News from VMS Software Inc. (VSI)

x86 OpenVMS V9.0 Update

Clair Grant, CTO

Terry R. Holmes, VP Sales & Marketing

VSI Recent Updates

- PERFDAT Acquired from HPE (January, 2019)
- Service Control (acquired with PERFDAT)
- Password Management (April, 2019)
- Several new BOE component updates including:
    - CSWS V2.4-38A / CSWS_JAVA V8.5-16A
    - NOTARY V0200
    - OpenSSL V1.0-2R / OpenSSL V1.1-1B
    - WSIT 3.4-1 / WSIT 3.4-2 (both for JAVA 8)
    - Layered Product: ACMS Dev, Remote & RT V5.3-2
- New Self Paced On-Line VSI Training program available now.

VSI Recent Updates

- VSI First Boot on x86 platform announced. (May 14)
- Several new Open Source Products including:
  - Lua V5.3-6
  - PHP V5.6-10J
  - Vgit2 V0.9-7
  - MariaDB 5.5-63
  - SAMBA (CIFS) 4.6.5
  - New IDE based on Visual Studio Code
  - PostgreSQL client V11.0-0A
- HPE ends all sales of VSI OpenVMS products effective June 30, 2019.
- New TCP / IP V10.5 (latest iteration) available now. V10.6 production version due our end of August, 2019.

Agenda

- Release Plan
- First Boot – Why was it important? What was It?
- V9.0
- Getting from First Boot to V9.0
- Current Status

Release Plan

Release Plan

- Cross Tools Kit: compile / link on IA64
  - Jan / 2019 – BLISS, C, XMACRO, Linker and associated tools
  - May / 2019 – Updates plus FORTRAN
  - Jul / 2019 – Updates plus PASCAL
  - COBOL and BASIC to follow
- V9.0 EAK – very limited developer kit (12-15 participants):
  - compile/link on IA64, run on x86
  - VirtualBox, kvm
  - Less than the complete OpenVMS production system
- V9.1 EAK – available to all customers; all system components
- V9.2 – production release

First Boot

Porting Play Book (The Plan)

Chapter 1 – Executable Images

- **Definition**: Register Mapping, Calling Standard extensions
- **Creation:** Compilers, Assembler
- **Action:** LIBRARIAN, LINKER, INSTALL, Image Activator
- **Analysis:** SDA, DEBUG/XDELTA, ANALYZE IMAGE, ANALYZE OBJECT

Chapter 2 – Architecture-Specific Needs (a.k.a. “The 5%”)

- Booting
- Interrupts, Exceptions
- Memory Management: protection types, access modes, address space, etc.
- Atomic Instructions
- Floating Point
- Special needs for code in assembler (e.g. VAX QUEUE instruction emulation)

Chapter 3 – Compiling and Linking Everything Else (a.k.a. “The 95%”)

- Large task but mostly mechanical
- Flush out any remaining ‘inter-routine linkage’ problems

First Boot – Mission Accomplished!

- Why was First Boot Important?
  - Identifiable point in the early life of the system
  - Good target for the engineers
  - Proof point for the customers

- Notable aspects of First Boot
  - Compilers and linker create executable code
  - Much new, platform-specific code is being executed
  - Much compiled MACRO-32 code is being executed
  - Increased the size of some data structures
  - Code runs in 64-bit space

Q: What Was Different for First Boot?
A: The operating environment (not the code)

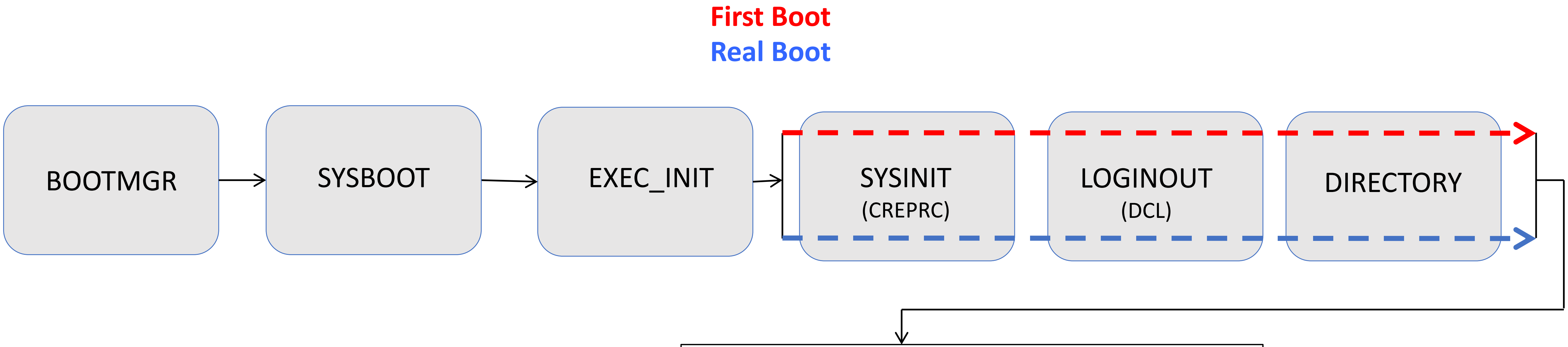
First Boot

1. All files were in the memory disk file
2. All files were built as execlets and therefore loaded into memory early during startup
3. System ran only in kernel mode
4. Booted memory disk over the network and started executing

Real Boot

1. Many files are on the system disk
2. Files not in the memory disk are found and loaded when needed
3. Need to switch modes and eventually run in user mode
4. Boot from system disk

System Startup



First Boot

Real Boot

BOOTMGR

SYSBOOT

EXEC_INIT

SYSINIT
(CREPRC)

LOGINOUT
(DCL)

DIRECTORY

```
$ DIR SYS$SYSTEM:

Directory SYS$SYSROOT:[SYSEXE]
X86_64VMSSYS.PAR;1

Total of 1 file.

Directory SYS$COMMON:[SYSEXE]

DCL.EXE;1          DIRECTORY.EXE;1     FASTPATH_SERVER.EXE;1
INDICTMENT_SERVER.EXE;1                LCKMGR_SERVER.EXE;1
LOGINOUT.EXE;1
SYS$CONFIG.DAT;1   SYSBOOT.EXE;1       SYSINIT.EXE;1

Total of 9 files.
```

XDELTA’s New Addition ;j

```
;J
Help for ;J - Jump to Debug Routine

   Usage   code[,data];J

      code - low 3 nibbles correspond to a debug routine index
             high bytes are for use by the debug routine
      data - 64bit value passed to the debug routine, optional

000 - This help display
001 - SHOW WSL, defaults to current process, data can be specific entry
002 - SHOW MEM
003 - SHOW PTEs for the VA passed in data
004 - SHOW ADDRESS for the VA passed in data (work in progress)
005 - SHOW PAGE for the VA passed in data
006 - SHOW POOL/RING [count]006,[address];J
007 - SHOW POOL (Non-Paged)
008 - SHOW LOG/SYSTEM
009 - SHOW DEVICE;J, Show summary of ALL configured devices
00A - SHOW DEVICE,[ucb_address];J, Show details of SELECTED device
00B - SHOW IRP,[IRP address];J, Show IRP I/O fields
00C - SHOW UCB,[UCB address];J, Show UCB I/O fields
00D - TR SHOW TRACE ,[count];J, Show TR Trace data, count default=32
00E - SHOW SUMMARY, Display processes summary
00F - SHOW VCB,[VCB address];J, Show VCB fields
010 - TQE SHOW TRACE ,[count];J, Show TQE Trace data, count default=32
011 - IO SHOW TRACE ,[count];J, Show IO Trace data, count default=32
012 - SHOW PFN ,[count],[address];J, Show PFN daGGta
013 - SWIS SHOW TRACE ,[count];J, Show SWIS Trace data, count default=32
014 - SHOW PROCESS ,[idx];J, Show Process details for index
```

9;j

```
9;j

Address of ioc$gl_devlist:  8000C5D0
Contents of ioc$gl_devlist: 80088380

Configured Devices:
-------------------

Device  Dev     Dev
Name    Class   Type    Op Cnt  Err Cnt Ref Cnt UCB Address        DDB Address
------  -----   ----    ------  ------- ------- -----------------  -----------------
OPA0    66      96      0       0       2       FFFFFFFF.80088558  FFFFFFFF.80088380
MBA1    160     1       0       0       1       FFFFFFFF.80089068  FFFFFFFF.80088C00
MBA2    160     1       0       0       1       FFFFFFFF.80089428  FFFFFFFF.80088C00
MBA3    160     1       0       0       1       FFFFFFFF.800897E8  FFFFFFFF.80088C00
NLA0    160     3       0       0       1       FFFFFFFF.80089C98  FFFFFFFF.80089BA8
DMM0    1       54      90      0       4       FFFFFFFF.8169A940  FFFFFFFF.81696CC0
SR0     0       0       0       0       0       FFFFFFFF.816ACA80  FFFFFFFF.816AC880

Found 7 configured devices
```

C;j

```
C,FFFFFFFF.8169A940;J

ucb$b_type = 10 (Valid UCB)

IO Related fields for UCB

        Field           Value
---------------         -----
ucb$l_devchar:          1E4C4008
ucb$l_devchar2:         00000200
ucb$l_ddb:              81696CC0  (Valid DDB)
ucb$l_vcb:              8169ED80  (Valid VCB)
ucb$l_ioqfl:            8169AA70  (Empty)
ucb$l_ioqbl:            8169AA70
ucb$l_irp:              8169F1C0  (Valid IRP)

ucb$l_sts:              08021810
ucb$l_devsts:           00000000
ucb$ps_io_counters:     00000000
ucb$l_bcnt:             00000400
ucb$l_boff:             00000000
ucb$l_pdt:              00000000
ucb$ps_sud:             8169ABC0
ucb$pq_svapte_sva:      0000000000000000
ucb$pq_extent:          000000008169F2D0
ucb$l_extent_boff:      00000000
```

V9.0

What is V9.0?

- VSI needs feedback from real customers doing real work.
- V9.0 will be “rough around the edges”
- Content - intersection between what people need to be productive and what VSI can have ready in a reasonable time
- Cross Tools Kit for compiling and linking
- Supported on VirtualBox and kvm
- Support will be directly from the engineering team
- Multiple updates prior to V9.1

V9.0 “IS NOT” (“MAYBE NOT” ?)

Would the absence of any of the following adversely affect your ability to make good use of V9.0?

- DECwindows server
- DECnet Phase IV
- DECnet Phase V (OSI)
- clusters
- volume shadowing
- reserved memory
- SMP
- XFC
- INSTALL /RESIDENT
- Support for privileged applications, for example 1) user written device drivers or 2) code that directly calls internal system routines such as those that manage page tables
- No VAX floating point support in the V9.0 cross compilers; all fp is IEEE. For V9.1 native compilers there will be VAX fp except for C++. (NOTE: It is TBD if it will ever be in C++ for x86.)

Sizing the V9.0 Proof Points

- Real Boot (L)
  - No special execlets
  - $ DIR
  - Boot from system disk
- kvm & VirtualBox booting are equivalent (M)
- Installation from webserver and DVD (XL)
- Crash Dumps (M), SDA (L)
- Conversational Boot (M)
- Create User Accounts (S)
- MOUNT/DISMOUNT disks (M)
- Run Batch Jobs (S)
- TCPIP: SFPT, SSH (XL)
- BACKUP (M)
- User mode DEBUG (XL)
- Run a threaded (POSIX) application (L)

S = easy, little work
M =
L =
XL = difficult, much work

Getting from First Boot to V9.0

Loaded Image List

```
1;L
Loaded Image List:
Seq  Image Name
68   [SYS$LDR]NT_EXTENSION
66   [SYS$LDR]VMS_EXTENSION
60   SYS$SRDRIVER
5E   SYS$LAN_VCITEST
5C   SYS$LAN_CSMACD
5A   SYS$LAN
58   SYS$EI1000X
56   SYS$DMDRIVER
54   SYS$TTDRIVER
52   SYS$ISA_SUPPORT
50   SYS$PCI_SUPPORT
4E   <SYS$LDR>TR$DEBUG
4C   <SYS$LDR>TQE$DEBUG
4A   <SYS$LDR>SYSINITX
48   <SYS$LDR>SYS$LOGINOUT
46   <SYS$LDR>SYS$DIRECTORY
44   <SYS$LDR>IO$DEBUG
40   ACME
3E   SYS$MME_SERVICES
3C   SYSLDR_DYN
3A   SYS$IPC_SERVICES
38   MULTIPATH
36   SYS$UTC_SERVICES
34   SYS$TRANSACTION_SERVICES
30   SYSLICENSE
2E   MESSAGE_ROUTINES
2C   SYS$VM
2A   SYSGETSYI
28   SECURITY_MON
26   IMAGE_MANAGEMENT
24   RMS
22   F11BXQP
20   LOGICAL_NAMES
1E   SHELL8K
1C   LOCKING
1A   PROCESS_MANAGEMENT_MON
18   SYSDEVICE
16   IO_ROUTINES_MON
14   EXCEPTION_MON
10   SYS$OPDRIVER
0E   SYSTEM_DEBUG
0C   SYSTEM_SYNCHRONIZATION_UNI
0A   SYSTEM_PRIMITIVES_2
08   SYS$ACPI
06   ERRORLOG
04   SYS$PLATFORM_SUPPORT
02   SYS$BASE_IMAGE
00   SYS$PUBLIC_VECTORS
```

New MDS Mitigation Informational

(Microarchitectural Data Sampling vulnerabilities)

Message during system startup…..

```
    VMS Software, Inc. OpenVMS (TM) x86_64 Operating System, XF8D-N4A
                 Copyright 2019 VMS Software, Inc.

SWIS-I-MDS Mitigation active, variant haswell(HASWELL/BROADWELL)
```

NOTES:

- Since the mitigation will cause a performance degradation, we will provide a method for disabling the mitigation. The default will be ‘enabled’.
- We will publish an estimate of the performance impact once we have a chance to do sufficient testing.

V9.0 = Real Boot and Much, Much More

- 344 individual developer tasks identified for V9.0 (approx. 20% are done)
- Eliminate “.IF DF X86_FIRST_BOOT” (and similar temporary mechanisms)
- Real Boot
  - Load all images (not just those needed for First Boot)
  - Image activation
  - Process rundown
  - Switch from Memory Disk to System Disk during startup
- Memory Management
  - Process page tables
  - Global sections
  - Adjust working sets
- Installation
- Exception Handling
- Run developers’ test programs
- Run Layered Products needed by participants
- Run UETP
- Run regression tests, I/O Hammer, etc.

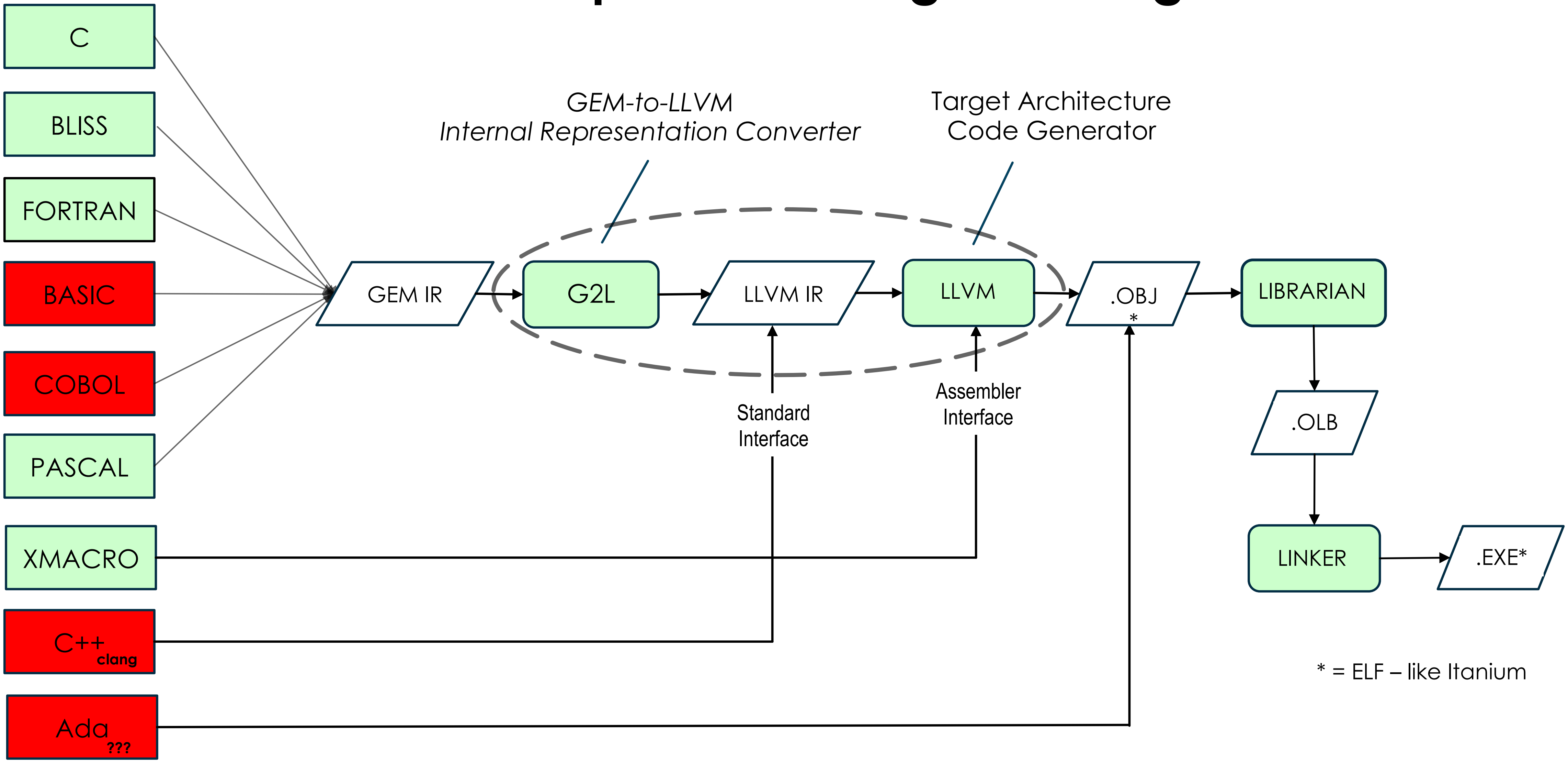
Current Status

Current Status of V9.0 Proof Points

- Real Boot (L)
  - No special execlets – DONE
  - $ DIR - DONE
  - Boot from system disk – 0%
- kvm & VirtualBox booting are equivalent (M) - DONE
- Installation from webserver and DVD (XL) – 75%
- Crash Dumps (M), SDA (L) – 75%, 25%
- Conversational Boot (M) – 90%
- Create User Accounts (S) – ready to test
- MOUNT/DISMOUNT disks (M) – ready to test
- Run Batch Jobs (S) - ready to test
- TCPIP: SFPT, SSH (XL) – 0%
- BACKUP (M) – ready to test
- User mode DEBUG (XL) – 10%
- Run a threaded (POSIX) application (L) – 25%

S = easy, little work
M =
L =
XL = difficult, much work

x86-64 OpenVMS Image Building



* = ELF – like Itanium

Bootstrapping LLVM to x86 for V9.1

- Current LLVM based on V3.4.2 (June 2014 release)
- The plan
  - Apply OpenVMS specific changes to LLVM 8.0.0 sources on a Linux system
  - Build it and move objects to OpenVMS IA64
  - Use cross-linker to create new LLVM for OpenVMS x86
  - Build libraries, like libcxx, and move objects to OpenVMS Itanium
  - Cross build everything and move to x86 for native builds
- Will upgrade to newer LLVM version prior to V9.1.

Factoids

- Current IA64 build – 887 images
- Current x86 build – 654 images (347 - 14 mos. ago)
- Need 122 more for V9.0
- Approx. 2500 individual module replacements, so far (excludes compilers)
  - New modules
  - Revised source modules
  - Upgrading build procedures
  - Verifying/updating conditionals (/* Verified for x86 port – John Smith */)
- First V9.0 build was 9 January 2017
- 26 modules in native assembler – most consist of a few short routines
- 1186 QTV test hours on V9.0 IA64 last week (approx. 30% of new/rewritten x86 memory management work is common code)

Thank You

To learn more please contact us:

vmssoftware.com
info@vmssoftware.com
+1.978.451.0110