Advanced Topics in Apache HTTP Server 2.2
What's New / Perf. Tuning

Jim Jagielski
http://www.jimjag.com/
jim@jaguNET.com
About me

- Longtime active contributor (July/Aug 1995)
- ASF Co-founder
- Other ASF titles as well
- Principle Engineer / COSS / Chief Architect for SpringSource
- Husband, father, all around nice guy
How did we get here?

- A short history of Apache HTTP Server (at least regarding 2.2)
  - Apache 1.3.0 released in June 1998
  - Apache 2.0a1 released in March 2000 (at ApacheCon!)
  - First GA version of Apache 2.0 released on April 2002: Apache 2.0.35
  - Apache 2.2.0 released on Dec. 2005
  - We are now at 2.2.11 (2.2.12 maybe soonish??)
Apache 2.0

- Apache 2.0 was designed to address shortcomings in 1.3
  - MPM
  - Module ordering dependencies
  - Hooks
  - Filters
  - Protocol modules
  - Sub-module concept
  - APR
  - IPV6
So Apache 2.0 was...

- Basically a rewrite of Apache 1.3
- An opportunity to rethink how Apache works
- An opportunity to make setup and config more elegant
  - de-merge proxy and cache
  - better authen. and authoriz
Did we succeed?

• To a great extent yes
• But some things lagged behind
• Or didn’t quite turn out the way we hoped
Why 2.2?

• Despite advances in 2.0.x tree, improvements needed to be made
• But those improvements would break the API
• Plus, many of them required later versions of APR
Apache 2.2 Goals

• Bring all functionality up to parity
• Be an evolutionary step from 2.0
• Incremental, logical steps
• 2.0 modules require (for the most part) just a simple recompilation
• Keep what 2.0 did right, and improve on remaining features
Apache 2.2 cool stuff

- Large file support
- Graceful stop
- mod_dbd
- mod_substitute
- Better Debugging and info
- Caching
- Event MPM
- Authn/Authz
- Proxy
Large file support

- 2GB is no longer a stupid limit
- Much better 64 bit awareness
- And much better behavior on 32 bit systems
- Thanks to APR
Graceful stop

• We all know about graceful restart
• Now Apache will also gracefully stop
  – when shutting down, Apache will let existing requests finish
  – But what about really, really long or nasty requests?
    • `GracefulShutdownTime`
      – # == number of seconds grace time
      – 0 == forever
Graceful start

• We have:
  – graceful restart
  – graceful shutdown

• How about a graceful start?
Ha ha

• Very funny
mod_dbd

• The problem
  – Lots of modules...
  – ... using lots of SQL connections
  – EG: authn/authz, logging, PHP...

• Even worse with threaded MPMs

• mod_dbd manages all that for you
  – ap_dbd_open, ap_dbd_prepare, ...

• Connection pooling comes to the party
mod_substitute

• The problem:
  – We need to do on-the-fly content substitution

• The solution:
  – mod_substitute
  – A “sed-like” interface
    • s/foo/bar/
Better Debugging

• mod_dumpio
  – Dumps all IO to the error log
  – Yep, all of it
    • DumpIOLInput On
    • DumpIOLOutput On
    • DumpIOLLogLevel Notice
  – What about SSL?
    • Dumping is done right after decryption or right before encrypting
mod_dumpio: dumpio_in [getline-blocking] 0 readbytes
mod_dumpio: dumpio_in (data-HEAP): 16 bytes
mod_dumpio: dumpio_in (data-HEAP): GET / HTTP/1.1
mod_dumpio: dumpio_in [getline-blocking] 0 readbytes
mod_dumpio: dumpio_in (data-HEAP): 13 bytes
mod_dumpio: dumpio_in (data-HEAP): Accept: */*
...
mod_dumpio: dumpio_out
mod_dumpio: dumpio_out (data-HEAP): 291 bytes
mod_dumpio: dumpio_out (data-HEAP): HTTP/1.1 200 OK
Date: Thu, 12 Oct 2006 15:35:52 GMT
Server: Apache/2.2.4-dev (Unix) DAV/2
Last-Modified: Fri, 10 Dec 2004 14:17:55 GMT
ETag: "7b3e83-2c-9eedeac0"
Accept-Ranges: bytes
Content-Length: 44
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html
mod_dumpio: dumpio_out
mod_dumpio: dumpio_out (data-FILE): 44 bytes
mod_dumpio: dumpio_out (data-MMAP): <html><body><h1>It works!</h1></body></html>
mod_dumpio: dumpio_out (metadata-EOS): 0 bytes
Better Debugging

• **mod_log_forensic**
  – forensic logging of each request
  – Each request results in 2 log lines
    • Initial request with unique ID
      – +yQtJf8AB4AAFNXQY | GET /manual/...
    • Response done “tag”
      – -yQtJf8AB4AAFNXQY
  – track and trace requests
Better debugging

• mod_info:
  – ?config: Just the configuration directives, not sorted by module
  – ?hooks: Only the list of Hooks each module is attached to
  – ?list: Only a simple list of enabled modules
  – ?server: Only the basic server information
mod_info screensnap
mod_info screensnap
Caching

• Dirty little 2.0 secret
  – When we separated mod_proxy and mod_cache, mod_cache didn’t get a lot of TLC

• Code was not clean
• Nasty performance
• disk cache lacked good maintenance
• Lacked RFC compliance
Apache 2.2 Caching

• No longer experimental!
• Caching stores copies of static or dynamic content (if possible) for quick access
• `mod_cache`:
  – The caching framework
• `mod_disk_cache / mod_mem_cache`
  – Determines cache implementation
Caching modules

• **mod_disk_cache**
  - Stores cached material on file system
  - Key based access

• **mod_mem_cache**
  - Stores cached material in shared memory cache.
  - Caches open file descriptors.
  - Caches content object.
disk vs. mem

• Lots of work done on both
• mem
  – fast because it uses shared memory
  – locking
  – restarts make cache go bye bye
  – other issues
• disk
  – long term storage
  – zero-copy transfer
Simple Config

• Just cache CSS files

LoadModule cache_module modules/mod_cache.so
LoadModule mem_cache_module modules/mod_mem_cache.so

CacheEnable mem /css

MCacheSize 1024
MCacheMaxObjectCount 100

MCacheMinObjectSize 1
MCacheMaxObjectSize 2048
htcacheclean

- mod_disk_cache places no limits on disk usage
- htcacheclean cleans up and limits utilization
  - run manually or in daemon mode
  - htcacheclean -p/var/db/httpd/cache \
    -l250M -d30
Event MPM

- Still considered experimental
- Seeing some extensive use
- The problem:
  - Those nasty keepalives
  - The worker thread is stuck waiting for the next persistent request
- The solution:
  - Pop that “waiting” connection back into the listener thread’s domain
An illustration to help

- Sockets of “interest”
- Listener
- Storage of “Ready” sockets
- Keepalive connection
- Actual workers
Authn / Authz

• Authorization
  – Permit access to a resource based on who/what/where/why/when

• Authentication
  – Determine who/what/where/why/when

• Two different concepts – 2.2 divides them.
Two implementations

- **mod_auth_basic**
  - Speaks PLAIN TEXT user and password over the wire – not secure

- **mod_auth_digest**
  - Speaks a hash of the host digest domain, user and password, this is much more secure over http: connections!

- Most browser supports Digest today, many ‘custom clients’ don’t
Providers for info

- **mod_authn_file**
  - the classic, a flat list of users and slows quickly as the list grows

- **mod_authn_dbm**
  - the classic, faster solution, plug into Berkeley DB, GDBM, SDBM etc

- **mod_authn_dbd**
  - the newest solution, use an Oracle / MySQL table for your user store
Providers for info

- **mod_authn_anon**
  - the Anonymous backstop, no password validation

- **mod_authn_default**
  - the absolute backstop (not-authenticated result)

- **mod_authn_alias**
  - Group the many directives of a provider into an `<AuthnProviderAlias>` block.
Authz

• mod_authz_user
  – Grant/restrict access based on Authenticated user

• mod_authz_groupfile
  – Store group -> users associations in a flat file

• mod_authz_dbm
  – Store user || group in a Berkley DB / GDBM flat database
Authz

- **mod_authz_owner**
  - Access files by OWNER, either user or group

- **mod_authz_host**
  - What you knew as ‘access’, restrict by the client’s IP/hostname

- **mod_authz_default**
  - the ‘backstop’ when no authorization is matched.
Authn / Authz

- **mod_authnz_ldap**
  - Both authn user and authz group principals apply at once to users authorized against an LDAP data store.
  - Basically, it does both
Simple example

AuthType Basic
AuthName "Restricted Files"
AuthUserFile /path-to/htpasswd
AuthBasicProvider file
Require user jim
<AuthnProviderAlias ldap ldap-alias1>
   AuthLDAPBindDN cn=youruser,o=ctx
   AuthLDAPBindPassword yourpassword
   AuthLDAPURL ldap://ldap.host/o=ctx
</AuthnProviderAlias>

Alias /secure /webpages/secure
<Directory /webpages/secure>
   Order deny,allow
   Allow from all
   AuthBasicProvider ldap-alias1
   AuthType Basic
   AuthName LDAP_Protected_Place
   AuthzLDAPAuthoritative off
   require valid-user
</Directory>
Proxy

• Becoming a robust but generic proxy implementation
• Supports various protocols
  – HTTP, HTTPS, CONNECT, FTP
  – AJP, FastCGI (coming “soonish”)
• Load balancing
• Clustering, failover
Load Balancer

• mod_proxy can do native load balancing
  – weight by actual requests
  – weight by traffic

• LB algo’s are impl as providers
  – easy to add
  – no core code changes required
Load Balancer

- Backend connection pooling
- Sticky session support
- Cluster set with failover
  - Lump backend servers as sets
  - balancer will try lower-valued sets first
- Hot standby
- Configurable in real-time
Example

```
<Proxy balancer://foo>
    BalancerMember http://php1:8080/ loadfactor=1
    BalancerMember http://php2:8080/ loadfactor=4
    BalancerMember http://phpbkup:8080/ loadfactor=4 status=+h
    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/
ProxyPass /serv/ balancer://javaapps/

ProxyPass /images/ http://images:8080/
```
### Load Balancer Manager for localhost

Server Version: Apache/2.2.4-dev (Unix) mod_ssl/2.2.4-dev OpenSSL/0.9.8d DAV/2
Server Built: Nov 2 2006 12:16:28

#### LoadBalancer Status for `balancer://foo`

<table>
<thead>
<tr>
<th>StickySession</th>
<th>Timeout</th>
<th>FailoverAttempts</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>byrequests</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Worker URL</th>
<th>Route</th>
<th>RouteRedir</th>
<th>Factor</th>
<th>Set</th>
<th>Status</th>
<th>Elected To</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>http://php1:8080/</code></td>
<td>1</td>
<td>0</td>
<td>Ok</td>
<td>311</td>
<td>125K</td>
<td>446K</td>
<td></td>
</tr>
<tr>
<td><code>http://php2:8080/</code></td>
<td>4</td>
<td>1</td>
<td>Ok</td>
<td>1232</td>
<td>433K</td>
<td>1743K</td>
<td></td>
</tr>
<tr>
<td><code>http://phpbackup:8080/</code></td>
<td>4</td>
<td>0</td>
<td>Sby Ok</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

#### Edit balancer settings for `balancer://foo`

- StickySession Identifier: 
- Timeout: 0
- Failover Attempts: 2
- LB Method: bytraffic

[Submit]
Oh yeah

• ProxyPassMatch
  – ProxyPass can now take regex’s instead of just “paths”
    • ProxyPassMatch ^(/.*\.(gif))$ http://backend.example.com$1
  – JkMount migration

• Shhhh
  – ProxyPass ~ ^(/.*\.(gif))$ http://backend.example.com$1

• mod_rewrite is balancer aware
• ProxyPassReverse is NOW balancer aware! (as of 2.2.9)
• The below will work:

```
<Proxy balancer://foo>
  BalancerMember http://php1:8080/    loadfactor=1
  BalancerMember http://php2:8080/    loadfactor=4
</Proxy>

ProxyPass /apps/ balancer://foo/

ProxyPassReverse /apps balancer://foo/
```
Workaround for $\leq 2.2.8$

• Instead, do this

```xml
<Proxy balancer://foo>
  BalancerMember http://php1:8080/ loadfactor=1
  BalancerMember http://php2:8080/ loadfactor=4
</Proxy>

ProxyPass /apps/ balancer://foo/

ProxyPassReverse /apps http://php1:8080/
ProxyPassReverse /apps http://php2:8080/
```
in trunk

• Cool new modules:
  – mod_session
  – mod_heartbeat / mod_heartmonitor
  – mod_slotmem
  – mod_lua (was: mod_wombat)
  – mod_sed

• Some additional potential backports
  – FastCGI proxy module
  – FastCGI module
in trunk

• SNI support (maybe backport?)
• True async server support
  – serf: http://code.google.com/p/serf/ ?
• 2.3.2 alpha
Apache Golden Rules

💡 Thou shalt avoid fork()

❖ In Ideal World: Apache creates processes once and never again

❖ Try to anticipate loading (look at logs)

❖ Thou shalt conserve resources

❖ Thou shalt not configure bogusly
Thread Creation -

- ThreadLimit, ThreadsPerChild and Min/MaxSpareThreads
  - For scalability, use large numbers of threads
  - For robustness, prefer larger number of process (decrease ThreadsPerChild)
  - When we need more idle threads, Apache will spawn off additional child processes
MaxRequestsPerChild:

- Intended to handle memory leaks
- After X "requests", child will exit
- Setting to 0 disables this

Further Limiting Unneeded Forks
KeepAlives prevent connection (socket) open/close for each request

Useful when a view contains lots of small content

Not so useful for dynamic content
Keepalives

Why not always use them?

- “idle” workers
- Result in excessive forking

Advice:

- `KeepAliveTimeout 2`
- Monitor your site’s usage
- 2.2’s Event MPM might be useful
Smart Configuration

- Configure Apache to not waste resources or horsepower
- Save cpu cycles for request/response
- Dependent on loading and traffic
DNS

- DNS lookups are out-of-process
- External factors influence your performance

Advice:
- HostnameLookups Off
- Apache will do DNS lookup when needed
- Use IP addresses when possible
Content Configuration Tips

 Website Content and symbolic links
   FollowSymLinks
   SymlinksIfOwnerMatch
 htaccess files (per directory access control)
   Expensive stat() on each directory
   AllowOverrides None
   Only enable if needed and where needed
On the fly Compression

- Apache provides for on-the-fly compression/decompression (c/d) of content via mod_deflate
- Compressed data is smaller, thus is quicker and easier on bandwidth
- Overhead is associated with the c/d on client and server side
- Usually, the bandwidth improvements outweigh that c/d overhead.
Be smart with

- See if more dedicated directive makes sense:
  - `RewriteRule /img/(.*) http://images.example.com/$1 [P]`
  - `ProxyPass /img/ http://images.example.com/`
- Avoid regexes at all if possible
- Smart ordering of RewriteRules
  - Assume 75% gif, 20% jpg, 5% png
Smart ordering

• No:
  - RewriteRule ^/img/(.*)\.png http://png.example.com/$1.png [P,L]
  - RewriteRule ^/img/(.*)\.jpg http://jpg.example.com/$1.jpg [P,L]
  - RewriteRule ^/img/(.*)\.gif http://gif.example.com/$1.gif [P,L]

• Yes:
  - RewriteRule ^/img/(.*)\.gif http://gif.example.com/$1.gif [P,L]
  - RewriteRule ^/img/(.*)\.jpg http://jpg.example.com/$1.jpg [P,L]
  - RewriteRule ^/img/(.*)\.png http://png.example.com/$1.png [P,L]
Monitor Apache

Apache Server Status for httpd.apache.org
Server Version: Apache/2.0.52 (Unix)
Server Built: Sep 23 2004 12:08:46

Current Time: Tuesday, 18-Jan-2005 06:21:28 PST
Restart Time: Friday, 17-Dec-2004 01:52:15 PST
Parent Server Generation: 34
Server uptime: 32 days 4 hours 29 minutes 13 seconds
Total accesses: 113530128 - Total Traffic: 3978.4 GB
CPU Usage: u79.6797 s102.75 cu764.695 cs0 - .0341% CPU load40.8
requests/sec - 1.5 MB/second - 36.7 kB/request
246 requests currently being processed, 41 idle workers
Monitor Server

Somewhat dependent on OS

- top
- vmstat
- lsof
- icps
- netstat
- fstat
Monitor Server

ps !! (good tune)

<table>
<thead>
<tr>
<th>PID</th>
<th>TT</th>
<th>STAT</th>
<th>TIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>184</td>
<td>??</td>
<td>Is</td>
<td>0:00.00</td>
<td>adjkerntz -i</td>
</tr>
<tr>
<td>272</td>
<td>??</td>
<td>Ss</td>
<td>30:08.99</td>
<td>/usr/sbin/syslogd -l /var/run/log -l /var/named/var/run/log -s</td>
</tr>
<tr>
<td>565</td>
<td>??</td>
<td>Is</td>
<td>1:20.42</td>
<td>/usr/sbin/inetd -wW -C 60</td>
</tr>
<tr>
<td>29178</td>
<td>??</td>
<td>Ss</td>
<td>0:12.29</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29183</td>
<td>??</td>
<td>S</td>
<td>0:23.72</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29184</td>
<td>??</td>
<td>S</td>
<td>0:24.47</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29185</td>
<td>??</td>
<td>S</td>
<td>0:32.26</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29186</td>
<td>??</td>
<td>S</td>
<td>0:23.96</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29187</td>
<td>??</td>
<td>S</td>
<td>0:28.99</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29188</td>
<td>??</td>
<td>S</td>
<td>0:21.64</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29189</td>
<td>??</td>
<td>S</td>
<td>0:22.99</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29190</td>
<td>??</td>
<td>S</td>
<td>0:28.98</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29191</td>
<td>??</td>
<td>S</td>
<td>0:28.64</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29192</td>
<td>??</td>
<td>S</td>
<td>0:26.06</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29193</td>
<td>??</td>
<td>S</td>
<td>0:22.17</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29194</td>
<td>??</td>
<td>S</td>
<td>0:23.03</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29195</td>
<td>??</td>
<td>S</td>
<td>0:25.43</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29212</td>
<td>??</td>
<td>S</td>
<td>0:23.89</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29213</td>
<td>??</td>
<td>S</td>
<td>0:23.83</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29214</td>
<td>??</td>
<td>S</td>
<td>0:23.82</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29215</td>
<td>??</td>
<td>S</td>
<td>0:24.32</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29246</td>
<td>??</td>
<td>S</td>
<td>0:24.52</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29260</td>
<td>??</td>
<td>S</td>
<td>0:26.66</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>29262</td>
<td>??</td>
<td>S</td>
<td>0:23.68</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>44206</td>
<td>p0</td>
<td>Ss</td>
<td>0:00.03</td>
<td>-zsh (zsh)</td>
</tr>
<tr>
<td>44213</td>
<td>p0</td>
<td>R+</td>
<td>0:00.00</td>
<td>ps ax</td>
</tr>
</tbody>
</table>
Monitor Server

ps !! (bad tune)

<table>
<thead>
<tr>
<th>PID</th>
<th>TT</th>
<th>STAT</th>
<th>TIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>184</td>
<td>??</td>
<td>Is</td>
<td>0:00.00</td>
<td>adjkerntz -i</td>
</tr>
<tr>
<td>272</td>
<td>??</td>
<td>Ss</td>
<td>30:08.99</td>
<td>/usr/sbin/syslogd -l /var/run/log -l /var/named/var/run/log -s</td>
</tr>
<tr>
<td>565</td>
<td>??</td>
<td>Is</td>
<td>1:20.42</td>
<td>/usr/sbin/inetd -wW -C 60</td>
</tr>
<tr>
<td>29178</td>
<td>??</td>
<td>Ss</td>
<td>0:12.29</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>32134</td>
<td>??</td>
<td>S</td>
<td>0:23.72</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>32137</td>
<td>??</td>
<td>S</td>
<td>0:24.47</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>32141</td>
<td>??</td>
<td>S</td>
<td>0:32.26</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>35467</td>
<td>??</td>
<td>S</td>
<td>0:23.96</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>36444</td>
<td>??</td>
<td>S</td>
<td>0:28.99</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>36786</td>
<td>??</td>
<td>S</td>
<td>0:21.64</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>36790</td>
<td>??</td>
<td>S</td>
<td>0:22.99</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>36795</td>
<td>??</td>
<td>S</td>
<td>0:28.98</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>36895</td>
<td>??</td>
<td>S</td>
<td>0:28.64</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>36899</td>
<td>??</td>
<td>S</td>
<td>0:26.06</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>38556</td>
<td>??</td>
<td>S</td>
<td>0:22.17</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>38622</td>
<td>??</td>
<td>S</td>
<td>0:23.03</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>38678</td>
<td>??</td>
<td>S</td>
<td>0:25.43</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>39005</td>
<td>??</td>
<td>S</td>
<td>0:23.89</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>39058</td>
<td>??</td>
<td>S</td>
<td>0:23.83</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>39275</td>
<td>??</td>
<td>S</td>
<td>0:23.82</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>39295</td>
<td>??</td>
<td>S</td>
<td>0:24.32</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>42587</td>
<td>??</td>
<td>S</td>
<td>0:24.52</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>42589</td>
<td>??</td>
<td>S</td>
<td>0:26.66</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>44198</td>
<td>??</td>
<td>S</td>
<td>0:23.68</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>44206</td>
<td>p0</td>
<td>Ss</td>
<td>0:00.03</td>
<td>zsh (zsh)</td>
</tr>
<tr>
<td>44209</td>
<td>??</td>
<td>S</td>
<td>0:00.38</td>
<td>./bin/httpd -d /staff/httpd</td>
</tr>
<tr>
<td>44213</td>
<td>p0</td>
<td>R+</td>
<td>0:00.00</td>
<td>ps ax</td>
</tr>
</tbody>
</table>
Thanks!

- Q&A
- Resources:
  - [dev@httpd.apache.org](mailto:dev@httpd.apache.org)
  - A certain Open Source support provider