ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

System Virtualization to Prevent Vendor Lock-in Issues

Mounika M[1], Chinnaswamy C N[2], T H Sreenivas[3]

PG Student, Department of Information Science, NIE, Mysuru, India[1]

Associate Professor, Department of Information Science, NIE, Mysuru, India [2]

Professor, Department of Information Science, NIE, Mysuru, India [3]

ABSTRACT: Whole system virtualization is type of system virtualization which requires the emulation of the complete guest operating system. The guest system has a different instruction set architecture when compared to the host system. In the context of general purpose computers where there are no resource constraints, this process might sound fairly straight forward and simple. However in an embedded environment with resource constraints mainly in terms of memory and processing speed, the task of emulating a processor with a different instruction set architecture is extremely challenging.

Manufactures using microcontrollers are often tied down a specific microcontroller vendor due to lack of compatibility between controllers from different vendors. This project proposes an emulation layer over the hardware that will remove any form of platform dependency. The emulation layer along with the underlying hardware together forms a whole system virtual machine monitor. With that implemented, any controller with an emulator running on it can virtually execute any program from another controller vendor which will make migration between vendors extremely easy as the cost of migration will reduce drastically.

KEYWORDS: Emulation, Virtualization, Vendor lock-in, platform independence

I. INTRODUCTION

The world today is at the brink of a revolution. As Internet of Things (IoT) and cloud computing become more than just concepts they require innovative changes in the way traditional computing is performed. The evolution of cloud computing lead to major changes in the way computing is done using general purpose computers. Virtualization, though an old concept has found new meaning and importance with evolution of cloud. In a similar fashion, connecting everyday objects using networks (the basic concept of IoT) in the most efficient and scalable way possible is a common topic of discussion.

The concept of connecting everyday objects in a network and connecting fully fledged computers via the internet take different approaches. This is because of the amount of resources at the disposal of both of them. While general purpose computers come with high computational power and easily expandable memories (in some cases apart from the already large internal memory) embedded systems which are controller based are always faced with the problem of restricted resources. For example, Platform as a Service, an integral feature of cloud involves shared usage of a particular platform without the hassles of having to spend money on installing hardware, operating systems, middleware such as databases. This is largely possible due to the partitioning and virtualization on general purpose and mainframe computers and is generally preferred when the requirement of this platform is for a short duration of time. However, in case of an embedded system which uses microcontrollers this approach can be fairly complicated. Each object in a network which connects controller-based devices may have a different architecture and come from different vendors. Manufacturers are locked with specific vendors because of intensive platform dependency.

For example, consider the case of connected cars. The electronic control unit (ECU) of the vehicle is the control system governing various parameters of the vehicle. The ECU is designed and developed using sets of components from various semiconductor manufactures. More often than not the designers of the ECUs, design them in such a way that the design can be reused over the years multiple times with minor changes as the industry requirements evolve.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Owing to the use of the same sets of components over the years they are restricted by the choice of components as selection of a new set of components or a new platform means loss of compatibility with existing systems. A lot of component vendors exploit this restriction making platform dependency a major problem.

In addition to vendor and developer deadlocks, platform dependencies cause other major problems. When control units of different vehicles are connected together and an attempt is made to integrate update processes of vehicles using the network, the different platforms they use becomes a major issue. It is important that control units using any platform or controller be updated in the same way, that is irrespective of the platform it uses or the controller which executes the instructions the patch should executed as if it were written specifically for that processor. The software patches or the messages should be completely platform agnostic.

The above stated example is one use case for embedded hypervisors or virtual machine monitors. This paper suggests the use of hypervisors to ensure platform independency and to prevent vendor lock-in issues in case of connected objects. Connected objects not just include objects which use fully fledged processors but also controllers which are extremely platform dependent.

The rest of the paper is divided into sections as follows. Section II gives details of the related work published in this field and section III describes the existing architecture. Section IV describes the proposed methodology. Section V outlines the implementation with an example and the section following it contains the sample results. Section VII describes the pros and cons and the final section concludes the paper.

II. RELATED WORK

In [1] authors used average residual batterVirtualization as a concept that was introduced decades ago but it has gained a new definition over the last few years due to the internet revolution. There is lot of documentation on the concept of virtualization and emulation. Most of the documentation is specific to general purpose processors. Hypervisors for embedded systems has also gained immense popularity over the few years because of the requirement of connecting controllers with different architecture. [2] byJames Smith and Ravi Nair gives the fundamentals of virtual machines. It gives the basic definition of a virtual machine and states that it can support individual processes or a complete system depending on the abstraction level where virtualization occurs thereby giving the definition of process virtualization and system virtualization. It also gives a definition for emulation as a process that converts instructions belonging to one instruction set architecture (ISA) to another. Mendel Rosenblum and Tal Garfinkel in [1] trace the history and the variation of the definition of virtualization and virtual machines based on the changing technology.

One of the first companies to use emulation to provide backward compatibility was IBM; one of their white papers [3] gives an overview of IBM'S Virtual Machine Facility/370. It describes the virtual machine concept and its capabilities and implementation in VM/370. [4] is a paper from the University of Michigan presented at a conference and specifically discusses type II virtual machines that builds on the abstractions provided by a host operating system. The authors discuss the performance of the system and suggest an extension to the host operating system to accommodate emulation without overhead.

[5] looks into the concepts of para virtualization and the scalability of tradition virtual systems. Through faithful emulation, VMMs support the execution of legacy guest operating systems such as Windows or Linux without modifications. However, traditional VMMs suffer from poor scalability and extensibility. To overcome the poor scalability and extensibility of traditional virtual machine monitors that partition a single physical machine into multiple virtual machines, the Denali VMM uses para virtualization to promote scalability and hardware interposition to promote extensibility.

[6] gives a detail study of virtualization of third generation of general purpose computers and describes formal techniques used to derive precise sufficient conditions to test whether a quintessential third generation architecture can support virtual machines.

In the paper [7] the author looks into the need for virtualization of embedded devices and runs a guest OS on the chip. It focuses on run a RTOS and a general purpose OS concurrently and to reuse software resources on both of them. The second contribution is improvement in system availability by enabling guest OS to reboot independently with a low performance overhead.

[8] and [9] are comprehensive studies on the role of virtualization in next generation embedded systems and the necessity of virtualization in the embedded world. System virtualization, which enjoys immense popularity in the enterprise and personal computing spaces, is recently gaining significant interest in the embedded domain. Starting from a comparison of key characteristics of enterprise systems and embedded systems, the papers examine the

difference in motivation for the use of system virtual machines, and the resulting differences in the requirements for the technology.

III. EXISTING SYSTEM

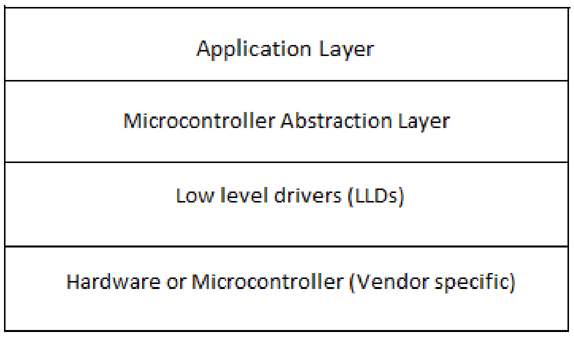

Application Layer
Microcontroller Abstraction Layer
Low level drivers (LLDs)
Hardware or Microcontroller (Vendor specific)

Fig 1: Layered Diagram of the existing system

Today in the existing system each microcontroller vendor has their own controller architecture based on a different processor. Though most of them have abstraction layers to ensure that multiple applications from different companies working on application software are compatible with the controller, there is no way to ensure that updates or patches from one vendor are compatible with the other. The layered architecture of the existing system is shown in Fig 1.

IV. PROPOSED SYSTEM

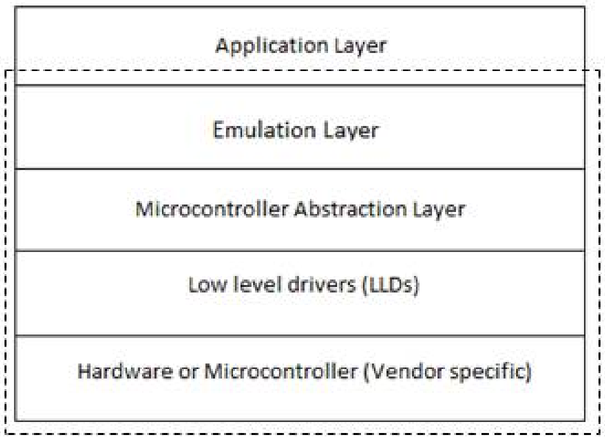

Application Layer
Emulation Layer
Microcontroller Abstraction Layer
Low level drivers (LLDs)
Hardware or Microcontroller (Vendor specific)

Fig 2: Layered Diagram of the Proposed System.

This paper proposes an emulation layer over the microcontroller abstraction (which makes the applications from different vendors' compatible with the microcontroller) as shown in Fig 2. The doted lines that encompass the layers represent the virtual machine. One of the biggest reasons for doing this is platform independence. With that, what one may have is a platform (any processor with the emulator running on it) which can virtually execute any program from another processor vendor. That simply means that by introducing an emulator, we make programs hardware (and operating system) agnostic.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

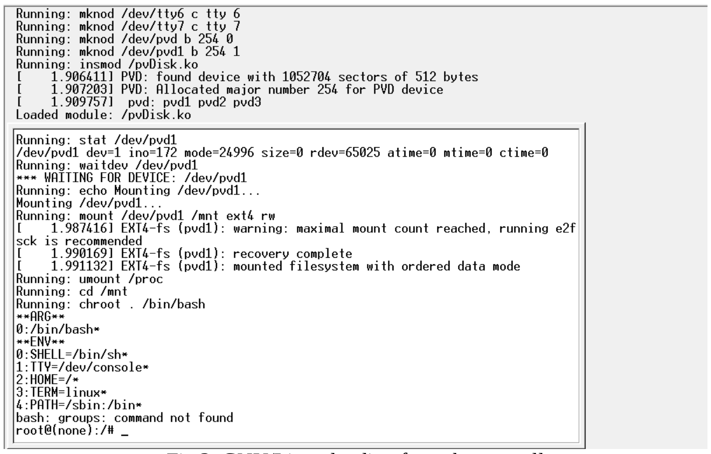
Vol. 4, Issue 5, May 2016

V. IMPLEMENTATION

The study of system virtualization was done by emulating PXA-255 XScale architecture on an ARM Cortex-M4 clone. The cortex clone does not have a Memory Management Unit and has an internal flash memory of 1MB and SRAM of 62kB. A GNU/Linux image built for the PXA-255 is stored on an external SD Card. The image is uncompressed and loaded sector by sector. It is executed from an external SDRAM interfaced with the controller.

VI. RESULTS

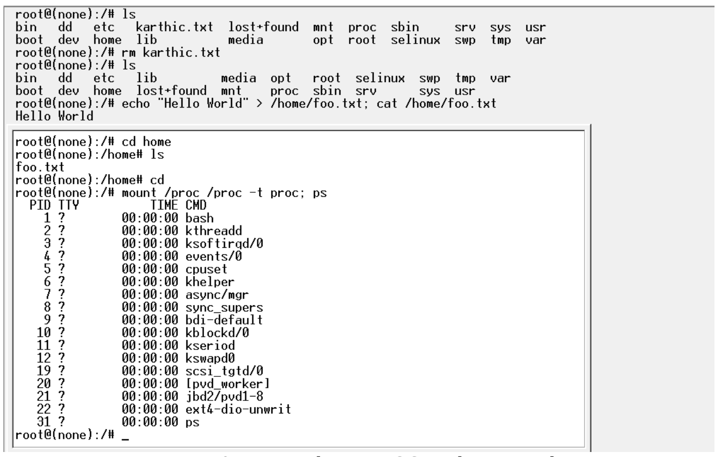
A GNU/Linux operating system image built for the PXA-255 core was run on the Cortex-M4 clone. It was observed that the operating system can be successfully run on the native chip and the user can access both the native processor with an RTOS and the guest operating system depending on the requirement. The results are shown in Fig 3 and Fig 4 which show the guest OS built for another architecture running natively on the controller. The results are snapshots of the terminal emulator which acts an interface between the user and microcontroller. This experiment proves two points; first a system with a specific ISA can be run on a system with a different ISA thereby getting rid of system dependencies. Being able to boot an OS built for a different ISA natively implies any other binary image can be successfully executed.



```
Running: mknod /dev/tty6 c tty 6
Running: mknod /dev/tty7 c tty 7
Running: mknod /dev/pvd b 254 0
Running: mknod /dev/pvd1 b 254 1
Running: insmod /pvDisk.ko
[    1.906411] PVD: found device with 1052704 sectors of 512 bytes
[    1.907203] PVD: Allocated major number 254 for PVD device
[    1.909757]  pvd: pvd1 pvd2 pvd3
Loaded module: /pvDisk.ko
```

```
Running: stat /dev/pvd1
/dev/pvd1 dev=1 ino=172 mode=24996 size=0 rdev=65025 atime=0 mtime=0 ctime=0
Running: waitdev /dev/pvd1
*** WAITING FOR DEVICE: /dev/pvd1
Running: echo Mounting /dev/pvd1...
Mounting /dev/pvd1...
Running: mount /dev/pvd1 /mnt ext4 rw
[    1.987416] EXT4-fs (pvd1): warning: maximal mount count reached, running e2f
sck is recommended
[    1.990169] EXT4-fs (pvd1): recovery complete
[    1.991132] EXT4-fs (pvd1): mounted filesystem with ordered data mode
Running: umount /proc
Running: cd /mnt
Running: chroot . /bin/bash
**ARG**
0:/bin/bash*
**ENV**
0:SHELL=/bin/sh*
1:TTY=/dev/console*
2:HOME=/*
3:TERM=linux*
4:PATH=/sbin:/bin*
bash: groups: command not found
root@(none):/# _
```

Fig 3: GNU/Linux loading from the controller.



```
root@(none):/# ls
bin   dd   etc   karthic.txt  lost+found  mnt  proc  sbin     srv  sys  usr
boot  dev  home  lib          media       opt  root  selinux  swp  tmp  var
root@(none):/# rm karthic.txt
root@(none):/# ls
bin   dd   etc   lib         media  opt   root  selinux  swp  tmp  var
boot  dev  home  lost+found  mnt    proc  sbin  srv      sys  usr
root@(none):/# echo "Hello World" > /home/foo.txt; cat /home/foo.txt
Hello World
```

```
root@(none):/# cd home
root@(none):/home# ls
foo.txt
root@(none):/home# cd
root@(none):/# mount /proc /proc -t proc; ps
  PID TTY          TIME CMD
    1 ?        00:00:00 bash
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    4 ?        00:00:00 events/0
    5 ?        00:00:00 cpuset
    6 ?        00:00:00 khelper
    7 ?        00:00:00 async/mgr
    8 ?        00:00:00 sync_supers
    9 ?        00:00:00 bdi-default
   10 ?        00:00:00 kblockd/0
   11 ?        00:00:00 kseriod
   12 ?        00:00:00 kswapd0
   19 ?        00:00:00 scsi_tgtd/0
   20 ?        00:00:00 [pvd_worker]
   21 ?        00:00:00 jbd2/pvd1-8
   22 ?        00:00:00 ext4-dio-unwrit
   31 ?        00:00:00 ps
root@(none):/# _
```

Fig 4: Accessing the guest OS on the native chip

Second, an embedded hypervisor can aid in accessing a fully blown OS on a controller. Though this is not straight forward it can be done after taking care of the bare minimum resource requirements. Hence, by using this concept with

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

some basic overhead (that can be optimized eventually) organizations need not be tied down to one vendor. Also, the concept makes connecting different objects to a network easier.

VII. Advantages And Drawbacks

The advantages of this type of approach include:

- Platform independence: Programs agnostic of the microcontroller.
- Ability to execute binary code of another microcontroller.
- Easy migration between controllers with different ISAs.
- Virtualization gives a sandbox environment which act as an excellent enabler of secure execution.
- Makes connectivity between different objects easier in an IoT environment.

The drawbacks include:

- Latency due to the time taken to emulate instructions.
- Certain special instructions extremely specific to the architecture cannot be emulated.
- Speed of emulation depends on the speed of the native machine.
- An architecture specific build of the operating system should be available for both the controllers.

VIII. Conclusion and Future Work

Embedded hypervisors and emulation provide hardware independence and support for multiple operating systems on a single processor. This helps in easy migration between controllers from different vendors and prevents vendor locking. It comes with multiple advantages which include system security, system reliability, the option of reuse of legacy code and migration of applications from uni-core to multi-core systems. While we step into an age of seamless connectivity this concept provides a platform for updating software dynamically.

Code changes to enable faster emulation or over clocking the native controller can marginally enhance the performance. Porting the same on multi-core controllers which support parallel processing of the emulator can greatly enhance the performance of the hypervisor.

References

1. Mendel Rosenblum, Tal Garfinkel, "Virtual Machine Monitors: Current Technology and Future Trends," IEEE Computer, May 2005, pp. 39-47.
2. James Smith, Ravi Nair, "The Architectures of Virtual Machines," IEEE Computer, May 2005, pp. 32-38
3. L.H. Seawright, R.A. MacKinnon, "VM/370 – a study of multiplicity and usefulness," IBM Systems Journal, vol. 18, no. 1, 1979, pp. 4-17.
4. S.T. King, G.W. Dunlap, P.M. Chen, "Operating System Support for Virtual Machines," Proceedings of the 2003 USENIX Technical Conference,June 9-14, 2003, San Antonio TX, pp. 71-84.
5. A. Whitaker, R.S. Cox, M. Shaw, S.D. Gribble, "Rethinking the Design of Virtual Machine Monitors," IEEE Computer, May 2005, pp. 57-62.
6. G.J. Popek, and R.P. Goldberg, "Formal requirements for virtualizable third generation architectures," CACM, vol. 17 no. 7, 1974, pp. 412-421
7. Kanda, Wataru, et al. "Spumone: Lightweight cpu virtualization layer for embedded systems." Embedded and Ubiquitous Computing, 2008. EUC'08. IEEE/IFIP International Conference on. Vol. 1. IEEE, 2008.
8. Heiser, Gernot. "The role of virtualization in embedded systems." Proceedings of the 1st workshop on Isolation and integration in embedded systems. ACM, 200
9. Aguiar, Alexandra, and Fabiano Hessel. "Embedded systems' virtualization: The next challenge?." Rapid System prototyping (RSP), 2010 21st IEEE International symposium on. IEEE, 2010.

BIOGRAPHY

Mounika M is a PG student in the Department of Information Sciences at The National Institute of Engineering, Mysuru. She has a Bachelor's degree in Electronics and Communication.

Mr. ChinnaswamyC N is an Associate Professor in the Department of Information Science & Engineering. He has received his M.Tech from VTU and B.E. from University of Mysore. He is pursuing his Ph.D and his teaching and research interests are in the field of Computer Networks and Internet of Things.

Dr. T H Sreenivasis Professor in the Department of Information Science & Engineering. He has received his Ph.D. from IIT Madras, M.Tech from IIT Kanpur and B.E. from University of Mysore.His teaching and research interests are in the field of Operating Systems, Networking, Schedule Optimization and Wireless Sensor Network.