International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 07 | July 2022 www.irjet.net p-ISSN: 2395-0072

Server Emulator and Virtualizer for Next-Generation Rack Servers

Chinmay N[1], Sujata D. Badiger[2]

[1,2]*Electronics and Communication Engineering, RV College of Engineering, Bengaluru, India*

Abstract – *The existing system of software development, debugging and software validation is heavily dependent on the actual physical hardware availability. The entire development process is slow and delayed without the availability of the actual hardware and thus affects the business function. In this paper, OpenBMC firmware is booted in the emulated ast2600-evb board using QEMU and all the required drivers and services such as networking, ssh, WebUI are enabled. The container image is created to automate the entire process which reduces the software development time and dependencies between the software development and the hardware availability.*

1. INTRODUCTION

Time to market for today's server is heavily dependent on physical hardware. The software development and validation is delayed because of the unavailability of the actual hardware. Adapting to the newer technologies is also hindered because of the unsupported hardware. All these affect the business requirements and the market of servers.. Emulation of physical hardware will solve the problem caused by any delays in chip tape out that will postpone the development of purely software applications.

1.1 Motivation

Today's servers market is heavily dependent on the availability of the physical and thus development process is slow. The main motivation is to reduce the time required for the software development process and testing high quality BMC images. This reduces the time for development as the image is ready with various applications and services enabled before the actual hardware is manufactured. This helps the development team work on the defects and try adding other new applications required and supported by the board. It helps in opening and running existing applications on the emulated board and also helps the developer team in testing the other new applications.

1.2 Importance of Emulator

The dependence of the software development timetable on hardware supply is significantly decreased by including emulated devices. As a result, a software development phase will start considerably earlier in the process of creating a product. The timeframe for developing a typical embedded device can be affected by emulated devices.Both the creation as well as testing of a software package and the hardware validation can benefit from emulated devices.

2. LITERATURE REVIEW

This section contains a survey of present technologies and research available related to the topic in an attempt to better understand the efforts that have gone into this field of study and also understand where the efforts should be focused while developing the product. The papers discussed in this section include work related to current OpenBMC image booting methods, Software design, implementation of virtual design and optimization using QEMU, Flashing OpenBMC images using QEMU, etc. These papers have state-of-art methods to boot and optimize OpenBMC images using the server emulator QEMU.

In [1], Rui Almeida has suggested using a simulation tool, namely QEMU, to aid in the creation and simulation of dependable systems. Aiming for a simulated environment which covers the multiplicity use case, allows validation of dynamic interaction under multiple architectures, and offers reliability calculations to contrast architecturally redundant systems, extensions based on this tool were built.

QEMU, which supports both Xen and KVM, is commonly used in cloud systems. Essentially, it is a dynamic translator that is quick and portable and an embedded device emulator that simulates numerous CPUs and board models. Additionally, it can offer a rapid virtual platform for software development. For example, Android Emulator uses it to simulate the entire mobile platform. To simulate new hardware, however, QEMU needs a new virtual device module because it only supports common hardware. XiaoXiao Bian et al. [7], have wrapped up the investigation into full system emulation, examine the internals and architecture of QEMU, provide precise procedures for building user-defined virtual hardware devices, and write the Linux kernel drivers for the new device.The findings of the research demonstrate that the entire environment for running, testing, and debugging is created, and user-level apps can be created again for new virtual hardware before the real device is made available.

Francesco Menichellin et al. [9] presents an emulation environment for rough memory architectures.

Approximate memories are important in the context of error-tolerant applications, which save energy at the expense of the incidence of data processing errors. Memories that allow controlled read/write errors are referred to as approximate memories. These faults are typically the result of architectural or circuitry approaches that were implemented to conserve energy. Since the amount of permitted error depends on the application, it is very crucial that these systems be able to be simulated. Through simulation, one may examine an application's behavior and study its tolerance for actual error rates, figuring out how much energy can be saved without sacrificing output quality. Based on QEMU, an emulation ecosystem was created for these architectures that enables the running of programmes that can assign some of their information in a memory region that is prone to errors. An emulated design, the fault injection model, and a case study demonstrating the outcomes of the emulator were demonstrated in this work.

In [11], Tse-Chen Yeh and Ming-Chao Chiang have offered an interface for QEMU and SystemC virtualized platforms to connect to the master/slave interfaces of hardware modelled in SystemC. The virtual platform may run a fully-Hedged operating system like Linux and makes use of QEMU as the instruction-accurate instruction set simulator (lA-ISS). In order to assist the co-creation of various hardware modeling techniques and software components at the preliminary phase of Electronic System Level (ESL) design, the proposed interface allows the hardware modeled in SystemC to access hardware modeled in QEMU. The experimental findings—using Direct Memory Access Controller (DMAC) as an example—show that the proposed interface enables the cross-validation of hardware models and device drivers as well as the migration of hardware models from QEMU to SystemC. Additionally, the virtual platform has the ability to deliver statistics that are instruction-accurate, making it simple to assess the hardware model performance and conduct design space exploration.

3. Methodology

The information required to develop the firmware is collected. Once the related information is collected and development of the OpenBMC firmware is done. The firmware developed is implemented in the emulated board in QEMU. The same firmware is implemented in the actual hardware. The functionality of the OpenBMC is validated and compared. Addition of new features are done to the firmware and then implemented in both hardware and emulator and these features are then validated. The figure 1 shows the methodology discussed.

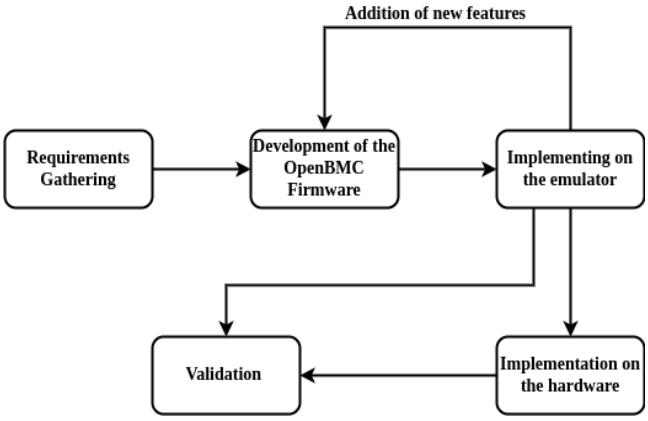

Figure-1: Design Methodology

The firmware developed needs to have applications and services that are completely software dependent and basic hardware services that are required to boot the kernel and get the BMC shell loaded in the emulated board.

4. Implementation

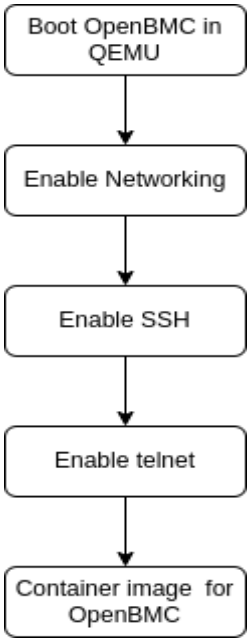

Figure-2: Flowchart

The design is implemented as shown in flowchart figure 2. The implementation of the server emulator is done with the following steps discussed in the flowchart. It involves the booting of OpenBMC in an ast2600-evb emulated board in QEMU. The BMC is loaded in QEMU and various other functionalities are added to the BMC loaded in the QEMU. Network is enabled in the BMC loaded. Once the BMC is up with the network it allows to enable other functions i.e ssh, telnet. These supports give the developer remote access to the BMC that fastens the development,

debugging and validation. The container image is created to automate the complete process that loads the OpenBMC firmware and BMC with all functionalities is loaded in the emulated board. The container image developed can be run on any platform and OS and there is no specific requirement to run the container image and set up the BMC in QEMU. This helps the developer to set up BMC in any given system and perform the operations with this container image.

The network is enabled using the tap interface. This interface just acts as a bridge between the host system and QEMU. The bridge is created by the host and QEMU needs to be identified about the bridge name through the different parameters that are passed in the command. In this interface both QEMU and host are in the same subnet and share the same default gateway. Through this interface ping function works within a guest and the BMC i.e emulation in QEMU is accessible from the host or any other network. This helps the user having remote access to the BMC shell.

3. RESULTS

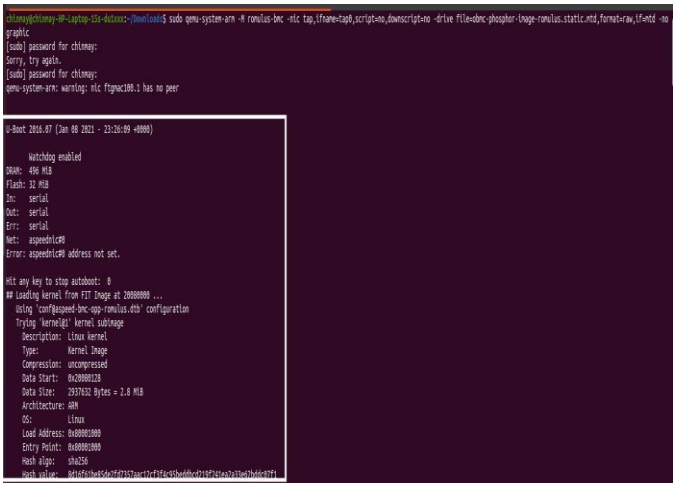

Figure-3: Boot Logs and device configuration

Figure 3 shows the boot logs of the OpenBMC firmware that is loaded in an ast2600-evb emulated board. The logs show the information regarding the memory that is allocated and the device configuration.

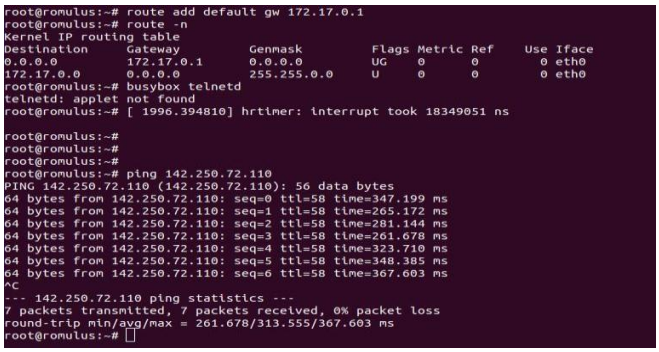

Figure-4: Routing table and IP configuration of the BMC

Figure 4 shows the IP configurations of the BMC i.e is loaded in QEMU . It also shows the routing table of the BMC. The BMC is up with the network and able to send and receive packets from different IP address with different subnets.

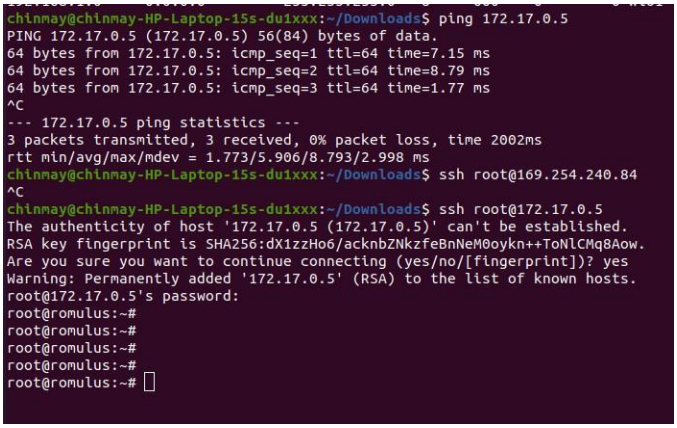

Figure-5: SSH support

Figure 5 shows the BMC has SSH enabled and can be remotely accessed through SSH. It also shows the BMC is able to receive packets that are transmitted from remote hosts .

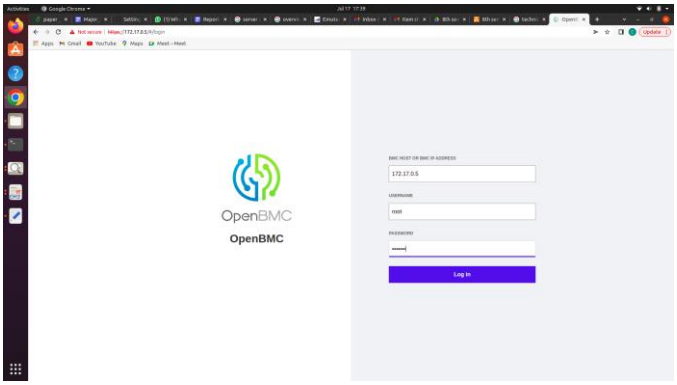

Figure-6:

Figure 6 shows the Login page for the OpenBMC that is loaded for the user interaction and the login page comes up when the user is working on the BMC or the BMC is powered on and has to perform operations to the BMC loaded in QEMU through webUI. This login page gives the user access to the information related to BMC i.e loaded in the emulated board.

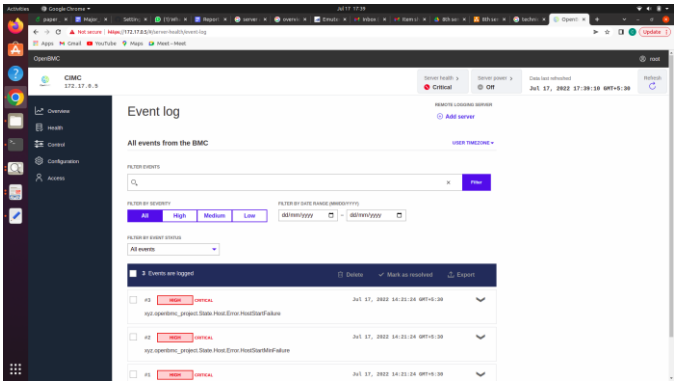

Figure-7: OpenBMC web page

Figure 7 shows the BMC health condition and the current state of the BMC. It also shows the date and time of the BMC being accessed. It shows the logs related to the BMC health. The page also has other features like network settings, user permissions, controlling the BMC states etc .

All these functionalities enabled in emulated boards makes development , debugging and validations of both software and hardware easier and faster.

4. CONCLUSION AND FUTURE SCOPE

In this paper, the Management controller firmware was emulated in QEMU and a complete BMC was set up with users having access to the BMC services and its application. The BMC setup had support for networking, ssh,telnet and various other software dependent applications that don't require hardware support to be emulated. Further the complete automation of emulating BMC in qemu was done with the help of a docker container image. This image makes all the required setup that are necessary to install a QEMU in any given environment. This project has significantly decreased the dependence of the software development timeline on hardware delivery by including QEMU emulated BMC devices. As a result, the software development process can now start considerably earlier in the process of creating a product.

In future, the WebUI can be enabled with more features added, virtual GPIO support can also be added to the emulated BMC. It is also possible to add emmc support and enable the development of drivers supported by QEMU in future.

REFERENCES

1] Rui Almeida et al. “Reliable Software Design Aided by QEMU Simulation”. In: 2021 22nd IEEE International Conference on Industrial Technology (ICIT). Vol. 1. 2021, pp. 797–804.

[2] Prachi Palsodkar et al. “Yocto Based Home Automation using Open BMC Platform and RestAPI”. In: 2022 IEEE Delhi Section Conference (DELCON). IEEE. 2019, pp. 1–3.

[3] Zhang Rongqiang. “Bringing the OpenBMC for platform management system in telco cloud”. In: (2019).

[4] Jae-Hoon An, Chanyeong Kim, and Younghwan Kim. “The design and development of integrated interface for provision BMC framework”. In: Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems. 2018, pp. 276–278.

[5] Calvin Muramoto, Stephen Dunlap, and Scott Graham. “Improving Hardware Security on Talos II Architecture Through Boot Image Encryption”. In: International Conference on Cyber Warfare and Security. Vol. 17. 1. 2018, pp. 489–496.

[6] Jae-Hoon An, Younghwan Kim, and Chang Won Park. “Design of Framework supporting IPMI and DCMI based on Open BMC”. In: Proceedings of the International Conference on Research in Adaptive and Convergent Systems. 2017, pp. 298–299.

[7] XiaoXiao Bian. “Implement a virtual development platform based on QeMU”. In: 2017 Interna- tional Conference on Green Informatics (ICGI). IEEE. 2017, pp. 93–97.

[8] Fabio Frustaci et al. “Approximate SRAMs with dynamic energy-quality management”. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems 24.6 (2016), pp. 2128–2141.

[9] Francesco Menichelli et al. “An emulator for approximate memory platforms based on qemu”. In: International Conference on Applications in Electronics Pervading Industry, Environment and Society. Springer. 2016, pp. 153–159.

[10] Fabio Frustaci et al. “SRAM for error-tolerant applications with dynamic energy-quality manage- ment in 28 nm CMOS”. In: IEEE Journal of Solid-state circuits 50.5 (2015), pp. 1310–1323.

[11] Tse-Chen Yeh and Ming-Chao Chiang. “On the interfacing between QEMU and SystemC for virtual platform construction: Using DMA as a case”. In: Journal of Systems Architecture 58.3-4 (2012), pp. 99–111