Archaeology for the Future - Ada on OpenVMS

Gérard Calliet

pia-sofer, vms_ada_alliance; 66 rue de la mascotte 17190 Saint Georges d'Oléron - Cheray ; France ;Tel: +33 (0) 670 716 179; email: contact@vmsadaall.org

Abstract

This paper presents a project to make Ada available on all VMS platforms. The project is put in perspective within the broader framework of sustainable development cycles. We elaborate a proposal to revisit upward compatibility as a mode of temporal portability. The methods and tools used help to rethink archaeology as an image of intelligence in programming activity. The paper also highlights the similarities between the destinies of the Ada and VMS ecosystems.

Keywords: Ada, VMS, OpenVMS, LLVM, Gnat-llvm, backward compatibility, portability, software conception

1 A time trap

In 2015, the author of this article found himself in a time trap. After a very long gestation period, the decision was taken to relaunch (with Ada) an urban transport control system on the same Operating System where it had been created (1993): OpenVMS.

But the decision, matured in 2013 and 2014 on the basis of support for Gnat Ada on OpenVMS (Itanium), came in 2015, the year in which AdaCore decided to stop supporting Gnat Ada on OpenVMS (Itanium). In the midst of a reinvestment drive, the weight of the past came to the fore.

So, we had to go back to basics and rebuild Gnat Ada for Itanium from the FSF source code.

2 Why OpenVMS?

You might wonder why we chose to go back to OpenVMS, an OS that is a priori on the way to obsolescence.

OpenVMS is an OS of original conception, founded by people from MIT. Similar to Ada: there are times in computer science when you can rationally synthesize a set of problems.

This specific OS was chosen for its qualities for demanding applications. For certain environments, it has stood the test of time. One essential quality that has made this possible: backwards compatibility. You can write an application for VAX, and it will live on Alpha; then on Itanium, and soon on x86 and the cloud, without any major transformation.

In 2013 OpenVMS was also caught in a time trap. HP decided to stop supporting it. But the force of the past meant that OpenVMS was taken over by a start-up, VMS Software Incorporated (VSI) [1] - which has now ported it to x86.

Collaborating with AdaLabs company, we succeeded in our gamble in 2016 [2].

The dual success that allowed us to avoid both time traps gave us food for thought. We needed to go further in terms of application, and we revisited the methodological elements that emerged during our work.

3 Ada and VMS through ages

3.1 Introducing the project

The history of Ada has had its twists and turns. It seems that Ada and OpenVMS share the ability to risk oblivion and rise from the ashes.

Observing these similarities in their destinies has led us to consolidate what was initially just a sketch. The current project is to offer Ada on OpenVMS across all the ages of VMS: VAX, Alpha, Itanium, x86 (all environments still 'up and running').

The dual success that allowed us to avoid both time traps gave us food for thought.

We are thus consolidating Ada and VMS and exploring a new technical and economic field. This new field is linked to the existence of new emergencies named under the concepts of sustainable development, sustainability, energy efficiency, and cyclic economics. This is a universal trend that has perhaps not yet been widely explored in the field of IT development.

We present the different stages of making a universal availability of Ada on VMS, whether already existing or implemented, awaiting refinement, or still in the early stages of development.

3.2 Vocabulary

To facilitate the rest of the presentation, let's review the concepts, events, systems, and tools involved.

VAX stands for Virtual Address eXtension. This is a 32-bit CISC architecture created by DEC in the mid-1970s. It gave rise to a range of computers that became widely distributed

worldwide. Some early Unix systems were developed on VAX systems.

DEC stands for Digital Equipment Corporation, co-founded by Ken Olsen. It was created with the idea of opening up wider access to computing resources by launching lines of minicomputers, PDP (16-bit), then VAX (32-bit).

VMS stands for Virtual Memory System. This is the operating system most often associated with VAX machines, to the point that the common name is VAX/VMS, which expresses the compatibility of hardware and software.

Alpha is the name of the new hardware architecture created in the early 1990s by DEC. It is a 64-bit RISC architecture. The VMS operating system was ported to Alpha. The name VMS was later changed to OpenVMS. This remains the official name for VMS to this day. However, as a paradigm of a certain computer culture, the name VMS is most commonly used.

Itanium is the name of a 64-bit EPIC (Explicit Parallel Instruction Computing) architecture created by several manufacturers, including DEC, Bull, and Intel, in the early 2000s. This architecture was supposed to revolutionize the field of hardware architecture, but it failed to achieve real commercial success.

After the commercial failure of DEC, which was acquired by Compaq and then by HP, VMS gradually lost significant market share. In 2013, HP decided to end its support.

VSI stands for VMS Software Inc. This startup was formed to acquire the rights to OpenVMS and has maintained the operating system ever since. Its major investment was to port VMS to the standard x86 architecture.

x86 is the name of an architecture that has survived the ages and is now widely used.

GNAT Ada is the name associated with the open-source Ada compiler and all the compiler-related components. It is based, at least initially, on the gcc project (GNU Compiler Collection). At the request of the US Air Force, this open-source compiler was developed with New York University. It is now supported by AdaCore. The GNAT Ada sources are deposited in the FSF (Free Software Foundation) repository.

gcc is an open-source project housing all compiler generation tools, front-end and back-end, multi-language and multi-target platforms. The project was initiated in 1987 by Richard Stallman and is maintained by a very large community.

LLVM initially stood for Low Level Virtual Machine. It is a project similar in its goals to gcc, initiated in 2000 at the University of Illinois. The notion of a virtual machine was confusing, and LLVM became an orphan initialism. The composition of the front-end, the back-end and the nature of Intermediate Representation (IR) have been rethought. The project is also open source, but under licensing modes considered more open for industrial uses.

Debian is a major distribution of the Linux operating system, the basis for other Linux distributions. Debian versions are given names, for example, "Trixie" for version 13, which is a recent version, and "Wheezy" for version 7, released in 2013.

SIMH is the name of a family of emulators for older hardware, including VAX. Initiated by Bob Subnick in 1993 to preserve the legacy of minicomputers. The SIMH community is still very active.

4 Steps

4.1 Keeping DEC Ada alive for VAX and Alpha

For VAX and Alpha, the only effort is to preserve what already exists. We have the excellent DEC Ada for OpenVMS [3], renowned in particular for its debugging.

The compiler documentation is extremely valuable, including an LRM (Language Reference Manual) that includes the universal LRM and is annotated with implementation details for VMS. The compiler only offers version 83 of Ada.

There are still applications developed in DEC Ada that run on VAX and Alpha environments. I still give ADA / VAX-VMS training courses for teams who have to maintain their platform until at least 2035. A recollection of these uses must be done.

The VAX/VMS and OpenVMS worlds on Alpha do not have the same industrial contexts. When VMS was taken over by VSI, only the rights for VMS on Alpha and VMS on Itanium, and future rights on other platforms, were acquired by VSI. As a result, VMS on Alpha and Itanium, as well as products such as the DEC Ada compiler, are maintained by VSI. But the VAX/VMS system is, in a sense, an orphan, with no maintenance from HP, the current owner.

So, DEC Ada on VAX is somehow orphan, whereas DEC Ada on Alpha is still supported by VSI.

For the actual hardware context, both VAX/VMS and VMS on Alpha currently still run on VAX or Alpha hardware, as well as on commercial or open-source emulation solutions. The open source SIMH solution is very often used for VAX environments.

A special effort must be made to maintain the archives for VAX. Some efforts have to be made to register the operating system sources as part of the Software Heritage project [4]. This type of operation requires negotiations with the current owner of the rights to VAX/VMS. Another project could be to rebuild from these sources, following the procedures used by DEC, the latest version of VAX/VMS, version 7.3, as well as products such as DEC Ada, to have a searchable reference version.

For DEC Ada on VAX and DEC Ada on Alpha, the effort essentially focuses on managing the reference archives. In this context, communities like the one surrounding SIMH are essential reinforcements.

4.2 Restarting GNAT ADA on OpenVMS Itanium

The Itanium adventure is more difficult. HP had abandoned Ada support to AdaCore, transition from DEC Ada to GNAT Ada on VMS/Itanium. And AdaCore stopped all support for Itanium. One of the reasons for this was the switch from gcc 4.7 to gcc 5, with the appearance of C++ in the tool chain. For AdaCore, the adaptation work for OpenVMS was not considered profitable, and we were in a period of disintegration of OpenVMS support by HP.

For the purposes of our application, we didn't need the latest GNAT version. The last (AdaCore) GNAT Ada version for OpenVMS Itanium offered Ada version 83 and Ada version 95. The applications we were porting to Itanium needed only version 83, with some Ada 95-like additions that had been offered by DEC Ada alpha. We rebuilt the latest GNAT Ada version for OpenVMS Itanium, which is based on gcc version 4.7. We rebuilt it from FSF sources.

Archaeology: gcc 4.7 was no longer really supported by the gcc community. And we had to find the build procedures used by AdaCore ourselves. In this kind of situation, the engineer is an archaeologist: reconstructing the procedures from the traces, looking for existing repositories, adapting to older versions.

Our build is available on github [5]. We had to use the wheezy version of Linux, accessed via schroot. Schroot allows the use of another version of Linux on a Linux host. To rebuild a gcc version 4.7, you must be in the same conditions as its initial build, namely Debian 7, named wheezy. Again, this requires archive work, because the wheezy repositories are now in the archive domain. It is also necessary to find all the necessary components in their version consistent with gcc 4.7.

The build procedure is of the Canadian type. This procedure has three phases: creation of a native gcc 4.7 compiler, that is, one that runs under Linux (Debian 7 wheezy); creation with this native gcc 4.7 of a gcc 4.7 cross-compiler that creates VMS binaries by working on Linux; and finally, use of this compiler to create a compiler executable on VMS producing VMS executables. This type of approach is one of the possibilities offered by gcc.

It is important to note that the operations are performed with major use of the Linux make tool, part of the Debian tool chain, and these tools are themselves versioned and relative to the Debian version used. In our approach, it is therefore necessary to choose a tool chain consistent with what we are rebuilding. It must be chosen and, if necessary, found in archives.

We were thus able to reconstruct the latest version of GNAT Ada existing on Itanium, based on version 4.7 of gcc.

The underlying gcc 4.7, on top of which the GNAT Ada compiler exists, could be extended and used as a C compiler (gcc-like) under OpenVMS Itanium. This compiler could itself be used to facilitate ports of open-source projects to OpenVMS Itanium of a similar era. The gcc C idiom is probably closer to what is used in open-source projects than the DEC C idiom existing under OpenVMS. This type of extension, which goes beyond the Ada domain, demonstrates the existence of additional benefits of our archaeological approach.

The next step for Itanium is to take the first step towards gcc 5. We'll have to reinvent what AdaCore could have done then. With the same problem of the extreme age of gcc 5. The industrial interest of this operation is reduced, since the lifespan of Itanium platforms is itself reduced, and new development on Itanium with more recent GNAT Ada versions seems unlikely. However, the scientific effort of redevelopment using archaeological methods is interesting.

4.3 Move to x86

The current phase, closer to industrial opportunities, is the move to x86. Formally, the switch is entirely feasible. VSI has chosen the LLVM tool chain for its compilations of the OS and its environment.

Here too, it was a Canadian-style strategy that allowed VSI to create VMS compilers on x86[6]. VSI began its journey by cross-compiling from OpenVMS Itanium to x86. In the first phase, it was only possible to refer to version LLVM 3.4.2, due to the antiquity of the C and C++ compilers present on OpenVMS Itanium. In 2025, VSI implemented, using Canadian-style procedures, an LLVM and Clang environment in version 10.0.1 on VMS x86. Very soon, using these tools, it will be possible to natively generate a more recent LLVM environment, with the associated tool chain tools, essentially a compatible version of CMake.

Furthermore, AdaCore has initiated a GNAT LLVM project that allows the construction of the GNAT Ada compiler with an LLVM underlying framework. The GNAT LLVM project [7] and the OpenVMS LLVM tool chain therefore need to be brought together.

Here again, archaeology. For reasons of feasibility, VSI has chosen older versions of LLVM as the initial basis of its effort and will migrate at its own pace to more recent LLVM versions.

While waiting for recent versions of LLVM to be made available, we need to search the GNAT LLVM project for dependencies on recent versions of LLVM and find workarounds. This implies tracing (archaeology again) the methods used to adapt GNAT to LLVM. Initially, we are aiming for cross-compilation for execution on OpenVMS.

4.4 Learned by doing, elements of the method

A first characteristic of the method is the great importance attached to archives. It is absolutely necessary to be able to locate all sources for a given period, and to be able to determine which versions of the different elements can be used together.

In this effort, the relationship with the communities likely to know and maintain the archives is crucial. It is likely that the approach even requires reactivating some communities that have previously lain dormant.

The situation in this type of project seems to be a few steps behind what is generally available. It would be a mistake to try to immediately make up for this delay. It would be a never-ending race. On the contrary, we must accept taking several steps back and equip ourselves with the means to analyze the reasons for the transitions.

The step back consists of understanding the dynamics of the transitions, their necessity, and what, in the change, has been preserved as essential. The result of this type of research converges towards a rediscovery of the fundamentals at play appearing in the dynamics of change. Once these analyses are completed, we can assess what can be reconstructed, or how we can circumvent problems that cannot be resolved as they stand.

The essence of the approach can be seen as a rebound. A search in the past for initial meanings, and a rethinking based on these initial meanings. Here we find the meaning and etymology of the notion of archaeology, a rediscovered logic of the principle, arche and logos.

Why is this archaeology particularly possible with VMS and Ada? Because both VMS and Ada were themselves conceived as a moment of synthesis, that is, a rebound from the fundamentals.

5 Towards temporal portability and its communities

Our approach, with its archaeological method, opens up another conception of the temporal organization of IT developments. Time is much more than a headlong rush. What we are working on with Ada on VMS is a deepening of what has until now been called backward compatibility.

The designers of OpenVMS were clearly aware of the need for backward compatibility. They associated this with all the qualities of robustness and a coherent, rational approach. They were not alone in guaranteeing this. Then this quality generally faded, giving way to LTS or assurance, which turned an intrinsic quality into a luxury.

We all know that the flight to the ever-new, with no regard for the old, is reaching its energetic and material limits. What is being developed throughout industry in terms of reusability, the circular economy and recycling is bound to affect the IT world in the long term. The concept of ‘temporal portability’ is therefore going to take hold in IT.

This can be particularly relevant with Ada. The issue of portability has always been very important in Ada. Until now, it has been primarily designed according to the notion of space (from platform to platform). We are probably in a time where portability according to the notion of time is becoming important.

6 More than just economic issues

The project is also driven by theoretical consideration. No foundation is possible without consideration of the fundamentals. Scientific thinking is akin to archaeology: looking for patterns in the traces of reality or previous theories. From this point of view, a project like ours has the potential to attract the interest of the intellectual elite.

The birth of Ada can be seen - like that of OpenVMS - as a moment of synthesis, a coherent re-elaboration of issues that have emerged over time. It's like a pause, a look back, then a rebound. It is this type of approach that is once again possible when we compare the life cycles of Ada and OpenVMS.

From this point of view, our project is a revival of this kind of gesture, a precursor of new ways of looking at IT life cycles.

7 Join the project

All specific skills in OpenVMS and tool chain management, knowledge of older versions of gcc, knowledge of the GNAT LLVM project, knowledge of LLVM versions, Ada theorists, etc. are welcome.

Those who join us have a taste for researching archives and historical witnesses to developments. They are decipherers of the treasures of ingenuity of which the programs are traces. They know how to reinvent the future from the past.

References

[1] New company for OpenVMS: VMS Software Incorporated: www.vmssoftware.com

[2] Our version of Gnat Ada for OpenVMS / Itanium: www.vmsadaall.org

[3] DECAda: https://docs.vmssoftware.com/docs/ada_lrm.pdf

[4] Software Heritage: https://softwareheritage.org.

[5] D. Sauvage, G. Calliet on github.com: https://github.com/AdaLabs/gnat-vms

[6] J. Reagan, journey porting OpenVMS compilers to x86: https://llvm.org/devmtg/2017-10/slides/Reagan-Porting%20OpenVMS%20Using%20LLVM.pdf and https://www.youtube.com/watch?v=mClf5Q7Aw6c

[7] Gnat Ada / LLVM project (community): https://github.com/AdaCore/gnat-llvm