VM System Specification for ARM Processors

Version 2.0, Last Revised: April 4th, 2016

Written by:
Christoffer Dall, Linaro, christoffer.dall@linaro.org

Edited by:
Charles Garcia-Tobin, ARM, charles.garcia-tobin@arm.com
David Brash, ARM, david.brash@arm.com

Table of Contents

Terminology

Virtual Machine (VM)	A virtual machine abstraction provided by software
Hypervisor	Any Type-1 or Type-2 software solution that supports running Virtual Machines. For example, Xen is a Type-1 hypervisor. KVM is a Type-2 hypervisor.
Guest OS	The operating system running inside the virtual machine

VM System Specification for ARM Processors

Version 2.0, Last Revised: April 4th, 2016

Scope

The goal of this specification is to provide a set of guidelines for both guest Operating System (OS) images and hypervisor implementations for ARM processors, such that building OS images according to the guidelines in this specification guarantees that those images can also run on hypervisors compliant with this specification.

ARM Server Architecture

This specification should be considered in the context of the ARM Server Base Boot Requirements (SBBR) [1] and the ARM Server Base System Architecture (SBSA) [2] documents. This specification strongly suggests that hypervisors for ARM processors follow the SBSA and SBBR as much as possible and makes the following additional notes:

Virtual UEFI firmware is expected to be booted in EL1 as described in the SBBR, since EL2 is reserved for the hypervisor and not available to software executing in the VM. This document relies on the SBBR to specify required ACPI data presented to VMs. Hypervisors must implement support for PSCI v0.2 [3] to be used by guests for secondary core boot and for VM shutdown and reboot.

The SBSA [2] requires support for ARMv8-A Exception level 2 (EL2), the level used for hypervisor execution. This specification supports SBSA compatible AArch64 VMs and can support compatible AArch32 VMs on SBSA platforms with the necessary ARMv8-A interprocessing support at EL1. EL1 is the exception level used for OS support, or VM execution, when under hypervisor control.

VM Image Format

A distributed OS image intended for execution in virtual machines must be in a well-defined format to be bootable across compliant hypervisors.

This specification does not mandate any specific file system distribution format such as qcow2 or compression formats, but focuses instead on the disk format presented to the VM by the hypervisor and its toolstack.

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 2.6 [4].

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 [4].

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 3.5.1.1 (Removable Media Boot Behavior) [4].

This ensures that specification compliant ARM hypervisors can rely on a UEFI firmware implementation to 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, or an OS installer packaged as an EFI application.

Virtual Firmware

The hypervisor must be UEFI compliant to run the EFI application in the ESP in VM file system images. It is recommended that this is achieved by the hypervisor loading a UEFI binary firmware implementation as the first software executed by the VM, which then executes the EFI application. The UEFI implementation should be compliant with UEFI Specification 2.6 [4] or later.

This specification strongly recommends that the VM implementation supports persistent environment storage for the virtual firmware implementation. This enables 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 specification 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 hypervisor 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 specification and private between the VM system and its associated UEFI implementation.

Hardware Description

The hypervisor must provide a UEFI compliant virtual firmware. The EFI Configuration Table (pointed to by the EFI System Table) contains pointers to hardware description data.

The virtual UEFI firmware must implement at least two EFI Configuration Table entries:

ACPI_GUID	**ACPI Pointer**
DEVICE_TREE_GUID	**FDT Pointer**
...	**...**

Both the ACPI tables and the DT tables must fully describe the entire VM and its peripherals.

This allows, for example, booting a mainline Linux using either ACPI or device tree.

For more information about the Linux arm and arm64 boot conventions, see Documentation/arm/Booting and Documentation/arm64/booting.txt in the Linux kernel source tree and Documentation/arm64/arm-acpi.txt for details on using ACPI with ARM servers.

For more information about UEFI booting, see [5] and [6].

The 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 a combination of HW description data (ACPI or FDT) and 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, but optional, that the hypervisor 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 hypervisor must provide a an emulated UART meeting the minimum requirements in SBSA UART [2].
- The hypervisor must support both an ARM Generic Interrupt Controller v2 (GICv2) [7] and GICv3 [8]. Some hardware platforms are limited to supporting only GICv2 or GICv3 and hypervisors must work on both types of platforms.[1]
- The ARM virtual timer and counter system register view must be available to the VM as per the ARM Generic Timers specification in the ARM Architecture Reference Manual [9].

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

VM Images

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

- The guest OS must include a console driver for a the SBSA UART and for a pl011 UART.
- The guest OS must include support for both GICv2 and GICv3 interrupt controllers.
- The guest OS must rely on the UEFI RTC API for real time clock services.

[1] It is recommended, but optional, to support a backwards compatible virtual GICv2 on GICv3 hardware systems where the physical GICv3 implementation supports legacy operation.

- 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.

Changes and Previous Versions

Version 2.0 incorporates the following changes:

- introduced requirements to support ACPI in VMs
- outlines a closer compliance to level 2 of the SBSA
- relies on the SBBR for expanded boot requirements
- requires GICv3 over GICv2
- has been partly rewritten to clarify certain concepts

Previous versions of this specification was previously referred to as the "ARM VM Image Specification". Version 1.0 of this spec can also be accessed in a clear-text version at [10]. The clear-text version also contains a changelog including the RFC changes.

Version 1.0 of this specification 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 [11] and RFC v2 [12] for more information.

References

[1] ARM Server Base Boot Requirements
http://infocenter.arm.com/help/topic/com.arm.doc.den0044a/index.html

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

[3] ARM Power State Coordination Interface System Software
http://infocenter.arm.com/help/topic/com.arm.doc.den0022c/index.html

[4] UEFI Specification 2.6
http://www.uefi.org/sites/default/files/resources/UEFI%20Spec%202_6.pdf

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

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

[7] The ARM Generic Interrupt Controller Architecture Specifications v2.0
http://infocenter.arm.com/help/topic/com.arm.doc.ihi0048b/index.html

[8] The ARM Generic Interrupt Controller Architecture Specifications v3.0
http://infocenter.arm.com/help/topic/com.arm.doc.ihi0069b/index.html

[9] The ARM Architecture Reference Manual, ARMv8, for the ARMv8-A architecture profile
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0487a.b/index.html

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

[11] RFC v1 of version 1.0 of this specification
http://lists.linaro.org/pipermail/cross-distro/2014-February/000589.html

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