ISSN: 0067-2904

Factors Affecting Application Launch Time with Android OS

Khalid Sabah Noori*, Assmaa A. Fahad
Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq

Received: 11/12/2019 Accepted: 15/3/2020

Abstract

Android OS is developing very fast, and because of being an open source OS, it is vulnerable to many problems that are manifested to users directly or indirectly. Poor application launch time is one of these problems. In this paper, a set of sixteen experiments is established to distinguish the factors that have the most evident effects on application launch time in Android mobiles. These factors are application, launch and kill, events, and storage. Mann Kendall (MK) test, one way analysis of variance (ANOVA), and Design of Experiment (DOE) are used to prove the influence of factors statistically. As a result of the experiments, the application factor, especially the third party applications level, has the most prominent effects on application launch time, followed by launch and Kill and events, while storage had the least influence.

Keywords: Launch time, Mann Kendal, one way ANOVA, DOE

العوامل التي تؤثر على وقت تشغيل التطبيق مع نظام التشغيل أندرويد

خالد صباح نوري* ,أسماء عبدالله فهد
قسم علوم الحاسوب, كلية العلوم, جامعة بغداد, بغداد, العراق

الخلاصة

نظام الاندرويد يتطور بشكل سريع. كونه نظام مفتوح المصدر, هذا جعله عرضة للعديد من المشاكل التي تظهر للمستخدمين بشكل مباشر او غير مباشر. ضعف وقت الاقلاع للتطبيق واحد من هذه المشاكل. هذا البحث, مجموعة من ستة عشر تجربة أنجزت للتمييز بين العوامل الاكثر تأثيراً على وقت الاقلاع للتطبيق في هواتف الاندرويد. هذه العوامل هي: التطبيق, وقت الاقلاع و الانهاء, الاحداث, و الخزن. أختبار مان كندل, أنوفا ذات الاتجاه الواحد, و تصميم التجارب تستخدم لاثبات تأثير هذه العوامل إحصائياً. نتيجة لهذه التجارب, عامل التطبيق و خصوصا مستوى تطبيقات الشخص الثالث الأكثر تأثيرا على وقت الاقلاع للتطبيق, ثم الاقلاع و الانهاء, الاحداث, و الخزن الاقل تأثيراً.

1. Introduction

Mobile Operating Systems are OSs which are designed specifically for mobile phones, tablets, and wearables. These OSs are basically a light weight systems which require low resources in terms of power, storage space, CPU, RAM, etc. [1].

Android OS, just like other OSs, has many problems such as CPU utilization, power consumption, and RAM shortage. However, rapid response time is one of the main features the users are looking for. Poor response time, which is the representation of Software Aging (SA), is one of these problems. SA

*Email: khalidsn82@yahoo.com

detection can be performed through many approaches, such as using machine learning algorithms [2, 3] or investigating OS metrics.

Poor application launch time can be perceived by users directly or indirectly. Direct perception is represented when the user runs an application and takes a long time to launch. This is caused by factors affecting the launch time of application, which are application, launch and kill, events, and storage factor [4], which are on the focus of this paper. Indirect perception is represented by system resource metrics that affect launch time of application. Some of these metrics include total free and lost ram metrics related to RAM resource as well as sectors read and sectors written related to storage resources [4].

2. Related Work

Application responsiveness is considered as one of the important concerns that is perceived directly by Android users. Poor launch time of applications is a result of SA which leads the mobile to gradual degradation over time. This paper aims to investigate the factors that have an influence on application launch time through reviewing the related recent researches.

Reference [5] suggested an approach to investigate the existence of SA indicators in Android applications. According to its cumulative characteristic, SA occurs in systems that are running for a long time period. A methodology was proposed that specifies memory leaking as an aging indicator in Android applications. Memory leaking happens when a process allocates memory during its execution and carries on allocating more memory in later executions without deallocating the previously used. The methodology was as follows; the monitoring strategy was applied to collect measures of resource consumption (memory utilization) of the mobile device. The workload generator was used to stress test applications in the development stage. A shell script was created to automatize the workload generation and was set to run at one second intervals. The data was collected every ten seconds through executing Android Debug Bridge (ADB) shell script. The stress test included an initial test which was performed for a short time to investigate any aging effect. If aging existed, a longer new experiment was made. A testbed (desktop machine and mobile device) was setup to test the effectiveness of the approach. 'Foursquare' Android application, which is a location-based social networking website for mobile devices, was chosen for the experiment. Two experiments were made to monitor the resource usage of 'adb' and 'top' processes. The results showed no significant increase in resident memory and virtual memory utilization, which implies that the monitoring strategy did not interfere in Android OS behaviour. The results affirmed the efficiency of the methodology and the presence of SA in 'Foursquare' application running on the Android OS.

Reference [4] considered the problem of SA in Android mobile OS. An experimental methodology was proposed that uses statistical methods to distinguish factors in the experiment plan that have effects on application Launch time. The experimental plan consisted of a combination of factors at each level, which are application set, device, workload and kill frequency, workload events configuration, and storage space usage, that are used with resource utilization metrics (memory and storage) with system operations (garbage collection and tasks) in relation with SA. As a result of experiments, it was recommended that Android software rejuvenation should adopt a measurement-based approach to be familiar with the workload conditions.

Reference [6] introduced an experimental study about SA manifestation in Android OS. These aging-related bugs are shown up through injecting memory leaks into different heap areas (Dalvik heap and Native heap) with processes having different priorities (cached, persistent). The results were analyzed from two viewpoints; user viewpoint and system viewpoint. From user viewpoint, the experiments showed that injecting memory leak in Dalvik heap, in case of not clearing recent task, affects user perception. There was an increasing trend of response time as the time going through and a failure of the application after running for a long time period. Also, the experiments showed that injecting memory leak in Native heap in case of persistent process affects user perception. An increasing of the response time for user's actions after running for long time period was recorded, and this phenomenon continues for a long time. From system's viewpoint, the experiments on memory information related to the whole system showed that injecting memory leak in Dalvik heap, in case of not clearing recent task, leads to a decreasing trend of the Realfree indicator. This can be utilized to indicate the existence of SA but not the failure of an application. Also, the experiments on memory information related to the whole system showed that injecting memory leak in Native heap, in case of clearing recent task, leads the Realfree and Memfree to exhibit a decreasing trend, from which the

Android OS can recover. Also, the results suggested that the swap space indicator cannot be used for SA in Android. As a result, the experiments on specific memory information related to an Android process (application process) showed that, when injecting memory leak in Dalvik heap in case of not clearing recent task, the allocated Dalvik Heap size still exhibited an upward trend according to the leak. However, the application crash could happen before the allocated Dalvik Heap memory of a process raises to ultimate size. Also, the utilization percentage of Dalvik Heap showed an upward trend according to the leak, which takes values higher than those in normal conditions.

Reference [7] investigated the SA indicators prediction concentrating on system's free memory, which represents system-level, and application's heap memory, which represents application-level. Long short-term memory neural network (LSTM NN) was utilized as a prediction method in comparison with the traditional prediction methods such as linear regression, Autoregressive Integrated Moving Average (ARIMA), Holt-Winter, and Multilayer Perceptron (MLP), as well as the traditional metrics that included Mean Absolute Percentage Error (MAPE)\ Mean Squared Error (MSE) and the proposed metrics that included trend accuracy (TA), fluctuation accuracy (FA), and small of variations accuracy (SVM) of aging indicators. In the experiments setup, the ADB was used for the communication between the computer and the mobile. To stress the Android OS, the Monkey tool and UI automator (testing framework) were applied to simulate user's actions. The data collected two types of aging indicators, namely the system-level aging indicator, which refers to system free memory from '/proc' virtual file system, and application-level aging indicator, which refers to utilized heap memory of application by using 'dumpty's' tool. The results under traditional and proposed metrics revealed that LSTM has an outstanding performance compared with other prediction methods.

3. Factors Testing Tools

Factor tools that are used in the experiments to collect application launch time are Monkey tool [8], Logcat tool [9], and ADB tool [10].

4. Factors Statistical Methods

Several statistical methods can be used to analyse the application launch times that are collected over the experiment time. Some of statistical methods that are used in this paper are Mann Kendall (MK) [11] test, one way analysis of variance (ANOVA) [12], and Design of Experiment (DOE) [13, 14].

5. The Experiment Designed Module

The experimental module of the paper designed to distinguish factors is partitioned into subsections: the experiment setup, the experiment design, and the experiment factors and levels.

A. The Experiment Setup: the following represents a complete view of the experiment platform which consists of several parts:

•The experiment testbed: Samsung Note3 mobile equipped with three giga bytes of RAM and thirty two giga bytes internal storage is used for conducting experiments. A 5.0 Lollipop OS is installed on the phone. For generating and injecting events into the mobile, A PC computer with eight giga bytes RAM and five hundred giga bytes hard disk is used to collect the desired data. The OS installed on the PC is 64-bit Ubuntu 18.04.1 LTS (Long Term Support).

•Experiment user events generator: Monkey tool is used to generate user events such as touch, motion, and trackball, with 10,000 events are injected into the mobile using five hundred milli seconds as a delay between each group of events.

•Experiment test applications: In the conducted experiments, two sets of applications are used: third party applications, and system applications.

•Experiment Data collection: sixteen experiments are conducted with different combinations of factors at each level. Each experiment lasts for one hour. During each experiment, five applications (third party applications or system applications) are launched and killed periodically according to launch and kill factor. A bash script is developed in order to launch and kill applications, generate events, collect data, and save the collected data to files in order to analyse them later using statistical methods. The collected data is made of applications' LTs every five and sixty seconds.

B.The Experiment Design

The experiment design is passing through many steps. Applications' LTs are collected every five and sixty seconds. The MK test is applied to the median value of the collected LTs of applications at each time period to check for upward or downward trends. The one way ANOVA is applied to the median values of the collected LTs of applications in order to detect the most influencing factors on launch time.

C.The Experiment Factors and Levels

The tests are run under many different configurations and stress applications, which represent the factors of the experiment plan. Four factors are considered in this paper, and the experiment plan is derived by varying the combinations of levels of these factors according to DOE (Table- 1) [4]. A full factorial design of the experiments conducted on the mobile is adopted (Table- 2).

Table 1- The experiment factors and level.

Factor	Level	Description
Application	third party applications	- io.faceapp - com.google.android.applications.translate - com.newpower.apkmanager - com.infraware.office.link - org.videolan.vlc
	System applications	- com.sec.android.applicationpopupcalculator - com.sec.android.applicationclockpackage - com.samsung.helphub - com.android.mms - com.sec.android.applicationmusic
Launch & Kill	High	Kill and re-launch every five seconds
	Low	Kill and re-launch every sixty seconds
Events	Events	Monkey tool sends touch, motion, trackball, navigation, majornavigation, systemkeys, switch, anyevent, flip, and pinchzoom
	None	Monkey tools is not sending events (not used)
Storage	Normal	Default storage space
	Full	Ninety percent storage space

Table 2- The experiment plan.

Application	Launch&Kill	Events	Storage
third party applications	High	Events	Normal
System applications	High	None	Normal
System applications	High	None	Full
System applications	High	Events	Normal
System applications	Low	None	Full
third party applications	High	Events	Full
System applications	Low	Events	Full
System applications	Low	None	Normal
System applications	Low	Events	Normal
third party applications	High	None	Normal
third party applications	Low	Events	Normal
third party applications	Low	None	Normal
third party applications	High	None	Full
third party applications	Low	Events	Full
third party applications	Low	None	Full
System applications	high	Events	Full

6. Results and discussion

After collecting LTs of the workload applications every five and sixty seconds and finding the median value of the LTs at each time period across the experiment, a statistical method, which is Mann Kendal test, is used to check for trends of the LTs of the applications. The existing of trend (increasing or decreasing) means that the LTs are affected by the combinations of factors at each level across experiments. Table- 3 displays Mann Kendal test trends for applications.

The significance level that is determined in the experiments is 95%, which means α=0.05. If the P-value is less than 0.05, this means that there is a trend. The positive or negative Z-value means an increasing or decreasing trend, respectively.

Table 3- Mann Kendal test trends for applications.

Experiment No.	Z – value	P - value
1	1.6569	0.0975
2	0.9939	0.3203
3	0.6501	0.5156
4	-5.7047	1.1654e-08
5	2.8318	0.0046
6	2.4786	0.0132
7	-1.2069	0.2275
8	1.3011	0.1932
9	-3.9990	6.3621e-05
10	-5.4298	5.6405e-08
11	-3.3484	8.1276e-04
12	1.0268	0.3045
13	19.5658	0
14	-3.5051	4.6268e-04
15	0.4847	0.6279
16	0.8496	0.3955

After checking the trends, one way ANOVA statistical method is performed on LTs of applications every five and sixty seconds to distinguish the affecting factors. The results showed that the factor that has the strongest influence on LT is the application factor, based on the largest F-value it has (Table-4). The level of the application factor that has the strongest influence is the third party applications, based on the largest mean value it has (Table-5). The next most influence factors after the applications factor are launch and kill**,** events**,** and storage, respectively.

Table 4- Results of one way ANOVA (tests of between-subjects effect)

Tests of Between-Subjects Effects

Dependent Variable: launchtime

Source	Type IV Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	6239.707[a]	15	415.980	34.315	.000
Intercept	5751.460	1	5751.460	474.443	.000
applications	1688.547	1	1688.547	139.290	.000
launckandkill	305.257	1	305.257	25.181	.000
events	266.484	1	266.484	21.982	.000
storage	255.486	1	255.486	21.075	.000
applications * launckandkill	130.196	1	130.196	10.740	.001
applications * events	14.852	1	14.852	1.225	.268
applications * storage	64.687	1	64.687	5.336	.021
launckandkill * events	301.347	1	301.347	24.858	.000
launckandkill * storage	158.789	1	158.789	13.099	.000
events * storage	218.110	1	218.110	17.992	.000
applications * launckandkill * events	115.701	1	115.701	9.544	.002
applications * launckandkill * storage	182.656	1	182.656	15.068	.000
applications * events * storage	39.117	1	39.117	3.227	.073
launckandkill * events * storage	183.319	1	183.319	15.122	.000
applications * launckandkill * events * storage	221.517	1	221.517	18.273	.000
Error	56454.695	4657	12.123		
Total	73437.815	4673			
Corrected Total	62694.402	4672			

Table 5- Results of one way ANOVA (descriptive statistics)

Descriptive Statistics

Dependent Variable: launchtime

applications	launckandkill	events	storage	Mean	Std. Deviation	N
thirdpartyjapps	high	events	normal	2.38922	1.094173	46
			ful	1.86737	4.122415	487
			Total	1.91068	3.955508	533
		none	normal	2.39764	.334199	654
			ful	2.67849	.385693	720
			Total	2.54481	.388212	1374
		Total	normal	2.39577	.425984	700
			ful	2.35122	2.663761	1207
			Total	2.36757	2.134639	1907
	low	events	normal	2.49923	2.885924	60
			ful	7.76590	25.417622	60
			Total	5.13257	18.168724	120
		none	normal	2.56015	.137427	60
			ful	2.59702	.153762	60
			Total	2.57858	.146384	120
		Total	normal	2.52969	1.879897	120
			ful	5.18146	18.084812	120
			Total	3.85557	12.898485	240
	Total	events	normal	2.44281	2.123871	106
			ful	2.51437	9.399077	547
			Total	2.50276	8.643340	653
		none	normal	2.41130	.325405	714
			ful	2.67222	.373583	780
			Total	2.54752	.374683	1494
		Total	normal	2.41537	.818906	820
			ful	2.60716	6.038577	1327
			Total	2.53381	4.774490	2147
system apps	high	events	normal	.41883	.761105	669
			ful	1.40846	6.172987	199
			Total	.64572	3.053184	868
		none	normal	.62987	.045475	720
			ful	.57682	.030066	720
			Total	.60335	.046787	1440
		Total	normal	.52823	.539433	1389
			ful	.75690	2.887397	919

7. Conclusions

After conducting the sixteen experiments, it is concluded that the applications factor is the strongest affecting factor on launch time, according to its F-value (largest value). Specifically the third party applications have the strongest influence according to the mean value. The next most affecting factors on LT are launch and kill, events, and storage, respectively, according to their F-values.

According to the experimental results, the paper recommends:

- Each conducted experiment should not be limited with a specified time and should continue until the device is in an unsteady state and not reacting to user actions. This gives a lot of information which makes the statistical results more accurate.
- In design of the experiments, it is preferred to conduct the experiment in a full factorial design but at the expense of time and cost. When the number of experiments is not too large, it is better to implement them in a full factorial design (taking all aspects of the factors (levels)) instead of fractional factorial design (taking part of the experiments) to discover their influence on LT.

As a future work, we will further investigate the effects of Android mobile resources on launch time of the applications.

References

1. Wukkadada, B. Nambiar, R. and Nair, A. **2015**. "Mobile Operating System: Analysis and Comparison of Android and iOS", *International Journal of Computing and Technology*, **2**(7).
2. Ayad R. Abbas, A. And Kareem, **R. 2018**. "Age Estimation Using Support Vector Machine", *Iraqi Journal of Science*, **59**(3C): 1746-1756.
3. Inas A., Sumaya S. And Safa A. **2016**. "Using One-Class SVM with Spam Classification", *Iraqi Journal of Science*, 57(1B): 501-506.
4. Cotroneo, D., Fucci, F., Iannillo, A.K., Natella, R. and Pietrantuono, R. **2016**. "SA Analysis of the Android Mobile OS," IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), Ottawa, ON, pp. 478-489.
5. Araujo, J., Alves, V., Oliveira, D., Dias, P., Silva, B. And Maciel, P. **2013**. "An Investigative Approach to SA in Android Applications," IEEE International Conference on Systems, Man, and Cybernetics, Manchester, pp. 1229-1234.
6. Qiao, Y., Zheng, Z. and Qin, F. **2016**. "An Empirical Study of SA Manifestations in Android," IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Ottawa, ON, pp. 84-90.
7. Qiao, Y., Zheng, Z. and Fang, Y. **2018**. "An Empirical Study on SA Indicators Prediction in Android Mobile," IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Memphis, TN, pp. 271-277.
8. UI/Application Exerciser Monkey | Android Developers. **2019**. Retrieved 31 October 2019, from https://developer.android.com/studio/test/monkey
9. Logcat command-line tool | Android Developers. **2019**. Retrieved 31 October 2019, from https://developer.android.com/studio/command-line/logcat#alternativeBuffers
10. Android Debug Bridge (adb) | Android Developers. **2019**. Retrieved 31 October 2019, from https://developer.android.com/studio/command-line/adb#IntentSpec
11. Gilbert, Richard O. **1987**. "*Statistical Methods for Environmental Pollution Monitoring*", Van Nostrand Reinhold Company Inc.
12. Weiss, N. A. **2017**. "I*ntroductory statistics*", Pearson Education Limited.
13. Montgomery, D. C. **2013**, "*Design and Analysis of Experiments*", John Wiley & Sons Inc., eighth edition.
14. Wagner, J. R., Mount ,E. M. and Giles,H. F. **2013**. "*Extrusion*", Elsevier Inc.