VSI's Open Source Strategy

Plans and schemes for Open Source software on OpenVMS

Brett Cameron / Camiel Vanderhoeven
April 2016
AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions
Programming languages

• Scripting languages
  – Lua
  – Perl (probably in reasonable shape)
  – Tcl
  – Python
  – Ruby
  – PHP
  – JavaScript (Node.js and friends)
  – Also need to consider tools and packages commonly used with these languages

• Interpreted languages
  – Scala (JVM)
  – Clojure (JVM)
  – Erlang (potentially a good fit with OpenVMS; can get good support from ESL)
  – All the above are seeing increased adoption
Programming languages

• Compiled languages
  – Go (seeing rapid adoption)
  – Rust (relatively new)
  – Apple Swift

• Prerequisites (not all are required in all cases)
  – LLVM backend
  – Tweaks to OpenVMS C and C++ compilers
  – Support for latest language standards (C++)
  – Support for some GNU C/C++ extensions
  – Updates to OpenVMS C RTL and threads library
Programming languages

RedMonk Q315 Programming Language Rankings

1. JavaScript
2. Java
3. PHP
4. Python
5. C#
6. C++
7. Ruby
8. CSS
9. C
10. Objective-C
11. Perl
12. Shell
13. R
14. Scala
15. Go
16. Haskell
17. Matlab
18. Swift
19. Clojure
20. Groovy
21. Visual Basic

See http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15/
Steve O’Grady published another edition of his great popularity study on programming languages: RedMonk Programming Language Rankings: June 2015. As usual, it is a very valuable piece. There are many take-away from this research. I will not go over Steve O’Grady findings, but what I found interesting is:

• **Open Source and license matters.** For two of the hot languages, Erlang and Swift, we have seen important changes in licensing that may have an impact in the coming months.
  - Erlang changed its license from its Erlang Public License to a more widely accepted Apache V2. Steve O’Grady notes that it will not change language popularity but will remove friction for adoption and will make the language more attractive for large contributors. It is worth noting that Erlang is still growing and is now in the top 25, thanks to the amazing projects build with it.
  - Apple announced that Swift will be open source by end of the year, with a Linux version coming at the same time. This was a much needed change that will expand the community and accelerate the adoption of one of the fastest growing programming languages.

• **There are 4 booming programming languages:** Go, Swift, Rust and Julia. Go and Rust are competing for the same type of projects and developers. In a sense, with an open sourced Swift, it could reach the same target of system programming, even if it will be difficult to overshadow its mobile development roots. It will be interesting to see how those three languages evolves comparatively in the next research. Julia is a scientific language and evolves in its own niche space.

• **My current favorite, Elixir,** is not progressing as fast as Rust, despite reaching version 1 and being developed at an incredible page under Jose Valim’s vision. My feeling is that it is still in developer projet inception phase and that it is winning the heart of Erlang developers first, and many Ruby developers. I expect it to grow slowly in the coming months, as Phoenix Web Framework matures.

The programming language space is still extremely interesting to see evolve and I am looking forward seeing what the developer community is doing with them. Actual projects are the language king makers. For example ejabberd has been critical for Erlang popularity. Docker project is boosting Go adoption. Let’s watch other big project to understand programming languages future.

Programming languages

The 15 most popular languages in GitHub since 2012

https://www.loggly.com/blog/the-most-popular-programming-languages-in-github-since-2012/
AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions
Integration technologies

• Data-level integration
  – FreeTDS (SQL Server, Sybase)
  – UnixODBC
  – Not Open Source, but don’t forget about partners like Attunity
  – Various others
Integration technologies – message queuing

• **Protocols:**
  – AMQP (0.9.1 and 1.0)
  – MQTT
  – STOMP
  – CoAP (maybe)

• **Products:**
  – Mosquito broker and Paho client
  – RabbitMQ (requires Erlang) and libRabbitMQ
  – Possibly Qpid or OpenAMQ (needs a lot of work)
  – ZeroMQ
  – Kafka (requires Scala)
  – ActiveMQ (Java; runs well with current versions of Java and OpenVMS)
  – …
Integration technologies

• Web services
  – gSOAP
  – AXIS2
  – RESTful services (libcURL is useful here)
  – ...

• API-level integration
  – Various WSIT enhancements (see later)
  – ...

• CIFS
  – Not exactly an integration technology, but might as well stick it in here
  – Needs updating
AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions
Databases

• SQL/relational:
  – PostgreSQL
  – MySQL and/or MariaDB
  – Firebird (ex-InterBase)
  – …

• NoSQL:
  – Riak (requires Erlang; potentially a good match with OpenVMS and clustering)
  – CouchDB (requires Erlang)
  – MongoDB (needs C++ at current standard)
  – Cassandra (Java)
  – Many other possibilities in this space (HBase, …)

• Caching:
  – Redis (certainly client APIs)
  – MemcacheD

In addition to the databases and caches themselves, it is also important to provide client API’s
AGENDA

- Programming languages
- Integration technologies
- Databases
- Web
- Libraries/utilities
- Software development tools
- Cloud
- UNIX compatibility
- Analytics
- Add-ons
- Other considerations
- Summary/conclusions
- Questions
Web (clients and servers)

- Web servers:
  - Apache HTTPD
    * New versions are a fairly high priority
    * More modules
  - Mongoose
  - Tomcat
    * New versions (need Java 1.7, 1.8)
  - Nginx

- HTTP clients:
  - cURL, libcURL
  - Other

- Also need to be thinking about HTTP/2
AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions
Libraries/utilities – building blocks

- YAML, JSON, XML, ...
- Google Protocol Buffers (and several similar such technologies)
- OAuth
- OpenSSL
  - Needs to be kept current
  - Ship object libraries as well as shareable images (done with Bolton)
- Gearman
- Others (libraries, API’s, ...)

- libffi (see https://sourceware.org/libffi/)
  - Has been ported before
  - Used with the likes of Python to interface with 3GL code
  - FFI == Foreign Function Interface
AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions
Software development tools

• IDE’s
  – eCube (Eclipse-based)
  – Other options
    • Possibly do more with NetBeans and Distributed NetBeans
    • Something like Code::Blocks and Uniwin perhaps

• Source code control
  – Git
  – Subversion
  – Mercurial
  – CVS (old, but simple to use, still a viable option)

• Testing tools
• Continuous integration
  – NXTware Remote for Jenkins from eCube
• Open Source package management (along lines of tools provided by Cygwin or Ubuntu)
  • ...
AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions
Cloud

- Client APIs and CLIs to facilitate interaction with cloud-based services
  - Amazon EC2
  - Google
  - OpenStack (HP Helion, Rackspace, ...)

- API support for services such as:
  - IronMQ
  - Amazon SQS, SNS
  - Dweet.io
  - Xively
  - ...

- Containers
  - Longer-term (x86) maybe

It is not really Open Source, but being able to hook into cloud-based services is something that is of considerable interest, and technically it is not too difficult to do.
AGENDA

- Programming languages
- Integration technologies
- Databases
- Web
- Libraries/utilities
- Software development tools

- Cloud
  - UNIX compatibility
- Analytics
- Add-ons
- Other considerations
- Summary/conclusions
- Questions
UNIX compatibility (GNV)

• Needs updating and expanding
  – Can leverage good work done by the community
  – Needs to be properly supported

• Probably a separate discussion...
AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions
Analytics

- R (extensible programming language for statistical computing)
- ...

- Apache Spark
- Apache Flink
- Kafka
- ...

- The likes of Big Data and Internet of Things need to be key focus areas

Many of these Java-based technologies such as Spark, Flink, and Kafka require higher versions of the JVM.

AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions
Add-ons (value-adds)

• More OpenVMS-friendly APIs for some Open Source products

Many OpenVMS users do not have developers that are able to incorporate C-based API’s into their legacy application code, which may be written in languages such as COBOL, Fortran, or BASIC. It is therefore important to provide wrapper APIs that can be more readily used with these languages in a more OpenVMS-like way.
Add-ons (value-adds)

- OpenVMS-specific extensions for languages such as Python, Ruby, Lua, PHP, Erlang, ...
- Integration with other OpenVMS-specific technologies
  - ACMS
    - Better access to audit trail data
    - Mechanism(s) for passing objects greater than 64K
  - WSIT
    - Support for protocols other than ICC
      - AMQP
      - ...
    - Support for other languages such as C#.NET (requires additional protocol support)
  - RTR (maybe)
  - UAF-based authentication (Mosquitto, RabbitMQ, ...)
  - ...
- Monitoring as a service
AGENDA

- Programming languages
- Integration technologies
- Databases
- Web
- Libraries/utilities
- Software development tools

- Cloud
- UNIX compatibility
- Analytics
- Add-ons
  - Other considerations
- Summary/conclusions
- Questions
Other considerations

• Prioritization
• Value to customers (usefulness)
• Resources
• Prerequisites (updates to the C RTL, C/C++ compilers, ...)
• Community involvement
  – Which Open Source packages do VSI port, maintain, and support; which packages are left to the community; collaboration?
• Input from community and customers
• IP considerations
• Open Source licensing considerations
• Support
• Consulting services and training
• Documentation
• ...
Value considerations

What Open Source software is likely to be of greatest use to current OpenVMS users?

• Short term
• Medium term
• Long term

Need to take a strategic approach and consider industry trends

• Internet of Things
• Big Data
• Containers
• Cloud
• ...
• Where does OpenVMS provide advantages?

Many OpenVMS users have very old application environments

• Need to provide software (and potentially services) that will help these users
Value considerations

What Open Source software will help to solidify OpenVMS' position with existing users?

• Integration technologies
  – Some users with legacy applications have very little perception of what can be done from an integration perspective

• Technologies that present opportunities to reduce 3rd party license and/or support costs
  – Open Source replacements for expensive, poorly supported, or unsupported software
  – ...

• Users on Alpha looking to move to Integrity who need replacement options for technologies not available on Integrity
Value considerations

What Open Source software is likely to attract developers to OpenVMS?
• Modern language environments such as Ruby, Python, Go, Erlang, Rust, Node.js, Scala, Clojure, latest Java versions, ...

Modern toolsets
• IDE’s and related development tools
• Source code management
• Testing tools
• ...

What Open Source software is likely to encourage developers to port applications to OpenVMS?
• As per the above
• The option of a good UNIX shell and related utilities (enhancements to GNV)
Dependencies and related matters

• Many Open Source products depend on other Open Source products
  – Libraries/API’s
  – These are often fundamental building blocks

• C RTL issues
  – Missing functions
  – Differences in header files
  – Behavioural differences

• C/C++ language standards
• ...
Dependencies and related matters

• Some positioning for the next slide…
  – Illustrates some (but not all) dependencies and issues associated with building Erlang on OpenVMS
  – Blue boxes are dependencies
  – Red boxes are problematical but optional aspects
  – Black boxes are just the usual sorts of things that have to be dealt with when porting complex Open Source application to OpenVMS
PCRE (Perl regular expressions library). Easily ported to OpenVMS; no code changes required.

zlib compression library. Easily ported to OpenVMS (has been done numerous times).

libreadline for command line editing and history (optional). Ported to OpenVMS as part of the GNV project; functionality easily replicated using SMG routines.

OpenSSL libraries. HP SSL for OpenVMS is at a high enough version. Otherwise can use http://www.polarhome.com/openssl/.

Use of syslog API. Potentially need to implement something comparable, but can live without it.

Essential to run with UNIX-style path and file naming otherwise extensive code changes would be required. Necessitates ODS-5.

Threads (Erlang scheduler). The default thread stack size on OpenVMS is too small. This is also an issue on other platforms, and code tries to accommodate such matters (need to ensure that certain macros are correctly defined). The OpenVMS POSIX threads library is missing some functions (pthread_sigmask() for example); however this is generally not a major problem to contend with.

Some terminal I/O code requires use of appropriate ioctl() calls in place of tcsetattr().

Behaviour of getcwd(). OpenVMS extension controls how the result is returned (UNIX-style or OpenVMS-style).

Sundry file differences and header files that don’t exist on OpenVMS (<sys/param.h>) would be one example.

fork() function not available on OpenVMS. Only used in a few places, and code can be compiled to not use it. Can get away with using vfork() in most cases.

poll() and select() functions only work with sockets on OpenVMS. It is therefore necessary to implement custom versions of poll() and select() that work with other types of descriptor. These custom versions are unlikely to be particularly efficient (scope for improvement). The select() function also behaves differently on OpenVMS under other certain circumstances (like when timeout is set to 0).

fcntl() C RTL function behaves differently on OpenVMS in certain situations. Cannot be used to toggle sockets blocking/non-blocking; necessary to use ioctl() instead. Some other platforms behave this way also, so Erlang code is aware of this and just need to ensure certain macros correctly defined when compiling.

Default behaviour of pipe() is not totally consistent with Linux. Can need to specify optional argument to set non-blocking and use decc$set_child_standard_streams() to ensure that things are set up correctly for communication between parent and child processes.

Native compilation of Erlang code (HIPE). Currently no support for the IA64 processor; need to consider whether implementing support for IA64 makes sense (given future plans around x86).

Certain OpenVMS C RTL functions will set errno to EWOULDBLOCK instead of EINTR. Other platforms (such as Windows) behave in a similar manner, and the Erlang code has (in most places) taken this matter into consideration.

64-bit versions of some functions missing from OpenVMS C RTL.
AGENDA

- Programming languages
- Integration technologies
- Databases
- Web
- Libraries/utilities
- Software development tools

- Cloud
- UNIX compatibility
- Analytics
- Add-ons
- Other considerations
- Summary/conclusions
- Questions
Summary

• There’s a lot to do!
  – New stuff as well as updates to existing packages
• Languages, databases, and integration technologies are arguably some of the big-ticket items
  – Both for existing users and to attract new users
  – Development tools can be added to this list
    • Source code control
    • Testing
    • ...
  – Package management is another important consideration
• There are some significant dependencies on Java
• Need to take a multi-pronged approach, involving some “quick wins” (small but strategic items) and larger pieces of work being done in parallel
• Identify, prioritise, and systematically address C RTL and compiler-related issues
• Model for community and partner engagement needs to be defined
• ...
AGENDA

• Programming languages
• Integration technologies
• Databases
• Web
• Libraries/utilities
• Software development tools

• Cloud
• UNIX compatibility
• Analytics
• Add-ons
• Other considerations
• Summary/conclusions
• Questions