OpenVMS Technical Journal V17

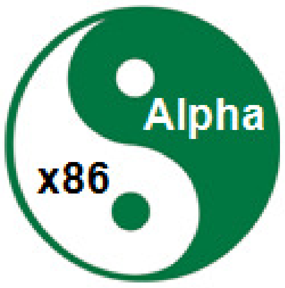
Running OpenVMS native on x86 . . . without rewriting OpenVMS

Arie de Groot, AVTware.
arie@avtware.com

Introduction

Running OpenVMS native on x86
. . . without rewriting OpenVMS.

That is the goal we set ourselves when we started to design a next-generation Alpha virtualization solution. Our objective was to build a thin layer between the x86 computer and the Alpha/OpenVMS architecture, providing just enough buffer to soften the sharp edges at both sides that prevent these two worlds from working together.



And we wanted our solution to serve both the OpenVMS and Tru64 communities.

The best way to achieve that goal is to offer these two operating systems the same Alpha hardware environment they expect to see. At the same time, we did not want to go down the road other Alpha virtualization developers took. They use a separate host operating system to run their product on, which introduced problems and concerns.

Our solution has to run directly on the bare x86 hardware: Just the computer hardware, no pre-installed operating system required. It is an x86-based Alpha. This is a unique proposition, something that differentiates us from other solutions.

Additionally it has to run on virtual machines, irrespective of the type or brand of the virtual machine. Virtual Machines are increasingly becoming the IT infrastructure of choice with our potential users, therefore our solution has to support that IT strategy.

The result is vtAlpha, a virtualization of the Alpha hardware environment that runs OpenVMS and Tru64 on 'bare metal' x86 hardware.

The new playing field

Because virtualization has proven itself as a functional and financially attractive option for restructuring modern data centers, IT managers typically respond positively when queried about its utilization. However, when Alpha virtualization is brought up, special factors are frequently raised. Most of these issues boil down to the fact that to date all available Alpha virtualizations run on Windows or sometimes a flavor of Linux. A quick summary of the top concerns includes:

1. Important Alpha applications are exposed to the vulnerability of the foreign host operating system.
2. Current virtual Alpha solutions are available on a limited selection of host OS options.
3. OS level upgrades (often automated) and service packs (especially frequent with MS Windows) increase the potential for version incompatibilities and instability.
4. User access to the host OS may compromise Alpha security.
5. Internal ownership of the virtual Alpha may switch departments once it runs on Windows or Linux.

Vulnerability is the most heard argument and a hot issue for the OpenVMS and Tru64 users who are used to the unhackable status of their current platforms. The idea of running these systems on top of Windows often scares these users (a Linux platform seems to be less of an issue in this area).

Currently available virtual Alphas ride on top of a host operating system. So far mainly Windows, but recently some vendors offer a couple of Linux flavors also.

The consequence is that only a limited number of host operating system types or versions are supported by the current virtual Alpha vendors, which may not match with the IT-infrastructure in your company.

Changes made to the host operating system are outside the control of the virtual Alpha software. Therefore the frequent bug-fix or service pack updates of these host operating systems sometimes cause compatibility problems with the execution of the virtual Alpha, resulting in unscheduled down-time for the virtualized Alpha system.

Having to reboot the host computer to activate the updates applied to it is also undesirable in the (often mission-critical) Alpha world.

Running the virtual Alpha on top of a relatively common operating system unlocks the gates to users who run other software or activities in parallel with virtual Alpha. This without the Alpha system administrator having any control over that extra software.

And there is sometimes the question of who reigns over your (virtual) Alpha now that it has become a Windows or Linux "application", causing you to lose control over your critical environment.

All these arguments bring us to this simple question: "Why don't we turn an x86 machine into an Alpha, so we can go on treating it as an Alpha?"

Good question. Why not? It has never been done, mainly because it is difficult.

So that was our challenge.

Bare Metal Approach

Turning an x86 into an Alpha requires expertise in the x86 host hardware architecture and its internals as well as in the guest environment (Alpha) you want to host. Fortunately AVTware has both types of engineering expertise combined in its team.

Our approach was to split the work into a host-oriented part, focusing on the creation of a kernel operating system that deals with the x86 side (CPU, memory, system busses, registers, etc.) plus driver support for existing and future peripheral equipment.

The other task was to create the Alpha architecture on top of that, adding functionality and support for all Alpha models and (most) peripherals that have been manufactured by Digital / Compaq / HP. This part also requires in-depth knowledge in the guest operating systems (OpenVMS, Tru64) that will run on top of the virtual Alpha.

This combination has to run directly on the x86-platform in all its variations (Intel Xeon or i7, AMD Opteron or Phenom, PCI-bus, storage, graphics, network, serial lines, in short: everything).

Our vtAlpha solution layers directly on top of the x86 foundation, with an Alpha look and feel at the topside so that your Alpha software can connect to that environment same as always. Configuration capabilities give you the ability to model the Alpha top-side into exactly the Alpha computer you intend to replace. We refer to this as the Bare Metal Installation since it requires no pre-installed operating system underneath.

Having no regular operating system underneath vtAlpha, we needed to build tools that help the system managers to shape and manage the virtual Alphas they have under their control, as well as to manage the vtAlpha host environment (vtMonitor tool).

vtAlpha does not depend on an external host operating system, thereby avoiding compatibility and vulnerability issues with that environment. It will also remove internal disputes about if and how to make updates to the host environment and about who has ultimate ownership over the virtual Alpha.

In short, you can simply avoid paying the cost of and dealing with the performance aspects of such an individual OS.

Consolidating multiple virtual Alphas in one x86 host

vtAlpha creates a virtual Alpha environment inside the x86 host computer that can run multiple Alpha configurations in parallel. These parallel virtual Alpha models can be of different kinds (Alpha system type, number of CPUs, memory size, etc.) as long as the combined resource demand does not supersede the available host resources.

As a rule of thumb every virtual Alpha demands 1.5 x the number of virtual Alpha CPUs. The smart resource allocation mechanism in vtAlpha allows the use of hyper-threading which increases the number of CPUs you can work with. Per virtual Alpha, you will need to allocate 1.5 GB of host memory PLUS the amount of Alpha memory you require.

So a basic host system with 4 CPU cores and 8 GB of memory can host 2 single-CPU virtual Alphas, each with up to 2.5 GB memory. When hyper-threading is switched on, it can host virtual Alphas with up to 4 Alpha CPUs using the "hyper-threaded" host CPUs for the required 50% extra processing capacity.

Example 1:

Replacing a 4 CPU AlphaServer 4100 with a modern multi-core host system already means a significant downsizing of the system footprint.

Imagine combining 3 of these systems in, for example, a HP Proliant DL360. 1U height, 12 CPU cores in total, with an additional 12 hyper-threading CPUs for the required extra capacity, connected to a SAN or using the ~7 TB storage capacity in the Proliant.

Example 2:

A HP Proliant DL980 can deliver up to 40 CPU cores, which would leave room for 10 ES40s, or 20 DS20s, or 30-40 Single-CPU Alphas, or any combination of the above.

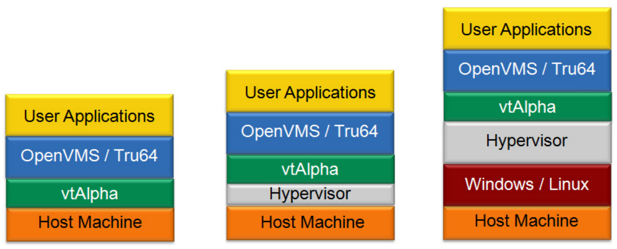
The result: consolidation and cost savings on a large scale.

Virtual Machine Support

Virtual Machines are rapidly becoming the preferred IT infrastructure of choice within medium and large size companies. So the vtAlpha solution has to fit well into that eco-system.

When the Virtual Machine product selected by the customer offers a regular x86-environment, vtAlpha will run on that virtual machine, just like it does on a physical machine.

vtAlpha is already compatible with the most common Virtual Machine products like VMware, Xen, KVM, etc. Others will likely work too, but have not yet been verified.



Bare Metal offers the best virtual Alpha performance, with Virtual Machines use thin layer hypervisors.

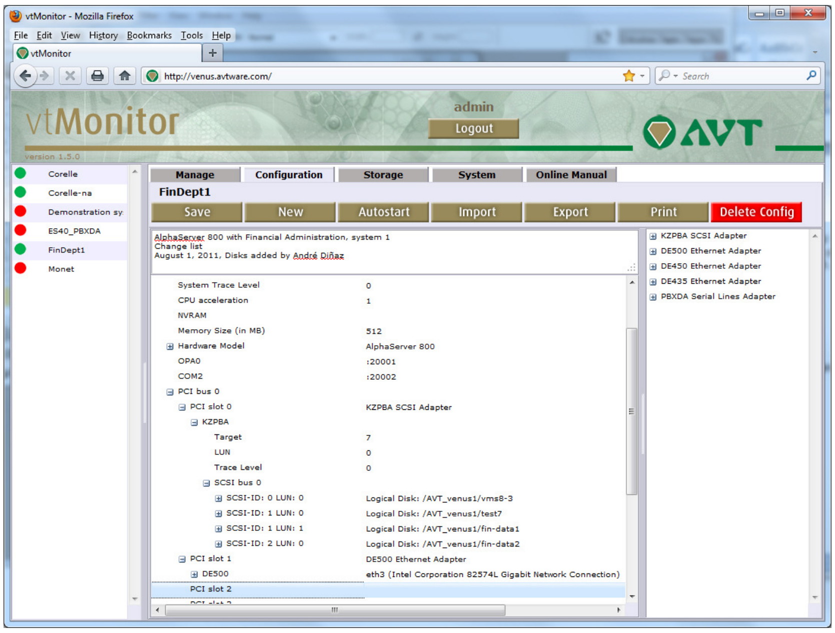
It is important to consider the following:

1. The Bare Metal approach (vtAlpha as a thin layer directly upon the x86 host hardware) offers the best performance results for your virtual Alphas.
2. Adding underlying virtual machines to that combination reduces performance, since the virtual machine layer claims some of the host CPU capacity.
3. Some x86 virtual machine versions require a separate host operating system underneath (Windows/Linux) which in itself will demand another piece of the CPU capacity, further negatively influencing the performance of the virtual Alpha.

vtMonitor

vtAlpha includes a management tool for your multi-Alpha capable environment, called vtMonitor. It is web-browser based and can be run from any location in your network that has access to the vtAlpha host.

vtMonitor allows you to setup virtual Alpha configurations for the Alpha systems you want to replace, start and stop these virtual Alphas, monitor their status and perform vtAlpha host environment settings.



```
AlphaServer 800 with Financial Administration, system 1
Change list
August 1, 2011, Disks added by André Diñaz
```

```
System Trace Level                  0
CPU acceleration                    1
NVRAM
Memory Size (in MB)                 512
⊞ Hardware Model                    AlphaServer 800
OPA0                                :20001
COM2                                :20002
⊟ PCI bus 0
  ⊟ PCI slot 0                      KZPBA SCSI Adapter
    ⊟ KZPBA
        Target                      7
        LUN                         0
        Trace Level                 0
      ⊟ SCSI bus 0
        ⊞ SCSI-ID: 0 LUN: 0         Logical Disk: /AVT_venus1/vms8-3
        ⊞ SCSI-ID: 1 LUN: 0         Logical Disk: /AVT_venus1/test7
        ⊞ SCSI-ID: 1 LUN: 1         Logical Disk: /AVT_venus1/fin-data1
        ⊞ SCSI-ID: 2 LUN: 0         Logical Disk: /AVT_venus1/fin-data2
  ⊟ PCI slot 1                      DE500 Ethernet Adapter
    ⊞ DE500                         eth3 (Intel Corporation 82574L Gigabit Network Connection)
  PCI slot 2
```

vtMonitor screen example

Summary

- Installing the virtual Alpha environment directly onto the x86 host avoids a lot of problems that current virtual Alpha installations are experiencing, like vulnerability and conflicts with the host operating system.
- vtAlpha is a thin layer on top of the x86 architecture that transforms it into and Alpha system interface and allows running OpenVMS and Tru64 on that x86 host.
- Physical and Virtual Machines are supported, as long as they are x86 compatible.
- vtAlpha supports consolidating multiple virtual Alphas inside one host, increasing the ROI.

Find more information about vtAlpha on the websites: http://www.avtware.com and http://vax-alpha-emulation.com , or send email to: arie@avtware.com

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304
www.hp.com