

Examples of Open Source software on OpenVMS

What, where, why, how and when

Brett Cameron

September 2014

Abstract

The speaker has done a considerable amount of work over the past decade porting various pieces of Open Source software to the OpenVMS platform and working with OpenVMS customers to craft solutions using this Open Source software. During this talk the speaker will describe some of software that has been ported and where and how this software might be useful, with particular emphasis on Open Source web services and message queuing technologies. Work in progress and future plans will also be discussed. Additionally, the talk will examine the general approach taken by the speaker to the porting process, along with some of the pitfalls that are commonly encountered and how these may be resolved. In addition to the porting of software to OpenVMS, work has been done to implement interfaces for some of the ported Open Source packages to simplify their use from OpenVMS high-level languages such as COBOL, Pascal, and FORTRAN. Examples of such interfaces will be presented and the approaches taken to implementing them will be briefly discussed.

About me

Brett Cameron currently works as a senior architect with HP's corporate Cloud Services group, focusing on the design and implementation of message queuing and related integration services for customers and for internal use. Brett lives in Christchurch, New Zealand, and has worked in the software industry for some 22 years. In that time he has gained experience in a wide range of technologies, many of which have long since been retired to the software scrapheap of dubious ideas. In recent years Brett has specialized in systems integration, and the design and implementation of large distributed systems for HP's enterprise customers. This work has seen Brett get involved in the research and development of low-latency and highly scalable messaging solutions for the Financial Services sector running on HP platforms, and as a consequence of this work, Brett has been involved in several interesting Open Source projects, and he has been responsible (or should that be irresponsible) for porting various pieces of Open Source software to the HP OpenVMS platform. Brett holds a doctorate in chemical physics from the University of Canterbury, and still maintains close links with the University, working as a part time lecturer in the Computer Science and Electronic and Computer Engineering departments. In his spare time, Brett enjoys listening to music, playing the guitar, and drinking beer.



AGENDA

- **Introduction**
- Overview of ported pieces of Open Source software
- General approach to porting
- Summary and questions

Motivations

- To make available on OpenVMS software technologies that help customers extend their computing environments to meet business needs
- More and better integration options for OpenVMS
 - Ability call or serve web services
 - Becoming a very common requirement
 - Web service-based integration with new Java or .Net-based solutions
 - Support for emerging integration standards
 - AMQP is being widely adopted by the financial sector
 - AMQP is a viable alternative to traditional OpenVMS message queuing solutions
 - Demonstrate that OpenVMS can participate in the “modern” integrated world
- Prove the viability of Open Source solutions on OpenVMS
 - And show that OpenVMS is an excellent platform for many Open Source technologies

5

Focus areas

- Key focus areas are:
 - Web
 - Integration (message queuing, data integration, cloud)
 - Caching and no-SQL databases
 - Programming languages and scripting
- In addition to just porting, the aim is to add value
 - Provide OpenVMS-friendly APIs
 - Develop utilities to streamline development and integration tasks
 - Example code
 - Consulting services
 - Support

6

AGENDA

- Introduction
- **Overview of ported pieces of Open Source software**
- General approach to porting
- Summary and questions

gSOAP

- Full-featured Open Source SOAP technology
 - See <http://www.cs.fsu.edu/~engelen/soap.html>
- More than 2500 registered users (including several Fortune 100 companies)
- Conforms to WS-I Basic Profile V1.0a
- Uses a source-to-source stub and skeleton compiler to automate the integration of SOAP RPC in applications
 - Client and server development
 - Automates the deployment of (legacy) 3GL applications as web services
 - Primarily intended for use with C/C++, but can be used (with a little effort) with any 3GL
 - Automates the development of clients
- Suitable for high-performance web service computing (very fast)
- Major components available on OpenVMS (Alpha and IA64)
 - Extensions to simplify use from languages other than C/C++
 - ACMS support (threaded agent gateway)
 - Apache support via `mod_gsoap`

gSOAP

- gSOAP goals:
 - Application-centric
 - Minimize legacy application code adaptation
 - Support (de)marshalling of application's native data structures in SOAP/XML
 - Preserve the logical structure of data
 - Minimize data migration overhead and formatting errors
 - Avoid (hand-written) wrappers
 - Generate fast (de)marshalling routines and streaming XML parsers
 - Efficient run-time remote object allocation
- The software is being used in production by at least 25 OpenVMS customers (and others are actively looking at using it)
 - See <http://www.johndapps.com/download> for more information
 - See also my joint talk "*OpenVMS and web services; theory and practise*"

9

FastCGI

- FastCGI is a protocol for interfacing interactive programs with a web server
 - A variation on the earlier Common Gateway Interface (CGI)
 - FastCGI's main aim is to reduce the overhead associated with interfacing the web server and CGI programs, allowing a server to handle more web page requests at once
 - FastCGI is generally many times faster than traditional CGI
- Instead of creating a new process for every request as per CGI, FastCGI can use persistent processes that handle many requests over their lifetime
 - Processing of multiple requests simultaneously is achieved either by using a single connection with internal multiplexing and/or by using multiple connections
 - Many such processes can exist; something that can increase stability and scalability
- Environment information and page requests are sent from the web server to FastCGI processes via TCP/IP or UNIX domain sockets
 - Responses are returned from the process to the web server over the same connection
 - The connection may be closed at the end of a response, but the web server and the process persist
 - Currently TCP/IP only on OpenVMS (could do UNIX domain sockets on VMS 8.4)

10

FastCGI

- The traditional CGI approach...
 - A new process is started by the web server to satisfy each request
 - Data for the request is read by the CGI program from the environment (a GET request) or via `stdin` (a POST request)
 - Response data is written to `stdout`
- The FastCGI approach...
 - The FastCGI application is started when the web server is started or when the first request for the service is received
 - The FastCGI processes communicate with the web server via a socket
 - The FastCGI process persists until the web server is shutdown or until the process is otherwise stopped
- Current status:
 - Ported to both Alpha and Integrity
 - Port includes `mod_fastcgi` for Apache, and facilities to integrate with gSOAP
 - Java support
 - See <http://www.fastcgi.com/drupal/> and <https://sites.google.com/a/johndapps.com/www/fastcgionopenvms> for more information

11

MOD_ISAPI

- ISAPI (Internet Server API) dates back many years
 - One of the earlier methods used by web servers to provide something more efficient than CGI (Common Gateway Interface) for interfacing with other programs
 - Was (and still is) used by Microsoft's IIS web server and it was supported by the Purveyor Web server on both Windows and OpenVMS
- ISAPI functionality is available for Apache 2.1 via the `MOD_ISAPI` module, which has been ported to OpenVMS (Alpha and Integrity)
 - A port of `MOD_ISAPI` is also available upon request for Apache 1.3 on OpenVMS
- ISAPI applications have a number of advantages over external CGI applications
 - For example:
 - They are loaded into and run in the web server's process space, thereby eliminating the time and resource demands of creating additional processes
 - All data and resources available to the web server are also available to the ISAPI application
- The primary disadvantage in using an ISAPI application is that a bug in the application can potentially impact the operation of the web server
 - This is arguably less of an issue for Apache on OpenVMS, which scales by creating additional worker processes as opposed to being a single multi-threaded process

12

MOD_ISAPI

- Implementing ISAPI applications is very straightforward
 - Basically involves developing a shareable image that contains two entry points:
 - `GetExtensionVersion`
 - Does not really do anything other than return version information
 - `HttpExtensionProc`
 - Where all the action happens
 - Gets passed a structure that includes pointers to the data associated with the HTTP request, pointers to various utility functions, values of HTTP headers (or a means of accessing such values), and a few other things
 - Uses a utility function to direct response data back to the client
 - There are other entry points you can implement, but these two are the important (and mandatory) ones

- See <http://modisapionopenvms.blogspot.com/> for more information

13

Mongoose

- An easy-to-use embeddable web server
 - See <http://code.google.com/p/mongoose/>
 - Can be used standalone or as an embedded web server library to provide a web interface to applications
- Very fast and light-weight
 - Fully multi-threaded, which can be an issue if integrating with single-threaded applications
- Does not depend on any external libraries or configuration files
 - Can simply be copied into a directory and started
 - Default parameters can be overridden via a simple configuration file
- Ideal for all sorts of demos, quick tests, file sharing, and assorted web programming tasks
- Initially ported to OpenVMS in 2010 in response to a specific customer request
 - Re-reported mid-2012 using version 3.3
 - Used to implement an HTTP consumer to work with RabbitHub for my talk "*The Polyglot Rabbit; Adventures on OpenVMS with RabbitMQ, Erlang, and multi-protocol messaging*"

14

Memcached

- A general-purpose, high-performance, distributed, in-memory object caching system
 - See <http://memcached.org/> and <http://libmemcached.org/libMemcached.html>
 - Not as functionally rich as Redis
- Has been used by several very large, well-known sites, including YouTube, Facebook, and Twitter
 - Also currently used for various things in HP Public Cloud
- Often used to speed up dynamic database-driven web sites by caching data into memory
- Effectively a giant hash table distributed across multiple machines
 - Keys are up to 250 bytes long and values can be at most 1 megabyte in size
 - Clients use client-side libraries such as `libmemcached` to interact with servers
 - Clients may communicate with one or more servers
 - A typical deployment will have several servers and many clients
 - When the table is full subsequent inserts cause older data to be purged in least recently used (LRU) order
 - Clients must therefore treat Memcached as a transitory cache
 - The cache may be backed by a Memcached-protocol compatible database that provides persistent storage

15

Memcached

- Associated components ported to OpenVMS include:
 - `libmemcached`
 - A C client library for interfacing to a Memcached server
 - `mod_memcached_cache`
 - Apache module for use with Memcached, for caching static web pages (Apache 2.1+ only)
- Current status:
 - Currently ported version is 1.2.8 (latest release is 1.4.15... so we're a bit behind)
 - Fully functional port
 - Some stability issues when using UDP, but otherwise stable
 - Mostly complete OpenVMS-style client API that can be readily called from any OpenVMS 3GL
 - Partially complete CLI-based administrative utility
 - Used to implement a method of dealing with the ACMS 64K workspace limit
 - See <http://assorteddrables.blogspot.com.au/2013/07/using-blobs-in-acms-applications.html>

16

Redis



- An open-source, networked, in-memory, persistent, journaled, key-value data store
 - See <http://redis.io/>
- One of the main differences between Redis and other such solutions is that values are not limited to strings
 - In addition to strings, the following abstract data types are supported:
 - Lists of strings
 - Sets of strings (collections of non-repeating unsorted elements)
 - Sorted sets of strings (collections of non-repeating elements ordered by a floating-point number called score)
 - Hashes where keys are strings and values are either strings or integers
 - The type of a value determines what operations (commands) are available for the value itself
 - Supports high-level atomic server-side operations like intersection, union, and difference between sets, and sorting of lists, sets and sorted sets
- Supported language bindings include C, C++, C#, Erlang, Java, JavaScript, Lua, Perl, PHP, Python, Ruby, and Tcl

17

Redis

- Redis typically holds the entire dataset in memory
 - Very fast
- Persistence can be achieved in two ways:
 - Snapshotting
 - A semi-persistent durability mode where the dataset is periodically asynchronously transferred from memory to disk
 - A journal file that is (asynchronously) written as operations modifying the in-memory dataset are processed
 - Redis is able to rewrite the append-only file in the background in order to avoid an indefinite growth of the journal
- Supports master-slave replication
 - Useful for read (but not write) scalability and for data redundancy
- Current ported version is 2.1.1 (latest release is 2.8.12... so we're a bit behind)
 - Still some work to be done
 - Journaling and snapshotting require some further work
 - Various performance optimizations for OpenVMS
 - C client works fine on OpenVMS... may look at implementing an OpenVMS language-friendly wrapper

18

Apache CouchDB

- A document-oriented No-SQL database that can be queried and indexed in a MapReduce fashion using JavaScript
 - Offers incremental replication with bi-directional conflict detection and resolution
 - Provides a RESTful JSON API than can be accessed from any environment that allows HTTP requests
- Written in Erlang (mostly)
 - Ideal for building concurrent distributed systems
 - Allows for a flexible design that is easily scalable and readily extensible
- The OpenVMS port is a work in progress (early days)
 - Looking promising
 - JavaScript engine seems stable (needs some tuning)
 - YAJL/Pillowtalk RESTful interface working well
 - CouchDB4j (Java interface) mostly working

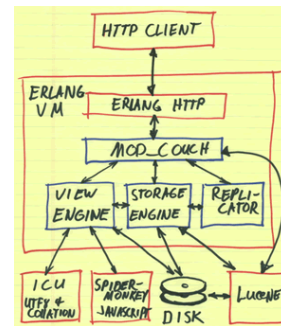


Image copied from <http://couchdb.apache.org/>

19

Google Protocol Buffers

- A serialization format with an interface description language
- Data structures and services are defined in a "Protocol Definition File"
 - This file is then compiled to generate code (in the desired language) that matches the defined services
 - This includes code to marshal and un-marshal buffers
- Developed by Google
- The original Google implementation for C++, Java, and Python is available under a free software, Open Source license
 - Various other language implementations are either available or in development
 - For instance, plain C is supported on OpenVMS
- The design goals for Protocol Buffers emphasize simplicity and performance
 - In particular, it was designed to be faster than XML
- Widely used at Google for storing and interchanging all kinds of structured information
 - Also serves as a basis for a custom RPC system that is used for practically all inter-machine communication at Google
- Support serialization into various other formats (XML, JSON)

20

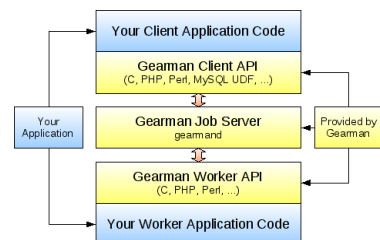
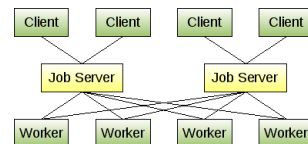
Google Protocol Buffers

- More information:
 - <http://code.google.com/apis/protocolbuffers/>
 - <http://gpbopenvms.blogspot.com/>
- Versions:
 - The current release of Google Protocol Buffers for OpenVMS is based on the Google Protocol Buffers 2.4.0 distribution and includes version 0.14 of Protobuf-C
 - Currently Integrity only
 - The latest version is 2.5.0
- Current status:
 - Being used in production by two large customers

21

Gearman

- A system to farm out work to other machines
 - Dispatch function calls to machines better suited to do work
 - Do work in parallel
 - Load balance large numbers of calls
 - Call functions between dispirit languages
 - Natural load distribution, easy to scale out
- See <http://gearman.org/> for more information
- Anagram for “manager”
 - Gearman, like managers, assigns tasks but does none of the real work!
- OpenVMS port is mostly working
 - Job server and C client API have been ported and are functional
 - Still a bit of a work in progress
 - Provides a number of interesting integration possibilities
 - Integrity only at this stage, but should be no problem building on Alpha
 - Based on a slightly older version of Gearman
 - Newer versions use the C++ Boost library, which is a bit of a beast to port to OpenVMS



22

Lua

- See <http://www.lua.org/>
- A powerful, fast, lightweight, embeddable scripting language that combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics
 - Dynamically typed
 - Runs by interpreting byte-codes using a register-based virtual machine
 - Automatic memory management with incremental garbage collection
 - Excellent scripting language for configuration, scripting, and rapid prototyping tasks
- OpenVMS port includes the core binary distribution and several useful extensions
 - LuaSocket
 - LuaExpat
 - LuaProc (coroutines in Lua)
 - LuaFileSystem
 - MOD_LUA (run Lua scripts via Apache)
 - Not much has been done on the OpenVMS Lua port for a while, but hopefully we'll get back into it soon
 - See <http://luaonopenvms.blogspot.com/>
 - The current build of Lua for OpenVMS is based on the Lua 5.1.4 distribution; also recently ported 5.2.3, but needs a little further work and testing



Note that Steve Hoffman has also done a port of Lua to OpenVMS (see <http://labs.hoffmanlabs.com/node/1425>)

23

Erlang

- A general-purpose programming language and runtime environment that has built-in support for concurrency, distribution, and fault tolerance
 - Also supports hot swapping, allowing code to be changed without stopping a system
- Way before Java could do such things!
- Designed by Ericsson to support distributed, fault-tolerant, soft-real-time, non-stop applications
 - Used by Ericsson in several large telecommunications systems
 - First version developed by Joe Armstrong in 1986
 - Released as Open Source in 1998
- Erlang implements a “shared-nothing” policy
 - Cooperating Erlang processes (as distinct from operating system processes) communicate using message passing instead of shared variables, which removes the need for any form of locking
- The popularity of Erlang has increased significantly in recent years due to its ability to address problems that cannot be readily or satisfactorily addressed by other languages
 - Particularly concurrency

24

Erlang

- Erlang is ideally suited to the implementation of distributed, highly-reliable, soft real-time concurrent systems
 - The language provides a simple and powerful model for error containment and fault tolerance
 - Concurrency and message passing are a fundamental to the language
 - Context switching between Erlang processes is typically several orders of magnitude cheaper than switching between threads in a C program
 - Writing applications made of components that execute on different machines is straightforward
 - The Erlang runtime environment is a virtual machine similar to the Java virtual machine
 - Code compiled on one platform can typically be run on any other platform that supports the Erlang virtual machine
- The capabilities of the core Erlang language are extended by OTP (Open Telecom Platform)
 - OTP is a large and collection of libraries for Erlang to do everything from compiling ASN.1 to providing a web server
 - Most Erlang applications make extensive use of facilities provided by OTP

25

Erlang

Some Erlang highlights:

Functional	<ul style="list-style-type: none"> • Light-weight processes • Highly scalable • Message Passing
Concurrent	<ul style="list-style-type: none"> • Erlang processes communicate by asynchronous message passing
Soft real-time	<ul style="list-style-type: none"> • Response times in the low milliseconds • Per-process garbage collection
Robust	<ul style="list-style-type: none"> • Simple and consistent error recovery • Supervision hierarchies • Cooperating processes
Distributed	<ul style="list-style-type: none"> • Explicit or transparent distribution • Network-aware runtime system
Hot code loading	<ul style="list-style-type: none"> • Easily change code in a running system • Enables non-stop operation • Simplifies testing
External interfaces	<ul style="list-style-type: none"> • “Ports” to the outside world behave as Erlang processes
Portable	<ul style="list-style-type: none"> • Erlang runs on a Virtual Machine (available for UNIX, Windows, VxWorks, OS X, OpenVMS, ...)

26

Erlang

- More information:
 - <http://www.erlang.org/>
 - <http://erlangonopenvms.blogspot.com/>
- The current build of Erlang for OpenVMS on Integrity is based on the Erlang R16A distribution
 - Current version is R16B02
 - A few issues to resolve, but generally the port appears quite stable and works very well
 - Full SMP and 64-bit support are probably the two biggest issues to be resolved...
 - Able to run (or mostly run) several large Erlang applications on OpenVMS
 - RabbitMQ
 - Yaws
 - CouchDB
 - Riak

27

RabbitMQ

- A powerful Open Source message broker (message-oriented middleware)
 - The leading (and arguably the most popular) implementation of AMQP
 - Provides a robust and flexible messaging platform designed to interoperate with other messaging systems
- RabbitMQ essentially comprises the following components:
 - The RabbitMQ broker (supports AMQP 0.8 and 0.9.1, and partial support for 1.0)
 - Gateways for HTTP, ZeroMQ, STOMP, MQTT, and other protocols
 - AMQP client libraries for Erlang, Java, .NET, and C/C++
 - AMQP clients for *numerous* other languages are available from other vendors and/or the Open Source community
 - Python, Ruby, PHP, ... just about any language you can think of
 - The "Shovel" plug-in that takes care of copying (replicating) messages from one broker to another
 - Numerous other useful plugins
- Clustering, queue replication for HA, ...
- Written entirely in Erlang/OTP



28

RabbitMQ

- In addition to the broker, we have also ported libRabbitMQ to OpenVMS
 - libRabbitMQ for OpenVMS provides an API that can be used by OpenVMS-based software applications to exchange data via AMQP
 - Based on an experimental C API originally developed by the RabbitMQ team
 - Designed so that it can be used with most programming languages that are available on OpenVMS, including C, FORTRAN, COBOL, and Pascal
 - Now working on tools using libRabbitMQ to simplify development of AMQP-based applications
 - A generic server/consumer (calls user-written functions in a shareable image)
 - A Tcl client/utility (useful for testing and some administrative functions)
 - Enhancements to WSIT to generate code for AMQP/libRabbitMQ
- More information:
 - <http://www.rabbitmq.com/>
 - <https://sites.google.com/a/johndapps.com/www/rabbitmqonopenvms>
 - <http://rabbitmqonopenvms.blogspot.com/>
 - <http://assortedrambles.blogspot.com/2012/11/the-polygot-rabbit.html>
 - http://assortedrambles.blogspot.com.au/2013/04/using-rabbitmq-from-cobol_9584.html

See my talk “*The Polyglot Rabbit; Adventures on OpenVMS with RabbitMQ, Erlang, and multi-protocol messaging*” for more information on RabbitMQ and AMQP, and on other Open Standards-based message queuing protocols such as MQTT and STOMP.

29

ØMQ (ZeroMQ)

- A high-performance (low-latency) messaging platform (see <http://www.zeromq.org>)
 - End-to-end latencies of 13.4 microseconds and up to 4,100,000 messages a second have been achieved on Linux over InfiniBand
 - Not quite that fast on OpenVMS
 - Very nice, easy-to-use API; great documentation
- Supports TCP, PGM, and SCTP
 - TCP only on OpenVMS
- Fully distributed
- Runs on numerous platforms, including OpenVMS, UNIX/Linux, and Windows
- Language bindings available for just about any language you can think of, including:
 - C, C++, Java, Python, .NET/Mono, Delphi, Ruby, PHP, Erlang, Perl, and many, many more
 - And there's an OpenVMS calling standard-compliant API's that can be used by any OpenVMS 3GL
- For details about the OpenVMS port see <http://zeromqonopenvms.blogspot.com/>
 - Versions 2.0.6 through to 2.1.4 have been ported; current version is 3.2.3 (so we're a bit behind at the moment)
 - Being used by a number of OpenVMS customers (with Pascal and C code)



30

Riak

- See <http://www.basho.com>
- Riak is a distributed NoSQL database designed to provide the following characteristics:
 - Availability
 - Riak replicates and retrieves data intelligently so it is available for read and write operations, even in failure conditions
 - Fault-tolerance
 - You can lose access to many nodes due to network partitions or hardware failure and never lose data
 - Operational simplicity
 - Additional servers can be added to a Riak cluster easily and without incurring a significant operational burden
 - Scalability
 - Riak automatically distributes data around the cluster and yields a near-linear performance increase as capacity is added
- Other features include:
 - Pluggable backend for its core shard-partitioned storage
 - Built-in MapReduce with native support for both JavaScript and Erlang
 - Multiple language drivers (including Python, Java, PHP, Ruby)



31

Riak

- We have much of Riak running on OpenVMS Integrity
 - Currently in-memory and Bitcask backends only
 - Issues with LevelDB backend primarily to do with lack of full SMP support in Erlang on OpenVMS
 - Clustering works well
 - Have also implemented a simple client API that can be used on OpenVMS from any language
 - See my 2013 talk "*Plugging HP OpenVMS into HP Public Cloud; Object Storage and Message Queuing on a Global Scale*" for additional details
- C client for Riak
 - <https://github.com/trifork/riack>
 - Easily ported
 - Uses Google Protocol Buffers-based Riak interface
 - Works well

32

Other Open Source stuff we've ported (1/10)

- libevent (<http://www.monkey.org/~provos/libevent/>)
 - An asynchronous event notification software library
 - Provides a mechanism to execute a call-back function when a specific event occurs on a file descriptor or after a timeout has been reached
 - Yes, okay, I know, OpenVMS already has such facilities...
 - Required by both memcached and gearman
- SpiderMonkey (<https://developer.mozilla.org/en/SpiderMonkey>)
 - JavaScript engine
 - Used by CouchDB and Riak (hooked into Erlang)
- FreeTDS (<http://www.freetds.org/>)
 - API's for communication with Sybase and SQL Server
 - Was actually ported to OpenVMS some years ago
 - We are working on an embedded SQL pre-processor for COBOL and a few minor enhancements...

33

Other Open Source stuff we've ported (2/10)

- Tokyo Cabinet (<http://tokyocabinet.sourceforge.net/>)
 - High-performance key/value pair database
 - Several storage options
 - Hash table
 - B+ tree
 - Fixed-length array
 - More work required (have not touched it for a while)
- YAJL (Yet Another JSON Library) (<https://github.com/lloyd/yajl>)
 - Written in C
 - Data representation independent
 - Supports stream parsing
 - Very fast
 - Tiny (small memory footprint)

34

Other Open Source stuff we've ported (3/10)

- Pillowtalk (<http://www.sevenforge.com/pillowtalk/>)
 - CouchDB C API based on libCurl and YAJS
 - A basic wrapper around libCurl and YAJS that attempts to provide a more generic interface to CouchDB
 - Mainly communicating via URL targets rather than explicit method calls
- JSON-C (<http://oss.metaparadigm.com/json-c/>)
 - A JSON implementation in C
 - Implements a reference counting object model that allows easy construction of JSON objects in C
 - Objects can then be output as strings
 - JSON-formatted strings can be parsed into JSON objects
 - Very useful when developing C API's for RESTful cloud services
- Twitter API and CLI utility for OpenVMS
 - Was working well until Twitter switched to using OAuth authentication protocol
 - Need to update code to use OAuth

35

Other Open Source stuff we've ported (4/10)

- libev (<http://software.schmorp.de/pkg/libev.html>)
 - A full-featured and high-performance event loop
 - Loosely modeled after libevent, but arguably more efficient
 - Version 4.15 ported to OpenVMS Integrity
 - Not all functionality has been tested
 - Port primarily done for jsonrpc-c (see below)
- jsonrpc-c (<https://github.com/hmnq/jsonrpc-c>)
 - Simple and efficient JSON-RPC server implementation in C
 - Receive JSON-RPC requests on TCP/IP sockets
 - Uses libev and cJSON (see <http://sourceforge.net/projects/cjson/>)

36

Other Open Source stuff we've ported (5/10)

- LevelDB (<http://code.google.com/p/leveldb/>)
 - An extremely efficient key/value storage library developed by Google
 - A backend storage option supported by Riak
 - Keys and values are arbitrary byte arrays
 - Data stored sorted by key
 - Callers can provide a custom comparison function to override default sort order
 - Multiple changes can be made in one atomic batch
 - Transient snapshots can be created to get a consistent view of data
 - Supports forward and backward iteration over the data
 - Data automatically compressed using the Snappy compression library
 - See <http://code.google.com/p/snappy/>
 - Snappy not yet ported to OpenVMS
 - Version 1.9.0 has been ported to OpenVMS Integrity as part of the Riak porting exercise
 - Some optimizations to be done
 - Test suite is able to totally swamp an old rx2660 (CPU and I/O)

37

Other Open Source stuff we've ported (6/10)

- stunnel (<https://www.stunnel.org/index.html>)
 - A TLS/SSL tunneling service
 - Can be used to provide secure encrypted connections for clients or servers that do not speak TLS or SSL natively
 - Relies on a separate library such as OpenSSL or SSLeay to implement the underlying TLS or SSL protocol
 - OpenSSL used on OpenVMS
 - Uses public-key cryptography with X.509 digital certificates to secure SSL connections
 - Clients can optionally be authenticated via a certificate
 - Version 4.52 ported to OpenVMS Integrity
- xmlrpc-c (<http://xmlrpc-c.sourceforge.net/>)
 - A lightweight RPC library based on XML and HTTP(S)
 - XML-RPC is a quick-and-easy way to make procedure calls over the Internet
 - It converts the procedure call into an XML document, sends it to a remote server using HTTP(S), and gets back the response as XML

38

Other Open Source stuff we've ported (7/10)

- GDBM (GNU dbm) (<http://www.gnu.org/software/gdbm/>)
 - An old favourite of mine
 - Library of database functions that use extensible hashing
 - Similar to the standard UNIX dbm library or Berkeley DB (BDB)
 - Facilitates creation and manipulation of a hashed database (unique key/data pairs)
 - Easily ported to OpenVMS
 - Also available as part of the Apache port to OpenVMS
 - Simple to use and can be very useful
 - I created a modified version some years back to use global sections

- LibYAML (<http://pyyaml.org/wiki/LibYAML>)
 - A YAML 1.1 parser and emitter written in C
 - YAML is a human-readable data serialization format (good for configuration files and similar such structures)
 - Version 0.1.4 (2011-05-30) ported to OpenVMS Integrity and Alpha

39

Other Open Source stuff we've ported (8/10)


- Mosquitto (<http://mosquitto.org/>)
 - An Open Source MQ Telemetry Transport (MQTT) v3.1 broker
 - MQTT provides a lightweight method of carrying out messaging using a publish/subscribe model
 - Suitable for "machine to machine" messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers
 - See <http://mqtt.org/> for more information

- Paho MQTT client library for C
 - Synchronous and asynchronous API's have been ported
 - Note that the C API requires OpenVMS 8.4 due to some CRTL dependencies (semaphores)
 - See <http://www.eclipse.org/paho/> for details on the Paho project
 - See also <http://assortedrambles.blogspot.com.au/2012/11/the-polygot-rabbit.html> for examples of how the C client can be used and how MQTT can be used in conjunction with other protocols supported by polyglot brokers such as RabbitMQ

40

Other Open Source stuff we've ported (9/10)

- GNU M4 (<http://www.gnu.org/software/m4/>)
 - An implementation of the traditional Unix m4 macro processor
 - Copies its input to the output expanding macros as it goes
 - Very powerful macro processor
 - Version 1.4.1 ported to OpenVMS
 - This version is a little old (2004); current version is 1.4.16

- Yaws (Yet Another Web Server) (<http://yaws.hyber.org/>)
 
 - A web server written in Erlang
 - Can be embedded into other Erlang-based applications or run as a regular standalone web server
 - Because it uses Erlang's light-weight threading system, Yaws performs very well under high concurrency
 - A load test conducted in 2002 comparing Yaws and Apache found that with the hardware tested, Apache failed at 4000 concurrent connections, while Yaws continued functioning with over 80000 concurrent connections... such is the power of Erlang
 - OpenVMS port is fully functional and working well with Erlang 16A

41

Other Open Source stuff we've ported (10/10)

- OpenCOBOL (<http://www.opencobol.org/>)
 - An open-source COBOL compiler
 - Implements a substantial part of the COBOL 85 and COBOL 2002 standards
 - Also includes many extensions of other common COBOL compilers
 - Translates COBOL into C and compiles the translated code using the native C compiler
 - Works on various platforms, including Unix/Linux, Mac OS X, Microsoft Windows, ... and OpenVMS Integrity

- GMP (<http://gmplib.org/>)
 - The GNU Multiple Precision Arithmetic Library
 - A free library for arbitrary precision arithmetic
 - No practical limit to the precision except any limit as might be imposed by available memory
 - Carefully designed to be as fast as possible for both small and large operands
 - Uses highly optimised assembly code for the most common inner loops for a lot of CPUs
 - General emphasis on speed
 - Used by OpenCOBOL
 - Ported version 5.0.1; current release is 5.1.2

42

Some other useful Open Source tools

A few things from the OpenVMS freeware distribution that I often find useful...

- `sed`
 - GNU `sed` text editor for OpenVMS
 - Very useful for things like global substitutions in large numbers of files
 - Easily ported to OpenVMS Integrity
- `syslog`
 - OpenVMS port of the `syslog` daemon and logger API
 - Useful for centralised logging of application events
 - Easily ported to OpenVMS Integrity
- `Tcl`
 - Powerful and extensible scripting language
 - I use this for all sorts of things
 - Testing tools
 - Created scriptable interfaces to assorted API's
 - ...
- `flex`
 - Fast lexical analyzer/generator
 - Like the UNIX `lex` utility
- `bison`
 - Syntax parser/generator
 - GNU version of the UNIX `yacc` utility
 - Typically used in conjunction with `flex` to implement language parsers

See <http://h71000.www7.hp.com/openvms/freeware/> for details of other freeware

43

AGENDA

- Introduction
- Overview of ported pieces of Open Source software
- **General approach to porting**
- Summary and questions

Porting approach

- Various approaches are possible
 - The selected approach is to some degree a matter of personal taste
 - Often depends on size and complexity of the software being ported
- A good knowledge of OpenVMS, UNIX, C, and the CRTL is important
 - Understanding platform and CRTL differences is critical
- Patience is useful
- If at first you don't succeed, sleep on it for a while and have another go

- I tend to follow a fairly simple formula
 - Best described in the context of a case study: porting Erlang to OpenVMS...

45

Porting Erlang to OpenVMS

Basic approach:

- The basic approach was to first build Erlang on a UNIX/Linux system (Ubuntu Linux in this case) and to capture a log of the build process
 - To keep things reasonably simple, I did not (initially) go for the SMP build
 - I also deliberately excluded a few other optional components from the build that I did not want or need on OpenVMS
 - ... or components that I knew would make the port even more complex
 - These other components would be dealt with later

46

Porting Erlang to OpenVMS

Compiling and linking:

- I then took the log of the Linux build and incrementally converted this into equivalent OpenVMS DCL build commands
 - Sorting out initial compiler options
 - Suggest using `/names=(as_is,shortened)`
 - Include-paths can be fun
 - Linker commands
 - Object libraries, shareable images
 - ...
 - Note that I probably could use GNV (GNU's Not VMS), but I don't think it would have been much help here, and frankly I'd rather do things in a "pure" OpenVMS environment so as not to mask anything in any way
- At this time I also looked at the `config.h` file created during the Linux build and modified it as appropriate for OpenVMS
 - You need to have a reasonably good knowledge of the OpenVMS CRTL when doing this
- Note also that I used simple DCL command procedures to compile the code, as opposed to attempting to use `make`-like utilities such as MMS or MMK
 - This is to some degree personal preference, but also because in the initial stages of the port I wanted to be sure to recompile everything and to make quite sure that I had not missed anything
 - Erlang is a substantial (and non-trivial) code base

47

Porting Erlang to OpenVMS

Adjusting for OpenVMS specifics:

- Once an initial (partial) build procedure was set up, I systematically started working through trying to compile up the code base, fixing compiler errors, sorting out include paths, eliminating warnings, and so on
 - At this point most of the trivial porting issues come into play
 - Differences in header files
 - Missing CRTL functions
 - Choice of compiler flags (qualifiers)
 - ...
 - It's a case of working through these issues and addressing them as appropriate
 - Having done such things a few times in the past (and wishing I'd made notes), it did not take too long to at least get most of the code to compile and link
 - But this is just the start of the battle!
- As a next pass, I searched through the code for known issues such as trying to use `fcntl()` to set sockets to non-blocking, and incrementally addressed these
 - Getting pipes to work correctly and dealing with `fork()/exec()` sequences required a little more thought (and luck)

48

Porting Erlang to OpenVMS

Adapting to the OpenVMS file system:

- The last hurdles to overcome in getting Erlang working were file-system related
 - It was reasonably easy to determine where the problems were in the C code
 - But when problems were at the Erlang level, they took a little more time to resolve (as at this time I was not particularly familiar with Erlang)

Making OpenVMS “look” like UNIX:

- In order to avoid having to make excessive changes to any of the Erlang (`erl`) run-time libraries, I “made” OpenVMS look like a variant of UNIX
 - This means that wherever the Erlang checks the operating system type it thinks it’s a variant of UNIX
 - And it doesn’t much care (except in a small number of cases) what the specific variant is

```
$ erl
Eshell V5.10 (abort with ^Z)
1> os:type().
{unix,openvms}
2>
```

49

Porting Erlang to OpenVMS

Testing:

- Most of the testing of the port was performed using RabbitMQ, which was quite easy to port once I had Erlang working (RabbitMQ is 100% Erlang code)
- RabbitMQ ships with a number of comprehensive test suites, some of which are written in Java
 - The Java test suite programs were used for much of the testing for the obvious reason that minimal porting effort was required!
- By using the RabbitMQ Java test suite, it was possible to uncover a few items requiring attention
 - The two most prevalent being some subtle file system-related issues and the method by which the main Erlang process communicates with its daemons (pipes/OpenVMS mailboxes)
- [Most good Open Source software packages typically include test programs, unit tests, example code, and so on](#)

50

AGENDA

- Introduction
- Overview of ported pieces of Open Source software
- General approach to porting
- **Summary and questions**

Summary and future plans

- OpenVMS is an Open Systems platform
- There is demand for Open Source solutions on OpenVMS
- There is plenty of work still to be done

Thank you

