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
  - KeepAliveTimeout 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
    - ...

This work is licensed under a Creative Commons Attribution 3.0 Unported License.
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

![Balancer Manager for localhost](http://localhost:8880/balancer-manager)

**Load Balancer Status for balancer://acna11**

<table>
<thead>
<tr>
<th>Worker URL</th>
<th>Route</th>
<th>RouteRer</th>
<th>Factor</th>
<th>Status</th>
<th>Elected</th>
<th>Busy Load</th>
<th>To</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www1.example.com">http://www1.example.com</a></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Init Ok</td>
<td>5</td>
<td>0 0</td>
<td>2.1K</td>
<td>110</td>
</tr>
<tr>
<td><a href="http://www2.example.com">http://www2.example.com</a></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Init Ok</td>
<td>5</td>
<td>0 0</td>
<td>2.1K</td>
<td>110</td>
</tr>
<tr>
<td><a href="http://www3.example.com/nap/trackle/bop/">http://www3.example.com/nap/trackle/bop/</a></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Init Stby Ok</td>
<td>0</td>
<td>0 0 0 0</td>
<td>0 0</td>
<td>0</td>
</tr>
</tbody>
</table>
Admin

Click here
**Admin**

![Balancer Manager for localhost](image)

**Load Balancer Manager for localhost**

- **Server Version:** Apache/2.3.15-dev (Unix) DAV/2
- **Server Built:** Nov 1 2011 06:19:34

**LoadBalancer Status for balancer://acna11**

<table>
<thead>
<tr>
<th>Worker URL</th>
<th>Route Redirect Factor</th>
<th>Status</th>
<th>Elected Busy Load To</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www1.example.com">http://www1.example.com</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://www2.example.com">http://www2.example.com</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://www3.example.com/snap/crackle/plop/">http://www3.example.com/snap/crackle/plop/</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Edit worker settings for http://www3.example.com/snap/crackle/plop/**

<table>
<thead>
<tr>
<th>Load factor</th>
<th>LB Set</th>
<th>Route</th>
<th>Route Redirect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ign</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>On</td>
</tr>
</tbody>
</table>

![Click here](image)
Admin

Changing the LB method

Adding new worker
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

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

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:

Setup 2:

Setup 3:
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)

Comparative graph showing performance metrics for nginx and Apache - Worker MPM. The graph is color-coded for Open, Write, Read, and Close operations.
nginx vs Prefork (typical)
Focus on open()

Comparison - opens

- Prefork
- Worker
- Event
- nginx
Focus on write()
Focus on read()
Resp to Req. Bursts - httpref

100 ---> 10000

prefork  | worker  | event  | nginx  |
---      | --      | --     | --     |
1.75     | 3.50    | 5.25   | 7.00   |

This work is licensed under a Creative Commons Attribution 3.0 Unported License.
Independent benchmark

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

This work is licensed under a Creative Commons Attribution 3.0 Unported License.
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!

- Jim Jagielski
- jim@jaguNET.com  jimjag@redhat.com
- @jimjag  www.jimjag.com
- people.apache.org/~jim/presos/