Journal of Systems Architecture 61 (2015) 361–373

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

LASER: Latency-Aware Segment Relocation for non-volatile memory

Myungsik Kim [a], Seongjin Lee [a], Jinchul Shin [b], Youjip Won [a,*]

[a] Department of Computer Science Engineering, Hanyang University, 17 Haengdang-dong, Sungdong-gu, Seoul, Republic of Korea
[b] SK Planet Co., Ltd., Bundang-gu, 264, Pangyo-ro, Seongnam-si, Gyeonggi-do, Republic of Korea

ARTICLE INFO

Article history:
Received 2 December 2014
Received in revised form 15 May 2015
Accepted 27 June 2015
Available online 4 July 2015

Keywords:
Latency-Aware Segment Relocation
LASER
Nonvolatile Random Access Memory (NVRAM)
Fast boot
Embedded linux

ABSTRACT

In this work, we develop Latency-Aware Segment Relocation (LASER) which relocates a subset of segments of binary image to NVRAM to reduce program launch latency. A significant amount of time is spent on loading the binary image to main memory and initializing it. We develop a new system startup mechanism, to reduce the boot time by using selectively relocating read-only sections in NVRAM. We develop a model to determine the set of segments to be loaded into NVRAM given the maximum launch latency constraint and the physical latency of NVRAM. We implement LASER scheme to commercially available embedded systems (S5PC100 and Zynq7020). LASER-enabled systems achieve 54% and 38% reduction in boot time in S5PC100 and Zynq7020 systems, respectively.

1. Introduction

Many devices, including mobile devices [1,2], video consumer electronics [3], and home appliances, [4] are implemented on an embedded system. As computing power of embedded devices increases, the devices are becoming more complex not only to accommodate user requirements but also to provide better user experience and services. Higher complexity in embedded systems results in larger program size which leads to longer program loading time and longer boot time. In a conventional embedded system, responsiveness is one of the most important traits that all users crave, and it can be improved by optimizing the booting process.

Since many programs are initialized during booting process, it is time consuming to access storage device during boot up process. In modern embedded Linux systems, kernel and user program image stored in storage or non-volatile memory are loaded to main memory during booting process. However, instead of executing the program directly from non-volatile memory, the system copies the program that is to be executed to a faster main memory, such as DRAM. Although the system may boot faster by reading programs from NVRAM, one of the drawbacks is that all of the program code has to be loaded to the DRAM every time the system boots up.

Non-Volatile RAM (NVRAM) is a common name for byte-addressable persistent memory which is known to be a fast, non-volatile, and low-power consumption device that can potentially replace traditional storage or memory devices [5–7]. However, since various NVRAMs have different characteristics and use different types of cell technologies, such as magneto-resistive type, phase-change type, etc., it is very difficult to generalize the performance of NVRAM based systems as shown in Table 1.

We found that kernel image used in our experiments show that about 90% of the kernel image has read-only attribute and the rest has read-write attribute. It is time consuming to load read-only sections of the program to the main memory every time the system boots. We propose Latency-Aware Segment Relocation (LASER) scheme that focuses on reducing the boot time by relocating the read-only sections to NVRAM, removing loading time of these sections. The proposed scheme is also a NVRAM latency-aware program loading scheme which is a function of program image size, bandwidth of NVRAM, and boot time budget in NVRAM based embedded Linux system. Since the volume and the speed of NVRAM vary depending on the purpose of the embedded system, measuring the performance of a system that exploits NVRAM as a booting memory is not trivial. As an effort to provide a resolution to the problem, we empirically make a linear model that projects boot latency of a program image with concerning a latency of NVRAM as well as loadable image size in a given boot time constraint.

In order to evaluate the proposed scheme, we used two types of evaluation boards: S5PC100 and Zynq7020. Both boards are based on embedded Linux, but they have different CPU and memory subsystems. We observe that implementing LASER scheme on S5PC100 board reduces booting time by 55%, from 16.1 s to 7.3 s,

* Corresponding author.

http://dx.doi.org/10.1016/j.sysarc.2015.06.003
1383-7621/

Table 1
Comparison of memory technologies [33].

Type	SRAM	DRAM	HDD	NAND	PCRAM	RRAM	MRAM
Maturity	Mass production					Development	
Non-volatility	No				Yes		
Byte-addressability	Yes		No			Yes	
Density (F^2)	>100	6–8	2/3	4–5	8–16	>5	37
Read Latency	<10 ns	10–60 ns	8.5 ms	25 us	48 ns	< 10 ns	< 10 ns
Write Latency	<10 ns	10–60 ns	9.5 ms	200 us	40–150 ns	10 ns	12.5 ns
Energy per bit	>1 pJ	2 pJ	0.1–1 J	10 nJ	100 pJ	2 pJ	0.02 pJ
Static power	Yes	Yes	Yes	No	No	No	No
Endurance	$>10^{15}$	$>10^{15}$	$>10^{15}$	10^4	10^8	10^5	$>10^{15}$

compared to the legacy booting scheme, and LASER scheme on Zynq7020 board reduces booting time by 49% from 7.3 s to 4.5 s.

The main contributions of this paper are the following: (1) Our LASER booting method reduces boot time by selectively loading sections to NVRAM. (2) We design boot time model having the detail process of loading program image to main memory. (3) We compare the proposed LASER scheme to the legacy boot scheme, and validate LASER scheme on two evaluation platforms. (4) We evaluate LASER boot with boot time budget factors based on different image sizes, boot time constraints, and NVRAM bandwidths.

2. Related works

Reducing boot time is not a trivial problem, but a profound one which involves optimizing boot time of various embedded devices through many trial-and-errors which is very time consuming. There are many factors that influence the boot time. During booting, H/W components as well as arbitrary data structures need to be initialized. Also, a number of programs have to be initialized before the system can service users, and these programs include BIOS, boot-loader, OS, and user applications.

There are many optimization techniques that can be applied on different steps of the booting process. One optimization technique makes image size small by removing debugging symbols or removing unnecessary packages and codes [8]. Another technique tries to reduce the boot time by exploiting parallelism to maximize the utilization of available resources [9]. Bot time optimization techniques are divided into many different categories. This work focuses on reducing the memory copy overhead using NVRAM. We dedicate this section to discuss prior works related to loading the program image.

There are a number of researches that take advantage of loading to reduce the boot time [10,8]. Before powering off, Hibernation [11][1] stores current state of the main memory to the storage as an image. When the system is turned on the next time, the system recovers its previous state of the system by loading the stored data to the main memory. A benefit of using Hibernation comes from fast recovery of system state and quick hand over of the control from the system to the user, thereby reducing the stand-by power. Another similar Suspend-to-Disk scheme is Snapshot boot scheme [12]. This scheme reduces startup time by taking a snapshot image out of memory dump just after completing the boot and loading the snapshot image to the RAM.

Main memory dump and restoration methods, such as Hibernation and Snapshot boot scheme, however, require a significant amount of time in resuming and loading the image back to the main memory. The time to hibernate and resume becomes longer as the size of the image gets larger. As a solution to reduce the time to create and resume in Hibernation, some of the works propose creating a partial snapshot image with stage classification [2,13–15].

One alternate way of creating a similar effect as Hibernation is to use Suspend/Resume power management function [16]. [2] In this approach, kernel manipulates PMU (Power Management Unit) to enter a sleep mode and allows only a few critical devices to be active with minimum standby power. A benefit of using Suspend/Resume is that the system is instantly powered on without causing overhead to load the data from storage. It has to be noted that the system is not resuming from complete power down state but from sleep mode which limits its applicability to those systems that need constant power or is battery backed systems.

Execute-In-Place (XIP) scheme does not load program code to the main memory; instead, it fetches program code directly from a byte-addressable non-volatile memory (e.g., NOR Flash). NOR-XIP is a suitable solution for single chip micro controller because it requires simple memory organization, which consists of limited amount of SRAM and NOR memory, in a lightweight embedded system. Kernel XIP scheme [17] extends the XIP scheme to embedded Linux. Kernel boot image is stored in NOR Flash memory and the system executes boot code from NOR Flash. Since NOR Flash, which runs program code, has slow access time compared to DRAM, XIP has a performance drawback. Another limitation of XIP is that it cannot compress the program image because the program code must be readily available for execution. Advanced XIP filesystem (AXFS) [18] addresses this problem by letting filesystem exploit XIP or compressed method selectively.

In previous boot researches, different program loading methods were used according to the memory performances. Therefore, NVRAM with its byte-addressable and non-volatile features, would need a new program loading scheme. There are a few works that use NVRAM to improve boot performance. Kim et al. [19] address NVRAM to reduce the device startup latency by merging a portion of read only section on kernel image. Lee et al. [20] go a step further than providing optimized scheme for segments described in Kim et al. [19]. They provide an object filter to analyze the volatile attribute of variables in codes. In this paper, we further develop the work done by Kim et al. [19]. We extend the idea of selectively initializing sections of kernel image on to a user program. To understand the relationship between boot time budget and loadable image size when using NVRAM, this paper analyzes the effect of NVRAM latency which is not considered in the previous work. The proposed method is not a stand-alone approach but a complimentary one that can be used along with aforementioned boot time reducing techniques.

The rest of this paper is organized as follows. Section 3 illustrates each step of booting process. Section 4 analyzes these steps and investigates ways to apply LASER scheme to user program area. Then, Section 5 explains how to implement LASER scheme. Section 6 presents our experiment setup and measurement methods. Along with the performance comparison between LASER boot

[1] Also known as STD (Suspend-To-Disk).

[2] Also known as STR (Suspend-To-RAM or Suspend).