VMS is VMS is VMS is VMS

-VAX Alpha IA64 +x86-64

mark.daniel@wasd.vsm.com.au

VMSgenerations April 2022

Good morning VMS colleagues.
First some thanks;
for the opportunity to chat about porting to X86 VMS.
to Jeremy Begg for providing the cross-compiler environment and currently on-line X86 WASD site.
and finally to VSI for the opportunity to participate in such an interesting and pivotal field-test.
As you can see from the title slide - with some caveats - VMS is VMS is VMS is VMS.
And the WASD porting experience underscores this for me.

Moving User Mode Application(s) To x86-64

(spoiler alert: c'est du gâteau)

This session concerns USER mode code.
Undoubtedly elevated modes will be less straight-forward.
And I seem to have given it aways already - the port was very straight-forward.
It really was a piece of cake.

VMS is VMS …
agenda

application environment
x86-64 porting (EAK V9.0)
    after one hour
    after eight hours
    x86-64 code specifics
x86-64 porting (EAK V9.1)
    development platforms
    cross compiling
    x86-64 performance
questions?

There are three main sections to this session.

1) A brief description of the application code environment.
2) A summary of the initial port (V9.0), eighteen months ago.
3) Some comments on the subsequent port (V9.1), six months ago.

Application Environment

WASD HTTP services package

- First developed early ’90s on VAX - 32 bit
- Initially VAX C, now DEC / VSI C
- Ported to Alpha (’95) and Itanium (’04)
- 32 bit with necessary 64 bit data as adjacent longwords
- Uses SYS$ services extensively
- Uses LIB$ routines (less) extensively
- All USER MODE code

Application Environment

WASD HTTP services package

- server code / comment counts

```
-------------------------------- OVERALL ---------------------------------
     ------Total------ -----------Code------------ ----------Comment----------
       Lines     Chars   Lines           Chars        Lines           Chars
     ------- --------- ------- ---- --------- ---- ------- ---- --------- ----
0172  237768   7861504  163085  77%   4388885  72%   21976  10%   1650720  27%
```

code: 163,000 lines 77% of total

comment: 21,976 lines 10% of total

x86-64 Porting

EAK 9.0 (September 2020)

- BXNUC10i7FNH4 6 core i7 1.10GHz 32GB
  (thanks to Jeremy Begg of VSM Software Services)

Generously resourced Intel NUC compact form-factor computer.
Linux O/S.
Virtualbox.
VMS V9.0 supplied as a (largely) pre-configured virtual HDD containing an ODS-5 file system.

x86-64 Porting

EAK 9.0 (September 2020)

- BXNUC10i7FNH4 6 core i7 1.10GHz 32GB
  (thanks to Jeremy Begg of VSM Software Services)

- After approximately one hour

```
$ mcr wasd_root:[x86_64]httpd /version
%HTTPD-I-SOFTWAREID, HTTPd-WASD/11.5.1 OpenVMS/X86
WASD VMS Web Services, Copyright (C) 1996-2020 Mark G.Daniel.
8< snip 8<
-HTTPD-I-SYSTEM, VBOX VBOXFACP 1 CPU 15361MB VMS V9.0-D
-HTTPD-I-TCPIP, Not (yet) for x86-64!
-HTTPD-I-TLS, none
```

  all the “core” code was built and running

Some preliminary hacking of source code placed __x86_64 conditionals in parallel with existing __VAX, __ALPHA __ia64.

It was then just a matter of running the server build procedures and correcting the typos.

x86-64 Porting

EAK 9.0 (September 2020)

- BXNUC10i7FNH4 6 core i7 1.10GHz 32GB
  (thanks to Jeremy Begg of VSM Software Services)
- After approximately one hour

```
$ mcr wasd_root:[x86_64]httpd /version
%HTTPD-I-SOFTWAREID, HTTPd-WASD/11.5.1 OpenVMS/X86
WASD VMS Web Services, Copyright (C) 1996-2020 Mark G.Daniel.
8< snip 8<
-HTTPD-I-SYSTEM, VBOX VBOXFACP 1 CPU 15361MB VMS V9.0-D
-HTTPD-I-TCPIP, Not (yet) for x86-64!
-HTTPD-I-TLS, none
```

  all the “core” code was built and running

- architecture-specific had hard-wired output (e.g. “Not (yet)…”)

  https://wasd.vsm.com.au/info-WASD/2020/0099

x86-64 Porting
EAK 9.0 (September 2020)

- Approximately eight hours (excluding chasing bugs in the EAK) filled in the majority of the architecture specifics

```
%HTTPD-I-SOFTWAREID, HTTPd-WASD/11.5.1 OpenVMS/X86 SSL
WASD VMS Web Services, Copyright (C) 1996-2020 Mark G.Daniel.
8< snip 8<
%HTTPD-I-SYSTEM, VBOX VBOXFACP VMS V9.0-D
8< snip 8<
%HTTPD-I-SSL, OpenSSL 1.1.1g  21 Apr 2020 (0x1010107F)
-SSL-I-PROTOCOL, TLSv1,TLSv1.1,TLSv1.2,TLSv1.3
-SSL-I-OPTIONS, 0x80410854
-SSL-I-SNI, Server Name Indication enabled
-SSL-W-DH, no ephemeral DH param
%HTTPD-I-HTTP2, enabled
8< snip 8<
%HTTPD-I-SERVICE, http://x86v1.vsm.com.au:7080
%HTTPD-I-SERVICE, https://x86v1.vsm.com.au:7443
%HTTPD-I-SSL, x86v1.vsm.com.au:7443
Generate x86v1.vsm.com.au 2048 bit private key:
.............++++++++++++
.......++++++++++++++
%HTTPD-I-DEMO, demonstration mode
8< snip 8<
%HTTPD-I-BEGIN, 18-SEP-2020 15:48:59, WASD:7080 accepting requests
```

A number of deficiencies were encountered with V9.0.
Establishing where the execution was breaking took time.
Creating simple reproducers to pass back to VSI took time.
However after eight hours of real work the server was basically up and running.

x86-64 Porting

EAK 9.0 (September 2020)

- 163,000 code lines
  X86 code specifics: 8
- each of these paralleled existing
  __VAX, __ALPHA, __ia64 specifics

```
$ search *.c "#ifdef __x86

*****************************
WASD_ROOT:[src.HTTPDX]httpd.c;6

#ifdef __x86_64
#ifdef __x86_64
#ifdef __x86_64

*****************************
WASD_ROOT:[src.HTTPDX]net.c;17

#ifdef __x86_64

*****************************
WASD_ROOT:[src.HTTPDX]sysplus.c;3

#ifdef __x86_64

*****************************
WASD_ROOT:[src.HTTPDX]tcpip.c;7

#ifdef __x86_64
#ifdef __x86_64

*****************************
WASD_ROOT:[src.HTTPDX]version.c;3

#ifdef __x86_64
```

https://wasd.vsm.com.au/info-WASD/2020/0102
https://wasd.vsm.com.au/other/WASD%20x86-64%2019-SEP-2020.html

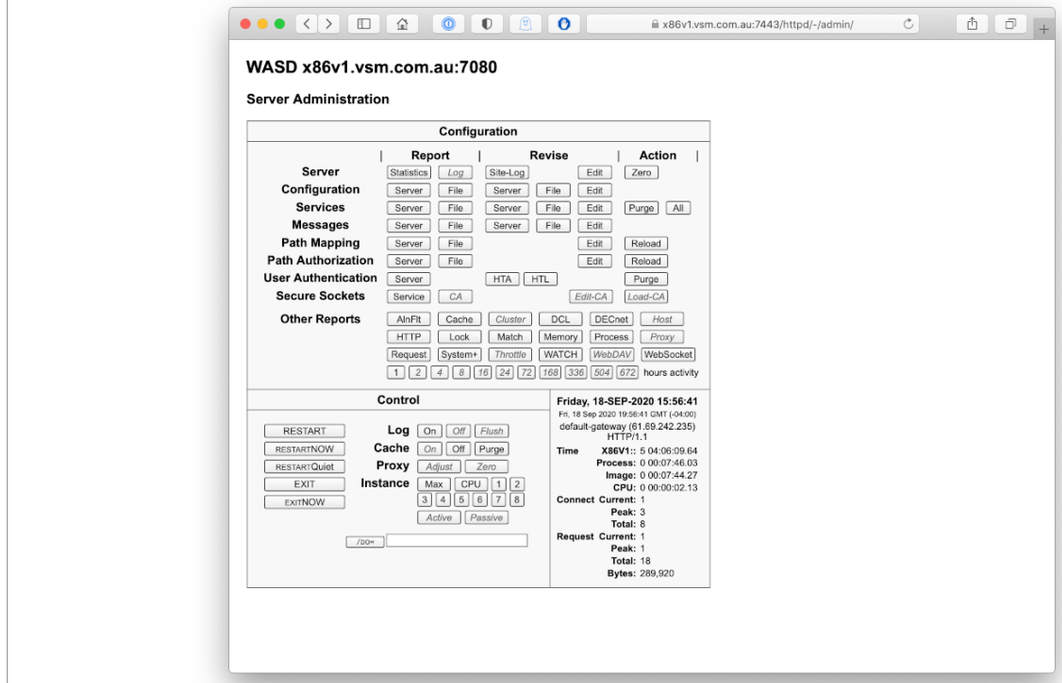
Just to reiterate.
Many of which were fairly trivial cut-n-paste modification of a literal.
But not all of them.

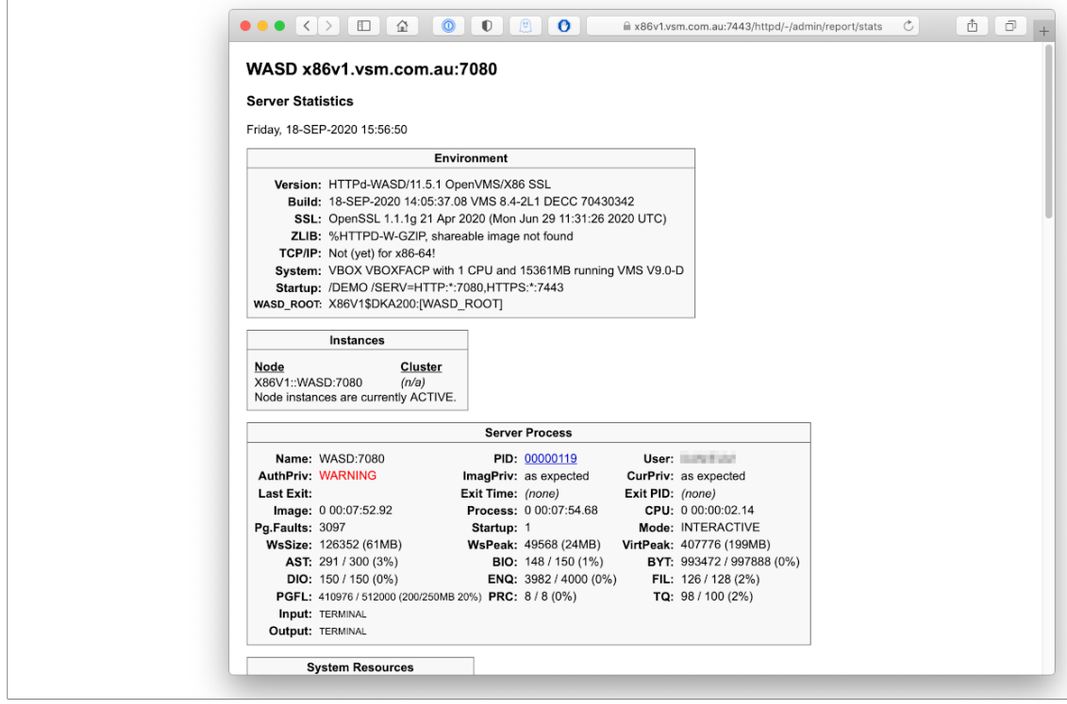
```
      if (ExitStatus != SS_W_CONTROLY)
      {
         /* add traceback information */
         int  SanityCheck = 100;
#ifdef __ALPHA
         lib$get_curr_invo_context (&icb);
         while (lib$get_prev_invo_context (&icb),
                !icb.libicb$v_bottom_of_stack && SanityCheck--)
            fprintf (stdout, "-HTTPD-F-TRACE, %08.08X%08.08X\n",
                     icb.libicb$q_program_counter[1],
                     icb.libicb$q_program_counter[0]);
#endif
#ifdef __ia64
         lib$i64_init_invo_context (&icb, LIBICB$K_INVO_CONTEXT_VERSION);

         lib$i64_get_curr_invo_context (&icb);
            while (lib$i64_get_prev_invo_context (&icb),
                   !icb.libicb$v_bottom_of_stack && SanityCheck--)
            fprintf (stdout, "-HTTPD-F-TRACE, %08.08X%08.08X\n",
                     ((ULONGPTR)&icb.libicb$ih_pc)[1],
                     ((ULONGPTR)&icb.libicb$ih_pc)[0]);
#endif
#ifdef __x86_64
         lib$x86_init_invo_context (&icb, LIBICB$K_INVO_CONTEXT_VERSION);

         lib$x86_get_curr_invo_context (&icb);
            while (lib$x86_get_prev_invo_context (&icb),
                   !icb.libicb$v_bottom_of_stack && SanityCheck--)
            fprintf (stdout, "-HTTPD-F-TRACE, %08.08X%08.08X\n",
                     ((ULONGPTR)&icb.libicb$ih_ip)[1],
                     ((ULONGPTR)&icb.libicb$ih_ip)[0]);
#endif
         /* list current and history list requests */
         RequestDump ();
      }
```

This is a portion of the server error exit handling.
Reinventing an X86-specific section of IA64 code.
Required checking the C header file for an X86 equivalent of the IA64 calls.
So - trivial bordering on the non-trivial - or non-trivial bordering on the trivial :-)
My thanks to Jean-Pierre Petit for providing the original code for this traceback. Jean-Pierre has never reported a problem (and he has a few over the years) without also providing a solution. Merci beaucoup.


x86v1.vsm.com.au:7443/httpd/-/admin/
WASD x86v1.vsm.com.au:7080
Server Administration
Configuration
Report
Revise
Action
Server
Statistics
Log
Site-Log
Edit
Zero
Configuration
Server
File
Server
File
Edit
Services
Server
File
Server
File
Edit
Purge
All
Messages
Server
File
Server
File
Edit
Path Mapping
Server
File
Edit
Reload
Path Authorization
Server
File
Edit
Reload
User Authentication
Server
HTA
HTL
Purge
Secure Sockets
Service
CA
Edit-CA
Load-CA
Other Reports
AInFit
Cache
Cluster
DCL
DECnet
Host
HTTP
Lock
Match
Memory
Process
Proxy
Request
System+
Throttle
WATCH
WebDAV
WebSocket
1
2
4
8
16
24
72
168
336
504
672
hours activity
Control
RESTART
RESTARTNOW
RESTARTQuiet
EXIT
EXITNOW
Log
On
Off
Flush
Cache
On
Off
Purge
Proxy
Adjust
Zero
Instance
Max
CPU
1
2
3
4
5
6
7
8
Active
Passive
/DO=
Friday, 18-SEP-2020 15:56:41
Fri, 18 Sep 2020 19:56:41 GMT (-04:00)
default-gateway (61.69.242.235)
HTTP/1.1
Time
X86V1:: 5 04:06:09.64
Process: 0 00:07:46.03
Image: 0 00:07:44.27
CPU: 0 00:00:02.13
Connect Current: 1
Peak: 3
Total: 8
Request Current: 1
Peak: 1
Total: 18
Bytes: 289,920


x86v1.vsm.com.au:7443/httpd/-/admin/report/stats
WASD x86v1.vsm.com.au:7080
Server Statistics
Friday, 18-SEP-2020 15:56:50
Environment
Version: HTTPd-WASD/11.5.1 OpenVMS/X86 SSL
Build: 18-SEP-2020 14:05:37.08 VMS 8.4-2L1 DECC 70430342
SSL: OpenSSL 1.1.1g 21 Apr 2020 (Mon Jun 29 11:31:26 2020 UTC)
ZLIB: %HTTPD-W-GZIP, shareable image not found
TCP/IP: Not (yet) for x86-64!
System: VBOX VBOXFACP with 1 CPU and 15361MB running VMS V9.0-D
Startup: /DEMO /SERV=HTTP:*:7080,HTTPS:*:7443
WASD_ROOT: X86V1$DKA200:[WASD_ROOT]
Instances
Node Cluster
X86V1::WASD:7080 (n/a)
Node instances are currently ACTIVE.
Server Process
Name: WASD:7080 PID: 00000119 User:
AuthPriv: WARNING ImagPriv: as expected CurPriv: as expected
Last Exit: Exit Time: (none) Exit PID: (none)
Image: 0 00:07:52.92 Process: 0 00:07:54.68 CPU: 0 00:00:02.14
Pg.Faults: 3097 Startup: 1 Mode: INTERACTIVE
WsSize: 126352 (61MB) WsPeak: 49568 (24MB) VirtPeak: 407776 (199MB)
AST: 291 / 300 (3%) BIO: 148 / 150 (1%) BYT: 993472 / 997888 (0%)
DIO: 150 / 150 (0%) ENQ: 3982 / 4000 (0%) FIL: 126 / 128 (2%)
PGFL: 410976 / 512000 (200/250MB 20%) PRC: 8 / 8 (0%) TQ: 98 / 100 (2%)
Input: TERMINAL
Output: TERMINAL
System Resources

8 hours
8 X86 specifics
163,000 code lines

x86-64 Porting

native 64 bit storage

- After the relative ease of the initial port (EAK 9.0)

  it was decided to retire VAX support (not a difficult decision)

  and remove all VAX 32 bit accomodations

  using native 64 bit data storage (e.g. [long][long] to [int64])

- This design decision was totally unrelated to X86 support
- Lots of fairly dry hack work and six months incremental field testing

  across 4 sites and 3 platforms later … v12.0.0

x86-64 Porting

EAK V9.1 (June 2021) and V9.1-A (September 2021)

- Deciding I needed some further hands-on, deployed existing

  MacBook Pro, Intel Core i5, 2.7 GHz, 8GB

  a basic laptop

- Then a AU$300 (~€200) eBay purchase

  Dell Optiplex 9020 SFF i7-4770 QC 3.4Ghz 16GB Windows 10 Pro

  a basic desktop

- I now have fairly cost-effective VMS development systems

  https://wasd.vsm.com.au/info-WASD/2021/0063
  https://wasd.vsm.com.au/info-WASD/2021/0091

Unlike V9.0, V9.1 was deployed by VSI as an ISO image and installed using the familiar

1) Upgrade, install or reconfigure OpenVMS …
2) Display layered products that this procedure can install

…

9) Shut down this system

x86-64 Porting

cross compiling

- All compilation (currently) requires an Itanium host

```
IA64$ product show history X86_XTOOLS
------------------------------------ ----------- ----------- --- -----------
PRODUCT                              KIT TYPE    OPERATION   VAL DATE
------------------------------------ ----------- ----------- --- -----------
VSI I64VMS X86_XTOOLS V9.1-A_XG1K    Full LP     Install     Val 27-OCT-2021
VSI I64VMS X86_XTOOLS V9.0-H_XFX8    Full LP     Remove       -  27-OCT-2021
VSI I64VMS X86_XTOOLS V9.0-H_XFX8    Full LP     Install     Val 18-APR-2021
VSI I64VMS X86_XTOOLS V9.0-F_XFRZ    Full LP     Remove       -  18-APR-2021
VSI I64VMS X86_XTOOLS V9.0-F_XFRZ    Full LP     Install     Val 18-DEC-2020
VSI I64VMS X86_XTOOLS V9.0-C_XFN5    Full LP     Remove       -  18-DEC-2020
VSI I64VMS X86_XTOOLS V9.0-C_XFN5    Full LP     Install     Val 13-SEP-2020
------------------------------------ ----------- ----------- --- -----------
7 items found
```

$ @SYS$MANAGER:X86_XTOOLS$SYLOGIN.COM

For formal V9.1 X86 release (i.e. WASD community) build procedures required some modification so that building under the cross-compiler (i.e. on Itanium) targeted X86 object directories.

x86-64 Porting

cross compiling

- Parallel source code trees

```
IA64$  SET DEFAULT WASD_ROOT:[000000]
IA64$  ZIP  “-V”  disk:[directory]file.ZIP  […OBJ_X86_64]*.OBJ
…
X86$  SET DEFAULT WASD_ROOT:[000000]
X86$  UNZIP  disk:[directory]file.ZIP
```

- a minor embuggerance
- Alternatively; use clustered IA64-X86 nodes and MSCP-shared volume
  Clustering works very effectively
- native X86 compilers promised incrementally during V9.2 releases

x86-64 Performance

V9.1-A (December 2021)

- VMS itself boots in seconds

- ```
  ! Dell Optiplex 9020 4 core i7 3.4Ghz 16GB
  X86VMS$ @vups.com
  innotek GmbH VirtualBox with 3 CPU and 4492MB running VMS V9.1-A
  Approximate System VUPs Rating : 244.9 ( min: 244.4 max: 245.4 )

  ! BXNUC10i7FNH4 6 core i7 1.10GHz 32GB
  innotek GmbH with 2 CPU and 7680MB running VMS V9.1-A
  Approximate System VUPs Rating : 456.8 ( min: 453.2 max: 459.2 )

  Digital Personal WorkStation with 1 CPU and 1536MB running VMS V8.4-2L1
  Approximate System VUPs Rating : 161.3 ( min: 161.2 max: 161.4 )

  AlphaServer DS20 500 MHz with 2 CPU and 1536MB running VMS V8.4-2L2
  Approximate System VUPs Rating : 254.8 ( min: 254.8 max: 254.8 )

  HP rx2660 (1.40GHz/6.0MB) with 4 CPU and 14335MB running VMS V8.4-2L1
  INFO: Preventing endless loop (10$) on fast CPUs
  Approximate System VUPs Rating : 499.6 ( min: 497.8 max: 501.4 )
  ```

This is the “standard” VUPS.COM DCL procedure that’s been kicking around for ever. Modified to output the hardware, CPU count, memory size, and VMS version.

Doesn’t provide an absolute measure of performance but is useful for comparative purposes.

Also remember, this is non-optimised operating system code!
  ```

x86-64 Performance

V9.1-A (December 2021)

- ```
  X86VMS$ TCPIP SHOW VERSION
  VSI TCP/IP Services for OpenVMS x86_64 Version X6.0
  on an innotek GmbH VirtualBox running OpenVMS V9.1-A
  ```

- ```
  X86VMS$ curl -ko nl: http://192.168.1.86/dka100/colossus.mp4
    % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                   Dload  Upload   Total   Spent    Left  Speed
  100 1398M  100 1398M    0     0  52.2M      0  0:00:26  0:00:26 --:--:-- 52.7M
  ```

- ```
  X86VMS$ MONITOR MODES
  Combined for  3 CPUs                0         75        150       225       300
                                      + - - - - + - - - - + - - - - + - - - - +
   Interrupt State                 39 |*****
   MP Synchronization                 |
   Kernel Mode                     36 |****
   Executive Mode                     |
   Supervisor Mode                    |
   User Mode                       47 |******
   Compatibility Mode                 |
   Idle Time                      177 |************************
  ```

Remember, this is non-optimised operating system code executing under a hypervisor!
innotek GmbH VirtualBox with 3 CPU and 4492MB running VMS V9.1-A
Approximate System VUPs Rating : 244.9 (min: 244.4 max: 245.4)
VSI TCP/IP Services 6.0 field test.
WASD X86 v12.0.0
cURL for X86 executing on the same X86 system (so no “real”) network I/O.
Sustained 500Mbps throughput.
SEK consuming 25% of available CPU, 16% USER, leaving 59% idle.
  ```

questions?

WASD development bench

- KLAATU the original Alpha PWS - 120W (regardless - and noisy!)
- BA356 storage shelf (courtesy Jeremy Begg) with 15k HDD - 25W
- X86VMS the Dell SFF (SSD x 2) - 20W (quiescent) 40-60W (active)

https://wasd.vsm.com.au/other/#WASD_x86-64

https://vsmx86.vsm.com.au

https://wasd.vsm.com.au/wasd/

https://vmssoftware.com/about/roadmap/

https://vmssoftware.com/about/openvmsv9-1/