WASD: why is it chosen when there’s VMS Apache?
WASD: why is it chosen when there’s VMS Apache?
An objective assessment
An objective assessment

by the author of WASD
An objective assessment
by the author of WASD

😊
WASD: why does it continue to be chosen when there’s Apache?
Apache: why is it chosen when there’s WASD?
Why would any developer choose something not ‘industry standard’, with a smaller user base, fewer tools and seemingly more tenuous support?
Why would any developer choose something not ‘industry standard’, with a smaller user base, fewer tools and seemingly more tenuous support?

Same might be asked of OpenVMS!
Rationale

Purpose of the session is not (necessarily) to proselytize WASD but to explain how it might end up the preferred option for given projects and sites.
Thanks to Sponsors

Hewlett Packard
   John Gillings and OpenVMS Engineering for the ‘admit one’

VSM Software Services
   Jeremy Begg for not needing to swim here

Defence Science and Technology
   Paul Amey for ‘board and lodging’

My wonderful spouse
   Robyn for a long-ish leash and some pocket-money
Session Overview

Selection Considerations
Apache and WASD Features
Scripting Support
Performance
Case Studies
  Education
  Finance
  Telecommunication
Poll
  Differentiators
  Testimonials
Rumination
Questions
Selection Considerations
Selection Considerations

- **Purpose**
  - Document publication
  - Data connectivity
  - Web ‘applications’

- **Security**
  - Authentication sources
  - Access control
  - Privacy (e.g. SSL)

- **Content**
  - Static
  - Dynamic
    - Scripting
    - *Pages (e.g. JSP, PHP)

- **Scripting**
  - CGI
  - Perl, PHP, Python, etc.
Selection Considerations

- **Load**
  - Peak
  - Average

- **Platform**
  - Alpha, Itanium, VAX
  - OpenVMS V.n.n
  - Hobbyist, SOHO, Enterprise

- **Other**
  - Policy
  - ‘Industry standard’
  - Documentation
  - Community
  - Contractual
  - Skills base
  - Comfort zone
Features

Apache and WASD Toe-to-Toe
## Macro Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Apache</th>
<th>WASD</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP/1.1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Alpha/Itanium</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Secure Sockets</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 &amp; IPv6</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Persistent Scripting</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Access Control</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Request ‘Rewrite’</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Proxy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Logging</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Perl, PHP, Python, etc.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>License</td>
<td>GPL</td>
<td>GPL</td>
</tr>
</tbody>
</table>
## Platform Support

<table>
<thead>
<tr>
<th></th>
<th>Apache</th>
<th>WASD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Itanium</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VAX</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>V6.0</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>V6.1</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>V6.2</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>V7.1</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>V7.2</td>
<td>V1.3</td>
<td>Yes</td>
</tr>
<tr>
<td>V7.3</td>
<td>V2.1</td>
<td>Yes</td>
</tr>
<tr>
<td>V8.2</td>
<td>V2.1</td>
<td>Yes</td>
</tr>
<tr>
<td>F8.3</td>
<td>V2.1(?)</td>
<td>Yes</td>
</tr>
<tr>
<td>Install ODS-2</td>
<td>V1.3</td>
<td>Yes</td>
</tr>
<tr>
<td>Install ODS-5</td>
<td>Mandatory V2.1</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Concurrent Serving

<table>
<thead>
<tr>
<th></th>
<th>Apache</th>
<th>WASD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server</strong></td>
<td>Child Processes</td>
<td>Single Process</td>
</tr>
<tr>
<td><strong>Concurrency</strong></td>
<td>Per-Process*</td>
<td>VMS AST</td>
</tr>
<tr>
<td><strong>Multi-CPU</strong></td>
<td>Per-Process</td>
<td>Multiple Instances**</td>
</tr>
<tr>
<td><strong>Scripting</strong></td>
<td>Per-Process or Subprocess</td>
<td>Detached Process</td>
</tr>
</tbody>
</table>

* To support 100 concurrent requests Apache requires a minimum of 101 processes.

** Multiple, per-CPU processes, cooperating via mutex and the DLM.
## Request Processing

<table>
<thead>
<tr>
<th>Connection</th>
<th>Request</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total: 11778542</td>
<td>Total: 47548025</td>
<td>1.1: 34749563 (60%)</td>
</tr>
<tr>
<td>Current: 90</td>
<td>Current: 3</td>
<td>1.0: 8271075 (20%)</td>
</tr>
<tr>
<td>Peak: 602</td>
<td>Peak: 177</td>
<td>0.9: 1318 (0%)</td>
</tr>
<tr>
<td>Busy: 2</td>
<td>Busy: 0</td>
<td></td>
</tr>
<tr>
<td>Persistent /Current: 05</td>
<td>Throttle /Queued: 0</td>
<td>Method</td>
</tr>
<tr>
<td>/Peak: 598</td>
<td>/Processing: 0</td>
<td>CONNECT: 0</td>
</tr>
<tr>
<td>Accepted: 11778540</td>
<td>/Busy: 0%</td>
<td>DELETE: 0</td>
</tr>
<tr>
<td>Rejected: 0</td>
<td>Redirect /Local: 4826103</td>
<td>GET: 47065561</td>
</tr>
<tr>
<td>IPv4: 11778542 (100%)</td>
<td>/Remote: 103607</td>
<td>HEAD: 12365</td>
</tr>
<tr>
<td>IPv6: 0 (0%)</td>
<td>Persistent /Total: 33549564</td>
<td>OPTIONS: 219526</td>
</tr>
<tr>
<td>SSL: 825983 (7%)</td>
<td>/Max: 1024</td>
<td>POST: 260154</td>
</tr>
<tr>
<td>Total /Rx: 20,721,796,536</td>
<td>Pipeline /Total: 45471</td>
<td>PUT: 69</td>
</tr>
<tr>
<td>/Tx: 550,615,568,045</td>
<td>/Max: 79</td>
<td>TRACE: 0</td>
</tr>
<tr>
<td>Error /Rx: 74336</td>
<td>Not From Cache: 1199612</td>
<td>Extension: 0</td>
</tr>
<tr>
<td>/Tx: 493057</td>
<td>Forbidden: 55367</td>
<td></td>
</tr>
</tbody>
</table>

### Response
- Duration: 91340
- Min: 0.000976
- Max: 01:00:17
- Ave: 0.348806
HTTPd ccuma.sci.uma.es:80

Server Activity  (HTTPd:80)

Wednesday, 10-MAY-2006 16:26:14

Period:  Wednesday, 3-MAY-2006 06:00 to Wednesday, 10-MAY-2006 05:58 (168 hours)
Requests:  6,721,086 total; 2,750 max; 270 peak
Bytes:  87,989,859,886 total; 210,683,340 max

(Data available from Thursday, 11-MAY-2006 08:09)
HTTPd ccuma.sci.uma.es:80

Server Activity  (HTTPe:80)

Wednesday, 10-MAY-2006 16:26:51

Period:  Tuesday, 9-MAY-2006 06:00 to Wednesday, 10-MAY-2006 05:59 (24 hours)
Requests:  1,228,685 total;  2,439 max;  253 peak
Bytes:  15,314,059,754 total;  72,657,901 max

(Data available from Thursday, 13-APR-2006 08:09)
## Authentication

<table>
<thead>
<tr>
<th></th>
<th>Apache</th>
<th>WASD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package*</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSUAF Module</td>
<td>Module</td>
<td>Yes</td>
</tr>
<tr>
<td>PKI** Module</td>
<td>Module</td>
<td>Yes</td>
</tr>
<tr>
<td>Custom***</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* package-specific username/password

** Public Key Infrastructure (X.509, etc.)

*** User-written authentication support
# Scripting Support

<table>
<thead>
<tr>
<th></th>
<th>Apache</th>
<th>WASD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGI</td>
<td>Just*</td>
<td>Yes</td>
</tr>
<tr>
<td>Perl</td>
<td>Module</td>
<td>RTE**</td>
</tr>
<tr>
<td>PHP</td>
<td>Module</td>
<td>RTE</td>
</tr>
<tr>
<td>Python</td>
<td>Module</td>
<td>RTE</td>
</tr>
<tr>
<td>Tomcat</td>
<td>Module</td>
<td>Reverse Proxy</td>
</tr>
<tr>
<td>Persistence</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* Implied criticism of OpenVMS Apache performance
** RTE is a persistent Run-Time Environment
Persistent Scripting

So what is ‘persistence’ then?

The ability of the server to reuse resources (such as processes) over multiple requests

A scripting/interpretation engine retaining it’s initialized state over multiple requests
Persistent Scripting

Why is ‘persistence’ so important?

Process activation expenses
- Latency
- CPU cycles

Scripting engine initialization
- Latency
- CPU cycles
Persistent Scripting

CGI paradigm is very expensive; solutions:

Apache
  child processes
  loadable modules

WASD
  reusable detached processes
  CGI plus
  Run Time Environment (RTE)
WASD CGI plus

- plus lower latency
- plus greater throughput
- plus far less system impact

CGIplus eliminates the overhead associated with creating the script process and then executing the image of a CGI script. It does this by allowing the script process and optionally any associated image/application to remain instantiated between uses, eliminating process and/or application startup overheads.

The script interface is still CGI, with all the usual environment variables and input/output streams available, which means a new API does not need to be learned and existing CGI scripts are simple to modify.

RTE (implemented using CGIplus) is intended as an environment in which a script source is interpreted or otherwise processed by the application. That is, for scripting engines, although it is not limited to that. Perl, PHP and Python engines for WASD are implemented using RTE. Start once - execute many.
The test system was a lightly-loaded AlphaServer 4100 4/400 (4 x 400MHz CPUs), OpenVMS V7.3-2 and DEC TCP/IP 5.4. No keep-alive functionality was employed so each request required a complete TCP/IP connection and disposal. DNS (name resolution) and access logging were disabled. The server and test-bench utility (ApacheBench v1.3) were located on separate systems with 100 Mbps Fast-Ethernet interconnection.

On clustered, multi-user systems too many things vary slightly all the time. Hence the batching of accesses, interleaved between servers, attempting to provide a representative result.

CSWS 1.3 (based on Apache 1.3.26)
WASD 9.0

Performance - 1 concurrent

Performance - 10 concurrent

Case Studies

OpenVMS+WASD success stories
Case Study - Education

Universidad de Málaga - Spain

4 campuses; 19 faculties; 65 undergraduate courses; 3760 staff; 40,000 students

A significant user of OpenVMS for email, Web, database and administration

Planning migration from OSU to Apache in late 2002 evaluation revealed show-stopping issue with Apache

“A second threat for [SSL certificate] key disclosure exists during script execution because scripts run in the context of the server and have complete access to key files no matter where they exist (as long as they exist in a directory accessible to APACHE$WWW). Therefore, it is not advisable to allow the execution of arbitrary user scripts when using SSL.” OpenVMS Apache Release Notes
Case Study - Education

Universidad de Málaga - Spain

Evaluated WASD in early 2003 and put it into production shortly after!
76 virtual servers
>1M requests and >15GB per weekday
>600 concurrent connections and >100 requests in-progress routinely supported (using 2-30 processes)
X.509 based PKI authentication access control
Extensive deployment using PHP, along with existing OSU scripts, and more recent CGI based applications
Case Study - Education

Universidad de Málaga - Spain

“WASD has allowed us to build a very robust, and above all, secure, web infrastructure, without having to give up twenty years of VMS knowledge. For us, the strongest points of WASD are excellent performance, excellent VMS security model integration and unbeatable support.”

site:  http://www.uma.es/
Case Study - Education

ESME-Sudria – France

Ecole d'Ingénieurs Généralistes

(College of Engineering)

- Automation
- Electronics
- Telecommunications
- Computer and Software Engineering
Case Study - Education

ESME-Sudria – France

Internet
Intranet
Standard proxy
Reverse proxy
Gatewaying
Les services Web à l’ESME-Sudria
Serveurs Intranet

Site interne
intranet.esme.fr
WASD

Proxy
WASD
Proxy standard d’accès à Internet

Cluster VMS/WASD

Proxy
WASD
Cache
HTTP
Reverse proxy HTTP

Site public
www.esme.fr
WASD

Reverse
Proxy
WASD
Reverse proxy HTTPS

Reverse Proxy
WASD

Site interne
intranet.esme.fr
WASD
DNS wildcard proxy
Case Study - Education

ESME-Sudria - France

“WASD has enabled us to webify more and more applications and develop brand new ones with excellent performances. CGI+ has provided us with applications that responds in a tenth of a second ... A lot of features are in use at ESME-Sudria: web servers, proxy, reverse proxy, DNS wildcards proxy... Even some IIS server are protected by authorization through a WASD reverse proxy, giving to VMS the ability to allow single sign-on to different platforms.”

site:  http://www.esme.fr/
Case Study - Finance

Coast Capital Savings - Canada

Coast Capital Savings is a credit union servicing 300,000 customers in the Lower Mainland and southern Vancouver Island regions of British Columbia, Canada. Coast Capital Savings banking system runs on OpenVMS AlphaServers and is written in Greystone Technology M (M).

WASD is principally used as an application server (middleware) for integrating traditional ‘green-screen’ financial database application with Windows-based (.NET) applications.

The XML-SOAP-RPC mechanism implemented for this serves approximately 1500 interactive workstations, as well as a busy customer-facing IVR system.
Case Study - Finance

Coast Capital Savings - Canada

- Ease of integration
  CGI or CGI plus programming in DCL, Python, MUMPS, C, etc.
- VMS security mechanisms
  persona scripting and particular account contexts
- Performance
  persistent CGI plus provides a low-latency (few milliseconds)
  high throughput transaction infrastructure
- Application management
  load-balancing, throttling, script process rundown allowing
  ‘gentle’ application/server shutdown and/or system
  migration
Case Study - Finance

Coast Capital Savings - Canada

“WASD came bundled with a friendly gentleman in Australia who appears to be online 24x7 … also appears to read all the latest specs, to do tons of testing, and keep pushing WASD forward … makes WASD worth it’s weight in platinum.”

more information:  http://h71000.www7.hp.com/openvms/journal/v7/wasd.html
Case Study - Telecommunications

EDS Telco Solutions Group - Australia

Developed by EDS on behalf of an Australian telecommunications carrier providing landline, cell phone and Internet services. Due to Commercial-in-Confidence considerations, the customer cannot be identified.

Service Profile data includes billing, product information, discounts, promotions, and mobile features information.

To permit existing corporate systems and middleware to exchange Service Profile information, web services technologies based on XML, SOAP 1.1, and HTTP were employed. These enable the exchange of XML encapsulated information to and from retailers, OpenVMS applications and the GSM network hardware.
Case Study - Telecommunications

EDS Telco Solutions Group - Australia

- Available for VAX platform
  some remaining systems required consideration
- CGI plus persistent scripting
  eliminate per-request process creation on busy systems
  allow database context(s) to remain instantiated
- Script process termination
  WASD issues $FORCEX before shutting-down idle scripts
  allows exit handlers to elegantly release database context(s)
- Monitoring and troubleshooting
  server statistics, WATCH facility, WOTSUP utility
Case Study - Telecommunications

EDS Telco Solutions Group - Australia

“Truth be known, I put my choice behind Apache initially due to the number of developers out there ... then I found out that WASD was developed specifically for VMS ... an OpenVMS solution. I’m glad my decision on choosing Apache was not adhered to because WASD has proved a very good choice indeed. WASD ... is cluster-aware ... synergic with the OpenVMS OS’s philosophy and design. WASD developer(s) and community are helpful and very responsive.

Because it simply kicks-arse!”

more information:  http://h71000.www7.hp.com/openvms/journal/v7/wasd.html
info-WASD Poll

These are lists distilled from respondent comments to the mailing list poll where some particular WASD attribute was of particular significance in the package preference.
Poll - Differentiators

VMS Integration
- AST event driven model
- OPCOM
- SYSUAF
- ACME
- DLM
- Mailboxes
- Cluster ‘awareness’
Poll - Differentiators

Performance
- AST event driven
- Single process model
- Conservative resource consumption
- Scripting
Poll - Differentiators

Monitoring and Administration

- WATCH
- Server configuration (loaded)
- WATCH
- Server statistics
- WATCH
- $HTTPD/DO=<something>[/CLUSTER]
- WATCH
Poll - Differentiators

WATCH …

provides an online, real-time, in-browser-window view of request processing in the running server. The ability to observe live request processing on an ad hoc basis, without changing server configuration or shutting-down/restarting the server process, makes this facility a great configuration, problem resolution and application development tool.
HTTPd slim.vsm.com.au:80

WATCH Report

Saturday, 29–APR–2006 07:09:27

<table>
<thead>
<tr>
<th>Request</th>
<th>General</th>
<th>Network</th>
<th>Proxy</th>
<th>Other</th>
<th>Proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ Processing</td>
<td>□ Connection</td>
<td>□ Activity</td>
<td>□ Processing</td>
<td>□ Request Header</td>
<td></td>
</tr>
<tr>
<td>□ Header</td>
<td>□ Mapping</td>
<td>□ Data</td>
<td>□ Request Header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>□ Body</td>
<td>□ Authorization</td>
<td></td>
<td></td>
<td>□ Response Header</td>
<td></td>
</tr>
<tr>
<td></td>
<td>□ Error</td>
<td></td>
<td></td>
<td>□ Response Body</td>
<td></td>
</tr>
<tr>
<td>Response</td>
<td>□ CGI</td>
<td></td>
<td></td>
<td>□ Cache</td>
<td></td>
</tr>
<tr>
<td>□ Processing</td>
<td>□ DCL</td>
<td></td>
<td></td>
<td>□ Cache Maintenance</td>
<td></td>
</tr>
<tr>
<td>□ Header</td>
<td>□ DECnet</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>□ Body</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* * *

Client Filter

Service Filter

Path/Track Filter

60 or | Seconds Duration

Include (only) in Server Process Log

WATCH  reset
Poll - Differentiators

Scripting

- Performance
- IPC based on mailboxes
- CLI activation and DCL symbols
- CGI plus / RTE
- Persona
- CGI response header directives
- OSU emulation
Poll - Differentiators

Proxy
- Standard proxy
- Reverse proxy
- Tunneling
- Disk cache model and implementation
Poll - Testimonials

“I have stayed with WASD because I like the product … Apache would have to be very much better than WASD … and I don’t see that happening, ever.”

“WASD just ran from the FREEWARE CD copy … Besides it was also a fast server which hardly needs significant attention. It is well designed (IMHO) and sports a huge number of interfaces …”

“Back in the beginning of the century [😊] we were faced to the necessity of making a lot applications available through the web. Most were based on the VMS security model. The ability to run scripts under the user’s persona …”

“WASD has been performing very well in our demanding conditions. It is even resilient enough to keep serving pages even when there was a hideous bug [😭] that killed some server processes, thus keeping us up. We couldn't be happier with the software and with the excellent mood of its author.”
“Although I have not used HP support for Apache, I have found that HP support for other HP-ported products (Kerberos, SSH, COM, etc.) to be a little difficult to obtain (‘Uh, do we support that product?’) … rather than ‘Here is how you do it, sorry my docs weren’t clear’ or ‘I will build that into the next release’…”

“I found the experience to be easy, and the support (documentation and mailing list) to be far superior to any other of the webservers.”

“WASD comes into the scene. It had the needed reverse proxy feature that made possible a connection to a Tomcat server (or to any other protected server) and had an excellent privilege separation model. We ported our configuration and, in record time, we had a test system running, that let us move quickly into production.”
Poll - Testimonials

“[VAX 6000 running VMS 6.0] ... we have been the ‘low-end hardware/software’ beta test site ... not that you would notice. WASD’s betas are arguably more stable than other people’s production releases.”

“... there is the best in class support that comes from the southern hemisphere. The kind of support you dream of ... you find a problem, send a mail, go to bed and, when you wake up in the morning, there is a cute [😊] answer!”

“Can’t improve on all the responses, but in terse terms ... VMS integration, security, performance, extensibility, reliability, support. Use Apache? ... Haven’t used it under VMS, but have under Tru64 and it is cumbersome by comparison.”
Rumination

WASD: why is it chosen when there’s OpenVMS Apache?
Rumination

WASD does everything Apache does
WASD CGI + CGIplus/RTE persistence
WASD performance
WASD conservative resource usage
WASD/OpenVMS integration
WASD tools – e.g. WATCH
WASD reliability
WASD support
Rumination

WASD: why ... ?

“Because it simply kicks-arse!”
Rumination

WASD: why … ?

“Because it simply kicks-ass!”
Demonstration?

Find me during the Bootcamp.
+61 407 883422

We’ll sit down at an Internet kiosk and spend some time at WASD sites large and small.

Want to know more about WATCH? Ditto!

Where to get it?
Questions?