American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-6, Issue-1, pp-245-248
www.ajer.org
Research Paper Open Access

Effective Memory Management for Mobile operating Systems

G.C.A.L. Aponso
Faculty of Graduate Studies and Research,Sri Lanka Institute of Information Technology, Colombo 03, Sri Lanka.

ABSTRACT: As technology is progress day by day, a person who is not using a mobile phone is hardly discovered nowadays because the mobile phone is a device which helps people to ease their tasks. As mobile phone users, much concern about the high quality and much user experience in mobile applications, the mobile application needs more memory. So it has to keep several processes in memory to improve CPU Utilization and the speed of its response to users. So memory management is a key issue for mobile operating systems (OS). Mainly its memory is limited and not expandable, and efficient memory management is the only solution for these problems.
Mostly when the memory is not enough to run the applications, mobile phones get stuck. So mobile operating systems use different techniques to manage memory in a proper way. In most of these techniques involved with terminating of applications forcefully when the device runs in low memory. But it does not consider whether it is a more usable application or an application that takes more time to launch. It is the most common issue in the OSs.There are many types of studies, various applications and approaches have been proposed in achieving the optimized use of memory in the mobile operating system. Still, there is no complete solution for this issue**Index-Terms:**launch, low memory, memory management, mobile operating systems, several processes

I. INTRODUCTION

New technologies are introduced in every day. Most of them are based on mobile phones. So people move to use smartphones more and more. A person who is not using a smart mobile phone is hardly discovered. There are many different mobile OSs which used by smartphones.

Most of the app users much concerned about user experience and high-quality of theapp, it takes the app developer to develop a high memory consuming apps. Especially users tend to use many apps repeatable. So the main problem inmobile phones is memory management. Since most of the mobile phone's memory is a limited resource, it is a must to manage memory well.As a cause of low memory, many phones are stuck time to time. Mostly the phone should be restart in a situation like that. This is a problem that should be solved in operating system level because if the memory management is properly done, memory problems do not rise.Themobile device manufacturers try to fulfill the demand of high memory by increasing RAM capacity. Today there are phones which are having more than 4GB RAM. The high usage of different apps by the users, but increasing RAM is not a sustainable solution for memory management.

When an app is launched there is a time which it needs to load. If the user uses the same app, again and again, launching time will cost for each time the user launch the app and it will not be in the state that the user stopped. So if the app stored in memory when it relaunched it will load from thememory with smaller time compared to launch time and it will restore to astate where the user stopped. This technique is used in most of modern OSs. But if the remaining memory became low, suddenly some of the apps which are in the memory is forcefully terminated. Before terminating the apps, it does not check whether this is the most usable, launching delay high or other reasons.All the platforms implement the similar app life cycle and memory reclamation scheme. Android is one of major OS which is using in smartphones. It uses several techniques to manage memory in a proper way. The Zygote process is one of key memory management in Android. The Zygote's main job is to launch the apps. This maps commonly used libraries into processors address space[1].Activity Manager Service(AMS) executes at framework layer of Android. It is responsible for receiving and responding to user requests in an appropriate way. Starting and killing processors is the main function which is doing by this[2].

Android keeps the apps which were launched in memory. When the free memory gets below a threshold, memory reclamation method calls Low Memory killer(LMK), choose an app to be terminated. The selection is done according to Least recent used (LRU) page. That means the app used. In this technique, if the chosen app is large and it's often used it can have a negative impact on user experience because to relaunch it

will take time[3],[4] and [5].Another technique which used is Out-of-Memory Killer(OOMK). It is done according to priority. As soon as memory get low it terminates apps with lower priority. But the operations of OOMK seriously degrades the execution speed of new app due to trashing[4].Mainly reclamation happened on an on-demand basis, a number of apps need to be killed to secure a large amount of memory. The apps are killed one after another. So to free particular amount of memory, it can kill more apps due to on-demand basis.

In current reclamation, the victim app selection policy is not completed one and has many main aspects to be optimizedin terms of user experience. These are some of the major issues which are in current app reclamation model:

1. It prefers the apps with a large memory footprint to those with small memory footprint. This is done because if the apps which have large footprint killed, users may need to wait for a long time.
2. It does not consider app launch time. Even the apps which have same memory footprint can exhibit different launch times.
3. Reclamation happens on an on-demand basis. So it can mislead the decision to a suboptimal sequence of app termination and cannot reclaim memory in a simultaneous way.

By considering these issues in current memoryreclamation caused in a negative impact on user experience. To overcome this issue many researchers have proposed and came with some solutions.

II. REVIEW ON SMARTLMK

This is a memory reclamation scheme for improving user-perceived app launch time. This was introduced by Sang-Hoon Kim, Jinkyu Jeong, Jin-Soo Kim and Seungryoul Maeng. Mostly mobile users use a set of app repeatedly. The LMK terminates app when memory gets lower than the threshold value and the last state of execution is stored in a persistent memory. Thus when the app is relaunching, it restored the previous state. Nowadays most of the apps are large and more complicated, these apps cannot be restored. Hence the users have to wait a significant amount of time until it is relaunched. And also in selecting an app to terminate, done on-demand basis. Since it needs time to launch, this affects user experience because any user does not like to wait and especially they want the state where they have stopped to continue the work. So S. Kim and et.al proposed a novel memory schema called SmartLMK which keeps app launch time, app usage statistics and app characteristics. Using the data SmartLMK estimates a temporal penalty. Based on the penalty, it picks the apps to terminate.The Knapsack Solver had been developed using dynamic programming. It also needs working space to solve the problem. Therefore this is running in low memory, allocating memory to the app is difficult. As a solution, the Knapsack Solver is pre-allocated during system boot-up.

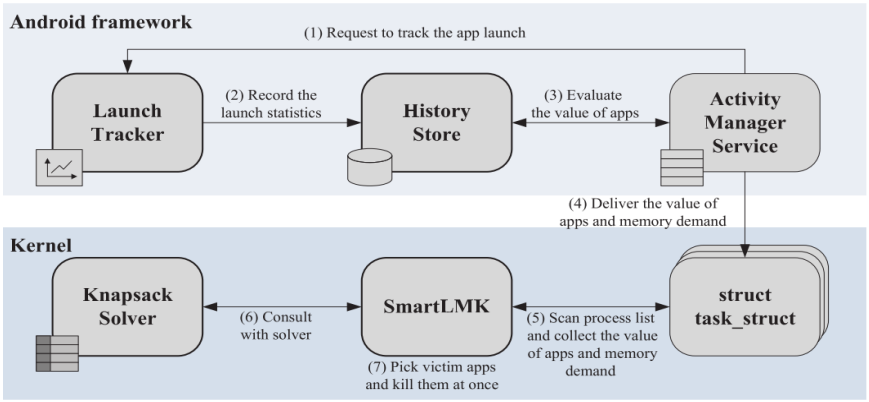

Figure 1-The overall Architecture of SmartLMK

Especially they have addressed the app launch time problem. They have used a technique which is calculated it by multiplying the influence of app's termination and the likelihood of being used.To represent memory footprint also they have come up with an own model. All the apps are forked from Zygote. So the memory that is not shared with Zygote is not shared with another processor as well. So the very amount of memory can be reclaimed by killing the app.As a prototype, the researchers had built and tested on Google Galaxy Nexus smartphone. There four modules in the system which are Lunch Tracker, History Store, SmartLMK and Knapsack Solver. Figure 1shows the proposed architecture.

Once an app launched AMS setup running environment and notify the Launch Tracker, to start tracking. The Launch Tracker starts a tracking session and monitors the system resource usage. After the app completely launched the Launch Tracker ends the tracking session and memory footprint in History Store. The History Store provides data that used to estimate the value and memory demand of apps.

When the free memory gets lower than the threshold value, SmartLMK collects all the data which stored in the

History Store to select an app and consults the Knapsack Solver to get optimal victim app. The Knapsack Solver gives the optimal victim app by solving the 0/1 Knapsack based equation which customized by the researchers (Equation 1)

$$\sum_{i=1}^{n} Value\ (\alpha_i).x_i \leq \sum_{i=1}^{n} Memory(\alpha_i) - Demand$$

Equation 1-Customized Knapsack Equation

It has improved app launch time by 13.2%. The idea is good up to certain level because of user's use apps in different according to their routines and freeness [6]. This is only introduced to android based systems.

In this, the impact to the user experience is minimized. According to the research, the researchers assumed that the usage of apps will be same to all the days and at any time. This research will be more succeed if the app usage is taken according to day by day usage and predict the usage of every app by historical data and especially it should consider the app's used time also [5].

Mainly it presents the results from memory management without memory pressure because researchers had assumed that the pattern will be same.

III. LITERATURE REVIEW

As discussed earlier mobile memory management is a most imperative topic that should consider. So this had been considered many different standpoints which provide a number of different solutions in each case. Even though many researchers have finished the massive effort, however, they were not acceptable as a complete solution for mobile memory management.

There have been several discussion and analysis on user's usage patterns of the mobile app[7], [8], [9] and [10]. D. Ferreira and et.al had shown that many users' usage patterns are varying with the place, time and day. They have chosena number of participants from different age group and analyzed their apps usages[6].

M. Linares-Vásquez and et.al have pointed the most of the mobile apps which developed reasonably, are same as equivalent to desktop applications.They have found many different usage patterns of apps of the same user in different eras. So they have highly suggested to developers to consider usage patterns of the app and it can lead to a better energy-aware solution [11].

Y. Kim and et.al have built a novel memory system based on DRAM-NVM hybrid memory architecture. This takes explicit account of the apps usage patterns to identify suitable swap candidates. Mainly this intelligently manages page migration between main memory and swap memory, to provide better user experience with respect to apps launch time. This was evaluated in Android smartphone using user app usage logs. The experimental results had shown that proposed technique achieved 32% faster launch time for mobile apps. The best thing they have mentioned is, they believed that accurate app launching prediction could help on resource management in mobile environments [12].

W. Soong and et.al have developed novel app usage models that capture user's app usage patterns. It is a user-specific personalizedoptimization module. They have implemented an app-launching experience optimization techniques which minimize user-perceived delays, extra energy consumption, and state loss when a user launches apps. This proposed system was tested on Nexus S. The experimental results showed that it avoids 78.4% of unnecessary restarts of apps [13].

K. Vimal and et.al have presented a design of a new memory management scheme for an android operating system that takes users usage patterns to decide the apps has to be killed from the main memory dynamically set the background process cache limit based on hit rate and a number of apps of user's interest. They have found by using thesame cluster of apps will definitely improve the overall experience of asmartphone. To identified the clusters app usage pattern is important[2].

S. Kim and et.al have implemented an approach that the apps are group by the same lifetime and stored them in memory continuous regions. This is tested on Nexus 10 tablet with Android. Mainly physical memory is divided into fixed-size regions and a number of regions that is to allocate for a process are depend on memory demand of the process. Unmovable files such as kernel stack, page global directory (PGD) and slab pages grouped into fixed memory and the remaining is divided into regions. In their virtual machine pages, File and buffer pages and GPU buffer stored in the Region-based allocator. The key idea of this is group memory pages with same de-allocation time in contiguous regions [14].

G. Lim and et.al have presented memory partitioning technique that resolves deterioration of existing techniques Low Memory Killer (LMK) and Out-of-Memory Killer (OOMK). This technique provides completely isolated virtual memory node at the operating system level and low performance of trusted apps induced by LMK/OOMK. This was tested in the virtual mobile environment [4].Y. Chung and et.al haveproposed an innovative app management strategies that terminate "unbeneficial" background apps. They have examined two different prediction methods which are Poisson and context-aware prediction to calculate

launch time. This strategy was evaluated through trace-driven simulation and a real experiment. As their experimental results, the app launch time can be reduced by 15% [15].

IV. CONCLUSION

There are some problems of current app reclamation model. It only considering the size of memory footprint and it does not consider launching time of the app. These two are the major issues in current app reclamation model.This paper has evaluated some research which proposed new mobile memory management techniques. There are some problems in memory management scheme which used in most of mobile OSs. The selection of victim apps to terminate to gain memory when the device runs in low memory has a negative impact on user experience.

Many researchers had concerned this issue and came up with different approaches. After considering these methodologies experimental results, the methodologies which are using user's apps usage has more success rates. To solve this problem, the victim selection has to do according to user's usage of that app.

REFERENCES

[1]. H. Kim, D. Manatunga, and G. Park, "Accelerating Application Start-up with Nonvolatile Memory in Android Systems," pp. 15–25, 2015.

[2]. K. Vimal, "A Memory Management Scheme for Enhancing Performance of Applications on Android," no. December, pp. 162–166, 2015.

[3]. K. Zhong, X. Zhu, T. Wang, D. Zhang, X. Luo, D. Liu, W. Liu, and E. H.-M. Sha, "DR. Swap," Proc. 2014 Int. Symp. Low power Electron. Des. - ISLPED '14, pp. 81–86, 2014.

[4]. G. Lim, S. Member, C. Min, and Y. I. Eom, "Virtual Memory Partitioning for Enhancing Application Performance in Mobile Platforms," IEEE Trans. Consum. Electron., vol. 59, no. 4, pp. 786–794, 2013.

[5]. S. Kim, J. JEONG, J.-S. KIM, and S. MAENG, "SmartLMK : A Memory Reclamation Scheme for Improving User-Perceived App Launch Time," vol. 15, no. 3, 2016.

[6]. D. Ferreira, J. Gonçalves, V. Kostakos, L. Barkhuus, and A. K. Dey, "Contextual experience sampling of mobile application micro-usage," 16th Int. Conf. Human-computer Interact. with Mob. devices Serv., no. SEPTEMBER, p. to appear, 2014.

[7]. "Multitasking the Android Way," 2010. [Online]. Available: http://android-developers.blogspot.com/2010/04/multitasking-android-way.html. [Accessed: 24-Sep-2016].

[8]. H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in Smartphone Usage," Proc. 8th Annu. Int. Conf. Mob. Syst. Appl. Serv. (MobiSys '10), pp. 179–194, 2010.

[9]. A. Rahmati, C. Shepard, C. Tossell, M. Dong, Z. Wang, L. Zhong, and P. Kortum, "Tales of 34 iPhone Users: How they change and why they are different," Analysis, pp. 1–12, 2011.

[10]. A. Rahmati and L. Zhong, "A longitudinal study of non-voice mobile phone usage by teens from an underserved urban community," Control, vol. 1, no. 1, pp. 1–10, 2009.

[11]. M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. Di Penta, and D. Poshyvanyk, "Mining energy-greedy API usage patterns in Android apps: an empirical study," Proc. 11th Work. Conf. Min. Softw. Repos. - MSR 2014, pp. 2–11, 2014.

[12]. Y. Kim, M. Imani, S. Patil, and T. S. Rosing, "CAUSE: Critical Application Usage-Aware Memory System Using Non-volatile Memory for Mobile Devices," Proc. IEEE/ACM Int. Conf. Comput. Des., pp. 690–696, 2015.

[13]. W. Song, Y. Kim, H. Kim, J. Lim, and J. Kim, "Personalizedoptimization for android smartphones," ACM Trans. Embed. Comput. Syst., vol. 13, no. 2, pp. 1–25, 2014.

[14]. S. Kim and J. Kim, "Controlling Physical Memory Fragmentation in Mobile Systems," Ismm 2015, no. Vm, pp. 1–14, 2015.

[15]. Y. F. Chung, Y. T. Lo, and C. T. King, "Enhancing user experiences by exploiting energy and launch delay tradeoff of mobile multimedia applications (Extended abstract)," 2012 IEEE 10th Symp. Embed. Syst. Real-Time Multimedia, ESTIMedia 2012, vol. 12, no. 1, p. 86, 2012.