VM System Specification for ARM Processors

Christoffer Dall, Virtualization Tech Lead
Version 1.0, Last Revised: August 4th, 2014

This spec can also be accessed in a clear-text version.

Table of Contents

Goal

The goal of this spec is to allow suitably-built OS images to run on all virtualization solutions for ARM processors, such as KVM or Xen.

Recommendations in this spec are valid for AArch32 and AArch64 alike, and they aim to be hypervisor agnostic.

Note that simply adhering to the ARM Server Base System Architecture (SBSA) [1] is not a valid approach, for example because the SBSA mandates EL2, which will not be available for VMs. Further, this spec also covers the AArch32 execution mode, not covered in the SBSA.

VM System Specification for ARM Processors

Christoffer Dall, Virtualization Tech Lead
Version 1.0, Last Revised: August 4th, 2014

Image Format

The image format, as presented to the VM, needs to be well-defined in order for prepared disk images to be bootable across various virtualization implementations.

The raw disk format as presented to the VM must be partitioned with a GUID Partition Table (GPT). The bootable software must be placed in the EFI System Partition (ESP), using the UEFI removable media path, and must be an EFI application complying to the UEFI Specification 2.4 Revision A [2].

The ESP partition's GPT entry's partition type GUID must be C12A7328-F81F-11D2-BA4B-00A0C93EC93B and the file system must be formatted as FAT32/vfat as per Section 12.3.1.1 in [2].

The removable media path is \EFI\BOOT\BOOTARM.EFI for the AArch32 execution state and is \EFI\BOOT\BOOTAA64.EFI for the AArch64 execution state as specified in Section 3.3 (3.3 (Boot Option Variables Default Boot Behavior) and 3.4.1.1 (Removable Media Boot Behavior) in [2].

This ensures that tools for both Xen and KVM can load a binary UEFI firmware which can read and boot the EFI application in the disk image.

A typical scenario will be GRUB2 packaged as an EFI application, which mounts the system boot partition and boots Linux.

Virtual Firmware

The VM system must be UEFI compliant in order to be able to boot the EFI application in the ESP. It is recommended that this is achieved by loading a UEFI binary as the first software executed by the VM, which then executes the EFI application. The UEFI implementation should be compliant with UEFI Specification 2.4 Revision A [2] or later.

This document strongly recommends that the VM implementation supports persistent environment storage for virtual firmware implementation in order to ensure probable use cases such as adding additional disk images to a VM or running installers to perform upgrades.

This document strongly recommends that VM implementations implement persistent variable storage for their UEFI implementation. Persistent variable storage shall be a property of a VM instance, but shall not be stored as part of a portable disk image. Portable disk images shall conform to the UEFI removable disk requirements from the UEFI spec and cannot rely on on a pre-configured UEFI environment.

The binary UEFI firmware implementation should not be distributed as part of the VM image, but is specific to the VM implementation.

Note that to comply with the UEFI specification mentioned above, the EFI implementation must support the UEFI RTC for real time clock services. To provide this API, the VM system will likely need to implement a real time clock device, but the implementation details of such a device are outside the scope of this spec and private between the VM system and its associated UEFI implementation.

VM System Specification for ARM Processors

Christoffer Dall, Virtualization Tech Lead
Version 1.0, Last Revised: August 4th, 2014

Hardware Description

The VM system must be UEFI compliant and therefore the UEFI system table will provide a means to access hardware description data.

The VM implementation must provide through its UEFI implementation:

A complete FDT which describes the entire VM system and will boot mainline kernels driven by device tree alone.

For more information about the arm and arm64 boot conventions, see Documentation/arm/Booting and Documentation/arm64/booting.txt in the Linux kernel source tree.

For more information about UEFI booting, see [3] and [4].

VM Platform

The specification does not mandate any specific memory map. The guest OS must be able to enumerate all processing elements, devices, and memory through HW description data (FDT) or a bus-specific mechanism such as PCI.

If AArch64 physical CPUs implement support for the AArch32 execution state in EL1 and EL0 execution, it is recommended that the VM implementation supports booting the VM at EL1 in both AArch32 and AArch64 execution states.

The virtual hardware platform must provide a number of mandatory peripherals:

- Serial console: The platform should provide a console, based on an emulated pl011, a virtio-console, or a Xen PV console.
- An ARM Generic Interrupt Controller v2 (GICv2) [5] or newer: GICv2 limits the the number of virtual CPUs to 8 cores, newer GIC versions removes this limitation.
- The ARM virtual timer and counter should be available to the VM as per the ARM Generic Timers specification in the ARM ARM [6].

It is strongly recommended that the VM implementation provides a hotpluggable bus to support hotplug of at least block and network devices. Suitable buses include a virtual PCIe bus and the Xen PV bus.

For the VM image to be compliant with this spec, the following applies for the guest OS in the VM image:

- The guest OS must include support for pl011 UART, virtio-console, and the Xen PV console.
- The guest OS must include support for GICv2 and any available newer version of the GIC architecture to maintain compatibility with older VM implementations.
- The guest OS must rely on the UEFI RTC API for real time clock services.
- It is strongly recommended to include support for all available (block, network, console, balloon) virtio-pci, virtio-mmio, and Xen PV drivers in the guest OS kernel or initial ramdisk.

Other common peripherals for block devices, networking, and more can (and typically will) be provided, but OS software written and compiled to run on VMs for ARM processors cannot make any assumptions about which variations of these should exist or which implementation they use (e.g. VirtIO or Xen PV). See Hardware Description above.

VM System Specification for ARM Processors

Christoffer Dall, Virtualization Tech Lead
Version 1.0, Last Revised: August 4th, 2014

Changes and Previous Versions

This spec was previously referred to as the "ARM VM Image Specification". It can also be accessed in a clear-text version at [7]. The clear-text version also contains a changelog including the RFC changes.

This spec was created as the result of discussions and sessions during Linaro Connect and a public RFC sent to various relevant mailing lists. See the RFC v1 [8] and RFC v2 [9] for more information.

References

[1] ARM Server Base System Architecture
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.den0029/index.html

[2] UEFI Specification 2.4 Revision A
http://www.uefi.org/sites/default/files/resources/2_4_Errata_A.pdf

[3] http://www.secretlab.ca/archives/27

[4] https://git.linaro.org/people/leif.lindholm/linux.git/blob/refs/heads/uefi-for-upstream:/Documentation/arm/uefi.txt

[5] The ARM Generic Interrupt Controller Architecture Specifications v2.0
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.b/index.html

[6] The ARM Architecture Reference Manual, ARMv8, Issue A.b
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.b/index.html

[7] Clear-text version of this specification
http://people.linaro.org/~christoffer.dall/arm-vm-spec-v1.0.txt

[8] RFC v1 of this specification
http://lists.linaro.org/pipermail/cross-distro/2014-February/000589.html

[9] RFC v2 of this specification
http://lists.linaro.org/pipermail/cross-distro/2014-July/000731.html