KVM-QEMU Virtualization with ARM64bit Server System

Jin-Suk Ma(✉), Hak-Young Kim, and Wan Choi

Server Platform Res Lab, ETRI,
161 Gajeong-dong, Yuseong-Gu, Daejeon, Korea
{majinsuk, h0kim, wchoi}@etri.re.kr

Abstract. In the conventional x86 or x86-64 bit system, virtualization is commonly achieved with KVM or Xen which is widely adapted in many experimental or commercial server system. At now, It is common sense that high and mid volume servers are virtualized with Xen, mid and small volume servers are virtualized with KVM. Recently, the microserver system concept was introduced, which contains low power and multicore ARM64bit sever SoCs. We present the KVM-QEMU virtualization technique and process in a real ARM64bit microserver system with ubuntu 14.04 ARM64 bit edition root file system.

Keywords: ARM64bit server SoC · KVM-QEMU · Virtualization

1 Introduction

Recently, ARM announced the ARMv8 architecture which could be used for low power and high performance server system. From ARMv7 cortex A15 architecture, ARM has supported the hardware level virtualization. ARMv8 architecture has either A57 or A53 core and both. [1] In some ARM SoC, it may have a big-little architecture with ARM57 and ARM53. ARM has licensed to APM (Applied Micro), AMD, Cavium, Broadcom and etc. One of these commercial SoC company, APM is now providing a commercial server SoC and a reference platform. The time of writing this paper, AMD is also providing the beta level test platform with NDA contract. Cavium announced the ARM64bit server SoC with 2.5 GHz clock speed, 48 ARM cores which is known to ThunderX. But it was hard to get a test level reference platform in any market.

The APM ARMv8 server SoC is called as X-Gene platform which has 4 or 8 ARMv8 cores (the code name is mustang). [2] It is widely known that APM is running the homepage webserver with X-Gene processors. As people knows that ARM architecture has been widely researched and adapted in commercial products, the microserver system which has ARMv8 server SoCs is now developing in big commercial server company such as HP, Dell and etc. We expect that the microserver will make some server product lines with not little volumes.

In view of a server software architecture, ARMv8 also supports hardware level virtualization techniques, which are basis of the conventional hardware support virtualization technique of KVM or XEN for microserver system. At current server

Y. Zhang et al. (Eds.): CloudComp 2015, LNICST 167, pp. 334–343, 2016.
DOI: 10.1007/978-3-319-38904-2_35

technology main stream, it is essential that the server must have some virtualization capability with multiple virtual machine instances. The microserver has no exception. Unfortunately, there is not much information or paper in KVM-QEMU virtualization based on the real ARM64bit SoC platform. Alex bennee showed the buildroot based QEMU virtualization in reference [5]. He did not address the information about KVM host system. Making buildroot image is not simple even if a linux expert. There are many configuration parameters alike to linux kernel .config. And also buildroot usability is not better than ubuntu root file system in view of package management and support library. Reference [6] showed many various KVM-QEMU examples. We referenced many setup techniques about QEMU parameters on this site. We have great thanks to this site operator. It has same issue like to former work also. It seemed that ARM64 QEMU work was archived in x86 or emulation environment not real ARM64bit server system base. In this paper, we archived KVM-QEMU virtualization with APM X-Gene 883208-x1 reference platform host with ubuntu root file system, which is the real hardware not ARM simulator or x86.

2 APM X-Gene Server Reference Platform

APM X-Gene server reference board is shown in Fig. 1. Target board is provided by APM (Applied Micro Co.) and it is based on ARM licensed Cortex A57 8-core. The detail hardware shape and component locations are shown in Fig. 1 and Table 1.

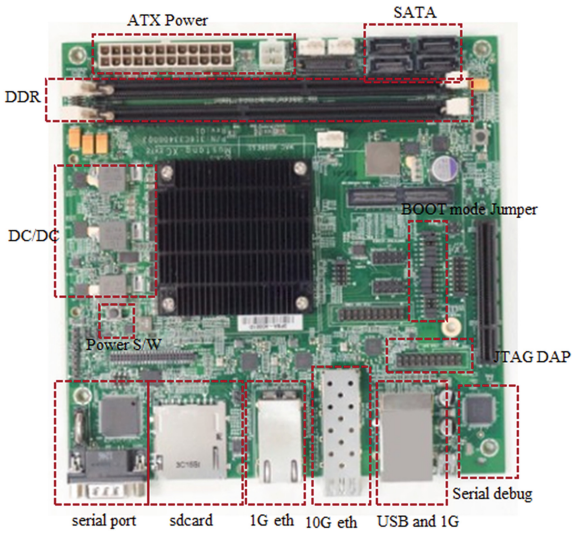

Fig. 1. APM X-Gene ARM64 server platform.

2.1 Software Development Environment

Software development environment is shown in Table 2, which is mainly supplied by APM with APM X-Gene Mustang reference platform.

Table 1. Board components and specification.

Components	Specification
Mainboard	Mustang – Applied Micro APM883208-X1-PRO-1
CPU	APM ARM64bit Cortex-A57 2.4 GHz 8Core
MEM	DRAM: ECC 16 GB @ 1600 MHz
HDD	Intel SSD 530 series 240G SATA3(for experiment)
Boot ROM	N25Q256 Serial Flash (32 MB)
Serial port	1 port
USB	2 port
Gigabit Ethernet	3 port
10G Ethernet	1 port

In this paper, we deal with the KVM-QEMU in ARM64bit server platform, so KVM version is not given in Table 1 because it is embedded in linux kernel internally. First of all, the boot disk partition is shown in Fig. 2. In Fig. 2, the system boot partition is /dev/sda1. Boot partition has kernel image (uImage) and device tree blob (apm-mustang.dtb). Kernel image is compiled from kernel source code with KVM configuration, and device tree blob is also too.

Table 2. Development software and version specifications.

Component	Version
Boot	Uboot-2013.04-mustang_sw_1.14.14
Kernel	mustang_sw-1.14.14
Root File System	AARCH64 ubuntu 14.04 LTS
Cross Compiler (GCC)	gcc-linaro-aarch64-linux-gnu-4.9 or apm-aarch64-8.0.3-le(little endian)

We will present the kernel configuration of ARM64 for KVM virtualization in Sect. 3, but the concluding result was that the complied output of uImage is about 12 MBytes and that of device tree blob(dtb) is about 26 Kbytes. So /dev/sda1 is enough disk space for housing uImage and dtb. In some case, even if you want to use ramdisk image (uInitrd), /dev/sda1 will be enough for that. /dev/sda3 is linux swap partition which is normally not included in the conventional ARM based embedded linux system. We allocated 16 Gyte of the linux swapping partition with /dev/sda3. As you know, the swap space has the double size of system RAM normally. But it is rule of thumb that the swap partition needs the almost same amount of system ram size in system with more than 8 Gbyte RAM, Actually we monitored the usage status of the

swap space in working, we can not find any swapping memory use status because of the reference platform has the large system memory (16 GBytes) which is more than any other type embedded systems with the conventional ARM core. But it is common sense that the server test environment has always the various limitations in view of many user specific use case.

/dev/sda2 contains the KVM host root file system and the required space of KVM guest. so /dev/sda2 has the whole disk spaces except for boot and swap space. In Xen virtualization case, this partition may have multiple physical or logical partitions for housing a Xen guest domain. But, we didn't find any needs of sub-partitioning on / dev/sda2 in researching this work.

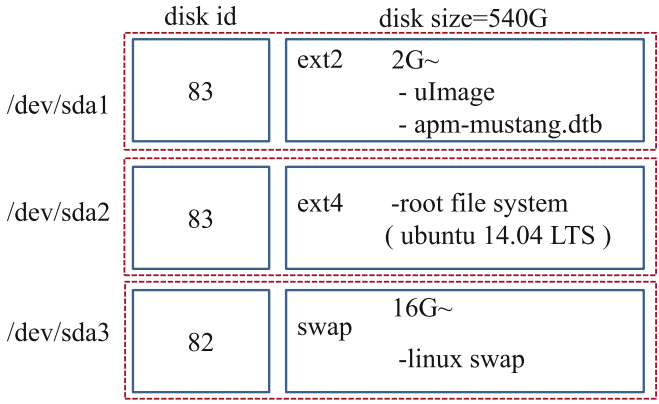

Fig. 2. Disk partition for KVM virtualization.

3 KVM-QEMU Virtualization on ARM64

3.1 KVM Kernel Configuration and Host OS

The kernel configuration for KVM virtualization in ARM64 server is essential. This is done by typing make menuconfig or manipulating .config file in linux kernel. The major kernel configuration parameters for KVM virtualization are shown in Table 3. Even thought Table 3 does not contain the whole parts of .config file due to paper space limitation, it is useful for understanding the major parts of KVM kernel configuration.

As shown in Table 3, kernel configuration parameters for KVM virtualization on ARM64 contain CPU, interrupt, IO, timer virtualization parameter setting. And also these contain the bridge network setting for host and guest OS. After we configures the linux kernel setting including Table 3 and compiles linux kernel, we could get linux kernel image (uImage) and dtb for KVM virtualization.

We select the ubuntu arm64 14.04 tar.gz style root file system for the host OS. [3, 7] Ubuntu is most widely adapted from world wide developers. Currently Canonical is quickly upgrading their public free root file system for ARMv7 (armhf) and

Table 3. Kernel parameters for KVM virtualization

Kernel parameters
CONFIG_HAVE_KVM_IRQCHIP=y
CONFIG_KVM_MMIO=y
CONFIG_VIRTUALIZATION=y
CONFIG_KVM=y
CONFIG_KVM_ARM_HOST=y
CONFIG_KVM_ARM_MAX_VCPUS=4
CONFIG_KVM_ARM_VGIC=y
CONFIG_KVM_ARM_TIMER=y
# CONFIG_AARCH32_EL0 is not set # CONFIG_ARM64_ILP32 is not set
CONFIG_IP_ADVANCED_ROUTER=y
CONFIG_NETFILTER=y
CONFIG_NETFILTER_ADVANCED=y
CONFIG_BRIDGE_NETFILTER=y

ARMv8 (arm64). Installation and setup of ubuntu ARM64 14.04 include some complicated process if user is not a linux expert. But many references can be found in the various internet sites. So we will not deal with the installation and setup step of ubuntu ARM64 root file system. After we install ubuntu ARM64 14.04.2, we had to install the utility library such as libvirt-bin, virtinst, bridge-utils and etc. via apt-get and aptitude which are the powerful package management tool in ubuntu.

3.2 ARM64-QEMU Setup

In case of realizing the virtualization with KVM for ARM64 SoC, almost use QEMU. It supports ARM64 from QEMU ver 2.1, and at the time of writing this paper, the result of cloning of QEMU git is ver 2.3. At above Sect. 3.1, we addressed ubuntu ARM64 14.04.2 installation for host side root file system, but we could not install a qemu-system-aarch64 package via apt-get with the lack of qemu-aarch64 repository binary. So we cloned the QEMU source repository and complied a qemu-system-aarch64 binary as shown in below script.

```
#git clone git://git.qemu.org/qemu.git qemu.git
#cd qemu.git
#./configure -target-list=aarch64-softmmu --enable-debug \
 --enable-kvm
#make -j8
#make install
```

3.3 Making a qcow2 Image for Guest OS

QEMU requires a guest disk image for virtualizing an ARM64 virtual machine. Xen virtualization require some additional physical and logical partitioning for guest domain. But to get a guest disk image in QEMU, we used qemu-img command for building a guest disk image with qcow2 format with online commanding. As explained in Sect. 3.2 previously, qemu-img command had already installed with using make install command automatically. Qemu-img supports that it makes and converts a raw or qcow2 disk image file. We made a raw image file (aarch64-v1.img) with 10 GBytes size by commanding qemu-img for installing of ubuntu root file system which is used by QEMU's guest emulation. After the completion of ubuntu netboot image which will be explained in the next Section, we changed that image to aarch64-kvm.qcow2 which has qcow2 file format. By using the qcow2 with QEMU ver 2.3, we could depress the warning message by QEMU system.

3.4 Getting Netboot Image

To install an ubuntu root file system for QEMU ARM64 in guest disk image, one need netboot images. Netboot image consists of temporary kernel (vmlinuz) and ramdisk image (initrd.gz) which have the main function for supporting ubuntu network installation. One can use the reference [8, 9] to get each of them.

3.5 Guest OS Setup Using Netboot Image

One can install the root file system for guest OS with below script and the downloaded netboot images (vmlinuz, initrd.gz). More detailed command line and argument are shown in below box lines. This process will take long time to complete.

```
qemu-system-aarch64                \
 -machine virt -cpu cortex-a57        \
 -nographic -smp 6 -m 4096          \
 -kernel ./netboot/vmlinuz             \
 -initrd ./netboot/initrd.gz             \
 -drive file=aarch64-v1.img,if=none,id=blk   \
 -device virtio-blk-device,drive=blk      \
 -net user,hostfwd=tcp::1014-:22         \
 -device virtio-net-device,vlan=0         \
 -append "console=ttyAMA0 --"
```

3.6 Starting Guest OS

Finally, we could start guest OS with KVM-QEMU as shown in below boxed line command script successfully. Real boot process took long time which was about more than 5 min.

```
qemu-system-aarch64                  \
 -machine virt -cpu cortex-a57        \
 -nographic -smp 4 -m 4096             \
 -localtime                           \
 -rtc base=utc                         \
 -kernel vmlinuz-3.13.0-49-generic      \
 -initrd initrd.img-3.13.0-49-generic     \
 -drive file=aarch64-kvm.qcow2,if=none,id=blk  \
 -device virtio-blk-device,drive=blk       \
 -net user,hostfwd=tcp::1014-:22          \
 -device virtio-net-device,vlan=0          \
 -append "root=/dev/vda2 rw console=ttyAMA0 --"   \
 -serial null
```

According to above setting procedures, we can show the whole KVM-QEMU virtualization system block diagram with ARM64 server system as shown in Fig. 3.

As known in the fact that KVM-QEMU virtualization has various benefits, we can use multiple virtual machines to increase the utilization factor of server system resources, to provide the complete independent computing environment for various

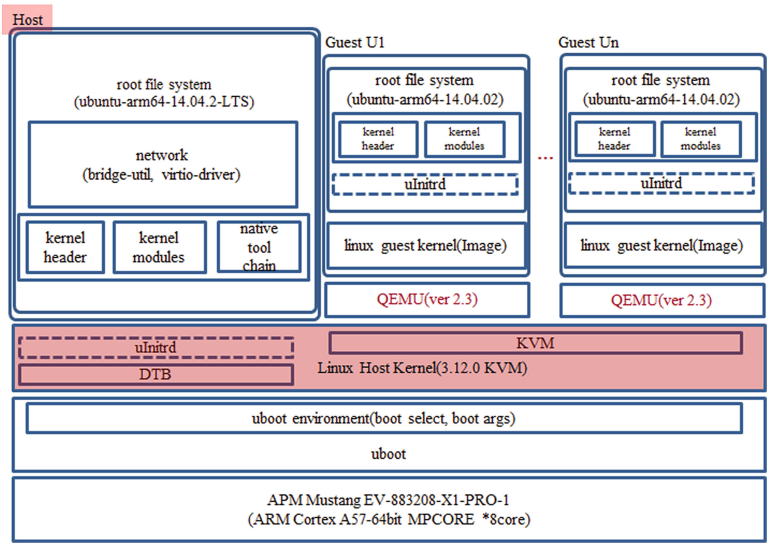

Fig. 3. KVM-QEMU virtualization with ARM64 server system

users, to migrate one physical machine to another, and etc. ARM64 server system with KVM-QEMU has the same virtualization benefits. But we have to point out the system performance degradation in our experimental system. Virtio still doesn't support the multi core in ARM64 completely. We found that the boot time of guest OS has taken long times (about more than 5 min) and some disk IOs are very slow. So we have to solve this problem for the real business deployment with ARM64 server system. This step will be the one of our next step research process.

4 Experimental Result

We installed and experimented KVM-QEMU virtualization with the ARM64bit server reference board. Our experimental system adapted the ARM64 ubuntu 14.04.2 root file system. The boot message of host OS in server board is captured and shown in Fig. 4. Host OS with KVM which has the kernel configuration described in Sect. 3.1 was good and had booted very quickly.

```
L3C: 8MB
Initializing cgroup subsys cpuset
Initializing cgroup subsys cpu
Initializing cgroup subsys cpuacct
Linux version 3.12.0-mustang_sw_1.14.14+ (root@jsma-i7) (gcc version 4.9
i Apr 10 16:51:28 KST 2015
CPU: AArch64 Processor [500f0000] revision 0
Machine: APM X-Gene Mustang board
bootconsole [earlycon0] enabled
efi: Getting parameters from FDT:
efi: Can't find System Table in device tree!
On node 0 totalpages: 4194304
  Normal zone: 57344 pages used for memmap
  Normal zone: 4194304 pages, LIFO batch:31
PERCPU: Embedded 10 pages/cpu @fffffffc3fff81000 s11776 r8192 d20992 u409
pcpu-alloc: s11776 r8192 d20992 u40960 alloc=10*4096
pcpu-alloc: [0] 0 [0] 1 [0] 2 [0] 3 [0] 4 [0] 5 [0] 6 [0] 7
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 413
Kernel command line: root=/dev/sda2 rw ip=129.254.75.125:129.254.75.123:
stang:eth0:off panic=1 console=ttyS0,115200 earlyprintk=uart8250-32bit,0
otlb=65536 pcie_ports=native
PID hash table entries: 4096 (order: 3, 32768 bytes)
Dentry cache hash table entries: 2097152 (order: 12, 16777216 bytes)
Inode-cache hash table entries: 1048576 (order: 11, 8388608 bytes)
software IO TLB [mem 0x43e8600000-0x43f0600000] (128MB) mapped at [fffff
Memory: 16379104K/16777216K available (6601K kernel code, 430K rwdata, 2
 bss, 398112K reserved)
```

Fig. 4. Host system boot messages.

After completing the boot process of host OS, verifying KVM support for host OS is done by commanding kvm-ok on system serial console. Host OS returned the support of KVM availability as shown in Fig. 5.

```
root@localhost:/# uname -a
Linux localhost.localdomain 3.12.0-mustang_sw_1.14.14+ #5 SMP Fri Apr 10 16
:51:28 KST 2015 aarch64 aarch64 aarch64 GNU/Linux
root@localhost:/# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.2 LTS
Release:        14.04
Codename:       trusty
root@localhost:/# kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
root@localhost:/#
```

Fig. 5. Verification of KVM support on host OS.

As explained in Sect. 3.6, the starting messages of guest OS in host OS are shown in Fig. 6. We checked out the normal startup messages of guest OS.

```
root@localhost:~/aarch64_kvm# ./start-ubuntu-kvm.sh
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuacct
[    0.000000] Linux version 3.13.0-49-generic (buildd@twombly) (gcc versio
n 4.8.2 (Ubuntu/Linaro 4.8.2-19ubuntu1) ) #81-Ubuntu SMP Tue Mar 24 19:35:3
7 UTC 2015 (Ubuntu 3.13.0-49.81-generic 3.13.11-ckt17)
[    0.000000] CPU: AArch64 Processor [411fd070] revision 0
[    0.000000] psci: probing function IDs from device-tree
[    0.000000] PERCPU: Embedded 12 pages/cpu @ffffffc0fffa6000 s17088 r8192
 d23872 u49152
[    0.000000] Built 1 zonelists in Zone order, mobility grouping on.  Tota
l pages: 1034240
[    0.000000] Kernel command line: root=/dev/vda2 rw console=ttyAMA0 --
[    0.000000] PID hash table entries: 4096 (order: 3, 32768 bytes)
[    0.000000] Dentry cache hash table entries: 524288 (order: 10, 4194304
bytes)
[    0.000000] Inode-cache hash table entries: 262144 (order: 9, 2097152 by
tes)
[    0.000000] software IO TLB [mem 0x138000000-0x13c000000] (64MB) mapped
at [ffffffc0f8000000-ffffffc0fbffffff]
[    0.000000] Memory: 4037352K/4194304K available (5718K kernel code, 590K
 rwdata, 2536K rodata, 304K init, 538K bss, 156952K reserved)
```

Fig. 6. System boot messages of guest OS.

Finally we showed the normal prompt of guest OS after completing the boot process of guest OS in Fig. 7. We verified the normal operation of guest OS with KVM-QEMU virtualization in ARM64 server system successfully.

```
Ubuntu 14.04.2 LTS ubuntu ttyAMA0

ubuntu login: root
root
Password:
Last login: Tue May 12 13:52:37 KST 2015 on ttyAMA0
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-49-generic aarch64)

 * Documentation:  https://help.ubuntu.com/

 System information disabled due to load higher than 4.0

33 packages can be updated.
26 updates are security updates.

root@ubuntu:~# 
```

Fig. 7. Guest OS console after booting

5 Conclusions

In this paper, we explained the process of KVM-QEMU virtualization technique in ARM64bit server reference platform sequentially. KVM-QEMU virtualization in ARM64bit server system has almost the same realization processes and advantages over the conventional x86-64 system. But we identified the performance issues with related to virtio library in our experimental server system. It will be one of the next research step.

Acknowledgments. This work was supported by the ICT R&D program of MSIP/IITP. [B0101-15-0548, Low-power and High-density Micro Server System Development for Cloud Infrastructure].

References

1. ARM Technical Conference (2014). http://www.armtechcon.com/
2. APM X-Gene Mustang. https://www.apm.com/products/data-center/x-gene-family/x-gene/
3. Ubuntu on Applied Micro Circuits Corp. X-Gene 1 (Storm). http://www.ubuntu.com/certification/hardware/201407-15333/
4. https://help.ubuntu.com/community/KVM
5. http://www.bennee.com/~alex/blog/2014/05/09/running-linux-in-qemus-aarch64-system-emulation-mode/
6. https://gmplib.org/~tege/qemu.html
7. Ubuntu Core 14.04.2 LTS (Trusty Tahr). http://cdimage.ubuntu.com/ubuntu-core/releases/14.04/release/
8. http://ports.ubuntu.com/ubuntu-ports/dists/trusty-updates/main/installer-arm64/current/images/generic/netboot/vmlinuz
9. http://ports.ubuntu.com/ubuntu-ports/dists/trusty-updates/main/installer-arm64/current/images/generic/netboot/initrd.gz
10. Jinsuk, M., Hakyoung, K., Wan, C.: Software development environment for ARM64bit server system. In: Proceedings of IEIE Conference, Deajeon, Korea, pp. 56–57 (2014)