Proc. of Sixth International Conference On Advances in Computing, Electronics and Electrical Technology - CEET 2016

ISBN no. 978-1-63248-109-2 doi: 10.15224/ 978-1-63248-109-2-14

A Novel on-demand storage space enhancement approach

for solving smart phone memory crunch

[Shyjumon N, Parvathi B, Sarsij Kaystha]

Abstract—The paper present a method/apparatus to solve the memory crunch situations which are present and probably arise during the software upgrades of the smart phones. The present date smart phones (Android based) are using the open source ext4 file system and our method is working on top of the same by utilizing the existing utilities of the ext4 file system. The approach can be utilized in two modes and the underlying approach is same for both modes. For making the method/apparatus we are using the re-sizing of the file system partitions and the updation of the partition table. This approach has scope for even changing the entire partition layout of the smart phone device. We also developed the algorithm to manage the partitions and we also addressed the potential problems during the power on/off scenarios. This approach gives easiness in user mode and mission possible in software upgrade mode. This is not restricted to smart phones, but can extend to the any system runs with Linux as OS.

***Keywords—Memory*, OS, Android, Smart Phones, file system, ext4, re-partitioning, software upgrade, re-size**

I. Introduction

The major concern of any smart phone user will be the memory and performance. This paper primary focus on the memory concerns. One of the major attractive features of the smart phone is to have the software upgrades. Due to the increase in demand of the memory (ROM) requirements during the software upgrade to support new features and latest applications, the device memory re-partitioning is very much required. Also during the normal usage of the smart phones users wish to get more memory from the ROM. There will lot of space available in the system memory area of the device, which is generally reserved for future enhancement of software by the OEM (like: Samsung). The existing approach is to keep those statically reserved, no matter whether the user will go for an upgrade.

The approach proposed in this paper will give an option to the users to make use of the reserved memory in the system partitioning and at the same time OEMs can re-claim that when they provide the software upgrades. In this approach we have given two modes :user mode and upgrade mode. The user mode is solely intended for end users while the update mode is for OEMs.

The entire solution and algorithms are built on top of existing open source file system (ext4) and its utilities, so as to provide ease of development and deployment.

II. Approach & Method

A. *Overview of the method*

The approach has two modes of operation:

- Update mode
- User mode

The first mode is used by the OEMs and the second is for end user.

The algorithms internally used for both the modes are almost same. The first mode works during the recovery mode of the Linux kernel and the second will get triggered from the running kernel (normal mode).

The key features of this method are as follows:

- This utility provides a provision to the users to do dynamic repartitioning on the smart phones with the help of existing utilities of the open source ext4 file system.
- Two modes operation makes this easy to use
- Software upgrades will be easy (No space constraints)
- Fulfillment of On-demand user space memory expansion
- Flexibility to enhance user memory requirements

Shyjumon N
Samsung Research Institute Noida, Noida-201301, Uttar Pradesh
India

Parvathi B
Samsung Research Institute Noida, Noida-201301, Uttar Pradesh
India

Sarsij Kaystha
Samsung Research Institute Noida, Noida-201301, Uttar Pradesh
India

Proc. of Sixth International Conference On Advances in Computing, Electronics and Electrical Technology - CEET 2016

ISBN no. 978-1-63248-109-2 doi: 10.15224/ 978-1-63248-109-2-14

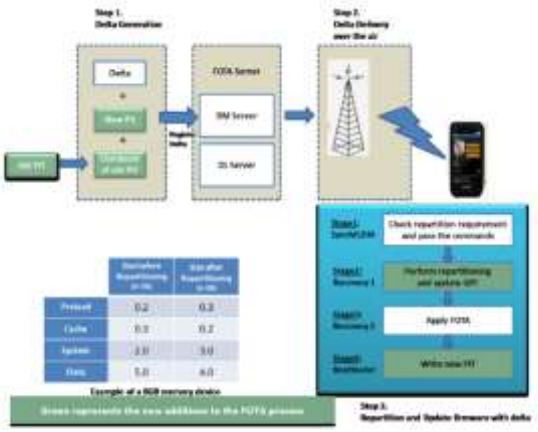

Figure 1. Overview of the process in update mode

B. *Approach and methodology*

We are making use of the existing open source ext4 file system utilities such as e2fsck, resize2fs, and parted lib.

As mentioned, the method has two modes and we have completed the concept proof implementation in one mode, which is update mode.

We have the main functional modules as below:

Computation:-

This module will tell you sequence of repartitioning because if you want to expand the partition you have to shrink the other partition first. Basically we will build a dependency graph of partitions and perform topological sort and the tasks will be performed in that order.

Generally we have three tasks in repartitioning.

- Resizing (resize2fs)
- Data shifting
- GPT (Global Partition Table) update

For a successful repartitioning, these tasks have to be executed sequentially in the same order.

The description of these tasks is as follows:

Resizing:-

Resizing is the major task which will enlarge or shrink the full disk file system layout and update the superblock as well as metadata. For this task we will be using open source utility (resize2fs).

The Figure 2 shows the layout, the data present and free space available in partition 3. As we can see the gray portion of the memory in figure 2 (just below partition 3) is an example of free space. Also the data in the partition 4 is scattered. The right side of the figure 2 shows the defragmented partition after resizing. All the free blocks of each partition is segregated to contiguous locations.

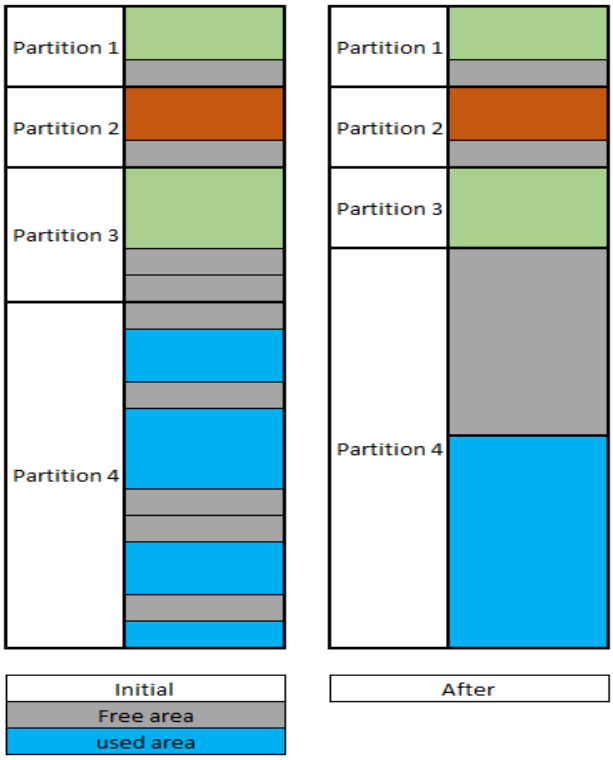

Figure 2. Memory Layout change in re-sizing

Data shifting:-

Resizing will segregate the data and the free space in the partition. However, the size of physical partition has not yet changed. The data needs to be shifted based on the location of the partition to which the free space should be appended.

Figure 3 shows the arrangement of the partition 4 after the data has been shifted to the end of the partition, thereby creating space at the beginning of the partition to be added to partition 3, just above it.

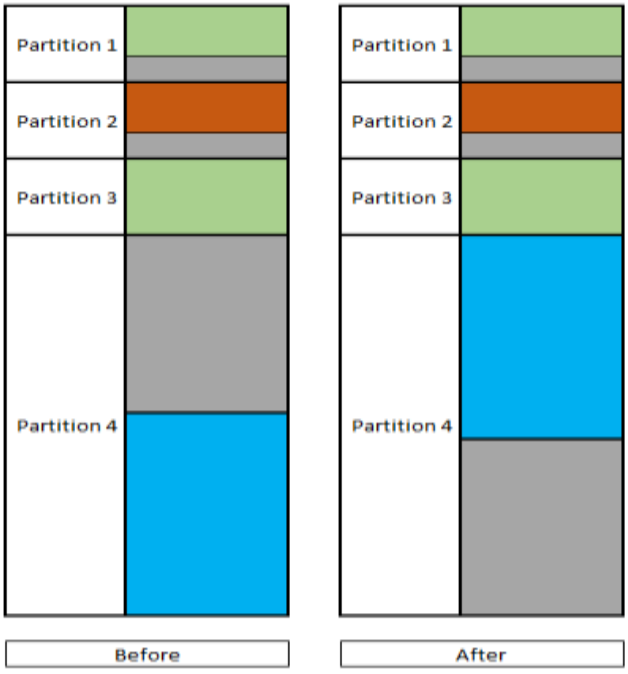

Figure 3. Memory Layout change in data shifting

Proc. of Sixth International Conference On Advances in Computing, Electronics and Electrical Technology - CEET 2016

ISBN no. 978-1-63248-109-2 doi: 10.15224/ 978-1-63248-109-2-14

GPT Update:-

After Data shifting operation, the GPT (Global Partition Information Table) will be updated to change the physical partition layout of the device. Figure 4 shows the updated physical partition layout

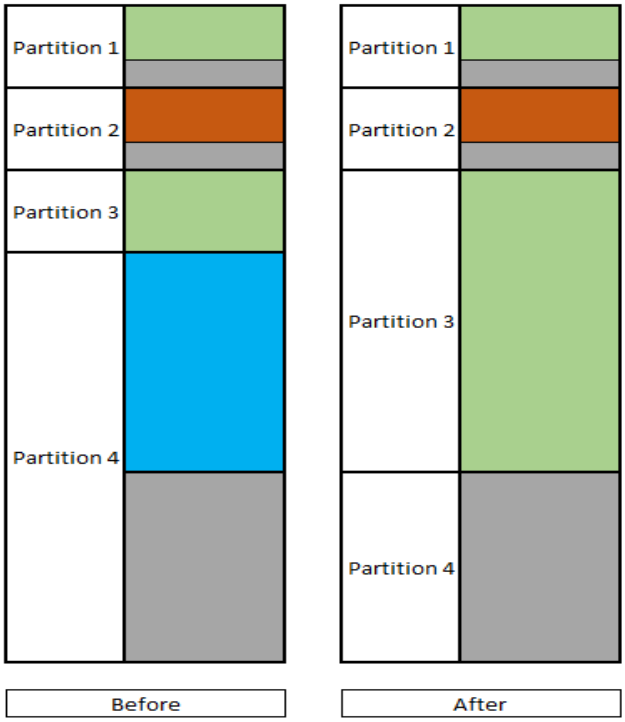

Figure 4. Memory Layout change when GPT update

C. *Implementation:*

The dynamic re-partitioning is carried out in 4 stages.

1. Stage 1: Validation of new Partition info
2. Stage 2: Recovery 1
3. Stage 3: Recovery 2 (Only for update mode)
4. Stage 4: Bootloader

Stage 1: New partition layout is received can be either for a software upgrade or as per the end user request, means the end user wishes to get more space in one of the partitions, mostly the user data partition. The new partition layout is then validated using the PIT (Partition Information Table) Parser utility in this stage. On successful verification, the appropriate recovery commands are written in the /cache/recovery/command file for OS upgrade mode or end user request mode. In the earlier case, 2 commands – one for repartition and other for applying the delta – are written. In the latter case, only one command - to repartition - is written. These commands will be executed by the init process when booted in recovery.

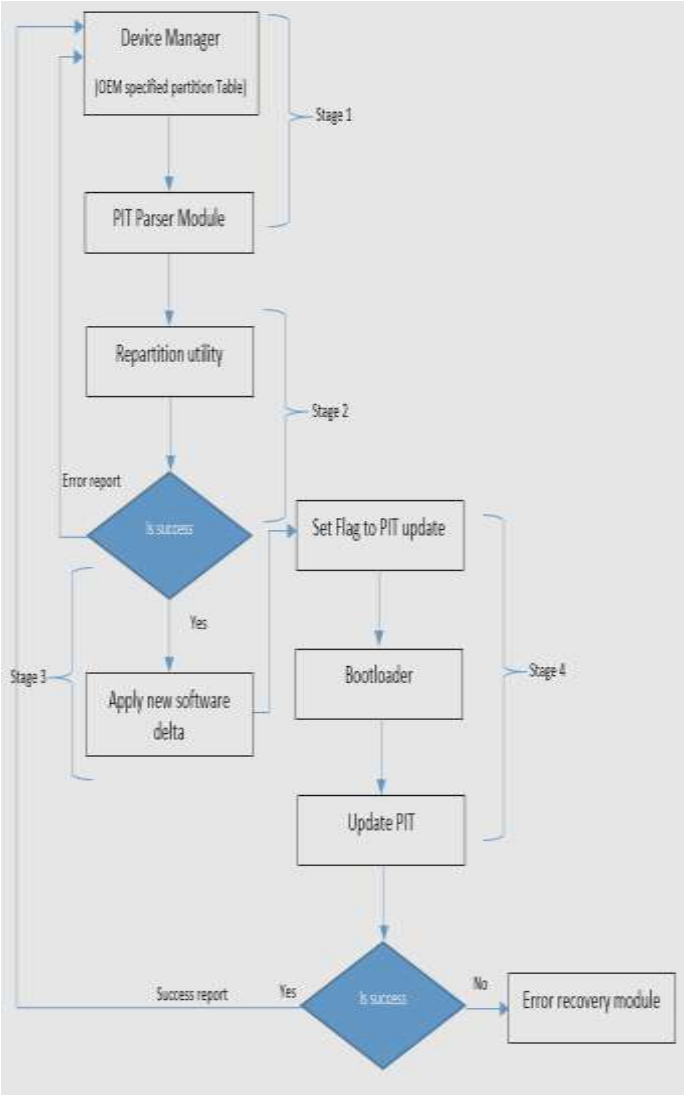

Figure 5. Implementation stages of repartitioning during OS upgeade.

Stage 2: In this stage, the device will boot in recovery and are execute only repartition commands. After successfully executing the repartition commands, in case of user triggered mode, a flag is set to indicate the bootloader to update the new partition layout in GPT. However, if the request is for OS upgrade, the device is rebooted in recovery mode (recovery 2) after successfully repartitioning.

In case of user triggered mode, after stage 2, a flag is updated for the bootloader stage and then device will be booted to normal mode.

Proc. of Sixth International Conference On Advances in Computing, Electronics and Electrical Technology - CEET 2016

ISBN no. 978-1-63248-109-2 doi: 10.15224/ 978-1-63248-109-2-14

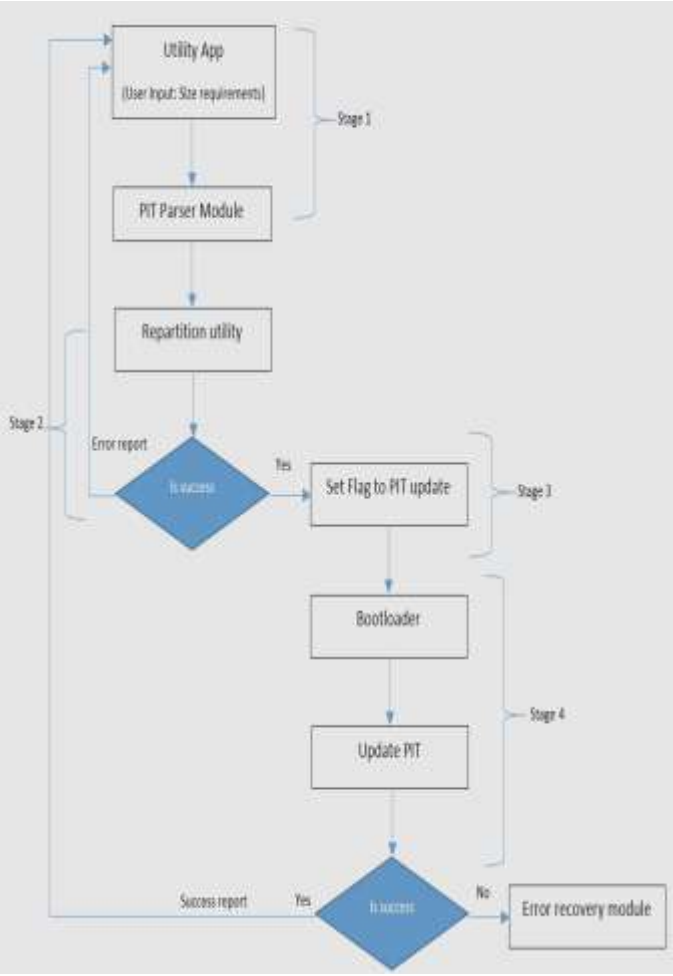

Figure 6. Implementation stages of reaprtitioning on user demand.

During the boot to normal mode, the bootloader stage acts to update PIT & GPIT tables with new partition information and user can enjoy the new created space seamlessly.

Stage 3: Recovery 2, this stage only applicable for the OEM triggered software update mode, this mode is designed to satisfy the OEM's requirements to update the new software on top of the existing software. This stage booting the bootloader acts to bridge the old partition information & new partition information table.

Stage 4: Bootloader Agent module, this module plays a vital role (after Stage 1 for user triggered mode and after Stage 2 (or rather coupled with Stage 2) OEM mode). The major duty is to check for the partition change update flag reading and accordingly modify the partition information table contents. Make the system ready with new layout of partitions.

All these stages plays in very much in the system level, any error in any stage can make the situation of making the mobile computing device dead forever. So every stage is incorporated with an error recovery system. The error recovery system is designed in two dimensional, one is to recover from the functional failures in the said module‘s functionality. The second dimension is the power failure recovery. In any stage or any time if the power is cut off due to lack of battery or due to malfunction of the power controller, the power recovery system will act and makes the system in-tact.

The basic philosophy of the recovery system is to retain the system, from where we started the process.

We have done extensive tests on our apparatus to make sure the recovery system is acting well, which include some set of automated robotic level testing process. As a result we could able to handle all such cases and make a robust system as our proposed apparatus.

D. *Future Scope*

The major limitation of this work is performance aspect. If the data is large in the partition where in the re-partitioning is performing there is significant performance problems. To improve the performance in data shifting, we recommend incorporating the compression techniques to this apparatus.

The proposal can be extended to other file systems like B-tree file system, where functionalities like on-line re-size and snap shot are available. This kind of file system can overcome the above mentioned overhead on performance easily. We have explored the same in B-tree file system and able to achieve significant improvement in performance. However present android smart phones are using the ext4 file system so we restricted our work to ext4 file system.

E. *Conclusions*

The approach and methodology mentioned in this paper is implemented for concept proofing and we could successfully complete achieve desired results. This is now in-progression for commercialization. This will be a handy method for all the normal users and OEMs.

Acknowledgment

This work had been done at Samsung R&D Institute India Noida. We thank our colleagues, Mr. Kashp Chawla and Mr. Tarun for helping us. We also want to thank all System Memory team members for the support they gave us. We would also like to thank our organization for creating such an informative atmosphere and helping us in whatever way they could.

We also wish thank, our fellow members in the Samsung R&D China, Nanjing for their proof of concept work on the command line utility for partition parsing. Our work taken their work reference to make one of the module in this proposed apparatus.

F. *References*

[1] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/ext4grow.html

[2] http://blog.fpmurphy.com/2011/10/fedora-16-gpt-grub2-bios-boot-partition.html?output=pdf

[3] Meaza Taye Kebede, Performance Comparison of Btrfs and Ext4 Filesystems, submitted in Oslo And Akersus University College Of Applied Science, 2012

Proc. of Sixth International Conference On Advances in Computing, Electronics and Electrical Technology - CEET 2016

ISBN no. 978-1-63248-109-2 doi: 10.15224/ 978-1-63248-109-2-14

G. *About Author (s):*

Shyjumon N did his B.Tech in Electrical & Electronics Engineering from Govt. Engineering College, Thrissur, Kerala in the year of 2002. He worked with Various R&D centers such as Samsung, Toshiba, Cranes, and Otwo in the field of Embedded Systems for the development of handheld devices. He is having more than 13 years of experience in the Linux Kernel and device drivers and also worked on different Bootloader Developments. He is specialized in BSP & Peripheral bus technologies such as SDIO, USB and presently working on Linux Kernel optimizations.

Parvathi Bhogaraju has graduated with master's degree in Process Control and Instrumentation from National Institute of Technology, Tiruchirappalli, in the year 2007. Before joining Samsung, she has worked with R&D centers of Toshiba and Emerson. She has more than 8 years of experience in device drivers and filesystem development in various OSes, including Linux. She is specialized in Software and Hardware interfacing of various embedded memory technologies and Filesystems. She is currently working on optimizing the Linux Block Layer for Android phones.

Sarsij Kaystha did his M.Tech in Software Engineering from Moti Lal Nehru National Institute of 'Technology, Allahabad in the year of 2010. He worked with Samsung R&D centers in the field of File System for the development of handheld devices. He is having more than 5 years of experience in the Linux Kernel. He is specialized in Filesystem and database for Android.