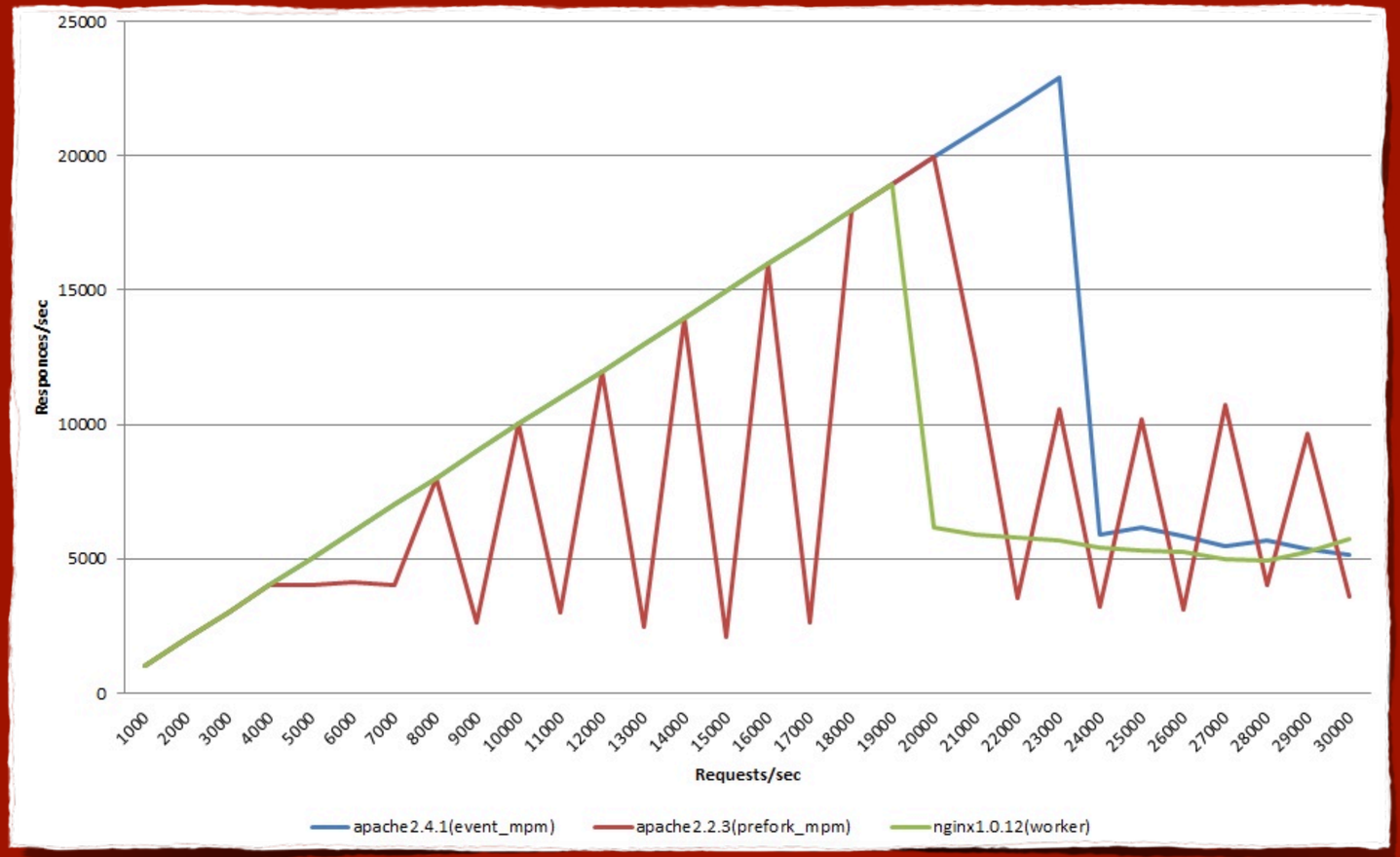
サーバ	
CPU	Intel Xeon X5355 2.66GHz
Memory	8GB
NIC	Broadcom BCM5708 1Gbps
OS	CentOS 5.6

Image 2 of 4

CLOSE

```
#!/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 reboot “Simple” mpm (*currently being done*)
- *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

In conclusion...

- Performance of Apache httpd 2.4 still in the big leagues (and on par with the “big boys” and the fanboi webserver du jure)
- For cloud environs, 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)
- Architecture of Apache httpd 2.4 allows a lot of room for growth and additional functionality (both for the cloud and not)
- There’s still a category of “edge cases” that require nginx, lighttpd, G-WAN, Apache Traffic Server, etc... If that’s you, don’t try to use Apache httpd (but if you do, provide patches!)
- lies, damned lies and benchmarks (sorry, statistics).

Thanks!

- Contact Info:
  -
  -
  -
  -

Jim Jagielski

jim@jaguNET.com jimjag@redhat.com

@jimjag www.jimjag.com

people.apache.org/~jim/presos/