Selecting a Web Environment
For OpenVMS

Mark Daniel

Employer: Defence Science and Technology Organisation
SOLID AS A ROCK

An Impartial Summary :-) 
by the author of the WASD Web Package
Opinions my own and on loan from others.

DECUS Australia Symposium 2002
Novotel Hotel Olympic Park NSW Australia - Home of the 2000 Olympics 21 – 23 July 2002
Objectives

1. Emphasize the availability and suitability of OpenVMS as a Web platform.
2. Suggest considerations for evaluating Web packages.
3. Examine current VMS options and their comparative characteristics.
4. Briefly consider strategies for reducing the impact of differences between them.
5. Not necessarily make recommendations!
Assumptions

Some experience with or exposure to …

OpenVMS
Web technologies

For example; no time will be taken to explain the differences between ASTs and POSIX Threads, latency and throughput, Perl and PHP.
Mud Map

Introduction
Why VMS!
Considerations
Package Evaluation
Development Environment
Summary
Questions
Disclaimer

As the author of the WASD package it is probably impossible for me to be completely dispassionate discussing this topic.

To the best of my knowledge the information in this session is accurate (or near-enough so). Any errors or omissions are not there deliberately to position one package to the detriment of another.
Why ‘The Web’?

Marketing
E-Commerce
Publication
Entertainment
‘Middleware’
Ubiquity
Why VMS?

Well, Why Not!
Why VMS!

● You already have it!
  ■ Then why not use it as a Web front/back-end?

● Currently evaluating?
  ■ VMS can provide COTS Web solutions

● Bet the business …
  ■ Many significant sectors do!

● Can’t afford to be down?
  ■ VMS clusters and disaster-tolerant solutions
Why VMS!

- **Security**
  - VMS has proved itself highly resistant to penetration
  - O/S architecture is designed to provide fine control of activities, even down to what memory can do what!

- **Contagion Proof**
  - It’s xenolithic nature and robust immune system mean pandemics sweep past OpenVMS not through it!
Can VMS really do what I need on the Web?

as always …

that depends!

and hopefully that’s what this session will go some way to answering
## Selected Web and VMS-Web History

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1989</td>
<td>Tim Berners-Lee &amp; CERN</td>
</tr>
<tr>
<td>1991</td>
<td>CERN at SLAC</td>
</tr>
<tr>
<td>1994</td>
<td>OSU from OSU</td>
</tr>
<tr>
<td>1995</td>
<td>Apache <em>(nee NCSA)</em> Purveyor and Cheetah</td>
</tr>
<tr>
<td>1996</td>
<td>WASD (then named HFRD)</td>
</tr>
<tr>
<td>1997</td>
<td>Netscape FastTrack</td>
</tr>
<tr>
<td>2000</td>
<td>CSWS (OpenVMS Apache)</td>
</tr>
<tr>
<td>2001</td>
<td>Tomcat</td>
</tr>
</tbody>
</table>
Servers Available and ‘Working’

CERN  
long-since obsolete
Cheetah  
retired
CSWS (Apache)  
active development
FastTrack  
retired
GFR HTTPd  
available
OSU  
active development
Purveyor  
retired
WASD  
active development
Xitami  
available
Package Selection Considerations

1. Purpose
2. Hardware
3. VMS Version
4. Dynamic Content
5. Security
6. Load
7. Support
8. Miscellaneous
Selection Considerations

Note that some of these will be technical issues, others non-technical.

This session may help with the technical ones but the non-technical are out-of-scope, except to point out they may be considerations.
Selection Considerations - Purpose

- **Publication**
  - Static pages
  - Dynamic pages

- **Data(base) connectivity**
  - Front-end
  - Middle-ware
  - Back-end

- **Ad Hoc serving**
Selection Considerations - H/Ware & VMS Version

- Alpha
- VAX
- (IA64)
- Hobbyist
- Enterprise
- Memory
- CPUs
- Dedicated

- Alpha
- VAX
- Pre-6.0
- Pre-7.1
- Pre-7.2
- Pre-7.3
Selection Considerations - Dynamic Content

- **Scripting** (in autonomous processes)
  - Ease
  - Languages / Environments
  - Standards

- **Pages** (interpretation of markup language)
  - SSI
  - PHP
  - JSP
Selection Considerations - Security

- **Authentication**
  - Credential sources

- **Access Control**
  - Who can do what to which and when

- **Transaction Privacy**
  - Secure Sockets Layer
Selection Considerations - Load

- **Concurrent Requests**
  - Peak
  - Average

- **Response Content**
  - Static
  - Dynamic
  - Script
  - Database
Selection Considerations - Support

- **Documentation**
  - Package
  - Third-party

- **User Community**
  - News Groups
  - Mailing Lists

- **Installation / Update**
  - Availability
  - Ease
Selection Considerations - Miscellanea

- **Policy**
  - ‘Standard’ or mandated product?
  - Contractual support

- **Skills Base**
  - Site administration
  - Developer requirements

- **Comfort Zone**
  - Package satisfaction
  - Longevity
<table>
<thead>
<tr>
<th>Package</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERN Cheetah</td>
<td>long-since obsolete retired</td>
</tr>
<tr>
<td>CSWS (Apache)</td>
<td>active development retired</td>
</tr>
<tr>
<td>FastTrack</td>
<td>available</td>
</tr>
<tr>
<td>GFR HTTPd</td>
<td>active development retired</td>
</tr>
<tr>
<td>OSU Purveyor</td>
<td>available</td>
</tr>
<tr>
<td>WASD Xitami</td>
<td>active development available</td>
</tr>
</tbody>
</table>
Package Evaluation - CSWS

- Compaq Secure Web Server
- Apache* for OpenVMS
  * Currently the most popular webserver on this small, blue-green planet.
- Ported & supported by VMS Engineering
- Available since 2000
  - currently in it’s third generation (v1.2)
- Concurrency using multiple ‘child-processes’
- Alpha only
- VMS 7.2-1 or later only
Package Evaluation - OSU

- aka. DECthreads Server
- Dave Jones of Ohio State University
  - established and active user community
- Available since 1994
- Single process
- Concurrency using POSIX Threads
  - allows VMS kernel threading
- Alpha or VAX
- VMS 5.5-2 through to 7.3  (min v3.6b for V7.2 or later)
Package Evaluation - WASD

- Named after a now-extinct race of engineers
- Mark Daniel* of DSTO
  * That’s me, just in case you missed the start of the session!
- established and active user community
- Available since 1996 (under development since 1994)
- Single process
- Concurrency using VMS ASTs
  - employs other core VMS technologies
- Alpha or VAX
- VMS 5.5-2 (pre-v7.0) through to 7.3
Package Evaluation - Purpose

- All three packages provide static and dynamic Web page generation

- All packages will interface (at least via scripting) to DBMS such as RDB and Oracle

- All have a long development history providing a broad spectrum of services to their users
## Package Evaluation - Hardware & VMS Version

<table>
<thead>
<tr>
<th>Package</th>
<th>Alpha</th>
<th>VAX</th>
<th>5.5-2</th>
<th>6.0</th>
<th>6.1</th>
<th>6.2</th>
<th>7.1</th>
<th>7.2</th>
<th>7.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OSU</td>
<td>X</td>
<td>X</td>
<td>X*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X**</td>
<td>X**</td>
</tr>
<tr>
<td>WASD</td>
<td>X</td>
<td>X</td>
<td>X***</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

* pre-v3.6  
** post-v3.6  
*** pre-v7.0
## Package Evaluation - Scripting

<table>
<thead>
<tr>
<th></th>
<th>CGI</th>
<th>Perl</th>
<th>PHP</th>
<th>Tomcat</th>
<th>Python</th>
<th>Persistence</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWS</td>
<td>Yes (wrapper*)</td>
<td>Module** (persistent)</td>
<td>Module (persistent)</td>
<td>Module (persistent)</td>
<td>Yes (wrapper*)</td>
<td>Yes</td>
</tr>
<tr>
<td>OSU</td>
<td>Yes (wrapper*)</td>
<td>Yes (wrapper* &amp; persistent)</td>
<td>Yes (wrapper* &amp; persistent)</td>
<td>No</td>
<td>Yes (wrapper*)</td>
<td>Yes</td>
</tr>
<tr>
<td>WASD</td>
<td>Yes (native)</td>
<td>Yes (persistent)</td>
<td>Yes (persistent)</td>
<td>Coming</td>
<td>Yes (persistent)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*A wrapper is a DCL procedure that provides the required support environment.**

**Apache provides some core and all additional functionality using discrete code ‘modules’.
Digression - Scripting

Why is ‘persistence’ so important?

- **Process activation expenses**
  - Latency
  - CPU cycles

- **Scripting engine initialization**
  - Latency
  - CPU cycles
Digression - Scripting

So what is ‘persistence’ then?

- The ability of the server to reuse resources (such as processes) over multiple requests
- A scripting/processing engine retaining it’s initialized state over multiple requests
Digression - Scripting

Interestingly, all three packages have had to tackle this issue (each in its own inimitable fashion)

- **CSWS**
  - child-processes and loadable modules
- **OSU**
  - reusable DECnet processes and HPSS
- **WASD**
  - reusable detached processes and CGIplus
## Package Evaluation - Authentication

<table>
<thead>
<tr>
<th></th>
<th>Local*</th>
<th>SYSUAF</th>
<th>PKI**</th>
<th>Other***</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWS</td>
<td>Yes</td>
<td>Module</td>
<td>Module</td>
<td>Module</td>
</tr>
<tr>
<td>OSU</td>
<td>Yes</td>
<td>Yes</td>
<td>Perhaps****</td>
<td>Yes</td>
</tr>
<tr>
<td>WASD</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* Package-unique username/passwords.
** Public Key Infrastructure (X.509, etc.)
*** User-written authentication support.
**** Requires building some experimental modules.
## Package Evaluation - Access Control

<table>
<thead>
<tr>
<th></th>
<th>Virtual Server</th>
<th>Path (allow/deny)</th>
<th>Method (read/write)</th>
<th>Discretionary*</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Require/Satisfy</td>
</tr>
<tr>
<td>OSU</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>WASD</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>‘Conditionals’</td>
</tr>
</tbody>
</table>

* ‘Discretionary’ in the sense access is based on some evaluation of characteristics of the request other than the path or authenticated user.”
## Package Evaluation - Secure Sockets Layer

<table>
<thead>
<tr>
<th></th>
<th>Which</th>
<th>How</th>
<th>PKI*</th>
<th>Support**</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWS</td>
<td>OpenSSL</td>
<td>Module</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OSU</td>
<td>OpenSSL</td>
<td>DIY***</td>
<td>Perhaps****</td>
<td>Yes</td>
</tr>
<tr>
<td>WASD</td>
<td>OpenSSL</td>
<td>Optional****</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Public Key Infrastructure (X.509, etc.)*  
**Support for self-signed certificates, OpenSSL swiss-army-knife, etc.*  
***Not part of the official package; requires obtaining and building the full OpenSSL kit.*  
****May be built either as an SSL or non-SSL executable.****  
*****Requires building experimental modules.
# Package Evaluation - CPU Load

<table>
<thead>
<tr>
<th></th>
<th>Server</th>
<th>Concurrency</th>
<th>Multi-CPU</th>
<th>Scripting</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWS</td>
<td>Multiple Processes</td>
<td>Per-Process**</td>
<td>Per-Process</td>
<td>Child Process</td>
</tr>
<tr>
<td>OSU</td>
<td>Single Process*</td>
<td>POSIX Threads*</td>
<td>VMS Kernel Threading***</td>
<td>Reused DECnet Process</td>
</tr>
<tr>
<td>WASD</td>
<td>Single Process*</td>
<td>VMS ASTs*</td>
<td>Multiple Instances****</td>
<td>Reused Detached Process</td>
</tr>
</tbody>
</table>

* It is broadly acknowledged that, all other things being equal, a single process with internal concurrency is significantly more efficient than multi-process concurrency.

** To support 150 concurrent requests CSWS requires a minimum of 150 processes.

*** On VMS 7.1 or later.

**** WASD v8.0 supports cooperating, multi-process, per-CPU serving.
## Package Evaluation - Support

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CSWS</td>
<td>Copious</td>
<td>Lots &amp; lots &amp; lots!</td>
<td>comp.os.vms</td>
<td>Multiple</td>
<td>Yes</td>
<td>Internet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>comp. infosystems...</td>
<td></td>
<td></td>
<td>PCSI CONDIST</td>
</tr>
<tr>
<td>OSU</td>
<td>Meager</td>
<td>comp.os.vms</td>
<td>Single</td>
<td>No</td>
<td></td>
<td>Internet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ZIP</td>
</tr>
<tr>
<td>WASD</td>
<td>Copious</td>
<td>comp.os.vms</td>
<td>Single</td>
<td>No</td>
<td></td>
<td>Internet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ZIP</td>
</tr>
</tbody>
</table>
## Product Evaluation - Installation

<table>
<thead>
<tr>
<th></th>
<th>Distribution</th>
<th>Kit</th>
<th>Installation</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CSWS</strong></td>
<td>Internet CONDIST</td>
<td>PCSI</td>
<td>PRODUCT INSTALL*</td>
<td>DCL Procedure +plus+ Manual</td>
</tr>
<tr>
<td><strong>OSU</strong></td>
<td>Internet</td>
<td>TAR archive</td>
<td>Compile+Link**</td>
<td>Manual</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ZIP archive</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>WASD</strong></td>
<td>Internet Freeware CD</td>
<td>ZIP archive</td>
<td>Link-only*** -or- Compile+Link</td>
<td>DCL Procedure +plus+ Manual</td>
</tr>
</tbody>
</table>

* CSWS is distributed with pre-built executables.
** OSU is distributed only as source files and requires DECC to build it.
*** WASD is distributed as source and optional pre-compiled object modules. This allows a site to fully build the package using DECC or link-only the available object modules.
Development Environment

What can be done to smooth-out differences between packages, maximize content sharing and portability, while minimizing migration issues when the inevitable looms?
Development Environment

- Of course package administration specifics cannot be avoided when using any server
- Site content is a different matter …
  - All packages serve static pages relatively uniformly
  - Dynamic pages are a different issue
- Choose cross-platform, ‘standard’ scripting and/or dynamic content tools
  - This should allow platform migration, and content-sharing, relatively seamlessly
Development Environment

- **Common Gateway Interface (CGI)**
  - Lowest common denominator scripting
  - Supported by all servers
  - Of course the underlying O/S may present migration difficulties … SO

- **CGI using** *(all ‘interpreted’, cross-platform scripting languages)*
  - Perl
  - PHP
  - Python
Development Environment

- Persistent Scripting
  - There is no generic, cross-platform, persistent scripting environment
  - Instead use a persistent implementation of
    - Perl
    - PHP
    - Python
    - JSP (Java Server Pages)
  - Endeavour to deploy a persistent DBMS agent
Development Environment

- **Dynamic Pages**
  - **SSI** (Server Side Includes)
    - Notoriously platform-specific
    - If you must use them, use a ‘standard’ subset
  - Perl, PHP and Python *with page templates*
    - Cross-platform
deploy or re-deploy across MS Windows, Unix, VMS
  - **JSP** (Java Server Pages, aka. Tomcat)
    - Cross-platform
deploy or re-deploy across MS Windows, Unix, VMS
Development Environment

Other cross-platform advantage

- The original and obvious example is HTML
- Others are
  - XML
  - SOAP
  - LDAP
  - Java
Development Environment

Although this is a *little* out of scope …

- Avoid IDEs that insist on server ‘extensions’
  - ‘Dreamweaver’ works well with generic environments
  - ‘FrontPage’ tends to assume IIS and an MS platform

- When client-side processing is required
  - ‘JavaScript’ is more universal than say
  - ‘ActiveX’ which needs a MS platform
Summary

It seems obvious that CSWS, OSU and WASD are all mature and quite capable packages.

No specific package recommendation is made.

Each may have specific strengths or advantages in given environments or for given tasks.
Summary - Opinion

Notwithstanding these conciliatory statements it seems the informed choice by the cognoscenti would be WASD^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d^d.

(Speak to me after the session. I’ll tell you what I really think!)
## Summary - Longevity

<table>
<thead>
<tr>
<th>System</th>
<th>Status</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERN</td>
<td>come+gone</td>
<td></td>
</tr>
<tr>
<td>Cheetah</td>
<td>come+gone</td>
<td></td>
</tr>
<tr>
<td>Apache</td>
<td>7 years (2 as CSWS on VMS)</td>
<td></td>
</tr>
<tr>
<td>FastTrack</td>
<td>come+gone</td>
<td></td>
</tr>
<tr>
<td>OSU</td>
<td>8 years</td>
<td></td>
</tr>
<tr>
<td>Purveyor</td>
<td>come+going</td>
<td></td>
</tr>
<tr>
<td>WASD</td>
<td>8 years</td>
<td></td>
</tr>
</tbody>
</table>
Recommended Reading
Author: Alan Winston
Manager Central Computing
Stanford University
Synchrotron Radiation Lab

Publisher: Digital Press

ISBN: 1555582648

(See me after the session for a flier)
References

OpenVMS
http://www.openvms.compaq.com/ebusiness/Technology.html

CSWS

OSU
http://www.er6.eng.ohio-state.edu/doc/serverinfo.html

WASD
QUESTION TIME

Thanks for attending!

DECUS Australia Symposium 2002
Novotel Hotel Olympic Park NSW Australia - Home of the 2000 Olympics 21 – 23 July 2002