BOOT TIME OPTIMIZATION OF AN EMBEDDED SYSTEM AND IMPLEMENTING AN USB DRIVER

[1]Bathini Avinash, [2]Mrs. K.Siva Sundari

[1]PG Research Scholar, [2]Associate Professor
Department of ECE, SNIST, Hyderabad, Telangana, India, 501301.

Abstract— Fast kernel boot-time is one of the major concerns in industrial embedded systems. Application domains where boot time is relevant include (among others) automation, automotive, avionics etc. Linux is one of the big players among operating system solutions for general embedded systems; hence, a relevant question is how fast Linux can boot on typical hardware platforms (ARM11) used in such industrial systems. One important constraint is that this boot-time optimization should be as nonintrusive as possible. The reason for this comes from the fact that industrial embedded systems typically have high demands on reliability and stability. For example, adding, removing or changing critical source-code (such as kernel or initialization code) is impermissible. Its shows the steps towards a fast-booting Linux kernel using non-intrusive methods. Moreover, targeting embedded systems with temporal constraints, this shows how fast the real-time scheduling framework ExSched can be loaded and started during bootup. This scheduling framework supports several real-time scheduling algorithms (user defined, multi-core, partitioned, fixed-priority periodic tasks etc.) and it does not modify the Linux kernel source code. Hence, the non-intrusive bootup optimization methods together with the un-modified Linux kernel and the non-patched real-time scheduler module offer both reliability and predictability.

Keywords— embedded, Linux, boot, optimization, kernel, system, embedded platform.

I. INTRODUCTION

Linux, developed by Linus Torvalds was specifically targeted to desktop PCs running an Intel 80x86 or compatible microprocessor. Today Linux has been ported to many different microprocessors and runs on all sorts of platforms, these devices are not even general purpose computer systems and include things such as network routers, heart monitors, and data collection units for tracking different weather pattern sensors from isolated unattended remote locations. It is these type of systems that have collectively come to be labeled as “Embedded Linux”. Linux meets the requirements of everyone in all fields such as embedded, real time, personal computer in terms of functionality, scalability and cost. Linux supports all these features because of its configurable nature and hugely supported open source community of developer all across the world. Linux meets the requirements of everyone in all fields such as embedded, real time, personal computer in terms of functionality, scalability and cost. Various mission critical systems such as an aircraft control system or an application platform running on a communication node require system to be started up in few seconds to a minute. To achieve the fast startup in these system it is necessary to examine the boot process of the operating system used which is Linux. Optimizing the Linux Bootup process should not override or change the existing functionality, nor the normal operation. On optimization we extend the system to a level where it reduced the time of the booting process, but fail to satisfy normal operation then whole process will be lost. Boot optimization should not affect the system functionality, but in turn help system to enhance its booting process for faster system upgrade.

II. BOOT PROCESS

The boot time for an embedded system is of paramount importance. To optimize the boot time of the Linux, let’s understand the process of booting a system running Linux operating system, the Linux booting process consists of multiple stages as shown in the figure below

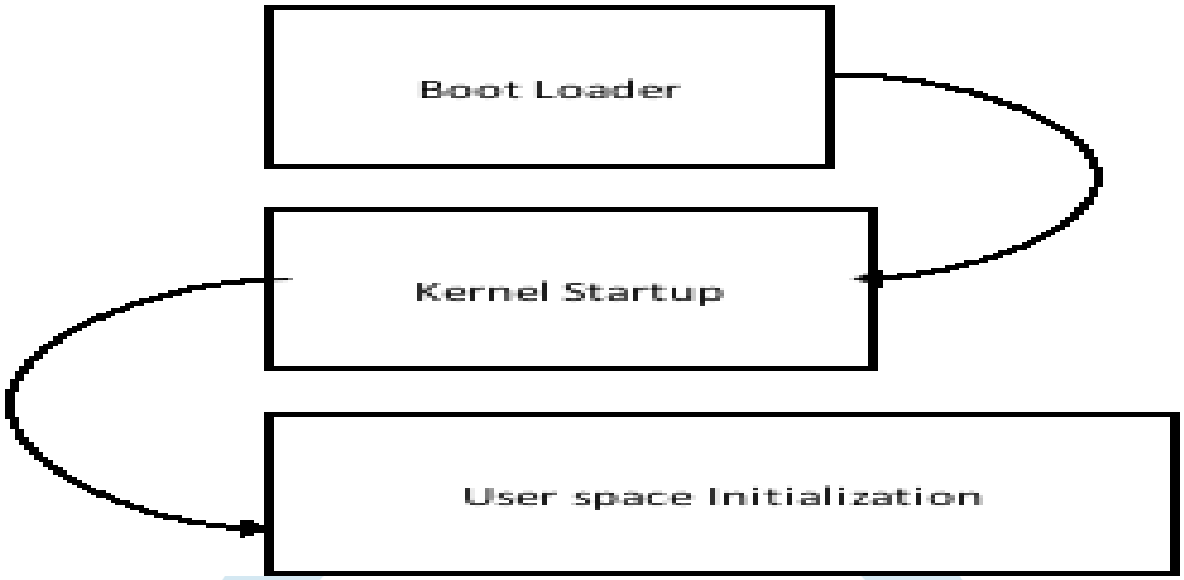

Fig. 1.High level Linux boot process

Boot loader process

This process includes early hardware initiation and interaction and load the kernel from flash to RAM. The time take during this process can be described as 1. Power/ Clock Stabilization ----- usually negligible but should be considered 2. Low Level CPU Initialization - ~ 100 ms ----- Bootloader (often multi-stage, ie secure boot)

Kernel Startup

This process does the following activities

1. Loading images (kernel, u -boot, rootfs, dtb)
    a. Usually from NOR or NAN D Flash
    b. Compressed kernel
2. Subsystem (Driver) initialization
3. Mounting a root file system

User space

The user space process covers

1. Init scripts
2. System processes
3. Applications

The user space process and configuration are very user and applications dependent; the user space can have a display terminal or may not have a display terminal. The best example of the user space is the user interface of the Android operating system which is working over the Linux kernel.

III. OPTIMIZATION PROCESS

Linux boot optimizations methods are very platform and application dependent; the optimization need to consider the whole system architecture for selecting the boot optimization strategies e.g.

1. What software update methods will be used (affect features needed in boot loaders)
2. What are the essential functions of the device that must be running first?
3. When are network features needed
4. The difference between the production and development system images can also pose a different optimization need e.g. during the development more feature rich image might be needed and based on the development and production need the kernel configuration might be different (to enable easier debugging) for different version of the images. Any optimization on the Linux booting process would fall into either of these 2 broad categories:

SIZE

The size dictates what would be kernel image size based on the available hardware or application, the size optimization process including

1. Reducing the size of binaries for each successive component loaded.
2. Removing features that are not required to reduce the size and complexity of the image.

SPEED

The speed optimization process includes

1. Optimize for target processor
2. Use faster medium for loading primary, secondary boot loaders and kernel.
3. Reduce number of tasks leading to the boot. The size and speed optimization process is dependent on the application need of the embedded system, if the system is designed to be working as an unattended remote sensor for weather collection then both the size and speed are of paramount the optimization process would require the identification of usability and then target time requirement, based on these criteria the optimization process for Size and Speed can be utilized.

A generic optimization process for the embedded system is depicted in the below diagram.

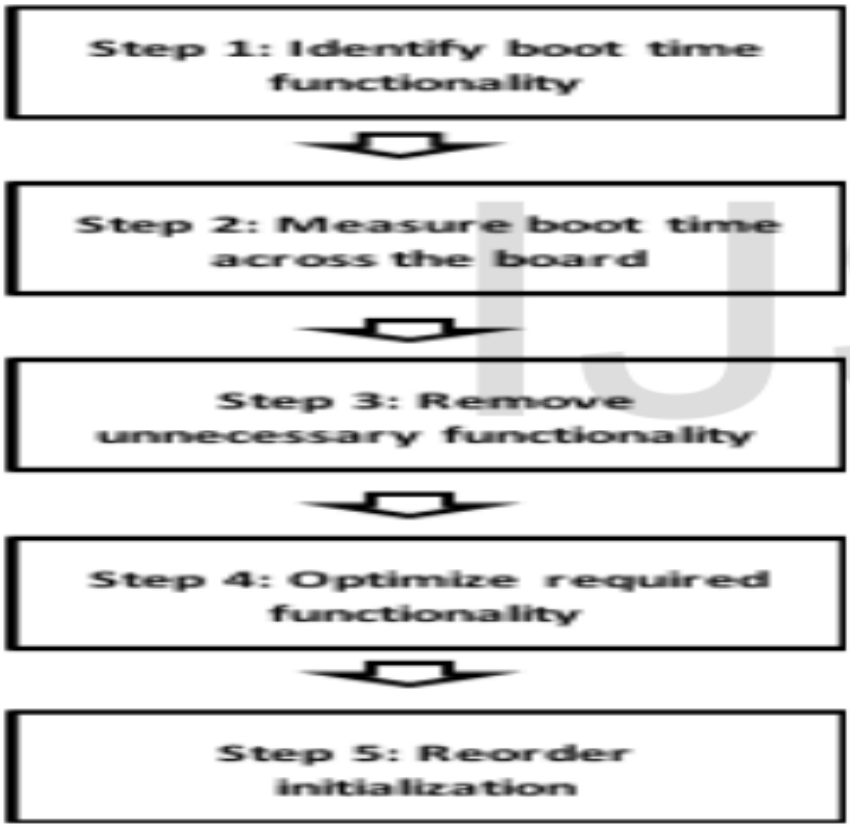

Fig: 2. Optimization Process

IV. HARDWARE IMPLEMENTATION

RASPBERRY PI BOARD:

The **Raspberry Pi** is a credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. The Raspberry Pi is manufactured in two board configurations through licensed manufacturing deals with Newark element14 (Premier Farnell), RS Components and Egoman. These companies sell the Raspberry Pi online.Egoman produces a version for distribution solely in China and Taiwan, which can be distinguished from other Pis by their red coloring and lack of FCC/CE marks. The hardware is the same across all manufacturers. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S

700 MHz processor, Video Core IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded to 512 MB. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and persistent storage.

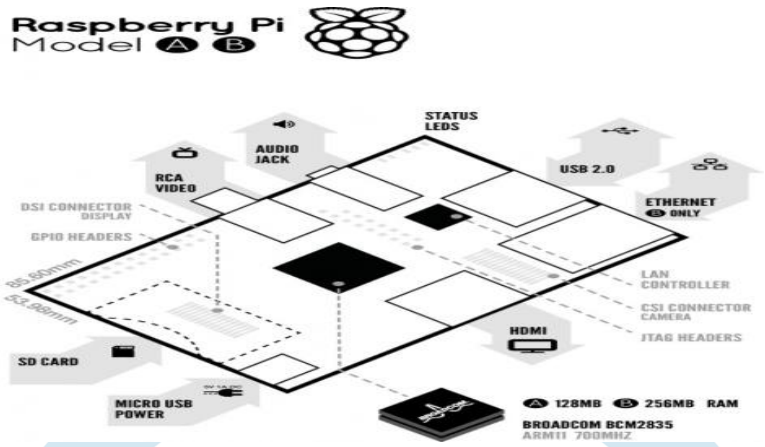

Fig: 3. Board features

The Foundation provides Debian and Arch Linux ARM distributions for download. Tools are available for Python as the main programming language, with support for BBC BASIC (via the RISC OS image or the Brandy Basic clone for Linux), C, Java and Perl.

V. SOFTWARE REQUIREMENTS

Linux Operating System:

Linux or GNU/Linux is an open source operating system for computers. The operating system is a collection of the basic instructions that tell the electronic parts of the computer what to do and how to work. Free and open source software (FOSS) means that everyone has the freedom to use it, see how it works, and changes it. There is a lot of software for Linux, and since Linux is free software it means that none of the software will put any license restrictions on users. This is one of the reasons why many people like to use Linux.

A Linux-based system is a modular Unix-like operating system. It derives much of its basic design from principles established in UNIX during the 1970s and 1980s. Such a system uses a monolithic kernel, the Linux kernel, which handles process control, networking, and peripheral and file system access. Device drivers are either integrated directly with the kernel or added as modules loaded while the system is running.

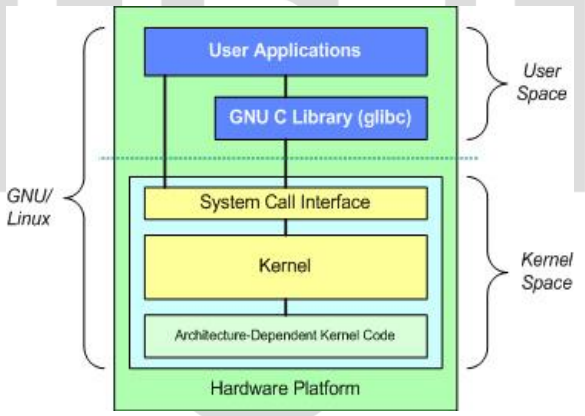

Fig:4. Architecture of Linux Operating System

VI. RESULTS

Figure 5, 6 are the results before and after optimization of Raspberry Pi which can be seen using Boot chart.

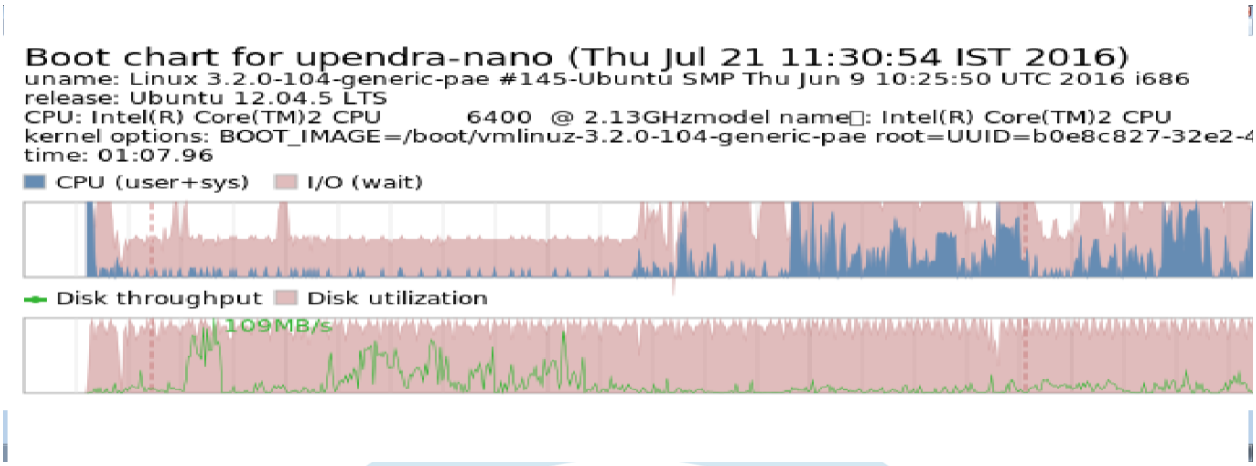

Fig: 5. Time before optimization in Boot Chart

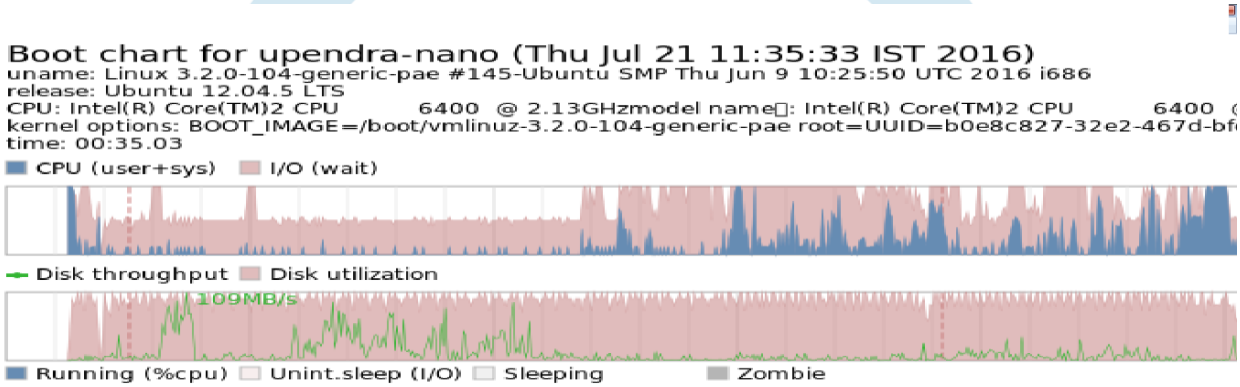

Fig: 6. Time after optimization in Boot Chart

VII. CONCLUSION

The project **"Boot Time Optimization for Embedded Linux Systems"** has been successfully designed and tested. It has been developed by integrating features of all the hardware components and software used. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. Secondly, using highly advanced ARM9 board and with the help of growing technology the project has been successfully implemented.

References

[1] Heeseung Jo; Hwanju Kim; Jinkyu Jeong; Joonwon Lee; Seungryoul Maeng; **"Optimizing the startup time of embedded systems: a case study of digital TV"** , Consumer Electronics, IEEE Transaction on , Publication year 2009 Volume: 55 , Issue : 4 page(s): 2242 -2247.

[2] Kyung Ho Chung; Myung Sil Choi; Kwang Seon Ahn; Kyungpook Nat. Univ., Daegu "**A Study on the Packaging for Fast Boot-up Time in the Embedded Linux"** Embedded and Real-Time Computing Systems and Applications, 2007, RTCSA 2007, 13th IEEE international conference, Publication Year: 2007, Page(s) 89-94

[3] K. H. Chung, H. Y. Cha, K.S. Ahn, **"A Study on the Effective File System Packaging in the Embedded Linux Systems"**, Proceedings of the 2005 International conference on Embedded System and Applications, Las Vegas USA, 2005, pp252-258.

[4] Joe, Inwhee; Lee, Sang Cheol; Division of Computer Science & Engineering, Hanyang University, Seoul, South Korea **"Bootup time improvement for embedded Linux using snapshot images created on boot time"** Next Generation Information Technology(ICNIT), 2011 The 2nd international Connference, Publication Year : 2011 Page(s) 193 – 196

[5] Kumar, L.; Kushwaha, R.; Prakash, R; Embedded Syst., Centre for Dev. of Adv. Comput., Noida, India **"Design & Development of Small Linux Operating System for Browser Based Digital Set Top Box"**

[6] Amir Nizam Ansari, Mohamed Sedkl, Neelam Sharma, Anurag Tyagi, **"An Internet of Things Approach for Motion Detection using Raspberry pi"**, International Conference on Intelligent Computing and Internet of Things (IC1T),IEEE,PP.131-134,2015

[7] Gurjashan Singh Pannu, Mohammad Dawud Ansari,Pritha Gupta, **"Design and Implementation of Autonomous Car using Raspberry pi"**, IJCA, Volume 113 – No. 9,PP.22-29, March 2015.