Intra-Device Transient Uninstall for Applications in Mobile Devices

Deepak Kumar Garg[1], Sunil Rathour[2], Ankit Agarwal[3], Nitesh Goyal[4]
[1,2,3,4]*Product Research &Development Department, Samsung Research Institute Noida, 201301, India*

Abstract

This paper describes a neoteric approach to save the memory consumption by applications installed by the user on mobile device. With the massive adoption of mobile equipment by the users, mobile device has become inevitable part of their life. The mobile devices are being used for a variety of business or non-business activities. With the exponential growth in mobile devices usage, the mobile applications usage by per user has significantly increased that has resulted into key issues with respect to the core device needs e.g. storage, battery and performance. To solve all these issues either user need to limit the usage of device or the user need to invariably keep extending the hardware configuration of the device. Towards such ends there is a need to develop a system which can save memory, enhance battery life and improve performance of device without restricting the user needs. For this purpose, a software based solution has been developed in this research by exploiting the existing technologies given by Google i.e. BackupManager [2]& UsageStatsManager [3] for Android OS. The proposedmechanism is called as Intra-Device Transient Uninstall. It saves the memory on mobile devices by managing the applications based on their usage and their memory consumption. It has two independent modules (1) Monitoring of Device activity. The monitoring module identifies the device usage based on any of the pre-determined factors – an event of predetermined time duration, location, operation profile of the device, storage space and time period associated with an application. (2) Zip module. It receives the application usage information from monitoring module and identifies the application data including files, databases, cache, and user data. Zip module removes this application from package manager after archiving the application and its data. The application has entered to a new state called Intra device transient uninstall. In this way, a new state of an application is introduced and it is based on the method of contextual & manual archiving of installed applications/packages on Android™ enabled mobile equipment for efficient memory management.

Keywords - *Archive, Restore, Backup Manager, Usage Stats Manager, Mobile Applications and Android.*

I. INTRODUCTION

Recent computing trends have seen a drastic increase in the market penetration of handheld personal computing platforms (further referred as mobile equipment). Although, advent of these devices has solved a number of problems for the user, but it has also given rise to a number of new challenges regarding system design and maintenance. In today's scenario, availability of low cost and fast internet connectivity on mobile equipment has enabled easy access of downloadable mobile applications to end user. It has resulted into a significantly higher number of applications being installed on current generation of mobile equipment as compared to those of previous generations, leading to increasedinternal memory consumption of mobile equipment.This has become a major concern in memory management for OEMs and other software providers.

Most of the available solutions for managing storage space in mobile devices are often cumbersome and complex to use for the general masses. Thus, there exists a need for a solution to overcome the aforementioned deficiencies.

Intra Device Transient Uninstall is based on contextual and manual archiving mechanism. It includes a monitoring module which analyses the current device usage, events (based on time & location) and device state (e.g. memory, battery etc.). This module determines the unused/sparsely used applications/packages based on the context and inputs them to zip module which includes a method to archive the given applications/packages after creating backup of its byte code and associated data.

This paper explains two modules of contextual and manual archiving of packages/applications i.e. device usage monitoring module and zip module. The techniques along with the components used have been described in detail, giving complete technical and logical insight of the methodology for each of the modules. Towards the end, results received from Samsung QA team and accuracy of the system is explained which helps evaluating the goodness of the proposed approach and obtained results.

II. PROBLEM

The low storage space in the mobile devices often affects overall performance of the device. For instance, an operating system of the device and process related thereto consumes a significant portion

of the storage space in the device. In addition a user may download numerous applications and may store user content, such as multimedia files, documents, and the like, in the storage space thereby further consuming the storage space. Thus, limited storage space is available for carrying the tasks and operations related to the device. This increases the load on the processor and significantly impacts device performance. For instance, a delay in loading applications and executing new tasks is observed, resulting in sluggish and delayed performance, heating, and high battery drain in the device. Further, the limited storage space causes user to uninstall applications and/or delete/move user content from time to time to make space for new applications and new user content.

As mobile operating systemshave moved towards more complex and evolved user experience, ithas led to increased processing and on-board memory requirements.Thisputs mobile equipment manufacturers (OEMs)in a challenging position to provide hardware equipped with higher configuration at a competitive rate in the market. In order to fulfil the above laid requirements, the mobile storage capacity has become one of the most criticalhardware parameter for OEMs.As OEMs try to cope up with the situation by increasing the device configuration, it has been observed that in average user scenario, it doesn't match up to the market requirements, because, frequent software updates regularly outpace the OEM hardware enhancements.

As the number of applications per device has increased, average use time per application has gone down. The general application usage pattern sees a small set of applications being a frequent interaction target, leaving the restto be launched veryrarely (usually the interaction frequency drops down to once a week or month and so on). Thissecondset of applications consumes device memory, battery, mobile data and other resources via background processes even when they serve no direct purpose to the user within that timeframe.

To manage this infrequently accessedapplication set,the Android™ Operating system provides various functionalities, viz. Disable, Clear Data and Uninstall. Disable operation disables all the application/package components, removes user data and uninstalls application/package updates. Clear Data operation clears all user data and application/package gets reset to its default settings. Uninstall operation uninstalls the application/package from device and all associated data and bytecode gets removed from the device.

All the abovelisted options,cause loss of user data and settings which brings us to our problem statement: "To find a memory management technique which retains application and its associated data while still reducingmemory consumption, thus, enablingmore efficient allocation of system resources."

III. CONCEPT

Intra-Device Transient Uninstall, a new state (as shown in Fig. 1) for the applications in device is proposed, which offers memory saving on the device by managing the applications based on contextual and manual archiving of packages without losing user data.

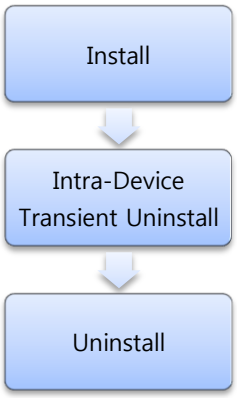

Fig. 1. The proposed state flow for an application

The proposed system as shown in Fig 2 has two modules: monitoring the device usage, zip module. The monitoring unit identifies the applications based on the device usage data and on the pre-determined factors. Thereafter, it inputs the applications to zip module which archives the application and its data and converts them to an executable archive file and a user data file.

IV. METHODOLOGY - DEVICE USAGE MONITORING MODULE

This part mainly focuses on real implementation of the proposed system. The system has been divided into two separate modules, device usage monitoring module and zip module. This implementation is carried out in controlled environment keeping track of all the variables and possibilities in real scenarios.

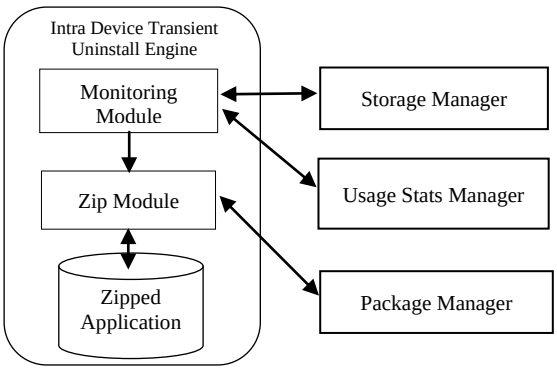

Fig. 2. Intra-Device Transient Uninstall

The monitoring module identifies the device usage based on any of the predetermined factors – an event of predetermined time& predetermined duration, location, operation profile of the device, storage space, battery status and time period associated with an application.

A. Implementation Approach

Various use cases are figured out to implement this module. As an example, for a sporting event, the application(s) least accessed during the last occurrence of the sporting event are identified. In another example, the applications least accessed based on a location of the device are identified. For instance, when a user is at his workplace, the applications least accessed by the user (social media applications) are identified. In yet another example, the applications are identified based on a time when the application was last run. For instance, applications that were run more than two weeks ago may be identified. In another example, the applications requiring maximum storage space may be identified based on the storage space requirement corresponding to the application.

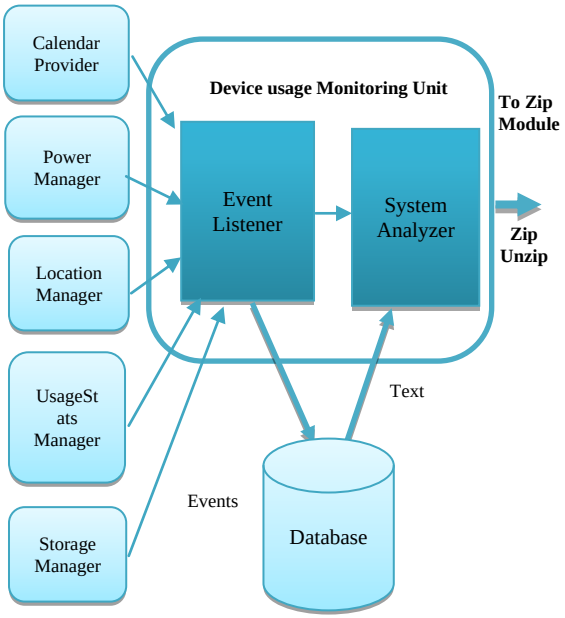

Fig 3. Architectural model of device usage Monitoring Unit

The architectural model of this module is represented in Fig. 3. The Event Listener shall get the events from Android system components such as Calendar Provider [4], Power Manager, Location Manager, Usage Stats Manager [3] and Storage Manager Etc. by listening to their broadcasts/callbacks, store the data to the database and notify System Analyzer about it. The system analyzer is solely responsible for making a decision based on the events data retrieved from database to send a request containing an action ZIP/UNZIP with the packages list to Zip module.

B. System Components

This module was build using the following fine grainedcomponents. Each component along with its purpose is described below.

1. Event Listener

In order to analyze the system usage,this system needs to retrieve the data corresponding to the events associated. This is done by Event Listener by registering to the system components like Power Manager, Calendar Provider [4], Usage stats Manager [3], Location Manager and Storage Manager Etc. and receive the broadcasts/callbacks and save it to the database. On saving the event data to the database, it sends a signal to the System Analyzer component to perform the analysis on data.

2. System Analyzer

This component retrieves the databy performing pre-defined queries on the events database. It finds out the relation between the different datasets retrieved and makes a decision whether to request or not for an action (ZIP/UNZIP) on certain packages to Zip module.

V. METHODOLOGY - ZIP MODULE

This part address the implementation techniques used tobuild the Zip module. Zipping/Unzipping the packages is done using Android system components Package Manager and Backup Manager [2]. The implementation details of the modules are as under.

A. Implementation Approach

This module receives the request containing packagelist along with the action(i. e. ZIP, UNZIP) from monitoring unit. After receiving the package list, this module identifies all the package information available on the device, creates a backup request [1] that includes Boolean variables for various kind of package files and sends it to the Backup Manager. After receiving the callback of success or failure, it updates the information in the database. If it has received the success, it will uninstall the application and notify it to the Package Manager. If it has received the failure, it will simply notify the user about the failure.

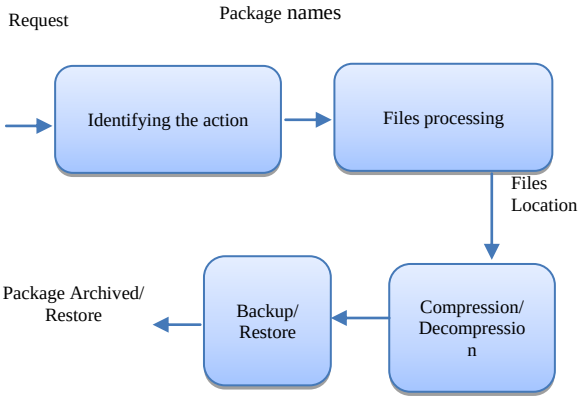

Fig. 4 Zip Module workflow

The architectural model of this module is shown in Fig. 5. The Zip module receives package names and inputs it to Package Scanner module. Package scanner sends the request for zip/unzip to Zip Processor/Unzip processor to complete the respective action. Both of these processor units makes use of Backup Manager

services-Archive Service, Restore Service to either archive or restore the requested package.

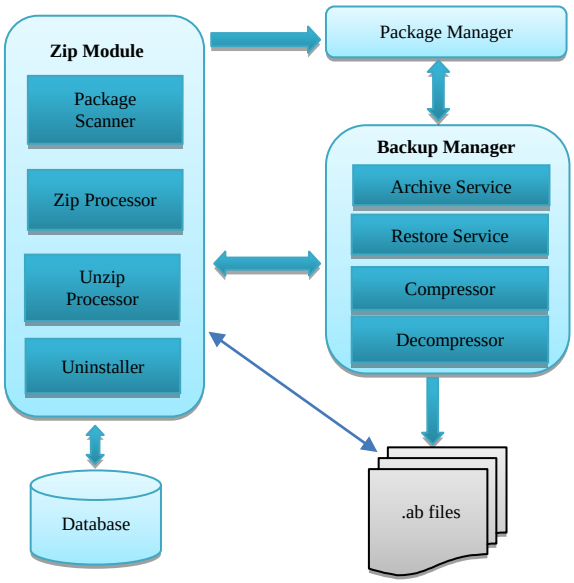

Fig. 5 Architectural model of Zip Module

An installed Android application has well defined directory structure that maintains a modular segregation of byte code and associated data. While archiving an application, the performance critical application components are identified according to rules listed in Table 1.The components listed for archivingare compressed along with their relative path information and savedintoa single backup fileas.ab (i.e. android backup) file type. While performingRestore operation, all these files are restored to their original locations.

Table 1. Android Application files on the Device

File Type	Description
Manifest file	Application manifest
Meta data file	Application meta data
Apk file	Application byte code
Cache file	Internal cached data
Database files	SQLite database file
Shared preferences	User settings file
Asset files	Asset files required by application
External cache files	External cache data
External obb files	Application expansion files

B. System Components

This module was built upon the interaction between newly defined components and modified existing Android OS components to do the required task. Each component along with its purpose is described below.

1. Package Scanner

This component identifies theperformance critical components of the requested package as described in Table 1 and declares the rules to include or not the components based on the application usage and device state like available storage, battery level etc.

2. Zip Processor

It processes the zip requests of multiplepackages by concurrent task scheduling. To zip each package, it sends a request along with package name and archive rule set to Archive service of Backup Manager [1], [2]. After archive completion, it receives a response from Archive service and it sends a request to uninstaller to uninstall the package and upon receiving a success result, it updates the data to database.

3. Archive Service

This component is a part of Backup Manager [2] and is modified to handle the archiving of packages based on the rule set sent by Zip Processor. It requests Compressor to compress the said components of package along with their relative paths and then saves them into a single archive file in .ab format and sends a success response to zip processor.

4. Unzip Processor

It processes the unzip requestofmultiple packages by concurrent task scheduling. It sends a request to Restore Service of Backup Manager [1], [2] for unzipping the requested package. Upon receiving a success result, it updates the data in database.

5. Restore Service

This service is modified to restore ofthe package components from .ab file, install the application and save the files to their respective locations in the system.It uses Decompressor module to retrieve the data from .ab file.

VI. RESULTS

We have observed that the compression ratio for each applicationvaries depending on its components (byte code and data). However, after testing with an experimental set of more than 100 applications installed across a variety of Android™ enabled equipment, it was found that proposed method of application compression and archiving results in an average 40% reduction in cumulative memory usage.

As per market reports, free memory available in mid end or inferior devices (8 GB or 16 GB storage) remains critically low. For example, experimental data collected with Galaxy J2 2016 model (containing only preloaded software with latest updates)shows that only 0.94 GB of free memory is available out of 8GB total device storage.

The applicationdeveloped for Samsung Galaxy J series implementsIntra Device Transient Uninstall mechanism. As shown in Fig. 6, the applications, which are in archived state, are displayed in Zipped apps folder with greyed icons.

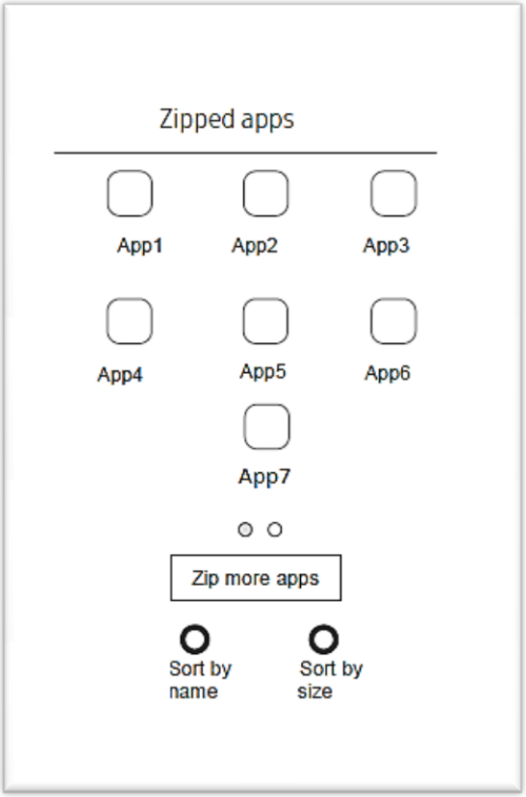

Fig. 6. Application Screen – Zipped Apps

Tests performed by Samsung QA indicate that this implementationresults in a cumulative memory usage reduction of more than 60% while testing with 5 applications* as listed in Table 2.

Table 2. Test results by Samsung QA Team

	Before Archiving	**After Archiving**
Total Memory on device	8 GB	8 GB
FreeMemory on device	1.48 GB	1.66 GB
Total Application* size	294.7 MB	113.2 MB
Memory Saved		181.5 MB (61.6 %)

*Applications installed: App1 [80.7 MB], App2 [43.3 MB], App3 [47.7 MB], App4 [101.1 MB], App5 [21.9 MB].

VI. CONCLUSION

This methodology has significant impact on devices where available memory has reached critically low level due to large number of installed applications. User can archiveinfrequently accessed applications to release a fraction of occupied memory. The same mechanism can be replicated in all the mobile operating systems.

REFERENCES

[1] WojtekKaliciński. Auto Backup for Apps made simple. https://android-developers.googleblog.com/2015/07/auto-backup-for-apps-made-simple.html (2015).

[2] Backup Manager. https://developer.android.com/reference/android/app/backup/BackupManager.html

[3] Usage Stats Manager. https://developer.android.com/reference/android/app/usage/UsageStatsManager

[4] Calendar Provider. https://developer.android.com/guides/topics/providers/calendar-provider