What's new in Apache HTTP Server 2.2

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
• CTO of Covalent Technologies
• 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 2.2.6 (2.2.7 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 authorz
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
So what’s new in 2.2?
So what’s new in 2.2?

• Nothing
So what’s new in 2.2?

• Nothing
• Thanks!
So what’s new in 2.2?

- Nothing
- Thanks!
- Be sure to tip your waiters!
No, really...

- Large file support
- Graceful stop
- mod_dbd
- mod_filter
- 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
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
• The problem:
  – filters are basically inserted “unconditionally”
  – Blunt tool approach - bad w/ dynamic content
  – Admins want more flexibility

• The solution:
  – A dynamic chaining of filters
  – Filters inserted based on req headers, resp headers and env-vars.
Better Debugging

• mod_dumpio
  – Dumps all IO to the error log
  – Yep, all of it
    • DumpIOInput On
    • DumpIOOutput On
    • DumpIOLogLevel Notice
  – What about SSL?
    • Dumping is done right after decryption or right before encrypting
mod_dumpio

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
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
- 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 \\n    -1250M -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

- Actual workers
- Listener
- Storage of "Ready" sockets
- Sockets of "interest"
An illustration to help understand the concepts of Sockets of "interest", Listener, Storage of "Ready" sockets, and Actual workers.
An illustration to help understand the process of managing 'ready' sockets.

- **Listener**: Acts as a recipient for incoming connections.
- **Storage of 'Ready' sockets**: Holds sockets that are ready for use.
- **Actual workers**: Represent the processes or threads that execute tasks.

The diagram illustrates the flow of connections from listener to storage, eventually reaching the actual workers.
An illustration to help

- Actual workers
- Storage of "Ready" sockets
- Listener
- Sockets of "interest"
An illustration to help

Actual workers

Sockets of “interest”  Listener  Storage of “Ready” sockets
An illustration to help

- Sockets of "interest"
- Listener
- Storage of "Ready" sockets
- Keepalive connection
- Actual workers
An illustration to help understand the workflow of actual workers, sockets of interest, and the storage of ready sockets. The diagram shows a keepalive connection between the listener and the storage of ready sockets.
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>
Interested in more 2.2 auth?

- Attend Brad Nicholes’ session
- Friday, 4pm
Interested in more 2.2 auth?

- Attend Brad Nicholes’ session
- Friday, 4pm
Proxy

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

• Apache can now talk AJP with Tomcat directly
• Other proxy improvements make this even more exciting
• mod_jk alternative
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

```xml
<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>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><a href="http://php1:8080/">http://php1:8080/</a></td>
<td>1</td>
<td>0</td>
<td>Ok</td>
<td>311</td>
<td>125K</td>
<td>446K</td>
<td></td>
</tr>
<tr>
<td><a href="http://php2:8080/">http://php2:8080/</a></td>
<td>4</td>
<td>1</td>
<td>Ok</td>
<td>1232</td>
<td>433K</td>
<td>1743K</td>
<td></td>
</tr>
<tr>
<td><a href="http://phpbkup:8080/">http://phpbkup:8080/</a></td>
<td>4</td>
<td>0</td>
<td>Stby Ok</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Edit balancer settings for balancer://foo

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>StickySession Identifier</td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td>0</td>
</tr>
<tr>
<td>Failover Attempts</td>
<td>2</td>
</tr>
<tr>
<td>LB Method</td>
<td>bytraffic</td>
</tr>
</tbody>
</table>

Submit
Oh yeah

- ProxyPassMatch
  - ProxyPassMatch ^(/.*\.gif)$ \ http://backend.example.com$1
Want more 2.2 proxy info?

• Attend Jim Jagielski’s session
• Friday, 3pm
• I hear he’s pretty good...
What’s on the horizon?

• Some additional potential backports
  – mod_substitute
  – FastCGI proxy module

• True async server support
  – serf: http://code.google.com/p/serf/ ?

• Code name: Amsterdam
  – tell us!
Thanks!

• Q&A

• Resources:
  – http://httpd.apache.org/
  – dev@httpd.apache.org
  – A certain Open Source support provider